USF - UNIVERSIDADE SÃO FRANCISCO
Curso de Engenharia Elétrica

ANTONIO RODRIGUES COSTA
LUIZ ALBERTO SOARES DA SILVA
LUIZ CARLOS CASSIMIRO

PLACA MICROCONTROLADA PARA A AQUISIÇÃO E CONDICIONAMENTO DE SINAIS ELÉTRICOS

Itatiba
2014

ANTONIO RODRIGUES COSTA **R.A. 002200901236**
LUIZ ALBERTO SOARES DA SILVA **R.A. 002201200159**
LUIZ CARLOS CASSIMIRO **R.A. 002200700907**

PLACA MICROCONTROLADA PARA A AQUISIÇÃO E CONDICIONAMENTO DE SINAIS ELÉTRICOS

Monografia apresentada ao curso de Engenharia de Elétrica da Universidade São Francisco, sob orientação do Prof.º Especialista André Renato Bakarereskis, como exigência para conclusão do curso de graduação.

Itatiba
2014

ANTONIO RODRIGUES COSTA
LUIZ ALBERTO SOARES DA SILVA
LUIZ CARLOS CASSIMIRO

PLACA MICROCONTROLADA PARA A AQUISIÇÃO E CONDICIONAMENTO DE SINAIS ELÉTRICOS

Monografia aprovada pelo programa de graduação do curso de Engenharia de Elétrica da Universidade São Francisco para a obtenção do título de engenheiro eletricista.

Área de concentração: Engenharia Elétrica.

Data da aprovação: 05/12/2014

Banca examinadora:

Prof.º André Renato Bakalereskis - Especialista (orientador)
Universidade São Francisco

Prof.ªAnnete Faesarella - Doutora (examinadora)
Universidade São Francisco

Prof.ºJoão Alex Vaz Franciscon - Graduado (examinador)
Universidade São Francisco

“No que diz respeito ao empenho, ao compromisso, ao esforço, à dedicação, não existe meio termo. Ou você faz uma coisa bem feita ou não faz.”

Ayrton Senna

RESUMO

Qualquer sistema físico para ser devidamente compreendido necessita ser detalhadamente estudado, analisando seu comportamento isoladamente ou em conjunto com o meio onde se encontra. Nesse aspecto foi desenvolvido e demonstrado o funcionamento de um sistema microcontrolado de baixo custo para aquisição e condicionamento de sinais elétricos aplicável em indústrias, laboratórios e universidades com interação em tempo real entre o fenômeno físico e sistema de monitoramento. Apresentam-se também as considerações necessárias ao projeto de sistemas de aquisição e condicionamento e aplicações diversas em locais remotos e hostis. A utilização de um microcontrolador com tecnologia ARM ARDUINO com código aberto foi em um computador com *interface* serial e ambiente de programação gráfica LabVIEW que compõem o conjunto.

Palavras-chave: Aquisição e condicionamento de sinais, microcontroladores, programação, ARDUINO e LabVIEW.

ABSTRACT

Any physical system to be properly understood in detail needs to be studied by analyzing behavior alone or in conjunction with the environment where it is. In this aspect was developed and demonstrated the operation of a low-cost microcontroller system for the acquisition and conditioning of electrical signals applicable in industries, laboratories and universities with real-time interaction between the physical phenomenon and monitoring system. We present also the considerations necessary for the project acquisition and conditioning systems and various applications in remote and hostile locations. The use of a microcontroller ARDUINO technology and *ARM* code was opened on a computer with a serial interface and LabVIEW graphical programming environment comprising the assembly.

Keywords: Acquisition and signal conditioning, microcontrollers, programming, ARDUINO and LabVIEW.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTAS DE ABREVIATURAS E SIGLAS

AD – Analógico/digital

ALU – *Arithmetic logic unit*

ARM – *Advanced RISC Machine*

CISC – *Complex Instruction Set Computer*

CMOS – *Complementary metal–oxide–semiconductor*

CPU – *Central Processing Unit*

IEEE – *Institute of Electrical and Electronic Engineers*

IGBT – *Insulated Gate Bipolar Transistor*

LabVIEW – *Laboratory Virtual Instrument Engineering Workbench*

LIFA – LabVIEW *Interface for* Arduino

LAN – *Local area network*

LSB – *Least significant bit*

LXI – *LAN extensions for Instrumentation*

MIT – *Massachusetts Institute of Technology*

MOSFET – *Metal Oxide Semiconductor Field Effect Transistor*

MSB – *Most significant bit*

PWM – *Pulse width modulation*

RISC – *Reduced Instruction Set Computer*

TTL – *Transistor-Transistor Logic*

USB – *Universal Serial Bus*

VI – *Virtual instrument*

VXI – *LAN for Instrumentation*

SUMÁRIO

1. INTRODUÇÃO

Qualquer sistema físico para ser devidamente compreendido necessita ser detalhadamente estudado. Segundo P.D Lawrence e K. Mauch (1998) na indústria mecânica o controle de robôs e máquinas não seria possível sem dados exatos sobre suas condições de funcionamento. Assim como na medicina a maior parte das técnicas utilizadas para o diagnósticos baseiam-se no conhecimento de variáveis biológicas através de equipamentos altamente complexos (P.M. Forgues e M. Goldeberg, 1979).

A dificuldade em observar esses sistemas, controlar as diferentes características e estímulos, dependendo da análise, motiva o desenvolvimento de um sistema programável que permita a aquisição e transdução destes estímulos, sua análise e apresentação automática de tais fenômenos.

A simples análise isolada de um ponto de um único estímulo, seja esse, por exemplo, um sinal de corrente ou tensão, não, necessariamente, retrata a condição geral do sistema. Muitas vezes múltiplas análises em pontos e períodos distintos de um mesmo sistema são necessárias (ROSÁRIO, 2005, p.IX).

Nesse sentido o presente trabalho descreve a criação de um sistema de aquisição e condicionamento de sinais elétricos, onde o usuário através da programação gráfica em LabVIEW coordena a sequência de ações a serem realizadas.

Desta maneira há flexibilidade e praticidade para análise de qualquer sistema, seja simples ou complexo, reduzindo o tempo no desenvolvimento de métodos e estruturas de análise.

2. OBJETIVO

Desenvolver um sistema microcontrolado de código aberto para a aquisição e condicionamento de sinais elétricos em tempo real. O sistema operará sob a supervisão de um computador conectado a este através da porta de comunicação serial. Permite–se ao usuário elaborar em ambiente gráfico a lógica de controle que atenderá a sua necessidade, podendo assim ser empregado para a validação de projetos e protótipos.

3. METODOLOGIA

- Realizar levantamento bibliográfico acerca dos temas: sistemas de aquisição e condicionamento de sinais elétricos, microcontroladores *ARM*, comunicação serial, sensores, programação gráfica em LabVIEW.
- Definir o microcontrolador que atenda as exigências do projeto;
- Desenvolver o *firmware* do microcontrolador em linguagem *C++*;
- Desenvolver as bibliotecas em LabVIEW para o usuário.
- Criação de circuito de aquisição de sinais analógicos e digitais.
- Criação de circuito de condicionamento de potência em corrente contínua.
- Desenvolver um protótipo do sistema de aquisição.
- Utilizar o sistema para o controle de um motor de corrente contínua e analisar seu comportamento em diversas situações de funcionamento.
- Validar o protótipo desenvolvido.

4. JUSTIFICATIVA

Tornar processos e análises mais confiáveis e práticos é uma necessidade diária exigida para a observação, compreensão e validação de sistemas físicos, sejam estes quaisquer. Nesse sentido a automação visa melhorar a eficiência, aumentar a produtividade e reduzir custos, introduzindo novas técnicas de controle para fazer o gerenciamento das ações mais apropriadas a cada etapa de um processo (SILVEIRA; SANTOS, 1999, p.23).

5. RESULTADOS ESPERADOS

Esse trabalho busca criar um sistema microcontrolado de código aberto com *interface* gráfica de fácil e rápida utilização para a aquisição e condicionamento de sinais elétricos em produtos e protótipos. Pretende–se também através do registro dos dados observados ampliarem a confiabilidade dos ensaios, possibilitando a revisão e validação do conjunto nas condições próximas de funcionamento.

6. REVISÃO BIBLIOGRÁFICA

6.1 SISTEMA DE AQUISIÇÂO DE DADOS

Os processos físicos nas mais diversas áreas dependem fortemente do conhecimento e monitoramento das grandezas associadas. Décadas passadas, no advento da eletrônica, os sistemas de aquisição de dados eram constituídos por instrumentos de medição sem nenhuma interação ou controle automático, porém a dificuldade de gestão dos meios era enorme e impossível para sistemas extramente completos (BOLTON, W. 2005).

Nesse sentido os organismos de normalização como, *ANSI*, *EIA* e *IEEE*, criaram padrões que possibilitaram uma melhor interação e controle entre o processo e instrumentos de medição (B. Mangolds. Rudi, 1971).

Tais padrões especificam e normalizam o método de controle remoto de instrumentos, protocolos de comunicação, meios físicos de controle. Com isso os fabricantes de equipamentos puderam criar plataformas e sistemas comuns entre si, padrões esses conhecidos e aceitos mundialmente.

Dentre esses se destacam *IEEE-488, VXI, LXI, RS232 e RS485* (*National Instruments*, 1993). Os padrões *USB* e *ETHERNET* foram acrescidos à lista na última década, porém sua ascensão e empregabilidade em equipamentos já são notórias.

O processo pelo qual se deseja controlar ou adquirir os dados é que definirá as características necessárias do sistema. Isso é facilmente visualizado em sistemas que atuam em ambientes hostis e de difícil acesso, como monitoramento de gases tóxicos, perfurações marítimas ou exploração aeroespacial.

Nesse sentido é possível subdividir o sistema, Figura 1, criando blocos de aquisição e condicionamento, análise e apresentação e por último armazenamento dos dados coletados e controlados.

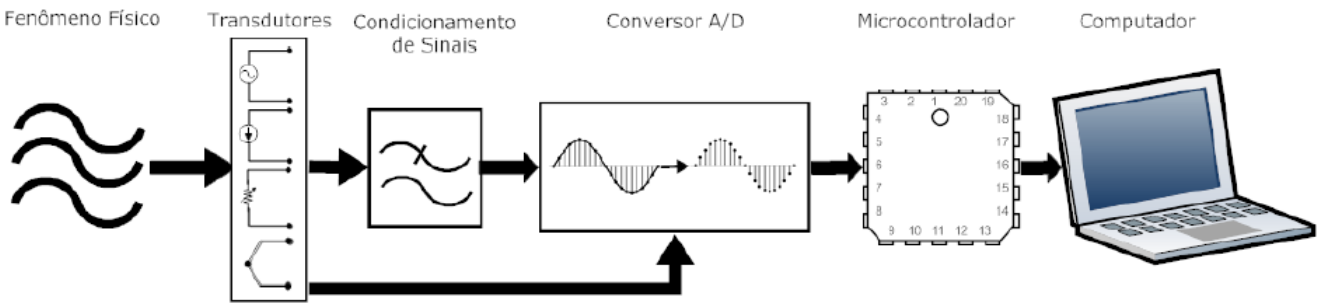

Figura 1: Estrutura básica de um sistema de aquisição de dados

Fonte: própria.

O observador não necessáriamente é um indíviduo. Este pode ser um algoritmo de controle que monitora e atua sobre o sistema.

6.2 TRANSDUTORES

Transdutores são componentes eletrônicos capazes de transformar grandezas físicas em sinais e impulsos elétricos a serem utilizados pelo sistema de aquisição e condicionamento de dados. São exemplos de transdutores: termopares, termistores, microfones, sensores de luminosidades, Figura 2.

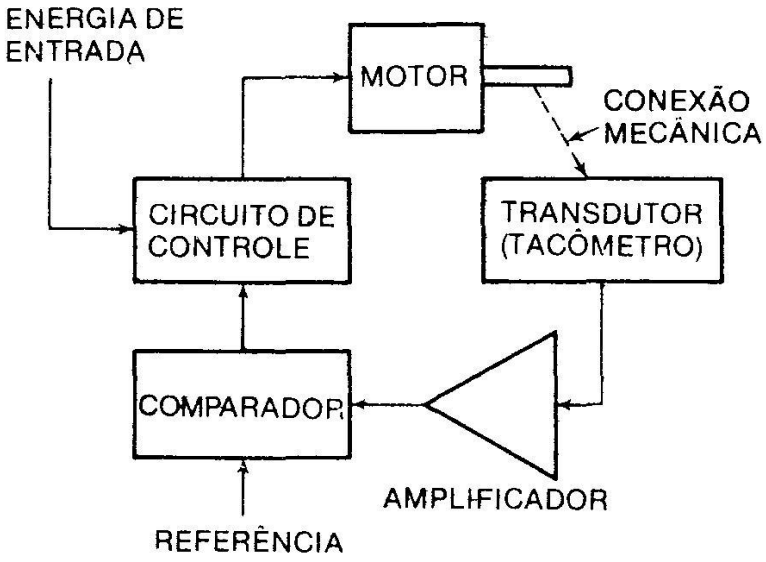

Figura 2 - Aplicação típica de um transdutor

Fonte: www.eletronicos.etc.br.

Esses componentes podem ser formados a partir de um ou mais sensores que alteram suas propriedades elétricas passivas (resistência, capacitância, indutância) em função da medição da grandeza física. Ou ainda podem ser ativos, os quais geram diretamente um sinal elétrico (corrente, tensão, carga elétrica) quando submetidos à excitação externa provocada pelo fenômeno físico.

6.3 CONDICIONAMENTO DE SINAIS

Os sinais elétricos provenientes dos transdutores precisam ser convertidos em uma forma mais aceitável pelo circuito de aquisição de dados. Os condicionadores, Figura 3, podem amplificar ou atenuar, deslocar nível, detectar vales e picos, linearizar, filtrar e isolar o sistema de medição de maneira exata e segura do fenômeno a ser medido (OLIVEIRA, 2006).

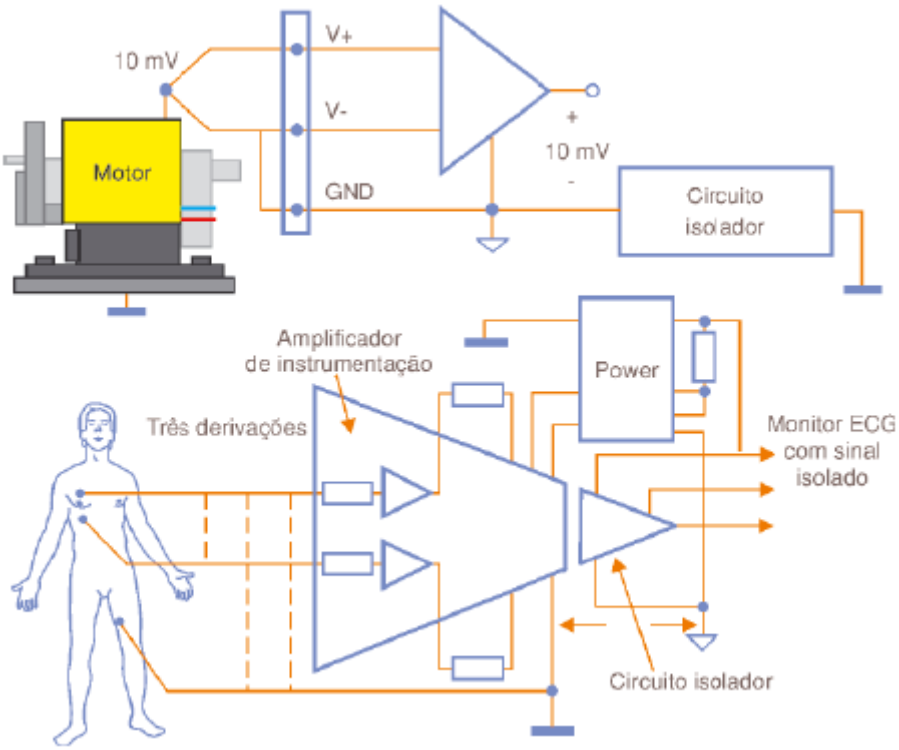

Figura 3 - Aplicações de condicionadores de sinais elétricos

Fonte: www.sabereletronica.com.br.

Dentre as principais funções realizadas pelos transdutores destacam-se:

- **Amplificação e atenuação**

A amplificação e atenuação são fundamentais para adequação das faixas de tensão entre os sinais provenientes do fenômeno físico e conversores A/D (analógico-digital) e melhor a resolução da medição realizada.

- **Deslocamento de nível**

Em alguns casos o fenômeno físico pode assumir valores numa faixa muito diferente, maior ou menor que os níveis aceitos pelo sistema de

aquisição. Assim deslocar o nível destes sinais para a faixa de resolução coberta pelo sistema aumenta a precisão e exatidão da medição.

- **Isolação**

Transitórios nos níveis de aquisição acima dos valores nominais podem ocorrer durante a medição e causar danos ao sistema, riscos aos operadores e inferir na medição de forma a prejudicar a sua análise.

Também podem ocorrer problemas quando o sistema de medição e o sinal adquirido possuem diferenças em relação a sua referência, ou seja, em relação nível de terra. Essa diferença prejudica medição e provoca erros na análise. Com o uso de isolação as tensões de modo-comum, e diferenças entre níveis de terra, são eliminadas.

- **Linearização**

Ela pode ser realizada de maneira analógica ou numérica. Adotando o método analógico é necessário projetar um circuito que possua resposta inversa ao comportamento do transdutor.

Como exemplo: um termistor varia sua resistência exponencialmente e para linearizar sua curva de resposta será necessária adotar um circuito amplificador logaritmo.

No método numérico o sinal proveniente pelo transdutor é convertido digitalmente e suas amostras são linearizadas a partir de equações e cálculos, no entanto a precisão deste segundo método dependerá da resolução do conversor responsável por converter os sinais analógicos em digitais.

- **Filtragem**

Os transdutores podem nos fornecer mais de uma informação sobre o sistema e também pode sofrer interferências eletromagnéticas do meio e outras fontes. Nesse sentido eliminar sinais não indesejados garante maior precisão, estabilidade e exatidão na análise.

6.4 CONVERSOR ANALÓGICO-DIGITAL

Conversor analógico digital, ou simplesmente conversores A/D, é responsável por converter os sinais que variam no tempo e em amplitude em sinais digitais com frequência e amplitudes predefinidos. Na Figura 4 temos a ilustração de um circuito conversor e componentes básicos.

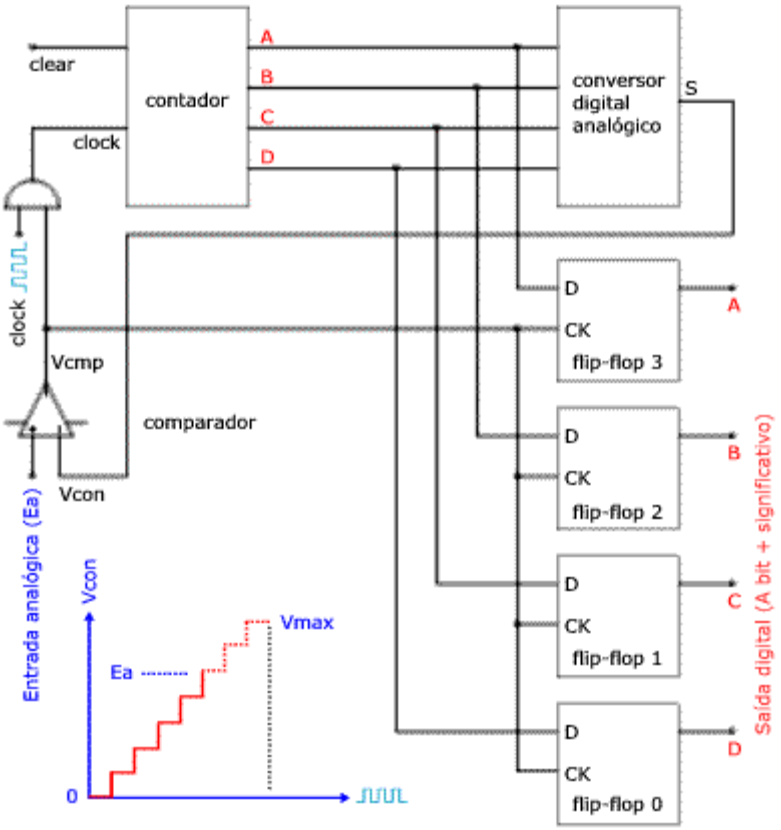

Figura 4 - Conversor analógico digital básico

Fonte: www.mspc.eng.br.

Suas principais características são resolução, a taxa de amostragem e erro de quantificação.

- **Resolução**

A resolução de um conversor representa a mínima mudança de nível de tensão necessária para garantir uma alteração no nível de código de saída. Essa mudança é denominada de *LSB*, menor bit significativo. Assim a resolução de um conversor é determinada pelo seu número de bits.

Como exemplo um conversor com o resolução de 8 bits é capaz de decodificar 255 níveis diferentes de tensão em sua entrada, pois 2^8 corresponde à

256. A faixa de tensão do conversor determinará qual a equivalência entre a informação binária e o nível tensão. Desta forma se a faixa de tensão for 5,0V a resolução por bit será de 19,60mV.

- **Taxa de amostragem**

Ao converter o sinal analógico em digital é necessário definir a taxa em que os novos valores digitais serão amostrados a partir do sinal analógico. A taxa dos novos valores é chamada de taxa de amostragem ou frequência de amostragem do conversor.

A taxa de amostragem deve ser maior que o dobro da frequência no sinal a ser amostrado para que possa sem reproduzido sem erro ou perda de dados. A metade da frequência de amostragem é denominada frequência de *Nyquist* e corresponde ao limite máximo de frequência do sinal que pode ser reproduzido.

Como não é possível garantir que o sinal não contenha sinais acima desse limite, é necessário filtrá-lo eliminando frequências menores que a de *Nyquist*. Aumentando a frequência de amostragem com valor superior a frequência de *Nyquist* aumenta-se a precisão do conversor.

- **Erro de quantificação**

Representa a diferença entre o sinal original e o sinal digitalizado, quanto maior for o número de bits do conversor menor será seu erro. O erro de quantização ocorre devido à resolução finita da representação digital do sinal. Característica essa existente em todos os conversores.

A Figura 5 ilustra a conversão básica de um sinal analógico em um sinal digital quantificado a partir de uma determinada amostragem.

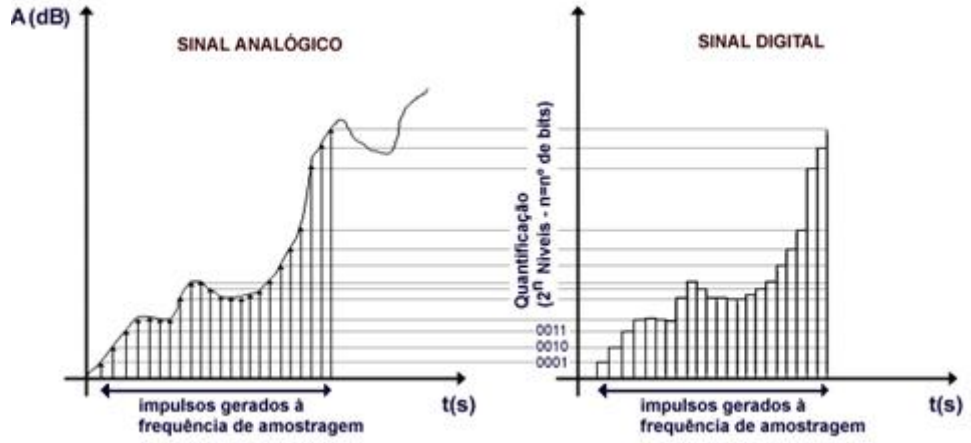

Figura 5 - Conversão A/D

Fonte: www. rafabee.wordpress.com.

6.5 MICROCONTROLADOR

Um microcontrolador é um sistema computacional completo capaz de realizar as operações lógicas previamente definidas pelo usuário programadas em sua memória e também interagir com outros periféricos. Basicamente pode ser dividida em:

- **Unidade Central de Processamento**

A unidade central de processamento é composta por uma unidade lógica aritmética responsável pelos cálculos algébricos, por uma unidade de controle e por unidades de memória especiais conhecidas por registradores, Figura 6.

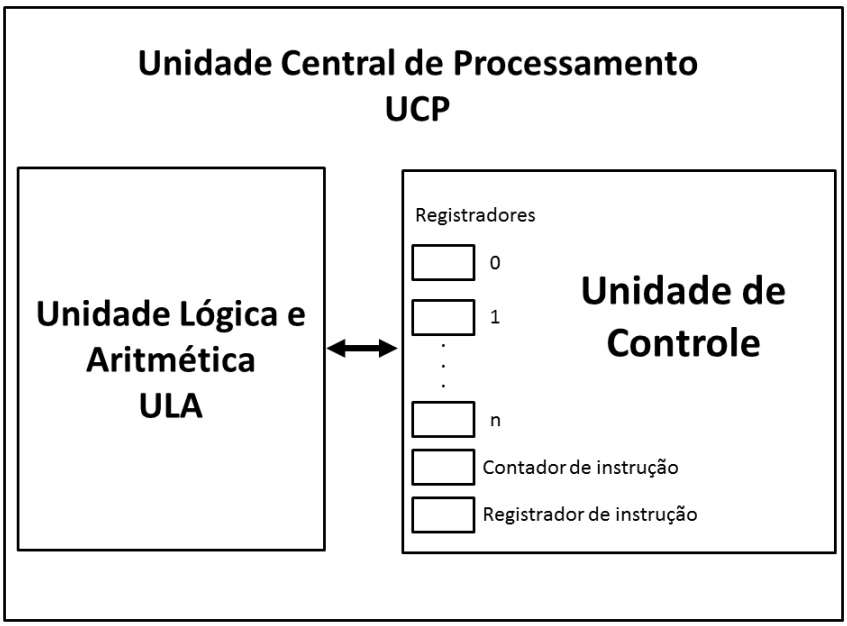

Figura 6 - Unidade central de processamento

Fonte: www. producao.virtual.ufpb.br.

A unidade central de processamento é o centro de todo sistema computacional, e não é diferente quando se trata de microcontroladores. O trabalho desta é executar rigorosamente as instruções de um programa, na sequencia programada, para uma aplicação específica realizando os devidos cálculos aritméticos, quando necessário.

Um programa computacional instrui essa unidade ler e escrever informações na memória de dados, avaliar e atuar sobre suas entradas e saída (DENARDIN).

- **Memória de dados e de programa**

  Essas duas memórias são responsáveis por armazenar a sequência lógica criada, programa, e a cada operação o resultado das ações realizadas, dados.

- **Entradas e saídas**

  Entradas e saídas são formados pelos terminais de acesso e controle de outros periféricos externos ao microcontrolador, através de sinais analógicos e digitais com amplitudes e frequências compatíveis com os limites do controlador. Esses terminais são controlados através do programa contido na memória do microcontrolador.

Existem inúmeros fabricantes e famílias de microcontroladores que operam com memória de dados de 8, 16 ou 32 bits de informação. Agrupando as estruturas temos o bloco funcional básico de um microcontrolador, Figura 7.

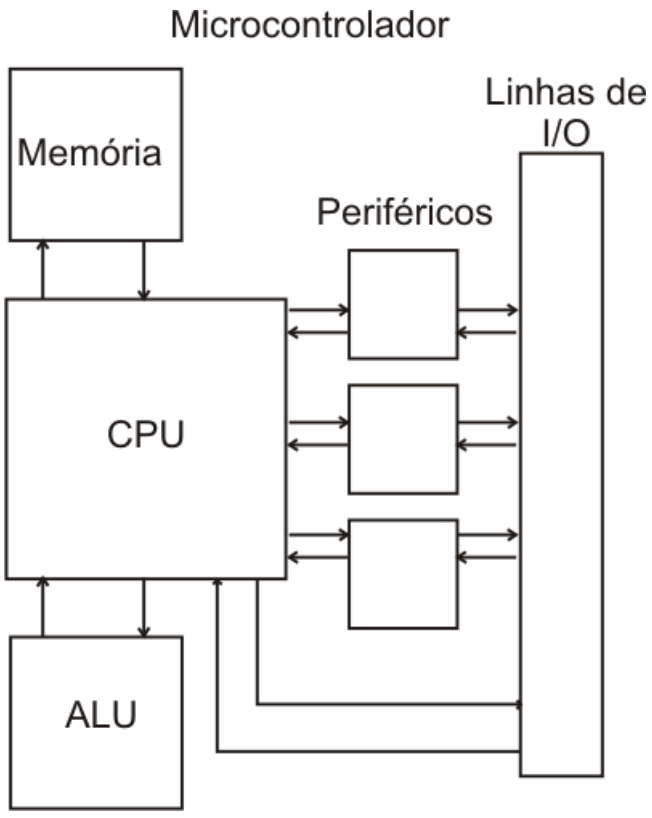

Figura 7 - Bloco funcional de um microcontrolador

Fonte: www.arnerobotics.com.br.

No entanto esses são classificados em dois grupos básicos de acordo com a sua arquitetura de instruções: ***R****educed* ***I****nstruction* ***S****et* ***C****omputer* – computador com um conjunto reduzido de instruções ou ***C****omplex* ***I****nstruction* ***S****et* ***C****omputer* – computador com um conjunto complexo de instruções.

6.6 LABVIEW

É uma linguagem de programação gráfica originária da *National Instruments*, Figura 8. A primeira versão surgiu em 1986 para o *Macintosh* e atualmente existem também ambientes de desenvolvimento integrados para os Sistemas Operacionais *Windows* e *Linux*.

Figura 8 - Programa LabVIEW

Fonte: www.ni.com.

A linguagem LabVIEW é largamente empregada em sistemas de medições e a automação em industrias e universidades. A programação é feita de acordo com o modelo de fluxo de dados, o que oferece a esta linguagem vantagens para a aquisição de dados e para a sua manipulação.

A programação é feita de acordo com o modelo de fluxo de dados, o que oferece a esta linguagem vantagens para a aquisição de dados e para a sua manipulação.

Os programas em LabVIEW são chamados de instrumentos virtuais ou, simplesmente Vis. São compostos pelo painel frontal, que contém a *interface*, e pelo diagrama de blocos, que contém o código gráfico do programa.

O programa não é processado por um interpretador, mas sim compilado. Deste modo seu desempenho é comparável à exibida pelas linguagens de programação de alto nível. A linguagem gráfica do LabVIEW é chamada "G".

Os blocos de funções são designados por instrumentos virtuais. Isto é assim porque, em princípio, cada programa pode ser usado como subprograma por qualquer outro ou pode, simplesmente, ser executado isoladamente. Devido à utilização do modelo do fluxo de dados, as chamadas recursivas não são possíveis, podendo-se, no entanto, conseguir esse efeito pela aplicação de algum esforço extra.

A apresentação gráfica dos processos aumenta a facilidade de leitura e de utilização. Uma grande vantagem em relação às linguagens baseadas em texto é:

- A facilidade com que se criam componentes que se executam paralelamente. Em projetos de grande dimensão é muito importante planear a sua estrutura desde o início (como acontecem nas outras linguagens de programação).

- Existem versões, como a utilizada nesse trabalho, que possibilitam o desenvolvimento e execução do código sem fins comerciais, o que facilita e difunde a aplicação.

As desvantagens do LabVIEW face à programação por texto são, essencialmente:

- Pequenas mudanças podem obrigar a profundas reestruturações do programa, uma vez que sempre que se insere um novo bloco é necessário voltar a ligar os fios e os símbolos para reestabelecer o funcionamento.

- Para evitar confusões de linhas é habitual introduzir mais variáveis do que aquelas que são estritamente necessárias, diminuindo-se assim a velocidade de programação e contrariando-se, de algum modo, o modelo de fluxo de dados.

6.7 ARDUINO

ARDUINO é uma plataforma integrada que possibilita a interação entre os computadores e mundo físico. É uma plataforma *open-source* de computação física baseada em uma placa com um microcontrolador simples e um ambiente de desenvolvimento de código aberto.

Esta pode ser utilizada para desenvolver objetos interativos, tendo entradas a partir de uma variedade de sensores ou interruptores, e atuando e controlando de uma variedade de luzes, motores e outras saídas físicas. Na Figura 9 alguns exemplos deste sistema e símbolos associados.

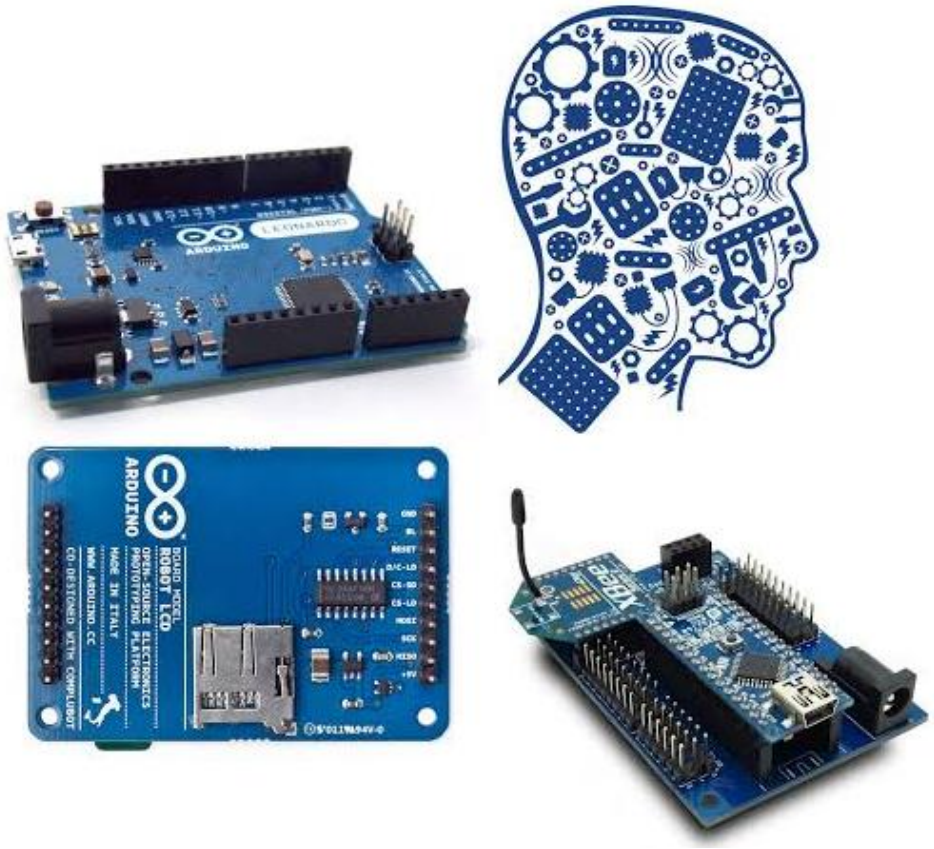

Figura 9 - Plataforma ARDUINO

Fonte: www.google.com.br/imagens

Há muitos outros microcontroladores e plataformas de microcontroladores disponíveis para a computação e interação física, tais como: *Basic Stamp Parallax, Netmedia do BX -24, Phidgets, Handyboard* do *MIT*, e muitos outros oferecem funcionalidade semelhante.

Todas essas ferramentas levam os detalhes complicados de programação. O ambiente de programação ARDUINO é fácil de usar para iniciantes, mas suficientemente flexível para usuários avançados.

Esta plataforma baseia-se em microcontroladores ATMEGA8 e ATMEGA168 da *Atmel*. Os planos para os módulos são publicados sob uma licença *Creative Commons*, para que projetistas possam fazer a sua própria versão do módulo, ampliando-a e melhorando-o.

Mesmo usuários relativamente inexperientes podem construir uma versão de placa para ensaio do módulo, a fim de entender como ele funciona e economizar dinheiro.

O código e ambiente de programação utilizam uma estrutura denominada *SKETCH*, Figura 10, o qual o programador poderá aplicar e utilizar com facilidade seus códigos e rotinas desenvolvidos em linguagem *C/C++.*

Figura 10 - Ambiente de programação ARDUINO

Fonte: http://arduino.cc/en

Neste ambiente é possível escrever, avaliar, programar o código no microcontrolador e testar se as funções definidas operam como o esperado. Por ser

extramente simples e robusto há enorme empregabilidade e agilidade na criação e modificação dos códigos.

A partir da instalação dos arquivos do sistema ARDUINO, automaticamente o próprio microcontrolador ao ser conectado a um microcomputador realiza a enumeração de uma porta *USB* que é reconhecida pelo sistema operação como uma porta *SERIAL RS232*, Figura 11.

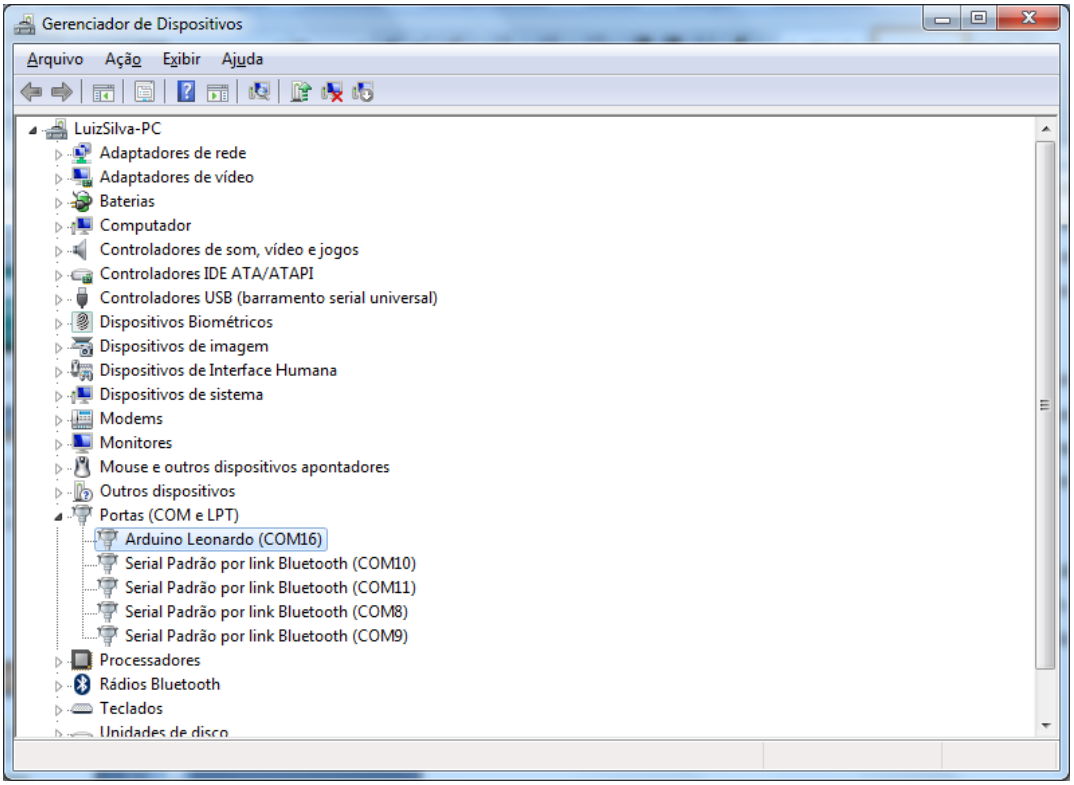

Figura 11 - Conexão do sistema ARDUINO ao computador

Fonte: Própria

A praticidade do sistema e existência de um código interno em sua memória pré-programado de fabrica, o qual possibilita a interação direta através de uma porta *USB* e programação através de *SKETCHs* traz outra vantagem da plataforma sob os demais, ou seja, não se faz necessário nenhum programador externo para escrever os códigos desenvolvidos.

A esse código interno programado em sua memória tem o nome de *bootloader*, sendo esse conjunto de instruções o primeiro a ser interpretado pelo microcontrolador quando o mesmo é energizado.

Esse conjunto de instruções não possui nenhuma correlação com o código escrito pelo usuário, mas sim com o funcionamento lógico inicial do sistema.

7. DESENVOLVIMENTO

7.1 VISÂO COMPLETA

Dentre os inúmeros sistemas de aquisição de sinais elétricos optamos por projetar um sistema que dependerá da interação constante entre a placa de aquisição e condicionamento de sinais e sistema de monitoramento – microcomputador.

Isso foi adotado para reduzir o processamento e complexidade do sistema e possibilitar uma maior interação entre o usuário e conjunto. Com isso temos a visão geral do bloco funcional da placa de aquisição e suas características básicas, Figura 12:

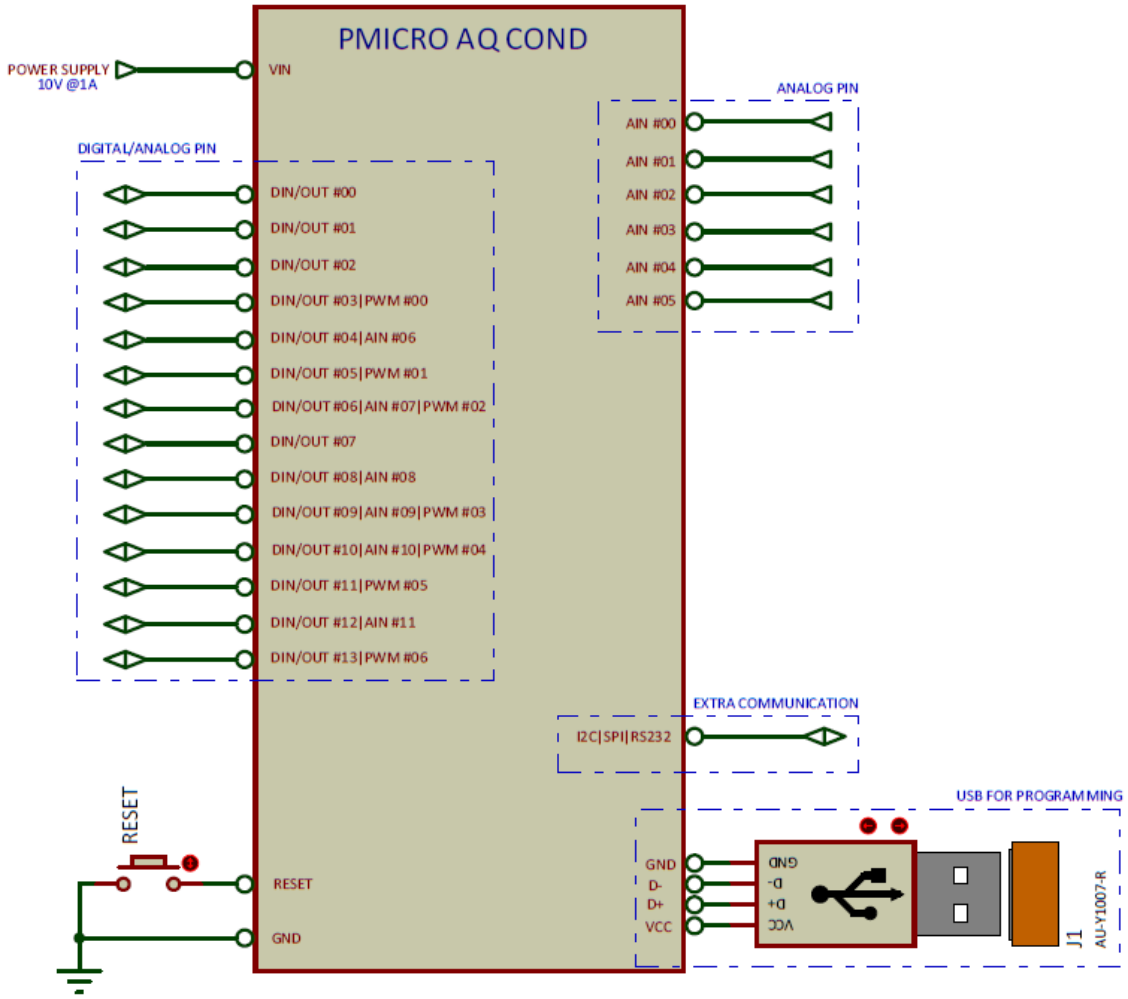

Figura 12 - Visão geral do sistema

Fonte: Própria

A particularidade de cada uma das entradas e saídas será exibido posteriormente. Desta a grande flexibilidade dos sinais, onde o usuário poderá optar por utilizar apenas entradas digitais, ou saídas, entradas analógicas ou ainda saídas do tipo *PWM*.

Em conjunto com esse sistema teremos blocos funcionais para o condicionamento e controle de sinais de potência ou com níveis de tensão diferente dos aplicados à alimentação do conjunto.

Esses blocos funcionais atuaram como elementos adicionais proporcionando ao usuário a expansão das funcionalidades do sistema e maior empregabilidade do conjunto. Além disso, por ser um sistema de código aberto o próprio usuário poderá elaborar seu bloco funcional adicional conforme sua necessidade.

A documentação dos circuitos e esquema elétrico eletrônico foi desenvolvida em ambiente gráfico *PROTEUS – iSIS DEMO* da *Labcenter Eletronics*, em uma versão destinada a estudantes e com recursos limitados.

7.2 ARDUINO - ATMEGA 32u4

O microcontrolador adotado para o sistema foi o ATMEGA 32u4 da ATMEL. A escolha deste deu-se pelas suas características e também já utilização no ARDUINO LEONARDO. Esse microcontrolador possui:

- Arquitetura *ARM RISC* de 08 bits e baixo consumo de corrente.
- USB 2.0 integrada.
- Compatibilidade com diagnostico *Boundary-scan*.
- Tensão de operação de 5.0 V.
- 20 pinos digitais que podem ser configurados como entradas e saídas.
- 07 canais do tipo *PWM*
- 12 pinos analógicos com resolução de 10 bits.
- 40 mA de corrente máxima por pino.
- 32 KB de memória *Flash*, sendo 4 KB utilizadas pelo *bootloader*.
- 25 KB de memória *SRAM*.
- 1 KB de memória e2prom interna.

Além destas características o conjunto ARDUINO LEONARDO, Figura 13, possui frequência de trabalho de 16 MHz, já integrado com conexão *USB* e sistema interno de regulagem de tensão.

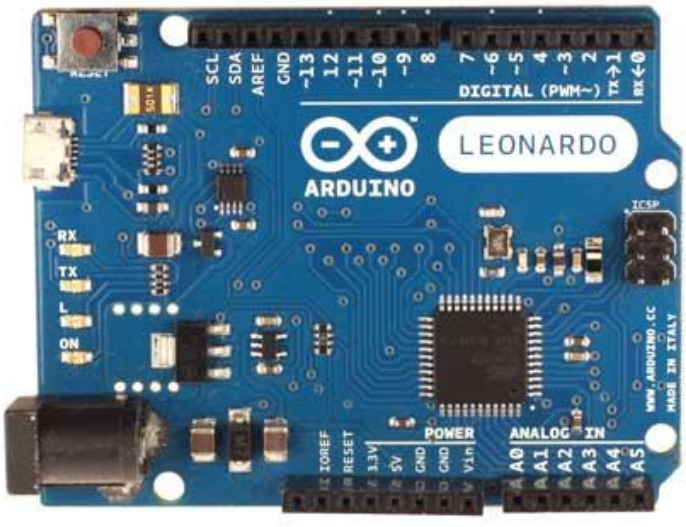

Figura 13 - ARDUINO LEONARDO

Fonte: http://arduino.cc/en

Todo o controle, gerenciamento e exibição dos dados ao usuário são realizados por este microcontrolador. Alterações deste nível não exigem modificações na montagem física, mas no código executado por este. Também possível modificar a montagem física para aumento da potência suportada pelo controlador sem impacto direto no código executado.

Como é possível configurar os pinos como I/O de uso geral esse microcontrolador possibilita ao projeto o acréscimo ou remoção de funcionalidades de forma rápida e confiável, dentro das limitações do espoco predefinido.

Para a execução do código o microcontrolador verificará as entradas e acionará das saídas nos pinos definidos para o projeto. A disposição e função de cada um dos pinos existente no conjunto são exibidas a seguir, Figura 14.

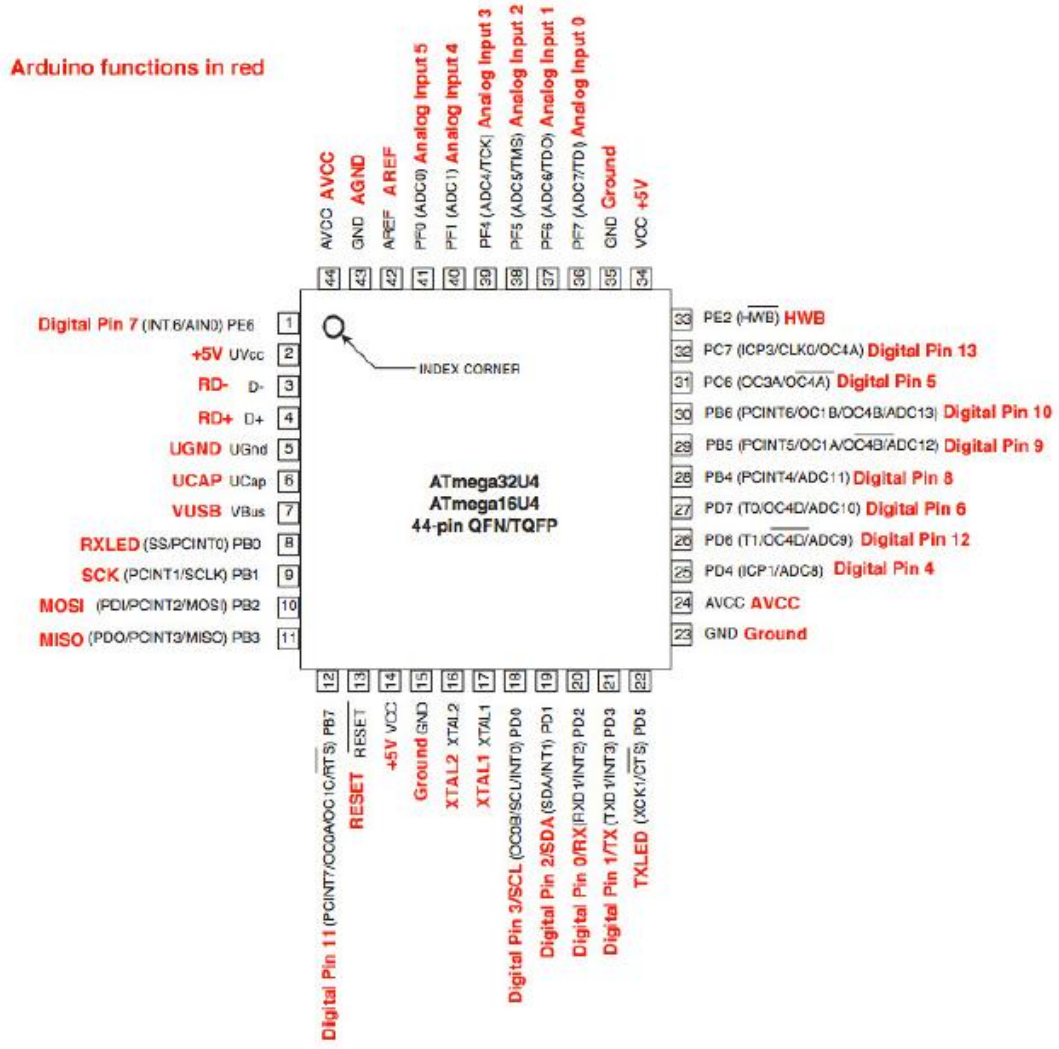

Figura 14 – Microcontrolador ATMEGA 32u4

Fonte: http://arduino.cc/en

Na Tabela 1 a correlação entre os pinos do microcontrolador, função e correlação com o código a ser executado:

Tabela 1 - Pinos e funções do microcontrolador

Pino	Port N	Função	Descrição
01	PE6(INT 6/AIN0)	*Digital pin 07*	DIN/OUT #07
12	PB7(#RST)	*Digital Pin 11*	DIN/OUT #11 ou PWM #05
18	PD0(SCL)	*Digital pin 03*	DIN/OUT #03 ou PWM #00
19	PD1(SDA)	*Digital pin 02*	DIN/OUT #02
20	PD2(INT2)	*Digital pin 00*	DIN/OUT #00
21	PD3(INT3)	*Digital pin 01*	DIN/OUT #01
25	PD4(ADC8)	*Digital pin 04*	DIN/OUT #04 ou AIN #06
26	PD6(ADC9)	*Digital pin 12*	DIN/OUT #12 ou AIN #11
27	PD7(ADC10)	*Digital pin 06*	DIN/OUT #06 ou AIN #07 ou PWM #02
28	PB4(ADC11)	*Digital pin 08*	DIN/OUT #08 ou AIN #08
29	PB5(ADC12)	*Digital pin 09*	DIN/OUT #09 ou AIN #09 ou PWM #03
30	PB6(ADC13)	*Digital pin 10*	DIN/OUT #10 ou AIN #10 ou PWM #04
31	PC6(OC3A)	*Digital pin 05*	DIN/OUT #05 ou PWM #01
32	PC7(CLK0)	*Digital pin 13*	DIN/OUT #13 ou PWM #06
36	PF7(TDI)	*Analog pin 00*	AIN #00
37	PF6(TDO)	*Analog pin 01*	AIN #01
38	PF5(TMS)	*Analog pin 02*	AIN #02
39	PF4(TCK)	*Analog pin 03*	AIN #03
40	PF1(ADC1)	*Analog pin 04*	AIN #04
41	PF0(ADC0)	*Analog pin 05*	AIN #05

Fonte: Própria

Conforme necessidade do usuário e projeto é possível controlar cada um dos pinos e defini-los como entrada ou saída, conseguindo assim maior flexibilidade do sistema. Existem pinos sem utilização no microcontrolador o que possibilita aumento nas funcionalidades do sistema se necessário.

Os pinos configurados como saída do microcontrolador atuaram como sistemas *open-collector* e caso sejam configurados como entradas há a possibilidade de incluir, através do código programado, resistores de *pull-up* internos.

A execução do código no microcontrolador depende da velocidade do *clock* adotado para o projeto, Figura 15. Neste escolheu–seu uma frequência de 16 MHz, resultando em um ciclo de máquina de:

$$Tempo\ de\ ciclo\ de\ clock = 1/\left(\frac{Clock}{4}\right) = \ 0.3 \mp 0.150\ us$$

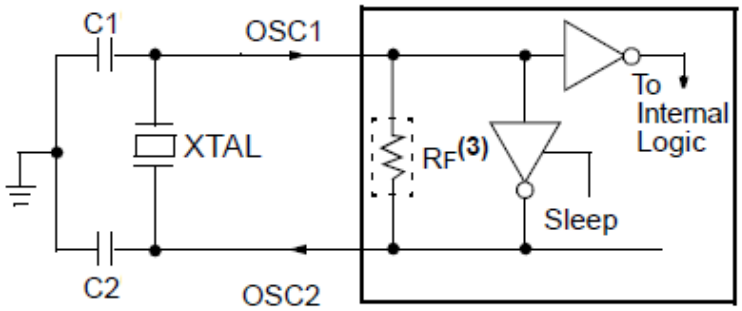

Figura 15 - Oscilador clock do sistema

Fonte: http://arduino.cc/en

A cada ciclo de máquina uma instrução do microcontrolador é executada, esse ciclo depende da instrução executada, ou seja, a mesma por ser executada utilizando um ou vários pulsos de clock. Como o código interno programado nestes apenas executará as funções definidas pelo usuário tanto o processamento quando o consumo de corrente será baixo, uma vez a maior parte do processamento será executada pelo microcomputador.

Uma das entradas do microcontrolador denominada *RESET* possui a função de interromper a execução do código e reinicializá-la quando acionada. Desta forma não é necessário desligar a alimentação caso o sistema apresente funcionamento inesperado ou necessário reinicializá-lo.

7.3 ARDUINO - *FIRMWARE*

Firmware é conjunto de instruções operacionais programadas diretamente relacionadas ao meio físico do equipamento. Esse está normalmente envolvido com operações básicas de baixo nível das quais sem o dispositivo ou equipamento não funcionaria.

Diferente do termo *software* que está relacionado ao alto nível do sistema, responsável também pelo funcionamento e interação com o usuário, o *firmware* é um código vital que realiza a interação entre o meio físico e software do sistema.

O firmware utilizado no sistema ARDUINO LEONARDO que proporcionará a interação entre este e o ambiente LabVIEW é denominado **LIFA**, LabVIEW *Interface* for ARDUINO. Desenvolvido pela *National Instrument* e de código aberto esse é capaz de configurar dinamicamente o sistema a partir da necessidade e programação do usuário.

A arquitetura do *firmware* LIFA é simples e possibilita que o LabVIEW realize a configuração e o controle das entradas e saídas. Basicamente existem três funções básicas e essenciais nesse que realizam todas as operações:

- ***syncLV*()**

Essa função estabelece a conexão entre o ARDUINO e LabVIEW. Essa função somente deverá ser chamada uma única vez durante a inicialização. Desta forma ao se estabelecer a conexão através da porta serial entre os dois sistemas ocorrem à sincronização e verificação da comunicação.

- ***checkForCommand()***

Essa função é executada no *loop* principal da sequência do ARDUINO. Essa função verifica se há dados disponíveis no *buffer* da porta serial. Também é avaliado se os dados recebidos são válidos e estão no padrão esperado.

- ***processCommand()***

Essa função interpreta o comando recebido, verifica a integridade de cada um dos dados e processa-os. Também essa função interpreta se há ou não necessidade de retornar informações do sistema ao LabVIEW.

A estrutura dos dados enviados a partir do sistema de controle é simples e facilmente modificada, permitindo assim que o usuário altere-o conforme necessidade.

Basicamente todo comando enviado e recebido entre os sistemas possui 15 *bytes*, Tabela 2, porém esse tamanho pode ser adaptado para melhor o desempenho, no entanto tanto o *firmware* quanto sistema de controle tem que possuir a mesma estrutura e quantidade de dados transmitidos. A estrutura de dados é subdividida em:

Tabela 2 - Estrutura do comando LIFA

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14
Hea	Cmd	D00	D01	D02	D03	D04	D05	D06	D07	D08	D09	D10	D11	Chk

Fonte: Própria

Cada byte do comando possui uma função para o sistema, assim:

- **Posição 00**

Denominado como *header*, ou cabeçalho, do comando. Esse *byte* é responsável por conter um identificador entre os dois sistemas, desta forma ao receber uma mensagem proveniente do LabVIEW o ARDUINO compara se *byte* com o valor predefinido antes de processar a informação, evitando erros.

- **Posição 01**

Essa posição encontra-se o comando que será processado pelo sistema. Devido às inúmeras funções existem valores hexadecimais foram adotados para identificar cada uma destas, Tabela 3. O valor desta posição está relacionado aos parâmetros contidos nos demais campos.

Tabela 3 - Funções LIFA

Função	Nome	Descrição
0x 00	*Sync Packet*	Sincroniza o ARDUINO com LabVIEW
0x 01	*Flush Serial Buffer*	Descarta os dados da porta serial
0x 02	*Set Pin As Input Or Output*	Define a função de cada pino
0x 03	*Write Digital Pin*	Controla uma saída digital
0x 04	*Write Digital Port*	Controla o porto digital
0x 05	*Tone*	Ativa a função de áudio do sistema
0x 06	*Read Digital Pin*	Lê uma entrada digital

0x 07	*Read Digital Port*	Lê o porto digital
0x 08	*Read Analog Pin*	Lê uma entrada analógica
0x 09	*Read Analog Port*	Lê o porto analógico
0x 0A	*PWM Write Pin*	Ativa uma saída PWM
0x 0B	*PWM Write 3 Pins*	Ativa três saídas PWM
0x 0C	*Configure 7 Seg Display*	Configura o sistema para controlar um display 7 segmentos
0x 0D	*Write To 7 Seg Display*	Controla o display 7 segmentos
0x 0E	*Initialize I2C*	Inicializa a comunicação i2c
0x 0F	*I2C Write*	Envia dado ao barramento i2c
0x 10	*I2C Read*	Lê dados do barramento i2c
0x 11	*Initialize SPI*	Inicializa a comunicação SPI
0x 12	*SPI Write*	Envia dado ao barramento SPI
0x 13	*SPI Read*	Lê dados do barramento SPI
0x 1E	*LCD Init*	Inicializa o sistema LCD
0x 1F	*LCD Set Size*	Configura o tipo e tamanho do LCD
0x 23	*LCD Print*	Envia mensagens ao LCD
0x 2A	*Continuous Aquisition Mode*	Ativa o modo de aquisição de sinais continuo

Fonte: Própria

- **Posição 02 a 13**

Contem os dados necessários para execução do comando definido e também são as posições que conterão o valor de retorno do sistema.

- **Posição 14**

Nesse campo temos a soma de todos os dados contidos no comando. Usualmente definimos essa soma como *CHECKSUM*, ou comparação por soma. Ambos os sistemas avaliam se a soma das posições de 0 a 13 coincidem com o valor contido no campo 14 antes de processar a informação. Isso é empregado para garantir que o pacote de dados enviado e recebido foi realmente transmitido de maneira correta e nenhum dado foi perdido.

O *firmware* LIFA desenvolvido pela *National Instruments* é genérico e pode ser aplicado em inúmeros microcontroladores da família ARDUINO.

Para a utilização deste no projeto foi necessário modificar desde as funções básicas de sincronização até as funções de temporização. Isso foi necessário para adaptar o código básico as necessidades do projeto.

Devido à simplicidade dos códigos e grande flexibilidade do sistema o usuário pode facilmente modificar esse *firmware* para melhor atender sua necessidade. O que é grande vantagem em sistemas de aquisição de dados.

7.4 LABVIEW - CÓDIGOS VIRTUAIS

Os códigos escritos na linguagem gráfica em LabVIEW são chamados de instrumentos virtuais ou, simplesmente Vis. Esses códigos interagem com o *firmware* **LIFA**, existente no ARDUINO, conforme a necessidade de programação definida pelo usuário.

Desenvolvido pela *National Instruments* o pacote de código aberto contendo as funções de interação direta com o ARDUINO fornecem flexibilidade e agilidade ao programador durante o desenvolvimento do sistema de controle. As funções existentes neste código podem ser divididas em 03 grupos básicos, são estes:

- **Grupo ARDUINO**

Nesse grupo, Figura 16, se encontra as funções básicas de interação entre os sistemas, tais como funções de inicialização, responsável por configurar a velocidade e características de conexão; de monitoramento; de definições da funcionalidade e portas e termino da comunicação.

Figura 16 - Código LabVIEW grupo ARDUINO

Fonte: LabVIEW

Nesse grupo de funções estão alocados os códigos fundamentais para a elaboração de qualquer sistema. Além disso, é possível acessar e utilizar as funções existentes em cada um dos subgrupos, ampliando assim a possibilidade e código do usuário.

Desta forma é possível desenvolver um sistema automono, flexível e robusto. Todas as funções existentes no subgrupo já estão pré-configuradas e também coexistem no LIFA.

Assim por exemplo, pode-se ler uma entrada analógica isolada ou ler vários pinos ao mesmo tempo, basta selecionar a função desejada e inseri-la no código, Figura 17.

Figura 17 - Código LabVIEW subgrupos

Fonte: LabVIEW

- **Grupo *UTILITY***

Nesse grupo, Figura 18, se encontra as funções que possibilitam controle de sincronismo; leitura e escrita de comandos livres entre os sistemas e temporização.

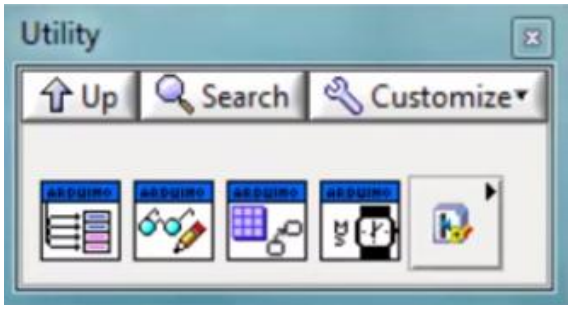

Figura 18 - Código LabVIEW grupo *UTILITY*

Fonte: LabVIEW

- **Grupo *EXAMPLES***

Nesse grupo, Figura 19, se encontra exemplos de aplicações com sensores; controle de iluminação; de sinais analógicos e digitais; som e também interação com outros sistemas através de comunicação *LAN, USB, SERIAL, SINGLE WIRE, SPI* ou *I2C*.

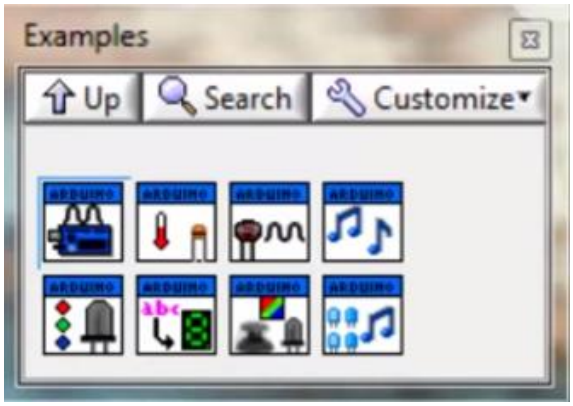

Figura 19 - Código LabVIEW grupo *EXAMPLES*

Fonte: LabVIEW

Ao selecionar um determinado exemplo é possível visualizar em seu código o descritivo de ajuda e uma breve explicação sobre as conexões físicas necessárias para o correto funcionamento, da função selecionada.

Caso se deseje controlar um display de sete segmento o código exibe, Figura 20, o descrito das conexões e comentários sobre o funcionamento.

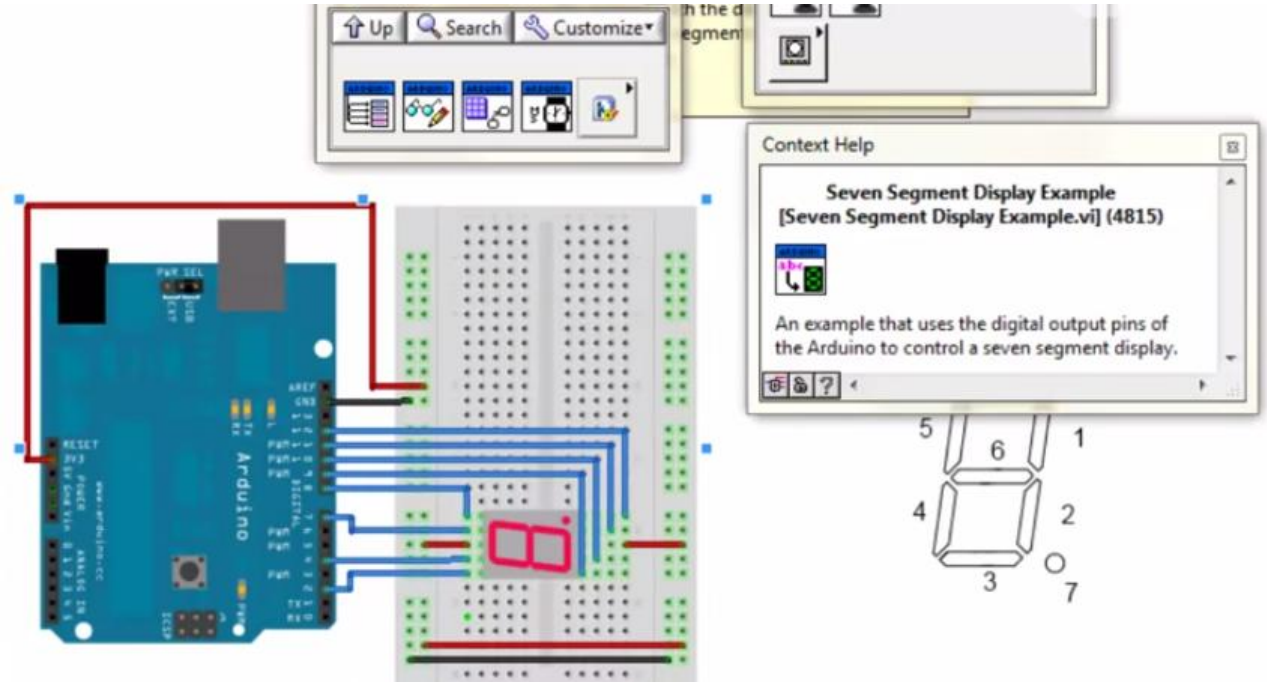

Figura 20- Código LabVIEW exemplo de aplicação

Fonte: LabVIEW

O usuário pode facilmente modificar os códigos de cada uma das funções para melhor atender sua necessidade. O que é grande vantagem em sistemas de aquisição de dados e reduz o tempo de desenvolvimento.

7.5 AQUISIÇÃO DE SINAIS

Os canais de medição de tensão operam em função da tensão de alimentação do microcontrolador e dependem das configurações estabelecidas para cada um deles. Além disso, estes canais possuem proteções internas contra sobretensão, dentro de um determinado limite de 30% superior a tensão de alimentação, e também filtros para a remoção de ruídos.

Para o projeto como todos os canais possuem a mesma faixa de tensão e a referência se dará através dos níveis de alimentação do microcontrolador, portanto a leitura dos canais de tensão, sejam digitais ou analógicos, situam-se entre 0.0 e 5.0V.

- **Aquisição de sinais digitais**

As entradas digitais do sistema foram projetadas para adquiri pulsos provenientes de botões, chaves ou interruptores, Figura 21.

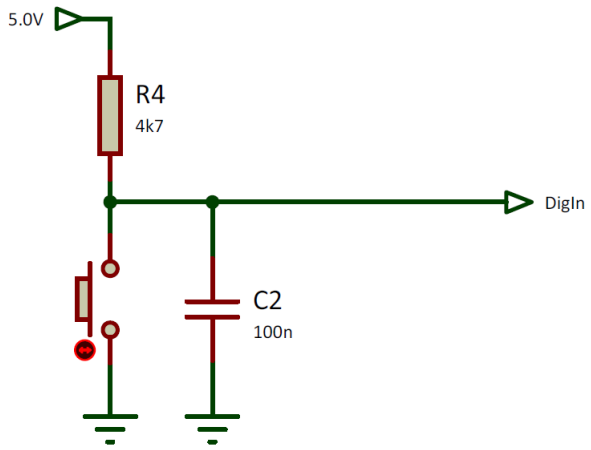

Figura 21 - Entradas digitais

Fonte: Própria

Sinal digital como trem de pulso ou com frequência especifica também podem ser adquiridos, no entanto alterações no código e circuitos são necessários. As entradas digitais atuaram com lógica invertida, pois quando desligadas terão o nível lógico *High*, devido ao resistor de *pull-up*, quando ligadas o nível lógico *Low* será entregue ao microcontrolador.

- **Aquisição de sinais analógicos**

O microcontrolador possui um conversor analógico de 10 bits capaz de adquirir sinais totalizando uma faixa de 1024 unidades. Enquanto a tensão de entrada pode variar de 0.0 a 5.0V as unidades correspondentes a cada valor de tensão de entrada por *bit* podem variar de 0 a 1023. Assim:

$$\text{Resolução AD (mV)} = \frac{(5{,}0 - 0{,}0)}{1023} * 1000 = 4{,}88$$

Com a resolução do canal AD definida pode–se medir quaisquer níveis de tensão nas entradas analógicas, situadas nos limites máximo e mínimo, e converter com base no sistema compreendido pelo microcontrolador. Para medir tensões superiores ao nível máximo suportado é necessário condicionar os sinais.

Como os níveis são superiores a 5.0V é necessário medir uma amostra destes sinais dentro da faixa de tensão do microcontrolador e referenciada a 0.0V.

- **Aquisição de sinais analógicos contínuos**

Esses são sinais que possuem como referência o sinal de 0.0v, portanto não possuem parte negativa. Desta forma para adequar os níveis de tensão de entrada do controlador e ampliar o limite de tensão permitido divisores de tensão resistivos foram utilizados.

Divisores resistivos são circuitos série que estabelecem uma relação direta entre a tensão de entrada e a queda de tensão sobre determinado elemento. A relação estabelecida neste circuito, adotando como referência a Figura 22, é:

$$\text{Relação(V)} = \text{Vin} * \left(1 + \frac{\text{R1}}{\text{R2}}\right) = \frac{\text{Vin}}{5{,}85}$$

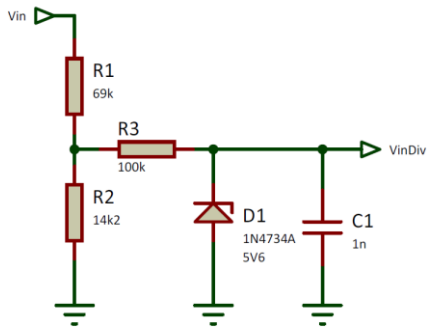

Figura 22 - Divisor de tensão resistivo

Fonte: Própria

É aconselhável à utilização de resistores de precisão para esse circuito, com isso reduz–se possíveis variações de tensão em função das resistências. Adotando uma precisão 5% tem–se uma variação:

$$\text{Relação Max}(V) = \text{Vin} * \left(1 + \frac{(1{,}05 * \text{R1})}{(0{,}95 * \text{R2})}\right) = \text{Vin}/6{,}30$$

$$\text{Relação Min}(V) = \text{Vin} * \left(1 + \frac{(0{,}95 * \text{R1})}{(1{,}05 * \text{R2})}\right) = \text{Vin}/5{,}10$$

Idealmente a maior tensão que o sistema poderá suportar é 28.0V. Com o valor máximo teórico e real encontramos a variações do divisor:

$$\text{Variação relativa}\ (\%) = \frac{1}{2} * \left(\frac{6{,}30}{5{,}85} - 1\right) * \left(\frac{5{,}10}{5{,}85}\right) = 3{,}35$$

O projeto do divisor admite uma variação relativa de 3,35% sobre o valor máximo de tensão do circuito. Também é possível alterar os valores da associação apara se ampliar o limite de tensão de entrada permitido, porém o novo fator encontrado deverá ser modificado no código programado.

- **Aquisição de corrente contínua e alternada**

Para a medição de sinais de corrente adotou-se um conversor corrente tensão industrial, ACS712ELCTR-05B-T, capaz de realizar a leitura de sinais de corrente continua ou alternada através de efeito hall.

Esse componente, Figura 23, possui capacidade de medir correntes entre -5.0 e +5.0A.

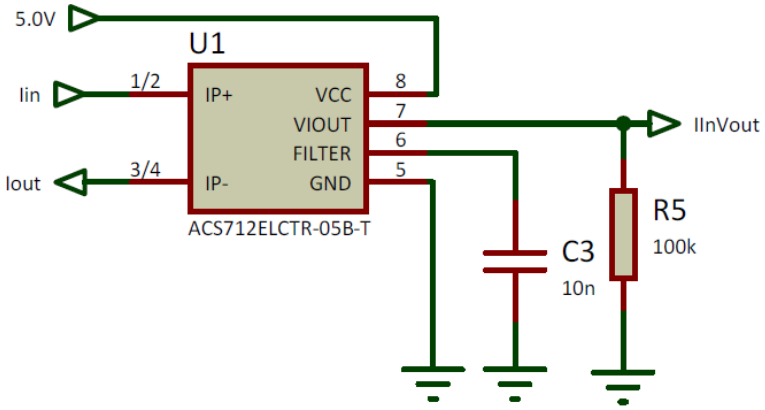

Figura 23 - Conversor corrente tensão

Fonte: Própria

Os sinais de tensão de saída entregue pode-ser então lido diretamente pelo canal analógico, uma vez que o nível máximo de tensão já está pré-fixado a alimentação do microcontrolador.

A taxa de correlação entre corrente e tensão está relaciona aos limites desta e sensibilidade do componente, Figura 24:

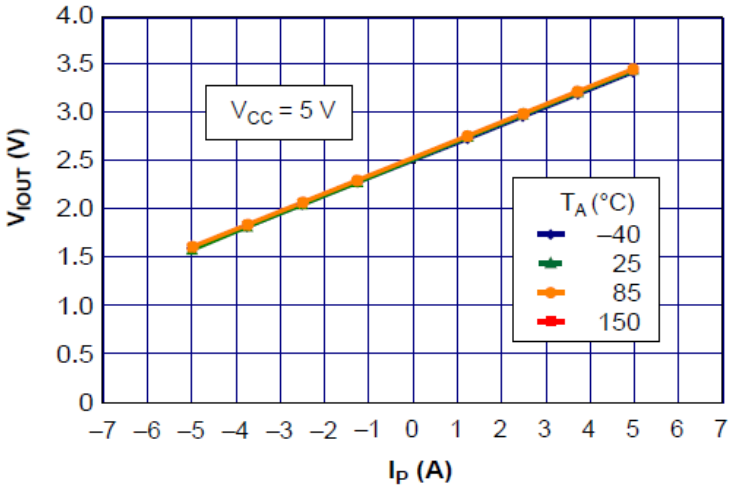

Figura 24 - Correlação corrente tensão ACS712ELCTR-05B-T

Fonte: www.allegromicro.com

Com isso temos que a função de tensão de saída do componente será:

$$\text{Tensão de saída (V)} = (((\text{I} + 5) * 2)/10) + 1{,}5$$

Onde mesmo com o valor mínimo negativo de corrente teremos um valor de tensão positivo com valor aceitável para o canal analógico. A relação entre a tensão lida pelo canal analógico e corrente do circuito será:

$$\text{Corrente de entrada (A)} = (((\text{V} - 1{,}5) * 10)/2) - 5$$

Com essas relações é possível determinais quaisquer valores de tensão e corrente dentro dos limites permitidos pelo conversor.

7.6 CONDICIONAMENTO DE SINAIS

O condicionamento consiste em atuar nas saídas com níveis de tensão e corrente diferentes dos sinais de controle disponíveis no microcontrolador. Existem inúmeras opções de controle, como transistores de potência, transistores de efeito de campo, transistores para comutação de cargas em sinais alternados e ainda relés eletromecânicos.

Cada um destes tipos de comutadores deve ser empregado conforme necessidade e características do projeto, respeitando os níveis de tensão, corrente e proteção para cada um dos casos.

- **Condicionamento de sinais analógicos contínuos**

As opções existentes para o chaveamento dos elementos de potência contínua, utilizados para o controle e comutação da carga restringiram–se a utilização de *IGBT* ou *MOSFET*. Essa escolha considerou os pontos favoráveis e desaforáveis de cada um destes elementos:

- **Potência de acionamento:** Os transistores bipolares de potência caracterizam-se pela relativamente alta potência necessária para o acionamento dos mesmos, uma vez que necessitam de corrente constante na base. Os *MOSFETs* e *IGBTs*, por sua vez, são acionados através do nível de tensão existente nos terminais de disparo, denominados de *Gates*. A alta impedância de entrada e saída destes representa substancial redução da potência de excitação e robustez a ruídos. Por essa razão os transistores bipolares foram descartados para o projeto.

- **Frequência de chaveamento:** Os *MOSFETs* têm capacidade para chavear potência em valores de frequência muito superiores aos suportados pelos *IGBTs*. Como o chaveamento atuará em ciclos de *ON/OFF* a capacidade de comutar ou não em alta frequência não é nenhum ponto desfavorável para o projeto.

- **Perdas durante o chaveamento:** Perdas nos dispositivos de chaveamento ocorrem durante o período de transição de estados - corte e saturação. Neste momento, a potência dissipada é considerável e, por esta razão, quanto mais rápido for à transição, menor será a dissipação e perda total.

Os *IGBTs* possuem transições mais lentas e dissipam mais potência durante o seu chaveamento, enquanto *MOSFETs* são mais rápidos e dissipam pouca potência no chaveamento.

Devido a maior velocidade e pouca dissipação durante o chaveamento o *MOSFET* foi adotado como elemento comutador de sinais em corrente contínua.

Como o controle deste é realizado através do nível de tensão no terminal de disparo fornecido pelo microcontrolador, seja um sinal estático proveniente de uma saída digital ou variável do tipo *PWM*, optou–se por utilizar um *MOSFET* que possuísse essa compatibilidade.

O *MOSFET MiniPROFET BSP 452* fabricado pela *Siemens/Infinium* é um *FET* de potência tipo canal N e controle digital *TTL/CMOS* com capacidade de comutação de 4A à 12V foi adotado para o projeto. Esse componente é comumente utilizado em sistemas automotivos e possui as seguintes características:

- Comutação *High-side*
- Proteção contra curto-circuito
- Proteção contra tensão reversa na saída
- Proteção contra sobretensão na entrada
- Proteção contra sobrecorrente
- Proteção contra descarga eletrostática (*ESD*)
- Compensação de temperatura
- Comutação de cargas indutivas

A seguir, Figura 25, o digrama funcional deste componente destacando os principais blocos e funções.

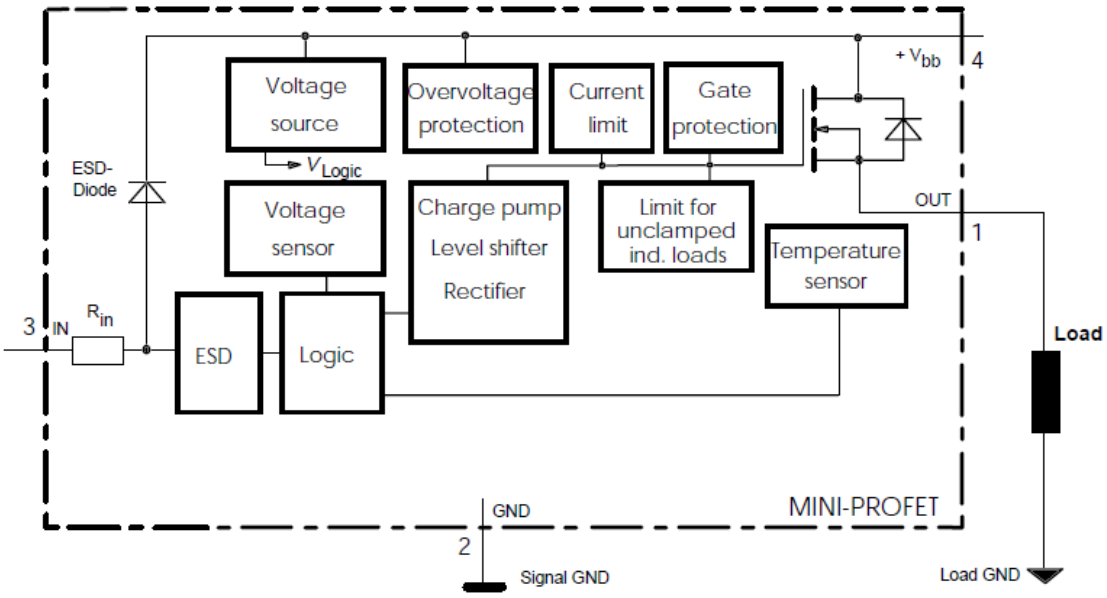

Figura 25 – Digrama MiniPROFET BSP452

Fonte: Infinium/PowerMosfet

Em conjunto com o *MOSFET* foi utilizado um filtro de primeira ordem, constituído por um capacitor e um resistor, para melhorar a estabilidade e resposta do acionamento realizado, Figura 26:

.

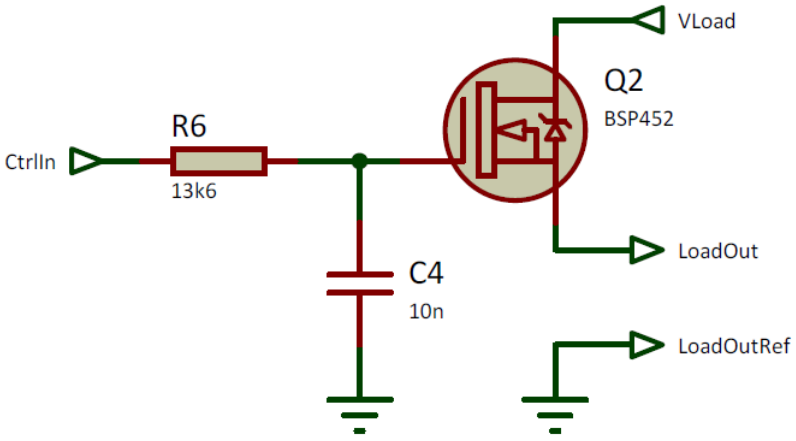

Figura 26 - Condicionamento de sinal analógico

Fonte: Própria

Desta forma seja um sinal estático ou variável com esse circuito temos a conversão do sinal digital em um nível de tensão analógica entregue ao terminal de disparo do *MOSFET*. Relacionando os sinais de entrada e saída do sistema tem-se a função de transferência do mesmo.

$$G(s) = \frac{Vo(s)}{Vi(s)} = \frac{Vc(s)}{Vr(s) + Vc(s)} = \frac{\frac{1}{sC}}{R + \frac{1}{sC}} = \frac{1}{sRC + 1}$$

A resposta característica deste sistema é visualizada na Figura 27.

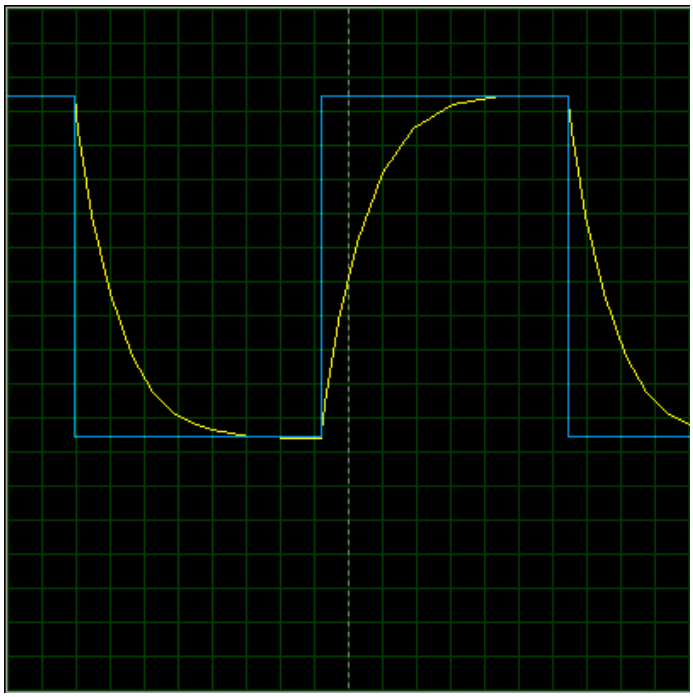

Figura 27 - Resposta do sistema

Fonte: Própria

É possível visualizar, Figura 27, o degrau unitário com amplitude de 5,0 V fornecido pela saída do microcontrolador e em azul e o sinal de saída, reposta ao degrau ao transiente de tensão.

7.7 DESENVOLVIMENTO DO PROTÓTIPO

Com as definições das funções é possível integrar todo o conjunto e desenvolver um protótipo para a de aquisição e condicionamento de sinais elétricos. Com isso pretende-se utilizar o conjunto para o controle de um motor de corrente contínua, podendo analisar seu comportamento em diversas situações de funcionamento.

O motor será controlado pelo usuário tanto através do acionamento físico e eletrônico realizado pelo programa. A Figura 28 exibe as conexões de sistema de aquisição, entradas de controle e saída.

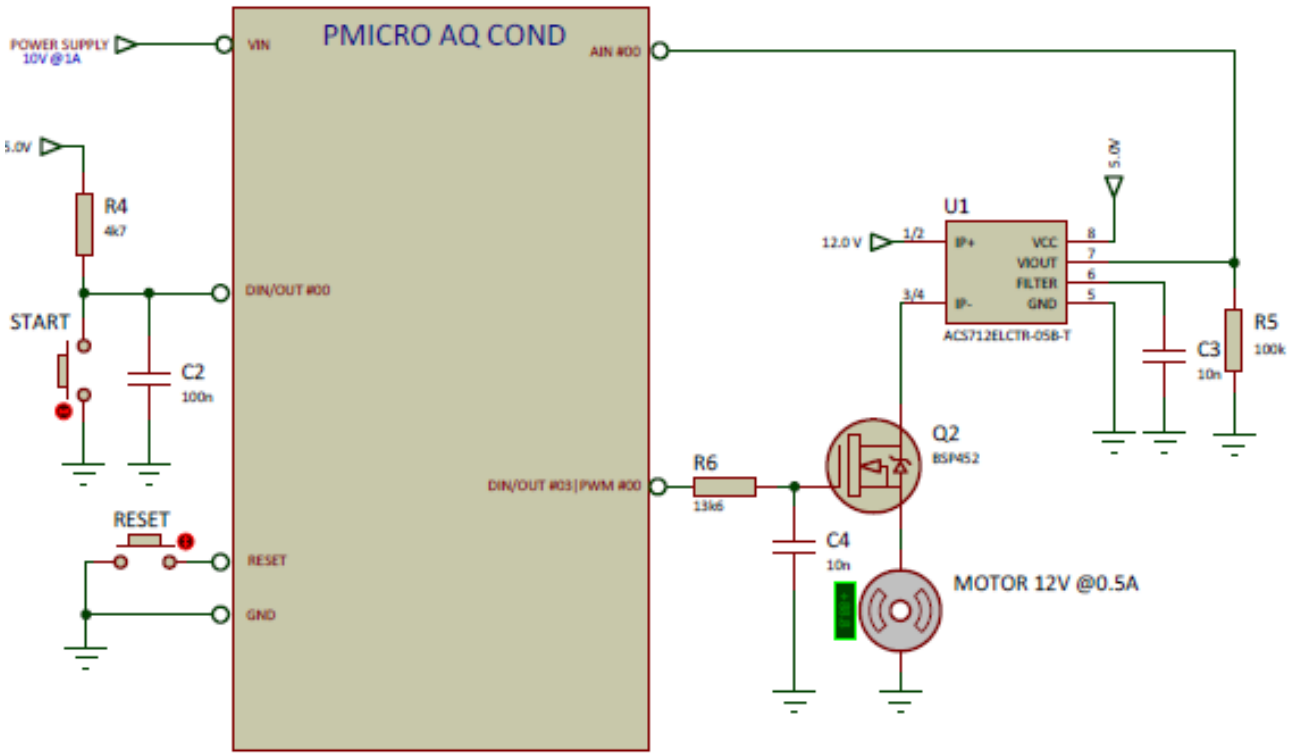

Figura 28 - Esquema elétrico protótipo

Fonte: Própria

Os materiais e custos para confecção protótipo estão relacionados na Tabela 4:

Tabela 4 - Materiais e custos do protótipo

Item	Qtde	Descrição	Custo unitário [R$]
1	1	ARDUINO LEONARDO	48,00
2	1	COMPONENTES [RESISTORES E CAPACITORES]	10,00
3	1	MOTOR DC 12 V	36,00

4	1	SENSOR DE CORRENTE ACS712ELCTR-05B-T	14,00
5	1	DRIVER MOTOR	12,00
			120,00

Fonte: Própria

O custo total de R$ 120,00 considera o sistema de aquisição e condicionamento de sinais e aplicação de controle do motor de corrente contínua. Se considerado apenas o sistema básico o custo é de R$ 58,00.

Por se tratar de uma plataforma de código aberto não há custos relacionados a programação, uma vez que o usuário é responsável por desenvolvê-la e o fluxo lógico gráfico pode ser alterado pelo usuário conforme a necessidade do projeto.

De maneira geral o sistema seguirá o fluxo lógico básico descrito na Figura 29:

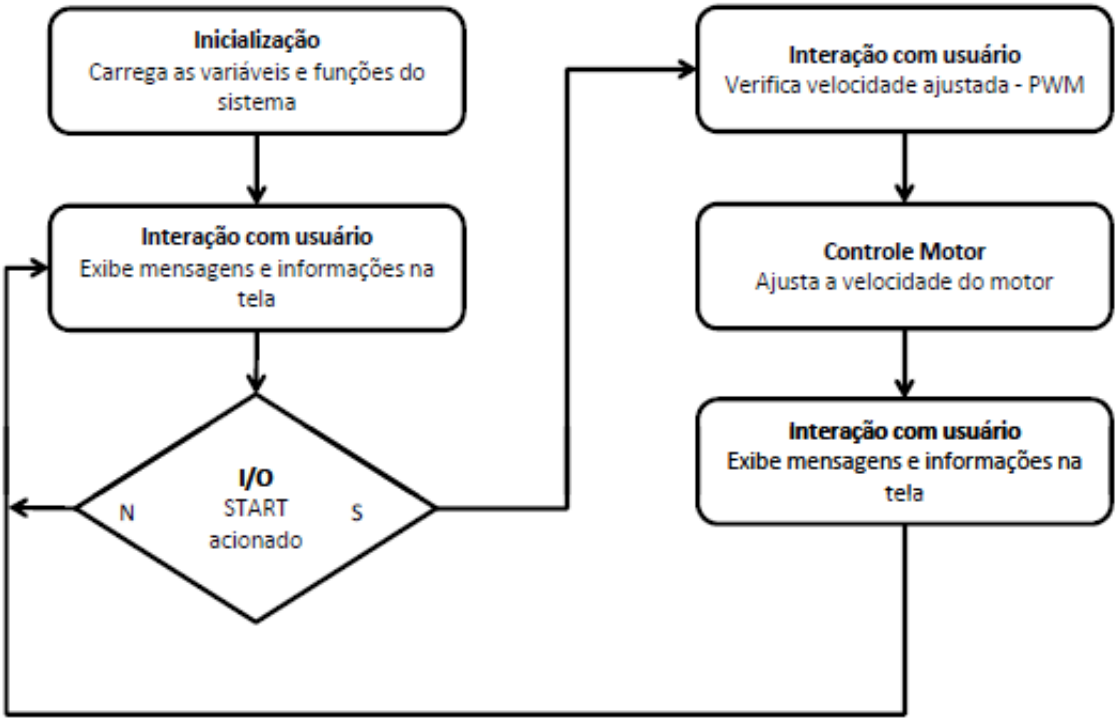

Figura 29- Fluxograma básico do protótipo

Fonte: Própria

Tem-se extrema flexibilidade de todo o sistema, além de ampliar sua empregabilidade. Variações do fluxo dependerão da necessidade do usuário. Com a sequência definida o código foi transcrito em linguagem gráfica em LabVIEW, Figura 30.

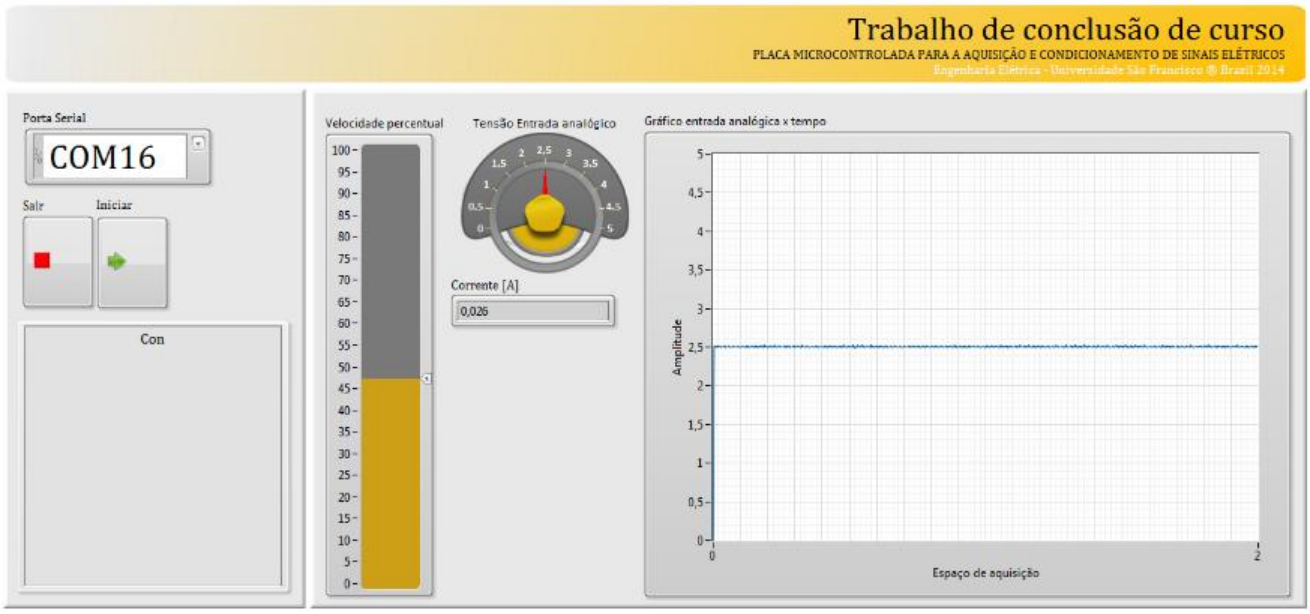

Figura 30 - *Interface* básica do usuário

Fonte: Própria

A *interface* gráfica possibilita ao usuário criar e interagir diretamente com o sistema. E possível alterar cores, tamanhos e posicionamento dos componentes que compõe o conjunto.

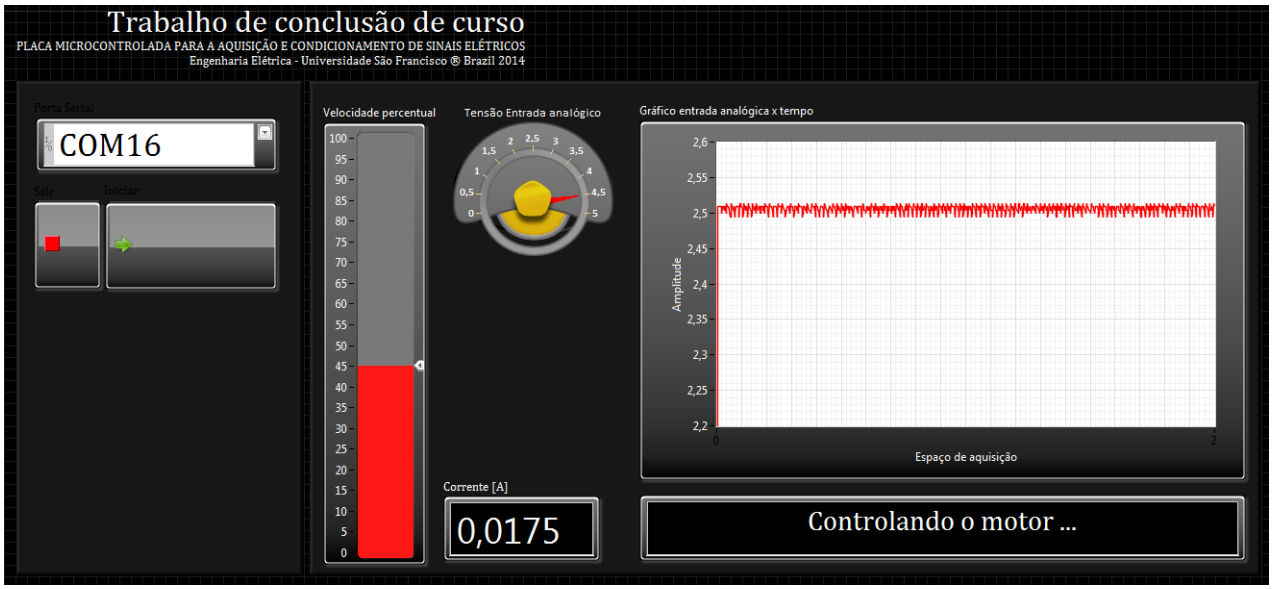

Figura 31 - *Interface* básica do usuário modificada

Fonte: Própria

Assim como na sequência definida no fluxograma primeiramente o sistema é inicializado configurando-se a porta de comunicação, nesse caso a porta serial para a velocidade de 115200 *bits* por segundo, e demais variáveis do sistema, Figura 32.

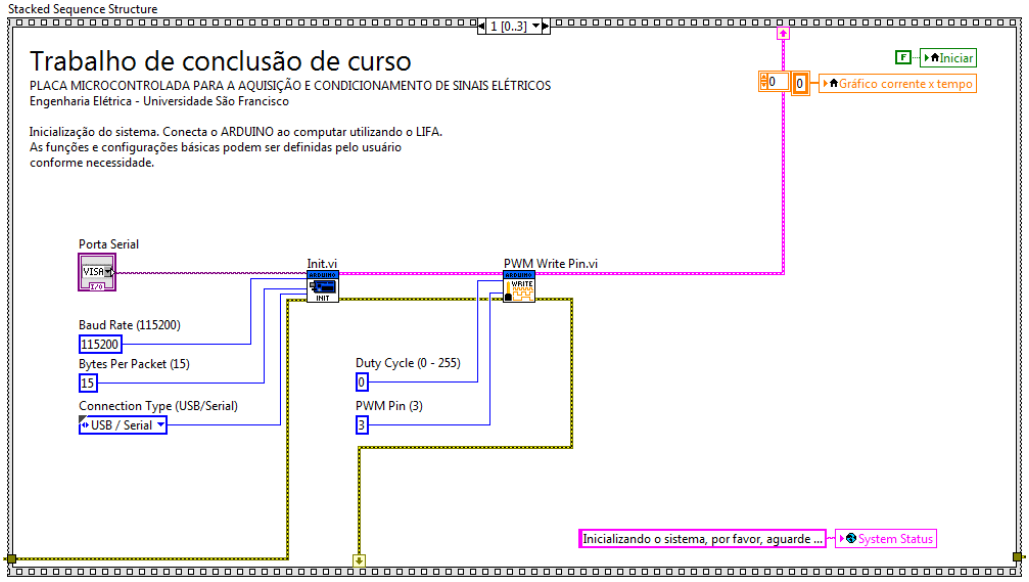

Figura 32 - Sequência de inicialização

Fonte: Própria

Como o sistema controlará a velocidade do motor através da variação da relação entre os níveis alto e baixo do sinal de uma saída digital, ou seja, através do *PWM*, é preciso configurar qual pino do sistema será responsável por fornecer tal sinal, Figura 33.

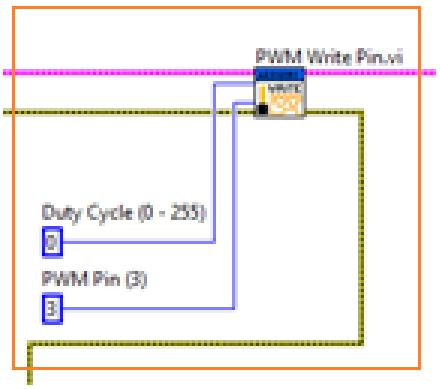

Figura 33 - Configuração do terminal de *PWM*

Fonte: Própria

Com o sistema configurado e inicializado aguarda-se o usuário definir o valor de velocidade relativa do motor e iniciar o sistema, através do acionamento do botão de iniciar na *interface* de usuário, Figura 34.

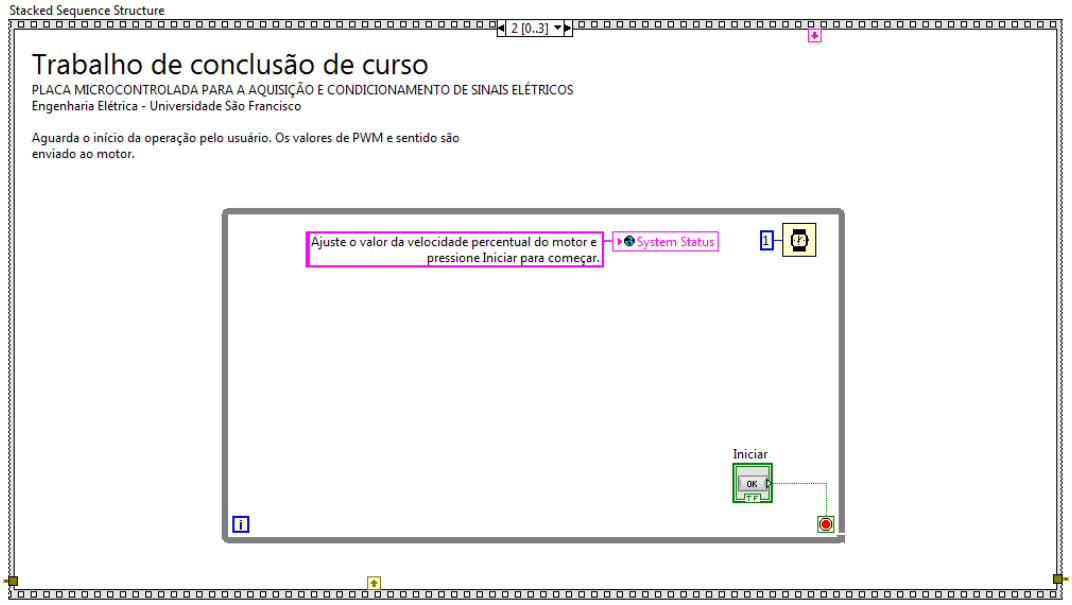

Figura 34 - Aguardando início do usuário

Fonte: Própria

Prosseguindo com a sequência o sistema realiza, nessa ordem, o ajuste da velocidade relativa percentual do motor, nesse caso o valor de *PWM*, e medição de tensão da entrada analógica relacionada à corrente consumida pelo motor, Figura 35.

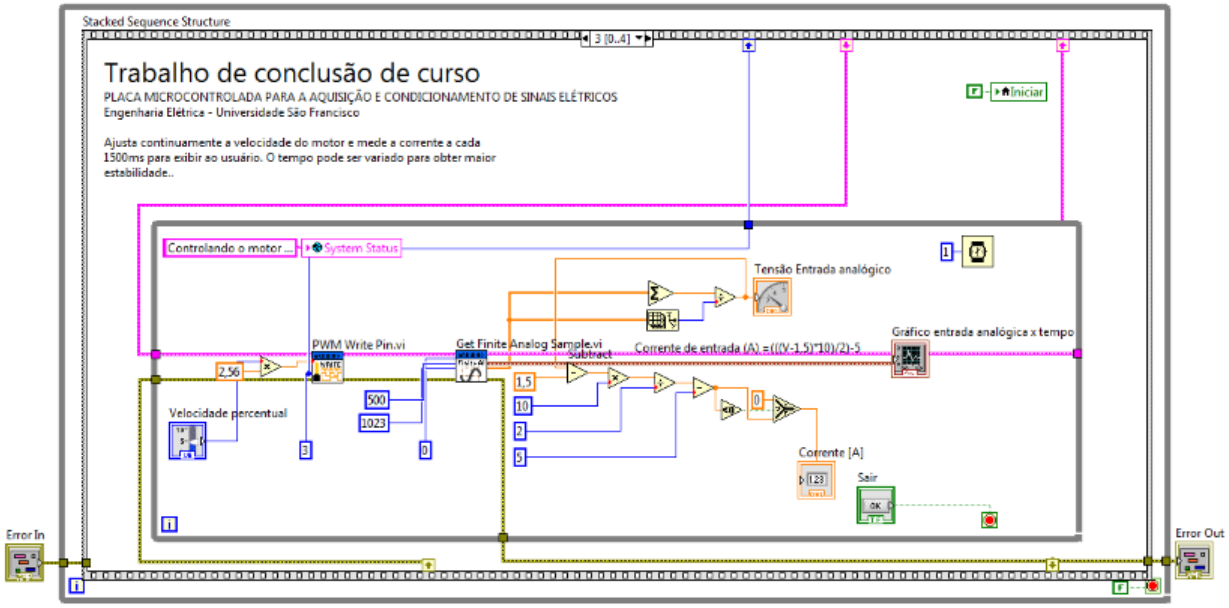

Figura 35 - Sequência de controle e aquisição de dados

Fonte: Própria

A aquisição de tensão analógica ocorre a uma taxa de 500 Hz, o que significa que cada 2 milissegundos aquisição é realizada, compondo um conjunto amostral de 1023 pontos, conforme Figura 36.

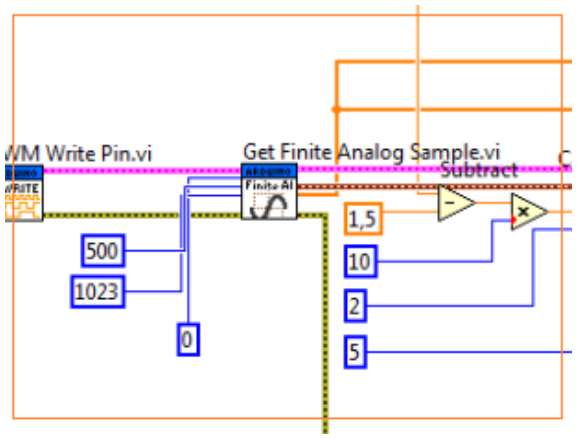

Figura 36 - Aquisição de sinal analógico contínuo

Fonte: Própria

Com o valor de tensão analógica o sistema aplica o fator de conversão tensão-corrente para exibir o valor real de corrente ao usuário e exibir graficamente a tensão analógica ao longo do tempo, Figura 37.

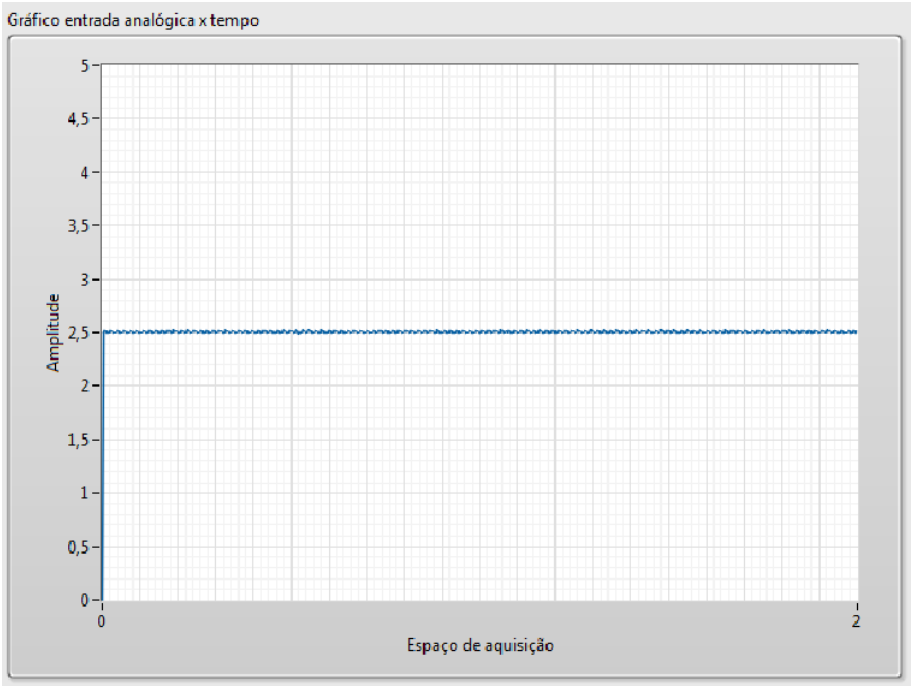

Figura 37 - Variação da tensão analógica de entrada

Fonte: Própria

O usuário pode alterar a sequência e particularidades do sistema conforme necessidade do projeto. Essa característica amplia a flexibilidade no desenvolvimento de sistemas de aquisição e condicionamento de sinais seja esse aplicado a um protótipo ou sistema final.

7.8 TESTE E VALIDAÇÂO DO PROTÒTIPO

Para a validação do sistema o mesmo foi submetido à aplicação de controle de velocidade de um motor de corrente continua com tensão de alimentação de 12 V, resistência interna de 250 ohms, sistema de redução e sem carga acoplada, Figura 38.

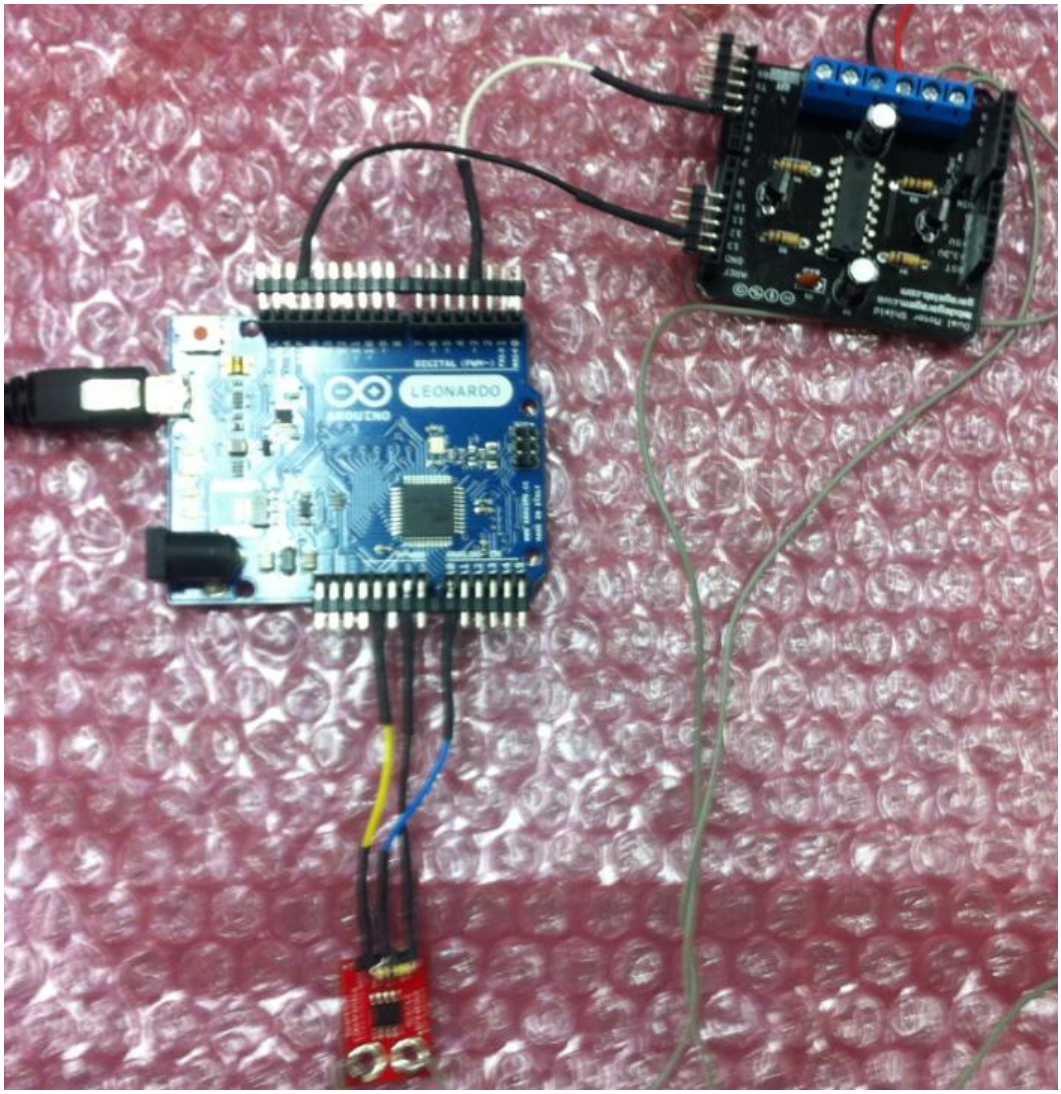

Figura 38 - Protótipo do sistema

Fonte: Própria

A sequência de teste adotada para a validação do protótipo foi:

- **Verificação da sequência lógica**

Para a verificação da sequência lógica o sistema foi conectado ao microcomputador e o código executado. Nessa verificação foi possível observar a sequência lógica definida no fluxograma, Figura 29, em plena operação. Nenhum erro ou anomalia foi observado.

O tempo de ciclo do sistema, ou seja, o número de repetições do fluxo ocorrem a cada 100 milissegundos, considerando que as aquisições da entrada analógica são realizadas a cada 2 milissegundos temos o total de 50 medições do sinal de entrada o que possibilita a análise de pequenas variações em tempo real.

Esse tempo de ciclo pode ser alterado conforme necessidade do sistema, atentando-se apenas a qualidade e recursos de processamento exigidos.

Caso o tempo de ciclo seja maior que o suportado pelo sistema ocorrerá perda de dados e consequentemente na qualidade do processamento.

- **Medição dos sinais em operação**

Com o sistema em operação é possível verificar os sinais a serem monitorados e controlados, avaliando e comparando assim sua funcionalidade, precisão e desempenho do conjunto. Essa análise foi realizada ao sinal de *PWM* e entrada analógica, *AIN*.

- **PWM**

A frequência do sinal de *PWM* é fixa e depende das características do microcontrolador. No sistema utilizado a frequência do sinal *PWM* do ARDUINO LEONARDO é fixa em 980 ± 5 Hz, operando com resolução de 8 bits.

Sendo a frequência fixa o controle da tensão média ocorre pela diferença entre os períodos em que o sinal encontra-se em nível alto e baixo, Figura 39.

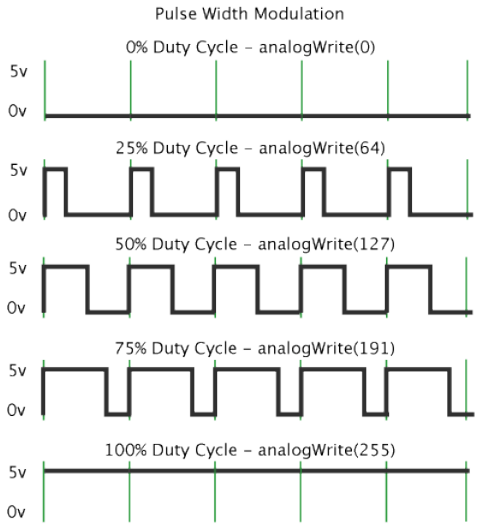

Figura 39 - Modulação por largura de pulso

Fonte: http://arduino.cc/en

O valor de *PWM* foi ajustado de forma a obter 20 e 75% de seu valor total, Figura 40 e Figura 41, respectivamente, e com isso comparar os valores teóricos com os práticos.

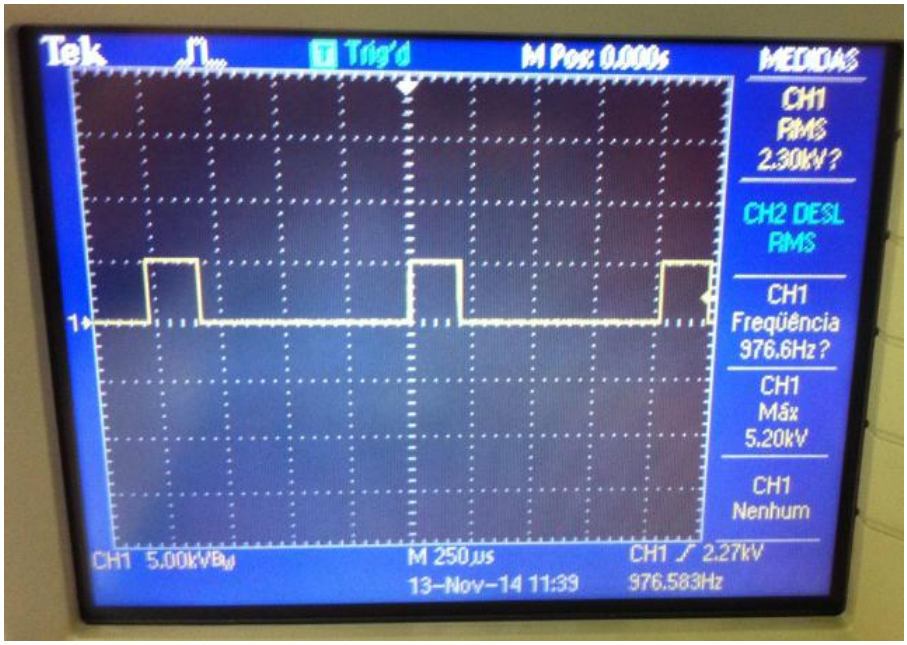

Figura 40 - *PWM* configurado a 20%

Fonte: Própria

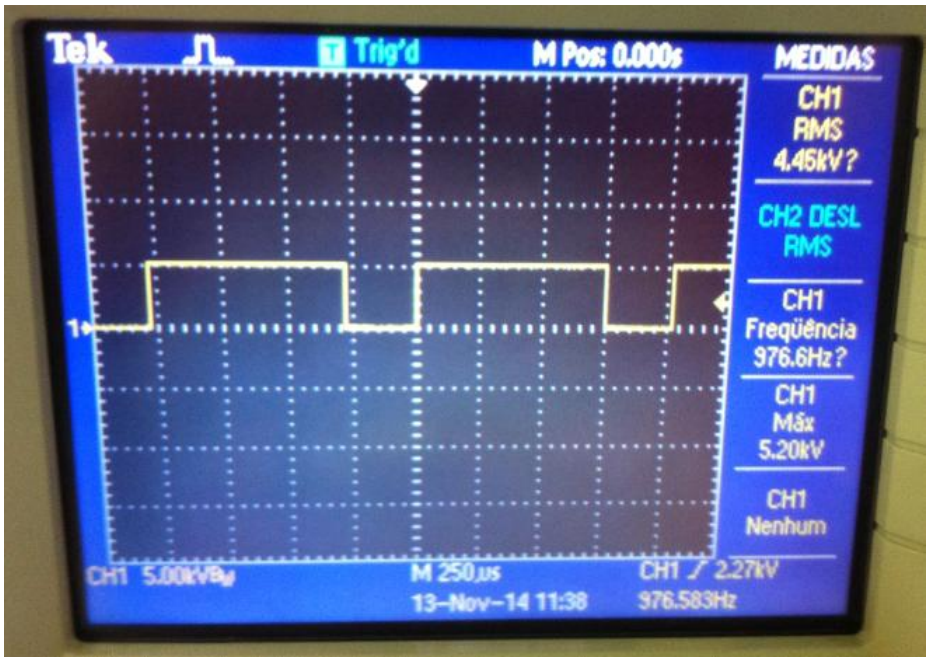

Figura 41 - *PWM* configurado a 75%

Fonte: Própria

Correlacionando os valores teóricos e práticos, incluindo o erro associado tem-se a Tabela 5.

Tabela 5 - Comparação teórica e prática do valor de *PWM*

PWM [%]	Tensão [V]		Frequência [Hz]		t alto [us]		t baixo [us]	
	Teórico	Prático	Teórico	Prático	Teórico	Prático	Teórico	Prático
20	1,00	1,01	980,0	976,583	204,0	203,5	816,4	820,5
75	3,75	3,79	980,0	976,583	765,3	755,5	255,1	268,5

Fonte: Própria

Comparando os resultados o erro apresentado em todos os casos é inferior a 1,5%, o que evidencia a precisão do sistema e robustez.

- **AIN**

A entrada analógica foi utilizada para medir o nível de tensão continua correspondente a corrente de operação do motor. O sensor de corrente ACS712ELCTR-05B-T, Figura 42, atua como transdutor responsável por converter o valor de corrente do motor em tensão elétrica.

Figura 42 - Sensor de corrente ACS712ELCRT-05B-T

Fonte: Própria

Durante os testes iniciais apresentou o valor médio de tensão em sua saída conforme especificado em sua curva correspondente, Figura 24.

Para a verificação do sistema a entrada analógica foi avaliada com os mesmos valores de *PWM* adotados para validação deste, portanto 20 e 75% de seu valor total e com isso compararem os valores teóricos com os práticos.

Correlacionando os valores teóricos e práticos tem-se a Tabela 6.

Tabela 6 - Comparação teórica e prática do valor de corrente

PWM [%]	Tensão Motor [V]		Corrente [mA]		Tensão AIN [V]	
	Teórico	Prático	Teórico	Prático	Teórico	Prático
00	0,00	0,05	00,00	10,50	02,500	02,502
20	2,40	2,41	09,60	15,50	02,501	02,510
75	9.00	9,10	36,00	42,00	02,508	02,520

Fonte: Própria

O sensor ACS712ELCTR-05B-T atua em uma faixa de corrente de -5 a 5 A, valores esses superiores a corrente do motor, situada na faixa de 40 a 80 mA, ou seja, muito inferior a faixa de operação do sensor, ampliando assim o erro encontrado.

Graficamente a medição de tensão correspondente ao erro quanto o motor não está em operação é exibido na Figura 43.

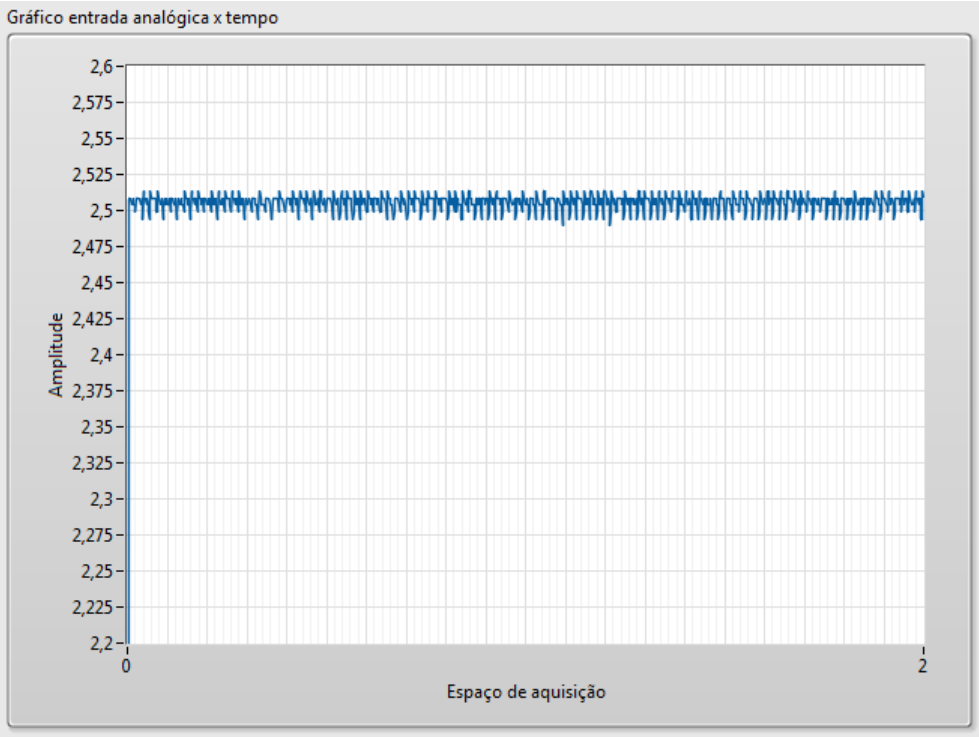

Figura 43 - Medição de corrente pelo sistema

Fonte: Própria

O erro apresentado com o sistema inoperante ocorre devido a ruídos no canal analógico e faixa de operação do sensor de corrente, em relação ao motor. Os valores de tensão a serem medidos situam-se em uma faixa próxima ao erro do canal analógico, o que prejudica a medição.

No entanto caso uma carga com valor de corrente mais próxima a faixa atual do sensor seja empregada o sistema operará normalmente.

Também como alternativa é possível elevar a variação de tensão de saída do sensor utilizando um circuito amplificador operacional, Figura 44.

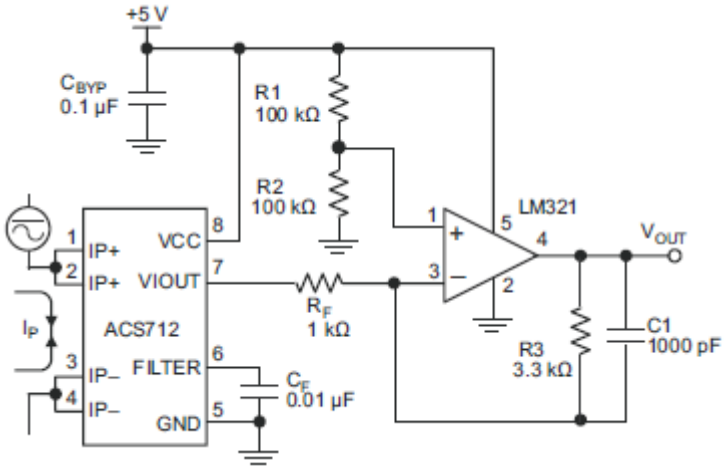

Figura 44 - Aumento do ganho do sensor de corrente em 610 mV/A

Fonte: www.allegromicro.com

O circuito proposto, Figura 44, elevara o ganho do sensor em 610 mV/A, com isso sinais que possuem pequena variação comparados com a faixa de operação sensor podem ser melhor mensurados, Tabela 7.

Tabela 7 - Ganho sensor ACS712ELCTR-05B-T

Saída sem ganho [mV/A]	Saída com ganho [mV/A]
200	810

Fonte: Própria

Comparando os resultados e o erro apresentado em todos os casos é possível verificar que mesmo com as limitações é possível medir a corrente do motor e avaliar seu funcionamento.

8. CONCLUSÃO

No presente trabalho foi apresentado o projeto e desenvolvimento de um sistema microcontrolado de código aberto para a aquisição e condicionamento de sinais elétricos em tempo real, podendo ser empregado em indústrias, universidades, laboratórios e protótipos gerais.

O grande desafio da atualidade, nos diversos ramos, é criar inovações seguras e confiáveis frentes a concorrência que fidelizem o usuário final. O sistema proposto apresenta uma solução viável, robusta, funcional e prática.

Comparando-o com as soluções comerciais atuais o mesmo tem um custo visivelmente menor, em média de oito a dez vezes. Sistemas similares ao proposto, porém sem a flexibilidade e possibilidade de alteração estrutural, tem custo médio de R$ 550,00.

Aliado a esses pontos ainda é possível modificá-la conforme necessidade o que reduz ainda mais o custo frente às soluções comerciais existentes e expande sua empregabilidade, seja ela simples ou complexa.

Todos os requesitos gerais do sistema foram atingidos e seus resultados visualizados dentro do esperado. As tarefas propostas foram concluídas e finalizadas de acordo com o previsto.

9. REFERÊNCIAS

BOLTON, W. **Instrumentação & Control**e. Brasil: Hemus, 2005.

B. Mangolds. Rudi: **A computer-controlled test-data-acquisition and processing system.** *IEEE Transactions on Instrumentation and Measurement*, pp 209-217, novembro 1971.

DENARDIN, G.W, **Microcontroladores**. Disponível em < http://pessoal.utfpr.edu.br/gustavo/apostila_micro.pdf/>. Acesso em 25 de abril de 2014.

LABCENTER ELETRONICS. **Proteus 8.2 VSM**. Disponível em < http://www.labcenter.com/download/prodemo_download.cfm#professional> Acesso em 28 de agosto de 2014.

NATIONAL INSTRUMENTS. **ARDUINO Firmware**. Disponível em < http://vishots.com/getting-started-with-the-labview-interface-for-arduino/>. Acesso em 15 de julho de 2014.

NATIONAL INSTRUMENTS. **ARDUINO LabVIEW Interface For ARDUINO**. Disponível em < http://www.labviewhacker.com/doku.php?id=libraries:lifa:how_it_works/>. Acesso em 16 de julho de 2014.

NATIONAL INSTRUMENTS. **História do LabVIEW**. Disponível em <http://www.ni.com/video/3024/pt/>. Acesso em 25 de abril de 2014.

NATIONAL INSTRUMENTS. **LabVIEW 2012**. Disponível em < http://www.ni.com/download/labview-development-system-2012-sp1/3692/en/>. Acesso em 25 de maio de 2014.

NATIONAL INSTRUMENTS. **IEEE 488 and VXIbus Control, Data Acquisition, and Analysis**, 1993.

OLIVEIRA, Alex Venâncio de. **Projeto e desenvolvimento de um condicionador de sinais**. Disponível em < http://www.bibliotecadigital.unicamp.br/>. Acesso em 24 de abril de 2014.

P.D Lawrence e K. Mauch. **Real-Time Microcomputer System Design: an introduction**. McGRAW-HILL, 1998.

P.M. Forgues e M. Goldeberg. **Microprocessor in biomedical instrumentation.** *IEEE Transactions on Instrumentation and Measurement*, pp 250-253, dezembro 1979.

ROSÁRIO, João M. **Princípios de mecatrônica**. São Paulo: Prentice-Hall, 2005.

SILVEIRA, Paulo R. da; SANTOS, Winderson E. dos. **Automação e controle discreto**. São Paulo: Érica, 1999.

WIKIPEDIA. **Firmware**. Disponível em <http://pt.wikipedia.org/wiki/Firmware>. Acesso em 16 de julho de 2014.

10. ANEXO

10.1 LabVIEWInterface.ino

```
/*********************************************************************************
 **  LVIFA_Firmware - Provides Func For Interfacing With The Arduino Uno
 **  Written By:    Sam Kristoff - National Instruments
 **  Written On:    November 2010
 **  Last Updated:  Dec 2011 - Kevin Fort - National Instruments
**  This File May Be Modified And Re-Distributed Freely.
**  Written By Sam Kristoff And Available At www.ni.com/arduino.
*********************************************************************************/
#include <Wire.h>
#include <SPI.h>
#include <LiquidCrystal.h>
//Includes for IR Remote
#ifndef IRremoteInt_h
#include "IRremoteInt.h"
#endif
#ifndef IRremote_h
#include "IRremote.h"
#endif
/*********************************************************************************
 ** Optionally Include And Configure Stepper Support
 *********************************************************************************/
#ifdef STEPPER_SUPPORT
  // Stepper Modifications
  #include "AFMotor.h"
  #include "AccelStepper.h"
  // Adafruit shield
  AF_Stepper motor1(200, 1);
  AF_Stepper motor2(200, 2);
  // you can change these to DOUBLE or INTERLEAVE or MICROSTEP
  // wrappers for the first motor
  void forwardstep1() {
    motor1.onestep(FORWARD, SINGLE);  }
  void backwardstep1() {
    motor1.onestep(BACKWARD, SINGLE);  }
  // wrappers for the second motor
  void forwardstep2() {
    motor2.onestep(FORWARD, SINGLE);  }
  void backwardstep2() {
    motor2.onestep(BACKWARD, SINGLE);  }
  AccelStepper steppers[8];  //Create array of 8 stepper objects
#endif
// Variables
unsigned int retVal;
int sevenSegmentPins[8];
int currentMode;
unsigned int freq;
unsigned long duration;
int i2cReadTimeouts = 0;
char spiBytesToSend = 0;
char spiBytesSent = 0;
char spiCSPin = 0;
```

```
char spiWordSize = 0;
Servo *servos;
byte customChar[8];
LiquidCrystal lcd(0,0,0,0,0,0,0);
unsigned long IRdata;
IRsend irsend;
// Sets the mode of the Arduino (Reserved For Future Use)
void setMode(int mode)
{  currentMode = mode;}
// Checks for new commands from LabVIEW and processes them if any exists.
int checkForCommand(void){
#ifdef STEPPER_SUPPORT
  // Call run function as fast as possible to keep motors turning
  for (int i=0; i<8; i++){
    steppers[i].run();
  }
#endif
  int bufferBytes = Serial.available();
  if(bufferBytes >= COMMANDLENGTH)   {
    // New Command Ready, Process It
    // Build Command From Serial Buffer
    for(int i=0; i<COMMANDLENGTH; i++)    {
      currentCommand[i] = Serial.read();   }
    processCommand(currentCommand);
    return 1;
  }  else  {
    return 0;   }}
// Processes a given command
void processCommand(unsigned char command[])
{    // Determine Command
  if(command[0] == 0xFF && checksum_Test(command) == 0)
  {    switch(command[1])    {
    /*************************************************************************************
    ** LIFA Maintenance Commands
    *************************************************************************************/
    case 0x00:     // Sync Packet
      Serial.print("sync");
      Serial.flush();
      break;
    case 0x01:    // Flush Serial Buffer
      Serial.flush();
      break;
   /*************************************************************************************
    ** Low Level - Digital I/O Commands
    *************************************************************************************/
    case 0x02:    // Set Pin As Input Or Output
      pinMode(command[2], command[3]);
      Serial.write('0');
      break;
    case 0x03:    // Write Digital Pin
      digitalWrite(command[2], command[3]);
       Serial.write('0');
      break;
    case 0x04:    // Write Digital Port 0
      writeDigitalPort(command);
```

```
      Serial.write('0');
     break;
   case 0x05:    //Tone
     freq = ( (command[3]<<8) + command[4]);
     duration=(command[8]+  (command[7]<<8)+
(command[6]<<16)+(command[5]<<24));
      if(freq > 0)  {
       tone(command[2], freq, duration);  }
     else {
       noTone(command[2]);  }
      Serial.write('0');
     break;
   case 0x06:    // Read Digital Pin
     retVal = digitalRead(command[2]);
     Serial.write(retVal);
     break;
   case 0x07:    // Digital Read Port
     retVal = 0x0000;
     for(int i=0; i <=13; i++)
     { if(digitalRead(i)) {
        retVal += (1<<i);  }  }
     Serial.write( (retVal & 0xFF));
     Serial.write( (retVal >> 8));
     break;
   /*****************************************************************************
   ** Low Level - Analog Commands
   *****************************************************************************/
   case 0x08:    // Read Analog Pin
     retVal = analogRead(command[2]);
     Serial.write( (retVal >> 8));
     Serial.write( (retVal & 0xFF));
     break;
   case 0x09:    // Analog Read Port
     analogReadPort();
     break;
    /*****************************************************************************
   ** Low Level - PWM Commands
   *****************************************************************************/
   case 0x0A:    // PWM Write Pin
     analogWrite(command[2], command[3]);
      Serial.write('0');
     break;
   case 0x0B:    // PWM Write 3 Pins
     analogWrite(command[2], command[5]);
     analogWrite(command[3], command[6]);
     analogWrite(command[4], command[7]);
      Serial.write('0');
     break;
   /*****************************************************************************
   ** Sensor Specific Commands
   *****************************************************************************/
   case 0x0C:    // Configure Seven Segment Display
     sevenSegment_Config(command);
      Serial.write('0');
     break;
```

```
    case 0x0D:    // Write To Seven Segment Display
      sevenSegment_Write(command);
       Serial.write('0');
      break;
    /*****************************************************************************
    ** Continuous Aquisition
    *****************************************************************************/
      case 0x2A:  // Continuous Aquisition Mode On
        acqMode=1;
        contAcqPin=command[2];
        contAcqSpeed=(command[3])+(command[4]<<8);
        acquisitionPeriod=1/contAcqSpeed;
        iterationsFlt =.08/acquisitionPeriod;
        iterations=(int)iterationsFlt;
        if(iterations<1)
        {iterations=1;}
        delayTime= acquisitionPeriod;
       if(delayTime<0){
         delayTime=0;}
        break;
      case 0x2B:  // Continuous Aquisition Mode Off
        acqMode=0;
        break;
     case 0x2C:  // Return Firmware Revision
         Serial.write(byte(FIRMWARE_MAJOR));
         Serial.write(byte(FIRMWARE_MINOR));
        break;
     case 0x2D:  // Perform Finite Aquisition
         Serial.write('0');

finiteAcquisition(command[2],(command[3])+(command[4]<<8),command[5]+(comman
d[6]<<8));
        break;
    /*****************************************************************************
    ** Unknown Packet
    *****************************************************************************/
    default:      // Default Case
      Serial.flush();
      break ; } }
  else{
    // Checksum Failed, Flush Serial Buffer
    Serial.flush(); } }
/*****************************************************************************
**  Functions
*****************************************************************************/
// Writes Values To Digital Port (DIO 0-13).  Pins Must Be Configured As Outputs
Before Being Written To
void writeDigitalPort(unsigned char command[])
{
  digitalWrite(13, (( command[2] >> 5) & 0x01) );
  digitalWrite(12, (( command[2] >> 4) & 0x01) );
  digitalWrite(11, (( command[2] >> 3) & 0x01) );
  digitalWrite(10, (( command[2] >> 2) & 0x01) );
  digitalWrite(9, (( command[2] >> 1) & 0x01) );
  digitalWrite(8, (command[2] & 0x01) );
```

```
  digitalWrite(7, (( command[3] >> 7) & 0x01) );
  digitalWrite(6, (( command[3] >> 6) & 0x01) );
  digitalWrite(5, (( command[3] >> 5) & 0x01) );
  digitalWrite(4, (( command[3] >> 4) & 0x01) );
  digitalWrite(3, (( command[3] >> 3) & 0x01) );
  digitalWrite(2, (( command[3] >> 2) & 0x01) );
  digitalWrite(1, (( command[3] >> 1) & 0x01) );
  digitalWrite(0, (command[3] & 0x01) ); }
// Reads all 6 analog input ports, builds 8 byte packet, send via RS232.
void analogReadPort()
{  // Read Each Analog Pin
  int pin0 = analogRead(0);
  int pin1 = analogRead(1);
  int pin2 = analogRead(2);
  int pin3 = analogRead(3);
  int pin4 = analogRead(4);
  int pin5 = analogRead(5);
  //Build 8-Byte Packet From 60 Bits of Data Read
  char output0 = (pin0 & 0xFF);
  char output1 = ( ((pin1 << 2) & 0xFC) | ( (pin0 >> 8) & 0x03) );
  char output2 = ( ((pin2 << 4) & 0xF0) | ( (pin1 >> 6) & 0x0F) );
  char output3 = ( ((pin3 << 6) & 0xC0) | ( (pin2 >> 4) & 0x3F) );
  char output4 = ( (pin3 >> 2) & 0xFF);
  char output5 = (pin4 & 0xFF);
  char output6 = ( ((pin5 << 2) & 0xFC) | ( (pin4 >> 8) & 0x03) );
  char output7 = ( (pin5 >> 6) & 0x0F );
  // Write Bytes To Serial Port
  Serial.print(output0);
  Serial.print(output1);
  Serial.print(output2);
  Serial.print(output3);
  Serial.print(output4);
  Serial.print(output5);
  Serial.print(output6);
  Serial.print(output7);
}
// Configure digital I/O pins to use for seven segment display
// Synchronizes with LabVIEW and sends info about the board and firmware
(Unimplemented)
void syncLV(){
  Serial.begin(DEFAULTBAUDRATE);
  i2cReadTimeouts = 0;
  spiBytesSent = 0;
  spiBytesToSend = 0;
  Serial.flush();
}
// Compute Packet Checksum
unsigned char checksum_Compute(unsigned char command[])
{
  unsigned char checksum;
  for (int i=0; i<(COMMANDLENGTH-1); i++)  {
    checksum += command[i]; }
  return checksum;
}
// Compute Packet Checksum And Test Against Included Checksum
```

```
int checksum_Test(unsigned char command[])
{ unsigned char checksum = checksum_Compute(command);
  if(checksum == command[COMMANDLENGTH-1])
  {return 0;
  }else  {
    return 1;
  }}
void sampleContinously()
{for(int i=0; i<iterations; i++)
  { retVal = analogRead(contAcqPin);
     if(contAcqSpeed>1000) //delay Microseconds is only accurate for values less that
16383
     { Serial.write( (retVal >> 2));
       delayMicroseconds(delayTime*1000000); //Delay for neccesary amount of time to
achieve desired sample rate
     } else{
        Serial.write( (retVal & 0xFF) );
        Serial.write( (retVal >> 8));
        delay(delayTime*1000);
     }  }}
void finiteAcquisition(int analogPin, float acquisitionSpeed, int numberOfSamples)
{//want to exit this loop every 8ms
   acquisitionPeriod=1/acquisitionSpeed;
     for(int i=0; i<numberOfSamples; i++)
  {retVal = analogRead(analogPin);
   if(acquisitionSpeed>1000)
     {Serial.write( (retVal >> 2));
       delayMicroseconds(acquisitionPeriod*1000000);
     }  else  {
       Serial.write( (retVal & 0xFF) );
       Serial.write( (retVal >> 8));
       delay(acquisitionPeriod*1000);
     } }}
```

10.2 LIFA_Base.ino

```
/*****************************************************************************
 **  LVFA_Firmware - Provides Basic Arduino Sketch For Interfacing With LabVIEW.
 **  Written By:    Sam Kristoff - National Instruments
 **  Written On:    November 2010
 **  Last Updated:  Dec 2011 - Kevin Fort - National Instruments
 **  This File May Be Modified And Re-Distributed Freely. Original File Content
 **  Written By Sam Kristoff And Available At www.ni.com/arduino.
 *****************************************************************************/
/*****************************************************************************
 ** Includes.
 *****************************************************************************/
// Standard includes.  These should always be included.
#include <Wire.h>
#include <SPI.h>
#include <Servo.h>
#include "LabVIEWInterface.h"
/*****************************************************************************
 **  setup()
 **  Initialize the Arduino and setup serial communication.
 **  Input:  None
 **  Output: None
 *****************************************************************************/
void setup()
{
  // Initialize Serial Port With The Default Baud Rate
  syncLV();
  // Place your custom setup code here}
/*****************************************************************************
 **  loop()
 **  The main loop.  This loop runs continuously on the Arduino.  It
 **  receives and processes serial commands from LabVIEW.
 **  Input:  None
 **  Output: None
 *****************************************************************************/
void loop()
{
  // Check for commands from LabVIEW and process them.
  checkForCommand();
  // Place your custom loop code here (this may slow down communication with
LabVIEW)
   if(acqMode==1)
  { sampleContinously();
  }}
```

10.3 Esquema elétrico

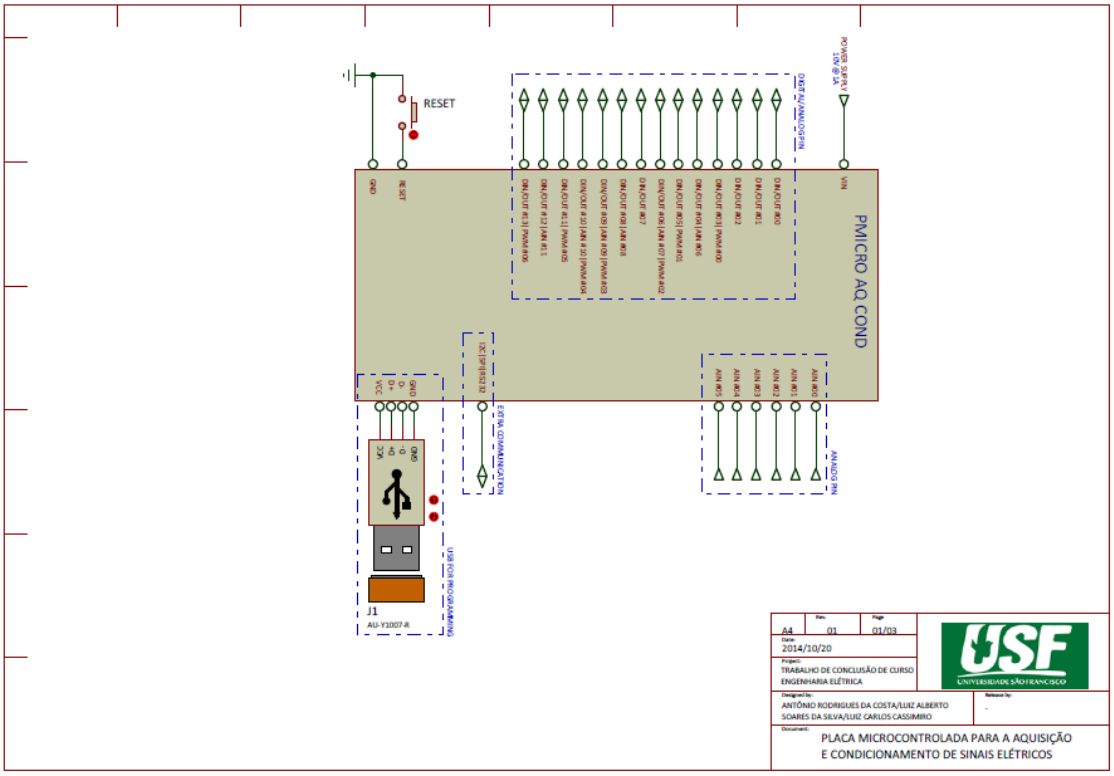

Figura 45 - Esquema elétrico geral

Fonte: Própria

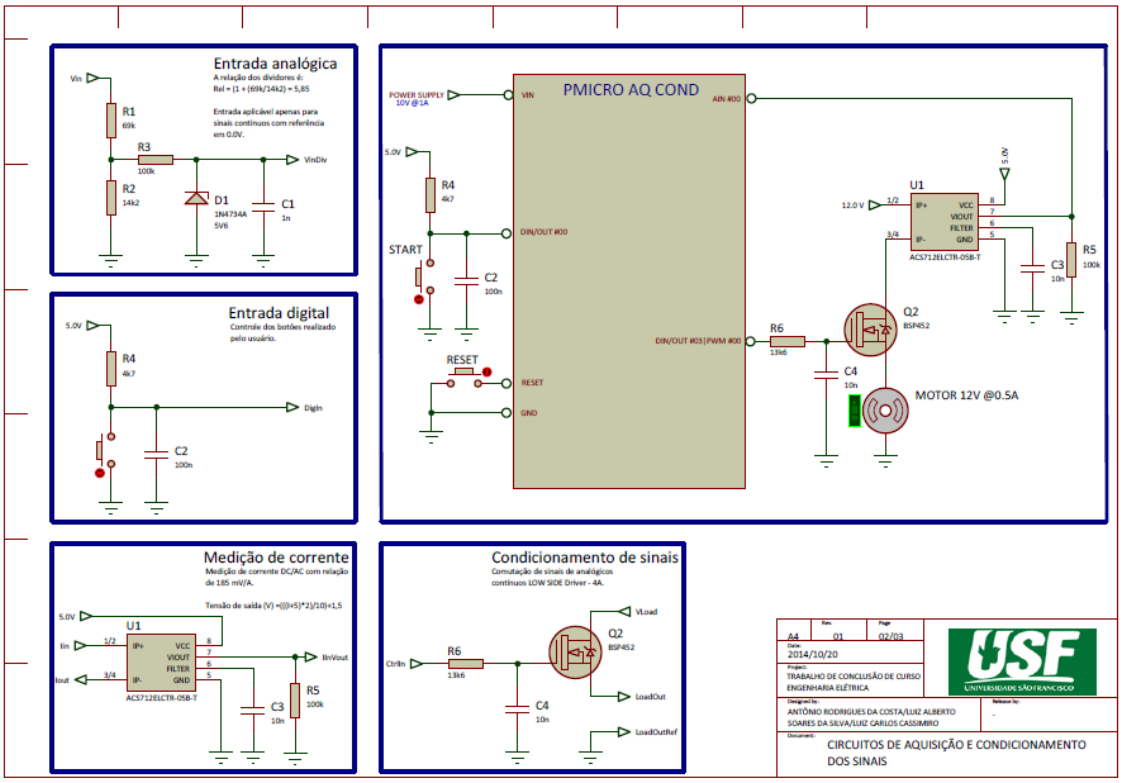

Figura 46 - Esquema elétrico subcircuitos

Fonte: Própria