CENTRO UNIVERSITÁRIO DE BRASÍLIA -UniCEUB

CURSO DE ENGENHARIA DE COMPUTAÇÃO

ALCIDES RAFAEL ZAGO BELEM

SISTEMA DE MONITORAMENTO DE TEMPERATURA E UMIDADE EM AVIÁRIOS

Orientadora: M.C. Maria Marony Sousa Farias Nascimento

Brasília

julho, 2011

ALCIDES RAFAEL ZAGO BELEM

SISTEMA DE MONITORAMENTO DE TEMPERATURA E UMIDADE EM AVIÁRIOS

Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB) como pré-requisito para a obtenção de Certificado de Conclusão de Curso de Engenharia de Computação.
Orientadora: M.C. Maria Marony Sousa Farias Nascimento.

Este Trabalho foi julgado adequado para a obtenção do Título de Engenheiro de Computação, e aprovado em sua forma final pela Faculdade de Tecnologia e Ciências Sociais Aplicadas - FATECS.

_____

Prof. Abiezer Amarilia Fernandez
Coordenador do Curso

Banca Examinadora:

_____

Prof. Maria Marony Souza Farias de Nascimento, Mestre em Engenharia Elétrica
Orientadora

_____

Prof. Luis Claudio Lopes de Araujo, Mestre em Matemática
Instituição

_____

Prof. Karin Astrid Marques dos Santos, Doutora em Química
Instituição

ALCIDES RAFAEL ZAGO BELEM

SISTEMA DE MONITORAMENTO DE TEMPERATURA E UMIDADE EM AVIÁRIOS

Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB) como pré-requisito para a obtenção de Certificado de Conclusão de Curso de Engenharia de Computação.
Orientadora: M.C. Maria Marony Sousa Farias Nascimento.

Brasília

julho, 2011

AGRADECIMENTOS

Agradeço à minha família, que sempre me incentivaram para a conclusão de meu ensino superior. Aos meus amigos e namorada pela compreensão, estímulo, paciência e incentivo para a concretização deste objetivo.

Agradeço também à orientadora Maria Marony pelo auxilio na elaboração do projeto e de igual parcela a todos os professores do curso de Engenharia da Computação do UniCEUB. Lembrar também das contribuições dos amigos Gabriel Santos, Gustavo Fontana, Jean Matheus e Rafael de Mello.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE DIAGRAMAS

RESUMO

Este projeto apresenta uma proposta de desenvolvimento de um sistema microcontrolador para monitoração da temperatura e umidade relativa dentro de um aviário utilizando um sensor de umidade e temperatura para a tomada de dados. Para isso, um protótipo foi desenvolvido com o objetivo, de, através das medições do sensor de umidade e temperatura, obter-se dados suficientes para realização de um histórico completo da variação térmica dentro do aviário ao longo dos dias. O sensor utilizado no projeto é o SHT15 (sensor de umidade e temperatura). O sensor do modelo SHT15 (sensor de umidade e temperatura) realiza a tomada de dados no ambiente do aviário. O ‘Arduino Uno’ é o microcontrolador utilizado para interagir entre o computador e o sensor de temperatura realizando a sincronização entre os mesmos. A plataforma de desenvolvimento JAVA é responsável para a leitura da porta serial onde o micro controlador estará emitindo os dados do sensor de temperatura, e também está responsável pela elaboração dos relatórios e históricos do aviário. Com o desenvolvimento de todos os componentes citados é possível à elaboração dos relatórios do aviário. Os resultados apresentados são os relatórios e históricos de quais as temperaturas e umidades que o aviário passou ao longo dos dias.

Palavras Chave: Sensor de temperatura e umidade, aviário.

ABSTRACT

This project presents a proposal to develop a microcontroller system for monitoring the temperature and humidity inside an aviary using a humidity sensor and temperature for the data taking. For this, a prototype was developed with the aim of, through the measurements of the humidity sensor and temperature, to obtain sufficient data to conduct a full history of thermal variation in the aviary during the day. The sensor used in the project is SHT15 (temperature and humidity sensor). The sensor model SHT15 (temperature and humidity sensor) performs the data taking in the environment of the aviary. The 'Arduino Uno'is the microcontroller used to interact with the computer and temperature sensor performing the synchronization between them. JAVA development platform is responsible for reading the serial port where the microcontroller will be sending the data from the temperature sensor, and is also responsible for reporting and historical roadway. With the development of all of these components is possible for the reporting activities of the aviary. The results are the reports and records of which the temperatures and humidities that the avian went over the day.

CAPÍTULO 1 – INTRODUÇÃO

1.1 – Motivação e Posicionamento

A motivação para a realização do projeto veio, primeiramente, da necessidade que os granjeiros têm de obterem informações sobre as condições as quais o aviário está submetido na ausência destes.

A ideia do projeto surgiu quando foi realizada uma entrevista com o proprietário da Granja Pérola do Sul localizada no paranoá para a avaliação da problemática. Foi mencionado pelo Sr. Ademir Xavier de Castro de que dependendo das condições de temperatura e umidade em que a ave é submetida, o crescimento dela é mais proveitoso ou não. De acordo com a idade das aves, elas devem ser submetidas às diferentes temperaturas e umidades que são aspectos extremamente importantes e que influenciam diretamente no crescimento saudável da ave. O sistema é uma demanda para o empreendimento, pois o proprietário precisa saber sobre qual temperatura e umidade o aviário está sendo submetido ao longo do dia, e poder tomar medidas para que a temperatura e umidade esteja sempre em condições perfeitas para a ave.

Manter a temperatura ideal é primordial também para que as aves não corram risco, pois em casos emergenciais e falhas consecutivas nos sistemas de resfriamento do aviário podem colocar em risco a vida de todas as aves. Há relatos de que meia hora com os sistemas de resfriamento desligados, a temperatura pode se elevar a ponto de deixar a vida das aves insustentável dentro do aviário.

O sistema precisa realizar o monitoramento para alertar o granjeiro de situações de risco para as aves, saber se o aviário está com a temperatura muito elevada e notificar o responsável da situação crítica da qual as aves estão sendo submetidas. O sistema necessita ficar sempre alerta e notificar tanto o granjeiro que pode não estar na granja no momento quanto o funcionário responsável que pode por algum motivo não ter acesso ao estado do aviário. Assim uma sirene seria disparada a fim de alertar os funcionários ao redor indicando que algo crítico está acontecendo.

1.2 – Visão Geral do Projeto

O projeto monitora e alerta em casos críticos a situação de um aviário, tomando dados da temperatura e a umidade relativa interna do aviário. A figura 1.2 ilustra a topologia do sistema empregado, mostrando os equipamentos utilizados, e seus cabeamentos.

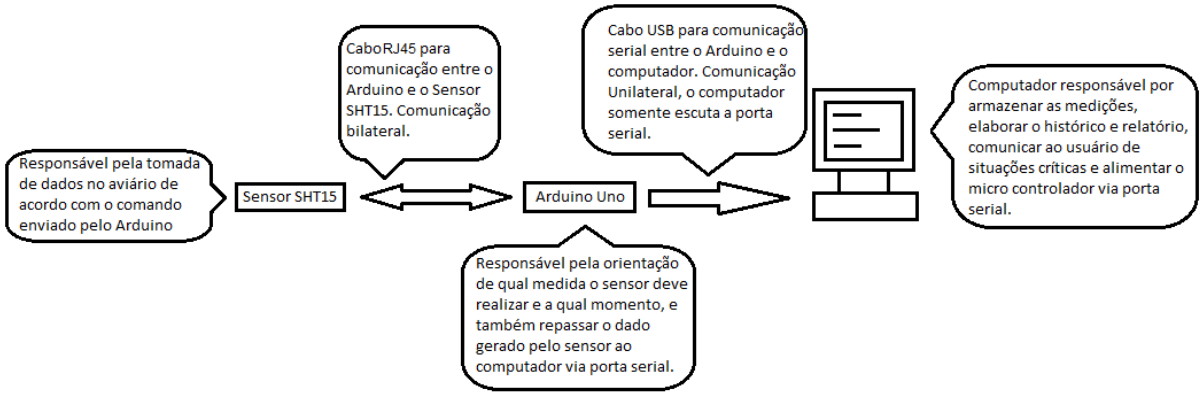


Figura 1.2: Visão Geral do Projeto

1.3 – Objetivos do Trabalho

O objetivo geral deste trabalho é o de apresentar um protótipo do dispositivo eletrônico que é responsável por medir a temperatura e umidade relativa interna do aviário e salvar as informações no computador, assim como gerar relatórios e históricos do ambiente do aviário.

Para isto algumas tarefas precisam ser executadas, tais como:

- Desenvolver um código no micro controlador ‘Arduino Uno’ para receber a leitura do sensor de umidade e temperatura SHT15 e emitir essas leituras até a porta serial do computador;

- Elaborar meio de comunicação entre o micro controlador e o sensor de temperatura de modo que o sensor possa ficar a uma distância aproximada de 50 metros;

- Elaborar um aplicativo em JAVA que monitore a porta serial para capturar os dados emitidos pelo micro controlador ‘Arduino Uno’ e, salve esses dados em um banco de dados, transformando-o em informação na forma de relatórios e históricos do aviário.

1.4 – Estrutura da Monografia

Além deste capítulo introdutório, esta monografia está estruturada em mais quatro capítulos e organizada do seguinte modo:

- Capítulo 2 – Apresentação do Problema. Neste capítulo é apresentado em detalhes o problema a ser tratado, em todo seu contexto, como são tratadas atualmente, soluções existentes, dificuldades encontradas, metodologias, justificando como a proposta aqui apresentada dará uma solução para o problema.
- Capítulo 3 – Referencial Teórico e Tecnológico – Nesse capítulo é apresentado o referencial teórico e tecnológico que embasa o projeto. A priori, ele trata do sensor de umidade e temperatura SHT15 utilizado no protótipo. Em seguida apresenta uma visão geral sobre o micro controlador ‘Arduino’ e sobre as classes JAVA utilizadas para o desenvolvimento do software e a comunicação serial com o micro controlador.
- Capítulo 4 – Desenvolvimento do Projeto – Este capítulo trata do desenvolvimento e a visão do projeto. Este capítulo também especifica as questões de hardware e software que fizeram parte do projeto assim como os testes realizados, e a simulação e resultados.
- Capítulo 5 – Testes – Resultados de testes realizados para a aplicação, hardware. Além de opinião do usuário final.
- Capítulo 6 – Conclusão – Esse capítulo trata especificamente o final do projeto com suas conclusões e apresentando propostas para projetos futuros.

1.5 – Resultados Esperados

Como resultado do projeto desenvolvido espera-se que o sistema de controle do aviário detecte as variações de temperatura e umidade. Elabore relatórios e gráficos aparti das medições de temperatura e umidade.

CAPÍTULO 2 – CENÁRIO ATUAL DAS TECNOLOGIAS EM AVIÁRIOS

Os aviários em geral contem seus próprios equipamentos para o acompanhamento da umidade e temperatura, além de mecanismos de acionamento de ventiladores e cortinas automáticos. Onde se tem um medidor de temperatura digital localizado dentro do aviário, esse medidor pode ser programado com temperaturas máximas e mínimas, o mesmo é ligado diretamento no quadro de força dos ventiladores acionando ou desligando a medida em que a temperatura sobe ou desce. As cortinas são ligadas também as esse quadro de força, no momento em que a temperatura fica acima do programado no medidor digital, as cortinas são liberadas abrindo o aviário.

Os medidores digitais em geral permitem apenas uma programação simples onde se escolhe a temperatura máxima e mínima juntamento com a umidade máxima e mínima, esses mostrando os dados em um display de sete segmentos.

2.1 – Sistemas de controle existentes

Os sistemas de controle variam de acordo com o fabricante e com as demandas apresentadas pelo granjeiro.

O fabricante mais conhecido no Brasil é a Full Gaude com o sistema Sitrad. Onde permite o controle do aviário e elaboração de relatórios do mesmo. Contando com diversos recursos Web.(http://www.sitrad.com.br/index.asp)

Vários fabricantes distribuem os seus sistemas de controle diferenciando cada um dos sistemas entre versões Locais, onde operam apenas no computador local, entre versões Web onde os dados são disponibilizados via Web e versões móbile que algumas funcionalidades são disponibilizadas para o usuário final através de seu aparelho de celular.

2.2 – Softwares de Gerenciamento Existentes

Como exemplo o Sitrad é um software de gerenciamento existente no mercado onde Avalia, controla e armazena dados de temperatura, umidade, tempo, pressão. Faz análise do histórico através de gráficos e relatórios, gerados a partir dos dados armazenados. Permite a alteração dos parâmetros de controle dos instrumentos, tais como limites máximo e mínimo de temperatura, tempo de processo, entre outros. A figura 2.2 ilustra a tela de monitoramento do sistema Sitrad.

Figura 2.2: Software Gerenciamento Sitrad.

Até o momento não se conhece um programa Open Source (Software Livre) que gerencie e execute as funcionalidades citadas.

Para se obter um software de gerenciamento de Aviário também se faz necessário a aquisição de sensores de temperatura e umidade, além de meios de comunicação entre os dispositivos, um computador, e equipamentos agregados. Geralmente os fabricantes de software de gerenciamento oferecem junto o hardware necessário que funciona especialmente para o seu software, e vice-versa. Amarrando assim a aquisição do software com a aquisição do hardware.

CAPÍTULO 3 – REFERENCIAL TECNOLÓGICO

O projeto desenvolvido utiliza diversas ferramentas com foco em seu objetivo, são essas ferramentas de linguagem de programação, barramento USB, microcontrolador, sensores de ambiente além de cabos e conectores. Cada um dos componentes e ferramentas utilizadas são aplicadas e utilizadas de modo específicos almejando o objetivo do projeto.

3.1 – Linguagem de Programação

A aplicação que ficou encarregada de controlar, gerenciar e se comunicar com usuário requerem domínios de métodos e técnicas da área de linguagens de programação. As informações que são capturadas pelos sensores, interpretadas pelo microcontrolador e disponibilizadas pelo computador necessitam de entendimento avançado de linguagem de programação para o entendimento aprofundado do sistema.

Através das diversas linguagens de programação o computador é capaz de interpretar comandos assim executando diversas tarefas. Entendem-se os comandos como algoritmos, onde se pode escrevê-los através das diversas linguagens existentes. Centenas de linguagens de computador estão atualmente em uso. Essas linguagens podem ser divididas em três tipos gerais; Linguagens de Máquina, linguagem *assembler* e linguagem de alto nível (DEITEL JAVA COMO PROGRAMAR 4ª EDIÇÃO). As linguagens de máquinas são compostas somente de *bits,* ou seja, apenas 0s e 1s. Já a linguagem *assembler* utiliza de abreviações para indicar operações computacionais para a máquina. Assim deixando a linguagem mais compreensível ao ser humano, e também, necessitando de um tradutor para a linguagem de máquina, a linguagem de alto nível deixou o ser humano ainda mais próximo da máquina, com uma linguagem de fácil compreensão mas ainda necessitando de um tradutor para a linguagem de máquina. Linguagens utilizadas no projeto como a linguagem Java e a Linguagem de programação do Arduino baseada em Wiring são exemplos de linguagens de alto nível.

3.1.1 – Algoritmo

Um algoritmo, intuitivamente, é uma sequência finita de instruções ou operações básicas (operações definidas sem ambiguidade e executáveis em tempo finito dispondo-se apenas de lápis e papel) cuja execução, em tempo finito, resolve um problema computacional, qualquer que seja a sua instância. (DIRCEU DOUGLAS SALVETTI, 1998).

Os algoritmos devem ser consistentes, válidos e objetivos, sempre tentando simplificar o problema e executar a tarefa da maneira correta. O algoritmo é uma sequência de instruções da linguagem de programação aplicada almejando um objetivo.

Do ponto de vista lógico, um algoritmo é constituído por três estruturas lógicas: sequencial, repetitiva e seletiva. Isto é, um algoritmo pode ser constituído por qualquer combinação dessas três estruturas. (DIRCEU DOUGLAS SALVETTI, 1998). A partir dessas estruturas lógicas os algoritmos são desenvolvidos sobre as linguagens de programação Java e Linguagem de programação do Arduino.

3.1.2 – Linguagem de programação Java

A linguagem de programação Java foi escolhida devido ao conhecimento do autor em programação Java além de, diversas bibliotecas já criadas que são utilizadas no projeto, ter uma boa comunidade de desenvolvedores na *Web*, ser bastante difundida e principalmente por ser uma linguagem com elevado grau de portabilidade, herdando ao sistema, a possibilidade de instala-lo em diversos sistema operacionais, desde que tenham a *JVM* instalada na máquina. A figura.

Linguagens como Java são *orientadas a objetos* – o ato de programar em uma linguagem como essa se chama programação orientada a objetos (*OOP – object oriented programming*), e permite aos projetistas o projeto orientado a objetos como um sistema que funcione. (DEITEL JAVA COMO PROGRAMAR 4ª EDIÇÃO).

A linguagem Java tem crescido enormemente nos últimos anos, juntamente com o número de programadores que aderem à programação orientada a objetos. Com o crescimento

do número de programadores, eleva-se a participação de comunidades *Web,* recebendo atualizações constantemente.

O projeto utiliza da linguagem de programação Java na elaboração de classes para o sistema que interpreta os dados capturados pela porta USB, e também fica encarregado de salvar todos os dados no banco de dados. À medida que recepciona os dados, ele avalia as medidas e decide a situação em que o aviário se encontra.

Toda a parte de programação em Java é utilizada a IDE (Integrated Development Environment) de desenvolvimento *Eclipse.* É utilizada pela familiaridade da ferramenta pelo autor além da grande facilidade de aplicações de bibliotecas externas ao projeto e plug-ins que possibilitam a exportação fácil do projeto para um pacote único e executável. A figura 3.1.2 exibe a tela de inicialização da IDE Eclipse Helios utilizada para o desenvolvimento do software.

Figura 3.1.2: Imagem de inicialização da IDE Eclipse Helios

3.1.3 – Programação para Web

A área de programação para Web conta com diversas ferramentas que incorporam várias linguagens. A área do projeto responsável pela programação para a Web é um ponto crítico do projeto uma vez que fica responsável de demonstrar o resultado do projeto para o usuário.

Como ferramenta para o desenvolvimento de código JSP e para armazenar os códigos HTML e os organizar, foi utilizado a framework WebIntegrator fornecido pela ITX Information Technology em parceria com a Tecnisys.

WebIntegrator é uma ferramenta que facilita o desenvolvimento de aplicações para Web. Construído em Java, é multiplataforma e integra-se facilmente com banco de dados.

O ambiente para desenvolvimento requer apenas um browser para ser utilizado. Oferece funcionalidades de *workflow* e segurança nativa. (http://www.webintegrator.com.br/). A figura 3.1.3 ilustra o logotipo da framework WebIntegrator utilizada no projeto.

Figura 3.1.3: Logotipo da Framework WebIntegrator

3.1.4 – Apache Tomcat 6

Apache Tomcat é uma implementação de software open source do Java Servlet e JavaServer Pages (JSP). O Java Servlet e JavaServer Pages são desenvolvidas sob a Java Community Process. (http://tomcat.apache.org/)

O Apache Tomcat 6 é o servidor Web utilizado para realizar o deploy da aplicação Web desenvolvida. Ela é utilizada porque é uma ferramenta Open Source não trazendo mais custos para o projeto e também porque é de grande desempenho e estabilidade. Contem uma vasta comunidade de desenvolvimento e um grande número de documentações para o auxilio do desenvolvimento da aplicação Web.

O Apache Tomcat também oferece um log detalhado que é de grande suporte a erros a depuração de erros da aplicação e facilitando o desenvolvimento do mesmo e nas correções de eventuais erros de programação encontrados durante o desenvolvimento. A figura 3.1.4 exibe o logotipo do servidor web Apache Tomcat 6 utilizado.

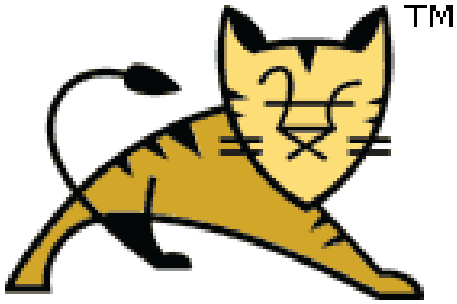


Figura 3.1.4: Logotipo Apache Tomcat

3.1.5 – Linguagem de programação do Arduino

A linguagem de programação do Arduino é uma linguagem própria do microcontrolador que é toda baseada em um ambiente de programação chamada Wiring. Sobre a linguagem de programação do Arduino que foi feito os algoritmos que são executados no microcontrolador.

Wiring é um ambiente de programação Open-Source com foco para a programação de eletrônicos, programação de entrada e saída de sinais, programação de computadores e prototipação de eletrônicos.(http://wiring.org.co/)

O ambiente de desenvolvimento do código do Arduino é todo baseado em Processing.

Processing é um ambiente de programação de código fonte aberto para escrever programas em outros computadores. Útil quando você quer que os outros computadores falem com um Arduino.(http://www.arduino.cc/playground/Interfacing/Processing)

Também é um ambiente para desenvolvedores que desejam criar imagens, animações e interações. Inicialmente desenvolvido para servir como um caderno de desenho de software e para ensinar os fundamentos da programação de computadores dentro de um contexto visual.(http://www.processing.org/)

O algoritmo a ser executado no microcontrolador é todo desenvolvido em Processing, uma linguagem de programação de alto nível, facilitando o desenvolvimento dos algoritmos e a manutenção do mesmo.

3.2 – Interface USB

O projeto desenvolvido em execução no computador capta os dados gerados pelo microcontrolador, interpreta-os e salva em um banco de dados. Para tal comunicação entre o computador e o microcontrolador foi escolhida a interface USB (Universal Serial Bus).

Universal Serial Bus (USB) é um padrão de barramento serial para conectar periféricos a um hospedeiro. Ele foi projetado para permitir que periféricos sejam conectados usando uma interface simples e padronizada e melhorar capacidades plug and play, permitindo que os periféricos a serem ligados ou desligados sem ter de reiniciar o computador ou desligar o dispositivo. (Ben Anton, 2009).

A interface USB foi escolhida pela grande popularidade nos computadores desktop e também notebooks onde grande maioria das máquinas contenha essa interface. Escolhida também por cauda da placa Arduino Uno conter apenas uma saída de dados que é a interface USB.

Conectores USB são projetados para facilidade de uso. Geralmente, o logotipo do tridente do conector deve estar voltado para cima quando conectar a uma porta. Os conectores são projetados para ser duráveis e de fácil inserção e remoção. Diferentes tipos de funções de servidor utilizam diferentes padrões de conectores. O padrão plug-A é frequentemente em cabos permanentemente ligado a dispositivos, como teclados de computador ou mouses. O plug-Padrão B tipicamente conecta dispositivos com cabos removíveis, como uma impressora. Conectores dispositivo USB evoluíram e se tornam menores como dispositivos eletrônicos de consumo tornaram-se também em tamanho menor. O conector padrão atual para pequenos dispositivos como celulares e câmeras é o conector Micro-B.(Ben Anton, 2009).

O Projeto utiliza um cabo USB com conectores do padrão A na conexão do computador e um conector do padrão B para a conexão com a placa Arduino Uno. A figura 3.2 exibe o modelo do cabo USB A/B que realiza a conexão entre o computador e o microcontrolador Arduino Uno.

Figura 3.2: Cabo USB modelo A/B

A interface USB é de fácil utilização, a sua característica “Plug and Play” permite que os dispositivos sejam conectados sem que seja necessário dar boot na máquina ou no periférico. Tornando o manuseio fácil até para usuários finais do sistema.

3.3 – Microcontrolador

O microcontrolador empenha um papel central no sistema, pois recebe a informação dos sensores, e interpreta as suas respostas. Além disso, envia esta respostas ao computador através da interface USB disponível na placa Arduino Uno.

Um microcontrolador é um sistema computacional completo, no qual está incluída uma CPU (*Central Processor Unit*), memória de dados e programa, um sistema de clock, portas de I/O (*Input/Output*), além de outros possíveis periféricos, tais como, módulos de temporização e conversores A/D entre outros, integrados em um mesmo componente. As partes integrantes de qualquer computador, e que também estão presentes, em menor escala, nos microcontroladores são: Unidade Central de Processamento (CPU), sistema de *clock* para dar sequência as atividades da CPU, memória para armazenamento de instruções e para manipulação de dados, entradas para interiorizar na CPU informações do mundo externo, saídas para exteriorizar informações processadas pela CPU para o mundo externo, programa (*firmware*) para definir um objetivo ao sistema.(Gustavo Weber Denardin)

A placa Arduino Uno utiliza um microprocessador ATmega8 que é responsável pelo processamento e execução dos algoritmos alojados em sua memória.

3.3.1 – Microcontrolador ATmega8

O microprocessador ATmega8, microprocessador de 8 bits desenvolvido pela empresa Admel(figura 4). Possui memória *EEPROM* de 512 Bytes, uma memória volátil *SRAM* de 1 Kbyte e uma memória *Flash* de 8 KByets. A figura 3.3.1 exibe os pinos do microcontrolador citado.

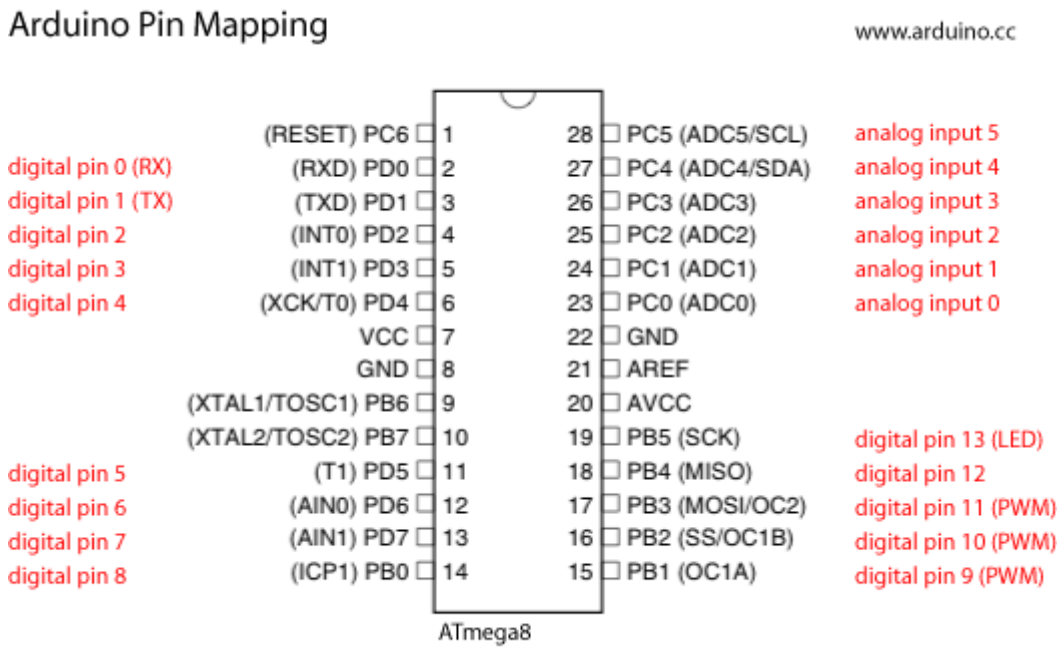


Figura 3.3.1: Configuração dos pinos do microcontrolador ATmega8. (Fonte Arduino.cc 2011)

3.3.2 – Placa Arduino Uno

A placa Arduino Uno foi escolhida pela fácil implementação, tamanho reduzido e pela interface de comunicação USB presente. Atendendo as demandas do projeto a um custo relativamente reduzido.

A placa Arduino Uno oferece uma IDE de desenvolvimento de códigos baseado no ambiente de programação Processing. A comunicação é bilateral realizada através da interface USB de comunicação por onde foram transferido os algoritmos desenvolvedor na Linguagem de programação do Arduino para a memória *Flash* do microprocessador ATmega8 da placa Arduino. Também da interface USB é utilizada a alimentação para suprir de força a placa Arduino e os seus componentes agregados.

A placa Arduino Uno, exibido na figura 3.3.2, e quem realiza a comunicação entre o computador e os sensores responsáveis pela tomada da temperatura.

Figura 3.3.2: Placa Arduino Uno. (Fonte: http://arduino.cc/en/Main/ArduinoboardUNo)

3.4 – Sensor SHT15

O SHT15 é o sensor escolhido para realizar as medições de umidade relativa e temperatura no meio. O sensor opera em baixa voltagem além de ser de ótima sensibilidade, tornando assim ideal para a implementação do projeto.

SHT15 sensor de umidade e temperatura é a versão final do sensor umidade com uma precisão de ponta de medição. Como cada tipo de sensor da família SHTxx, o sensor de umidade capacitivo é totalmente calibrado e fornece uma saída digital. Cada sensor é testado individualmente o cumprimento, qualidade e precisão. (http://www.sensirion.com/en/01_humidity_sensors/03_humidity_sensor_sht15.htm). O sensor SHT15 está ilustrado na figura 3.4 a seguir.

Figura 3.4: Sensor de temperatura e umidade SHT15

O projeto inclui quatro sensores SHT15 para obter maiores quantidades de medições nos aviários e melhorando a tomada de decisões do sistema.

3.4.1 – Soquete SHT15

O Soquete SHT15 produzido pela SparkFun.com, é uma placa de circuito impresso onde contem um sensor de umidade e temperatura SHT15 da fabricante Sensirion e todos os componentes e ligações necessárias para o funcionamento do sensor de umidade e temperatura.

A figura 3.4.1 mostra o desenho esquemático da placa SHT15 da SparkFun.com

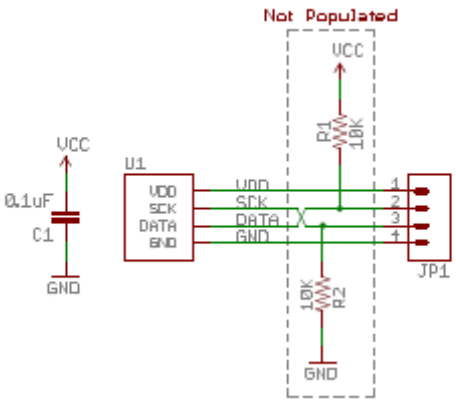


Figura 3.4.1: Desenho esquemático Placa SHT15 Sparkfun. (Fonte: http://sparkfun.com/datasheets/Sensors/Pressure/SHT1x-Breakout-v13.pdf))

A placa é de fácil implementação, onde é necessário que seja conectado dois pinos de dados do microcontrolador Arduino, um pino de alimentação vcc de cinco volts do Arduino, e também um pino de terra também localizado no Arduino.

Figura 3.4.1a: Placa SHT15 SpackFun. (Fonte: http:// www.sparkfun.com/products/8257)

3.5 – Banco de Dados

Um banco de dados típico é formado por uma ou mais tabelas. As *Tabelas*, por sua vez, são formadas por *Colunas* e *Linhas.* As colunas, também chamadas de campos, servem para dar nome ao espaço reservado a determinada informação. Já as linhas de uma tabela, também chamadas de *Registros* ou *Tuplas,* são os espaços reservados para a informação a ser armazenada. (JAVA2 E BANCO DE DADOS. Marco Aurélio Thompson).

O Banco de dados foi a forma escolhida para o armazenamento das informações obtidos pelo sistema. Características como a sigilosidade das informações, autenticidade, capacidade de realização de back-up e recuperação ágil das informações foram os fatores primordiais para a escolha desse método para o armazenamento dos dados.

Dentre os vários bancos de dados disponíveis no mercado, tanto dentre os bancos de dados pagos como o Oracle e o SQL Server quanto dos free, como o, HSQLDB, MySql, Postgrees e Firebird, o que apresenta maior compatibilidade e atende aos requisitos do projeto é o Firebird.

3.5.1 – Firebird

Firebird é um banco de dados relacional que oferece muitas características no padrão ANSI SQL que funciona em Linux, Windows e em uma variedade de plataformas Unix. Firebird oferece concorrência excelente, alta performance e linguagem poderosa que suporta stored procedures e triggers.

O Firebird é o banco de dados escolhido pela sua portabilidade onde ele possui drivers de conexão JDBC onde aplicações Java podem se comunicar independente da plataforma em que estejam rodando. Pela sua sigilosidade onde o banco de dados solicita uma senha de conexão para realizar consultas, e inserções de dados no banco. Também por poder comportar grandes volumes de dados, oferecer recursos de back-up e também por trabalhar com linguagem ANSI SQL facilitando nas elaborações de consultas e scripts do qual o autor já está familiarizado.

Um banco de dados totalmente Open Source é uma característica fundamental para a escolha do mesmo, uma vez que diminui drasticamente os custos operacionais.

Foi utilizada uma distribuição Embedded (Embarcado), caracteristica extremamente importante, onde o banco de dados é embarcado dentro da aplicação, tornando muito mais simples a implementação do sistema no computador do usuário final.

Esta versão é, na verdade, uma biblioteca cliente especial, que inclui o servidor em si. Quando um aplicativo chama essa biblioteca, ela carrega o servidor e permite acesso direto a qualquer banco de dados acessível ao computador local. Dessa forma, não faz uso do banco de dados de segurança. (http://www.firebirdsql.org/manual/pt_br/fbmetasecur-embedded-pt_br.html)

3.5.2 – JDBC JayBird

JayBird é um driver JDBC para se conectar ao servidor de banco de dados Firebird. Historicamente, a Borland abriu as fontes do tipo driver JDBC 3 chamado InterClient. No entanto, devido a algumas limitações inerentes a biblioteca Firebird Client foi visto que o driver tipo 3 era um beco sem saída, a equipe Firebird desenvolveu uma implementação pura em Java do protocolo wire. Esta implementação tornou a base para JayBird, um driver Java puro para o banco de dados relacional Firebird.(http://www.firebirdsql.org/index.php?op=devel&sub=jdbc&id=aboutjbird)

O Driver JDBC Jaybird foi escolhido por ser o driver mais recente e bem atualizado, possui uma exelente portabilidade, sendo feito puramente em java.

Através da classe java.sql.Connection pode realizar uma conexão da sua aplicação fisicamente com o banco de dados. Assim utilizando a classe java.sql.Statement podemos executar quaisquer instruções SQL de forma simples e podendo armazenar o resultado no objeto da classe java.sql.ResultSet onde fica o conjunto de dados retornado pela consulta ou instrução qualquer. Também através de outras classes podemos executar stored procedures entre outras funções.

CAPÍTULO 4 – MODELO PROPOSTO

4.1 – Apresentação Geral

A proposta é composta por módulos separados, que se interagem de diversas formas almejando o objetivo do projeto. Esses módulos são as diversas aplicações, hardwares e meios de comunicação.

O conjunto desses diversos módulos, resulta no sistema de gerenciamento do aviário, onde pode-se obter em tempo real as leituras de temperatura e umidade. O sistema irá alertar o gerente do aviário de que o mesmo se encontra em condições críticas de umidade e temperatura. Para tal, o sistema conta com diversas linguagens de programação, hardwares e meios de comunicação. Cada um executando papeis distintos que se interagem e complementam uns aos outros, seguindo uma sequência de passos pré-definidos.

Para iniciar o sistema, uma interface web é responsável por preencher o banco de dados com as informações básicas sobre o aviário e o administrador. Por seguinte, os sensores ficam acarretados de coletarem as leituras de temperatura e umidade, e transmitirem ao microcontrolador que por sua vez envia ao computador via interface USB onde a aplicação Java avalia a temperatura e umidade e salva no banco de dados. Em seguida o administrador pode visualizar na interface web as leituras obtidas pelos sensores, podendo editar a tabela de níveis críticos de temperatura e umidade. A figura 4.1 traz uma breve descrição das etapas do sistema.

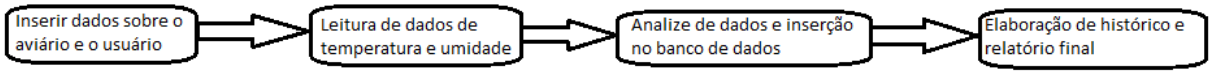


Figura 4.1: Sequência das etapas da solução. (Fonte: Autor)

A interface com o usuário é desenvolvida para o ambiente web sobre a linguagem HTML, CSS, AJAX executando em um servidor local Apache Tomcat 6 onde é executado no computador local (Windows 7 Home Edition) do usuário administrador, onde também está sendo executado a aplicação Java do qual escuta a porta USB esperando dados do microcontrolador. A figura 4.1a ilustra como é realizado a comunicação entre as diversas entidades do sistema, e com quem cada uma se comunica ou reporta.

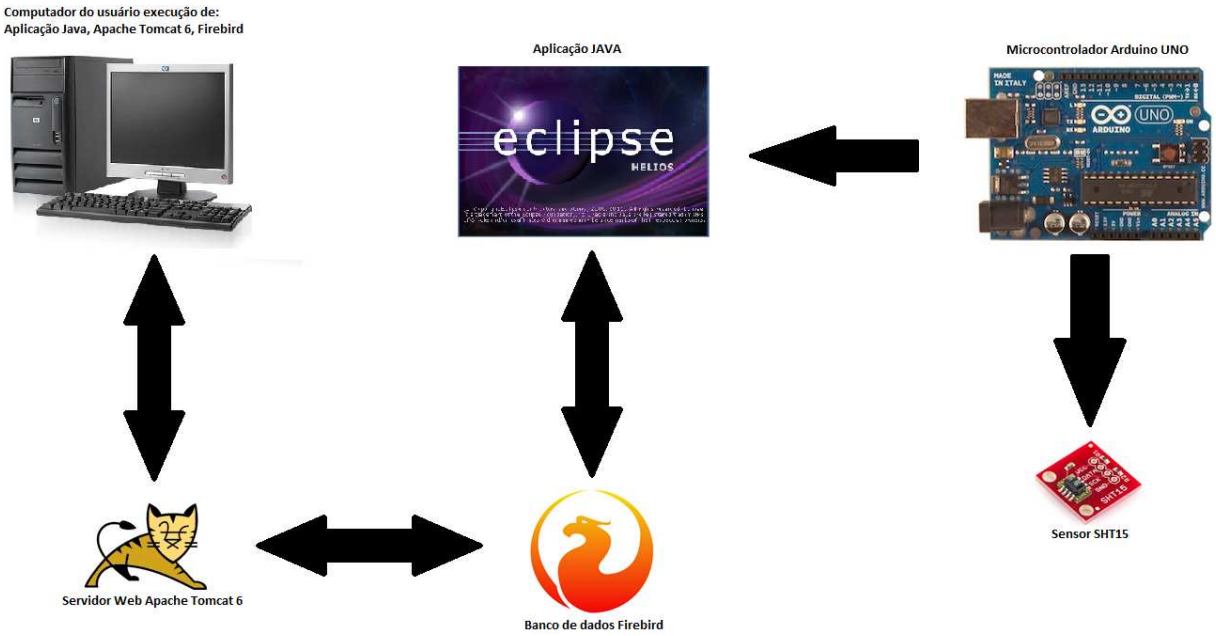


Figura 4.1a: Desenho esquemático da solução. (Fonte: Autor)

4.2 – Software

A parte de software do projeto se divide em três subpartes: a desenvolvida em linguagem de programação do Arduino executada no microcontrolador Arduino Uno. Aplicação em Java ordenada a recepcionar os dados emitidos pelo microcontrolador pela porta USB, além da aplicação Web executada em Servidor Apache Tomcat. Contando também com um Banco de Dados Firebird para o gerenciamento e armazenamento de todos os dados coletados.

O diagrama de atividades 4.2 ilustra as atividades realizadas para a tomada da temperatura e umidade no aviário e como o sistema se comporta para as situações encontradas.

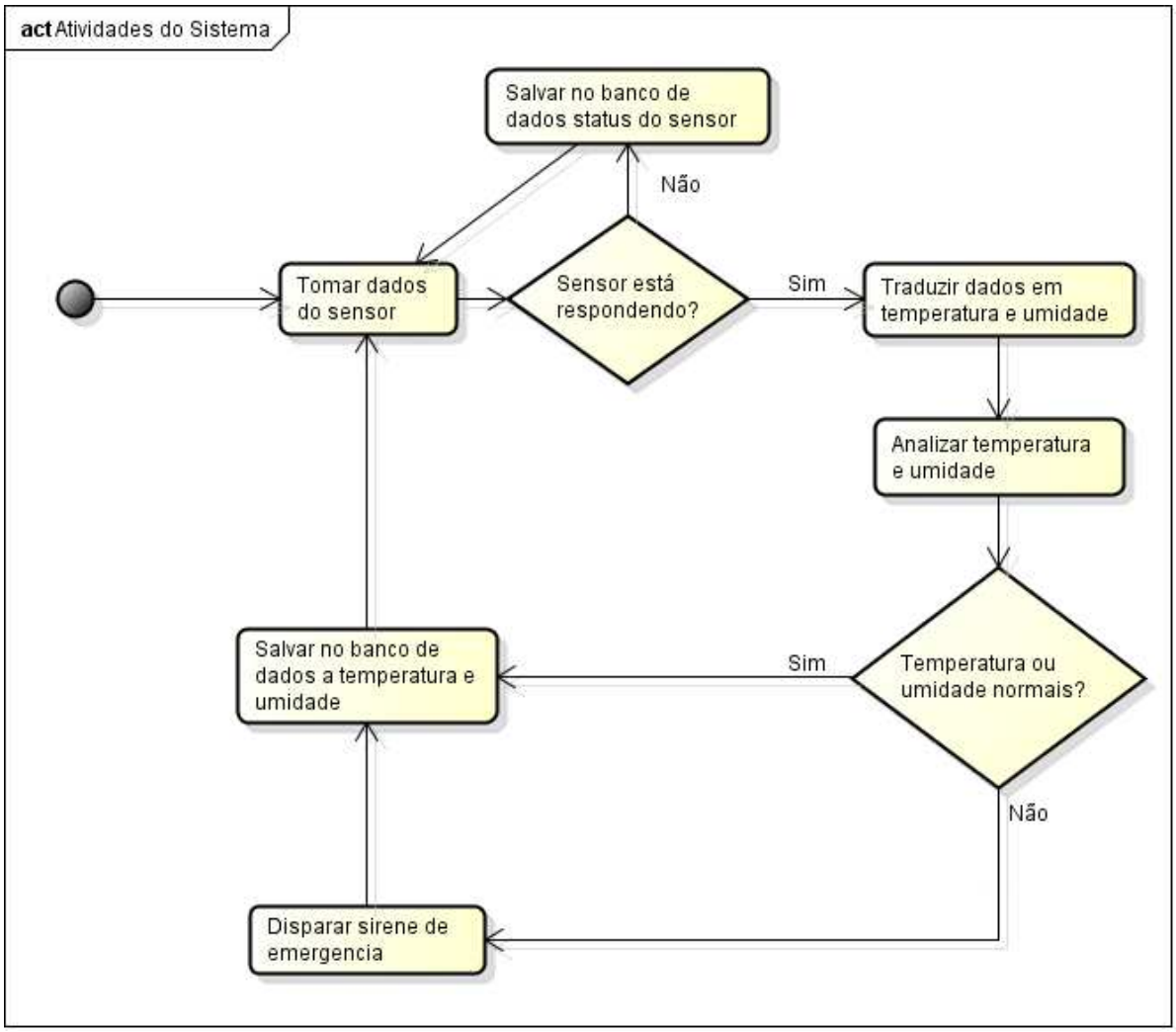


Diagrama 4.2: Diagrama de atividades do sistema. (Fonte: Autor)

O diagrama de atividades 4.2 exibe as atividades que devem ser desempenhadas pelo administrador do sistema desde o início do sistema até o momento em que são impressos os relatórios do aviário.

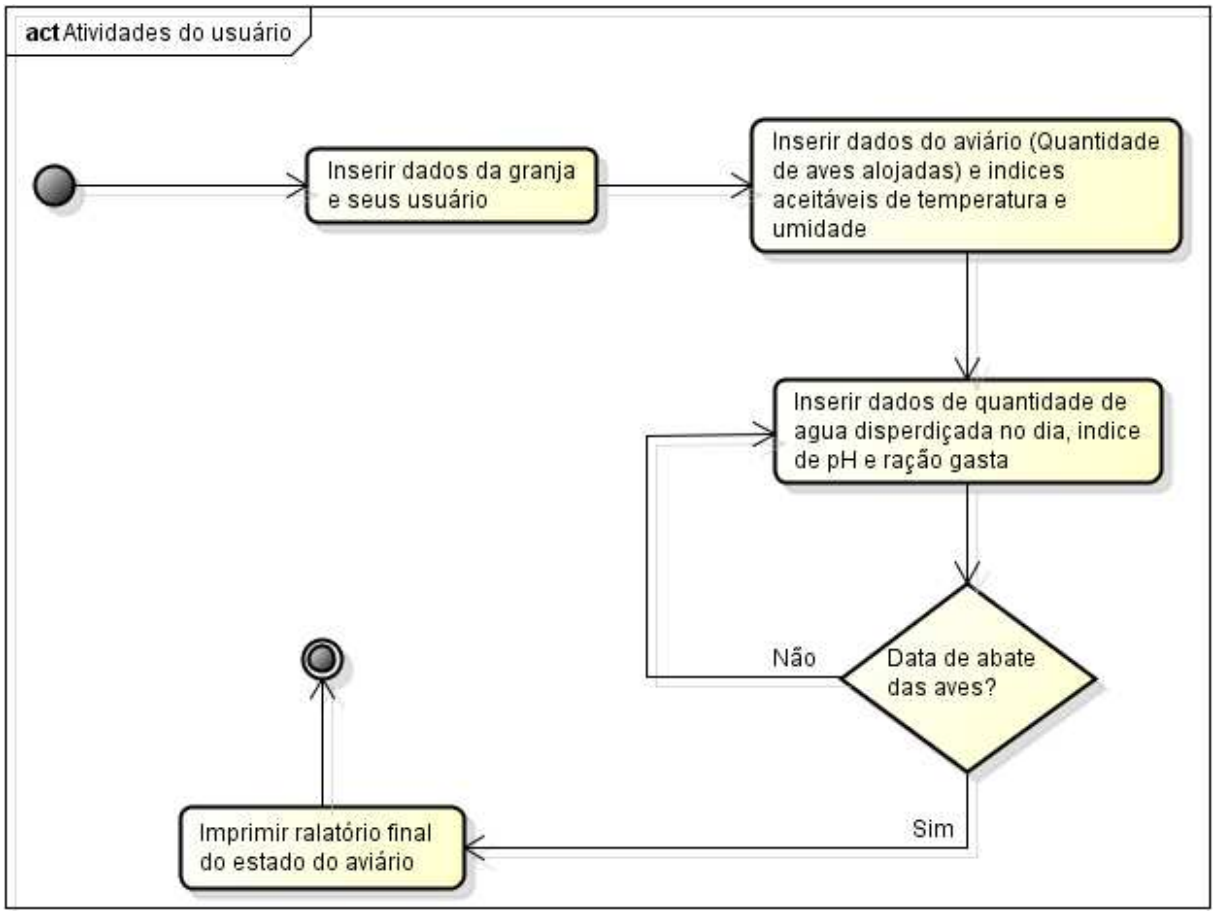


Diagrama 4.2a: Diagrama de atividades do sistema. (Fonte: Autor)

4.2.1 – Aplicação Java

A aplicação Java desempenha várias funções. Entre elas estão:

- Ler informações da porta Serial USB
- Analisar informações comparando com banco de dados;
- Salvar informações no banco de dados;

São essas descritas no diagrama de classe de uso a seguir:

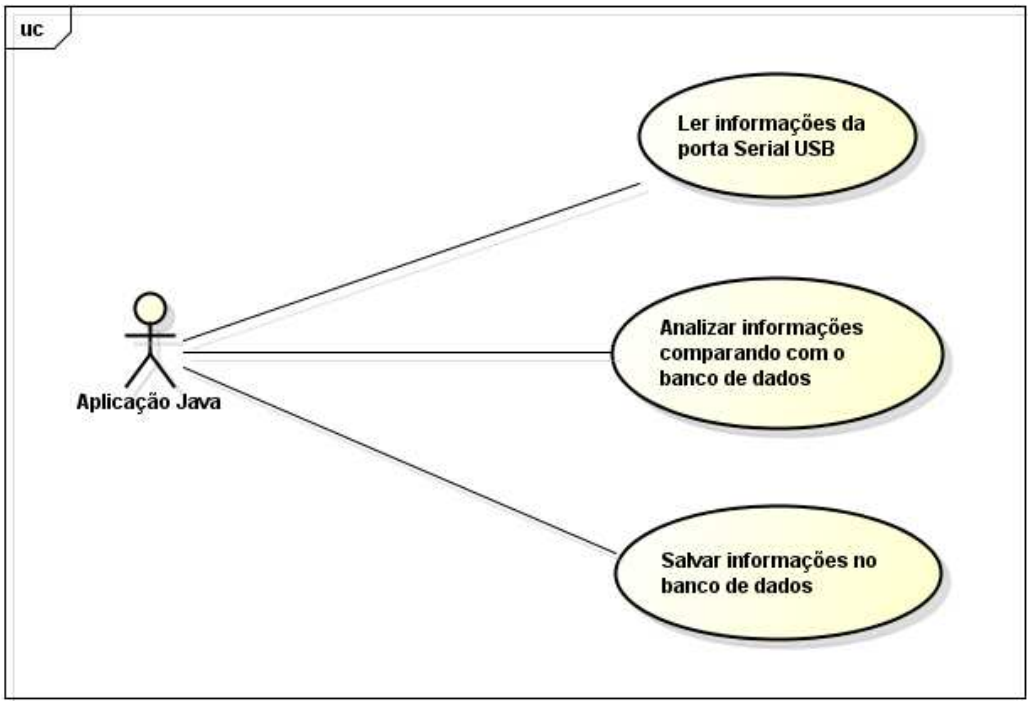


Diagrama 4.2.1: Diagrama de casos de uso da aplicação Java. (Fonte: Autor)

Todos os métodos são separados em classes dentro da aplicação Java;

- Classe SerialTest.java é responsável por testar a porta serial e verificar se há comunicação disponível; Nessa classe que se inicia o sistema(Método Main)
- A Classe Conectar_db.java é incumbida de conectar o sistema ao banco de dados, e retornar às outras classes o Handle de conexão;
- Criar_db.java é a classe que cria o banco de dados caso ele não exista do diretório de instalação;
- Iniciar_db.java é a classe chamada por Criar_db que inicia o banco de dados, criando tabelas e inserindo os primeiros registro do mesmo;
- A classe Inserir_db.java é onde a aplicação insere os dados no banco de dados após realizada a leitura da porta serial. Essa classe é chamada pela classe SerialTest.java, onde a leitura é feita. Inserir_db também verifica em qual lote será inserido o dado, dependendo do sensor do qual enviou a informação;
- Classe Data.java é uma classe que auxilia na manipulação de informações de data e hora, convertendo Strings e Dados para serem comparados de maneira correta e também inseridos no banco de dados;

- A Classe Numero.java é um classe similar a Data.java mas ao invés de trabalhar com data, trabalha com números inteiros e de ponto flutuante, além de gerar médias ponderadas para analise de outras classes;
- Analizar_normalidade.java é a classe que é chamada toda vez que um dado é lido para que seja analisado a temperatura e a umidade e verificar se encontram-se dentro da normalidade;
- Classe somAlerta.java é a classe que dispara um alerta sonoro em thread para alertar os usuário da temperatura fora da normalidade;

O sistema inicia no método Main localizado na classe SerialTest.java. Ao se iniciar o sistema, ele verifica primeiramente se o banco de dados foi localizado, caso contrário, cria um banco de dados novo no diretório de instalação. Verifica também se a DLL rxtxserial.dll está devidamente registrada no sistema operacional. Caso contrário, a DLL deve ser copiada para a pasta \windows\system32 do computador do usuário e registrada pelo comando regsvr32.

Após o sucesso nas verificações a aplicação fica a espera de uma String específica na porta Serial. Essa String contém o seguinte layout:

X00,11,2,Y

Onde o número 00 é o valor da temperatura medida, o número 11 o valor da umidade medida e o número 2 o identificador do sensor que realizou a medição. O caractere X indica o começo de uma transmissão e o caractere Y o final da transmissão, enquanto os caracteres “,” funcionam como separadores de campos.

Após a leitura dos dados, o sistema chama um método Jaybird_inserirDB da classe Inserir_db para inserir. O método verifica para qual lote será inserido o dado, chama o método Analisar da classe Analisar_normalidade para verificar a normalidade e insere os dados no banco de dados.

O método Analisar_normalidade.java verifica o dado que está sendo lido pela aplicação. Extrai as últimas cinco leituras do banco de dados para o sensor, remove a maior leitura e a menor leitura, gera uma média das três leituras restantes e compara com a leitura obtida pelo sistema. Caso a leitura esteja acima ou abaixo do normal, o sistema irá chamar o método Main da classe somAlerta.java para iniciar uma thread que irá disparar um alerta sonoro. O alerta sonoro só irá parar quando o método for invocado novamente.

Após as devidas verificações e tratamentos de dados, ele é enfim inserido no banco de dados, e o sistema espera pela próxima leitura em um ciclo eterno, o sistema só para de capturar quando o usuário envia para a execução um sinal de interrupção.

4.2.2 – Aplicação Web

A aplicação web foi desenvolvida para rodar em um servidor web APACHE TOMCAT 6. A aplicação é desenvolvida em JSP, JavaSript e HTML.

A aplicação tem as seguintes funcionalidades;

Disponibilizar para o usuário os status do sistema;

Salvar configurações do usuário no banco de dados;

Exibir e imprimir os relatórios de desempenho do sistema;

A seguir segue o caso de uso descrevendo das funcionalidades do sistema web;

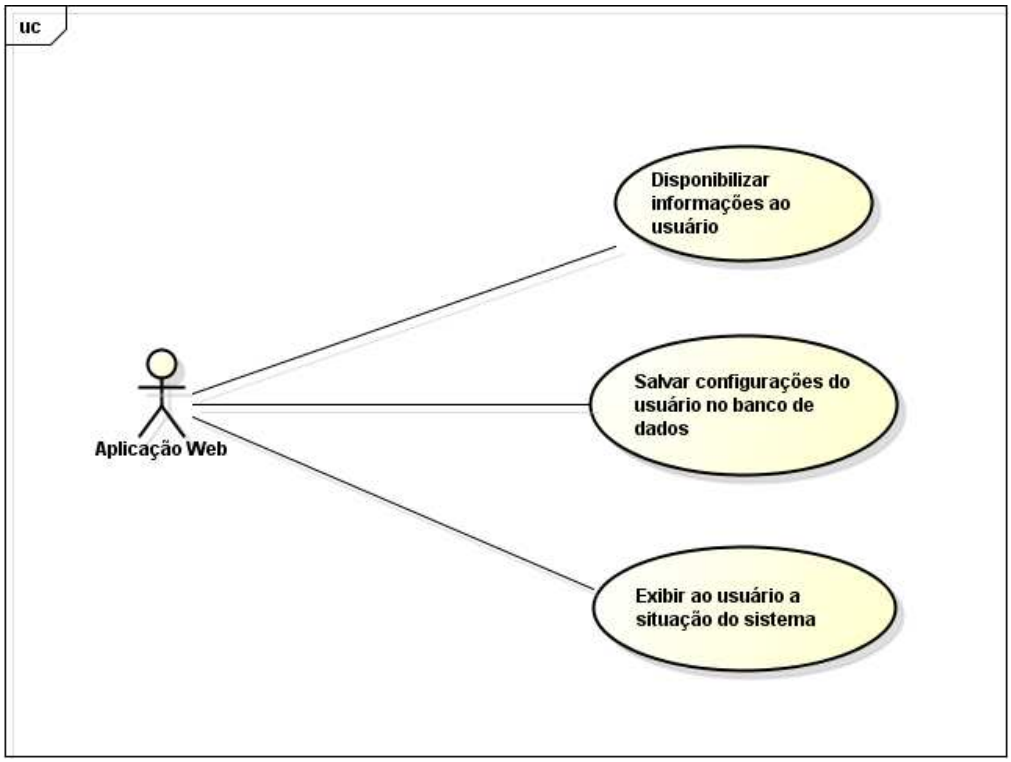


Diagrama 4.2.2: Diagrama de casos de uso da Aplicação Web. (Fonte: Autor)

O computador do usuário deve ter o Apache Tomcat 6 instalado, e também copiar para a pasta webapps dentro do diretório de instalação do Tomcat o arquivo: Aviario.war.

O sistema web inicia-se acessando o link http://localhost:8080/Aviario/Login.wsp através de algum browser de preferência do usuário, e a seguinte tela irá aparecer:

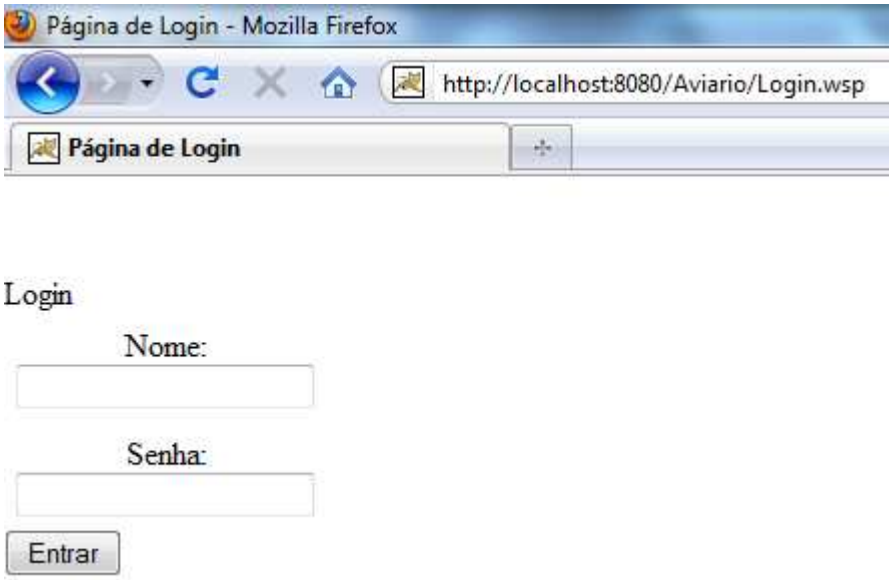


Figura 4.2.2: Página de Login. (Fonte: Autor)

A página de Login é usada para dar segurança no sistema, a fim de permitir apenas usuários cadastrados terem acesso a aplicação. O sistema vem com o usuário administrador cuja senha é admin.

O sistema irá mostrar ao usuário vários menus e opções do que ele pode estar realizando. Os menus são divididos em Gerência, Monitoramento, Relatórios e Gráficos.

- O menu de Gerência irá expandir o sub-menu onde o usuário pode escolher entre as opções: Informações Diárias, Cadastro de Aviário e Cadastro do Sensor;
    - O sub-menu informações diárias,é onde irá disponibilizar uma página que o administrador ou usuário delegado irá inserir informações diárias de leitura de água, mortalidade, ração, também de cadastro do lote e linhagem do respectivo lote.

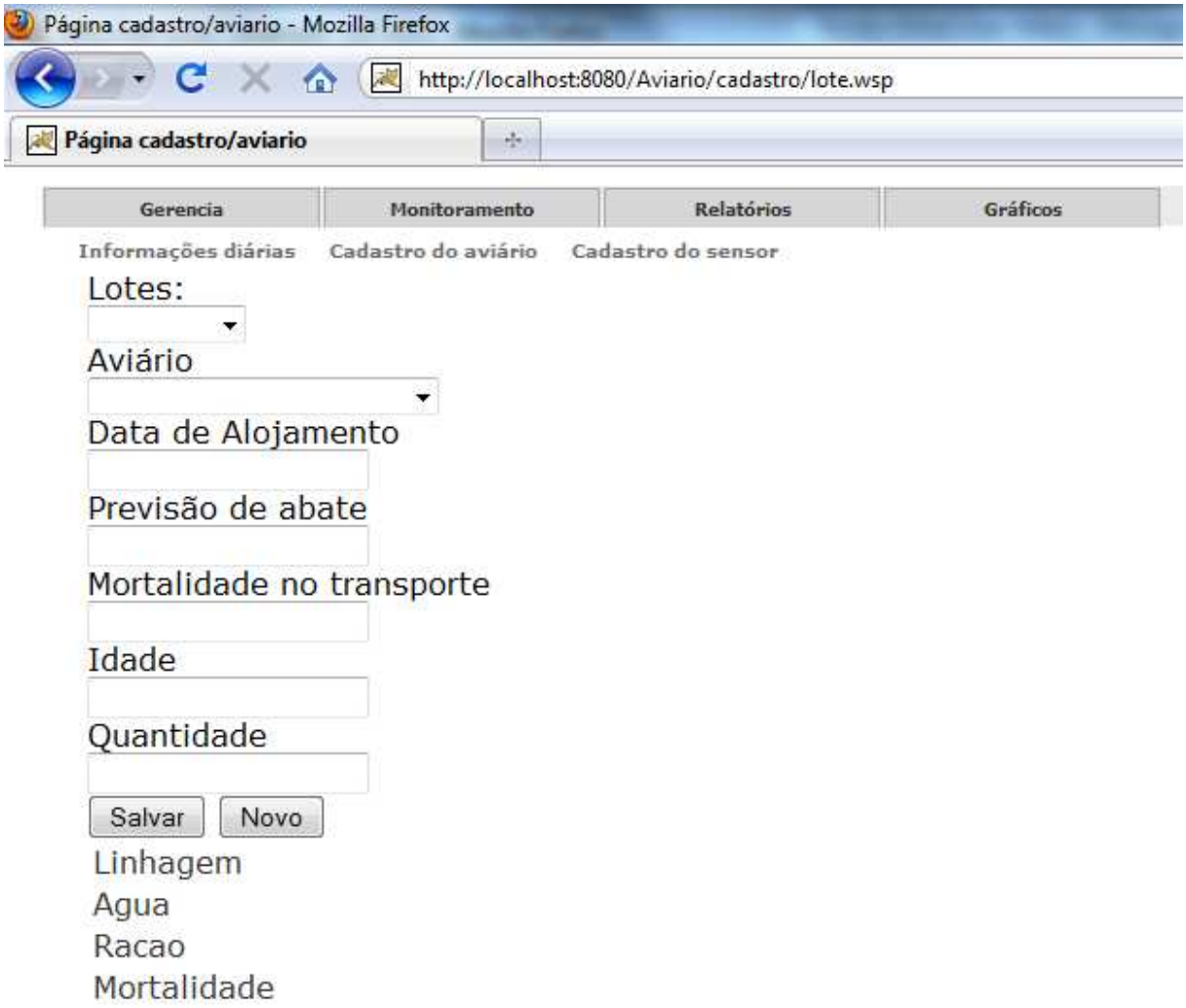


Figura 4.2.2a Página de Cadastro de informações diárias

- Cadastro de aviário é a opção do sub-menu de Gerência onde o administrador pode cadastrar outro aviário ao sistema, ou atualizar informações do mesmo.

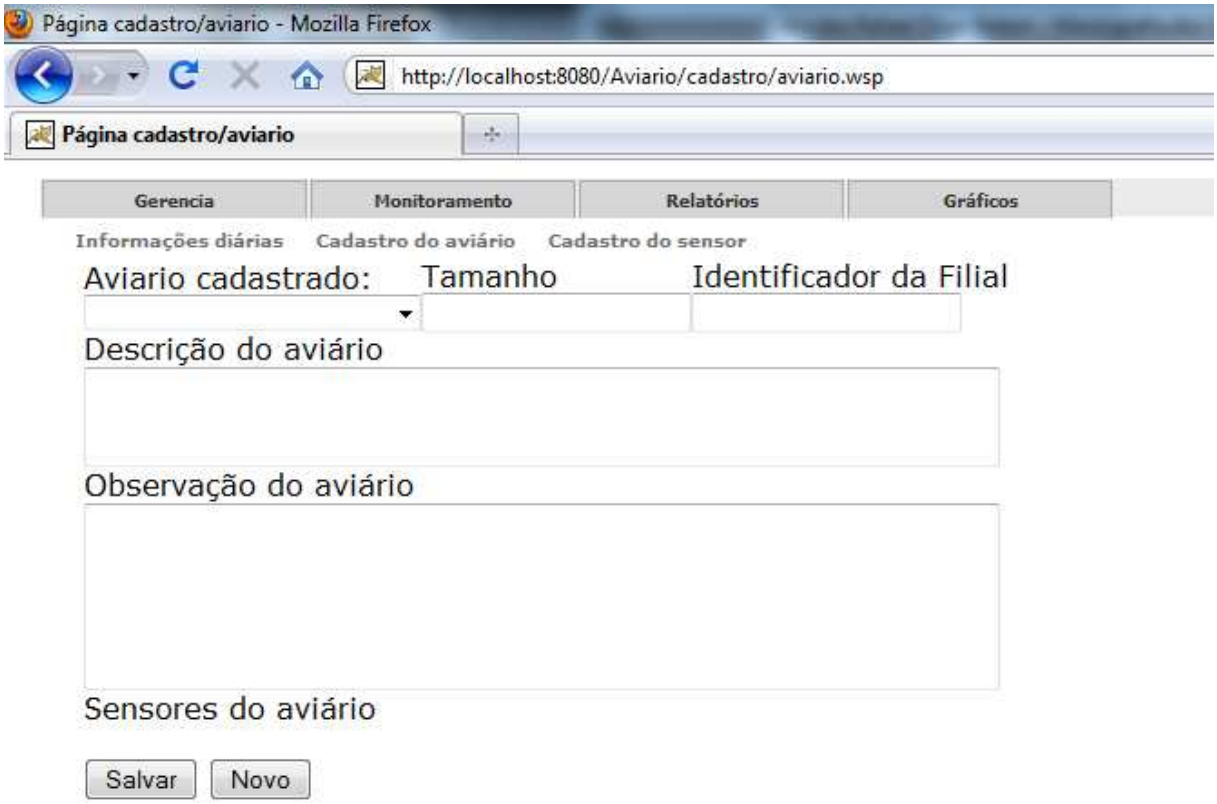


Figura 4.2.2b: Página de Cadastro de aviário. (Fonte: Autor)

- O sub-menu Cadastro do sensor é utilizado para indicar em qual aviário cada sensor se encontra.

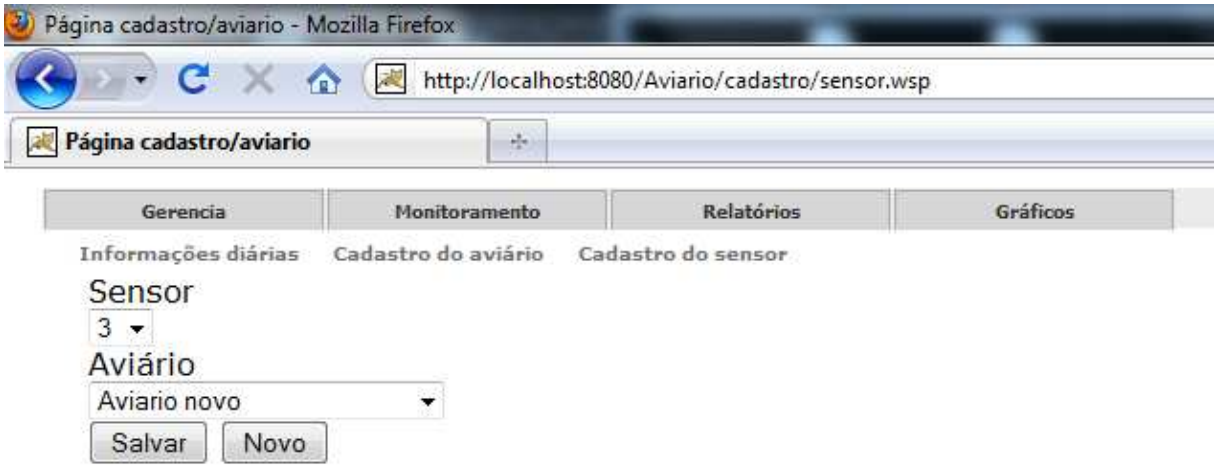
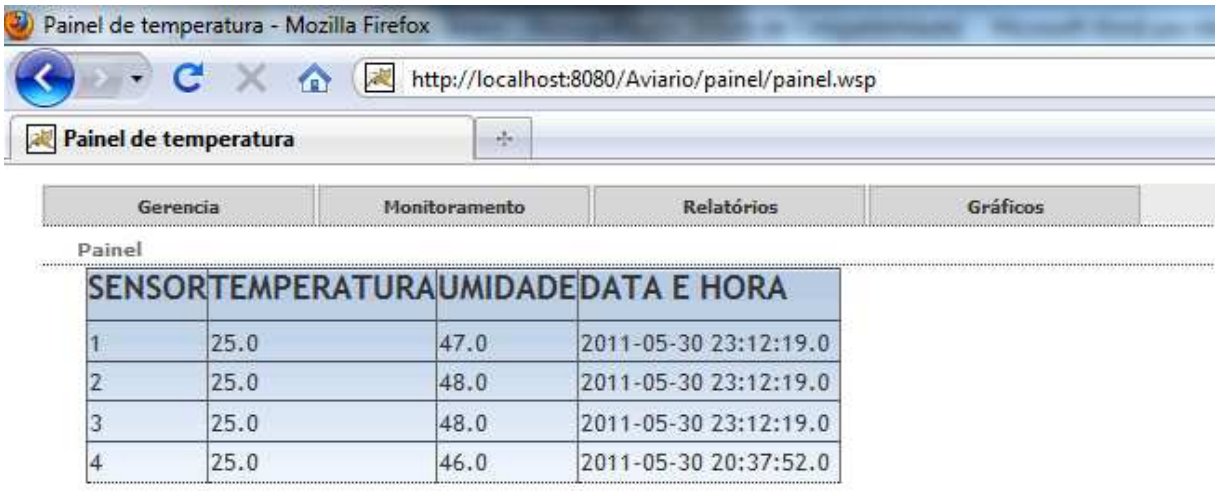


Figura 4.2.2c: Página de Cadastro do sensor. (Fonte: Autor)

- O menu Monitoramento é o menu responsável por monitorar o sistema;

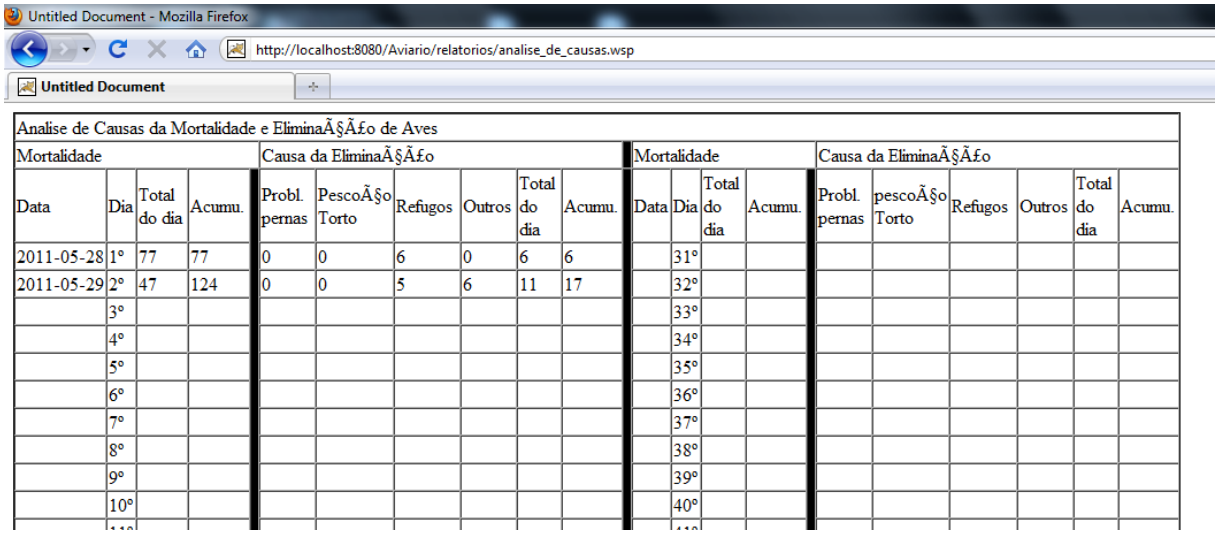
- O menu Monitoramento contém apenas o sub-menu painel onde o mesmo disponibiliza ao usuário uma tabela contendo as informações de temperatura e umidade, identificador do sensor, além da data e hora do sistema, atualizadas constantemente através de Ajax.



SENSOR	TEMPERATURA	UMIDADE	DATA E HORA
1	25.0	47.0	2011-05-30 23:12:19.0
2	25.0	48.0	2011-05-30 23:12:19.0
3	25.0	48.0	2011-05-30 23:12:19.0
4	25.0	46.0	2011-05-30 20:37:52.0

Figura 4.2.2d: Página de Monitoramento em Painel. (Fonte: Autor)

- O menu relatórios é o menu responsável por comportar os sub-menus que imprimem os relatórios do sistema;
    - O sub-menu Mortalidade traz a tela de impressão já com os dados impressos do último lote sobre a mortalidade obtida através dos dados inseridos pelo usuário no sub-menu de informações diárias. Bastando apenas selecionar a opção de impressão do seu browser para imprimir o documento. Consultar Anexo A;

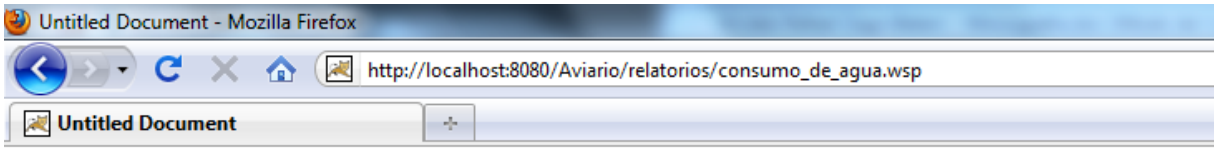


Analise de Causas da Mortalidade e EliminaÃ§Ã£o de Aves

Mortalidade				Causa da EliminaÃ§Ã£o						Mortalidade				Causa da EliminaÃ§Ã£o					
Data	Dia	Total do dia	Acumu.	Probl. pernas	PescoÃ§o Torto	Refugos	Outros	Total do dia	Acumu.	Data	Dia	Total do dia	Acumu.	Probl. pernas	pescoÃ§o Torto	Refugos	Outros	Total do dia	Acumu.
2011-05-28	1°	77	77	0	0	6	0	6	6		31°								
2011-05-29	2°	47	124	0	0	5	6	11	17		32°								
	3°										33°								
	4°										34°								
	5°										35°								
	6°										36°								
	7°										37°								
	8°										38°								
	9°										39°								
	10°										40°								

Figura 4.2.2e: Página de relatório de mortalidade. (Fonte: Autor)

- Consumo de Água é a segunda opção do sub-menu que traz na tela o documento de impressão do consumo de água do aviário. Consultar Anexo A;



Data	Idade	Volume leitura	Água eliminada(L)	Consumo		Data	Idade	Volume leitura	Água eliminada(L)	Consumo	
				Diário(L)	Acumulado(L)					Diário(L)	Acumulado(L)
	1						36				
	2						37				
	3						38				
	4						39				
	5						40				
	6						41				
	7						42				
	8						43				
	9						44				
	10						45				
	11						46				
	12						47				
	13						48				
	14						49				
	15						50				
	16						51				
	17						52				
	18						53				
	19						54				

Figura 4.2.2f: Página de relatório de consumo de água. (Fonte: Autor)

- O menu Gráficos é o menu onde disponibiliza gráficos que o sistema desenha;
    - Gráfico da temperatura disponibiliza na tela o gráfico da temperatura mostrando três linhas. Linha verde significa a temperatura máxima, linha azul a temperatura mínima e linha vermelha a temperatura do sistema. A temperatura é medida a partir da média diária das temperaturas. Consultar Anexo A;

Figura 4.2.2g: Gráfico da análise de temperatura. (Fonte: Autor)

    - O gráfico da umidade tem comportamento similar ao gráfico da temperatura, com a cor verde para umidade máxima, cor azul umidade mínima e cor vermelha para umidade do sistema.

Figura 4.2.2h: Gráfico da análise de umidade. (Fonte: Autor)

- O eixo X de ambos os gráficos indica a data da média da temperatura, e o eixo Y mostra ou a temperatura em ºC ou a umidade relativa em % para o gráfico de temperatura e umidade respectivamente.

4.2.3 – Algoritmo Arduino

O algoritmo em execução no microcontrolador Arduino Uno é desenvolvido na linguagem de programação do Arduino, o Algoritmo é ordenado a verificar se contem algum sensor conectado aos pinos de comunicação digital do microcontrolador Arduino Uno. Todo o algoritmo é dividido em Funções, uma vez que a linguagem de programação é estruturada e não orientada a objetos como a linguagem de programação Java. O diagrama a seguir exibe uma breve descrição das funções delegadas ao microcontrolador.

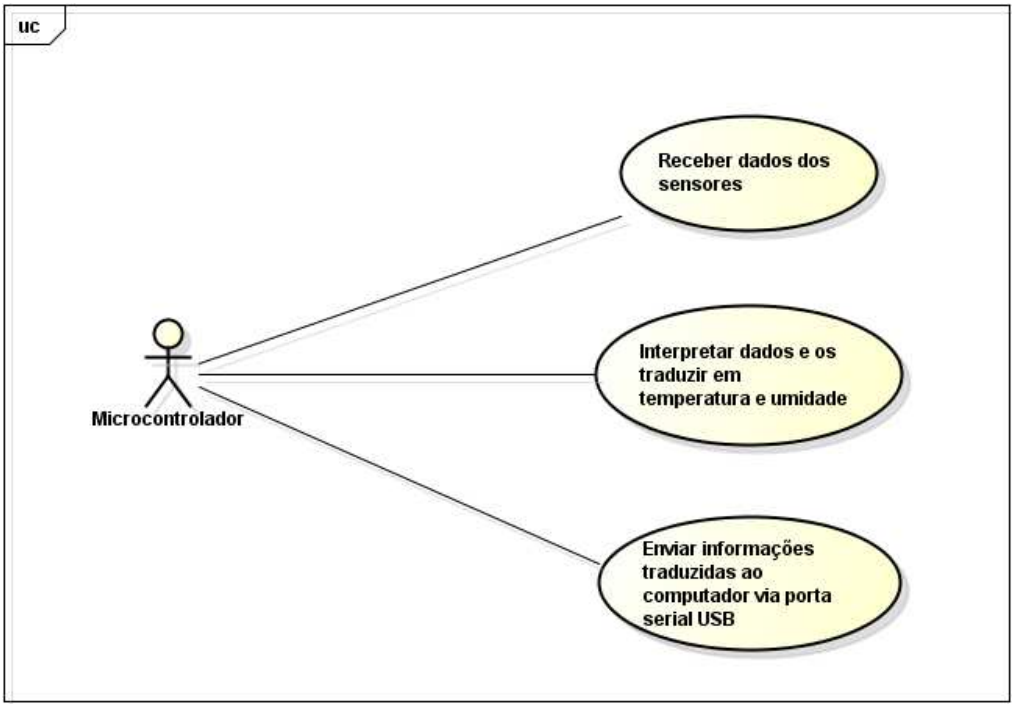


Diagrama 4.2.3: Diagrama de casos de uso do microcontrolador. (Fonte: Autor)

Cada função desempenha um papel único na aplicação. O programa se inicia quando o microcontrolador é ligado a uma porta USB e sequentemente inicializando a porta USB a 9600 bounds. Ao iniciar o Algoritmo se inicializa também variáveis globais como as portas utilizadas para cada um dos sensores, variáveis que armazenam a temperatura lida e a umidade, também os comandos do sensor de temperatura em binário.

Após a inicialização da porta, o programa entra em Loop através da função void loop. O loop realiza apenas chamadas da função callMultSensor que recebe como parâmetros os pinos do sensor do qual se deseja realizar a leitura, entre cada chamada da função o loop realiza um delay de 100 milissegundos. No final da execução de todo o loop é realizado um delay de 2000 milissegundos.

Dentro da função callMultSensor, é chamada a função sendCommandSHT passando como parâmetro o comando de temperatura e os pinos do sensor em questão. Essa função é responsável por ordenar o sensor SHT15 a retornar pelo pino de data a temperatura que está sendo lida no momento. Após a execução da função é necessário se chamar a função waitForResultSHT passando como parâmetro o pino de data do qual será lido o dado que foi ordenado anteriormente ao sensor que realiza a leitura.

O sensor precisa preencher um dado do tipo inteiro de oito bits contudo ele transmite os bits em pacotes de 4 bits, primeiramente os quatro mais significativos, posteriormente os quatro menos significativos, para organizar o dado a função getData16SHT é chamada.

Para cada comando, temperatura (0b00000011) e umidade (0b00000101) em binário, são retornadas o seu valor correspondente, porêm o valor retornado pelo sensor ainda precisa ser tratado para que seja traduzido em graus Celsius (Temperatura) e Umidade Relativa (Umidade). O calculo da temperatura é:

temperature = (float)temp * 0.01 – 40

O calculo de tradução da Umidade Relativa é:

humidity = -4.0 + 0.0405 * val + -0.0000028 * val * val

Métodos fornecidos pela fabricante do sensor de umidade e temperatura SHT15 SparkFun.com (http://www.sparkfun.com/)

Após a construção do valor, ele é concatenado em uma String para ser enviado pela porta USB de forma que a Aplicação Java possa identificar e traduzir os dados em informação. A String é inicializada pelo caractere “Y” em seguida passa-se o valor da temperatura em Celsius logo após é concatenado o caractere “;” para que delimite o valor da temperatura com o da Umidade Relativa que vem em seguida, e para concluir a String e sinalizar que finalizou o pacote é concatenado o caractere “X”. Dessa forma a Aplicação Java é capaz de interpretar as informações que são recebidas pela porta serial USB.

4.2.4 – Modelo do banco de dados

O modelo do banco de dados visa a escalabilidade do sistema, possibilita a montagem das páginas Web de forma dinâmica de acordo com o perfil de usuário logado no sistema. Permite a implementação de vários sistemas dentro do mesmo banco de dados, desde que se adicione as tabelas para os novos sistemas.

O modelo propõe a criação de módulos e rotinas, onde cada módulo pode conter uma ou mais rotinas. Os módulos são os menus superiores da página web e as rotinas são os sub-menus onde são salvas os endereços das aplicações web, para no momento em que for solicitada cada rotina, ela poder direcionar para a página correspondente. A seguir segue o modelo do banco de dados.

O usuário está ligado diretamente ao perfil, e a partir do perfil que se sabe quais as rotinas disponíveis para o usuário. A tabela rotina_perfil que possibilita a relação de muitos para muitos que é necessário entre a tabela de perfil e rotina.

A tabela Status permite a exclusão lógica de cada item das rotinas, módulos, perfil e sistema.

Pelo autor a tabela de Lote é considereda a tabela principal, nela é feita a ligação entre as tabelas que auxiliam na montagem dos relatórios e as tabelas de (Linhagem_lote, Água, Qualidade_ar, Mortalidade, Ração), onde a tabela de lote se liga ao aviário, podendo cada aviário conter vários lotes. O aviário por sua vez é ligada á filial onde cada filial pode conter vários aviários. A tabela de sensor também é ligada à tabela de aviário, podendo cada aviário conter um ou mais sensores.

A tabela de normalidade não está ligada a nunhuma tabela, pois ela auxilia apenas na montagem do gráfico, mostrando a temperatura e umidade máxima e mínima. Também é usada pela aplicação java para avaliar se a umidade ou temperatura ultrapassaram os limites. As figuras 4.2.4 e 4.2.4a ilustram o modelo do banco de dados.

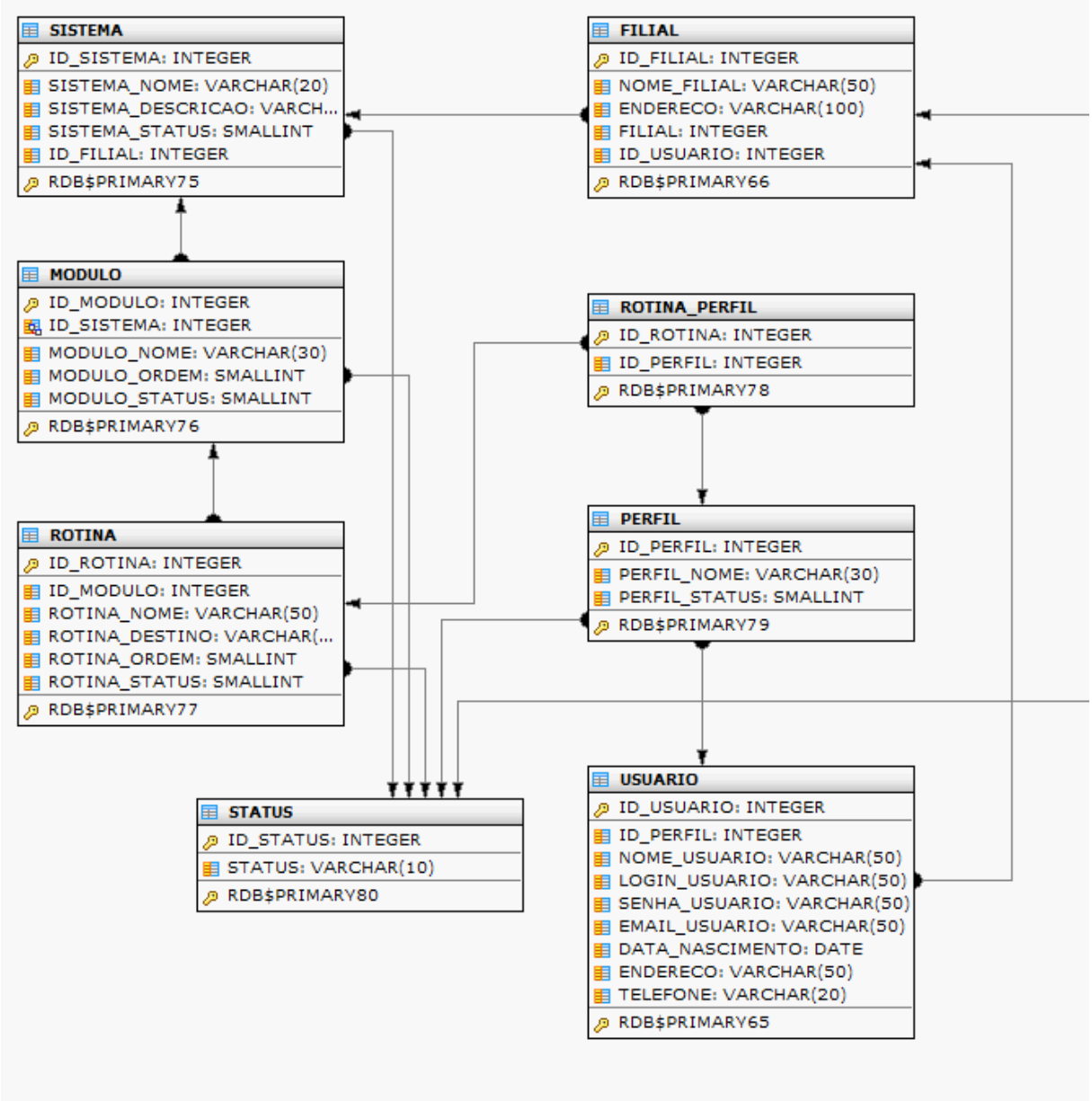


Figura 4.2.4: Modelo do banco de dados, página 1. (Fonte: Autor)

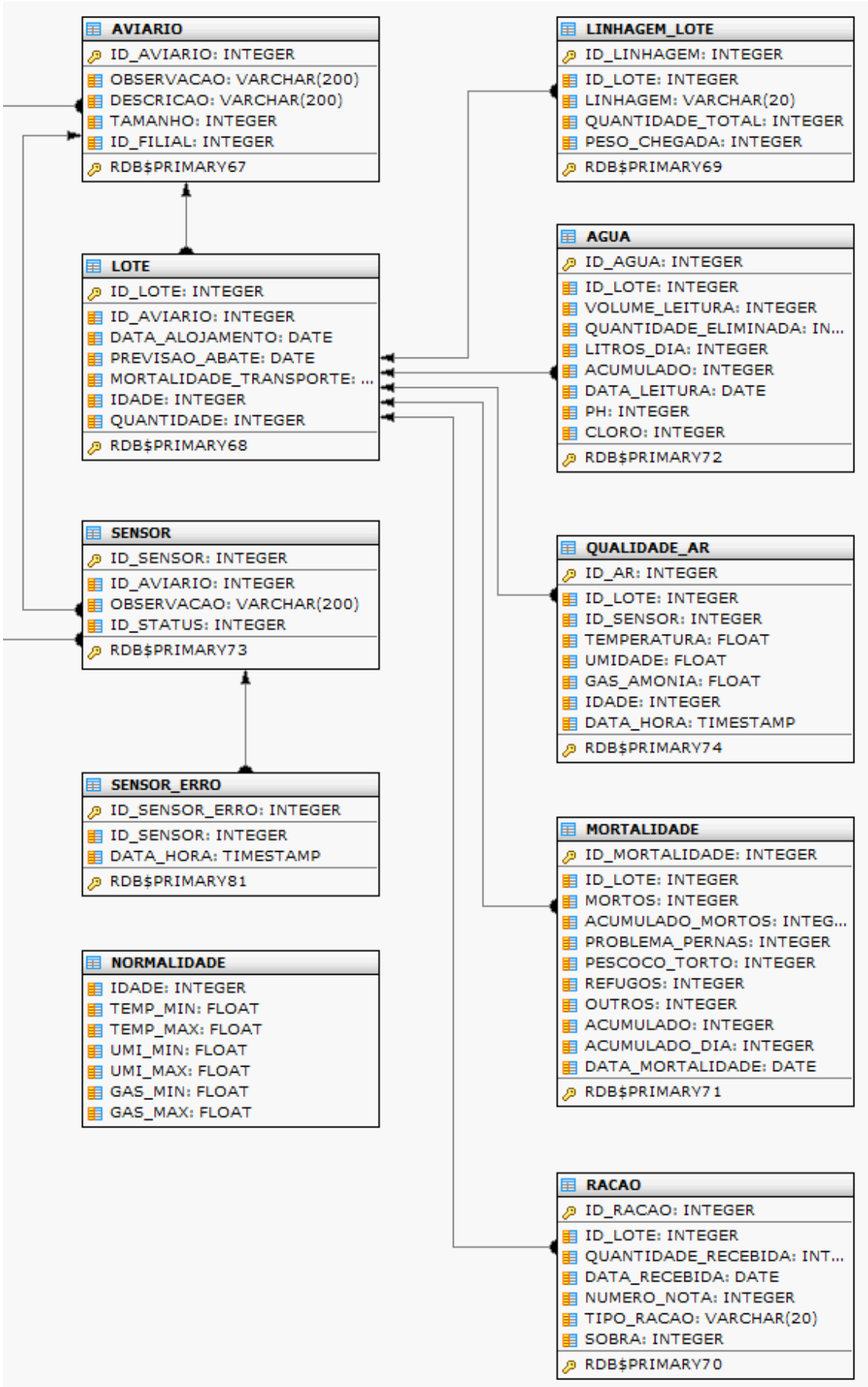


Figura 4.2.4a: Modelo do banco de dados, página 2. (Fonte: Autor)

4.3 – Hardware

Os componentes de *hardware* são compostos por um microcontrolador Arduino Uno onde se encontra o microprocessador ATmega 8, sensores de umidade e temperatura SHT15 que se encontram soldadas na placa da distribuidora Sparkfun.com. Utiliza-se também cabos de rede e conectores RJ45 para a comunicação entre o microcontrolador e os sensores. Um cabo USB do modelo A para B realizando a comunicação entre o microcontrolador e o computador. Uma caixa Patola PB-207 para alojar o microcontrolador e o cabeamento interno. Também se utiliza quatro conectores RJ45 fêmea para realizar a conexão do cabo de rede com a caixa Patola onde se encontra o microcontrolador.

4.3.1 – Caixa Patola PB-207

O microcontrolador é um dispositivo sensível necessitando de um compartimento que o proteja de quedas, esbarrões e o acúmulo de poeira, dentre outros resíduos que possam danificar o equipamento. Foi escolhida a Parola PB-207 para alojar o microcontrolador e suas conexões de forma segura contra essas adversidades citadas.

A patola não só protege o microcontrolador mas como também serve de abrigo as conexões dos conectores RJ45 fêmea tornando também um meio prático a qualquer usuário conectar o cabo de rede com o conector RJ45 ao microcontrolador.

A patola PB-207 contem uma altura de 46mm largura de 142mm e comprimento de 130mm. Gabinete termoplástico injetado sob alta pressão de injeção; em duas partes com acabamento fosco; painéis para fechamento frontal e traseiro na cor cinza; as partes são fixadas por trilhos laterais; alça para transporte; torres de fixação para placa de circuito impresso; abertura para ventilação; modificações ou outras cores sob consulta. Material ABS preto. A figura 4.3.1 exibe uma imagems ilustrativa da caixa patola utilizada. (http://www.patola.com.br/?pagina=info_produto.php&produto_id=144&titulo=PB-207).

Figura 4.3.1: Imagem ilustrativa Patola PB-207.

A patola precisa ser adaptada para o suporte as necessidades do projeto, essas necessidades são quatro cortes na tampa frontal para o encaixe dos conectores RJ45 fêmea, um corte na tampa trazeira para o encaixe do conector USB. Os cortes são realizados para que não seja necessário a abertura da parola para se realizar as conexões dos cabos de comunicação. A Imagem 4.3.1a traz a caixa patola utilizada em fase de prototipação.

Figura 4.3.1a: Imagem Patola PB-207 com cortes frontais. (Fonte: Autor)

4.3.2 – Conectores

4.3.2.1 – Conector RJ45

Conector RJ45 facilmente encontrado no mercado, a um preço muito baixo, e fácil manipulação, são as características que tornaram a escolha certa para o tipo de conector a ser utilizado no projeto. A figura 4.3.2.1 ilustra os conectores RJ45.

Figura 4.3.2.1: Conector RJ45.

O conector RJ45 foi bastante usado no projeto pois ele possibilitou a conexão entre o microcontrolador e os sensores, através de um cabo de rede de 4 pares. Foi possível realizar todas as ligações entre o microcontrolador e o sensor.

4.3.2.2 – Conector RJ45 Fêmea

O conector RJ45 Fêmea é utilizado dentro da caixa em que Patola que comporta o microcontrolador tanto na caixa que comporta os sensores, ele é utilizado para receber o conector RJ45 e distribuir os contatos para dentro dos seus respectivos compartimentos. A imagem 4.3.2.2 exibe o conector RJ45 Fêmea.

Figura 4.3.2.2: Conector RJ 45 Fêmea.

O conector RJ45 Fêmea tem as suas ligações internas do tipo push-down. São conexões mais confiáveis, dando maior segurando para que não aja mau contato ou curto circuito, são também muito seguras para que não se desconectem com facilidade.

A seguir segue a figura 4.3.2.2a exibindo o protótipo em fase de confecção onde estava sendo conectado a placa Arduino uno aos conectores RJ 45 Fêmea.

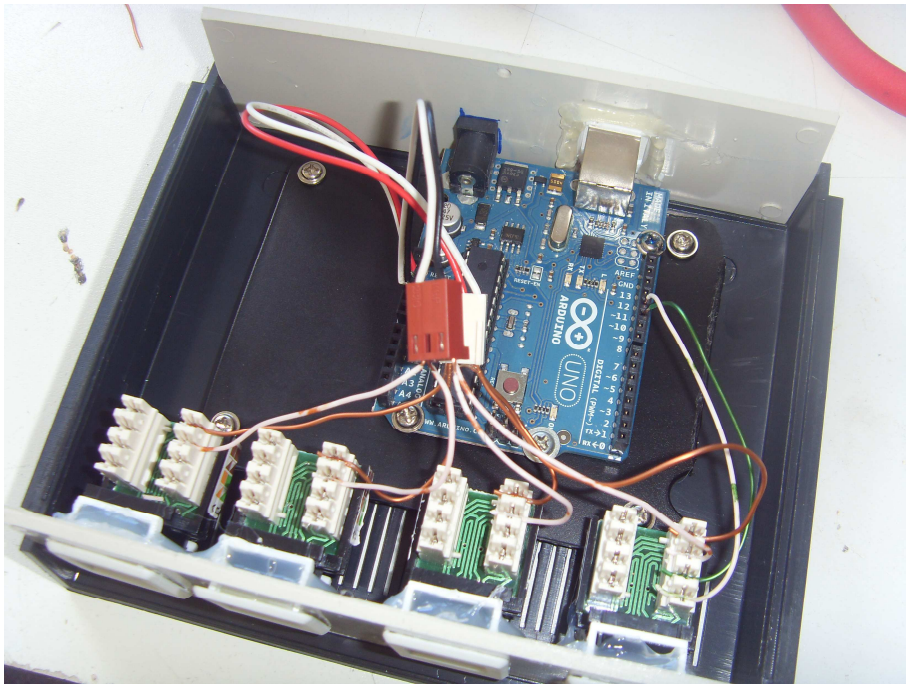


Figura 4.3.2.2a: Protótipo em confecção. (Fonte: Autor)

CAPÍTULO 5 – TESTES

5.1 – Modelo de Aplicação

A área de aplicação do sistema é o Aviário número 1 da Granja Perola do Sul e o avaliador fica sendo o usuário final do sistema, o próprio granjeiro, colocando o sistema em ambiente de produção comercial para o qual foi desenvolvido. É válido como resultado positivo do projeto a tomada de dados de temperatura e umidade, captura dos dados, e elaboração dos relatórios e gráficos que o sistema propõe. A figura número 5.1 ilustra a localização do aviário e o escritório onde se localiza o computador para a gerencia da granja.

Figura 5.1: Imagem satélite da granja (Fonte: Google Earth)

A Figura 5.1a ilustra o cabeamento instalado do escritório onde se localiza o computador do usuário até o aviário número 1 onde os dados devem ser tomados, a distância entre o escritório e o aviário é de 30 metros em linha reta.

Figura 5.1a: Cabeamento instalado do escritório até o aviário

O sensor ficou atrelado ao bebedouro das águas para dar suporte ao sensor e também para o mesmo se deslocar junto ao equipamento, uma vez que o bebedouro é elevado para que outros equipamento passem na cama das aves. Vide figura 5.1b.

Figura 5.1b: Sensor de umidade e temperatura atrelado ao bebedouro das aves

Ao mesmo tempo em que o sensor ficou atrelado ao bebedouro das aves, ficou também a cinco metros de distância do aquecedor do aviário, para que o sensor capte as mudanças de

temperatura mais rapidamente ao se acionar o aquecedor ou desativa-lo. O aquecedor é um equipamento movido a lenha e bastante utilizado nos primeiros dias de vida das aves onde elas precisam de mais temperatura. O aquecedor do aviário é localizado no centro do aviário, assim elevando a distância do cabo mais vinte metros. A figura 5.1c ilustra o aquecedor do aviário.

Figura 5.1c: Aquecedor do aviário movido

5.2 – Descrição da Aplicação do Modelo

Primeiramente, o sistema é instalado no computador do usuário, todas as ferramentas descritas no projeto. É instalado também o hardware da caixa Patola contendo o microcontrolador com as conexões e software instalados. Passa-se todo o cabeamento entre a caixa Patola e o computador (Cabo USB), cabeou-se entre o aviário e o local onde fica o computador do usuário com cabo de rede, a fim de ligar os sensores a caixa Patola.

Iniciar o aplicativo Java onde já irá criar o banco de dados, e capturar os dados. Com a aplicação funcionando basta acessar o link de Login para ter acesso aos dados de temperatura e umidade do sistema, além de poder inserir dados da tarefa do usuário.

5.3 – Resultados

5.3.1 – Resultados obtidos

A Aplicação Java foi instalada com êxito no computador de trabalho do usuário final, não apresentou nenhuma dificuldade em se tratando de performance, instalando todas as suas variáveis e arquivos com velocidade e confiabilidade, gerando o banco de dados e se comunicando com o mesmo com êxito.

O sistema Web não apresentou dificuldade alguma no momento da instalação, configuração ou utilização. A instalação procedeu sem falhas e a conexão com o banco de dados foi realizada com êxito. Foi passado um treinamento para o usuário sobre como se utiliza o sistema Web e se gerência todo o sistema. O usuário conseguiu manusear o sistema de maneira adequada. A Figura 5.3.1 exibe o aplicativo Web exibindo os dados de temperatura e umidade.

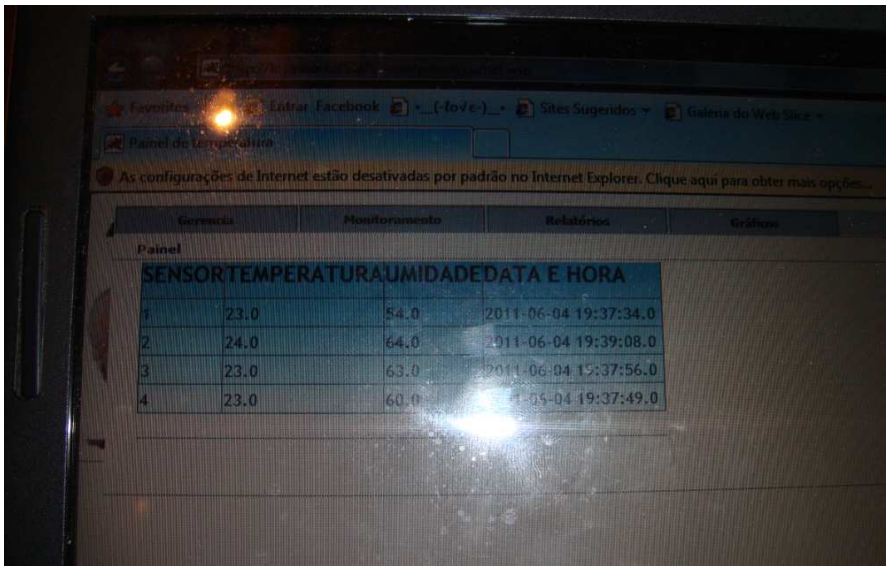


Figura 5.3.1: Aplicação Web exibindo dados de temperatura e umidade

O microcontrolador esteve operante a todas as etapas do projeto, no entanto, a distância elevada entre o escritório e o aviário dificultaram no recebimento do sinal dos sensores, o cabo utilizado entre os dois pontos acima de 70 metros impossibilitou a troca de informação entre os dispositivos. O microcontrolador funcionou corretamente, emitindo os

dados para o sensor, mas não recebia resposta do mesmo. Um teste foi realizado onde se conectou um cabo de curta distância no sensor e microcontrolador e os dados foram lidos corretamente. A figura 5.3.1a exibe o microcontrolador conectado ao computador do usuário e ao sensor no aviário.

Figura 5.3.1a: Escritório com o computador e microcontrolador conectados

5.3.3 – Comparação entre os resultados

A Aplicação Java atendeu as expectativas do autor ao criar o banco de dados e se comunicar com a porta serial e também ao criar o banco de dados e se comunicar com o mesmo.

A aplicação Web procedeu conforme o esperado, exibindo as informações ao usuário e comunicando com o banco de dados, tanto recuperando dados do mesmo, quanto escrevendo as configurações do usuário no banco de dados.

O banco de dados atendeu as expectativas, armazenado as configurações do usuário e os dados do sistema de modo confiável e seguro, atendendo também as requisições simultâneas por parte da aplicação Java quando do aplicativo Web com desempenho e performance esperados pelo autor.

O microcontrolador foi testado e funciona corretamente no meio de desenvolvimento, onde a distância cabeada testada que se tornou operante, obteve um alcance de até quinze

metros testado pelo autor. No meio de produção onde a distância é superior aos quinze metros mencionados pelo autor, o projeto fica inoperante.

5.4 – Custos

Um dos objetivos do autor durante o projeto é tentar manter um nível de qualidade dos equipamentos elevados para garantir a confiabilidade e segurança, junto da procedência dos fornecedores, e minimizar custos na elaboração do projeto.

A placa Arduino Uno tem valor no mercado brasileiro de R$ 136,00 o sensor de umidade e temperatura SHT15 da Sparkfun tem valor de mercado de R$ 160,00 utilizando-se 4 sensores no projeto totalizando R$ 640,00, o cabo USB do modelo A/B tem custo de R$ 15,00. Para conectar o microcontrolador e os sensores foram utilizados aproximadamente duzentos metros de cabo de rede com o metro custando R$ 1,00 totalizando R$ 200,00.

As caixas Patola para o microcontrolador e para o sensor custa R$ 15,00 e R$ 2,00 cada uma respectivamente. Os conectores RJ45 custam R$ 0,10 e o RJ45 Fêmea para push-down custa R$ 2,00 utilizando 8 para os conectores RJ 45 Fêmea e também para os RJ45 totalizam R$ 16,80 de conectores.

5.5 – Avaliação Global

De uma maneira geral o projeto cumpriu com o seu objetivo de realizar a tomada de dados de umidade e temperatura e gerar os relatórios. O sistema é capaz de receber, traduzir e disponibilizar as informações ao usuário. A única ressalva se deve ao comprimento do cabo que impossibilita a comunicação entre o microcontrolador e o sensor de umidade e temperatura. No entanto, a aplicação do padrão rs 485 de comunicação entre o microcontrolador e o sensor, possibilitaria a implementação e correção do empecilho, com um cronograma mais largo, o autor poderia ter desenvolvido tal dispositivo.

Evidentemente, que o software tanto da aplicação em Java quanto da aplicação Web são pontos fortes do sistema. Ambos funcionando conforme as expectativas do autor tanto na fase de desenvolvimento quanto na fase de implementação do sistema.

O sistema microncontrolador apresentou deficiência ao não conseguir trafegar os dados de umidade e temperatura para longas distâncias cabeadas, conforme o autor, para se realizar o trâmite de informações seria necessário a implementação do padrão rs 485 de comunicação. Apesar de ser a parte crítica do projeto para o pleno funcionamento, o cronograma não permitiu a implementação do mesmo.

CAPÍTULO 6 – CONCLUSÃO

6.1 – Conclusões

O desenvolvimento do projeto agregou conhecimento a partir de diversos pontos de vista. Para o cumprimento das demandas exigiu-se diversas horas de esforço, muita dedicação e seriedade para com o objetivo do projeto.

O projeto de aquisição, análise e elaboração de relatórios a partir de informações de umidade e temperatura do aviário mostrou-se cumprido parcialmente, onde uma distância de 70 metros entre o microcontrolador e os sensores inviabilizaram a implementação do projeto pelo arquitetura proposta.

Toda a área de software foi desenvolvida e cumprida conforme o cronograma e as necessidades do usuário final, onde todas as informações podem ser salvas e exibidas ao usuário de modo interativo e intuitivo.

Os resultados obtidos a partir de teste realizados pelo autor comprovam a viabilidade do projeto proposto. Todos os dados são trafegados, salvos, e exibidos ao usuário até a fase de elaboração de relatórios. Os dados foram traduzidos para graus célsius e umidade relativa, salvas no banco de dados e exibidas pelo aplicativo Web para o usuário.

6.1 – Sugestões para Trabalhos Futuros

Muitas melhorias ainda podem ser adicionadas ao projeto, primeiramente tornando viável o tráfego de dados a longas distâncias, tanto utilizando meio cabeados quanto utilizando protocolos de comunicação wireless.

Outra maneira para agregar mais valor ao projeto seria a implementação de comunicação do usuário administrador via SMS. Onde o mesmo deverá receber uma mensagem de texto quando o aviário encontra-se em estado crítico.

Outra demanda é a implementação do sistema a outros microcontroladores, assim não deixando o projeto amarrado a apenas uma tecnologia de microcontrolador sendo o Arduino utilizado pelo projeto.

REFERÊNCIAS

ABNT. Norma Brasileira 9050: Acessibilidade a edificações, mobiliário, espaços e equipamentos urbanos. 2ª edição. Rio de Janeiro: ABNT, 2004. 97 p.

ARDUINO PLAYGROUND. Arduino playground - Java. 2010. Disponível em: <http://www.arduino.cc/playground/Interfacing/Java> Acesso em: 04 mar. 2011.

ARDUINO. Arduino Uno. 2011. Disponível em: < http://arduino.cc/en/Main/ArduinoBoardUno> Acesso em: 02 abril. 2011.

ATMEL. ATmega8 Datasheet: 2010. 38 p. Disponível em: < http://www.atmel.com/dyn/resources/prod_documents/2486S.pdf> Acesso em: 06 mai. 2011.

FIREBIRD. About Firebird: 2010-2011. Disponível em: < http://www.firebirdsql.org/index.php?op=devel&sub=jdbc&id=aboutjbird

TANENBAUM, Andrew S. Redes de Computadores. Tradução da 4ª Edição. São Paulo: Editora Campos, 2003. 945 p.

DEITEL, Paul; DEITEL, Harvey. Java: Como programar. 4ª edição. New Jersey: Prentice Hall, 2004. 1214 p.

THOMPSON, Marco Aurélio; Java 2 & Banco de dados: Aprenda na prática a usar Java e SQL para acessar bancos de dados relacionais. Editora Érica LTDA: São Paulo, 2002. 198 p.

SALVETTI, BARBOSA, Dirceu Douglas, Lisbete Madsen; Algoritmos. MAKRON Books do Brasil Editora Ltda, 1998. 272 p.

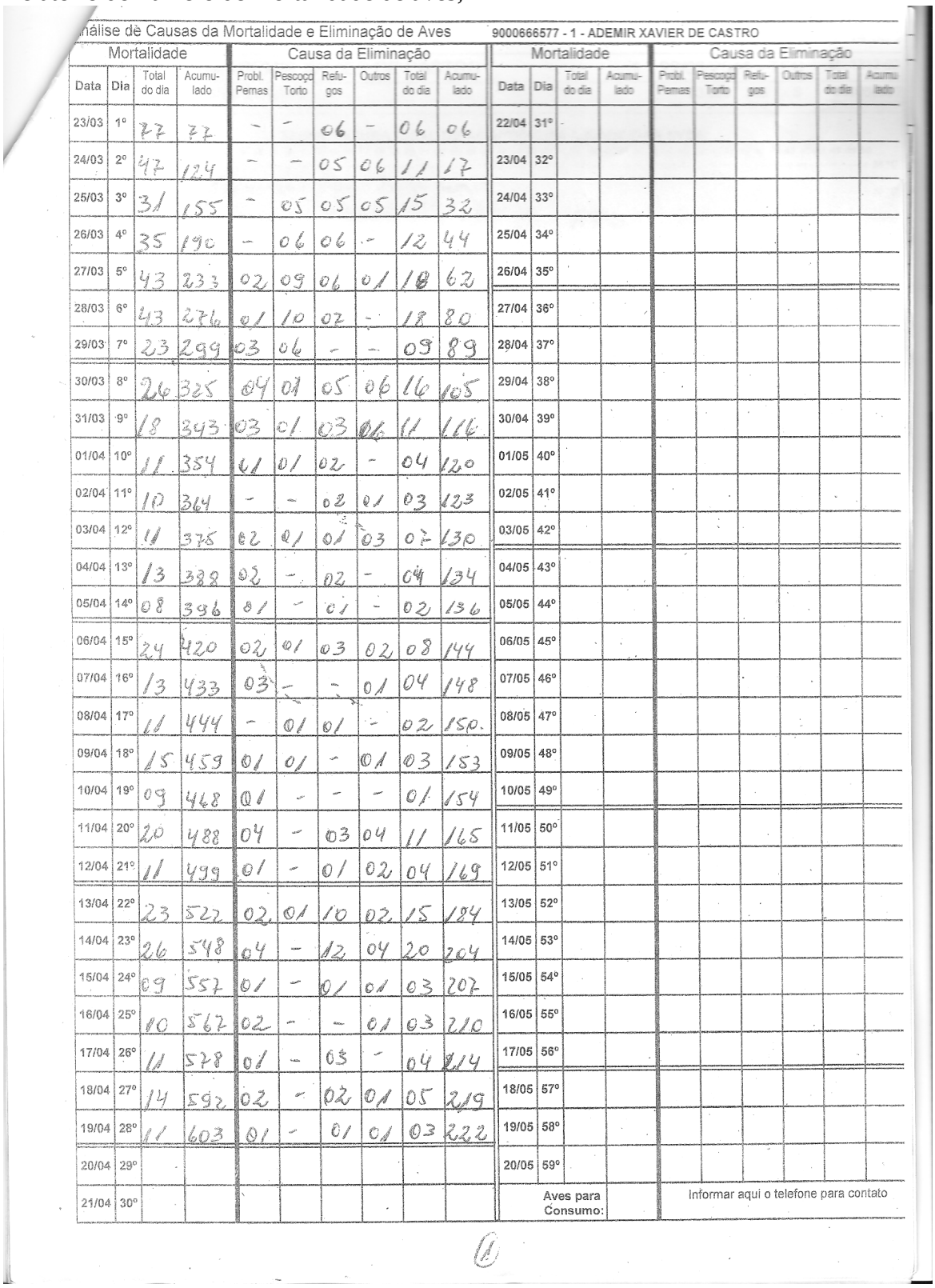
MECENAS, Ivan; JAVA 2 Fundamentos, Swing e JDBC. Editora Alta Books Ltda, 2003. 378 p.

SENSIRION. SHT1x Datasheet Humidity and temperature sensor. Sensirion: 2011. 11 p. Disponível em: <http://www.sparkfun.com/datasheets/Sensors/Datasheet-humidity-sensor-SHT1x.pdf> Acesso em: 08 abril 2011.

SPARKFUN. SHT1x Break-Out Board SHT1x-Breakout-v13. Sparkfun: 2009. 1p. Disponível em: <http://www.sparkfun.com/datasheets/Sensors/Pressure/SHT1x-Breakout-v13.pdf> Acesso em: 08 abril 2011.

ANEXO A – Relatórios e gráficos utilizados pelo usuário final

Relatório do número de mortalidade de aves;



Análise de Causas da Mortalidade e Eliminação de Aves — 9000666577 - 1 - ADEMIR XAVIER DE CASTRO

Mortalidade				Causa da Eliminação					
Data	Dia	Total do dia	Acumulado	Probl. Pernas	Pescoço Torto	Refugos	Outros	Total do dia	Acumulado
23/03	1º	27	27	-	-	06	-	06	06
24/03	2º	47	124	-	-	05	06	11	17
25/03	3º	31	155	-	05	05	05	15	32
26/03	4º	35	190	-	06	06	-	12	44
27/03	5º	43	233	02	09	06	01	18	62
28/03	6º	43	276	01	10	02	-	18	80
29/03	7º	23	299	03	06	-	-	09	89
30/03	8º	26	325	04	01	05	06	16	105
31/03	9º	18	343	03	01	03	01	11	116
01/04	10º	11	354	01	01	02	-	04	120
02/04	11º	10	364	-	-	02	01	03	123
03/04	12º	11	375	02	01	01	03	07	130
04/04	13º	13	388	02	-	02	-	04	134
05/04	14º	08	396	01	-	01	-	02	136
06/04	15º	24	420	02	01	03	02	08	144
07/04	16º	13	433	03	-	-	01	04	148
08/04	17º	11	444	-	01	01	-	02	150
09/04	18º	15	459	01	01	-	01	03	153
10/04	19º	09	468	01	-	-	-	01	154
11/04	20º	20	488	04	-	03	04	11	165
12/04	21º	11	499	01	-	01	02	04	169
13/04	22º	23	522	02	01	10	02	15	184
14/04	23º	26	548	04	-	12	04	20	204
15/04	24º	09	557	01	-	01	01	03	207
16/04	25º	10	567	02	-	-	01	03	210
17/04	26º	11	578	01	-	03	-	04	214
18/04	27º	14	592	02	-	02	01	05	219
19/04	28º	11	603	01	-	01	01	03	222
20/04	29º								
21/04	30º								

Mortalidade				Causa da Eliminação					
Data	Dia	Total do dia	Acumulado	Probl. Pernas	Pescoço Torto	Refugos	Outros	Total do dia	Acumulado
22/04	31º								
23/04	32º								
24/04	33º								
25/04	34º								
26/04	35º								
27/04	36º								
28/04	37º								
29/04	38º								
30/04	39º								
01/05	40º								
02/05	41º								
03/05	42º								
04/05	43º								
05/05	44º								
06/05	45º								
07/05	46º								
08/05	47º								
09/05	48º								
10/05	49º								
11/05	50º								
12/05	51º								
13/05	52º								
14/05	53º								
15/05	54º								
16/05	55º								
17/05	56º								
18/05	57º								
19/05	58º								
20/05	59º								

Aves para Consumo:

Informar aqui o telefone para contato

Relatório do consumo de água;

Controle Diário - Consumo de Água

1360 - Sadia S. A. - Brasília - DF

22/03/2011
14:32:05

CliFor/Nome: 8427100A01-ADEMIR XAVIER DE CASTRO

Nº Contrato: 9000666577

Data	Idade	Volume leitura (L)	Água eliminada(L)	Consumo Diário (L)	Consumo Acumulado(L)
23/03/2011	1	02615	670	284	284
24/03/2011	2	03631	550	466	750
25/03/2011	3	05009	640	738	1488
26/03/2011	4	06509	650	850	2338
27/03/2011	5	08312	830	973	3311
28/03/2011	6	10359	829	1118	4429
29/03/2011	7	12861	1364	1.238	5667
30/03/2011	8	14501	210	1430	7097
31/03/2011	9	17116	982	1628	8725
01/04/2011	10	19521	537	1868	10593
02/04/2011	11	21421	80	1820	12413
03/04/2011	12	23524	75	2028	14441
04/04/2011	13	26041	80	2437	16878
05/04/2011	14	28803	70	2092	18970
06/04/2011	15	30382	80	2099	21069
07/04/2011	16	33065	80	2603	23672
08/04/2011	17	39202	4070	2067	25739
09/04/2011	18	40802	60	1540	27279
10/04/2011	19	42579	70	1707	28986
11/04/2011	20	46550	80	3891	32877
12/04/2011	21	50881	70	4261	37138
13/04/2011	22	55273	60	4832	41970
14/04/2011	23	60636	80	4783	46753
15/04/2011	24	65221	60	4525	51278
16/04/2011	25	69889	80	4588	55866
17/04/2011	26	74240	60	4291	60157
18/04/2011	27	80742	80	6422	66579
19/04/2011	28				
20/04/2011	29				
21/04/2011	30				
22/04/2011	31				
23/04/2011	32				
24/04/2011	33				
25/04/2011	34				
26/04/2011	35				

Data	Idade	Volume leitura (L)	Água eliminada(L)	Consumo Diário(L)	Consumo Acumulado(L)
27/04/2011	36				
28/04/2011	37				
29/04/2011	38				
30/04/2011	39				
01/05/2011	40				
02/05/2011	41				
03/05/2011	42				
04/05/2011	43				
05/05/2011	44				
06/05/2011	45				
07/05/2011	46				
08/05/2011	47				
09/05/2011	48				
10/05/2011	49				
11/05/2011	50				
12/05/2011	51				
13/05/2011	52				
14/05/2011	53				
15/05/2011	54				
16/05/2011	55				
17/05/2011	56				
18/05/2011	57				
19/05/2011	58				
20/05/2011	59				
21/05/2011	60				
22/05/2011	61				
23/05/2011	62				
24/05/2011	63				
25/05/2011	64				
26/05/2011	65				
27/05/2011	66				
28/05/2011	67				
29/05/2011	68				
30/05/2011	69				
Consumo Total					

Volume Leitura - Volume registrado pelo hidrômetro no momento da leitura (Efetuar leitura todo dia no mesmo horário).
Água Eliminada - Água gasta diariamente para limpeza dos bebedouros ou da tubulação do nipple.
Consumo Diário - Volume de água que as aves consumiram no dia (Volume leitura menos Água eliminada).
Consumo - Volume de água que as aves consumiram no lote (Consumo diário mais Consumo acumulado do dia anterior).

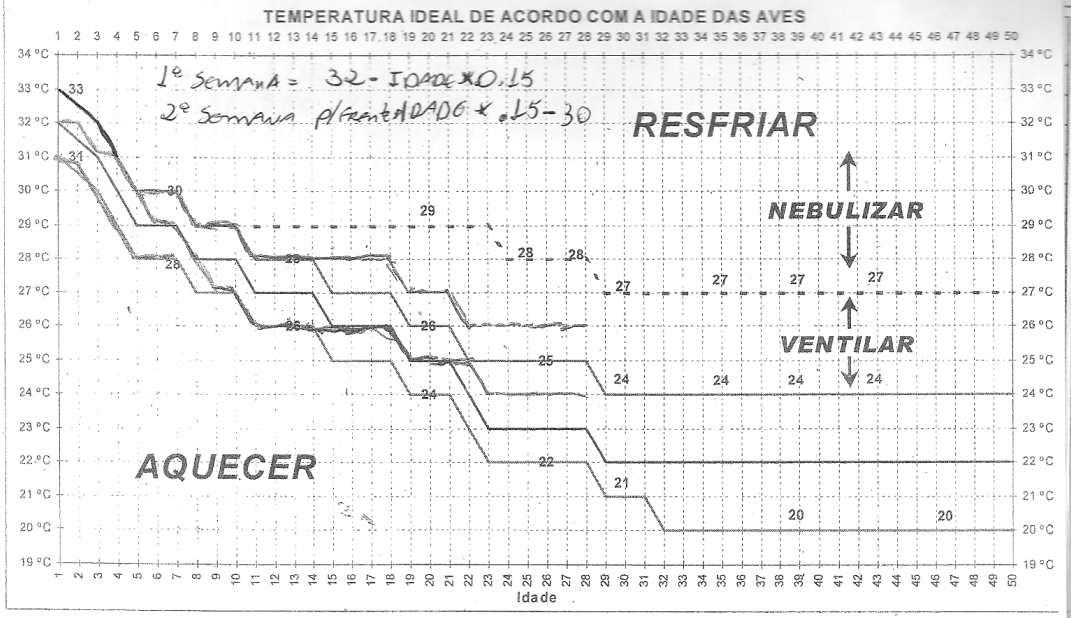
Gráfico do controle de umidade e temperatura;

1360 - Sadia S. A. - Brasília - DF 22/03/2011 14:30:2

8427100A01-ADEMIR XAVIER DE CASTRO 1 22/03/2011 - TER M 1312-02

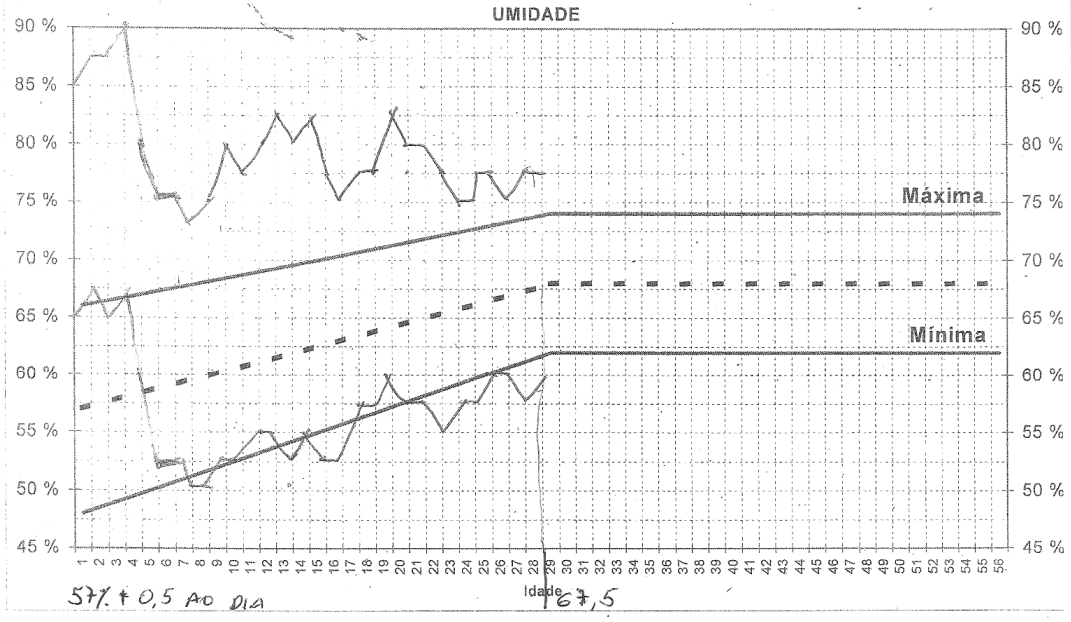
1360-003-ANTONIO CLAUDIO 23.000 45 8 900086657

TEMPERATURA IDEAL DE ACORDO COM A IDADE DAS AVES



Vazão do Bebedouro (ml / minuto)

Idade	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
Vazão	10	50 <----> 60						60 <----> 70							80 <----> 100							100 <----> 120																													



TEMPERATURA E UMIDADE DE CONFORTO DAS AVES