Levantamento de dados através de sensores acoplados ao Arduino

Alan Gilberto Gonçalves
Graduado em Sistemas de Informação pela Libertas Faculdades Integradas
Dorival Moreira Machado Junior
Mestre em Tecnologias da Informação e professor da Libertas Faculdades Integradas

RESUMO

Para a maioria das empresas que utilizam prensas industriais de bojo é extremamente necessário uma melhor analise de controle e utilização. Assim, se torna totalmente viável a qualquer empresa que se preze um melhor rendimento de produção. Este projeto se caracteriza como uma pesquisa experimental, utilizando a experimentação em laboratório como principal método. Consiste em um modo de levantar informações detalhadas de uso de uma prensa industrial de bojo. As informações levantadas poderão facilitar a análise final de uso do equipamento. Para a realização deste projeto, foi necessário uma placa microcontroladora arduino uno com seu ide de programação, o *netbeans* (software necessário para o desenvolvimento das classes em java para comunicação com a porta serial) e banco de dados *mysql*. Toda a coleta de informação foi feita através de um sensor tipo chave, a qual envia o sinal binário para o Arduino, o qual interpreta o sinal "1" como prensa fechada e sinal "0" como prensa aberta. Esta interpretação é feita pelo programa desenvolvido em c/c++ que foi escrito e carregado através de upload (envio) para a placa Arduino via cabo USB. Esta coleta dos dados capturados pelo sensor, só é possível com a utilização da api (biblioteca) rxtxcomm.

Palavras-chave: arduino; comunicação serial; api rxtxcomm; jfreechar, pdfbox.

1 INTRODUÇÃO

Com a chegada de novas tecnologias e a necessidade de aperfeiçoar a produção, as empresas utilizam máquinas de todos os tipos, sendo assim, torna-se necessário repassar as informações de funcionamento diário para a gerência.

Através deste trabalho serão levantadas informações de funcionamento diário, análise de tempo ocioso das máquinas, funcionamento com produção, funcionamento sem produção e quantidade total de utilização do equipamento gerando assim maior controle sobre os

equipamentos em análise pensando no seguinte problema. Como fazer levantamento de uso efetivo em máquinas industriais de prensa utilizando Arduino?

Em fábricas e indústrias é primordial o levantamento e à análise de informações sobre o funcionamento de máquinas para avaliar a quantidade de horas trabalhadas durante um dia. Sendo assim, podemos utilizar tecnologias voltadas à geração de informação para um melhor aproveitamento dos funcionários e do tempo de serviço, evitando que as máquinas fiquem ociosas, aperfeiçoando sua utilização. Lembrando que a principal necessidade de hoje é tecnologias de baixo custo, trabalhar com Arduino garante a economia desde a aquisição das placas e sensores a criação do projeto e sua implantação.

Tem como objetivo geral, avaliar o tempo de funcionamento de máquinas utilizando sensores acoplados ao Arduino, armazenando em banco de dados, gerando gráficos para análise, obtendo uma melhor produtividade, maior ganho e menor custo de produção.

Bem como objetivo específico, a geração de informações para uma melhor análise de uso da prensa e estimular o estudo sobre Arduino.

2 METODOLOGIA

Este trabalho o qual é de natureza quantitativa, se caracteriza como uma pesquisa experimental, utilizando a experimentação em laboratório como principal método.

Para embasamento e formulação dos experimentos, tem-se como método complementar a revisão de literatura, a fim de identificar e denominar todos os elementos envolvidos na pesquisa.

2.1 Montagem do sensor

Para que a coleta de dados pelo arduino seja feita, será necessária uma chave liga desliga com a função de um sensor, este sensor terá que ser acionado ao fechar a tampa da prensa, para que este sensor não seja afetado por interferências externas é necessário realizar a montagem dos componentes que são, um botão do tipo "*push button*" e um resistor de 10 K ohm (símbolo: Ω), interligando-os como o modelo da figura 1.

Dessa forma será possível capturar os dados de estado da prensa de forma binária, ou seja, 1(um) fechada e 0(zero) aberta.

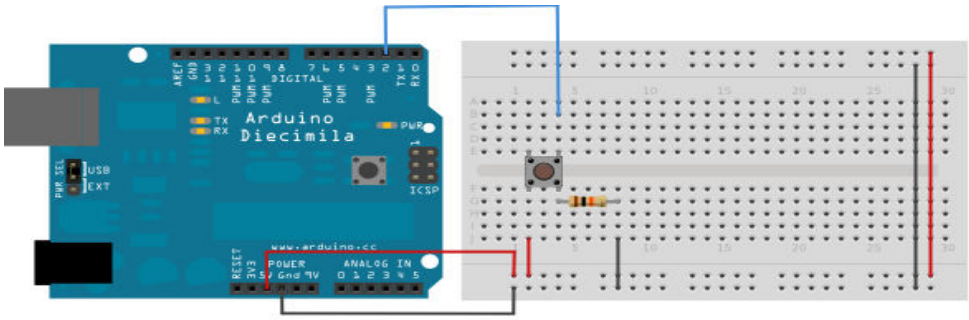

Figura 1 – Disposição dos componentes do sensor

2.2 Instalação do IDE de desenvolvimento do Arduino

Para a instalação do IDE (software para programação) do Arduino é necessário primeiramente que seja feito o download. *A* versão compatível ao seu sistema operacional. Foi utilizada a versão *arduino- 1.5.2-windows.zip* de 121MB, após concluir o download, descompacte para um local de seu gosto, crie um atalho do ícone do aplicativo que está dentro da pasta descompactada, na área de trabalho e pronto, já pode ser executada.

Como padrão, o IDE já inicia uma página para iniciar a programação, a linguagem utilizada para programar o arduino será em C/C++, para isso, será necessário entender as portas que serão utilizadas no projeto, para assim fazer a comunicação com o sensor responsável pela coleta dos dados gerados, foi utilizado às mesmas portas que estão na figura 1.

É possível imprimir a letra “L” ou a “D”, o valor da variável “tempoligado” e “tempodesligado” e o caráter”;” (utilizado para separação de *string's*) em “*println*” diferentes, mais optei por adicionar os três valores à mesma variável e depois imprimir todos os dados em apenas uma *string*, facilitando assim a divisão em substring, esta opção foi tomada tanto para imprimir o valor do tempo da prensa ligada, quanto desligada.

Segue na Figura 2.1 e 2.2 o código para envio da placa arduino.

```
int pushButton = 2; //Criando Botão de leitura Binaria (0 e 1)
int tempo = 0;// Variavel que armazena o tempo ligado, iniciada com valor (0)
```

```
int tempodesligado = 0;// Variavel que armazena o tempo deslidado, iniciada com valor (0)
        void setup() {
        // Inicializar a comunicação serial em 9600 bits por segundo:
        Serial.begin(9600);
        //Define o que cada porta será
        pinMode(pushButton, INPUT);//pushButton(2) sera de INPUT(entrada)
        pinMode(13,OUTPUT);//13 sera de OUTPUT(saida)
        }
        //A rotina de loop é executado repetidamente para sempre:
        void loop() {
        // ler a porta de entrada:
        int bs = digitalRead(pushButton);
                //Compara, se o que entrou pela porta é sinal (1)
                if(bs == 1){
                //Se for, acende o led da porta 13
        digitalWrite(13, HIGH);
                //e soma o tempo que ficou aceso em milisegundos.
                tempo++ ;
                //Compara se o que entrou foi sinal(0)
                }else{
                //Se a variavel tempo for menor que (0)
        if(tempo > 0){
                String stringlig = "L"; //Cria uma variavel "stringlig" do tipo String e inicializa-
o
                com valor "L"
                stringlig += tempo/10;// Variavel stringlig recebe mais o valor da variável
                tempo.
                stringlig += ";";//Adiciona o caracter ";" à variavel "Stringlig" ja agrupada
                anteriormente.
```

Figura 2.1 – Código para envio à placa Arduino

```
Serial.println(stringlig); // Imprimi o valor da variavel stringlig somada a
                valiavel tempo.
                //Variavel tempo é zerada
                tempo = 0;
                //E o led da porta 13, apaga
                digitalWrite(13, LOW); }
        }
        //Compara, se o que entrou pela porta é sinal (0)
        if(bs == 0){
        //Se for, apaga o led da porta 13
        digitalWrite(13, LOW);
                tempodesligado++ ;
        }else{
                if(tempodesligado > 0){
                String stringdesl = "D";//Cria uma variavel "stringdesl" do tipo String e
                inicializa-o com valor "D"
                stringdesl += tempodesligado/10; // Variavel stringlig recebe mais o valor da
                variável tempodesligado.
                stringdesl += ";";//Adiciona o caracter ";" à variavel "Stringdesl" ja agrupada
                anteriormente.
                Serial.println (stringdesl); // Imprimi o valor da variavel stringdesl somada a
                valiavel tempodesligado.
                tempodesligado = 0;
                //E o led da porta 13, acende.
                digitalWrite(13, HIGH);
        }
        }
        //Atrasa 100 milisegundos.
delay(100);}
```

Figura 2.2 – Continuação do Código para envio à placa Arduino

2.3 Instalação do driver Arduino para comunicação com o Windows 8

Para iniciar a transferência do código programado para o arduino com o Upload do IDE de desenvolvimento é necessário que o driver para Arduino uno esteja instalado.

A instalação do driver baixado com o IDE não é compatível com o Windows 8 que também funciona também para Windows 7, e não há para download até a data do término deste Artigo, portanto deve-se forçar um driver de Modem e assim criar e liberar a porta de comunicação COM3.

Na janela "Meu Computador", no item explorar, com o botão direito sobre Computador e selecionado a opção Gerenciar.

O gerenciador de dispositivo é exibido clicando clique com o botão direito em dispositivo desconhecido e selecione atualizar driver.

Na próxima janela aberta, em "Procurar Software de Driver no Computador".

Em "Permitir que eu escolha em uma lista de drivers de dispositivo no computador".

É Selecionado o tipo de dispositivo "Modems" na lista exibida e selecionado o Fabricante "Compaq" e Modelos "Ricochet Wireless USB Modem".

Após isso, é selecionado "Fechar".

Na tela de gerenciamento do computador, com o botão direito no driver do modem instalado é selecionado propriedades.

Na janela aberta na aba modem, em velocidade máxima da porta, o valor "9600" é selecionado.

Na aba avançadas, em configurações avançadas de porta, na janela aberta, em "numero da porta COM", é selecionado COM3.

2.4 Criação do banco de dados em MYSQL

Para iniciar a criação do banco de dados do projeto em *MySql*, é necessário levantar as informações das tabelas, dos atributos e seus respectivos tipos (int, varchar, etc..) que serão salvos, assim, será necessário a instalação do *MySql Workbench*[2], selecionando a plataforma

Windows (plataforma utilizada) e no item (*Windows x 86,32-bit, MSI Instaler*) clique em *download*, após o *download* execute e siga as etapas ate a conclusão da instalação.

Após os atributos levantados de acordo com o projeto, inicialmente será criado uma tabela com os seguintes atributos:

- status - INT [45]
- tempo - INT [11]
- data - Date
- hora - Hora
- relatório – VARCHAR[100]

Com o levantamento pronto, inicie o *mysql workbench*, crie a conexão com o usuário *root* e a senha, clique na conexão com o nome do projeto. Após carregar o editor SQL, Crie um novo banco (*Create a New Squema*), coloque o nome do banco (o projeto foi nomeado como arduino), clique em aplicar e na próxima janela aplicar, ok e fechar. Clique na seta do lado esquerdo do esquema criado, após aberto às opções clique com o botão direito do mouse sobre tabelas (*tables*) e na opção criar tabela (*create table*...), na janela que se abriu coloque o nome da tabela como "dados", logo abaixo clique na aba *columns* e insira os 5 atributos citados acima com seus respectivos tipos e tamanhos, clique em *apply, apply, finish* e *close.*

2.5 Instalação do IDE de desenvolvimento netbeans e jdk

Para a criação das classes Java para a comunicação com o banco, leitura serial, coleta de dados da porta serial, interface do usuário, classe de atributos, e classe de persistência foi utilizado o IDE *Netbeans* para desenvolvimento Java e seu JDK (Kit de desenvolvimento Java), a biblioteca *MySql Connector Java*, para conexão com o Banco de dados *Mysql* , a biblioteca *RXTXcomm* para leitura e escrita serial.

Esta etapa se inicia fazendo o download do pacote JDK/Netbans pelo site da *Oracle*, depois de carregado escolha a versão Windows "x86" ou "x64", selecione o local para salvar, aguarde o final do download, execute e faça a instalação.

2.6 Criação de projeto e instalação do driver MYSQL conector

Depois de instalado o Netbeans se cria um projeto Java, clique em arquivo, novo projeto, Java, Aplicativo Java, próximo, coloque o nome do projeto (Arduino) e clique em finalizar. Para que a comunicação com o banco de dados possa ser feira, é necessária a instalação do driver Mysql Conector assim, faça o download, selecionando o "ZIP Archive" e clicando em Download, salve o arquivo em um local e extraia-o.

No projeto que foi criado clicando em bibliotecas com o botão direito e depois adicionar arquivo Jar/pasta, busque na pasta que foi extraída o arquivo Java mysql-connector-java xxx Bin e clicando em abrir, o driver é instalado no projeto.

2.7 Instalação e configuração da API RXTXCOMM

Agora o próximo passo será instalar a biblioteca (API) RXTXcomm, esta é responsável pela leitura e escrita da porta serial, sem ela não seria possível colher as informações mandadas pelo arduino, que são impressas na porta serial.

Baixe a biblioteca clicando em Binary rxtx-2.1-7-bins-r2.zip(Final), salvando o arquivo. A primeira coisa a se fazer é extrair o arquivo em formato ZIP para dentro de alguma pasta, sugere-se que o arquivo seja descompactado na mesma pasta do JDK, porque irá facilitar o trabalho de manutenção e facilitará a sua instalação.

Nesse projeto a pasta de instalação será *C:\Program file\Java\rxtx-2.1-7-bins-r2*, mas isso irá depender de onde está instalado o seu SDK.

Caso a versão do Windows for 64bits, será utilizada a versão RXTX-2-2-20081207[1], faça o download clicando em Windows-x64, salvando o arquivo e descompactando-a no mesmo local.

O próximo passo é copiar os arquivos *rxtxParallel.dll* e *rxtxSerial.dll* que se encontram dentro da pasta *\Windows\i368-mingw32* que está dentro da pasta de instalação da RXTX (na pasta 32 bits) ou na pasta raiz extraída da versão 64 bits, para a pasta *Windows\System32* que fica dentro da pasta de instalação do Windows.

O local onde será feita a cópia dos arquivos *rxtxParallel.dll* e *rxtxSerial.dll* para a pasta *System32* dentro da pasta de instalação do Windows.

Também é necessário copiar os arquivos *rxtxParallel.dll* e *rxtxSerial.dll* para dentro da pasta de instalação do seu SDK e do seu JRE na pasta */bin*. Vale notar que se você for um

desenvolvedor e instalou o kit de desenvolvimento Java (SDK) irá ter as duas pastas de instalação do JDK e do JRE e precisar copiar os arquivos para as duas bases de instalação, porém se você for apenas um cliente terá somente o JRE instalado, nesse caso basta copiar os arquivos para a base do JRE, nesse caso é necessário copiar os arquivos *rxtxParallel.dll* e *rxtxSerial.dll* tanto para a pasta */bin* como para a pasta */bin/client*.

Agora é necessário copiar o arquivo *RXTXcomm.jar* que está localizado dentro da pasta base de instalação do RXTX para a pasta */lib* tanto do seu SDK quanto do JRE caso conter a pasta.

O procedimento de adicionar a Api RXTXcomm para dentro do projeto é o mesmo do driver MySql Conector, clicando em bibliotecas com o botão direito e depois adicionar arquivo Jar/pasta, busque na pasta que foi extraída o arquivo RXTXcomm e clicando em abrir, o driver é instalado no projeto, tanto a versão 32 quanto a 64 bits.

2.8 Criação da classe de comunicação serial

Com a biblioteca adicionada ao projeto, é necessária a criação das classes Java, crie um pacote Java, clicando com o botão direito sobre o pacote de códigos fonte e nomeando-o como desejar, a classe de comunicação serial será nomeada como “SerialComm” e deverá ser criada dentro do pacote que foi criado e nomeado anteriormente, realizando todos os imports e adicionando os métodos da Figura 3.1 e 3.2.

```
public class SerialComm implements SerialPortEventListener {
InputStream inputStream;
private String resultado = "";
public String getResultado() {
return resultado;
}
        public void setResultado(String resultado) {
        this.resultado = resultado;
        }
        public void execute() {
        String portName = getPortNameByOS();
```

```
CommPortIdentifier portId = getPortIdentifier(portName);
if (portId != null) {
        try {
        SerialPort serialPort = (SerialPort) portId.open(this.getClass().getName(),
2000);
        inputStream = serialPort.getInputStream();
        serialPort.addEventListener(this);
        serialPort.notifyOnDataAvailable(true);
        serialPort.setSerialPortParams(
        9600,
        SerialPort.DATABITS_8,
        SerialPort.STOPBITS_1,
        SerialPort.PARITY_NONE);
        } catch (PortInUseException e) {
        } catch (IOException e) {
```

Figura 3.1 – Classe SerialComm

```
        } catch (UnsupportedCommOperationException e) {
        e.printStackTrace();
        } catch (TooManyListenersException e) {
        }
        } else {
        System.out.println("Porta Serial não disponível");
        }}
        public String getPortNameByOS() {
                String osname = System.getProperty("os.name", "").toLowerCase();
                        if (osname.startsWith("windows")) {
                                // windows
                                return "COM3";
                        } else if (osname.startsWith("linux")) {
                                // linux
                                return "/dev/ttyS0";
```

```
                    } else if (osname.startsWith("mac")) {
                        // mac
                        return "???";
                    } else {
                        System.out.println("Desculpe Sistema não suportado");
                        System.exit(1);
                        return null;
                    }}
            public CommPortIdentifier getPortIdentifier(String portName) {
            Enumeration portList = CommPortIdentifier.getPortIdentifiers();
            Boolean portFound = false;
            while (portList.hasMoreElements()) {
            CommPortIdentifier portId = (CommPortIdentifier) portList.nextElement();
            if (portId.getPortType() == CommPortIdentifier.PORT_SERIAL) {
                System.out.println("Available port: " + portId.getName());
                if (portId.getName().equals(portName)) {
                System.out.println("Found port: " + portName);
                portFound = true;
                return portId;
            }}}return null;
}
            public void serialEvent(SerialPortEvent event) {
            switch (event.getEventType()) {
                case SerialPortEvent.BI:
                case SerialPortEvent.OE:
                case SerialPortEvent.FE:
                case SerialPortEvent.PE:
                case SerialPortEvent.CD:
                case SerialPortEvent.CTS:
                case SerialPortEvent.DSR:
                case SerialPortEvent.RI:
                case SerialPortEvent.OUTPUT_BUFFER_EMPTY:
```

```
                    break;
                    case SerialPortEvent.DATA_AVAILABLE:
                    byte[] readBuffer = new byte[20];
                            try {
                            int numBytes = 0;
                            while (inputStream.available() > 0) {
                            numBytes = inputStream.read(readBuffer);
                            }
                            String result = new String(readBuffer);
                            result = result.substring(0, numBytes);
                            String validos = ";DL0123456789";
                                    for (int i = 0; i < result.length(); i++) {
                                            if (validos.contains(result.substring(i, i + 1))) {
                                            resultado = resultado + result.substring(i, i + 1);
                                            System.out.println(resultado);
}}}
 catch (IOException e) {
}
        break;
}}}
```

Figura 2.3 – Classe SerialComm Continuação

2.9 Criação da classe de atributos

Os atributos da classe “Dados” que será criada na mesma forma e pacote anterior são:

- Status – Armazena o valor de prensa, ligado ou desligado.
- Tempo – Armazena o tempo do status em segundos.
- Data – Armazena a data que foi captada o funcionamento.
- Hora – Armazena a hora que foi captada o funcionamento.
- Relatório – Armazena o relatório dos estados que a máquina foi utilizada no decorrer do tempo.

Serão programados de forma que comuniquem com a classe de persistência e com o banco de dados que foi criado, pois cada atributo é de um tipo e todos eles devem ser encapsulados, não se esquecendo do construtor Dados passando todos os parâmetros, criando seus get's e set's e importando todos os pacotes necessários para a construção da classe.

2.10 Criação da classe de conexão com o banco

Alguns métodos de conexão ao banco também são necessários, a classe será chamada de "ConexaoJDBC", será responsável pelos métodos, abreConexao, executaSQL, executaConsulta, prepare e fechaConexao, o código esta no Anexo 4 e o método abreconexao, necessita de atenção.

O código em negrito será onde a conexão ao banco será realizada, seguindo os padrões da conexão que foi realizada no *MySql workbench*, como o local onde o banco se encontra e seu respectivo nome do projeto, o usuário e a senha do banco que foi criada, no caso desse projeto, o usuário(user) será "root" e a senha(password) "123456" essa senha é definida na criação da conexão ao banco, portanto, siga a conexão que foi criada no seu projeto e ajuste os métodos da Figura 4 a sua String JDBC.

```
public class ConexaoJDBC {
public static Connection conn;
private static PreparedStatement pst;
private static ResultSet rs;

public static void abreConexao() {
        String driver = "com.mysql.jdbc.Driver";
        String url = "jdbc:mysql://localhost:3306/arduino";
        String user = "root";
        String password = "123456";
        try {
        Class.forName(driver);
        conn = DriverManager.getConnection(url, user, password);
        System.out.println("Conexão realizada");
```

```
                } catch (Exception erro) {
                        System.out.println("Erro: " + erro.getMessage());
                }
        }
public static void executaSQL(String sql) {
        try {
        pst = conn.prepareStatement(sql);
        pst.executeUpdate();
        System.out.println("SQL:insert, update ou delete realizado");
        } catch (Exception erro) {

        System.out.println("Erro: " + erro.getMessage());
        }
}
public static ResultSet executaConsulta(String sql) {
        try {
        pst = conn.prepareStatement(sql);
        rs = pst.executeQuery();
                System.out.println("SQL: select realizado");
                return rs;
        } catch (Exception erro) {
                System.out.println("Erro: " + erro.getMessage());
                return null;
        }
}
public static PreparedStatement prepare(String sql) {
        try {
                return conn.prepareStatement(sql);
        } catch (Exception erro) {
                System.out.println("Erro: " + erro.getMessage());
```

Figura 4.1 – Classe ConexaoJDBC

```
                return null;
        }
}
public static void fechaConexao() {
        try {
                if (rs != null) {
                        rs.close();
        }
                if (pst != null) {
                        pst.close();
        }
                if (conn != null) {
                        conn.close();
}
        System.out.println("Conexão fechada!");
        } catch (Exception erro) {
                System.out.println("Erro " + erro.getMessage());
        }
}
}
```

Figura 2.4 – Classe ConexaoJDBC Continuação

2.11 Criação da classe para gerar relatórios em imagens PDF

Esta classe será responsável por criar relatórios, para que isso aconteça é necessário baixar a API pdfbox-app-1.8.1.jar responsável por gerar o arquivo PDF, selecionando o item abaixo de HTTP e salvando em um local de sua preferência.

Crie uma nova classe Java chamada “GerarPdf”, Figura 5 no mesmo pacote que foi criado anteriormente, clicando em bibliotecas com o botão direito e depois adicionar arquivo Jar/pasta, busque na pasta que foi salvo o arquivo e clicando em abrir, o driver é instalado no projeto.

É necessário criar um arquivo em branco onde o relatório que foi criado será salvo, ou seja, em **C:/Relatórios/Relatório Diário.pdf,** assim ao clicar em Gerar relatório o arquivo com os dados coletados serão salvos neste arquivo.

```
public class GerarPdf {
    public void Executa(Tela tela) throws IOException, COSVisitorException {
        String RelatorioPdf = tela.getTotal();
        TextToPDF textToPDF = new TextToPDF();
        String novoTexto = "Relatorio Diario" + "\n" + "\n" + RelatorioPdf;
        StringReader sReader = new StringReader(novoTexto);
        PDDocument novoPdDocument = textToPDF.createPDFFromText(sReader);
        novoPdDocument.save("C:/Relatórios/Relatório Diário.pdf");
    }
}
```

Figura 2.5 – Classe GerarPdf

2.12 Criação da classe de persistência

Como padrão de programação Java, as classes DAO são responsáveis pela persistência de dados, ou seja, através dessa classe, todos os atributos da classe Dados, são salvos no banco.

Os atributos da classe Dados, são preparados e salvos de modo correto em uma classe DadosDao.java, dentro dessa classe são implementados os métodos chamados CRUD (Criar , ler , atualizar e deletar um objeto).

Para criar essa classe, cria-se um novo pacote Java e neste pacote, uma nova classe Java chamada “DadosDao”.

A classe foi implementada apenas com o método inserir, ou seja, o projeto só irá salvar no banco, os dados coletados.

Deve se observar com atenção, a palavra dados em negrito é onde se insere o nome da tabela que foi criada no banco para salvar os dados, de acordo com a Figura 6

```
public class DadosDao {
```

```
public void inserir(Dados c) throws SQLException {
        ConexaoJDBC.abreConexao();
        String sql = "insert into dados values (?,?,?,?,?)";
        try {
                PreparedStatement pst = ConexaoJDBC.prepare(sql);
                pst.setString(1, c.getCodstatus());
                pst.setInt(2, c.getTempo());
                pst.setDate(3, c.getData());
                pst.setTime(4, c.getHora());
                pst.setString(5, c.getRelatorio());
                pst.executeUpdate();
                System.out.println("Inserção realizada");
        } catch (Exception erro) {
                System.out.println("Erro " + erro.getMessage());
        }
    ConexaoJDBC.fechaConexao();
}}
```

Figura 2.6 – Classe DadosDao

2.13 Criação da classe de interface do usuário

Esta classe é a mais complexa, portanto é necessária atenção redobrada para que não haja nenhum erro de compilação.

A classe de exibição dos dados coletados para o usuário será chamada de “Tela”, para sua criação, clique no pacote criado anteriormente com o botão direito selecione, Novo e depois formulário JFrame, para criar a interface, basta utilizar o painel direito “paleta” que foi carregado após concluir a criação da classe, clicando sobre o item desejado e arrastando-o para o local desejado, também é possível dimensiona-lo clicando na diagonal do objeto criado. Será de grande ajuda acompanhar o desenho da interface abaixo e também, renomear as variáveis de campo de texto, áreas de texto, botões e o painel que fica posicionado logo abaixo do botão “Exibir Gráficos”, segundo figura 7.

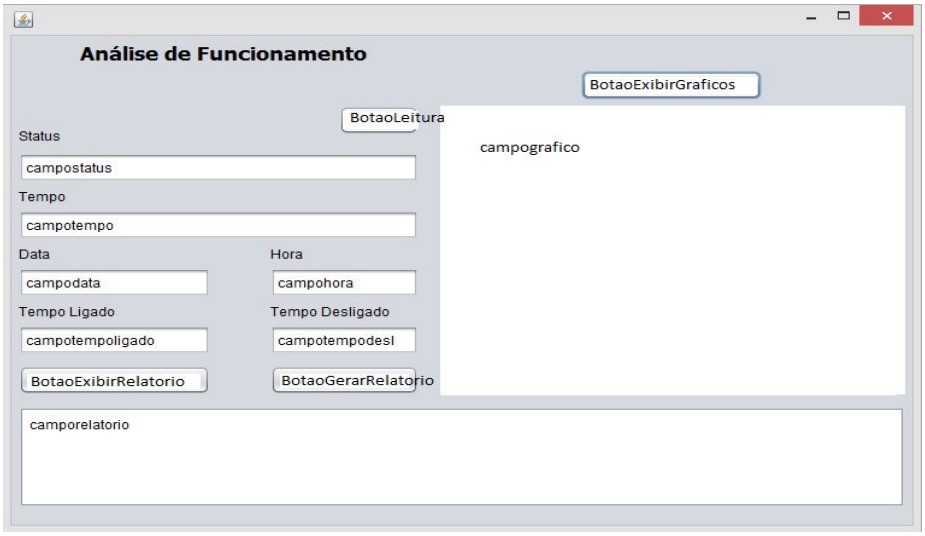

Figura 2.7 – Tela com nomes de variável dos campos.

Feito isso não há mais nenhuma alteração visual, apenas será necessário inserir os atributos e métodos da classe Tela.java, para inserí-los clique no campo código-fonte, logo acima da interface criada, os códigos criados são fruto da parte gráfica, ou seja, ao arrastar e criar os rótulos, campos, botões e painéis seus respectivos códigos são adicionados automaticamente.

Logo abaixo onde a classe é declarada, insere-se os objetos, atributos e os get's, set's que serão utilizados nos métodos e na exibição dos dados para o usuário que são: "SerialComm serial = new SerialComm", "Tela t = Tela.this" , "String Total", "String status", "String relatório", "String relatório_banco", "int tempodesl = 0", "int tempolig = 0" , "int tempo", "Time hour" , "Date dt" , seus gets e sets respectivamente não esquecendo dos import's.

A parte onde os primeiros métodos que serão executados é instanciado, ficam dentro de "initComponentes()" , o método "serial.execute()" é instanciado para ser executado junto com a classe principal.

Esses métodos são chamados para validar a porta "COM3", verificando se ela é válida e se esta liberada para leitura, caso a placa do arduino não esteja conectada, aparecerá a mensagem de "Porta serial não disponível".

Agora será implementado o método ler, Figura 8, este método é responsável pela leitura dos dados da porta serial coletados pela classe Serialcomm , exibir os dados na interface visual e também por salvar estes dados no Banco de Dados .

```
public void Ler() {
      String resultado = serial.getResultado();
      serial.setResultado("");
      String vetor[] = resultado.split(";");
      for (int i = 0; i < vetor.length; i++) {
      String res = vetor[i];
        if ("D".equals(res.substring(0, 1))) {
        campostatus.setText("Desligado");
        status = "Desligado";
        campotempo.setText(res.substring(1, res.length()));
        tempo = Integer.parseInt(res.substring(1, res.length()));
        tempolig += Integer.parseInt(res.substring(1, res.length()));
        campotempodesl.setText(Integer.toString(tempolig));
        } else {
           if ("L".equals(res.substring(0, 1))) {
             campostatus.setText("Ligado");
        status = "Ligado";
        campotempo.setText(res.substring(1, res.length()));
                tempo = Integer.parseInt(res.substring(1, res.length()));
        tempodesl += Integer.parseInt(res.substring(1, res.length()));
        campotempoligado.setText(Integer.toString(tempodesl));
           } else {
           }
        }
        GregorianCalendar gc = new GregorianCalendar();
        int hora = gc.get(gc.HOUR_OF_DAY);
        int minuto = gc.get(gc.MINUTE);
        int segundo = gc.get(gc.SECOND);
        hour = new Time(hora, minuto, segundo);
        int dia = gc.get(gc.DAY_OF_MONTH);
        int mes = gc.get(gc.MONTH) + 1;
        int ano = gc.get(gc.YEAR);
```

```
        dt = new Date((ano - 1900), mes, dia);
        campohora.setText(Integer.toString(hora) + ":" + Integer.toString(minuto) + ": " +
Integer.toString(segundo));
        campodata.setText(Integer.toString(dia) + "/" + Integer.toString(mes) + "/" +
Integer.toString(ano));
        relatorio = ("Status - " + campostatus.getText() + " / " + " Tempo - " +
campotempo.getText() + " segundos / " + " Data - " +
campodata.getText() + " / " + " Hora " + campohora.getText());
        Total += relatorio + "\n";
        relatorio_banco = ("Status - " + status + " / Tempo - " + Integer.toString(tempo) + " /
Data - " + Integer.toString(dia) + "/" + Integer.toString(mes) + "/" + Integer.toString(ano) + "
/ Hora - " + Integer.toString(hora) + ":" + Integer.toString(minuto) + ": " +
Integer.toString(segundo));

Dados d = new Dados();
d.setCodstatus(status);
d.setTempo(tempo);
d.setData(dt);
d.setHora(hour);
d.setRelatorio(relatorio_banco);
        DadosDao Dd = new DadosDao();
        try {
           Dd.inserir(d);
        } catch (SQLException ex) {
           Logger.getLogger(Tela.class.getName()).log(Level.SEVERE, null, ex);
        }
     }
  }
```

Figura 2.8 – Método Ler

Os métodos da Figura 2.9 serão responsáveis por gerar o gráfico com os dados de tempo total ligado e desligado, estes métodos trabalham em conjunto sendo necessário que os dois sejam criados.

```
public CategoryDataset createDataset() {
        DefaultCategoryDataset dataset = new DefaultCategoryDataset();
        dataset.addValue(t.getTempolig(), "Tempo Desligado", "Dia/Semana");
        dataset.addValue(t.getTempodesl(), "Tempo Ligado", "Dia/Semana");
        return dataset;
}
 public void criaGrafico() {
        CategoryDataset cds = createDataset();
        String titulo = "Gráfico de Tempo Total";
        String eixoy = "Tempo/seg";
        String txt_legenda = "Legenda:";
        boolean legenda = true;
        boolean tooltips = true;
        boolean urls = true;
        JFreeChart graf = ChartFactory.createBarChart3D(titulo, txt_legenda, eixoy, cds,
        PlotOrientation.VERTICAL, legenda, tooltips, urls);
        ChartPanel myChartPanel = new ChartPanel(graf, true);
        myChartPanel.setSize(campografico.getWidth(), campografico.getHeight());
        myChartPanel.setVisible(true);
        campografico.removeAll();
        campografico.add(myChartPanel);
        campografico.revalidate();
        campografico.repaint();}
```

Figura 2.9 – Métodos para Gerar Gráfico

Esses métodos são de classes específicas de Api's responsáveis por criar e exibir gráficos e também trabalham em conjunto, faça os import's necessários para que funcionem sem erro, é necessário a Api jfreechart-1.0.13.jar e a Api jcommon-1.0.17.jar e as adicione ao projeto. Os métodos estão implementados, agora é necessário adicionar o evento a cada botão, na classe tela logo acima da interface criada, clique em "Projeto", será exibida a

interface com os botões e campos implementados clique duas vezes sobre o botão "ExibirGraficos" o evento do botão será aberto, escreva o método "t.criaGrafico();", este método executara o código responsável por desenhar o gráfico com os valores de tempo total, ligado e desligado, clique agora no botão Leitura, escreva o método "t.Ler();", este método executará a exibição dos dados coletados e salvará as informações no banco. Agora sobre o botão "Exibir Relatório", insira o comando "camporelatorio.setText(Total);", este comando exibira no camporelatorio, todas as strings preparadas no método ler, uma abaixo da outra, como um relatório e agora no botão "Gerar Relatório" e insira no seu método da Figura

```
@SuppressWarnings("LocalVariableHidesMemberVariable")
    GerarPdf t = new GerarPdf();
    try {
       try {
          t.Executa(this);
       } catch (Exception e) {
          Logger.getLogger(Tela.class.getName()).log(Level.SEVERE, null, e);
       }
    } catch (Exception ex) {
       Logger.getLogger(Tela.class.getName()).log(Level.SEVERE, null, ex);
    }
```

Figura 2.10 – Ação do botão Gerar Pdf

4 CONSIDERAÇÕES FINAIS

Com o desenvolvimento desse projeto é possível perceber que a necessidade de análise de funcionamento de prensa é essencial para um bom aproveitamento de mão de obra, todas as informações são de grande importância para a gerência de uma empresa, principalmente com informações diárias de utilização de prensas de bojo.

O trabalho realizado com a placa arduino é interessante, mais merecem alguns cuidados para remover as interferências de leitura de porta serial, qualquer caráter impresso erroneamente na porta serial pode acarretar falhas de leitura e inverdades nas informações.

A utilização das bibliotecas foi essencial para poupar tempo, e seu funcionamento foi impressionante, pois, a facilidade na criação dos códigos não deixou a desejar.

A geração de relatórios específicos acessando as informações no banco de dados ficará para projetos futuros.

REFERENCIAS

ARDUINO, **Arduino**, disponível em <http://www.arduino.cc>, acessado em 01/03/2013 as 8h00.

CENAPAD, **Cenapad**, disponível em <http://www.cenapad.unicamp.br/servicos/treinamentos/apostilas/apostila_C.pdf>, acessado em: 10 de Abril de 2013.

CENTRODECOMPUTAÇÃOUNICAMP, **Centrodecomputaçãounicamp**, disponível em http://www.fsc.ufsc.br/~canzian/root/tutorial-c-unicamp.pdf, acessado em: 08 de Março de 2013.

DEITEL, Paul; DEITEL, Harvey. **Java Como Programar** . São Paulo: Pearson, 2009.

DEITEL, Paul; DEITEL, Harvey. **Java Como Programar** . São Paulo: Prentice Hall, 2005.

FERRARI, Fabricio Augusto. **Criando Banco de Dados em Mysql.** São Paulo: Digerati Books, 2007.

FROYD, Thomas L. **Sistemas Digitais Fundamento e Aplicações.** Porto Alegre: ArtMed, 2007

KRIS Jamsa e LARS Klander. **Programando em C/C++ "A Bíblia"**. São Paulo: Makron, 1999.

LABDEGARAGEM, **Labdegaragem,** disponível em <http://www.labdegaragem.org>, acessado em 03/03/2013 as 22h00.

MASSIMO, Banzi. **Primeiros Passos com o Arduino**. São Paulo: O'Reilly Novatec, 2011.

MCROBERTS, Michael. **Arduino Básico.** São Paulo: Novatec, 2011.

SILVEIRA, João Alexandre da. **Experimentos com Arduino.** São Paulo: Ensino Profissional, 2011.

SILVA, Renato A. **Programando Microcontroladores PIC : Linguagem "C"** São Paulo : Ensino Profissional, 2006.