Universidade Federal de Campina Grande

Centro de Engenharia Elétrica e Informática

Curso de Graduação em Engenharia Elétrica

DÉBORA DINIZ DE MELO

UTILIZAÇÃO DA PLATAFORMA ARDUINO PARA AQUISIÇÃO DE DADOS

Campina Grande, Paraíba.
Maio de 2013

DÉBORA DINIZ DE MELO

UTILIZAÇÃO DA PLATAFORMA ARDUINO PARA AQUISIÇÃO DE DADOS

Trabalho de Conclusão de Curso submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Automação

Orientador:
Professor Alexandre Cunha Oliveira, Dr.

Campina Grande, Paraíba.
Maio de 2013

DÉBORA DINIZ DE MELO

UTILIZAÇÃO DA PLATAFORMA ARDUINO PARA AQUISIÇÃO DE DADOS

Trabalho de Conclusão de Curso submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Automação

Aprovado em / /

Professor Avaliador
Universidade Federal de Campina Grande
Avaliador

Professor Alexandre Cunha Oliveira, Dr.
Universidade Federal de Campina Grande
Orientador, UFCG

Dedico este trabalho a todos familiares e amigos, que me trouxeram até aqui.

AGRADECIMENTOS

À Deus, por ter me dado toda a perseverança necessária para que eu concluísse este curso.

Aos meus pais, em especial à minha querida mãe, Cleide Alves, por fazerem da minha existência e da minha felicidade elementos motrizes de suas vidas.

Ao meu esposo e inspiração sempre, Thiago de Freitas, pelo encorajamento, cumplicidade, compreensão, confiança e apoio.

Às minhas irmãs, por comporem cada singelo elemento da minha formação como irmã, filha, tia, esposa e Engenheira.

A todos os amigos, em especial, ao amigo e companheiro de estudos Miquéias Melo, por ter em incansáveis momentos, me transferido, pacientemente, parcela de seu conhecimento.

Ao professor Alexandre Cunha Oliveira, por toda sua presteza e solicitude na orientação deste trabalho.

"A simplicidade é a sofisticação máxima."

Leonardo da Vinci.

RESUMO

O ambiente industrial está em constante mudança ao passo das necessidades surgidas e a incorporação de novas tecnologias. Porém, nem sempre a resposta mais viável a essa problemática é a substituição de maquinário, ou mesmo um longo, complexo e caro pedido de customização aos fabricantes. Muitas vezes, os equipamentos instalados em uma planta possibilitam a expansão de suas funcionalidades e incorporação de novas tecnologias de forma prática e barata. Nesse contexto, pode-se inserir como alternativa, sistemas de aquisição de dados como o Arduino.

Desse modo, com vistas ao melhoramento, expansão do estudo em pauta e a avaliação dos recursos da plataforma Arduino para implementação de um sistema de aquisição de dados, desenvolveu-se um protótipo com indicação de uso em um cenário com características industriais.

Palavras-chave: Arduino, Aquisição de Dados, Tear, Indústria.

ABSTRACT

The industrial environment is constantly changing, with new requirements that arise whereas new technologies are incorporated. However, not always the most viable answer to this problem is the replacement of machinery, or even a long, complex and expensive application customization by the manufacturers. Often, the equipment installed in a plant allows the expansion of its functionality and the incorporation of new technologies in a practical and inexpensive procedure. In this context, systems such as Arduino can be inserted as an alternative data acquisition device.

Thus, aiming to improve, to expand the study agenda and evaluate Arduino as a platform to implement data acquisition systems, we developed a prototype that is suitable to a scenario with industrial characteristics.

Keywords: Arduino, Data Acquisition, Loom, Industry.

LISTA DE FIGURAS

LISTA DE TABELAS

ÍNDICE

1 INTRODUÇÃO

Os microcontroladores têm surgido como alternativa viável para o desenvolvimento de soluções em âmbito industrial, alcançando-se o bom desempenho na eficiência de energia, redução de custo e espaço, suporte para alta velocidade em comunicação, atendendo-se satisfatoriamente à demanda de transformação exigida pelo ambiente.

Todavia, dado o ambiente hostil e ruidoso, alguns cuidados especiais devem ser tomados quando se utilizam microcontroladores para automatizar processos industriais, tais como: tolerância de *spikes* intensos de corrente, interferências eletromagnéticas e descargas eletrostáticas. A mitigação desses fatores de perturbação possibilita uma operação confiável e segura do sistema.

Paralelamente à evolução e usabilidade de microcontroladores em ambientes industriais, deu-se, em 2005, o desenvolvimento de uma plataforma dotada de microcontrolador, voltada para a prototipagem eletrônica, utilizada, inicialmente, para a elaboração de protótipos simples e de prova de conceito, o Arduino, uma plataforma de hardware e software, com bibliotecas *open-source*, criado com o objetivo de permitir o desenvolvimento de controle de sistemas interativos de baixo custo.

Dado sua viabilidade, o Arduino tem sido utilizado, nos últimos anos, nas mais diversas aplicações, sendo, porém, ainda evitado seu uso em ambiente industrial, devido à limitação de robustez no tratamento das problemáticas já citadas, inerentes ao ambiente. Contudo, mesmo dentro do ambiente industrial, vários subprocessos podem ser implementados, lançando-se mão das funcionalidades e facilidades dessa plataforma.

Esse trabalho propõe-se, então, a utilizar as potencialidades da plataforma, no desenvolvimento de protótipo de sistema para acompanhamento do *status* de funcionamento de uma máquina, em que os sinais já oferecem tratamento prévio pela própria máquina, sendo suficiente para tal o conhecimento do sistema, não conferindo ao Arduino, nenhuma característica desvantajosa à implementação.

O Arduino, Versão Uno, eleito para a prototipagem, é composto basicamente por um microcontrolador ATmega328 com quatorze pinos de entrada/saída digital, das

quais seis podem ser utilizadas como saídas PWM e 6 entradas analógicas; além de um cristal oscilador de 16MHz; uma conexão USB; uma entrada de alimentação; uma conexão ICSP e um botão de reset.

2 SISTEMAS DE AQUISIÇÃO DE DADOS

Os processos desempenhados nas áreas indústrias, científicas, médicas, entre outras, dependem fortemente do conhecimento e monitoramento de grandezas físicas associadas aos processos [1]. Desse modo, a aquisição de dados tem sido, com o advento dos microcontroladores, de essencial importância para o conhecimento, análise, controle e melhoramento desses processos, tornando-se cada vez mais utilizado nessas áreas.

As características necessárias a um sistema de aquisição de dados são ditadas pelo processo do qual se deseja adquiri-los, tal como as necessidades do usuário do sistema [1]. No exemplo prático, abordado em tópicos à frente, escolhe-se o tipo de sistema de aquisição de acordo com os requisitos da aplicação, avaliando-se critérios como: quantidade de variáveis a serem analisados, método de apresentação dos dados, etc.

2.1 DEFINIÇÃO

Um sistema de aquisição de dados é composto por um conjunto de elementos capazes de fornecer informações detalhadas a cerca do processo, permitindo, dessa forma, a aquisição, análise e apresentação de seus dados, pondo-se entre o processo e o observador (Figura1).

Figura 1. Sistema de aquisição de dados.
(Fonte: FRANÇA, 1997, p. 18).

Os elementos que constituem um sistema de aquisição de dados, ou bloco funcional, são basicamente: transdutor, condicionador, aquisição de dados, análise de dados e apresentação de dados (Figura 2).

Figura 2. Blocos funcionais de um sistema de aquisição de dados.
(Fonte: FRANÇA, 1997, p. 18).

2.1.1 TRANSDUTOR

O transdutor é o componente responsável por converter variações de uma determinada grandeza física em outra, que por sua vez possa ser utilizada pelo sistema de aquisição de dados. Em geral, as grandezas físicas são convertidas em sinais elétricos adequados a aplicação em sistemas de aquisição de dados, nesse processo de conversão, podem ser usados sensores ativos ou passivos. Os sensores ativos são responsáveis pela detecção do evento e geração direta do sinal elétrico. Enquanto nos sensores passivos, alguns dos seus parâmetros intrínsecos variam em função de variações da grandeza medida. Esta variação paramétrica pode então ser convertida em informação mensurável através de circuitos de condicionamento apropriados.

Na aplicação prática, abordada neste trabalho, não foi necessário dotar o sistema de sensoriamento, um sensoriamento ativo já se fazia existente, como descrito nos tópicos adiante, sendo necessária apenas a interpretação da informação entregue por esses sensores.

2.1.2 CONDICIONADOR

Uma vez que o evento já tenha sido traduzido em sinais elétricos, se faz necessário tratar esses sinais para que os mesmos sejam compatíveis com o método de aquisição eleito, ou mesmo adaptá-los para que os mesmos possam ser interpretados corretamente. Desse modo, podemos utilizar os condicionadores de sinais para: amplificar ou atenuar, deslocar nível médio, detectar pico, nível médio ou valor eficaz, linearizar, filtrar e/ou isolar o sistema para uma medição mais exata e segura.

Novamente, no exercício prático abordado, se fez suficiente apenas a atenuação do sinal, não sendo necessários os demais tratamentos dos mesmos.

2.1.3 AQUISIÇÃO, ANÁLISE E APRESENTAÇÃO DOS DADOS

O 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 E/S(Entrada/Saída), além de outros possíveis periféricos, tais como módulos de temporização e conversores A/D, que integrados em um mesmo componente permitem o constante acesso e controle de todos os periféricos (Figura 3) [2].

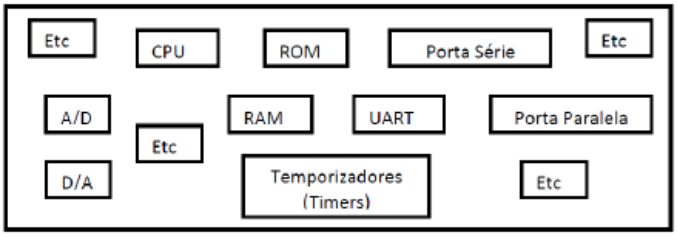


Figura 3. Elementos constituintes de um microcontrolador.
(Fonte: DINIS, 2010, p. 36).

Com o conjunto de periféricos que disponibiliza, os microcontroladores podem se adequar às mais diversas aplicações de aquisição de dados e controle de sistemas, possuindo alto desempenho, sendo assim, uma plataforma bastante utilizada no projeto de sistemas de aquisição de dados.

Na última década, esforços têm sido empregados na elaboração de plataformas de Hardware e Software dotadas de microcontroladores e que permitissem a aquisição, análise e apresentação de dados incorporados em um só sistema, o Arduino é um exemplo de uma dessas plataformas.

3 APRESENTAÇÃO DA PLATAFORMA ARDUINO

Em meados de 2005 o Professor Massimo Banzi, ministrando aulas no Instituto de Design de Interação Ivrea, na Itália, sentiu a necessidade de estimular seus alunos a colocarem em prática o conhecimento adquirido em sua disciplina de Design da Integração, cuja filosofia baseava-se no desenvolvimento de protótipos de artefatos

interativos, aplicando-se conceitos de observação das experiências e testes com usuários.

Banzi, amante de longa data da Eletrônica, se propôs então a utilizar a computação física em suas aulas, empregando a eletrônica para o desenvolvimento da mencionada prototipagem de novos materiais. Porém, seu grande desafio seria o de delimitar a Eletrônica como meio, e não finalidade, evitando que os alunos tivessem que se debruçar por horas no entendimento de conceitos elementares, e outros não tão elementares, de Eletrônica. Dessa forma o Prof. Banzi objetivava fazer com que os alunos concentrassem seus esforços no desenvolvimento dos protótipos e no desenvolvimento de caminhos cada vez mais rápidos e poderosos para a construção de melhores soluções às suas necessidades.

Nesse contexto, começou-se a elaboração de uma plataforma de Hardware e Software capaz de controlar e mensurar o mundo físico através do processamento de entradas e saídas, que abstraísse ao máximo a parte eletrônica, permitindo que o foco fosse mantido nos demais aspectos mencionados. Denominou-se tal plataforma de Arduino (Figura 4), nome de origem germânica *Hardwin*, *hard*(forte) e *win*(amigo), e que em italiano tornou-se Arduino.

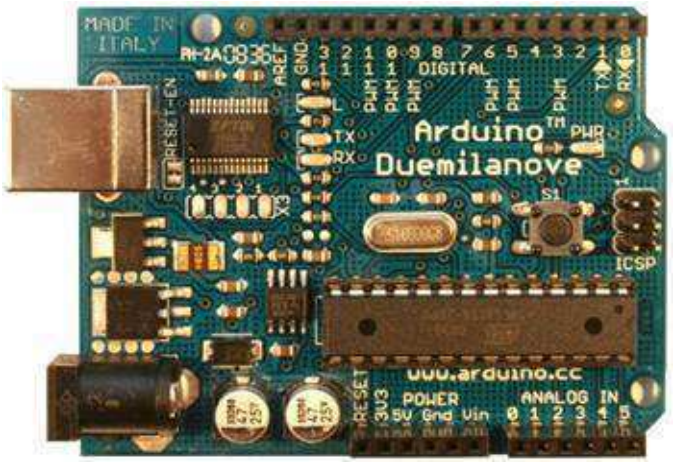


Figura 4. Primeira placa Arduino elaborada por Banzi: Duemilanove
(Fonte: DINIS, 2010, p. 36).

3.1 CARACTERÍSTICAS ESTRUTURAIS

O Arduino é composto por um pequeno computador, o qual pode ser programado para processar suas entradas e saídas, interagindo com dispositivos de entrada e componentes externos conectados ao mesmo (Figura 5) [3].

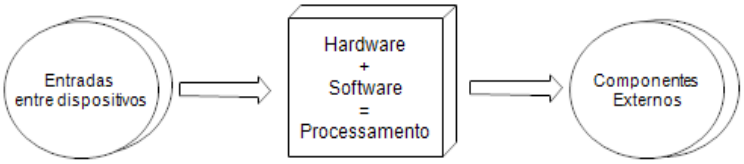


Figura 5. Interação da Plataforma com meio

3.1.1 CARACTERÍSTICAS DE HARDWARE

Existem atualmente 15 versões diferentes do dispositivo, diferenciadas por suas funcionalidades específicas e com tipos distintos de microcontrolador. O tipo de plataforma eleita para análise foi a última versão do Arduino Uno, a versão 3, cujo microcontrolador é um ATmega 328. Atendem-se, com a placa, os requisitos básicos do projeto: custo reduzido, número de entradas e saídas mínimos.

Em sua versão preliminar do Arduino Uno, utilizava-se o chip FTDI para implementar um conversor USB para a Serial. Na versão dois, passou-se a utilizar um Atmega8U2, com um conversor USB para serial programado, podendo o firmware ser atualizado e reconhecido pelo computador como qualquer outro dispositivo externo, como mouse, por exemplo. Na versão três, substitui-se por um Atmega16U2, além de se ter adicionado ainda uma maior robustez no circuito do reset e alguns outros pinos. Com a substituição do FTDI, foi possível um barateamento da placa [3][4].

Ainda são disponíveis dois tipos de Arduino Uno Versão 3: o SMD e DIP. Tendo por diferenciação aspectos meramente físicos, de modo que no primeiro caso o chip já se encontra soldado na placa, enquanto o DIP permite ser removido ou mesmo substituído em caso de dano (Figura 6) [5].

A placa Arduino é dotada de catorze pinos de entrada e saída digitais, podendo seis ser configurados para geração de sinais PWM, com seis pinos analógicos, três pinos de aterramento, um pino de reset, um pino com $3{,}3\ V$ e outro de $5V$. Além disso, têm-se o pino de referência analógica AREF.

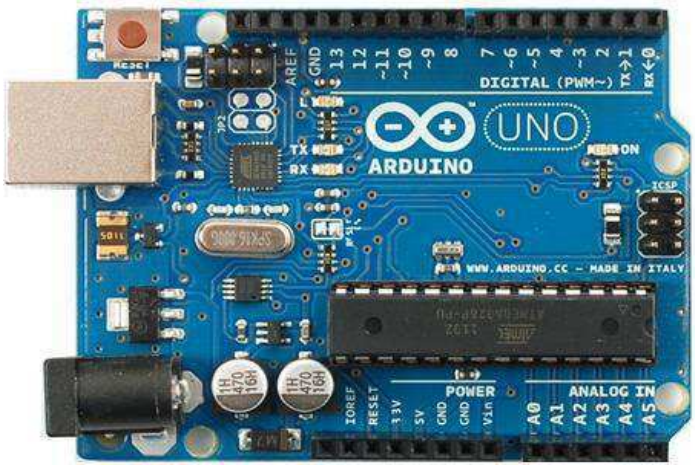


Figura 6. Versão mais simples e atual da placa Arduino, utilizada na aplicação prática.
(Fonte: ARDUINO, Plataforma de Prototipagem Eletrônica Open-source. Disponível em: <http://www.arduino.cc/en/Main/Products>. Acesso em: 15 fev. 2013).

3.1.1.1 Principais Características do Microcontrolador

O funcionamento de um microcontrolador ATmega328 pode ser resumido através de seus blocos funcionais (Figura 7) [6].

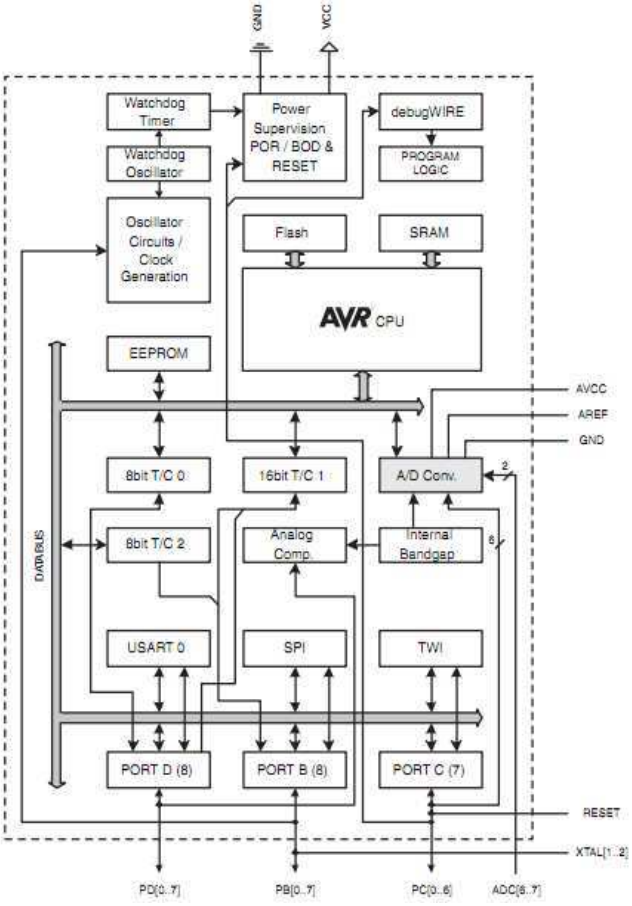


Figura 7. Blocos funcionais do ATmega328.
(Fonte: ATMEL, Datasheet: 8 bit AVR Microcontroller with 4/8/16/32k Bytes In- System Programmable Flash. ATmega 48 PA, 88 PA, 168 PA, 2009, p.5).

O núcleo AVR tem trinta e dois registradores de uso geral, estando todos conectados à Unidade Lógica Aritmética. O AVR permite que dois registradores independentes sejam acessados em uma única instrução, executada em um único ciclo de *clock*. Essa característica operacional resulta em uma arquitetura mais eficiente, além de permitir obter taxas de transferência de dados 10 vezes mais rápidas do que os microcontroladores CISC[1] convencionais.

Além das memórias descritas posteriormente, há ainda três Timer/Counters com modos de comparação, interrupções internas e externas, serial programável USART, uma porta SPI, um *Watchdog Timer*[2] com oscilador interno, e cinco modos selecionáveis de economia de energia.

No modo de "*wait-state*" da CPU os módulos: SRAM,*Timer/Counters*, USART, porta SPI e a interrupção continuam funcionando.

No modo de economia de energia, o temporizador assíncrono continua funcionando, permitindo ao usuário manter um temporizador de base enquanto o resto do dispositivo encontra-se hibernando.

O modo ADC de redução de ruído interrompe a CPU e todos os módulos E/S, exceto, o temporizador assíncrono ADC, para minimizar o ruído de comutação durante as conversões ADC.

No modo de *"wait-state"*, o oscilador está em funcionamento enquanto o resto do dispositivo hiberna, o que permite um *start-up* rápido e com baixo consumo de energia.

A AVR ATmega328 é suportada por um pacote completo de programas e ferramentas de desenvolvimento de sistemas, incluindo compiladores C, Macro Assemblers e depurador/simulador de programa. A plataforma Arduino consiste em um arcabouço de hardware e software auxiliares ao AVR para facilitar o desenvolvimento de soluções que utilizem este microchip da fabricante ATMEL.

A grande diferença entre os vários modelos de Arduino disponíveis no mercado reside na memória do microcontrolador, sendo essa uma característica determinante em seu desempenho [7].No microcontrolador ATMega328 há três tipos de memória: *Flash*,

[1] CISC(*Complex Instruction Set Computer*) é uma linha de arquitetura de processadores capaz de executar centenas de instruções complexas diferentes. Sendo assim, extremamente versátil. Os processadores baseados em um conjunto de instruções complexas contêm microprogramação, ou seja, um conjunto de códigos de instruções, que são gravadas no processador, permitindo-lhe receber as instruções dos programas e executá-las, utilizando as instruções contidas na sua microprogramação.

[2]*Wacthdog Timer* é um dispositivo eletrônico temporizador que dispara um reset ao sistema, se o programa principal devido a alguma condição de erro deixar de fazer reset no *Watchdog timer*.

SRAM, EEPROM. A memória *Flash* é utilizada para o armazenamento do programa que é carregado no microcontrolador para ser executado, bem como para o armazenamento do *bootloader*. O *bootloader* é responsável pelo gerenciamento da inicialização do microchip, e desempenha algumas tarefas programadas como determinar quando reprogramar ou passar para a aplicação principal[3].

A memória SRAM (*Static Random Access Memory*) é a memória utilizada para armazenamento das variáveis de processamento. Quando o Arduino é desenergizado, as informações armazenadas nesta memória são perdidas. Diferentemente de EEPROM (*Electrically Erasable Programmable Read-Only Memory*), que é responsável pelo armazenamento de constantes ou dados de configuração, e apresenta longos ciclos de gravação e rápida leitura de dados.

O modelo Arduino Uno Versão 3 tem $32\,KB$ de memória Flash, dos quais $0.5\,KB$ são destinados ao *bootloader,*$2\,KB$ de SRAM e uma EEPROM de $1\,KB$.

O ATMega328, assim como outros microcontroladores, oferece ainda um conversor A/D. Na conversão de um sinal analógico em digital, fazem-se amostragens, e representa-se a leitura realizada através de um valor equivalente binário, de acordo com a resolução do conversor A/D. A precisão do conversor é determinada pelo número de bits que o mesmo gera para representar as grandezas analógicas. O conversor A/D do ATmega é de $10\,bits$, representando 2^{10} níveis de tensão distintos. Assim, a resolução do conversor pode ser expressa por:

$$Resolução = \frac{V_{ref}}{2^n}, \qquad (1)$$

Em que V_{ref} é a tensão de referência [V]; e n é a número de bits [7].

Desse modo, sendo a opção *default* da tensão de referência $V_{ref} = 5V$, para o Arduino, têm-se uma resolução de:

$$Resolução = \frac{5}{1024} \cong 5\,mV, \qquad (2)$$

Diz-se, portanto, que o Arduino pode detectar variações de apenas $5\,mV$. Há, porém, a possibilidade de alterar-se o V_{ref}, através do pino de entrada, já mencionado, A_{REF}. Desse modo, com a possibilidade de diminuição da resolução de referência, têm-

se também uma diminuição dos valores de tensão detectáveis e uma consequente diminuição do nível máximo de medição de tensão permitido, o que pode não ser desejável em determinadas aplicações [1].

O ATmega328 é um microcontrolador com elevada capacidade de processamento e um conjunto de periféricos diversificado, conferindo, dessa forma, uma solução altamente eficaz para muitas aplicações de controle embarcado.

3.1.2 CARACTERÍSTICAS DE SOFTWARE

A plataforma de software do Arduino, modelada na linguagem de programação *Processing*[3], denominada IDE (*Integrated Devolopment Environment*), trata-se de um programa executado em um computador, que permite a codificação de programas em linguagem específica, *sketches*, a serem carregados na plataforma de Hardware[8].

Uma vez requerido o *upload* do sketch na placa Arduino, o código escrito em linguagem específica é traduzido na Linguagem C. Na sequência, o mesmo é compilado pelo AVR- GCC, responsável pela tradução final para a linguagem entendida e conhecida pelo microcontrolador, simplificando, dessa forma, a programação de microcontroladores.

Desse modo, o ciclo de programação de um Arduino pode ser resumido como:

1. Escreve-se o *Sketch*;
2. Compila-se o *Sketch;*
3. Liga-se à placa a uma porta USB do computador;
4. Envia-se o *Sketch* para a placa, através da USB;
5. A placa executa o programa;

O IDE pode ser executado em qualquer sistema operacional, é dividido em três partes: o Toolbar no topo, *Sketch* Windows no centro e a janela de mensagens na base (Figura 08).

[3] *Processing* é uma linguagem de programação de código aberto e ambiente de desenvolvimento integrado desenvolvido em 2001 pelo MIT (Massachusetts Institut of Tecnology), construído para comunidades de projetos visuais com o objetivo de ensinar noções básicas de programação de computador em um contexto visual.

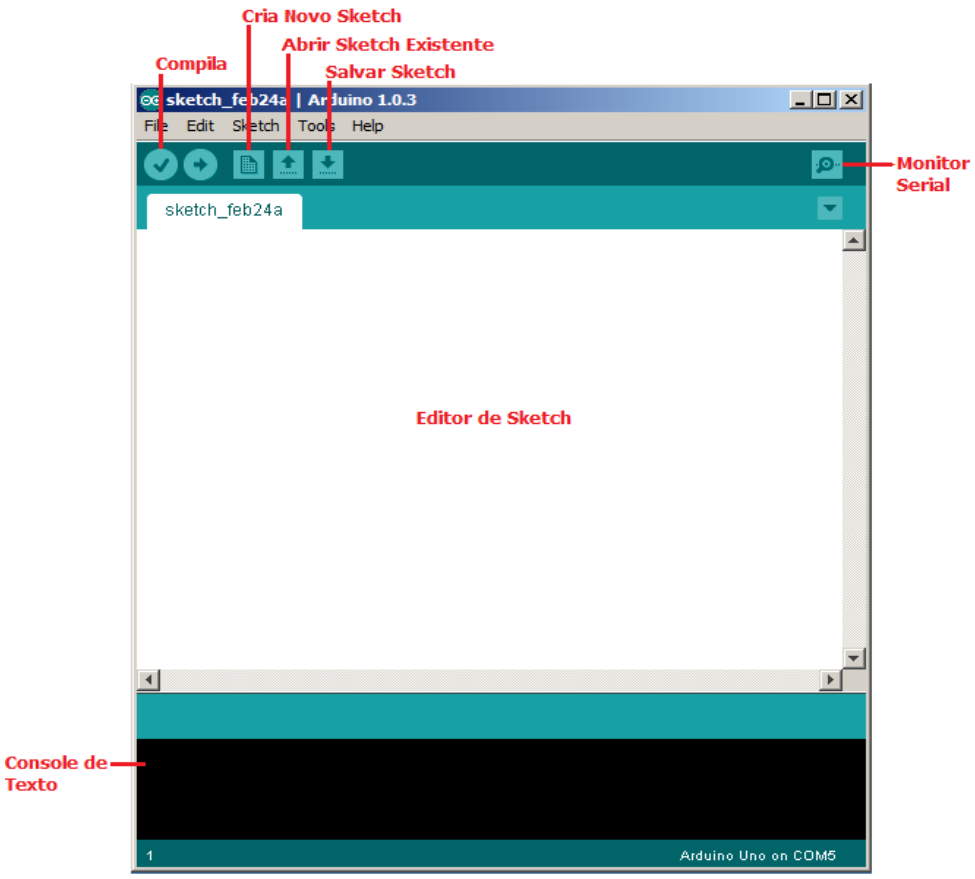


Figura 8. Plataforma de Software do Arduino: IDE.

3.2 OUTROS RECURSOS DO ARDUINO

Placas externas, denominadas *Shield,* podem ser acopladas ao Arduino, com a finalidade de expandirem-se suas funcionalidades, de maneira prática, eficiente e barata. Há modelos de *Shields* para comunicação, como Ethernet, Wi-Fi, Xbee e Bluetooth, ou mesmo para controle de atuadores e sensores. Outras opções são disponíveis como: *Shield* Joystick, que contém todos os componentes necessários para se implementar um Joystick com o Arduino; o Relê *Shield*, diretamente controlado por um Arduino, através de suas saídas digitais com alimentação externa de $9V$, capazes de controlar quatro relês, com chaveamento de potência de $90W$ em CC ou $360\,VA$; o *Shield* celular, dotado de todos os componentes necessários para conectar o Arduino com o módulo celular SM5100B, quad-band, permitindo o envio de SMS e outras funcionalidades; *Shield* GPS, MP3 player, reconhecimento de voz EasyVR, MicroSD, de controle de corrente de entrada na placa, o Power Driver *Shield, Shield* MIDI, permitindo comunicação do Arduino com o protocolo MIDI, fazendo com que o mesmo possa

controlar sintetizadores, sequenciadores e outros dispositivos musicais, além dos módulos de expansão da EEPROM, o Multiplexador de entradas e saídas, etc. (Figura 9) [9] [10].

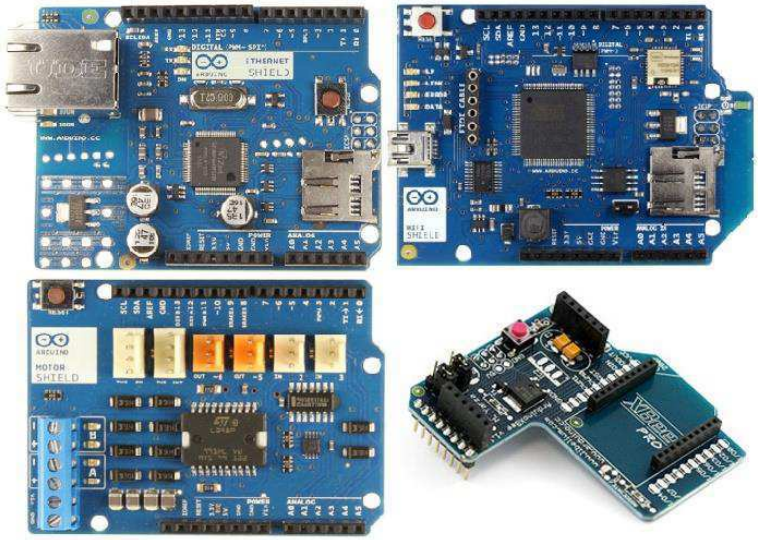


Figura 9. Shields Arduino: Ethernet, Wi-Fi, Motor e Xbee.
(Fonte; ARDUINO, Plataforma de Prototipagem Eletrônica Open-source. Disponível em: <http://www.arduino.cc/en/Main/Products>. Acesso em: 15 fev. 2013).

3.3 VANTAGENS DO ARDUINO

Apesar de ter sido inicialmente destinado a grupos amadores de eletrônica, com a finalidade de aplicação de suas capacidades de prototipagem, a plataforma Arduino acabou tendo grande aceitabilidade por parte dos mais diversos grupos e para distintas finalidades. Para muitos, é desinteressante construir-se uma plataforma do zero, em um processo que requer tempo e conhecimento avançado, uma vez que se têm como alternativa a disponibilidade de dispositivos pré-fabricados com a capacidade de configurá-los de acordo com sua necessidade, por um custo muito reduzido [11].

A plataforma também aparece como uma excelente alternativa para prova de conceito, podendo-se rapidamente verificar se algo está funcionando, motivando-se a passar para o próximo passo, ou mesmo favorecendo o recebimento de incentivo financeiro para promover tal ideia.

Podemos ainda enumerar diversas outras vantagens da plataforma Arduino. Tais como:

- Seu Hardware é um projeto aberto e o Software é código aberto, possibilitando seu melhor entendimento e podendo-se adequá-los às mais diversas necessidades, agregando maior confiabilidade à plataforma;
- Grande comunidade de usuários, muitas pessoas desenvolvendo as mais diversas aplicações, compartilhando códigos e diagramas, para que outros copiem e modifiquem;
- Multi-plataforma, seu Software pode ser executado em Windows, Macintosh e Linux;
- A Comunicação é via USB para upload do programa, aumentando-se consideravelmente a praticidade da plataforma, uma vez que a porta serial, utilizada em outras plataformas, não é uma opção presente nos computadores mais modernos;
- Seu Software é baseado no Processing Programming IDE (*Intergrated Development Environment*), de fácil utilização e desenvolvido para projetistas e designers;
- Custo reduzido;
- Facilidade de manipulação, foi desenvolvido em ambiente educacional, tem uma maneira eminentemente didática de apresentação do Hardware e Software.

3.4 EXEMPLOS DE APLICAÇÃO DA PLATAFORMA

O Arduino nos últimos anos vem sendo utilizado na prototipagem, implementação ou emulação da supervisão ou controle de sistemas interativos a nível doméstico, comercial, móvel e industrial (análogo ao CLP no controle de sistemas). Porém, observa-se como uma das suas maiores aplicações, o desenvolvimento de versões simplificadas, eficientes e, sobretudo, baratas para sistemas eminentemente caros. Alguns exemplos ilustrativos são: o quadricóptero Arducopter e a impressora 3D RepRap.

O Arducopter é um veículo aéreo não tripulado, criado pela Drones DIY (Figura10). Baseia-se na plataforma Arduino, e foi o primeiro quadricóptero de fonte aberta capaz de voar completamente autônomo [12]. Possui uma solução UAV(Unmanned Aerial Vehicle) modular completa, permitindo que se escolham os

recursos desejados de acordo com a funcionalidade. É também *open-source* e vem sendo utilizado nas mais diversas aplicações; em pesquisas, trabalhos acadêmicos, hobby, gestão de propriedades, acompanhamento de construções, inspeção industrial, gestão florestal, infraestrutura, etc.[13].

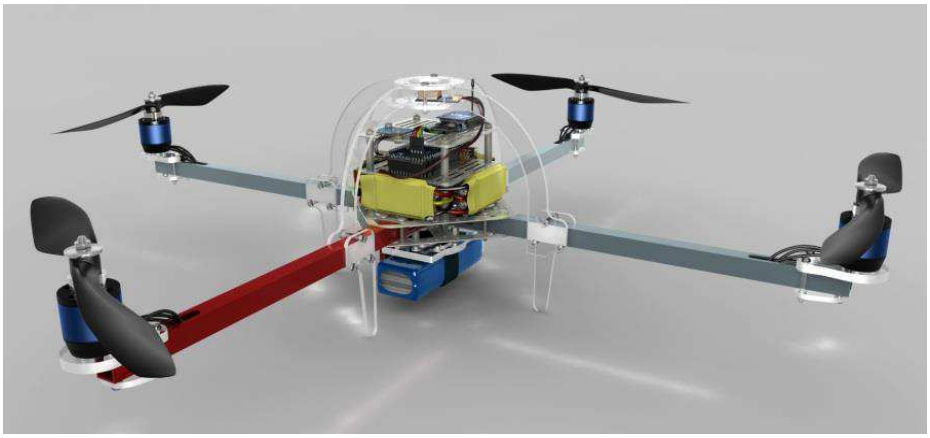


Figura 10. Quadricóptero com Arduino
(Fonte: ARDUCOPTER. Piloto Automático baseado em Arduino. Disponível em: <http://www.code.google.com/p/arducopter>. Acesso em: 17 fev. 2013).

Também baseada em Arduino, a RepRap é a primeira impressora 3D de baixo custo, também *open-source*, permitindo que a tecnologia capaz de imprimir objetos de plástico seja compartilhada para benefício de todos (Figura 11). Atualmente, é a impressora 3D mais utilizada entre os membros globais da comunidade, segundo a pesquisa datada de 17 de julho de 2012, Mailanen, J& Vadén, T.: Manufacturing in motion: First Survey on the 3D Printing Community, Statistical Studies of Peer Production[14].

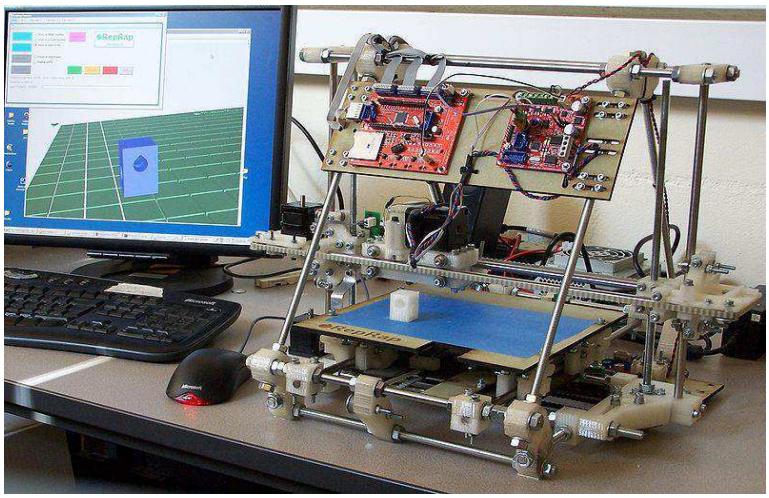


Figura 11. Impressora 3D com Arduino
(Fonte: REPRAP Fabricação de Máquinas de Auto-replicação. Reprap, 2012. Disponível em: <http:/www.reprap.org/wiki/Main_Page>. Acesso em: 18 fev. 2013).

3.5 MICROCONTROLADORES NO AMBIENTE INDUSTRIAL

Devido ao fluxo de informações e aumento da complexidade das variáveis a serem monitoradas no ambiente industrial, as tecnologias de automação devem ser desenvolvidas em um ritmo bastante acelerado. Diante desse panorama, os microcontroladores têm se apresentado já há algumas décadas como alternativas viáveis e recorrentes para o desenvolvimento de soluções que busquem alto desempenho; como eficiência de energia, suporte para alta velocidade em comunicações com e sem fio, redução de custo e espaço, etc.

Porém, a utilização de microcontroladores para aplicações industriais requer cuidados especiais, dado o ambiente hostil e com nível elevado de ruido. Justamente pela demanda de imunidade a esses ruídos elétricos, utilizam-se tensões de níveis lógicos de $24\,V$, para que a interface tolere picos de corrente, interferências eletromagnéticas e descargas eletrostáticas.

Desse modo, utilizar um microcontrolador em aplicações embarcadas, como controle de máquinas, implica em projetar cuidadosamente as interfaces, de modo que elas sejam confiáveis e permitam uma operação segura.

Na prototipagem escolhida como exemplificação do potencial da plataforma Arduino, alguns desses aspectos já oferecem tratamento pelos próprios microcontroladores pré-existentes na máquina, que fornecem também a possibilidade de adquirirem-se seus dados de modo a possibilitar a expansão de suas funcionalidades. O sistema de aquisição de dados idealizado é responsável, portanto, apenas pela supervisão do processo produtivo e não na atuação no mesmo, não sendo, dessa forma, crítico os aspectos supracitados, o que possibilitou a utilização do Arduino[15].

4 APLICAÇÃO PRÁTICA: REGISTRADOR DE DADOS DE EFICIÊNCIA DE MAQUINÁRIO

Com vistas ao melhoramento e expansão do estudo em pauta, buscou-se o desenvolvimento de uma aplicação prática do estudo anteriormente descrito.

No contexto supracitado, toma-se como plano de fundo o conhecimento de informações sobre o processo produtivo de um tear, a partir de dados coletados da máquina. O princípio básico de tecelagem: entrelaçamento de fios de trama (transversal) e urdume(longitudinal) fornecem informações quanto ao rendimento da máquina e o histórico de erros da mesma.

4.1 TEAR

Estima-se que o ato de tecer data ainda da Era Neolítica, onde já se estabelecia o princípio da tecelagem, entrelaçando-se pequenos galhos e ramos para construir barreiras, escudos e cestas. Já quanto à data exata de quando se passou a utilizar fibras de origem vegetal e animal, entrelaçadas, não se tem uma definição por parte dos estudiosos.

O exemplo mais antigo de tecido descoberto na Europa data do fim da Era Mesolítica entre 4600 e 3600 a.C.

Aos gregos é atribuída a transferência de posição vertical para horizontal, e aos egípcios a fixação dos fios de urdume em dois galhos a fim de facilitar o entrelaçamento.

Durante a Revolução Industrial, século XVIII, o setor têxtil foi um dos que mais se favoreceu dos investimentos ingleses, fazendo-se com que máquinas e mais máquina fossem criadas para substituir a força humana e animal pela mecânica, e melhorar a qualidade dos fios e beneficiar o algodão.

Em 1733, o inventor John Kay daria inicio ao desenvolvimento de toda a tecnologia para a produção de tecidos, com a lançadeira volante. Com ela, permitia-se lançar mecanicamente o fio transversal da trama entre a urmideira longitudinal, formando o tecido, com maior velocidade e fazendo com que o tecido não fosse mais limitado ao máximo comprimento dos braços abertos do artesão. Com o uso da lançadeira volante o fio de trama não seria lançado de uma mão para outra, mas com o auxílio de rodinhas fixadas numa espécie de ranhuras de madeira.

4.2 FIOS

Ainda com relação à produção de fios, em 1764 James Hargeaves criou a Spinning Jenny, uma roda de fiar múltipla, capaz de produzir dezesseis fios ao mesmo tempo, porém de baixa qualidade (Figura 12) [16].

Somente em 1771, um barbeiro chamado Richard Arkuwright patenteou sua máquina de fiar Water Frame, tendo como princípio de funcionamento a força hidráulica.

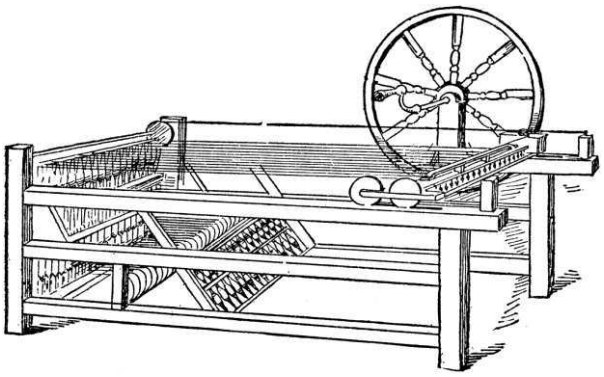


Figura 12. Máquina de James Hargeaves de produção de fios: a Spinning Jenny.
(Fonte: WIKIPEDIA. Disponível em: <http://www.en.wikipedia.org/wiki/Spinning_jenny>. Acesso em: 18 fev. 2013).

Em 1779, Samel Crompton uniu a Spinning Jenny e Water Frame, criando a Mule, que fabricava fios finos e resistentes.

Estando a fabricação de fios acelerada, o processo de tecelagem não atendia sua demanda impulsionando Edmund Cartwright em 1785 a desenvolver o tear mecânico.

Apesar de o princípio propulsor ter sido alterado com o passar dos anos, com surgimento dos teares elétricos, após a invenção da máquina a vapor, até finalmente em 1951 se dá o desenvolvimento do tear jato de ar, em que Maxbo apresenta um modelo com 350 batidas por minuto, a estrutura de funcionamento dos teares iniciais se manteve: abertura de cala, inserção de trama e a batida do pente.

Desse modo, podemos precisar:

- Abertura de cala: operação realizada para selecionar os fios do urdume, formando duas mantas de fio uma mais alta e outra mais baixa, para a introdução da trama;

- Inserção da trama: processo realizado para inserir a trama, após aberta a cala pode se utilizar diversos mecanismos de propulsão: pinça, jato de ar, jato de água, etc. (Figura 13);

- Batida do pente: faz com que o fio de trama encoste no restante do tecido, realizando o acabamento do tecido [17].

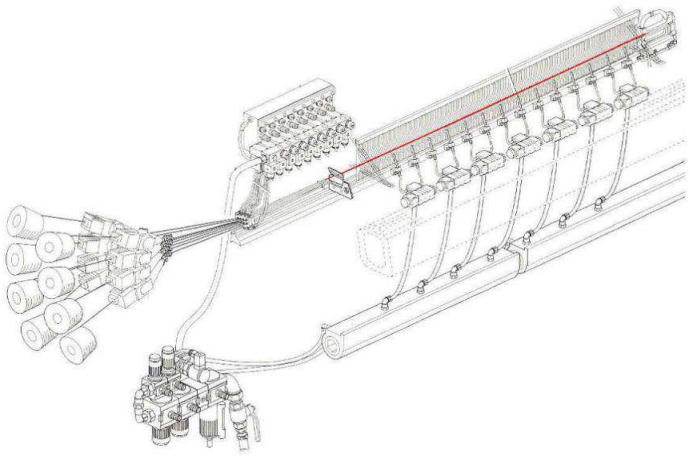


Figura 13. Processo de Inserção da trama
(Fonte: SANTOS, Felipe; GOMES, Winston.; 2010, p.22).

4.3 DESCRIÇÃO DO SISTEMA DO REGISTRADOR

Um estudo prático, utilizando-se as ferramentas até então vistas, seria a contabilização da quantidade de falhas ocorridas no processo de tecelagem, dado um intervalo de tempo, de modo a determinar os motivos mais recorrentes de falhas e ainda se o número obtido encontra-se dentro de uma margem de aceitabilidade das mesmas, e por fim, o próprio rendimento da máquina.

A forma de apresentação dessas mensagens pode adicionar mais funcionalidades ao sistema. Uma possibilidade bastante relevante seria a de apresentarem-se essas mensagens de duas maneiras distintas: uma mensagem periódica de envio da quantidade de batidas do pente desenvolvidas pela máquina, relacionada ao rendimento da mesma; e a mensagem de erro, mostrada apenas dada a ocorrência do evento, complementada por outra mensagem, caso o erro tenha sido corrigido. Desse modo, têm-se como obter o tempo demandado para a correção de cada erro e o tempo total em que a máquina esteve parada.

4.3.1 DESCRIÇÃO DO MAQUINÁRIO

A máquina escolhida para prototipagem é a TSUDAKOMA ZAX, que já é dotada de sensores e permite uma saída distinta para cada tipo de erro, além de mudança no nível de tensão, quando se dá a ocorrência do evento.

A inserção da trama na cala, para a ZAX, é feita por jato de ar, injetado através do canal do pente pelas tubeiras principais e bicos de estafetas. A velocidade de propulsão pode ser ajustada de acordo com a velocidade desejada do tear e do tipo de fio.

Desse modo, o itinerário da máquina TSUDAKOMA ZAX representa o processo de formação dos tecidos a partir dos fios (Figura 14). Iniciando-se pelo desenrolar dos fios para formação de mantas: superior e inferior, denominadas urdume, passando pela inserção da trama por alimentadores externos, até a formação do tecido [18].

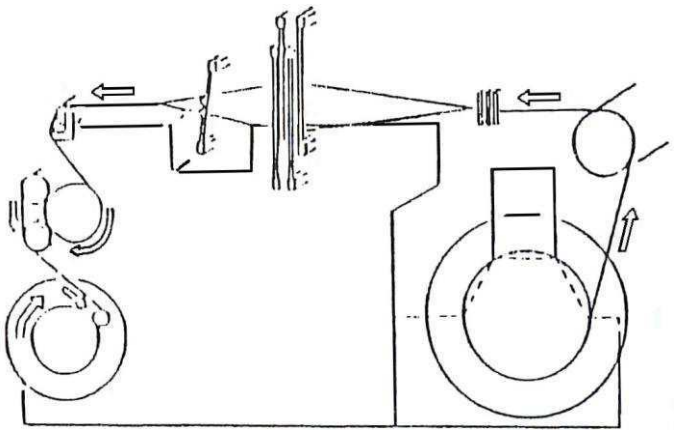


Figura 14. Itinerário de funcionamento da máquina de tecelagem.
(Fonte: TSUDAKOMA ZAX, Manual, p.6).

4.1.1.1 SENSORES

Quando um fio de urdume se rompe, existe um dispositivo mecânico, chamado *lamela*, que quando acionado encosta na barra de contato (Figura 15), detectando falha e fazendo com que o tear pare na posição programada e uma lâmpada laranja acenda.

A sensibilidade dessa barra de contato pode ser ajustada na placa SBU (Unidade de freio sensor) da máquina, tal como o peso e formato da lamela possam ser selecionadas de acordo com o tipo e o método de inserção da trama. Denomina-se a ocorrência dessa falha em termos de simplificação, de **erro urdume**.

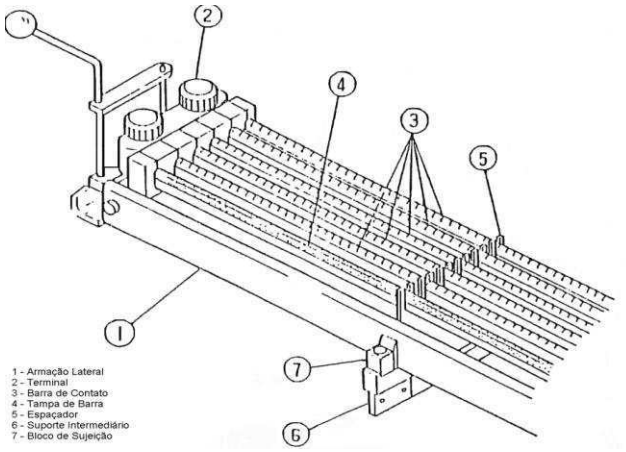


Figura 15. Sistema de acionamento do rompimento do fio de urdume.
(Fonte: TSUDAKOMA ZAX, Manual, p.14)

Os sensores de trama estão montados no prendedor do pente e na extremidade do lado do acionamento, monitorando fotoeletricamente se existe ou não fio de trama. Estão divididos em dois tipos de sensoriamento H1 e H2, cada um indicando um evento de importância distinta:

- Sensor H1: É o sensor mais importante, pois atua em todas as batidas da máquina. Durante a sincronização regulada, ele detecta e para o tear quando não há fio de trama à sua frente. São descobertas as inserções erradas, em que o fio de trama não chega ao sensor H1 porque é curto ou torto, e as inserções são repelidas (Figura 16);

- Sensor H2: Sua atuação é inversa ao sensor H1, ou seja, tem por função detectar os fios que alcançam o sensor H2, sua varredura é nos erros complementares aos cobertos pelo sensor H1.

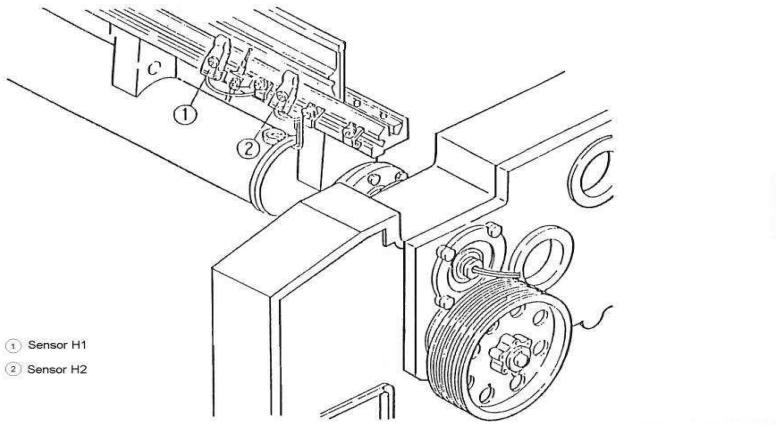


Figura 16. Sensoriamento de rompimento de trama: H1 e H2.
(Fonte: TSUDAKOMA ZAX, Manual, p.51).

O sensor Encoder é um transdutor de posição angular, eletromecânico, que conta ou gera pulsos elétricos a partir do movimento rotacional de seu eixo. Desse modo, esse codificador angular é utilizado na TSUDAKOMA ZAX, indicando o momento exato de inserção da trama, tal como a quantidade de ciclos finalizados de batidas do pente (Figura 17).

Figura 17. Encoder utilizado para mensurar a rotação do pente de batidas

O sinal de STOP, por sua vez, é detectado mecanicamente assim que acionada a botoeira destinada para tal na máquina, enquanto o sinal de Avaria, ou pane elétrica, é detectado assim que interrompida a comunicação entre o barramento das placas.

4.1.1.2 Os sinais da máquina

Existe na máquina descrita o painel SQC, supervisor dos sinais de sensoriamento e outros status de funcionamento (Figura 18). Interceptando o sinal antes que o mesmo alimente as saídas luminosas de ocorrência de erro, podem-se obter, com o osciloscópio, os sinais de saída do sensoriamento da máquina.

Figura 18. Placa SQC: supervisor dos sinais de sensoriamento e outros status de funcionamento.

O sinal captado, que indica a quantidade de batidas do pente, e a consequente eficiência da máquina, pode ser obtido através da placa SLC da máquina (Figura 19), que recebe da placa SVU o sinal já parcialmente tratado, uma vez que a SVU interpreta o sinal obtido do codificador angular, Encoder (Figura 20).

Figura 19. Placa responsável pelo tratamento do sinal de batidas do pente.

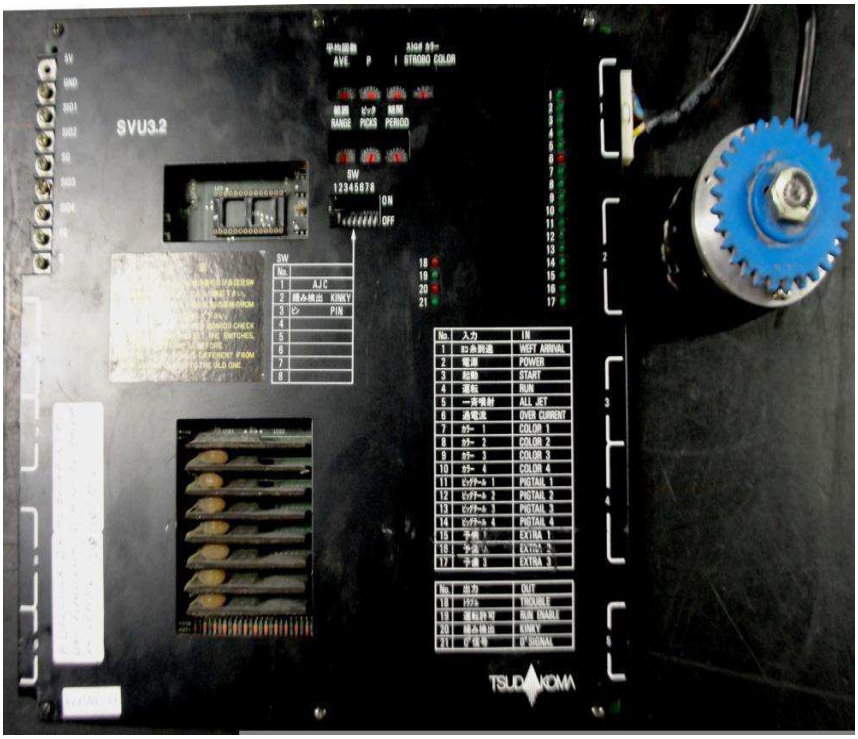


Figura 20. Placa SVU: designada à captação dos sinais do Encoder.

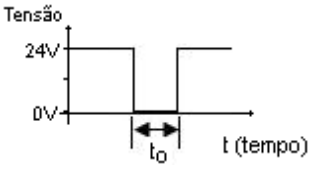
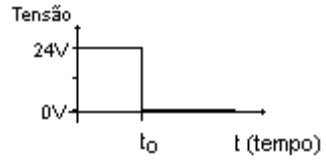
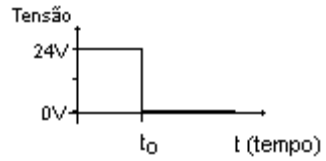
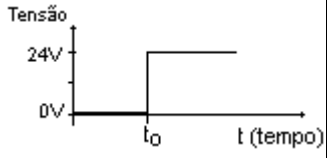
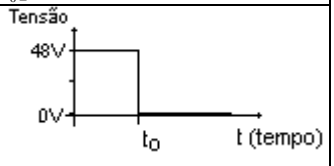
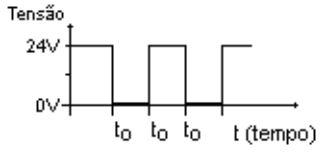
Entradas do Arduino	Descrição do funcionamento	Placa da Máquina	Entrada na placa	Sinalização (LED)	Saída Luminosa	Descrição do sinal elétrico
URDUME	Erro provocado pelo rompimento de um fio de urdume, indicado pela "queda" das lamelas.	SQC	DR	412	**- O LED apaga:** não ocorrência do erro mencionado; **- O LED acende:** ocorrência do erro;	 t_0 = Ocorrência do evento: 100 ms
TRAMA H1	Sinalização da ocorrência do fio de trama ser menor do que o esperado ou mesmo a ausência desse fio	SQC	H1S	304	**- O LED apaga:** não ocorrência do erro mencionado; **- O LED acende:** ocorrência do erro;	 t_0 = Ocorrência do evento
TRAMA H2	Sinalização da ocorrência de desperdício do fio de trama, um fio com tamanho além do esperado	SQC	H2S	303	**- O LED apaga:** não ocorrência do erro mencionado; **- O LED acende:** ocorrência do erro;	 t_0 = Ocorrência do evento
AVARIA	Sinalização da ocorrência de defeito elétrico	SQC	CE	102	**- O LED apaga:** o erro de avaria ocorre; **- O LED acende:** não se têm defeito elétrico.	 t_0 = Ocorrência do evento
STOP	Sinalização do acionamento da parada	SQC	PS3	003	**- O LED apaga:** o evento de parada ocorre; **- O LED acende:** não ocorre evento de parada;	 t_0 = Ocorrência do evento
BATIDAS	Sinaliza o evento "batida do pente"	SLC	CPK	CPK	**- O LED apaga:** não está ocorrendo a batida do pente; **- O LED acende:** a batida do pente ocorre;	 t_0 = Ocorrência do evento

Tabela 1. Representação dos sinais de erro e localização de onde obter o sinal.

Ao fim dos experimentos, pode-se realizar um guia resumo de como obter as informações desejadas, como demonstradas através da Tabela 01.

Na primeira coluna, ilustram-se os tipos de erro analisados. Na segunda coluna, têm-se uma descrição concisa do que se trata cada evento. Outra informação imprescindível para intervenções futuras é a documentação da terceira e quarta colunas: a placa e qual o ponto do circuito na mesma onde é possível se obter os sinais

desejados. Na sequência, relata-se à conferência da ocorrência do evento, através das saídas luminosas da placa, seguido pela descrição de seu comportamento uma vez ocorrido o evento. Na última coluna, têm-se a representação dos sinais visualizados no osciloscópio.

Como mostrado na última coluna da Tabela 1, para o sinal de Urdume, antes da ocorrência do evento, a tensão de leitura era estável em $24\,V$, na ocorrência do evento t_o, o valor comuta para zero durante $100\,ms$, retornando para $24\,V$ uma vez decorrido esse tempo. Já para os sinais da Trama H1 e Trama H2 a tensão inicial de leitura era estável em $24\,V$, comuta para zero e é mantido em zero até que o erro seja removido da máquina. Para o sinal de Stop, o processo é basicamente o mesmo, mudando apenas que sua comutação é para $48\,V$, ao invés de $24\,V$. Quanto ao caso de defeito elétrico, descrito como Avaria, ocorre de forma antagônica, em que o estado natural desse sinal é zero, ocorrendo o evento em t_o, esse valor comuta para $24\,V$. Para o sinal de Batidas, em que são contabilizadas as batidas do pente da máquina, cada vez que ocorre o evento o sinal vai de zero para $24\,V$.

4.1.1.3 O CONDICIONADOR

Como mencionado em tópicos anteriores, denomina-se de condicionador os componentes responsáveis por tornar o sinal compatível com o método de aquisição escolhido. Pelas características dos sinais obtidos da máquina, é suficiente fazer a atenuação do sinal de tensão para valores toleráveis.

Temos, portanto, que a tensão que pode ser submetida a entrada de uma placa Arduino é de $5\,V$, enquanto a corrente máxima permitida é de $40mA$. Contudo, como já demonstrado até aqui, os níveis máximos de tensão obtidos da máquina são de $24\,V$ para quase todas as entradas e $48\,V$, para o caso da ocorrência do Stop.

Um método eficiente, prático e barato para essa atenuação é um mero divisor de tensão (Figura 21), conforme enunciado nas linhas que se seguem.

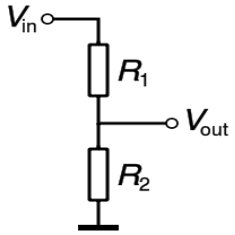


Figura 21. Divisor de tensão(Fonte: WIKIPEDIA. Disponível em: <http://www.pt.wikipedia.org/wiki/Divisor_de_tensão>. Acesso em: 20 fev. 2013).

$$\boldsymbol{V_{out} = \left(\frac{R_2}{R_1 + R_2}\right).V_{in}}\,, \tag{3}$$

Sabemos já que a tensão a ser obtida será a obtida na saída do redutor conectada a entrada do Arduino, desse modo tem-se que V_{out} deverá ser um valor menor do que $5\,V$, em termos de proteção da placa, e maior que $3\,V$, para que o sinal seja satisfatoriamente interpretado como nível lógico alto na leitura da porta de entrada digital.

Por fim, tem-se o valor de V_{in} como $24\,V$ para a maioria das entradas, e V_{in} como $48\,V$ para o caso de Stop. De modo que, ao atribuirmos um valor grande de resistência para o R_2, respeitando os limites de corrente, é possível obter o valor de R_2, e consequentemente todos os parâmetros desejados, como demonstrado nas linhas seguintes.

Para $V_{in} = 24V$ e $V_{out} = 4{,}56\,V$, adotando-se um valor para $R_2 = 4{,}7\,k\Omega$, substituindo-se tais valores na Equação (3), temos que:

$$4{,}56 = \left(\left(\frac{4{,}7k}{4{,}7k + R_1}\right).24\right)V, \tag{4}$$

$$R_1 = 18{,}8\,k\Omega. \tag{5}$$

A corrente de saída, para o caso supracitado é de:

$$I_{out} = \frac{V_{out}}{R_2}\,, \tag{6}$$

$$I_{out} = \frac{4{,}56}{4{,}7k} = 0{,}96\,mA. \tag{7}$$

Atendendo satisfatoriamente, uma vez que o valor é inferior a $40mA$.

Para $V_{in} = 48V$ e $V_{out} = 4{,}72\,V$, adotando-se um valor para $R_2 = 4{,}7\,k\Omega$, substituindo-se tais valores na Equação (3), temos que:

$$4{,}72 = \left(\left(\frac{4{,}7k}{4{,}7k + R_1} \right) . 48 \right) V \tag{8}$$

$$R_1 = 43\ k\Omega \tag{9}$$

A corrente de saída, para o caso supracitado é de:

$$I_{out} = \frac{V_{out}}{R_2} \tag{20}$$

$$I_{out} = \frac{4{,}72}{4{,}7k} = 1{,}01\ mA \tag{31}$$

Atendendo-se satisfatoriamente, uma vez que o valor é inferior a $40mA$.

Desse modo, utilizando apenas resistores, desenvolveu-se um condicionador de atenuação analógica de tensão a ser ligado entre os sinais da máquina e o Arduino (Figura 22).

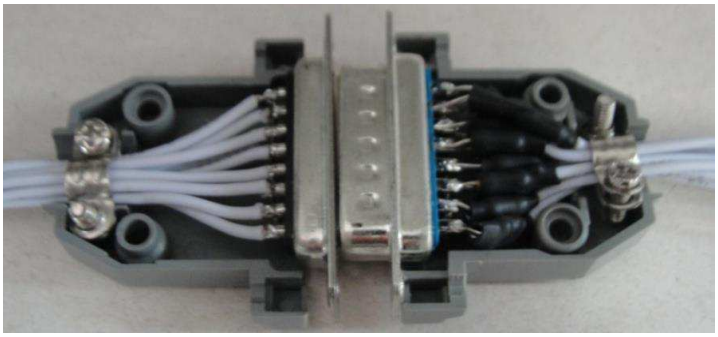


Figura 22. Atenuador de tensão, utilizando resistores, a ser interligado entre a máquina e o Arduino.

4.4 ADICIONANDO REDUNDÂNCIA DE ALIMENTAÇÃO

A alimentação da plataforma é realizada através de uma das fontes de tensão fornecida pela própria máquina. De modo que, para garantir-se o funcionamento da

plataforma mesmo quando a máquina for reiniciada devido a qualquer motivo técnico, acoplou-se ao sistema uma redundância em sua alimentação: uma bateria de $9\,V$, de modo a manter a plataforma constantemente carregada.

4.5 CARACTERIZAÇÃO DO SISTEMA PROPOSTO

Tendo sido elaborado um estudo preliminar de cada módulo do sistema, os mesmos podem ser conectados de maneira a se atingir o objetivo desejado (Figura 23).

Através do estudo dos sinais fornecidos pelo sensoriamento da máquina, atenuando adequadamente esses sinais, pode-se utilizar um microcontrolador para processá-los, com o auxílio da plataforma Arduino, que por sua vez transforma esses sinais em informações úteis para o usuário, apresentando-lhes os dados em um computador (Figura 23) (Figura 24).

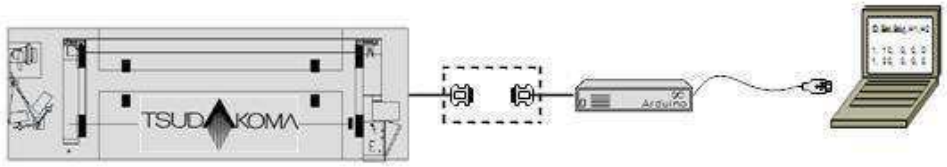


Figura 23. Esquema do sistema proposto

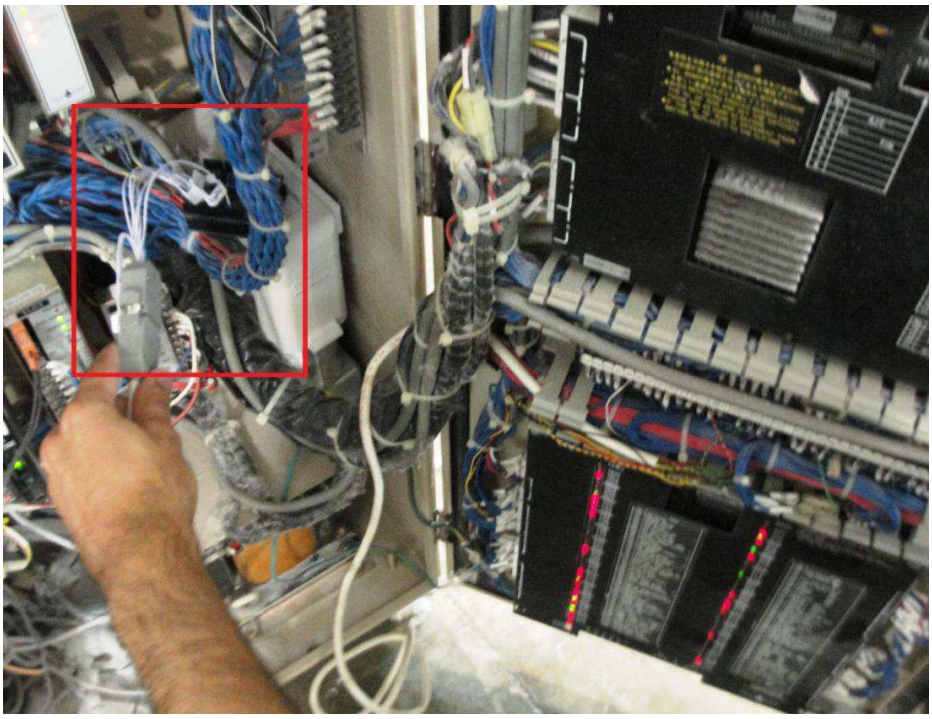


Figura 24. Interligação da plataforma Arduino à máquina.

4.6 IMPLEMENTAÇÃO DA AQUISIÇÃO DE DADOS: TESTES EFETUADOS EM LABORATÓRIO

Para implementação da aquisição de dados no Arduino, utilizou-se o IDE oferecido pela própria plataforma em seu sítio eletrônico, onde foi feita a codificação do *sketch* em linguagem específica, semelhante à Linguagem de programação C.

4.6.1 TESTES DE PRECISÃO DE CONTAGENS

Nessa etapa do projeto, o sketch elaborado tem a função elementar de contar as mudanças de estado de cada sinal, armazenar em variáveis e por fim mostrar essas informações ao usuário (Figura 25).

A contagem da quantidade de batidas do pente é realizada através de uma interrupção. Assim que detectada a mudança de estado da porta 2, destinada a contagem das batidas, o programa é interrompido e dá-se o incremento da variável. O processo de interrupção é adequado para atender eventos aleatório ou periódicos, nesse último caso, onde requisitos temporais são críticos. Com o uso do processo de interrupção é possível atender aos requisitos de contagem das batidas.

Figura 25. Fluxograma da lógica de funcionamento do programa.

A instrução de interrupção descreve o pino em que a interrupção se dá, 0 caso o pino utilizado seja o pino 2 e 1 caso seja o pino 3; seguido pelo tipo de interrupção. No caso em pauta, seleciona-se o modo CHANGE, fazendo-se com que ocorra uma interrupção no programa sempre que ocorrer uma mudança de *status* no sinal do pino sendo monitorado, conforme apresentado adiante no trecho do código.

```
//Chamada da interrupção, o parâmetro 1 indica a observância do pino 3, sempre que ocorrer uma mudança de estado

attachInterrupt(0, interrupcao, CHANGE);
```

Já para as demais variáveis, como a frequência de mudança de estado é bastante inferior, a contagem dentro do próprio loop principal do *sketch* é satisfatória.

Para que a contagem de erros ocorra de maneira eficiente, evitando-se que a condição de erro seja identificada mais de uma vez dentro de um mesmo evento, devido a velocidade de execução do código, utilizou-se uma variável, **prev_[nome do erro]**, para cada erro, fazendo-se com que previamente seja analisado o *status* anterior, realizando o incremento das variáveis apenas quando ocorrer a mudança do status, conforme o código apresentado a seguir.

```
//Rotina de leitura dos erros
      void loop()
      {
      state_stop = digitalRead (input_stop);

         if(state_stop == 0)
      {
      if(prev_stop !=0)
      {
      count_stop++;
      prev_stop =0;
          }
        }
        else
        {
          prev_stop = 1;
        }
      state_h1 = digitalRead (input_h1);
      if(state_h1 == 1)
        {
```

```
if(prev_h1 !=1)
    {
    count_h1++;
    prev_h1 =1;
    }
  }

  else
  {
    prev_h1 = 0;
  }
  state_h2 = digitalRead (input_h2);
 if(state_h2 == 1)
  {
if(prev_h2 !=1)
    {
    count_h2++;
prev_h2 =1;
    }
  }
else
  {
    prev_h2 = 0;
}

  state_urd = digitalRead (input_urd);
if(state_urd == 1)
  {
if(prev_urd !=1)
    {
    count_urd++;
    prev_urd =1;
    }
  }

  else
  {
    prev_urd = 0;
  }
state_avarias = digitalRead (input_avarias);
if(state_avarias == 1)
  {
if(prev_avarias !=1)
    {
    count_avarias++;
prev_avarias =1;
```

```
  }
}

else
{
  prev_avarias = 0;
}
```

4.4.1.1 PROCEDIMENTO DE TESTE

A título de simulação, antes de se efetuar o teste diretamente na máquina, utilizou-se um pino da E/S digital da placa do Arduino para geração de um trem de pulsos, conforme descrito no código a seguir.

```
//Declaração do pino 6 como saída digital

const intchange_state = 6;
pinMode(change_state, OUTPUT);

//Gerando o sinal de saída no pino 6

if(elapsedTime > 100)
{
digitalWrite(change_state, HIGH);
startTime = millis();
  }

else
  {
digitalWrite(change_state, LOW);
}
```

O pino 6, utilizado para geração do trem de pulsos foi conectado ao pino:

- Pino 2, destinado à contagem de batidas;
- Pino 5, destinado à contagem dos eventos de Stop;
- Pino 7, destinado à contagem dos eventos de Urdume;
- Pino 8, destinado à contagem de Trama H2 e
- Pino 9, destinado à contagem de Avaria.

O pino que seria destinada à contagem do evento Trama H1encontra-se sendo utilizada para gerar o sinal de saída (pino 6). Os demais pinos de E/S, ainda não utilizadas, são destinados à comunicação com *Shields*, que possam ser utilizados no futuro, à transferência do software, ou modificação da referência do conversor A/D, não podendo, assim, serem utilizados. A simulação efetuada, portanto, deve contar igualmente em todas suas entradas, exceto à destinada a contagem da Trama H1(Figura 26).

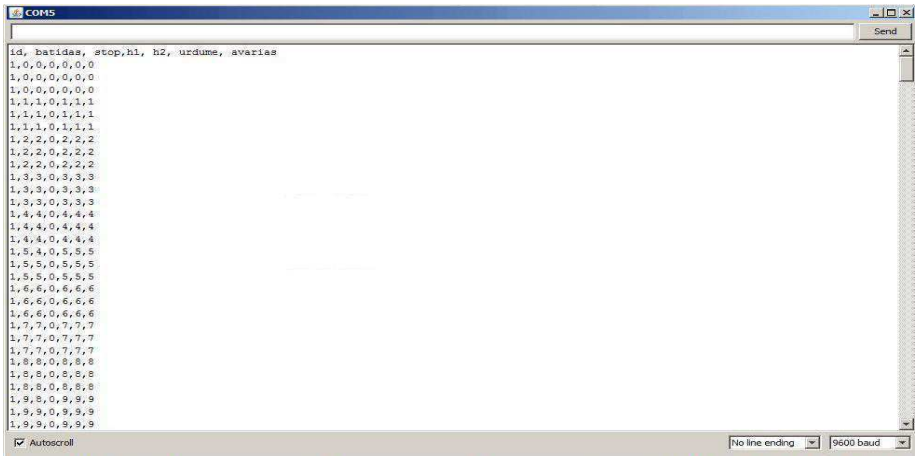


Figura 26. Simulação da contagem de pulsos nas entradas digitais.

4.6.2 TESTES DA AQUISIÇÃO DE DADOS E MAIS FUNCIONALIDADES

Uma vez implementados os testes anteriores e verificado ser a plataforma capaz de contabilizar a ocorrência de cada evento de maneira eficiente, podem-se ampliar suas funcionalidades. Uma forma bastante simples para que isso seja feito é de se dividir a mensagem anterior em três informações: a primeira, uma mensagem periódica da quantidade de batidas do pente, uma segunda mensagem, a de erro, e por fim uma mensagem de erro corrigido.

Paralelamente à mensagem que mensura a eficiência da máquina, contabilizando a quantidade de batidas do pente de maneira periódica, envia-se a mensagem de erro apenas dada a ocorrência de algum evento de erro, se apresentado seus contadores: de erro e o de batidas, até o momento de ocorrência do erro. Uma mensagem complementar indica que o erro fora corrigido, uma vez verificada a mudança no contador das batidas, permitindo que se saiba quanto tempo demandou-se para se concluir a correção do erro, permitindo mensurar a eficiência do funcionário ou mesmo da máquina, avaliando caso exista algum problema mecânico ou elétrico na mesma.

O algoritmo de funcionamento do programa foi elaborado fundamentando-se na observação constante das entradas e o incremento de variáveis, dado a ocorrência dos eventos.

De modo geral, o princípio de envio das mensagens é:

I. Uma vez que transcorridos 30s, envia-se a mensagem contendo o número de batidas do pente nesse intervalo de tempo e zera-se a variável, no intuito de se evitar a manipulação de números muito grandes e o estouro da variável em momento inapropriado;

II. Caso não haja incremento da variável que mensura as batidas do pente, demonstra-se que houve uma parada da máquina. Neste caso, a mensagem de ocorrência de erro é enviada, dado que se a máquina encontra-se parada é provável a ocorrência de erro e incremento de alguma das variáveis que os contabilizam;

III. Uma vez enviada a mensagem anterior, assim que reconhecido o incremento na variável de batidas, indica-se que a máquina tenha voltado ao seu funcionamento, de modo que outra mensagem deva ser enviada, indicando que o erro foi corrigido, reiniciando o ciclo (Figura 27).

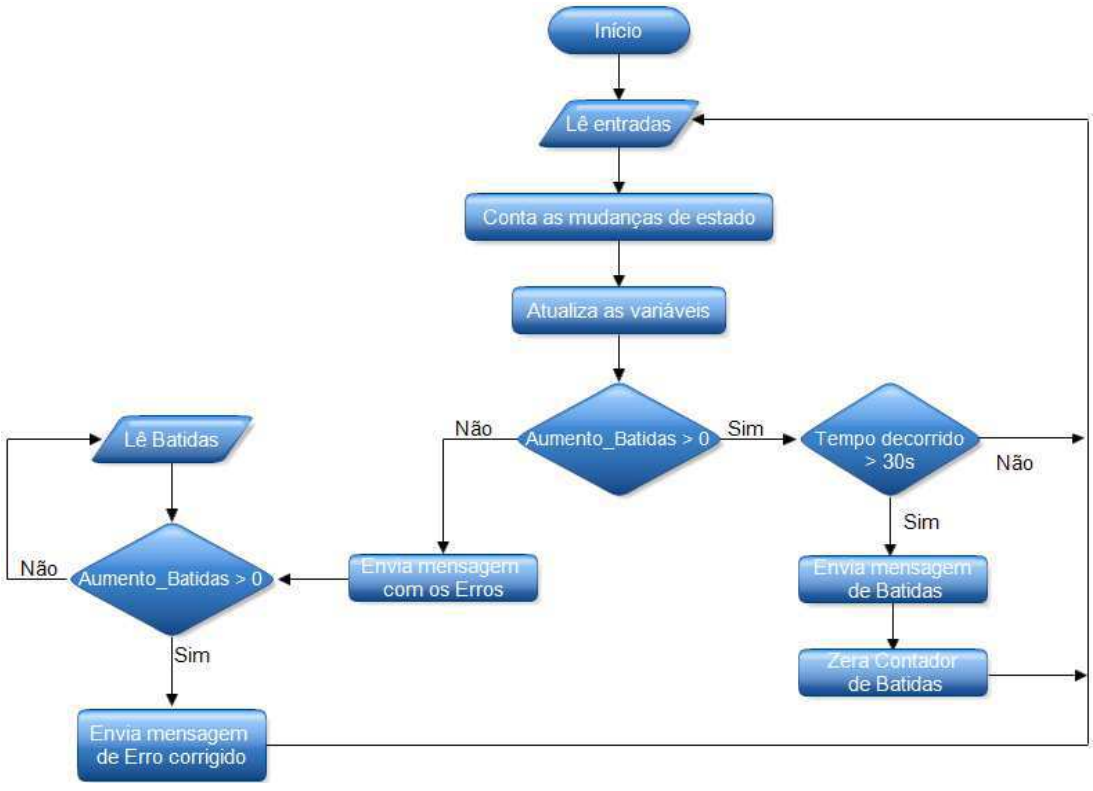


Figura 27. Fluxograma da lógica do programa, adicionando-se funcionalidades.

4.4.2.1 PROCEDIMENTO DE TESTE

Com intuito de simular o funcionamento da máquina, estabeleceu-se inicialmente a simulação de seus sinais, lançando-se mão da utilização de outro microcontrolador: PIC 16F628A, em conjunto com um botão de intervenção sobre o sinal, representando o erro.

Com a finalidade de simplificar-se tal simulação, restringiu-se à geração de apenas um dos sinais de erro pelo PIC, elegendo-se devido às suas características peculiares em relação aos demais, o sinal de urdume, de modo que satisfatoriamente os outros sinais de erro são reproduzidos variando-se os valores de tensão adequadamente no próprio *Protoboard*, através do conjunto de resistores empregados no circuito condicionador associados ao circuito teste.

De acordo com as características dos sinais já analisados, o sinal de urdume, a ser gerado, consiste em submeter inicialmente a saída do PIC a nível lógico alto, de modo a comutar, assim que acionado o botão, para o nível lógico baixo, durante $110\ ms$ e retornar-se ao nível lógico alto.

Quanto a simulação do sinal de batidas, procede-se da seguinte maneira: estabelece-se previamente para qual velocidade de funcionamento da máquina deseja-se simular. No estudo em pauta, optou-se por gerar um sinal de acordo com os valores reais de $480\ RPM$. Sendo assim, a partir da frequência atribuída obtêm-se o tempo por ciclo de $125\ ms$, mostrados através das equações (12), (13) e (14).

$$f = 480\ \text{RPM} = \frac{480}{60}\frac{\text{ciclos}}{\text{s}} \tag{12}$$

$$f = 8\ \text{Hz} \tag{13}$$

$$Tempo = \frac{1}{f} = \frac{1}{8} = 125\ ms \tag{14}$$

```
//Programação do microcontrolador PIC16F628A, utilizando a ferramenta PIC C.
//Código de geração de sinais.

#include<16F628A.h>
#fuses XT,NOWDT,NOPROTECT,PUT
#use delay(clock=4000000)

#define pulso          pin_b3                    // pino de dados d7 do LCD.
#define led            pin_a1
#define urd            pin_b2

#define lcd_enable     pin_a3                    // pino de cont  cs do LCD.
#define lcd_rs         pin_a0                    // pino de cont  rs do LCD.
#define lcd_d4         pin_b5   // pino de dados d4 do LCD.
#define lcd_d5         pin_b4                    // pino de dados d5 do LCD.
#define lcd_d6         pin_b1                    // pino de dados d6 do LCD.
#define lcd_d7         pin_a2             // pino de dados d7 do LCD.

#include <mod_lcd.c>// Sub-rotina do LCD 16x2.

//============================================================================*/

void main()
{

// As duas proximas intruções são de configuração das portas

set_tris_a(0);
   set_tris_b(0b00000001);


output_high(urd); //Início da geração do sinal de urdume
output_high(led);

  delay_ms(1000);

  while(1)
{
if(!input(PIN_B0)){ //Se acionado o botão

       //Geraçãoda comutação do sinal de urdume
output_low(urd);
delay_ms(110);
        output_high(urd);
        delay_ms(11000);

       }
  //Geração do sinal de batidas
output_high(pulso);
delay_ms(70);
  output_low(pulso);
delay_ms(55);

}
```

Seja para a implementação virtual ou real utilizando-se PIC, é conveniente avaliar a elaboração de seu circuito oscilador. O *clock* especificado do microcontrolador é de $4\,MHz$, determinando sua velocidade de operação. Sendo suficiente para a implementação correta do circuito oscilador, adotar-se a configuração apropriada (Figura 28), e respeitarem-se os valores de capacitância de acordo coma frequência de oscilação, conforme demonstrado através da Tabela 2.

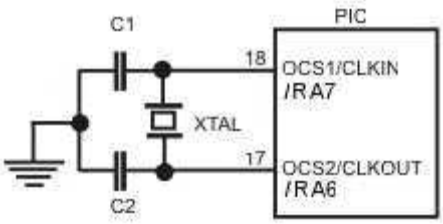


Figura 28. Configuração de Clock do PIC.
(Fonte: ARNEROBOTICS. Disponível em: <http://www.arnebobotics.com.br/eletronica/Microcontrolador_PIC_teoria_2.htm>. Acesso em: 15 fev. 2013).

Frequência	Capacitores recomendados
$32\ KHz$	$68\ pF - 100\ pF$
$200\ KHz$	$15\ pF - 33\ pF$
$2\ MHz - 4\ MHz$	$15\ pF - 33\ pF$
$4\ MHz - 20\ MHz$	$15\ pF - 33\ pF$

Tabela 2. Convenção dos valores de capacitância de acordo com a frequência de oscilação.

Dotado do arcabouço necessário à elaboração do circuito de geração dos sinais, utiliza-se a plataforma de simulação de circuitos *Proteus*®, avaliando-se o funcionamento do circuito teste (Figura 30). Podendo-se inclusive, carregar o código fonte gerado pelo *PIC C Compiler*®, demonstrado anteriormente, avaliando-se tanto o circuito elétrico quanto a geração dos sinais quanto à correspondência da informação desejada.

Assim como no circuito físico, demonstrado mais adiante, a geração do sinal de batidas é simulada conectando-se ao mesmo o **LED BAT**, que piscar com a mesma frequência programada do sinal.

Além disso, simula-se a comutação do sinal de urdume, mudando de estado durante os $110\ ms$, assim que acionado o botão $\boldsymbol{t}$, fazendo com que o **LED URD** acenda e o **LED BAT** apague. Após o tempo programado de 11 segundos, altera-se o *status* dos LEDs, indicando retorno de funcionamento da máquina.

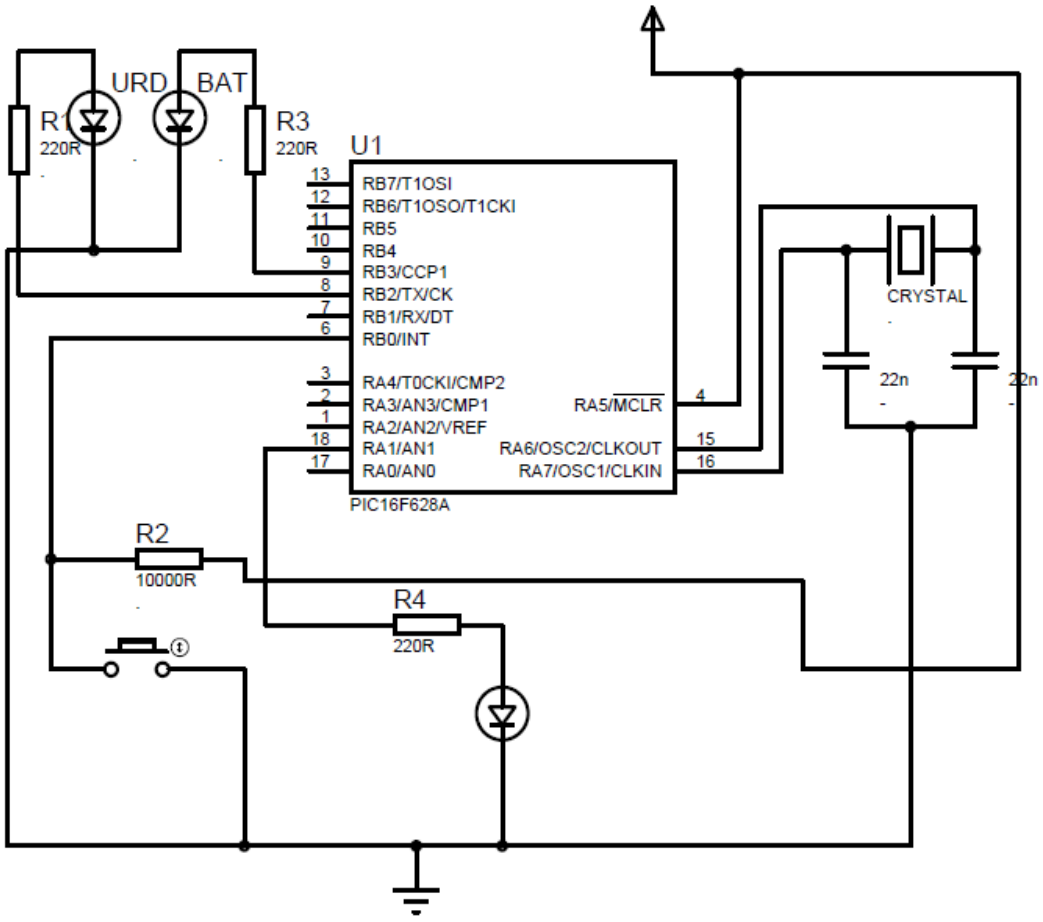


Figura 29. Simulação no Proteus da Geração dos sinais da máquina.

Montando-se agora o circuito já simulado obtêm-se nas portas de entrada do Arduino os sinais com as características já descritas, gerados pelo PIC (Figura 31).

Em uma de suas portas, o PIC gera o sinal de batidas, cuja quantidade de comutações é contabilizada pelo Arduino e apresentado após 30s. Como mencionado, a frequência escolhida de comutação é de $480\ RPM$, fazendo com que a cada minuto sejam contabilizadas 480 comutações, de acordo com os valores apresentados (Figura 32).

Conforme também apresentado, simula-se a geração do sinal de erro de urdume, em que apertando um botão, a mensagem de erro é enviada e a variável de urdume incrementada. Exatamente como previsto em simulação, passados alguns segundos, a variável de batidas continua a ser incrementada, de modo que a mensagem de erro corrigido é enviada, até que a mensagem de batidas seja novamente enviada.

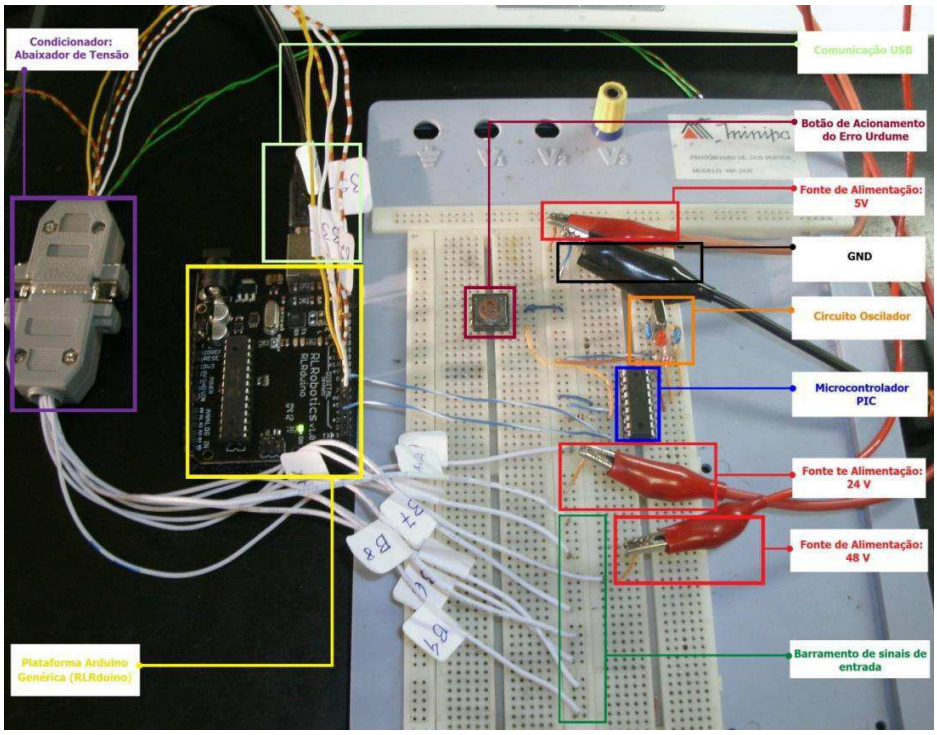


Figura 30. Circuito de Simulação dos sinais da Máquina TSUDAKOMA.

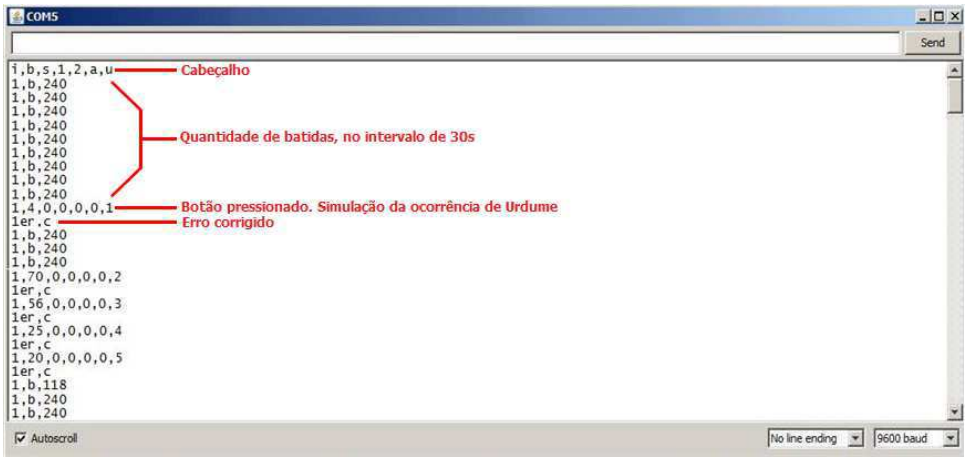


Figura 31. Informações mostradas na Serial, captadas do circuito teste.

4.6.3 Método de Contagem apenas dos Erros Causadores das Paradas

Uma consideração obrigatória para que a contagem de eventos de erro seja fiel à realidade é impedir que fosse contabilizado algum evento de erro quando a máquina estiver parada.

Se eventualmente algum dos erros ocorre em sequência, seja por descuido do operador, disparando, por exemplo, o botão de Stop mais de uma vez seguida, ou quaisquer motivos de qualquer outra ordem, um segundo evento, que não veio a ocasionar a parada da máquina, deverá ser desprezado. Assim, o desenvolvimento de

um método de contagem dos eventos apenas quando a máquina estiver em funcionamento tornou-se imperativo. Para isso, adicionou-se a variável **msg_enviada**.

Estabeleceu-se a condição de leitura dos erros, avaliando-se o valor da variável **msg_enviada**. Uma vez apresentada a mensagem de erro, a variável **msg_enviada** é setado em 1, evitando-se que nesse momento se dê a leitura dos erros. Após o erro ser corrigido, ocorrendo-se a mudança no valor no contador de batidas, a variável **msg_enviada** é setada novamente em 0, procedendo-se a leitura das portas responsáveis pela contagem dos erros novamente.

4.7 IMPLEMENTAÇÃO DA APRESENTAÇÃO DOS DADOS: TESTES EFETUADOS NA MÁQUINA

Com a finalidade de visualizar e interferir na eficiência do protótipo é de fundamental importância que se programe a apresentação desses dados.

Essa apresentação é bastante facilitada pelo próprio IDL do Arduino, através do monitor serial. O monitor serial exibe os dados seriais enviados de seu Arduino, é uma ferramenta bastante útil para apresentação dos dados, tal como depuração do código.

4.7.1 TESTES DE PRECISÃO DE CONTAGENS

São fornecidas funções de inicialização e escrita na serial. Na inicialização, a taxa de transmissão (Baud Rate) foi configurada para o valor padrão de 9600 *baud*.

```
//Inicialização da Serial

voidsetup() {

Serial.begin(9600);

// Enviando a sequência de contadores dos eventos para porta Serial

String msg = String(id) + "," + String(count_batidas) + "," +
String(count_stop)+ "," + String(count_h1) + "," +
String(count_h2)+ "," + String(count_urd) + "," +
String(count_avarias);
Serial.println(msg);
```

Fornece-se, ainda, um atalho para a serial, possibilitando a visualização de tais informações. No estudo em pauta, a informação a ser enviada consta inicialmente de um cabeçalho, com todas as características das informações a serem apresentadas.

```
//Enviando cabeçalho

String header = "ID, Batidas, Stop, H1, H2, Urdume, Avarias";

Serial.println(header);
```

Desse cabeçalho constam: a identificação da máquina, o número de batidas do pente efetuadas no intervalo de 30 segundos e a abreviação dos tipos de erros; H1, H2, urdume, avaria e stop.

Em um teste prévio, estabeleceu-se a verificação da contagem das mudanças de estado no pulso simulado. Uma vez obtidos dados satisfatórios, partiu-se para a avaliação dos resultados na máquina. Interligando-se todos os componentes do sistema proposto (Figura 32).

Sabendo que a velocidade da máquina é constante e igual a aproximadamente $510RPM$, pode-se estimar que a cada 30 segundos, aproximadamente 255 batidas são realizadas. Avaliando-se as informações obtidas, nota-se que a contagem corresponde ao esperado, demonstrando-se a confiabilidade da informação, uma vez que sempre que simulado a ocorrência de erro, o mesmo era incrementado na variável a ser mostrada no monitor serial.

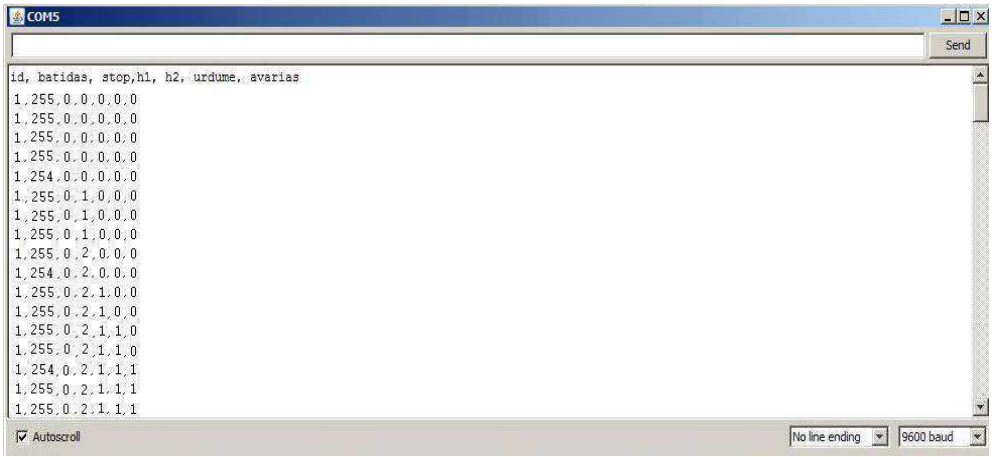


Figura 32. Resultado do Arduino ligado à máquina.

4.7.2 TESTES DA AQUISIÇÃO DE DADOS E MAIS FUNCIONALIDADES

Uma vez implementado o teste inicial e verificada a corretude das informações obtidas, pode-se testar diretamente na máquina a adição de mais funcionalidades, tais como; o envio da mensagem de batidas do pente periodicamente, a cada 30 segundos, o envio da mensagem de erro, sempre que o evento ocorra e uma mensagem complementar de que o erro fora corrigido.

A velocidade real da máquina é de $450\,RPM$, espera-se, portanto, que ocorram $225\,RPM$ por interação. Além disso, acompanhando-se o funcionamento da máquina durante algumas horas, notamos a corretude das informações mostradas, quanto a ocorrência de erro, em que a respectiva variável é de fato incrementada e a mensagem de erro é apresentada, tal como ocorre quando o erro é corrigido.

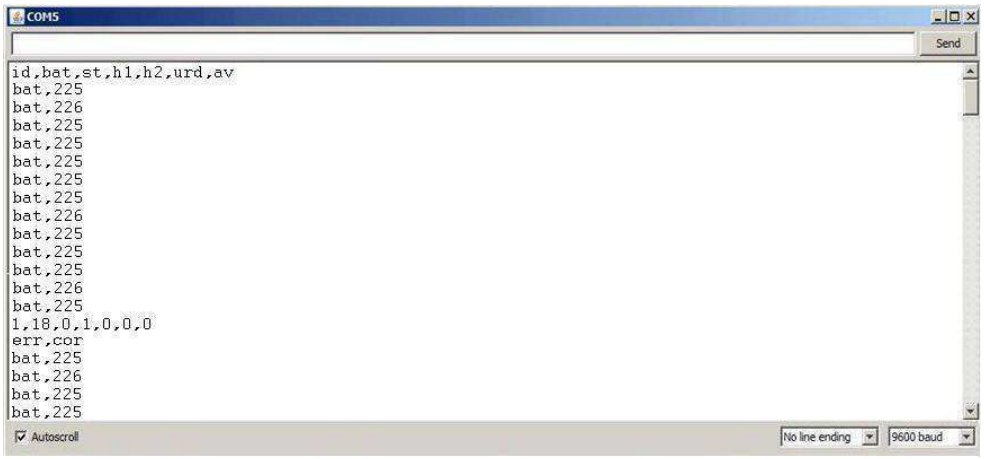


Figura 33. Funcionamento o sistema agregado à máquina.

5 CONCLUSÃO

Neste trabalho, teve-se como objetivo, validar a utilização eficaz da placa Arduino em ambiente industrial, lançando-se mão de um conhecimento razoável da plataforma e do sistema sobre análise.

O sistema eleito para ilustrar o conceito diz respeito à automatização do processo de tecelagem, destinado à máquina TSUDAKOMA ZAX, no âmbito da indústria têxtil. A ZAX já é dotada de todo sensoriamento para obtenção das informações necessárias e permite uma saída distinta para cada ocorrência dos eventos, além de mudança no nível de tensão.

Os dados a serem monitorados da máquina são basicamente: o número de batidas do pente da máquina, remetendo-se ao rendimento da mesma; o erro de trama, inserção errônea do fio de trama, o erro de urdume, rompimento desse tipo de fio, falha elétrica e parada mecânica.

É designada ao Arduino, dessa forma, a função de captar esses sinais em suas entradas digitais e contabilizar a ocorrência desses eventos. Por fim, a próprio IDE da Arduino fornece um meio de visualização de tais informações; um monitor serial. Neste, pode-se constatar a veracidade das informações apresentadas, seja em termos de velocidade da máquina, em que se comparou a velocidade modificável da máquina com o número de eventos apresentado no monitor; seja quando simulada a ocorrência de um erro e a variável a ser apresentada, respectiva ao erro, é incrementada imediatamente.

A utilização do Arduino é justificada também, pela simplicidade com que se podem adicionar funcionalidades. Através dos seus *Shields*, podendo de maneira eficiente, otimizar o sistema além de captar essas informações, sendo possível enviá-las, e acompanhá-las remotamente, elaborando-se um pequeno supervisório.

BIBLIOGRAFIA

[1] FRANÇA, José.A. **Sistemas de Aquisição de Dados Baseados em Microcontrolador**. 1997. 110p. Dissertação (Mestrado em Engenharia Elétrica). Universidade Federal da Paraíba, Campina Grande, 1997.

[2] DENARDIN, Gustavo W. **Microcontroladores**. 2010. 34 p. Apostila do Curso de Eletrônica. Universidade Tecnológica Federal do Paraná.

[3] McROBERTS, Michael. **Arduino Básico**. 1.ed. Tradução de Rafael Zanolli. São Paulo: Novatec, 2011, 456 p.

[4] ARDUINO. Plataforma de Prototipagem Eletrônica Open-source. **Arduino**. 2005. Disponível em: <http://www.arduino.cc/en/Main/Products>. Acesso em: 15 fev. 2013.

[5] ATMEL. **Datasheet: 8 bit AVR Microcontroller with 4/8/16/32k Bytes In- System Programmable Flash. ATmega 48 PA, 88 PA, 168 PA e 328 P**. 2009. Disponível em: <http://www.atmel.com/Images?doc8161.pdf>. Acesso em: 20 fev. 2013.

[6] SOARES, Márcio.J. **Como Utilizar o Conversor AD do Microcontrolador PIC**. 2010. Disponível em: <http:/www.arnerobotics.com.br/eletronica/Medida_Analógica_W_PIC_html>. Acesso em: 18 fev. 2013.

[7] DINIS, Francisco J.V. **Sistema de Instrumentação de Baixo Custo compatível com o LabVIEW.** 2010. 150 p. Dissertação (Mestrado em Engenharia de Telecomunicações e Redes). Universidade da Madeira, Funchal, Portugal, 2010.

[8] MARGOLIS, Michael. **Arduino Cookbook**. 2.ed. Sebastopol: O'Reilly Media, 2011, 726 p.

[9] MULTILOGICA-SHOP. Open-Source Hardware. Multilógica-Shop, 1990. Disponível em: <http://www.multilogica-shop.com/catalogo/shields-0>. Acesso em: 22 fev. 2013.

[10] TIMMIS, Harold. **Pratical Arduino Engineering**. 1.ed. Nova York: Spring Science, 2011, 328 p.

[11] BANZI, Massimo. **Getting Started With Arduino**.1.ed. Sebastopol: O'Reilly Media, 2008, 128 p.

[12] ARDUCOPTER. Piloto Automático baseado em Arduino. **ArduCopter**, 2011. Disponível em: <http://www.code.google.com/p/arducopter>. Acesso em: 17 fev. 2013.

[13] OPENCOPTER. Multi-rotores e Eletrônicos.**OpenCopter**, 2013. Disponível em: <http://www.opentronics.com.br>. Acesso em: 18 fev. 2013.

[14] REPRAP Fabricação de Máquinas de Auto-replicação. **Reprap**, 2012. Disponível em: <http:/www.reprap.org/wiki/Main_Page>. Acesso em: 18 fev. 2013.

[15] HAUS, IC. **Interfaceamento de Microcontroladores com o Mundo Industrial**. Tradução da IC-BR.Alemanha, 2011.

[16] ABIMAQ. **A História das Máquinas**. 1.ed. São Paulo: Magma, 2006, 168 p.

[17] SANTOS, Felipe; GOMES, Winston. **Tear Jato de Ar**. 2010. 34p. Seminário (Curso de Operador Polivalente de Tecelagem Plana). SENAI de Americana, São Paulo, 2010.

[18] TSUDAKOMA ZAX. **Electronics Generalidades**. Versão 5.01. 1998.