UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

DEPARTAMENTO ACADÊMICO DE ELETRÔNICA

CURSO SUPERIOR DE TECNOLOGIA EM AUTOMAÇÃO INDUSTRIAL

ALISON CALIXTO WEBER

IAN ATHAYDES FADANELLI

MÓDULO DE AUTOMAÇÃO RESIDENCIAL CONTROLADO VIA PLATAFORMA WEB

TRABALHO DE CONCLUSÃO DE CURSO

PONTA GROSSA

2013

ALISON CALIXTO WEBER

IAN ATHAYDES FADANELLI

MÓDULO DE AUTOMAÇÃO RESIDENCIAL CONTROLADO VIA PLATAFORMA WEB

Trabalho de Conclusão de Curso apresentada como requisito parcial à obtenção do título de Tecnólogo em Automação Industrial, do Departamento Acadêmico de Eletrônica, da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. M.Sc. Frederic Conrad Janzen

PONTA GROSSA

2013

TERMO DE APROVAÇÃO

MÓDULO DE AUTOMAÇÃO RESIDENCIAL CONTROLADO VIA PLATAFORMA WEB

por

ALISON CALIXTO WEBER
IAN ATHAYDES FADANELLI

Este Trabalho de Conclusão de Curso foi apresentado em 24 de abril de 2013 como requisito parcial para a obtenção do título de Tecnólogo em Automação Industrial. Os candidatos foram arguidos pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Prof. M.Sc. Frederic Conrad Janzen
Prof. Orientador

_____
Prof. M. Eng. Murilo Oliveira Leme
Membro titular

_____
Prof. Dr. Max Mauro Dias Santos
Membro titular

Dedicamos este trabalho primeiramente a Deus, que nos deu forças para continuar e às nossas famílias, pelos momentos de ausência.

AGRADECIMENTOS

Primeiramente gostaríamos de nos desculpar a aquelas pessoas que não iremos citar aqui, pois sabemos que foram muitas estas que passaram por nós e nos ajudaram a enfrentar este caminho árduo, de muitas dificuldades e barreiras. Muitos foram os dias que deixamos de estar presentes, congratulando, para ficar estudando, pesquisando, calculando para o desenvolvimento deste trabalho. Mas acreditem, todos de alguma maneira são lembrados.

Agradecemos ao nosso orientador Prof. Frederic Conrad Jansen M.Sc, pela sabedoria, paciência e dedicação com que nos guiou nesta trajetória.

Aos nossos colegas de sala, que nos ajudaram em momentos de dificuldade, onde não sabíamos como executar tal tarefa, e àqueles que ficaram para trás do mesmo modo agradecemos e esperamos que cheguem onde hoje estamos.

Gostaria de deixar registrado também, o nosso reconhecimento às nossas famílias, pois acreditamos que sem o apoio deles seria muito difícil vencer esse desafio, pois foram eles que nos momentos em que o pensamento era desistir, nos incentivaram a permanecer.

Enfim, agradecemos a todos os que por algum motivo contribuíram para a realização desta pesquisa.

Determinação coragem e autoconfiança
são fatores decisivos para o sucesso.
Se estamos possuídos por uma inabalável
determinação conseguiremos superá-los.
Independentemente das circunstâncias,
devemos ser sempre humildes, recatados
e despidos de orgulho.

(LAMA, Dalai, 1989)

RESUMO

WEBER, Alison Calixto; FADANELLI, Ian Athaydes. **Módulo de Automação Residencial Controlado Via Plataforma Web**. 2013. 65 F. Trabalho de Conclusão de Curso (Tecnologia em Automação Industrial) – Universidade Tecnológica Federal do Paraná, 2013.

A Automação Residencial tem sido adotada em diversos lugares, como casas, centros comerciais, grandes redes de banco, empresas, os quais buscam ter maior controle e segurança sobre aquilo que possuem. Porém ainda é bastante elevado o custo de uma Automação Residencial, pelo fato de que as empresas que disponibilizam esse serviço têm seu projeto de forma proprietária. Neste contexto, este trabalho apresenta o estudo e a implementação prática de um módulo de automação baseado em *software* e *hardware* livre, dedicado a residências e controlado via Plataforma *Web*. Para tanto, foram realizadas pesquisas de projetos já existentes e como estes funcionam, além de pesquisas das ferramentas disponíveis para a o desenvolvimento, tais como softwares e elementos para a aplicação do sistema. O sistema de controle do módulo é feito utilizando a plataforma Arduino e é responsável por fazer a leitura das requisições dos usuários, autenticá-las, validar seus parâmetros e executar comandos de acordo com o que foi previamente programado. É composto também por uma placa de aquisição de dados, responsável por fazer a conversão dos sinais recebidos do campo, adequando-os ao Arduino. O sistema de supervisão é composto pela interface, onde o usuário interage com sistema através de uma aplicação Web. Este sistema é responsável por gerenciar contas de usuários, cadastrar os dispositivos conectados ao sistema de controle, gerar os alarmes do módulo, registrar as atividades realizadas em um histórico e coletar e visualizar dados dos dispositivos lidos através do sistema de controle. O desenvolvimento deste trabalho foi composto pela definição dos requisitos e pela modelagem do sistema. Por último, foi feita a implementação e foram realizados testes práticos para analisar o funcionamento do projeto. Com os resultados obtidos pôde-se concluir que o sistema de supervisão e o sistema de controle oferecem as funcionalidades necessárias para o controle residencial de forma satisfatória e com baixo custo de implantação.

Palavras-chave: Automação Residencial. Arduino. Sistema de Supervisão. Sistema de Controle. GWT.

ABSTRACT

WEBER, Alison Calixto; FADANELLI, Ian Athaydes. **Home Automation Module through a Web Platform.** . 2013. 65 F. Trabalho de Conclusão de Curso (Tecnologia em Automação Industrial) – Federal Technology University - Parana. Ponta Grossa, 2013.

Home automation has been applied in several places, houses, shopping centers and large networks of bank companies, which seek to have greater control and security over what they own. However, the home automation is still very expensive, as the companies that provide this service have their own proprietary tools. In this context, this work presents the study and the practical implementation of an home automation module controlled by a Web Platform. The study was conducted through research of existing projects and how they work, including the research of some development tools applicable to this work, like software and components for the system application. The module's control system was developed with the Arduino platform and is responsible to read the users requests, authenticate them, validate their parameters and execute commands according to what has been previously established. It also consists of a data collection board that receives the field signs and adapts them to the Arduino. The supervision system is composed by an interface, where the user interacts with the system through a web application. This system is responsible for managing user accounts, registering the devices connected to the control system, create alarms on the module, register history logs and collect and view devices' data read through the control system. The development of this work consisted of the definition of the requirements and the system modeling. Finally, it was implemented and practical tests were performed to analyze the behavior of the project. With these results it was conclude that the supervision system and the control system provide the functionality needed to control a residence satisfactorily and with low deployment cost.

Keywords: Home Automation. Arduino. Supervision System. Control System. GWT.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

A	Ampére
AJAX	*Asynchronous Javascript And XML*
CLP	Controlador Lógico Programável
CPU	*Central Processing Unity*
CRC	*Cyclic Redundancy Check*
CSS	*Cascading Style Sheets*
DAO	*Data access object*
DOM	*Document Object Model*
FCS	*Frame Check Sequence*
GWT	*Google Web Toolkit*
HTML	*HyperText Markup Language*
HTTP	*HyperText Transfer Protocol*
HTTPS	*HyperText Transfer Protocol Secure*
IDE	*Integrated Development Environment*
IEEE	*Institut of Electrical and Eletronic Enginners*
IHM	Interface Homem-Máquina
JRE	*Java Runtime Environment*
JSNI	*JavaScript Native Interface*
JSON	*JavaScript Object Notation*
LAN	*Local Area Network*
MAC	*Media Access Control*
PC	*Personal Computer*
PWM	*Pulse-Width Modulation*
RAM	*Random Access Memory*
ROM	*Read Only Memory*
RPC	*Remote Procedure Call*
RTU	*Remote Terminal Unit*
SCADA	*Supervisory Control and Data Aquisition*
SSL/TSL	*Security Sockets Layer/Transfer Layer Security*
TCP-IP	*Transmission Control Protocol - Internet Protocol*
UML	*Unified Modeling Language*

USB	*Universal Serial Bus*
Vca	Tensão de Corrente Alternada
Vcc	Tensão de Corrente Contínua
WWW	*World Wide Web*
XML	*eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

A Automação Residencial tem sido adotada em diversos lugares, como casas, centros comerciais, empresas, os quais buscam ter maior controle e segurança sobre aquilo que possuem. Pode-se dizer que há alguns anos o principal objetivo de se automatizar um local era o de proteger seus bens através de sistemas monitorados de alarme. Mas o conceito de Automação Residencial tem mudado durante esse período, onde as pessoas têm buscado cada vez mais o conforto, abrangendo esse conceito aos mais diversos ambientes de uma residência tais como:

- Controle de iluminação;
- Climatização;
- Sonorização;
- Eletroeletrônicos com ativação e desativação programada;
- Controle de entrada de pessoas.

Entretanto, ainda é bastante elevado o custo de uma Automação Residencial, pelo fato de que as empresas que disponibilizam esse serviço têm seu projeto de forma proprietária. Como grande parte dos clientes busca segurança, conforto e comodidade, e não existem muitas empresas que oferecem esses serviços com baixos custos, eles acabam não tendo opções senão ser atendido por algum prestador de serviço com preços elevados, restringindo essa parcela de clientes apenas à população rica.

A ideia de *hardware* e *software* livre deixa todo esse conceito mais aberto, para que assim, todo aquele que tendo conhecimento e desejar, pode alterar o projeto. O *hardware* de desenvolvimento de projetos eletrônicos Arduino é baseado nesse conceito, o qual já vem conquistando diversas áreas do mercado, onde empresas desenvolvem placas para assim estender as funcionalidades do *hardware*, e isso deixa o dispositivo final com o custo reduzido.

Com as mais diversas funcionalidades e vantagens de uma automatização, procura-se melhorar em todos os sentidos o dia a dia, para que assim as atividades cotidianas se tornem mais fáceis de serem executadas e, consequentemente, agilize toda e qualquer situação ocorrida. Nesse sentido, pode-se ter a tranquilidade de que

algumas coisas funcionarão automaticamente, sem a necessidade de supervisão, ou até mesmo recebendo informações sobre o sistema.

1.1 TEMA DA PESQUISA

O presente trabalho propõe o desenvolvimento de um sistema de supervisão baseado em software e *hardware* livre para controle de dispositivos aplicados a automação residencial.

1.2 DELIMITAÇÃO DO TEMA

O trabalho visa desenvolver um módulo de automação residencial baseado em tecnologias abertas, composto por um sistema de supervisão *web* e por um sistema de controle usando a plataforma Arduino, possibilitando enviar comandos e a trocar dados entre os sistemas através da tecnologia *ethernet*.

1.3 PROBLEMA

Com a análise de algumas situações reais enfrentadas, é observada a possibilidade de aplicar tecnologias existentes em um sistema de supervisão, com o intuito de reduzir a possibilidade existente de se esquecer uma janela aberta, luzes e aparelhos ligados, como exemplo, enquanto não se está presente no local e assim evitar um provável dano a bens pessoais e o desperdício de energia, e também a facilidade de poder controlar os equipamentos presentes na residência de qualquer lugar.

1.4 HIPÓTESE

Através do desenvolvimento de uma aplicação *web* com acesso a uma placa de aquisição de dados via um protocolo de comunicação, é possível criar um sistema de supervisão que ofereça funcionalidades para o controle remoto de dispositivos em uma residência, por meio de um navegador *web*.

1.5 OBJETIVOS

1.5.1 Objetivo Geral

Desenvolver um sistema de automação controlado via plataforma *web* que permita controlar remotamente dispositivos em uma residência.

1.5.2 Objetivos Específicos

- Realizar um levantamento bibliográfico.
- Especificar os requisitos dos sistemas a serem desenvolvidas.
- Criar uma interface entre os dispositivos de *hardware* e o sistema de supervisão utilizando a plataforma Arduino.
- Desenvolver uma aplicação *web* que forneça uma interface ao usuário e implemente o controle no lado do servidor.
- Escolher um sistema de gerenciamento de banco de dados para o armazenamento dos dados.
- Modelar e Desenvolver um protótipo do projeto.
- Análise e comparação dos resultados práticos e simulados.

1.6 JUSTIFICATIVA

A Automação Residencial é um segmento que tem sua aplicação direcionada às pessoas que utilizam o local onde ela é aplicada, trazendo comodidade, segurança e controle. Isso reflete no crescimento da aplicação de tecnologias voltadas para este fim.

Uma das partes fundamentais da automação é o sistema de supervisão, pois propicia ao usuário ferramentas para o gerenciamento dos dispositivos presentes, tornando possível um controle mais intuitivo do processo.

Unindo os fatores já citados com a necessidade de ter uma tecnologia de baixo custo que seja competitiva, existe a possibilidade de aplicar os conceitos de *software* e *hardware* livre, trazendo benefícios como um suporte mais amplo entre a comunidade desenvolvedora e acesso facilitado às informações da tecnologia.

1.7 MÉTODO DA PESQUISA

O projeto será desenvolvido através de pesquisa exploratória e explicativa, pois serão utilizados os conhecimentos teóricos existentes para determinar as ferramentas adequadas a serem aplicadas na metodologia.

Este trabalho está organizado em quatro capítulos. O capítulo 2 contém a revisão da literatura, apresentando conceitos básicos sobre sistemas de supervisão e sistemas embarcados. O capítulo 3 apresenta a descrição das ferramentas utilizadas e a proposta de implementação, composta pelo levantamento dos requisitos a modelagem do sistema. O capítulo 4 contém uma avaliação dos resultados da aplicação desenvolvida e o capítulo 5 apresenta as conclusões do trabalho, incluindo sugestões para trabalhos futuros.

2 REVISÃO DA LITERATURA

2.1 AUTOMAÇÃO RESIDENCIAL

A automação residencial nada mais é do que a integração de subsistemas existentes em uma residência, como por exemplo, equipamentos de áudio e vídeo, eletrodomésticos, portas e portões, janelas, todos ligados a um módulo de controle, e também podem ser visualizados em telas, sendo IHM's (Interface Homem-Máquina) no local ou de forma remota, através de dispositivos móveis.

A integração destes subsistemas é feita através de uma rede doméstica, na qual os dispositivos são ligados e estes são gerenciados por um controlador, responsável por tratar os dados que são recebidos do campo e executar as tarefas programadas. Mas antes de existirem as redes domésticas, os subsistemas exerciam suas funções de maneira independente, atuando de forma isolada, ou seja, cada dispositivo realizava sua função, obedecendo apenas ao subsistema do qual fazia parte.

Assim como na Automação Industrial, a comunicação entre os subsistemas é um fator importante, pois é através dela que os dispositivos irão transmitir seus dados. Segundo Krippendorf (1994) um modelo de comunicação trata-se de um modelo matemático, para permitir a transmissão de um conjunto de informações quantificáveis de um lugar para outro, através de um ponto A (o emissor) para um ponto B (o receptor). A informação, uma vez codificada em sinais por um emissor, seria transmitida através de um canal (a mídia) para um receptor que processaria a sua decodificação. A boa comunicação é o que vai garantir com que as tarefas gerenciadas pelo controlador, sejam executadas da forma como foram programadas, e também com que os dados obtidos do campo sejam confiáveis, para que o usuário visualize o que está acontecendo.

O termo Automação Residencial também pode ser chamado de Domótica. Este termo vem da fusão da palavra latina *domus* (casa) e da palavra robótica. Conforme Breternitz (2001), as primeiras aplicações domóticas utilizavam sensores e atuadores (dispositivos que alteravam os parâmetros em função de informações captadas pelos sensores), que numa arquitetura centralizada eram ligados a um

controlador onde estava a inteligência necessária. Quase sempre eram sistemas proprietários, pouco flexíveis, e principalmente caros.

2.2 MICROCONTROLADOR

O microcontrolador é um circuito integrado que inclui todos os componentes necessários para um sistema computacional funcionar. Por este motivo ele difere de um microprocessador, pois para que este possa ser usado outros componentes devem ser adicionados (MATIC; ANDRIC, 2000).

O microcontrolador possui uma Unidade Central de Processamento (CPU), unidades de memória ROM e RAM, barramentos, unidades de entrada e saída, temporizadores e outros periféricos opcionais, como conversores Analógico/Digital ou Digital/Analógico. A Figura 1 mostra a arquitetura genérica de um microcontrolador.

Por esse motivo, a utilização de um microcontrolador simplifica bastante a construção de um sistema embarcado, já que possui integrado todos os periféricos necessários para comunicação, armazenamento e aquisição de informações.

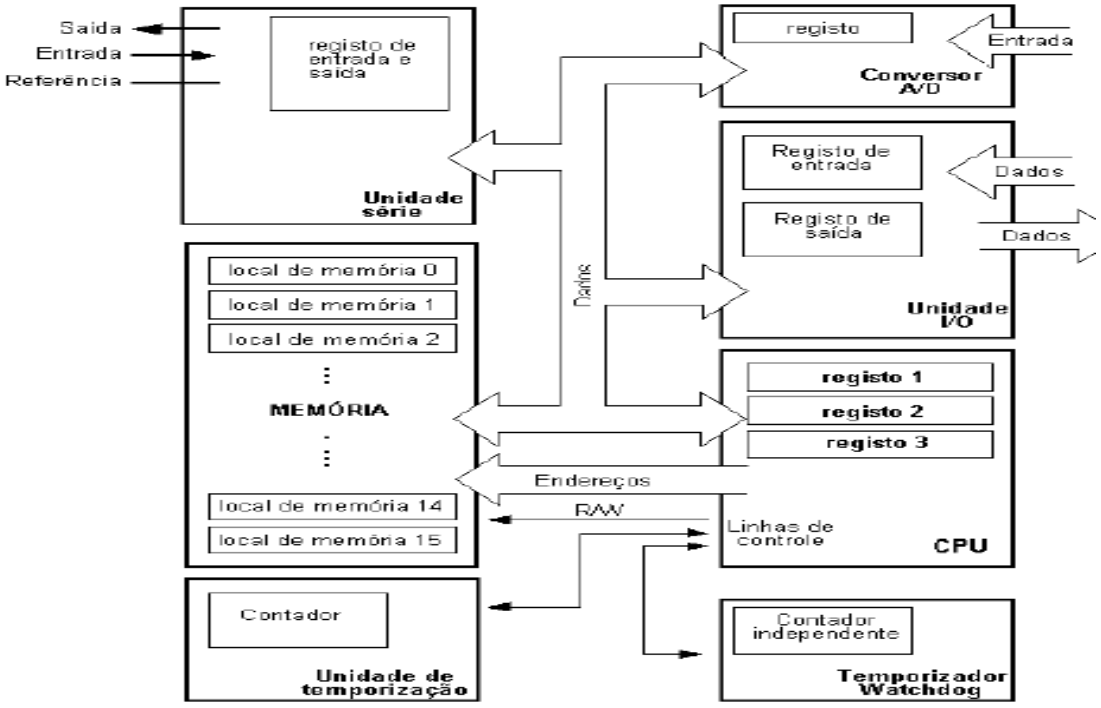


Figura 1: Esquema genérico de um microcontrolador.
Fonte: (MATIC; ANDRIC, 2000)

2.3 *HARDWARE* LIVRE E ARDUINO

Nas ultimas décadas se popularizou a ideia do *software* livre, que é definido por Hippel e Krogh (2003) como qualquer programa de computador que pode ser distribuído livremente para uso, cópia, estudo ou ainda redistribuição. Desta forma, a comunidade se opõe ao conceito de *software* fechado, onde apenas o proprietário detém o conhecimento da tecnologia. Segundo Hippel e Krogh (2003) um *software* de código aberto é um *software* que é feito de graça e disponível para todos, onde os projetos são mantidos por comunidades de desenvolvedores criadas na internet e que se voluntariam para colaborar com o desenvolvimento do *software* que eles ou suas organizações necessitam.

Com a evolução do *software* livre surgiu a proposta de herdar as suas características para o desenvolvimento de *hardware*. Desta forma criou-se o conceito de *hardware* livre. Segundo Rubow (2008) não existe uma definição totalmente aceitável e clara para o *hardware* livre, pelo fato de este ter sido criado com *software* em mente. Portanto, da mesma forma que o *software* livre, todo o projeto de *hardware* é disponibilizado livremente para a comunidade, incluindo o esquema elétrico e a placa de circuito impresso, possibilitando que o usuário crie seu próprio *hardware*, podendo alterar o circuito original se desejar.

O Arduino é um projeto de *hardware* livre desenvolvido na Itália e que atualmente é o projeto mais importante deste movimento, pelo fato de estar sendo cada vez mais utilizado. O Arduino Mega é uma plataforma de desenvolvimento que fornece uma interface entre o microcontrolador ATmega1280 da família AVR, e o desenvolvedor (MELLIS, 2008). Suas principais vantagens são a simplicidade de utilização, o que permite que seja utilizado por usuários com diversos níveis de conhecimento, e a possibilidade de ampliar suas funcionalidades facilmente através de novos módulos chamados de *shields*. Os *shields* são criados seguindo a mesma ideia do Srduino, ou seja, são fáceis de utilizar e mantém sua escalabilidade, de forma com que vários *shields* diferentes podem ser usados em conjunto num único arduino.

Figura 2: Placa Arduino Mega.
Fonte: MELLIS (2008)

A Figura 2 mostra a parte física do Arduino, que possui o circuito básico para o funcionamento do microcontrolador, algumas proteções, além de possibilitar a comunicação com o computador através do circuito integrado FTDI *Chip*[1], que faz a conversão de sinais seriais no padrão RS232 para o padrão USB (MELLIS, 2008). A Tabela 1 apresenta as principais características do Arduino Mega.

Característica	**Valor**
Microcontrolador	ATmega1280
Tensão de Operação	5V
Tensão de Entrada Recomendada	7-12V
Tensão de Entrada Limite	6-20V
Pinos Digitais E/S	54 (15 podem ser saídas PWM)
Pinos de Entrada Analógica	16
Corrente por pino digital E/S	40 mA
Corrente no pino 3,3Vcc	50 mA
Memória Flash	128 KB (4 KB usados pelo bootloader)
SRAM	8 KB
EEPROM	4 KB
Frequência do microcontrolador	16 MHz

Tabela 1: Características do Arduino Mega
Fonte: MELLIS (2008)

O Arduino também possui uma interface para o desenvolvimento do *software,* onde estão incluídas diversas ferramentas que fazem o processo de compilação e gravação do programa no microcontrolador, transparente ao desenvolvedor.

[1] http://www.ftdichip.com/

2.4 ETHERNET

A *ethernet* é uma tecnologia desenvolvida para conectar computadores em uma rede local – LAN. Sua origem se deu no início da década de 70, no Havaí, onde foi projetada e implementada por Bob Metcalfe e David Boggs como trabalho em um centro de pesquisa da Xerox (TANENBAUM, 2003).

Uma rede LAN deve ser compatível com a maior quantidade de equipamentos possível para prover uma grande flexibilidade. Com isso em mente, Metcalfe propôs que a *ethernet* fosse um padrão aberto, independente de fabricantes (SPURGEON, 2000). Devido ao fato de ser uma tecnologia de baixo custo, alta velocidade, de fácil utilização e altamente padronizada, a *ethernet* se popularizou rapidamente e atualmente é a principal tecnologia de rede de computadores locais.

Com alguns projetos implementados, verificou-se a necessidade de padronizar esse protocolo, então o *Institut of Electrical and Eletronic Enginners (IEEE)* criou o Protocolo 802.3, que normatiza a Camada Física de uma rede local. Este Protocolo fornece especificações para todos os componentes de uma rede local *ethernet* e regulamenta também a inter-conexão de componentes, o número de conexões do barramento, o número total de dispositivos de usuários e retardos de sinais que são permitidos.

2.4.1 Codificação do Sinal *Ethernet*

Os sinais transmitidos são constituídos de sinais binários, conhecidos como *bits*, os quais são constituídos de dois níveis de tensão (Vcc), chamados de nível lógico *0* e nível lógico *1*.

O método de codificação utilizado por este protocolo é o *manchester*, que proporciona ao sinal digital um método de sincronização que garante que as transições aos níveis lógicos ocorram somente durante o centro do período do *bit*. Os *bits* são normalmente colocados em grupos de oito, formando um *byte,* onde seu formato é expresso em caracteres hexadecimais, que variam de *0* a *9* e de *A* a *F.*

2.4.2 Pacotes *Ethernet*

O pacote transmitido é constituído pelo *byte* citado anteriormente, e este pacote contêm informações sobre o endereçamento, a sincronização, o protocolo, a correção de erros e também os dados a serem enviados.

Figura 3: Representação do Pacote *Ethernet*
Fonte: Autoria Própria

Destination Address – Endereço para onde o pacote será enviado. Possui um tamanho de 48 *bits* (6 *bytes*).

Source Address – Endereço de onde o pacote está sendo enviado. Possui um tamanho de 48 *bits* (6 *bytes*).

Type – Indica o protocolo e nível 3 que está sendo transportado.

Data – é onde contém os dados que estão sendo enviados juntamente com algumas informações de controle. Possui um tamanho de 46 *bytes*, e se caso os dados enviados sejam menores que isso, utiliza-se um padrão especial e *bit* chamado PAD para completar o número de *bytes*.

Teste de Redundância Cíclica (CRC) – pode ser chamado de Teste de *Frames* (FCS), garante que os dados recebidos sejam os mesmos que foram enviados.

O tamanho fixo mínimo determinado é de 64 *bytes* (12 *bytes* de endereços, 2 *bytes* de tipo, 46 *bytes* de dados e 4 *bytes* de CRC) e o tamanho fixo máximo é de 1518 *bytes* (o que muda aqui em relação ao tamanho mínimo é apenas o campo de dados que é de 1500 *bytes*). Esta determinação é importante na detecção de uma colisão (SOARES, 1995).

A detecção de uma colisão se dá quando uma estação ainda está transmitindo seus dados. Devido a retardos de propagação em eletrônica e *wiring closet*, seria normal que na metade do tempo de transmissão, o sinal transmitido estivesse no ponto mais distante da rede e no caso de uma colisão neste ponto, este sinal teria a outra metade do tempo para retornar ao ponto de transmissão e então alertar ao nó a necessidade de uma retransmissão.

2.4.3 Esquemas de endereçamento

Existem dois tipos de esquemas de endereçamento usados em rede *ethernet* e cada um possui uma finalidade específica.

- Endereçamento Específico – endereço; determinado pelo IEEE; que cada hardware que é fabricado deverá possuir um, composto de 6 *bytes*. Os 3 primeiros *bytes* contêm a identificação do fabricante e os 3 últimos a numeração sequencial, em formato hexadecimal.
- Endereço *Broadcast* – endereço destinado a ser ouvido por todas as estações. Em eventuais momentos as estações enviarão mensagens de *broadcast* informando que o nó está *online*. Um endereço *broadcast* possui todos os caracteres hexadecimais (FF).

2.5 SISTEMAS DE SUPERVISÃO

Os sistemas de supervisão têm por finalidade colocar à disposição do usuário todas as informações das quais ele precisa em um processo, além de poder controlar instrumentos no campo, e tudo isso com acesso rápido em um conjunto de telas (DANEELS; SALTER, 1999). As telas podem conter informações tais como quais dispositivos estão ligados ou desligados, dados de um instrumento, de um processo, além de gerar relatórios gravando o que têm acontecido num período determinado. Para que aconteça a aquisição de dados, o supervisório deve utilizar *softwares*, ligados a um *hardware* de controle através de uma rede de comunicação.

Para obter os dados que o usuário necessita, sensores são utilizados para fazer as leituras, transformando um fenômeno físico em sinais elétricos que devem ser apropriados ao *hardware*, e assim o sistema realiza tarefas pré-programadas, atuando de maneira automática, ou até mesmo aguardando uma ação do usuário. O sinal que vai ser gerado por um sensor, quando necessário deve ser tratado; amplificado, filtrado, isolado para que o sistema interprete e processe.

Assim que o sistema processa os sinais recebidos e toma alguma ação, ele tem a opção de criar históricos, os quais podem ter seus intervalos definidos, sendo gerados automaticamente, ou apenas quando um novo evento acontece. Com os dados dos históricos, o supervisório gera relatórios, descrevendo os eventos que

aconteceram em um período determinado, e estes podem ser impressos ou até mesmo enviados ao usuário através de *e-mail*.

Além de históricos e relatórios, gráficos também podem ser gerados para que o usuário possa visualizar o que tem acontecido com alguns eventos, analisando estes dados e assim tomando alguma ação, mas ainda tem a opção de verificar em tempo real estes mesmos eventos, agindo de maneira imediata para que a eficiência do sistema seja melhorada.

Dependendo do *software* utilizado e também do conhecimento de quem está desenvolvendo o programa, o sistema de supervisão pode ter rotinas já programadas, facilitando e acelerando processos, pois assim o usuário não necessita realizar estas ações, deixando que o sistema atue de maneira independente, como foi especificado pelo programador.

Quando algo deixa de acontecer ou acontece em momento inadequado, ou ainda quando uma falha acontece, o supervisório emite alarmes, para que o usuário fique atento e então realize alguma ação, intervindo para que o processo volte ao seu curso normal, e também o sistema pode intervir de maneira automática, parando um processo específico, referente ao que o alarme está indicando. Analisando estes alarmes, o usuário pode verificar onde têm acontecido mais problemas e identificar quais as possíveis causas.

Para que um sistema de supervisão seja implementado se utiliza o conhecido SCADA (*Supervisory Control and Data Aquisition*), que disponibiliza as funções citadas acima, além de estruturar o sistema, concedendo as ferramentas para fazer a comunicação entre o *software* e o *hardware*.

2.6 SISTEMAS SCADA

Inicialmente utilizados para monitorar de forma simplificada um processo industrial, visualizando estados de equipamentos e os representando através de lâmpadas, indicando ao usuário se estes equipamentos estão ligados ou desligados, se estão com alguma falha ou ainda se estão disponíveis para utilização. Monitoravam também sinais de medidas, mostrando valores pré-determinados, para que o usuário pudesse manter o controle do equipamento.

Com o crescimento acelerado das grandes empresas, os sistemas SCADA tiveram que ser atualizados, conforme novas tecnologias iam sendo lançadas.

Com o poder de processamento de computadores e uma comunicação veloz e eficiente entre os equipamentos de campo e o controlador, o monitoramento e o controle de processos passaram a intervir em locais cada vez mais distantes e complexos, coletando, analisando, armazenando e apresentando ao usuário de forma com que ele possa entender e tomar as decisões necessárias.

Muitas das ações realizadas por este sistema são executadas de maneira automática através de Unidades Terminais Remotas (RTUs) e também por Controladores de Lógica (CLPs). O supervisório normalmente apenas manipula as informações, deixando com que o controle fique com os controladores, mas este sistema também dispõe de ferramentas capazes de trabalhar com os dados, auxiliando os CLPs no processamento destes dados. Com essa divisão de tarefas, o supervisório se torna um sistema de fácil entendimento e utilização, pois o usuário manipula aquilo que lhe é proposto pelo próprio sistema.

O banco de dados utilizado pelo Sistema SCADA é chamado de *tagname*, e é onde as variáveis de entrada e saída contém seus respectivos dados. Os *tagnames* são armazenados de duas maneiras, um apresentando o valor instantâneo da variável e outro demonstrando valores anteriores, ou seja, um histórico. Os valores armazenados pelo sistema são visualizados pelo usuário através de uma Interface Homem Máquina (IHM).

Inicialmente as IHMs eram equipamentos de comunicação estreita, limitando a interação entre dois eficientes processadores de informação, o homem e a máquina. Atualmente, através de tecnologias baseadas na plataforma PC, podem também como citado anteriormente, gerar relatórios, e ainda por meio de redes de comunicação como *Modbus, Profibus, Foundation Fieldbus, DeviceNet* e *Ethernet/TCP-IP* se comunicar à rede corporativa, para que assim esses dados sejam analisados e ações possam ser tomadas.

O banco onde os dados são armazenados, que fazem parte do Sistema SCADA é ligado a IHM, onde estes dados são manipulados, promovem registros, diagnóstico de dados e informações de administração. O usuário visualiza estes dados graficamente, em forma de sinópticos, ou seja, tem uma visão geral do processo.

Além da representação geral do processo, o usuário ainda tem a opção de visualizar de maneira mais detalhada uma determinada área, recorrendo a outra

camada, chamada de Sistema *Multi Layer*, assim podendo navegar entre todas as partes deste processo.

Um Sistema SCADA é composto com alguns subsistemas tais como, Estação de Supervisão, responsável por fazer a interação entre o usuário e o processo, o Sistema de Controle e/ou Aquisição de Dados, conectado aos sensores e atuadores de campo, fazendo as conversões necessárias ao sistema e a Infraestrutura de comunicação do Sistema, conectando a estação de supervisão ao sistema de controle (VIANNA, 2008).

2.7 DESENVOLVIMENTO *WEB*

A *World Wide Web (WWW)* – Rede de Alcance Mundial, é a forma mais utilizada de se compartilhar informação através da internet. Esse modelo faz uso de hipertexto para representar a informação, por meio de textos, imagens, sons, vídeos e páginas, e são visualizados através de programas conhecidos como navegadores. A forma tradicional na qual a maioria das aplicações *web* trabalham é através do protocolo de rede HTTP – *HyperText Transfer Protocol* (KRISHNAMURTHY; REXFORD, 2008).

De acordo com SCHNEIDER et. al. (2008), além do protocolo HTTP, outra tecnologia essencial para a *web* é a linguagem HTML – *HyperText Markup Language.* Sua função é descrever o conteúdo de páginas w*eb* e permitir essas sejam compartilhadas através de ligações de hipertexto conhecidas como *hyperlinks*.

O HTTP utiliza um modelo de Requisição – Resposta para que o usuário acesse páginas ou dados através de *hyperlinks* (SCHNEIDER et. al. 2008). Isso significa que, ao solicitar um recurso do servidor, o usuário deve enviar uma requisição. Essa requisição será então processada pelo servidor, que enviará uma resposta ao usuário. Dessa forma, a cada requisição o usuário deve esperar a resposta do servidor para continuar a execução da aplicação, de forma síncrona, conforme está representado da figura 4.

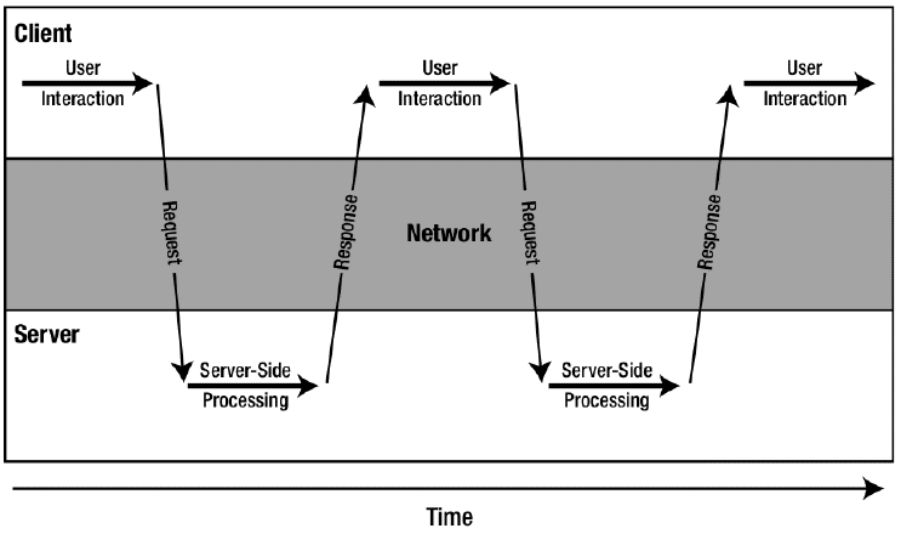


Figura 4: Aplicação *Web* Tradicional
Fonte: (Garrett, 2005)

2.7.1 Requisições Assíncrona com AJAX

O desenvolvimento de novas tecnologias permitiu que aplicações *web* possuíssem maior interação com o usuário, principalmente devido a possibilidade da comunicação com o servidor ser feita de forma assíncrona. Segundo Schneider et. al. (2008), a popularidade desse novo modelo de comunicação trouxe mudanças significantes no modelo clássico de requisição-resposta HTTP.

Para tornar possível a comunicação assíncrona com o servidor, é comumente utilizada a técnica AJAX (*Asynchronous Javascript And* XML). SCHNEIDER et. al. (2008) define AJAX como uma coleção de tecnologias que permitem que os navegadores *web* requisitem dados de um servidor, através do protocolo HTTP, de forma assíncrona, isto é, sem necessidade de intervenção do usuário com botões ou *hyperlinks*. Para isso, AJAX usa uma combinação de quatro componentes (GARRETT, 2005):

- *XMLHttpRequest*: É um objeto desenvolvido na linguagem de script *JavaScript*, cuja função principal é carregar dados de um servidor e manipulá-los sem que seja necessário mostrá-los para o usuário.
- *JavaScript*/DOM: É uma interface que permite a aplicação manipular e mostrar as informações para o usuário.
- CSS: Linguagem usada para definir os estilos das páginas w*eb*.

- XML: Linguagem usada como padrão para transferência de dados entre o servidor e o cliente.

A Figura 5 representa a comunicação entre o navegador e o servidor usando AJAX, onde o elemento AJAX *Engine* representa o código *JavaScript* que gerencia o objeto *XMLHttpRequest* e sua interação com a interface DOM:

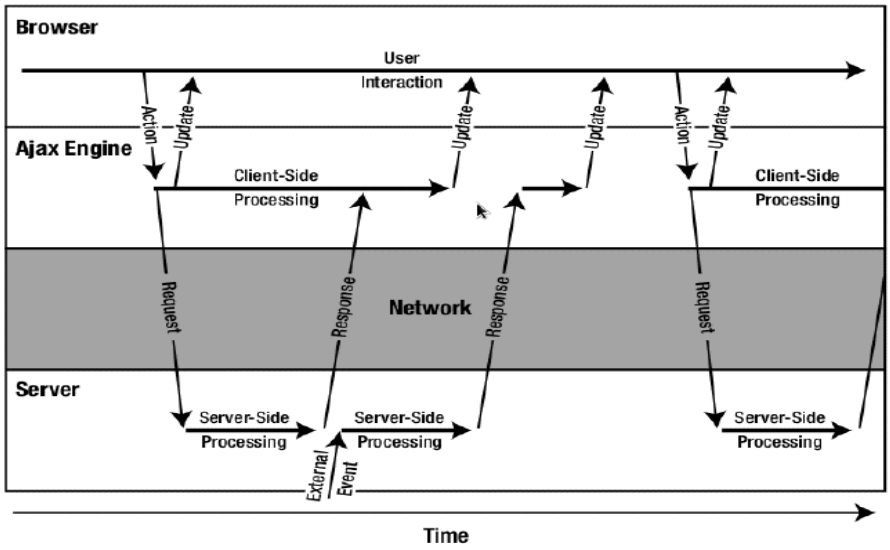


Figura 5: Aplicação *Web* Assíncrona com Ajax
Fonte: (Garrett, 2005)

2.7.2 Google Web Toolkit

O GWT - *Google Web Toolkit* é *um framework* desenvolvido pela empresa Google em 2006, que fornece um conjunto de ferramentas cuja função é auxiliar na criação de aplicações complexas para a *web* (HANSON; TACY, 2007). O foco principal do GWT é trazer o processo de desenvolvimento de uma aplicação *desktop* para uma aplicação *web*, ou seja, criar aplicações *web* com a aparência e o funcionamento de uma aplicação *desktop* tradicional.

Uma característica que torna o GWT diferente de outros *frameworks* é o fato da aplicação ser toda desenvolvida na linguagem de programação Java, incluindo a parte do navegador, o que permite o aproveitamento de todas as vantagens dessa linguagem e a utilização de diversas IDEs – *Integrated Development Environment* (HANSON; TACY, 2007).

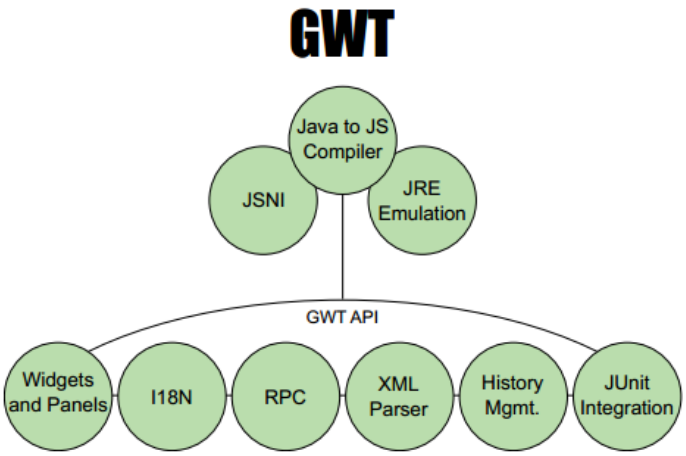


Figura 6: Aspectos principais do GWT
FONTE: (HANSON; TACY, 2007)

A figura 6 representa diversos aspectos do GWT, que podem ser organizados em alguns componentes, de acordo com (SMEETS et. al. 2008):

- Compilador Java para *JavaScript*: É o componente fundamental para o funcionamento do GWT. Sua função é traduzir todo o código desenvolvido em Java, e que será executado no lado cliente, em código *JavaScript*. Com esse compilador é possível desenvolver interfaces complexas, utilizando diversos recursos da linguagem Java.
- Bibliotecas de emulação JRE: Pelo fato da linguagem *JavaScript* ser bastante diferente de Java, para que o compilador do GWT funcione corretamente é necessário fornecer um emulador para que as classes e bibliotecas centrais e os construtores da linguagem possam ser traduzidos para código *JavaScript*.
- JSNI - *JavaScript Native Interface*: Ocasionalmente pode ser necessário escrever código *JavaScript* nativo dentro da aplicação, sem utilizar o compilador GWT. Portanto, foi implementado o JSNI para comportar essa funcionalidade.
- Bibliotecas de Construção da Interface do Usuário: Esse componente abrange os aspectos restantes do GWT, incluindo os *Widgets*, componentes gráficos disponíveis para auxiliar na criação da interface, e o RPC – *Remote Procedure Call*, forma com que a aplicação cliente se comunica com o a aplicação servidor, permitindo que se transfiram objetos entres as aplicações, utilizando uma política de serialização.

3 DESENVOLVIMENTO

Como descrito anteriormente, este trabalho objetiva desenvolver um sistema completo que permita supervisionar e controlar uma residência remotamente. Para alcançar esse objetivo, foram estudadas tecnologias relacionadas a microcontroladores, sistemas de supervisão e sistemas *web*, o que permitiu o desenvolvimento da revisão bibliográfica. Neste capítulo serão descritas as ferramentas escolhidas para o desenvolvimento do projeto, tendo como base a utilização de tecnologias abertas, e a proposta de como este será implementado. Também com base nos objetivos deste trabalho serão especificados os requisitos e a modelagem do sistema.

3.1 FERRAMENTAS UTILIZADAS

A partir da revisão bibliográfica foram definidas as ferramentas utilizadas para o desenvolvimento do projeto. O controle dos dispositivos da residência foi feito através da plataforma Arduino[2]. Essa plataforma inclui uma interface de *hardware* livre com um microcontrolador atmega1280 e uma interface de *software* livre, que inclui ferramentas para auxiliar no desenvolvimento, compilação e gravação do código na linguagem de programação C. Foi utilizado também o conjunto *shield* e biblioteca *ethernet* do Arduino, que permite a criação de servidores *web*. O circuito da placa de conversão de sinais foi projetado através do software *Eagle*[3] versão 6.1.0.

O sistema *web* foi desenvolvido na linguagem de programação Java em conjunto com o GWT. Para tanto foi utilizada a IDE Eclipse[4] Juno e o *Google Plugin for Eclipse*[5]. O sistema de gerenciamento de banco de dados escolhido foi o PostgreSQL[6] versão 9.2. Foi utilizado o *software* Dia[7] para auxiliar na criação de diagramas UML (*Unified Modeling Language*) e na criação de fluxogramas.

[2] http://www.arduino.cc/
[3] http://www.cadsoftusa.com/
[4] http://www.eclipse.org/
[5] https://developers.google.com/eclipse/
[6] http://www.postgresql.org/
[7] http://dia-installer.de/

3.2 PROPOSTA DE IMPLEMENTAÇÃO

A proposta definida para a implementação do projeto foi a de dividi-lo em dois sistemas independentes, onde um é responsável pela parte embarcada, que inclui a placa de conversão de sinais e o programa microcontrolado e o outro é um sistema web que inclui a interface do usuário.

Para cada sistema foram identificados os requisitos para alcançar os objetivos propostos. A partir dos requisitos foi feita a modelagem dos sistemas. No caso do sistema w*eb* a modelagem foi baseada na UML, com diagramas de implantação, diagrama de casos de uso e diagramas de sequência. Para o sistema embarcado foram feitos um fluxograma do programa e o diagrama esquemático da placa de conversão de sinais.

3.3 ESPECIFICAÇÃO DOS REQUISITOS

A especificação dos requisitos é uma etapa fundamental para o processo de desenvolvimento de um sistema, pois é nela que são identificadas todas as necessidades do usuário e definidas as condições a serem alcançadas pelo sistema. Isto permite ter um entendimento mais claro sobre o comportamento deste sistema e diminui a necessidade de alterações durante o desenvolvimento. Para este projeto foram levantados os requisitos dos sistemas de forma independente.

3.3.1 Sistema De Controle

O sistema de controle deve possuir um programa que execute um servidor *web*. Este deve ser capaz de receber requisições HTTP do tipo GET e utilizar a autenticação *basic* para restringir o acesso ao servidor *web.* Deve também fazer a leitura dos parâmetros da requisição e validá-los de acordo com o conjunto de regras definidas. Estas regras formam comandos, listados na Tabela 2, que o programa deve ser capaz de identificar e executar uma função específica.

Comando	Descrição	Entradas/Saídas
AN010 [1 a 2]	Faz a leitura de uma entrada analógica no padrão de 0 a 10 Vcc	Portas Analógicas 14 e 15
AN420 [1 a 2]	Faz a leitura de uma entrada analógica no padrão 4 a 20 mA.	Portas Analógicas 12 e 13
LDIGC [1 a 3]	Faz a leitura de uma entrada digital comum.	Portas Digitais 31, 33 e 35
LDIGR [1 a 4]	Faz a leitura de uma entrada digital a Relé	Portas Digitais 37, 39, 41 e 43
EDIGC [1 a 3]	Envia um comando para uma saída digital comum. Ex.: EDIGC1	Portas Digitais 30, 32 e 34
EDIGR [1 a 4]	Envia um comando para uma saída digital a Relé. Ex.: EDIGR3	Portas Digitais 40, 42, 44 e 46
PWM [1 a 3]	Envia um comando variando de 0 a 255 (0 a 5 Vcc) para uma saída PWM. Ex.: PWM2	Portas Digitais 4, 6 e 7

Tabela 2: Lista de Comandos do Arduino
Fonte: Autoria Própria

O programa também deve ser capaz de enviar respostas para a requisição feita pelo cliente, podendo ser:

- Requisição Não Autorizada [401]: Quando o programa não identificar no cabeçalho do protocolo o campo de autorização algum *login* e senha válidos, deve enviar como resposta um código de estado “401 *Unauthorized*”.
- Requisição com Sintaxe Inválida [400]: Quando o programa não conseguir processar os comandos passados como parâmetros na requisição. Este deve enviar uma resposta de sintaxe inválida com um código de estado “400 *Bad Request*”.
- Comando Válido [200]: Quando o programa validar os comandos corretamente, este envia uma resposta com um código de estado “200 OK”. Caso possua comandos de entrada, deve enviar no corpo da resposta um objeto JSON contendo informações sobre o nome da entrada e o valor lido.

Além do programa, o sistema deve possuir uma placa de conversão de sinal utilizada para adaptar padrões comumente empregados no mercado para sensores e atuadores às entradas e saídas do Arduino. Com esse objetivo, definiram-se os sinais que devem ser suportados pela placa, descritos na Tabela 3:

Quantidade	**Sinal**
2	Entrada Analógica [0 a 10Vcc]
2	Entrada Analógica [4 a 20mA]
3	Entrada Digital Comum [0-5Vcc]
1	Entrada a Relé [5Vcc]
1	Entrada a Relé [12Vcc]
2	Entrada a Relé [24Vcc]
3	Saída Digital Comum [0-5Vcc]
2	Saída a Relé [0 a 127Vca]
2	Saída a Relé [0 a 36Vcc]
3	Saída PWM [0 a 5Vcc]
1	Alimentação [5Vcc]
1	Alimentação [9Vcc]
1	Alimentação [24Vcc]

Tabela 3: Requisitos da Placa para Conversão de Sinais
Fonte: Autoria Própria

A alimentação da placa será fornecida por uma fonte externa de 24Vcc/2,5A. O circuito deve disponibilizar saídas de 5, 9 e 24Vcc, nas quais a corrente é limitada pela fonte. O Arduino deve ser alimentado internamente pelo circuito da placa com uma tensão de 9Vcc.

3.3.2 Sistema De Supervisão

O sistema de supervisão deve fornecer a interface com que o usuário irá interagir para ter acesso aos dispositivos do sistema embarcado, através de uma aplicação w*eb* compatível com os navegadores *Internet Explorer*, *Chrome* e *Firefox*. Os dados persistentes dessa aplicação devem ser armazenados em um banco de dados.

Os requisitos que devem ser considerados no desenvolvimento dessa aplicação são:

- Contas de usuário: Apenas usuários cadastrados poderão utilizar o sistema. Deve existir uma conta padrão do administrador e, a partir dessa conta, outros usuários podem ser cadastrados. Devem existir também três níveis de privilégio que podem ser atribuídos a um usuário durante o cadastro, são eles:
    - Usuário Administrador: Possui acesso a todas às funcionalidades do sistema;
    - Usuário Normal: Possui acesso a maioria das funcionalidades do sistema, porém não pode cadastrar novos usuários;
    - Usuário Restrito: Pode apenas visualizar e controlar os dispositivos já existentes no sistema, não podendo alterar o conteúdo do banco de dados.
- Gerenciamento de sessão: O servidor deve manter uma sessão para cada usuário que fizer *login* no sistema, de forma que, a cada requisição feita ao servidor, seja possível identificar o usuário e se ele possui privilégio suficiente para executar a ação requisitada. O usuário pode encerrar a sua sessão manualmente, fazendo *logout* do sistema. A sessão também pode ser encerrada automaticamente, caso o usuário fique inativo por um período de 10 minutos.
- Dispositivos: Os dispositivos conectados ao sistema embarcado devem ser cadastrados pelo usuário no sistema de supervisão, informando:
    - Nome: Identificação do dispositivo;
    - Descrição: Informações adicionais sobre o funcionamento do dispositivo;
    - Tipo (Entrada/Saída): Especifica se o dispositivo é sensor ou atuador;
    - Subtipo (Tipo do Sinal): Especifica o padrão de sinal usado pelo dispositivo;
    - Slot da placa: Especifica em qual slot da placa o dispositivo foi conectado.

  Também deve ser possível alterar ou excluir os dispositivos cadastrados, seguindo algumas regras:

  - O usuário deve possuir privilégio normal ou administrador para gerenciar os dispositivos;
  - Cada slot da placa só pode possuir um dispositivo cadastrado;
  - Não é possível excluir um dispositivo que possui regras de alarme associadas a ele.
- Alarmes: Deve ser possível criar associações entre dispositivos de entrada e saída, de forma que quando um determinado valor da entrada for atingido deve executar uma determinada ação no dispositivo de saída. Ao cadastrar um alarme, o usuário deve informar:
  - Tipo (Mínimo/Máximo): Especifica se o alarme será acionado quando ultrapassar o valor mínimo ou máximo definido;
  - Origem: Especifica um dispositivo de entrada cadastrado no sistema;
  - Destino: Especifica um dispositivo de saída cadastrado no sistema;
  - Valor de Entrada: Define o valor que o dispositivo de entrada deve ultrapassar para acionar o alarme;
  - Valor de Saída: Define o valor que será enviado para o dispositivo de saída.

  Também deve ser possível alterar ou excluir os alarmes existentes, seguindo a mesma regra de privilégios para os dispositivos.
- Supervisório: Espaço onde deve ser possível se comunicar com o sistema embarcado e fornecer as seguintes funcionalidades:
  - Coletar valores de leitura dos sensores conectados ao sistema embarcado e cadastrados no sistema de supervisão, utilizando um intervalo de 10 segundos entre cada leitura;
  - Enviar comandos aos atuadores conectados ao sistema embarcado e cadastrados no sistema de supervisão;
  - A cada leitura dos sensores, verificar se algum alarme atingiu a condição necessária para ser acionado e, se for o caso, executar o comando de saída especificado no alarme.
  - Fornecer uma interface gráfica ao usuário para que ele possa visualizar os valores lidos dos sensores e enviar comandos aos atuadores.

- Histórico: As atividades realizadas dentro do sistema supervisório devem ser registradas em arquivos de log, informando junto com a descrição da atividade o horário que aconteceu e o usuário pertencente a sessão atual.

3.4 MODELAGEM DO SISTEMA

A modelagem é uma etapa importante no desenvolvimento de um sistema, pois permite identificar os recursos que serão necessários para atingir os requisitos, como será a estrutura do sistema e diminuir a necessidade de alterações durante a implementação do projeto.

É importante escolher uma linguagem de modelagem adequada para cada sistema, utilizando diagramas que representem corretamente sua estrutura. Portanto cada sistema deste projeto foi modelado separadamente conforme a Figura 7:

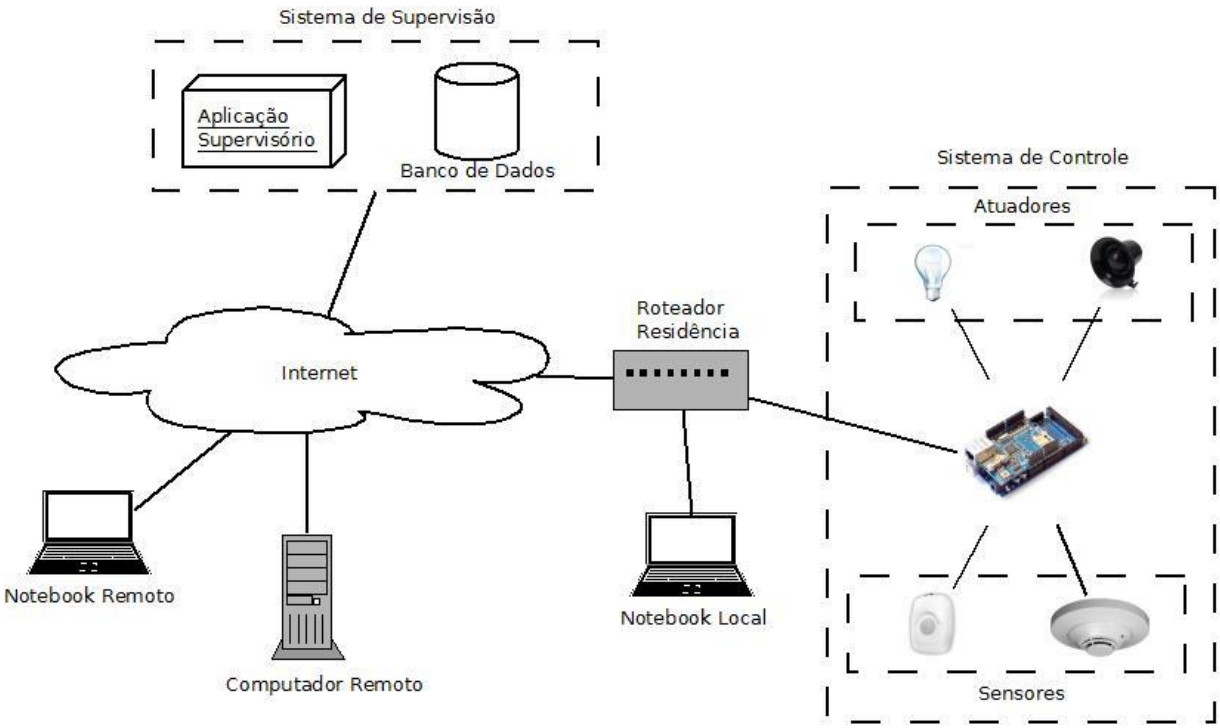


Figura 7: Estrutura Geral do Sistema
Fonte: Autoria Própria

3.4.1 Sistema De Controle

A placa de conversão de sinais utilizada pelo sistema de controle foi projetada contendo circuitos específicos para as conversões necessárias, conforme está descrito na Tabela 4:

Porta Arduino	**Entrada/Saída da Placa**	**Descrição**
Entrada Analógica [0 a 5Vcc]	Entrada Analógica [0 a 10Vcc]	Um sinal de 0 a 10Vcc é convertido em um sinal de 0 a 5Vcc através de um divisor de tensão com dois resistores.
Entrada Analógica [0 a 5Vcc]	Entrada Analógica [4 a 20mA]	Um sinal de corrente de 4 a 20mA é convertido em um sinal de tensão de 0 a 5Vcc através de um divisor de tensão com um resistor em paralelo com a entrada do Arduino.
Entrada Digital [0-5Vcc]	Entrada Digital [0-5Vcc]	O sinal é enviado diretamente ao Arduino.
Entrada Digital [0-5Vcc]	Entrada a Relé [5, 12 e 24Vcc]	O sinal de entrada faz o chaveamento de um rele, que manda um sinal para um optoacoplador e este envia um sinal de 0 ou 5Vcc para a entrada do Arduino.
Saída Digital [0-5Vcc]	Saída Digital [0-5Vcc]	O sinal do Arduino é enviado diretamente para a saída da placa.
Saída Digital [0-5Vcc]	Saída a Relé [0 a 127Vca e 0 a 36Vcc]	O Arduino fornece um tensão de 0 ou 5Vcc, chaveando um relé, onde seus contatos suportam tensões até 36Vcc ou 127Vca.
Saída PWM [0 a 5Vcc]	Saída PWM [0 a 5Vcc]	O sinal do Arduino é enviado diretamente para a saída da placa.

Tabela 4: Lista de Conversão de Sinais
Fonte: Autoria Própria

A partir da análise da Tabela 4, foi construído o diagrama esquemático da placa, o qual é mostrado na Figura 15 no Apêndice A, que tem a função de

representar o circuito eletrônico de forma esquemática, onde os componentes são representados pelos seus símbolos, e ainda, como são conectados.

A outra parte deste sistema, composta pelo programa do Arduino, tem o seu comportamento descrito no fluxograma da Figura 16 no Apêndice A. Nessa etapa a tecnologia *ethernet* foi aplicada em conjunto com o Arduino através do *shield ethernet*, que fornece uma interface de comunicação entre o Arduino e uma rede *ethernet.* Seus componentes principais são um controlador *ethernet* W5100, próprio para aplicações embarcadas, entrada para conector RJ45 e *socket* para cartão microSD, conforme a Figura 8 (FITZGERALD, 2010).

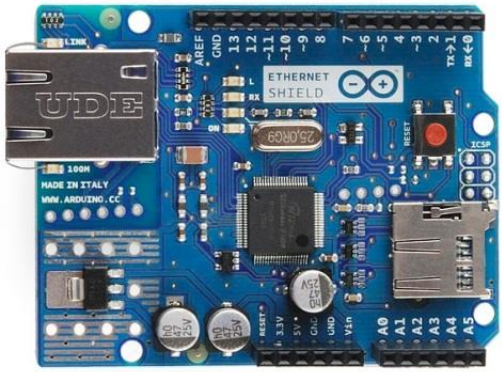


Figura 8: Arduino *Shield Ethernet*
Fonte: FITZGERALD (2010)

A etapa inicial do programa do Arduino é composta pelo *setup*, onde são inicializadas as variáveis do programa e os pinos do microcontrolador utilizados pela placa de conversão de sinais. Também é nessa etapa que o servidor w*eb* é configurado e inicializado, sendo definido o endereço IP e a porta do servidor, além do endereço MAC do *shield ethernet*.

Após a inicialização, o programa aguarda que uma nova conexão seja estabelecida com o servidor para então fazer a leitura da requisição enviada pelo cliente, que deve ser feita através do método GET do protocolo HTTP. A requisição deve possuir uma autenticação no seu cabeçalho, contendo um usuário e senha válidos. Para verificar essa condição o programa busca pelo campo de autenticação e compara seu valor com um usuário e senha pré-definidos e, caso sejam diferentes, envia uma resposta ao cliente informando uma falha na autenticação.

A próxima etapa do programa é processar os parâmetros contidos na requisição. Nessa etapa os parâmetros são analisados, com o objetivo de identificar e separar os comandos de seus valores, baseando-se nos comandos especificados

na tabela 1. Caso a sintaxe dos parâmetros não seja reconhecida, uma resposta é enviada ao cliente informando que existem erros na sintaxe da requisição.

A última etapa do programa é responsável por executar as instruções referentes aos comandos lidos. No caso de comandos de saída, os valores são atribuídos aos pinos de saídas do microcontrolador e, no caso dos comandos de entrada, é feita a leitura dos pinos de entrada, enviando uma resposta ao cliente contendo um objeto JSON formado pelos dados lidos.

Ao final de cada resposta enviada ao cliente, a conexão é encerrada e o programa volta a aguardar que uma nova conexão seja estabelecida com o servidor.

3.4.2 Sistema de Supervisão

A partir dos requisitos levantados anteriormente na seção 3.3.2 deste trabalho, foram modelados diagramas que representam a estrutura e o comportamento do sistema de supervisão.

3.4.3 Diagrama de Casos de Uso

Um diagrama de casos de uso representa o comportamento do sistema, pois ele informa através dos casos de uso quais as funcionalidades que o sistema deve oferecer, sem se preocupar com a forma que isso será implementado ou com a sequência em que devem ocorrer.

Com base nos requisitos definidos na seção 4.1.2 deste trabalho, foi modelado um diagrama de casos de uso representando a aplicação *web* do supervisório, conforme a Figura 17 no Apêndice A.

O diagrama é composto por atores, representando entidades externas que interagem com o sistema, e casos de uso, que representam as funcionalidades do sistema. Existem ainda as associações entre atores e casos de uso, ou entre dois casos de uso. No primeiro caso, essas ligações informam com quais funcionalidades do sistema um ator interage diretamente, já o segundo caso pode representar tanto uma inclusão, onde um caso de uso depende de outro para completar seu funcionamento, e uma extensão, onde um caso de uso pode ou não estender sua funcionalidade para outro caso de uso. O sistema de supervisão possui quatro atores, são eles:

- Usuário: Pessoa que utiliza o sistema e interage com a maioria dos casos de uso, com exceção dos casos de uso: Verificar Privilégio e Verificar *Login*.
- Banco de Dados: É onde são armazenados os dados persistentes do sistema, sendo utilizado pelos casos de uso que manipulam esses dados.
- Arquivo: É o arquivo onde são armazenados os logs do sistema e interage com o caso de uso Visualizar Histórico.

A Tabela 5 contém uma breve descrição sobre a funcionalidade de cada caso de uso do sistema:

Caso de Uso	**Descrição**
Efetuar *Login*	Iniciar uma sessão no servidor através de um usuário e senha válidos.
Efetuar *Logout*	Encerrar uma sessão no servidor.
Alterar Senha	Redefinir a senha do usuário da sessão atual.
Cadastrar Usuário	Cadastrar um novo usuário no sistema.
Gerenciar Dispositivo	Cadastrar/Alterar/Excluir um dispositivo no sistema.
Gerenciar Alarmes	Cadastrar/Alterar/Excluir um alarme no sistema.
Visualizar Histórico	Mostrar os *logs* dos eventos que ocorreram no sistema.
Ler Valor do Dispositivo	Mostrar os valores dos dispositivos recebidos do sistema de controle.
Executar Comando	Enviar um comando para o sistema de controle.
Verificar Privilégio	Verificar se o usuário da sessão atual possui privilégios para executar alguma função do sistema.
Verificar *Login*	Buscar o usuário pertencente a sessão atual, caso ela não tenha expirado.

Tabela 5: Descrição dos Casos de Uso
Fonte: Autoria Própria

3.4.3.1 Diagrama de Classes

O diagrama de classes tem o objetivo de representar a composição e a estrutura estática de um determinado sistema orientado a objetos através de suas classes. Esse diagrama é usado como uma forma simplificada de mostrar como objetos do sistema interagem entre si.

O diagrama é formado por classes, contendo seus métodos e atributos, e seus relacionamentos, sendo possível suprimir os métodos e atributos das classes para exibir um diagrama simplificado. Os relacionamentos entre classes podem possuir cardinalidade, explicitando a quantidade de objetos que fazem parte da relação. Os principais relacionamentos são:

- Generalização: É quando uma classe herda atributos ou métodos de outra classe mais genérica. É representado por uma seta vazia apontando para a classe genérica;
- Realização: É quando uma classe implementa métodos definidos em outra classe, geralmente uma interface. É representado por uma seta vazia tracejada apontando para a classe que define o método;
- Associação: É quando existe qualquer ligação simples entre duas classes. É representado por uma linha simples;
- Agregação: É quando uma classe é agregada a outra, mas sem criar uma dependência entre elas, ou seja, a classe A contém a classe B, mas a classe B existe independentemente da classe A. É representado por uma linha com um losango vazio no lado da classe agregadora;
- Composição: É quando uma classe é composta por outra, criando uma dependência entre elas, ou seja, a classe A contém a classe B e, se a classe A deixar de existir, a classe B também será eliminada. É representado por uma linha com um losango preenchido no lado da classe com a composição.

A partir do diagrama de casos de uso e dos requisitos especificados na seção 4.1.2 deste trabalho, foi possível modelar as classes do sistema de supervisão, dividindo-as em três diagramas de classe simplificados, para facilitar a sua compreensão.

A Figura 18 no Apêndice A apresenta um diagrama de classes simplificado referente às classes que compões os dados manipulados pelo sistema de supervisão. Todas as classes desse diagrama implementam a interface *Serializable*, disponibilizada pela biblioteca do GWT. Isso é necessário, pois as instâncias das classes de dados serão transferidas entre o cliente e o servidor através dos RPCs, que precisam serializar o objeto para transmiti-lo pela rede. Os relacionamentos presentes nesse diagrama são realização, agregação e composição.

O segundo diagrama, representado pela Figura 19 no Apêndice A, expõe as classes referentes às telas do sistema de supervisão. As classes *Composite* e *DialogBox* funcionam como contêineres de componentes gráficos pertencentes a biblioteca de *Widgets* do GWT. Todas as classes de telas do sistema derivam de um *Composite*, com exceção das classes AdicionaDispositivo e AdicionaAlarme que derivam da classe *DialogBox*, para que possuam características de janela *pop-up*. Além disso, existem duas interfaces *AdicionaDispositivoOpener* e *AdicionaAlarmeOpener*, que são implementadas pelas classes TelaDispositivo e TelaAlarme, com a função de permitir a troca de objetos entre as telas e as janelas *pop-up*. Os relacionamentos presentes nesse diagrama são generalização, composição e realização.

As classes do sistema de supervisão que têm a função controlar o sistema, implementando a lógica de seu funcionamento, e estão representadas pelo diagrama da Figura 20 no Apêndice A. As classes com sufixo DAO (*Data Access Object*) são responsáveis por manipular o banco de dados, de forma que cada uma controla os dados referentes a uma classe de dados equivalente. Portanto, as classes DAO são associadas à classe *ConnectionDBFactory*, que por sua vez mantém uma conexão com o banco de dados através da classe *Connection*. Outra classe importante é a VerificaLogin, que consegue obter dados sobre a sessão do usuário através da classe *HttpSession*, e é utilizada por todos os serviços antes de executar suas funcionalidades, para verificar se a sessão ainda existe e se o usuário possui os privilégios necessários. As outras classes do diagrama fazem parte da comunicação entre o cliente e o servidor. Para que seja possível utilizar a tecnologia RPC do GWT, é necessária a criação de três classes, são elas:

- *ClasseService*: Para que a aplicação no lado cliente possa chamar um serviço no lado servidor, deve ser criada uma interface com o sufixo *Service*, contendo a definição de todos os métodos desse serviço.
- *ClasseServiceAsync*: Uma interface com sufixo *ServiceAsync* também deve ser criada no lado cliente, fazendo com que a chamada ao servidor seja feita de forma assíncrona.
- *ClasseServiceImpl*: Esta classe deve conter a implementação dos métodos do serviço no lado do servidor. Ela deriva da classe *RemoteServiceServlet* da biblioteca do GWT, de forma que o serviço será transformado internamente em um *Servlet*.

Portanto, o diagrama contém as classes referentes aos serviços do Arduino, Usuário, Dispositivo, Alarme e *Login*. Os relacionamentos presentes nesse diagrama são generalização, realização, agregação e associação.

3.4.3.2 Diagrama Entidade Relacionamento

A forma mais comum de se representar a estrutura de um banco de dados é pelo modelo Entidade Relacionamento, A ideia desse modelo é criar uma abstração do mundo real que seja independente da tecnologia usada no banco de dados. As entidades representem objetos reais e seus atributos, como pessoas ou eventos, e os relacionamentos representam associações entre as entidades.

Os relacionamentos podem ter três tipos de cardinalidade:

- Um para Um: Cada ocorrência de uma entidade A está relacionada a apenas uma ocorrência de uma entidade B.
- Um para Muitos: Cada ocorrência de uma entidade A está relacionada a uma ou mais ocorrência de uma entidade B, porém uma ocorrência da entidade B só está relacionada a uma ocorrência da entidade A.
- Muitos para Muitos. Cada ocorrência de uma entidade A está relacionada a uma ou mais ocorrência de uma entidade B. Uma ocorrência da entidade B está relacionada a uma ou mais ocorrência da entidade A.

A Figura 21 no Apêndice A apresenta o diagrama do banco de dados usado pelo sistema de supervisão, baseado no modelo Entidade Relacionamento. Algumas entidades, como Tipo, Subtipo e Alarme_Tipo possuem dados pré-definidos, não podendo ser alterados por um usuário. Outras entidades, como Dispositivo, Alarme e *Login*, armazenam os dados cadastrados pelo usuário do sistema.

3.4.3.3 Diagrama de Implantação

O sistema de supervisão é formado por uma aplicação *web* desenvolvida na linguagem de programação Java. Como toda aplicação *web*, parte de suas funcionalidades são executadas do lado do cliente, através de um navegador e parte de suas funcionalidades são executadas em um servidor *web*. Conforme definido previamente, a aplicação deve ser compatível com os navegadores *Internet*

Explorer, *Firefox* e *Chrome*, O *Apache Tomcat*[8] foi o contêiner *Servlet* escolhido para executar o código Java do lado do servidor, junto com o sistema gerenciador de banco de dados *PostgreSQL*.

Com o objetivo de auxiliar na criação dessa aplicação, escolheu-se a utilização do *framework* GWT, devido ao fato de não exigir conhecimentos profundos sobre *Web Standards* e especificações Java EE, permitindo que o desenvolvimento seja direcionado diretamente às regras de negócio. Além disso, o GWT possui flexibilidade em relação a compatibilidade com os principais navegadores existentes atualmente, incluindo aqueles especificados para o sistema supervisório.

Portanto, a comunicação entre o lado cliente e o lado servidor da aplicação é feita através de chamadas assíncronas através de serviços RPCs fornecidos pelo GWT e a comunicação entre o lado servidor e o sistema de controle é feita no modelo Requisição-Resposta do protocolo HTTP.

A Figura 22 no Apêndice A apresenta um diagrama de implantação do sistema de supervisão, representando a arquitetura física sobre a qual esse sistema foi implementado.

[8] http://tomcat.apache.org/

4 RESULTADOS E DISCUSSÕES

Neste capítulo é descrito como foi feita a implementação do sistema de controle e do sistema de supervisão. Em seguida, são mostrados os resultados obtidos a partir de testes realizados para verificar o funcionamento dos sistemas.

4.1 IMPLEMENTAÇÃO DO SISTEMA DE CONTROLE

Para desenvolver o software do sistema de controle, duas bibliotecas do Arduino foram essenciais: *String Object* e *Ethernet*. A biblioteca *String* fornece funções que permitem manipular cadeias de caracteres armazenadas em objetos *String*. Suas funcionalidades foram aplicadas no tratamento das requisições HTTP, armazenando e manipulando seu conteúdo.

A biblioteca *Ethernet* funciona em conjunto com o *shield ethernet*, de forma que possibilita o Arduino se conectar à internet. Ela pode trabalhar tanto como um servidor, recebendo conexões externas, quanto como um cliente, se conectando a outros servidores. Para o sistema de controle a biblioteca foi utilizada como um servidor, recebendo conexões do sistema de supervisão.

Para facilitar a organização do código, foi criada a classe Requisição, com a função de converter os parâmetros da requisição em comandos e valores. Também possui a função de validar a requisição, autenticando o usuário e verificando a sintaxe dos parâmetros. A Figura 9 representa um diagrama contendo a classe Requisição:

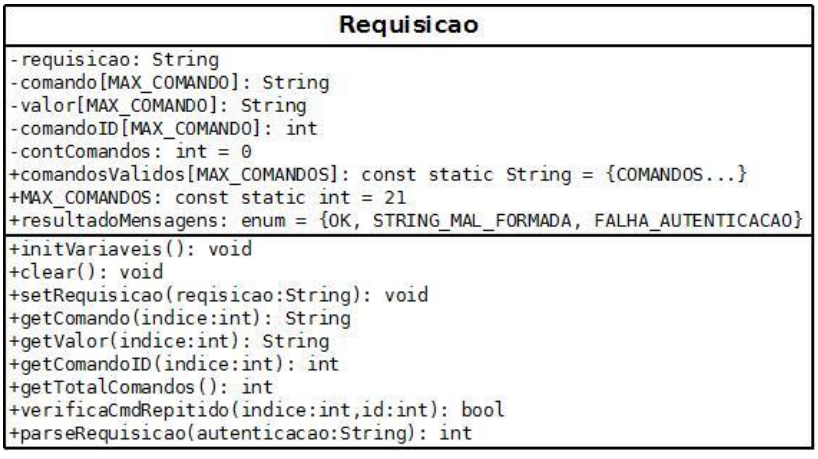


Figura 9: Classe Requisição
Fonte: Autoria Própria

As principais funções do programa são:

- `inicializaPinos(`*`void`*`)`: Inicializa os pinos do Arduino que são utilizados pela placa de conversão de sinais.
- `aguardaRequisicao(`*`void`*`)`: Aguarda que uma nova conexão seja estabelecida com a biblioteca *Ethernet*, através da função server.available.
- `recebeRequisicao(`*`void`*`)`: Essa função é chamada depois de se estabelecer uma conexão com um cliente. O conteúdo da requisição é armazenado e enviado para a função `parseRequisicao`.
- `parseRequisicao(`*`String`*`)`: Autentica e valida a requisição. Para fazer a autenticação, o programa busca pelo campo *`Authentication basic string_base_64`* no cabeçalho da requisição HTTP, onde *`string_base_64`* representa uma *string* contendo os dados do usuário no formato `usuário:senha`, codificados usando o método *base64*. Os comandos e seus valores são recebidos como parâmetros da requisição, que devem seguir o padrão:

  `http://endereco/?CMD1=VALOR1&CMD2=VALOR2`

  O programa identifica os comandos e seus valores através dos caracteres separadores '=' e '&' e armazena-os no objeto `requisicao`. Durante esse processo, o programa verifica também se existem erros de sintaxe como caracteres inválidos, comandos repetidos e comandos ou valores inválidos.
- `processaComandos(`*`void`*`)`: A partir dos comandos gerados pelo `parseRequisicao`, executa o código equivalente a cada um. Para comandos de entrada, os valores dos pinos são lidos através das funções do Arduino *`analogRead`* e *`digitalRead`*. Para os comandos de saída, os valores são atribuídos aos pinos pelas funções *`analogWrite`* e *`digitalWrite`*.
- `enviaResposta(`*`int`*`)`: Gera uma resposta HTTP de acordo com o resultado das funções `parseRequisicao` e `processaComandos`, e a envia ao cliente através da biblioteca *Ethernet*.

Em conjunto com o *software* do Arduino foi desenvolvida a placa de conversão de sinais, baseada no diagrama esquemático da Figura 15 no Apêndice A. O resultado é demonstrado na figura 10:

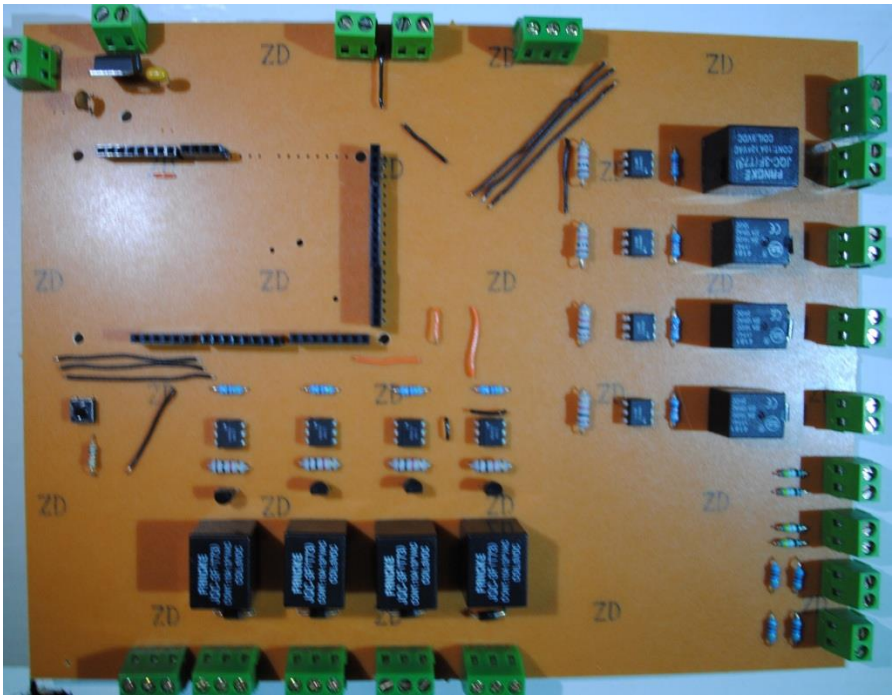


Figura 10: Placa de Conversão de Sinais
Fonte: Autoria Própria

4.2 IMPLEMENTAÇÃO DO SISTEMA DE SUPERVISÃO

O sistema de supervisão foi implementado através de uma aplicação w*eb*. Essa aplicação foi desenvolvida na linguagem Java, com o auxílio do *framework* GWT.

Ao contrário da maioria dos *frameworks* para desenvolvimento w*eb* com Java, o GWT permite criar aplicações onde a interação do usuário com o sistema é próxima a proporcionada por aplicações *desktop*. Para isso, o código é desenvolvido separadamente, divido em duas partes distintas, onde as classes relacionadas à visão ficam no pacote cliente, as classes relacionadas às funcionalidades ficam no pacote servidor e as classes relacionadas a dados são compartilhadas pelos dois. Essa divisão é necessária por que as classes do lado cliente possuem restrições para poderem ser compiladas em código *JavaScript*.

Com base nos diagramas modelados anteriormente, a lógica de funcionamento da aplicação foi implementada através das classes de serviço descritas na Figura 20 no Apêndice A. Dessa forma, todas as funcionalidades do

sistema ficaram no servidor, incluindo a manipulação do banco de dados, a comunicação com o sistema de controle e o gerenciamento de sessão, o que garantiu uma segurança maior ao sistema.

A interface do usuário foi feita a partir do diagrama de classes da Figura 19 no Apêndice A, implementando todas as telas especificadas no diagrama. A criação das telas dependeu de *Widgets* como campos de texto, botões, imagens, e rótulos, que foram inseridos em diversos painéis, todos parte da biblioteca gráfica do GWT. O comportamento desses componentes foi estabelecido através de respostas a eventos externos, como o clique do *mouse* sobre um botão.

As classes de dados, representadas na Figura 18 no Apêndice A, foram implementadas tanto no lado servidor quanto no lado cliente, pois foram utilizadas para transferir objetos entre as duas partes do sistema.

Outro fator que se considerou durante o desenvolvimento foi a segurança. Duas medidas foram tomadas para fazer com que a aplicação fosse mais confiável:

- Criptografia de senhas: Quando um usuário é cadastrado no sistema, antes da senha ser inserida no banco de dados ela é criptografada usando o método *bcrypt*. Após isso, não há como recuperar a senha original, portanto, mesmo que se tenha acesso ao banco de dados, as senhas dos usuários não serão expostas.
- Conexão com o servidor por meio de HTTPS (*HyperText Transfer Protocol Secure*): Esse tipo de conexão utiliza o protocolo SSL/TSL (*Security Sockets Layer/Transfer Layer Security*) em conjunto com o protocolo HTTP criar um canal seguro sobre uma rede aberta. Isso é garantido pelo protocolo SSL/TSL, que é responsável por criptografar os dados antes de serem enviados e retorná-los ao formato original após chegar ao destino, tanto do navegador para o servidor web, quanto no sentido inverso. Para poder usar essa funcionalidade no sistema de supervisão, foi necessário gerar um certificado digital e adicioná-lo ao *Tomcat*, por esse ser o contêiner de *servlets* utilizado para armazenar e gerenciar a aplicação. É através desse certificado que um navegador reconhece a utilização do protocolo SSL/TSL e inicia uma conexão segura com o *Tomcat*.

4.3 TESTES PRÁTICOS

Para avaliar a qualidade do projeto, foram realizados testes com o objetivo de verificar o funcionamento das principais características do sistema de supervisão e do sistema de controle.

Os testes foram feitos de modo com que os requisitos especificados na seção 3.3 desse trabalho fossem validados. Para tanto, foi definida uma sequência de testes, onde um componente dos sistemas foi acrescentado a cada novo teste, até que o projeto inteiro fosse testado de forma integrada. Esse tipo de teste é chamado de caixa preta, pois os componentes são testados como um todo, ou seja, consideram-se apenas as suas entradas e as saídas esperadas e obtidas, sem se preocupar as etapas internas do processo.

Dois testes foram realizados para validar o funcionamento do sistema de controle. Primeiramente, foi verificado o funcionamento do programa do Arduino, com o auxílio do monitor serial, onde é possível visualizar mensagens enviadas pelo Arduino através da comunicação RS232, funcionando como um *debugger* para o programa. Nesse teste foram geradas mensagens de saída corretamente, conforme a figura 11, para três valores de entrada diferentes:

- Caso 1: Foi simulada uma situação normal, onde se recebeu uma requisição válida através de uma autenticação válida. Isso resultou na execução do programa até o final, gerando a resposta esperada (código 200);
- Caso 2: Foi simulada uma situação em que se recebe uma requisição contendo comandos e valores incorretos. O resultado disso foi a suspensão da execução normal do programa e o envio da mensagem de sintaxe inválida (código 400);
- Caso 3: Foi feita uma requisição ao endereço do servidor do Arduino sem informar um usuário válido. O programa não identificou o campo de autorização no cabeçalho da requisição e enviou uma mensagem de erro (código 401).

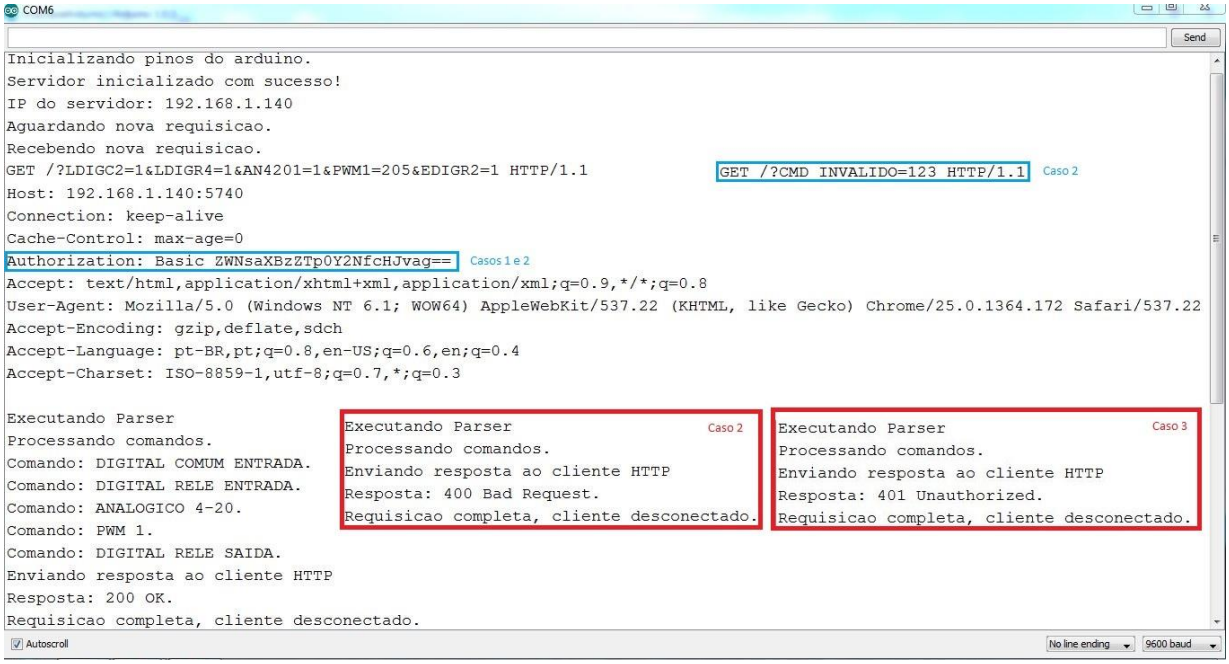


Figura 11: Teste de execução do programa do Arduino
Fonte: Autoria Própria

Em seguida foi realizado um teste integrando a placa de conversão de sinais com o programa do Arduino. As entradas digitais da placa foram alimentadas por uma fonte de tensão externa e seus valores foram lidos corretamente utilizando o programa do Arduino através de um navegador *web*. O procedimento para as entradas analógicas foi semelhante, porém os sinais de entrada foram obtidos por meio de um gerador de funções.

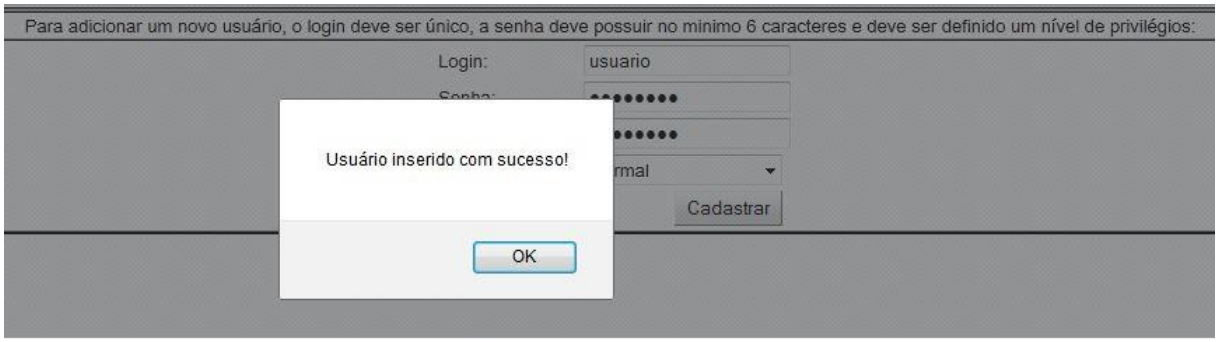
As saídas da placa foram todas testadas por comandos enviados ao programa do Arduino através do navegador *web*, fazendo a leitura dos valores de tensão nas saídas da placa com um multímetro, verificando se eram compatíveis.

Os resultados das entradas corresponderam aos valores esperados, com pequenas variações nas leituras das entradas analógicas, já que o circuito de conversão dos sinais 4-20 mA e 0-10 Vcc para 0-5 Vcc não possuem precisão muito apurada. Foi observado que os valores lidos nas saídas da placa são um pouco inferiores aos valores esperados. Isso ocorreu devido a quedas de tensão no circuito da placa, que ocasionaram nessa restrição. Portanto, o teste cumpriu parcialmente os resultados esperados, o que não impediu de o sistema de controle funcionar normalmente.

Para verificar o funcionamento do sistema de supervisão, foram realizados testes para validar as ações em relação ao gerenciamento de usuários, dispositivos

e alarmes, o funcionamento da tela de supervisão e a comunicação entre os dois sistemas.

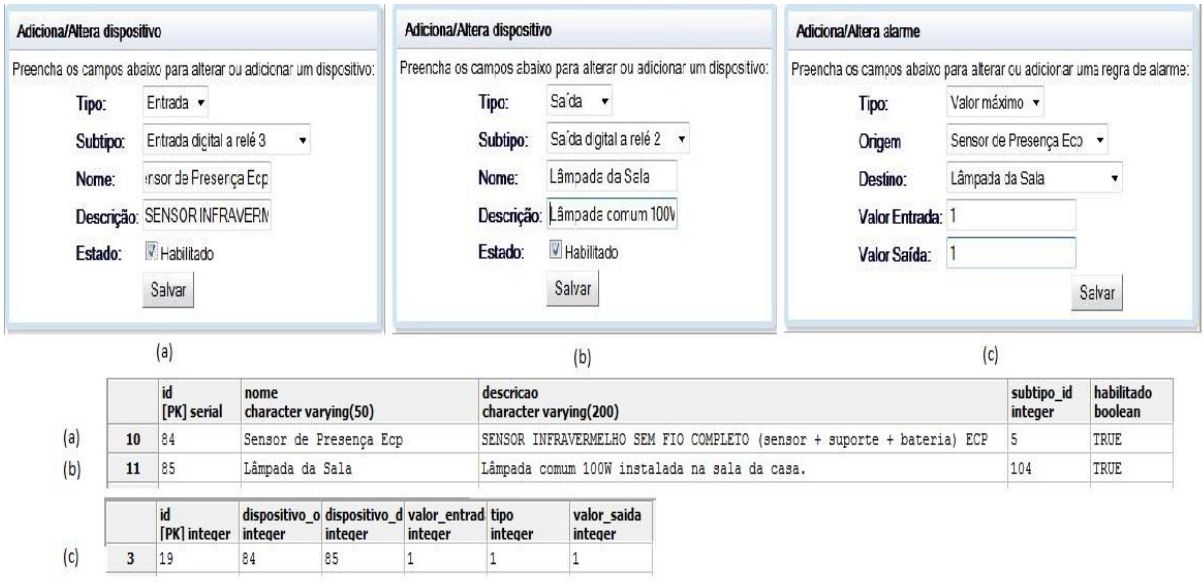
O gerenciamento de usuários foi testado cadastrando um novo usuário, iniciando uma sessão com a nova conta e alterando sua senha. Verificou-se através do banco de dados que o usuário foi cadastrado corretamente e depois sua senha alterada com sucesso, conforme demonstrado na figura 12. Também é possível observar que as senhas já criptografadas foram armazenadas no banco de dados, de forma que não se expõe a senha original do usuário.



		senha character varying	previlegi integer	id [PK] ser	usuario character var
Senha original	6	$2a$10$0tfqo1iYTq16nya4i2dKtuvctJ5bDICtMd2b3UvDAc//WiuI2pkX2	1	24	usuario
Senha alterada	6	$2a$10$/9slECOvLBkvT.czscXlAu8dhoOIzJ4cxUl2OlbY2jczuICdABnJa	1	24	usuario

Figura 12: Teste de Gerenciamento de Usuário
Fonte: Autoria Própria

O gerenciamento de dispositivos e alarmes foi testado cadastrando um sensor de presença em uma entrada digital à relé e uma lâmpada a uma saída digital à relé. Foi cadastrado também um alarme, onde a lâmpada deve ser acionada quando o sensor enviar um sinal de presença. Os resultados foram visualizados através do banco de dados, conforme a figura 13, onde o item (a) representa o sensor de presença, o item (b) representa a lâmpada e o item (c) representa o alarme.



	id [PK] serial	nome character varying(50)	descricao character varying(200)	subtipo_id integer	habilitado boolean
(a) 10	84	Sensor de Presença Ecp	SENSOR INFRAVERMELHO SEM FIO COMPLETO (sensor + suporte + bateria) ECP	5	TRUE
(b) 11	85	Lâmpada da Sala	Lâmpada comum 100W instalada na sala da casa.	104	TRUE

	id [PK] integer	dispositivo_o integer	dispositivo_d integer	valor_entrad integer	tipo integer	valor_saida integer
(c) 3	19	84	85	1	1	1

Figura 13: Teste de Gerenciamento de Dispositivos e Alarmes
Fonte: Autoria Própria

Por fim, a comunicação entre o sistema de supervisão e o sistema de controle e o funcionamento do supervisório foram testados enviando um comando para o Arduino a partir da tela de supervisão, que resultou no comportamento esperado. É possível observar o resultado pelo campo `User-Agent` do cabeçalho da requisição, destacado na figura 14, cujo conteúdo é o nome do cliente HTTP, no caso a linguagem Java, utilizada no sistema de supervisão.

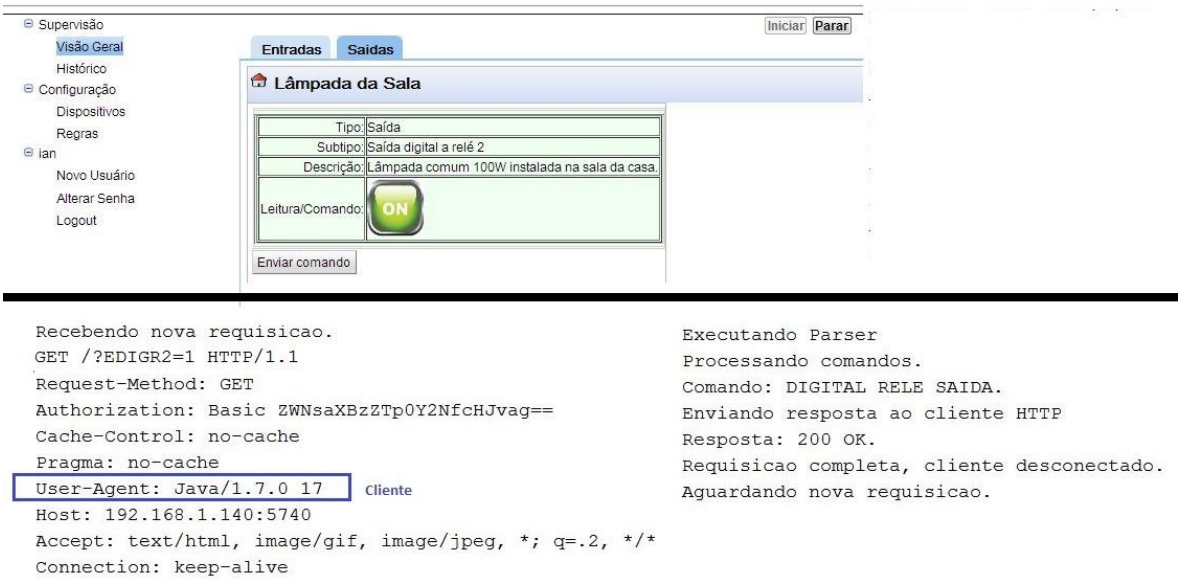


Figura 14: Teste de Comunicação Entre Sistemas
Fonte: Autoria Prórpia

Esse mesmo teste foi realizado nos três navegadores especificados nos requisitos do sistema: *Internet Explorer*, *Firefox* e *Chrome*. Com isso verificou-se que o sistema de supervisão possui compatibilidade com todos eles.

Para verificar a estabilidade do sistema, foi aplicado um teste exaustivo, onde o sistema de supervisão ficou em funcionamento por um período de 2 horas, fazendo a leitura contínua dos dispositivos ligados ao sistema de controle e verificando as condições dos alarmes. Nesse período não ocorreu nenhum erro de comunicação e os sistemas se comportaram como esperado.

5 CONCLUSÃO

Com os resultados obtidos foi possível afirmar que o sistema de supervisão e o sistema de controle oferecem as funcionalidades necessárias para o controle residencial via um navegador *web* de forma satisfatória, cumprindo os requisitos propostos.

Conforme pôde ser observado pelos testes realizados no sistema de controle, foi possível utilizá-lo de forma independente do sistema de supervisão. Isso garantiu que o projeto ficasse mais flexível, pois o controle dos dispositivos pode ser feito pela rede local doméstica, caso o acesso à internet esteja indisponível.

Para o desenvolvimento do sistema de supervisão foi essencial o uso do *framework* GWT. A princípio não havia muito conhecimento sobre essa tecnologia, portanto, foi necessário dedicar um período de pesquisa sobre seu funcionamento e suas ferramentas. Devido a esse fator, o progresso do desenvolvimento da aplicação deu-se de forma lenta no início, porém, após certo domínio da tecnologia, o processo se tornou mais eficaz.

Alguns problemas foram identificados na implementação do projeto, como a variação dos valores de tensão nas portas analógicas da placa de conversão de sinais e valores de tensão flutuantes lidos nas entradas analógicas sem dispositivos conectados, isso pode causar interferência no controle do supervisório e gerar alarmes indevidos.

Com a utilização somente de tecnologias abertas e gratuitas, as despesas no desenvolvimento deste projeto foram reduzidas a apenas o custo dos componentes de *hardware,* o que viabilizou a utilização do sistema para implantações de pequeno porte sem a necessidade de grandes investimentos, resultando em um beneficio bom em relação aos custos de implantação.

Melhorias podem ser feitas no projeto de forma a aumentar sua escalabilidade e seu desempenho:

- Ajustes na placa de conversão de sinais com a finalidade de eliminar as variações e flutuações nos sinais;
- Implementar novas funcionalidades no sistema de supervisão, como geração de relatórios e gráficos;

- Aprimorar a interface gráfica do sistema de supervisão, facilitando a visualização do usuário;
- Implementar uma IHM para facilitar o gerenciamento do sistema de controle;
- Estudar novas tecnologias que possam ser aplicadas no proposito deste projeto, superando as restrições existentes nas tecnologias atuais.

REFERÊNCIAS

BRETERNITZ, Vivaldo José. **Domótica**: as casas inteligentes. jun. 2001. Disponível em: <http://www.widebiz.com.br/gente/vivaldo/domotica.html>. Acesso em 20 mar. 2013

FITZGERALD, Scott. **Arduino *Ethernet Shield***. Disponível em: <http://www.arduino.cc/en/Main/ArduinoEthernetShield>. Acesso em 02 dez. 2012.

GARRETT Jesse J. **Ajax: *A New Approach to Web Applications***. *Adaptive Path*. San Francisco, 2005

HANSON, Robert; TACY, Adam. **GWT *In Action: Easy Ajax with the Google Web Toolkit.*** *Manning Publications* Co. 2007

HIPPEL, Von E; KROGH, Von G. ***Open source software and the "private-collective" innovation model: issues for organization science.*** *Organization Science*, v. 14, n. 2, 2003.

KRIPPENDORF, Klaus. ***Der verschwundene Bote Metaphern und Modelle der Kommunikation*** in MERTEN, Klaus et al (Hrsg): *Die Wirklichkeit der Medien*. Opladen: Westdt. Verlag. 1994.

KRISHNAMURTHY Balachander; REXFORD Jennifer. ***Web protocols and practice: HTTP/1.1, networking protocols, caching and traffic measurement***. Addison-Wesley, 2001

MARDEGAN, Ronaldo; AZEVEDO, Rodrigo C.; OLIVEIRA, João F. G. de. **Os Benefícios da Coleta Automática de Dados do Chão-de-Fábrica para o Processo de Negócio Gestão da Demanda**. In: ENCONTRO NACIONAL DE ENGENHARIA DE PRODUÇÃO. Curitiba, 2002,

MATIC, Nebojsa; ANDRIC, Dragan. **Microcontroladores PIC.** mikroElektronika, 2000.

MELLIS, David A.. **Arduino *Board* Mega**. Disponível em: <http://arduino.cc/en/Main/ArduinoBoardMega>. Acesso em 02 dez. 2012.

RUBOW, Erik. ***Open Source Hardware*.** 2008. Disponível em: <http://cseweb.ucsd.edu/classes/fa08/cse237a/topicresearch/erubow_tr_report.pdf>. Acesso em 01 dez. 2012.

SCHNEIDER Fabian; AGARWAL Sachin; ALPCAN Tansu; FELDMANN Anja. ***The New Web: Characterizing AJAX Traffic****. Proceedings of the 9th international conference on Passive and active network measurement*. Universidade Técnica de Berlim. Berlim, 2008.

SILVA, Ana Paula G. da; SALVADOR, Marcelo. O que são Sistemas Supervisórios?.**WECTRUS**. dez. 2005.

SILVA, Danise S. da. **Desenvolvimento e Implementação de um Sistema de Supervisão e Controle Resindencial.** Dissertação (Mestrado) – Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal do Rio Grande do Norte. Natal, 2009

SMEETS Bram; BONESS Uri; BANKRAS Roald. ***Beginning Google Web Toolkit: From Novice to Professional***. Apress, 2008

SOARES, Luis Fernando G.; **Redes de Computadores: das LANs, MANs e WANs às Redes ATM.** 2. ed. Campus, 1995.

SOUZA, Rodrigo B. de. **Uma Arquitetura para Sistemas Supervisórios Industriais e sua Aplicação em Processos de Elevação Artificial de Petróleo**. Dissertação (Mestrado) – Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal do Rio Grande do Norte. Natal, 2005.

SUPERGEON, Charles E. ***Ethernet: The Definitive Guide*.** Sebastopol, California: O'Reilly, 2000.

TANENBAUM, Andrew S. **Redes de Computadores**. 4. ed. Amsterdam, Holanda: Campus, 2003.

VENTURI, Eli. **Protótipo de um Sistema para Controle e Monitoração Residencial através de Dispositivos Móveis Utilizando a Plataforma .Net.** Trabalho de Conclusão de Curso (Bacharelado). Universidade Regional de Blumenau, 2005

VIANNA, Willian da S; **Sistema SCADA Supervisório**, Campos dos Goytacazes, Rio de Janeiro. Instituto Federal Fluminense, 2008.

APÊNDICE A – Diagramas da Modelagem dos Sistemas.

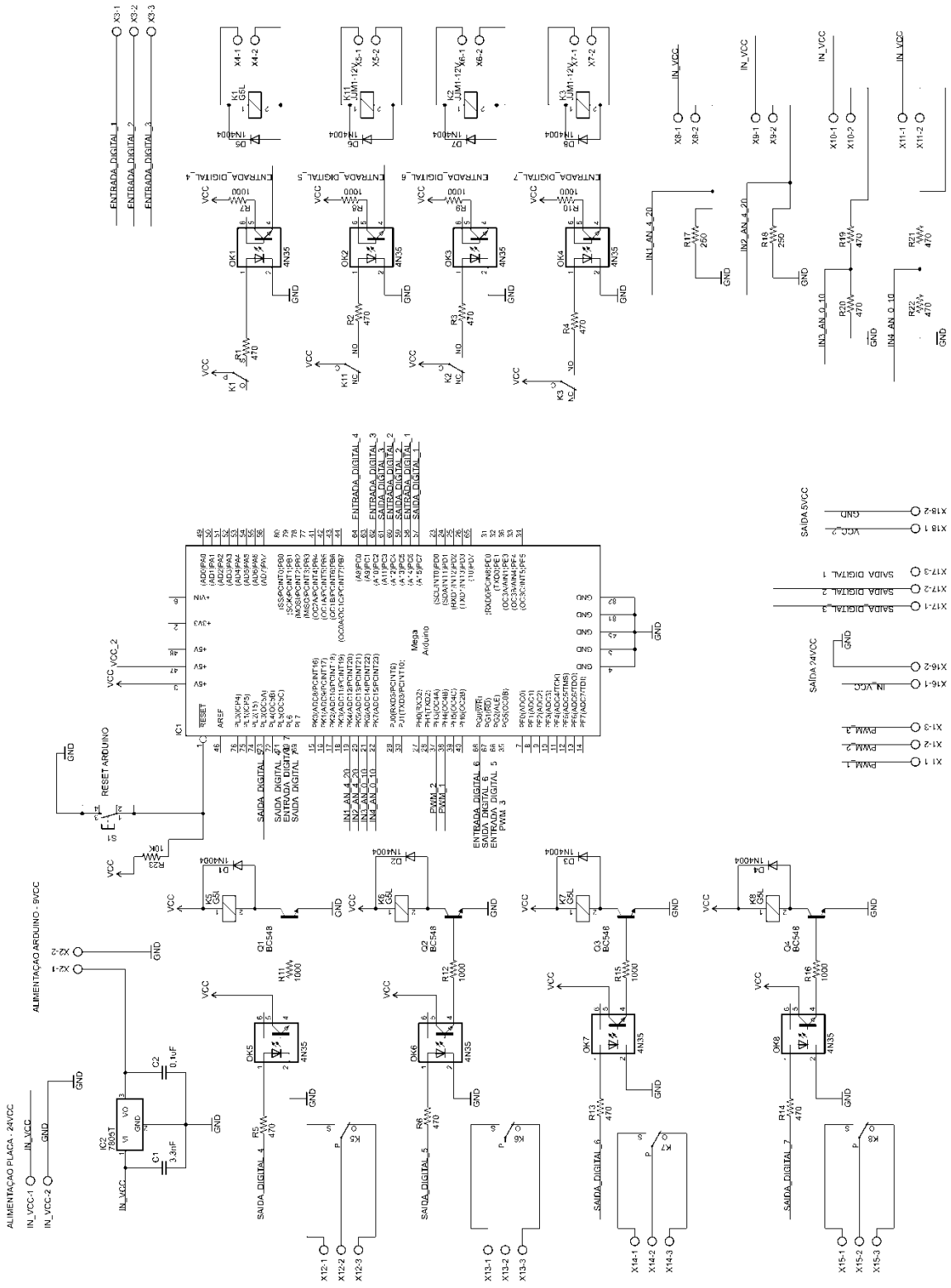


Figura 15: Diagrama Esquemático da Placa de Conversão de Sinais
Fonte: Autoria Própria

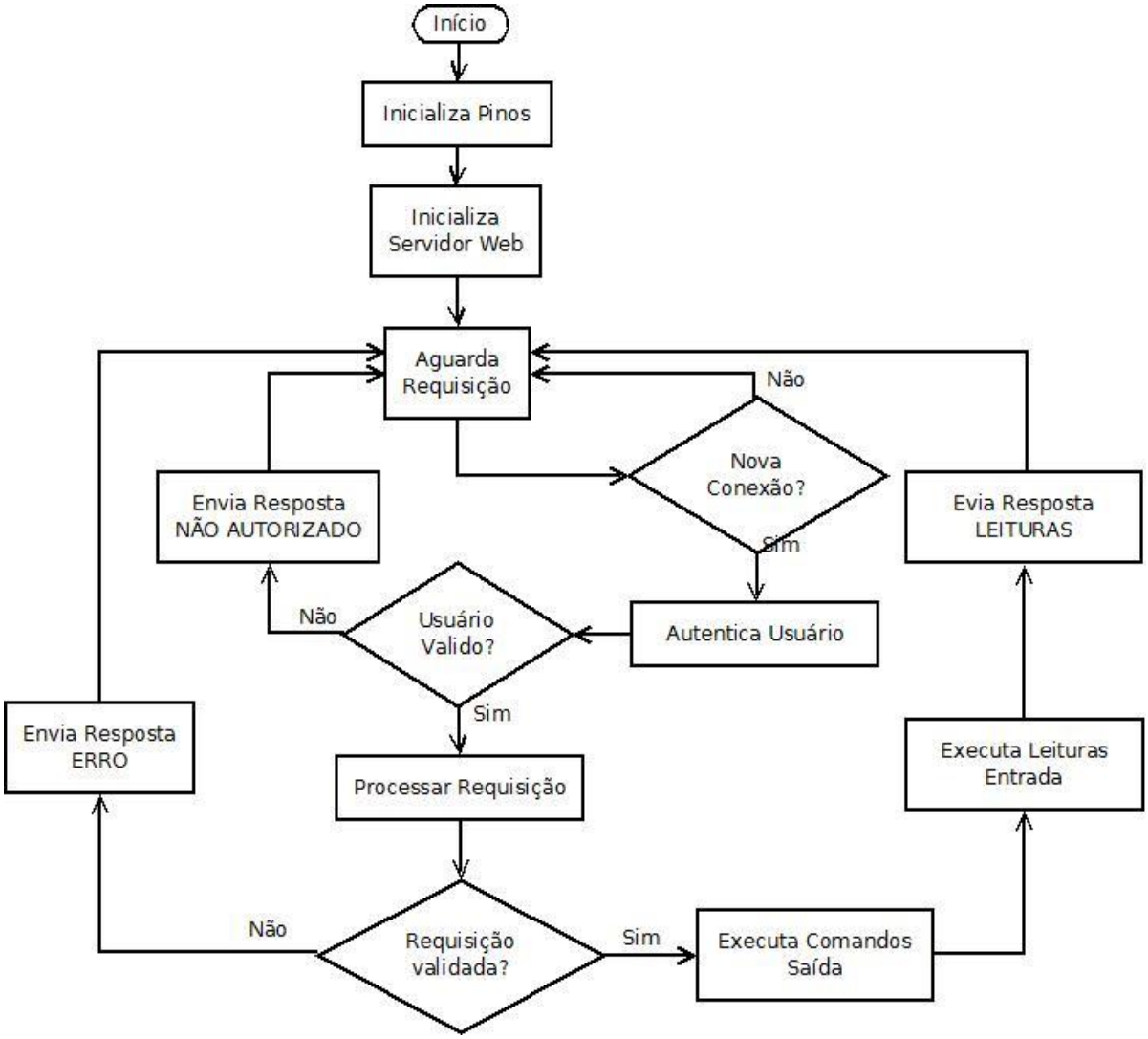


Figura 16: Fluxograma do Programa do Arduino
Fonte: Autoria Própria

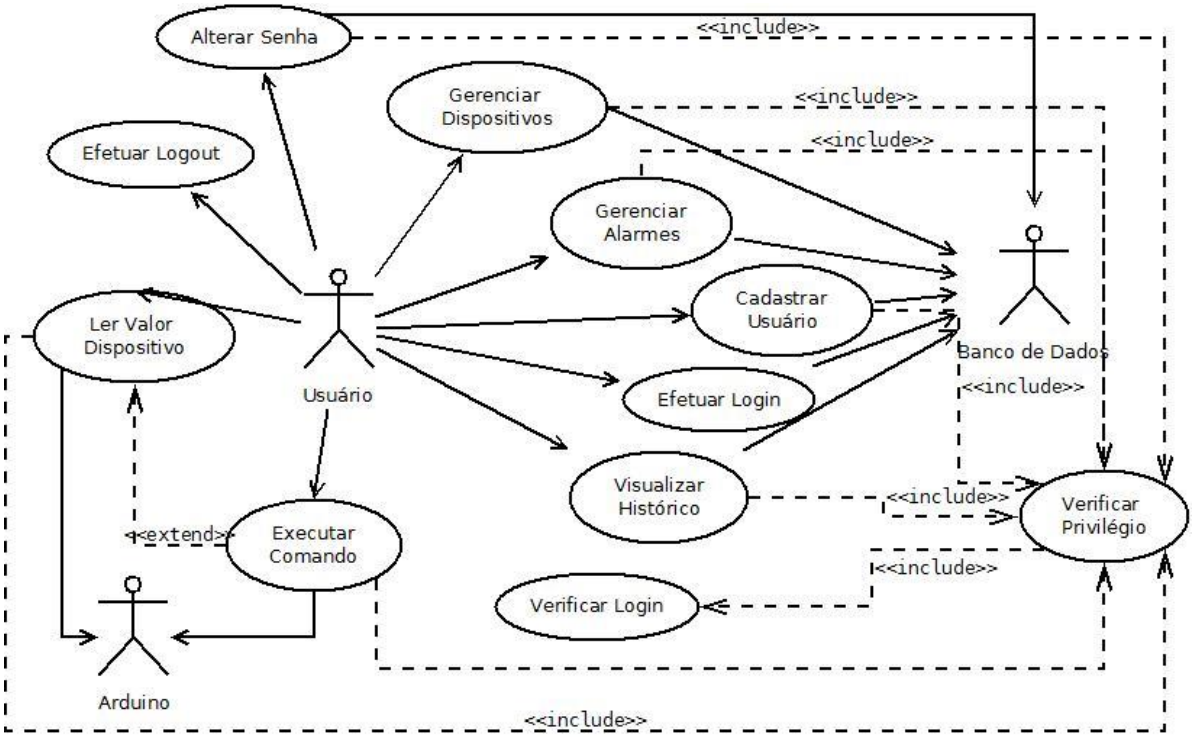


Figura 17: Diagrama de Casos de Uso
Fonte: Autoria Própria

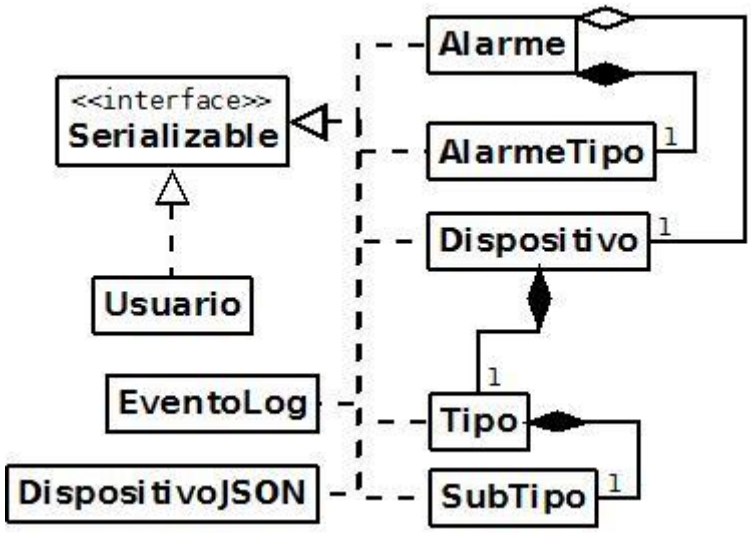


Figura 18: Diagrama de Classes Simplificado – Dados
Fonte: Autoria Própria

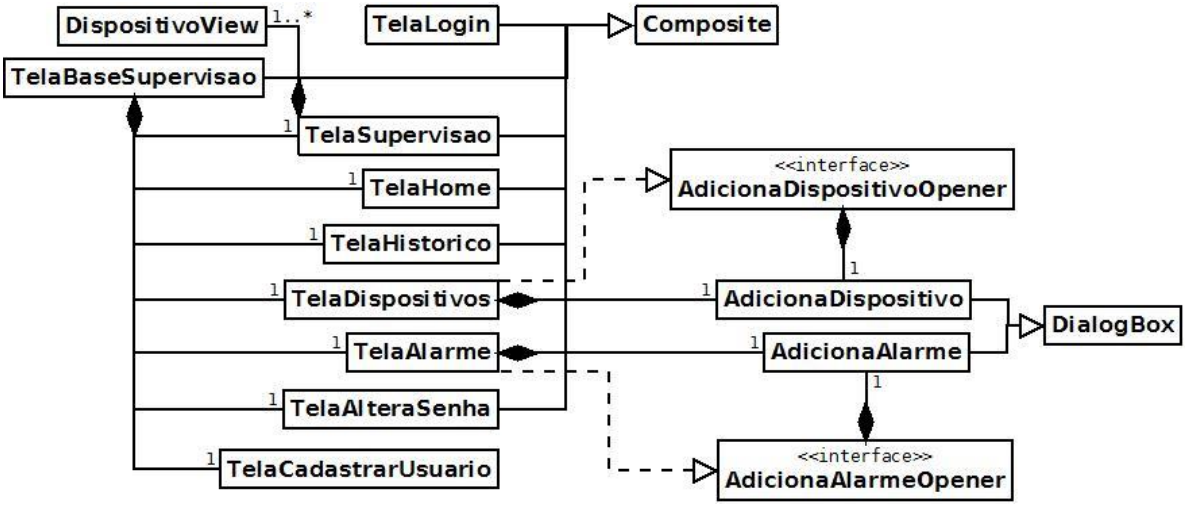


Figura 19: Diagrama de Classes Simplificado – Telas
Fonte: Autoria Própria

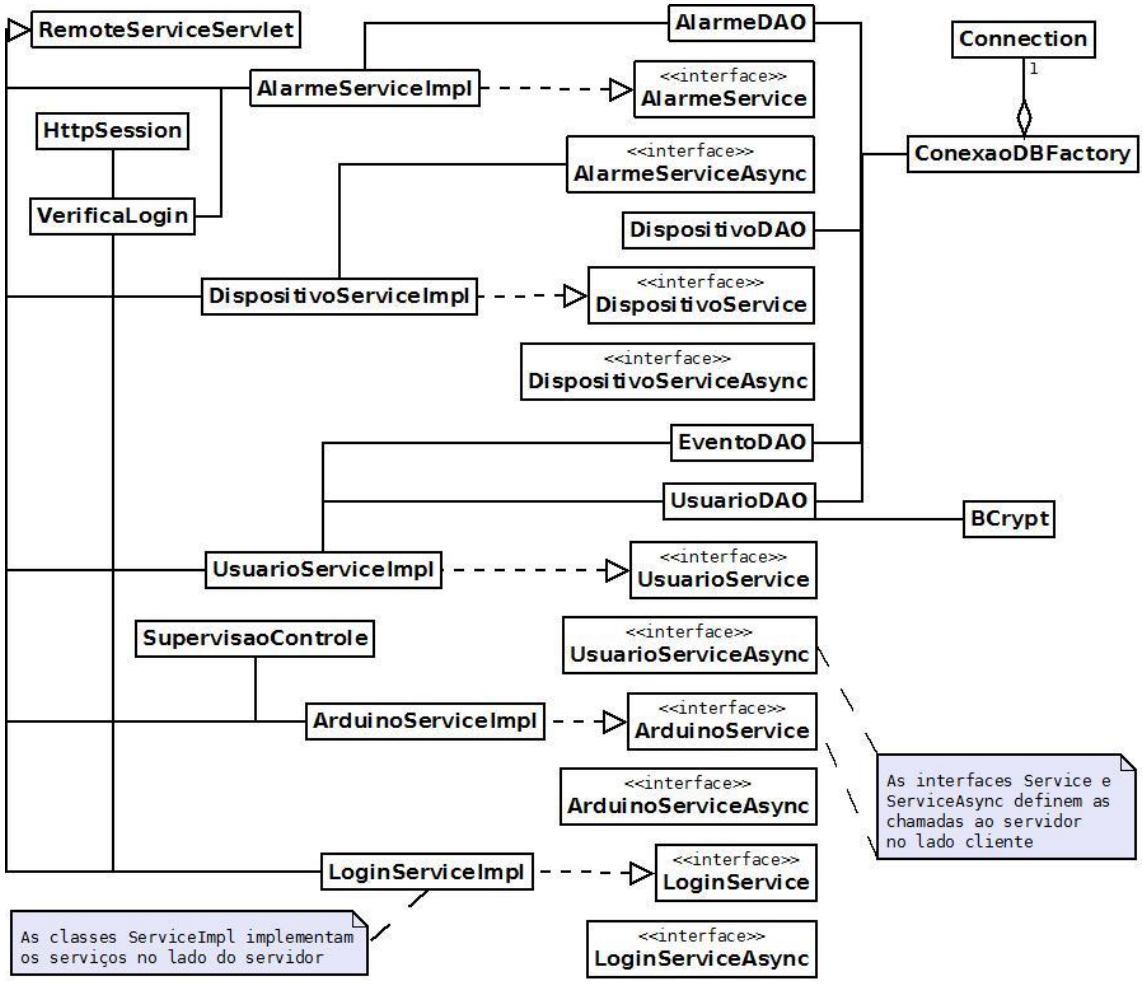


Figura 20: Diagrama de Classes Simplificado – Lógica
Fonte: Autoria Própria

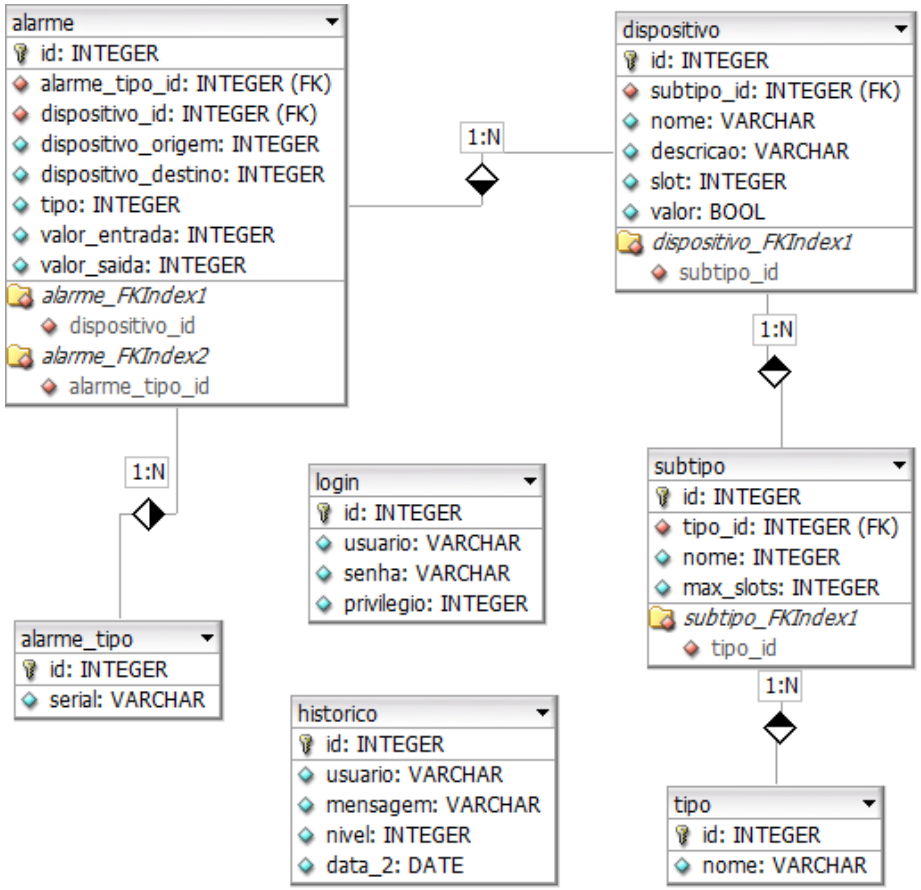


Figura 21: Diagrama Entidade Relacionamento do banco de dados
Fonte: Autoria Própria

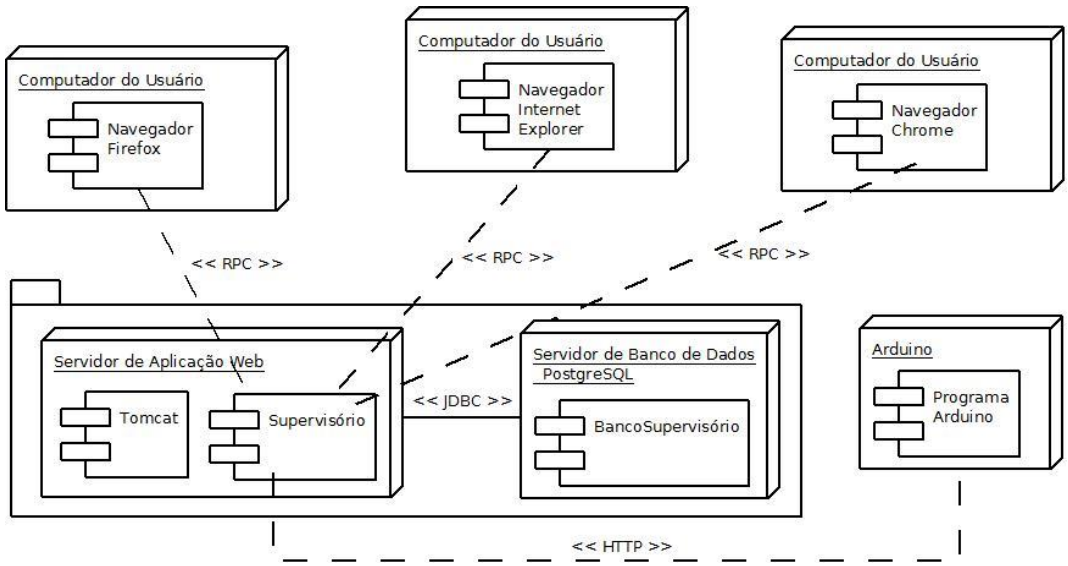


Figura 22: Diagrama de Implantação
Fonte: Autoria Própria