



UNIVERSIDADE FEDERAL DE UBERLÂNDIA

FACULDADE DE ENGENHARIA MECÂNICA

PEDRO HENRIQUE DEMÉTRIO ABRÃO

**DESENVOLVIMENTO DO DISPOSITIVO PARA MOVIMENTO DE UMA SONDA
PARA A ANÁLISE DA TENSÃO NO ARCO DE SOLDAGEM**

Uberlândia

2020

PEDRO HENRIQUE DEMÉTRIO ABRÃO

**DESENVOLVIMENTO DO DISPOSITIVO PARA MOVIMENTO DE UMA SONDA PARA A
ANÁLISE DA TENSÃO NO ARCO DE SOLDAGEM**

Monografia apresentada ao Curso de Graduação em Engenharia Mecatrônica da Universidade Federal de Uberlândia, como parte dos requisitos para obtenção do título de ENGENHEIRO MECATRÔNICO.

Área de Concentração: Instrumentação e

Automação industrial

Orientador: Professor Volodymyr Ponomarov

Uberlândia

2020

Monografia defendida e aprovada, 13 de março de 2020, pela comissão avaliadora constituída pelos professores:

Professor: Volodymyr Ponomarov

Professor: José Jean-Paul Zanlucchi de Souza Tavares

Professor: Rodrigo Hiroshi Murofushi

ABRÃO, P. H. D., **DESENVOLVIMENTO DO DISPOSITIVO PARA MOVIMENTO DE UMA SONDA PARA A ANÁLISE DA TENSÃO NO ARCO DE SOLDAGEM. 2020. 47p.** Monografia de Conclusão de Curso, Faculdade de Engenharia Mecânica, Universidade Federal de Uberlândia, Uberlândia, MG, Brasil.

Resumo

Com a popularização dos microcontroladores e do *open hardware* (“*Hardware livre*”), a construção de sistemas embarcados tornou-se economicamente viável. A construção de um sistema embarcado envolve conceitos relacionados com *hardware* e *software*, que necessitam estarem integrados através de um microcontrolador. Neste trabalho, propomos o projeto de controlar o movimento (posição e velocidade) de um motor de corrente contínua, utilizando a plataforma Arduino® e o microcontrolador ATmega 328.

Palavras chave: Arduino®, ATmega328, Sistemas Embarcados, Motor CC.

ABRÃO, P. H. D., **DEVELOPMENTS OF THE DEVICE FOR MOVING A PROBE FOR THE ANALYSIS OF THE TENSION IN THE WELDING ARC VOLTAGE. 2020. 47p.** Completion Monograph, Mechanical Engineering School, Federal University of Uberlândia, Uberlândia, MG, Brazil.

ABSTRACT

Along with the popularization of microcontrollers and open hardware, the construction of embedded systems has become economically viable. The construction of embedded system involves the use of concepts related to hardware and software that need to be integrated through a microcontroller. In this work we propose the project to control the movement of a DC motor, using the Arduino® platform and the ATmega 328 microcontroller.

Keywords: Arduino®, ATmega328, Embedded Systems, DC Motor.

Lista de Figuras

Figura 1: Estrutura e a distribuição da tensão ao longo do arco elétrico.

Figura 2: Proposta de um dispositivo para medir as quedas de tensão no arco elétrico usando uma sonda.

Figura 3: Diagrama esquemático de um microcontrolador.

Figura 4: Display LCD 16x4.

Figura 5: Motor de corrente contínua.

Figura 6: Diagrama de um circuito "Ponte H".

Figura 7: Circuito integrado L293D.

Figure 8: Exemplo de sistema em malha fechada.

Figure 9: Exemplo de sistema em malha aberta.

Figure 10: Montagem do circuito.

Figura 11: Perfil em U.

Figura 12: Modelo de estrutura de sustentação para o controlador do motor.

Figura 13: Estrutura montada.

Figura 14: Protótipo do sistema proposto.

Figura 15: Placa de Prototipagem e controlador.

Figura 16: Potenciômetro para controle do motor.

Figura 17: Interface Arduino IDE.

Figura 18: Arduino Mega 2560 R3.

Figura 19: Shield L293D.

Lista de Tabelas

Tabela 1: Características do Arduino® Mega 2560 R3.

Lista de Unidade e Siglas

RISC	Reduced Instruction Set Computer
SRAM	Static random-access memory
EEPROM	Electrically-Erasable Programmable Read-Only Memory
ADC	Analog-digital Converter
DAC	Digital-analog Converter
PWM	Pulse Width Modulation
LCD	Liquid Crystal Display
[V]	Volt
[MHz]	Megaheartz
USB	Universal Serial Bus
[°C]	Graus Celsius
[mA]	Milliampere
[μA]	Microampere
TWI	Two Wire Interface
USART	Universal Synchronous Asynchronous Receiver Transmitter
I2C	Inter-Intergrate Circuit
CC	Corrente Contínua
SPI	Serial Peripheral Interface
I/O	Input/Output

Sumário

CAPÍTULO I – INTRODUÇÃO	09
1.1 Objetivos	13
1.2 Objetivos Específicos	13
1.3 Justificativa	14
CAPÍTULO II – INTRODUÇÃO	15
2.1 Sistemas Embarcados	15
2.2 Microcontroladores	15
2.2.1 Microcontrolador AVR	16
2.2.2 Atmega328	18
2.3 Software e Hardware	19
2.4 Arduino®	20
2.5 Shields	21
2.5.1 LCD 16x4	21
2.5.2 Motor de Corrente Contínua	22
2.6 Ponte H	22
2.7 Controle de Processos	24
2.7.1 Controle em Malha Fechada x Malha Aberta	24
CAPÍTULO III – METODOLOGIA	27
3.1 Avaliação do Sistema	27
3.1.1 Montagem do Circuito	27
3.1.2 Instalação do Circuito na Base	28
3.1.3 Instalação do Código	29
CAPÍTULO IV – DESCRIÇÃO DO PROJETO	30

4.1 Hardware	30
4.1.1 Materiais Utilizados	30
4.1.2 Descrição do Hardware	30
4.2 Software	32
4.2.1 Descrição do Programa	32
4.2.2 Linguagem de Programação	33
4.2.3 Arduino IDE	34
4.2.4 Arduino® Mega 2560	35
4.3 Shield L293D	36
CAPÍTULO V – CONCLUSÃO	38
CAPÍTULO VI – TRABALHOS FUTUROS	39

CAPÍTULO I

INTRODUÇÃO

As melhorias tecnológicas demandam cada vez mais dispositivos eletrônicos. Assim, a cada dia são criados componentes eletrônicos mais versáteis e poderosos. Nessa categoria, os microprocessadores, aos quais os microcontroladores pertencem, têm alcançado grande desenvolvimento. Sua facilidade de uso em amplas faixas de aplicações permite o projeto relativamente rápido e fácil de novos equipamentos.

Atualmente, os microcontroladores estão presentes em quase todos os dispositivos eletrônicos controlados digitalmente, como por exemplo: nas casas, em máquinas de lavar, fornos de micro-ondas, televisores, aparelhos de som e imagem, condicionadores de ar e telefones; nos veículos, em sistemas eletrônicos de controle de injeção de combustível, controle de estabilidade, freios ABS “*Anti-lock Braking System*”, computadores de bordo e GPS “*Global Positioning System*”; nos eletrônicos portáteis, tocadores de mídia eletrônica, vídeo games e relógios; na indústria, em controladores lógicos programáveis, de motores e fontes de alimentação. Em resumo, os sistemas microcontrolados encontram-se em todos os segmentos eletrônicos, desde simples equipamentos domésticos até sistemas industriais complexos (VILLAÇA, 2012).

Dentre a enorme gama de componentes e equipamentos controlados por microcontroladores, motores de corrente contínua são um dos que se destacam. Diz-se que um motor é de corrente contínua quando em todos os seus terminais as grandezas que a caracterizam (tensões e correntes) são unidireccionais.

Além do desenvolvimento dos microcontroladores, outro ramo da engenharia que possibilitou o avanço tecnológico é o controle, extremamente importante atualmente para a eficiência de processos industriais. O controle automático tem papel fundamental no avanço da engenharia e da ciência, ele está presente nos diversos campos e vai desde o controle de veículos espaciais, direcionamento de mísseis a máquinas industriais utilizadas diariamente, com como as de controle de pressão, temperatura, umidade, etc (OGATA, 2010).

Dentre a gama de aplicações dos microcontroladores, a soldagem por fusão a arco elétrico é um processo industrial que pode ser estudado e controlado. Neste processo a transferência de calor para o material de base é de extrema importância. As propriedades térmicas determinam como serão as propriedades finais da junta, e também dos parâmetros da microestrutura.

O arco elétrico é composto por três regiões definidas, a zona anódica, catódica e a coluna, como visto na Figura 1. A física do arco é definida por reações elétricas e químicas, que é convertida em calor e luz, onde para valores maiores de correntes geram mais calor. O processo de soldagem TIG “*Tungsten Inert Gas*” utiliza do calor originado do arco para fundir a peça a soldar.

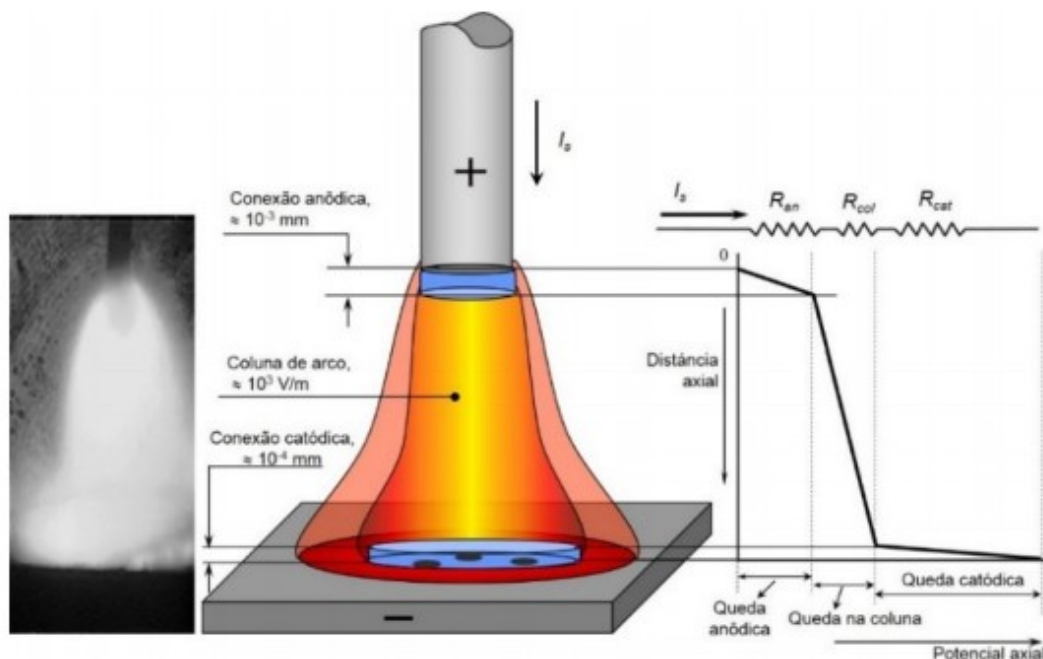


Figura 1: Estrutura e a distribuição da tensão ao longo do arco elétrico.

Fonte: Por Prof. Dr. Volodymyr Ponomarov

Uma das formas de estudar o comportamento das quedas de tensão e corrente no arco elétrico, é por meio da coleta de dados utilizando sondas que trabalham passando no interior das regiões do arco. Tal estudo tem sido recriado por inúmeros pesquisadores, como o realizado na instituição *Cranfield University* no ano de 2002, com o tema *Characterisation of TIG Arcs Structures Using Experimental Techniques*

(VILARINHO, 2002). O presente trabalho visa uma releitura da estrutura necessária para tais estudos.

1.1 OBJETIVOS

Este projeto de conclusão e curso tem como objetivo principal o desenvolvimento de um dispositivo para medir as tensões no arco elétrico de soldagem, por meio de sondas acopladas ao eixo de um motor de corrente contínua, com rotação variável atravessando o arco na direção transversal, similar ao mostrado na Figura 2.

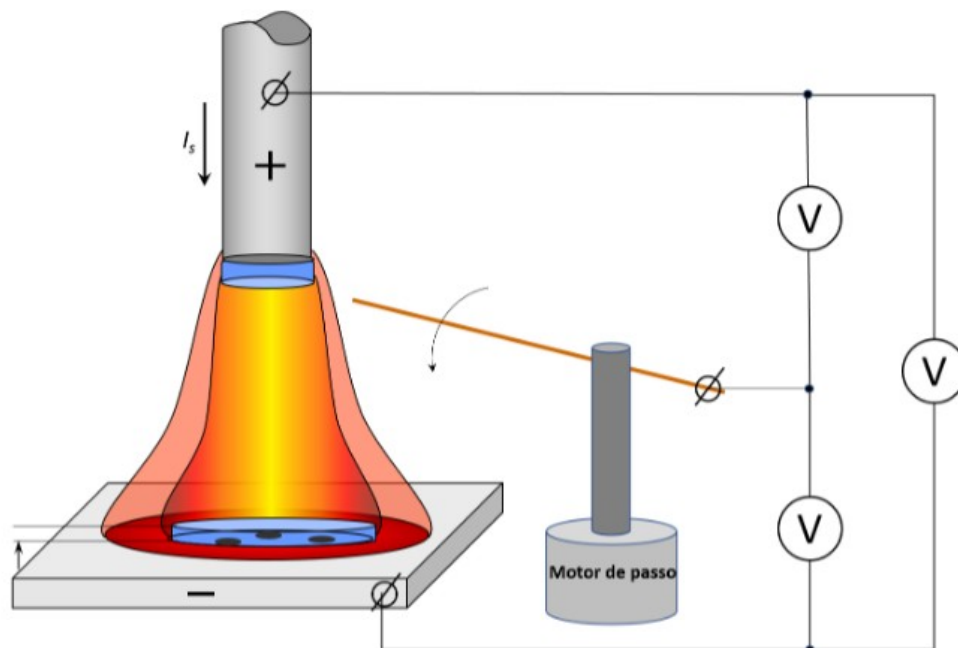


Figura 2: Proposta de um dispositivo para medir as quedas de tensão no arco elétrico usando uma sonda.

Fonte: Por Prof. Dr. Volodymyr Ponomarev

1.2 OBJETIVOS ESPECÍFICOS

Os objetivos específicos deste trabalho são:

- Avaliação do sistema proposto;
- Identificar os componentes que melhor se adequem ao projeto proposto;
- Montagem de um circuito para o controle do motor de corrente contínua;

- Instalar toda instrumentação e implementar o dispositivo para o controle do motor e coleta de dados do arco elétrico de soldagem;
- Implementa o código computacional a ser carregado na placa de controle;
- Realizar testes para a correção de eventuais erros;
- Montagem final do dispositivo na estrutura.

1.3 JUSTIFICATIVA

A pesquisa desenvolvida no Laboratório para Pesquisa e Desenvolvimento de Processos em Soldagem (LAPROSOLDA), envolve uma equipe com vários integrantes, sendo cada discente responsável por uma etapa do projeto global.

A necessidade do presente trabalho surgiu a partir do propósito de realizar estudos mais detalhados sobre a física do arco elétrico de soldagem, focando principalmente na corrente elétrica e as quedas de tensões geradas nas zonas anódica, catódica e coluna.

CAPÍTULO II

FUNDAMENTAÇÃO TEÓRICA

2.1 SISTEMAS EMBARCADOS

Um sistema embarcado é um sistema baseado em um microcontrolado no qual o computador é completamente encapsulado ou dedicado ao dispositivo ou sistema que ele controla.

Diferentemente de computadores de propósito geral, como o computador pessoal, um sistema embarcado realiza um conjunto de tarefas predefinidas, geralmente com requisitos específicos. Já que o sistema é dedicado a tarefas específicas, através de engenharia pode-se otimizar o projeto reduzindo tamanho, recursos computacionais e custo do produto. Em geral, tais sistemas não podem ter sua funcionalidade alterada durante o uso. Caso queira-se modificar o propósito é necessário reprogramar todo o sistema (VILLAÇA, 2012).

2.2 MICROCONTROLADORES

Um microcontrolador é o um sistema microprocessado baseado em circuitos integrados com várias funcionalidades (periféricos). Basicamente, um microcontrolador é um microprocessador com memórias de programa, de dados e RAM, temporizadores e circuitos de *clock* embutidos. O único componente externo que pode ser necessário é um cristal para determinar a frequência de trabalho. A grande vantagem de se colocar várias funcionalidades em um único circuito integrado é a possibilidade de desenvolvimento rápido de sistemas eletrônicos com o emprego de um pequeno número de componentes.

Dentre as funcionalidades encontradas nos microcontroladores, pode-se citar: gerador interno independente de *clock* (não necessita de cristal ou componentes externos); memória SRAM, EEPROM e *flash*; conversores analógicos-digitais (ADCs), conversores digitais-analógicos (DACs); vários temporizadores/contadores; comparadores analógicos; saídas PWM; diferentes tipos de interface de comunicação, incluindo USB, USART, I2C, CAN, SPI, JTAG, Ethernet; relógio de tempo real; circuitos para

gerenciamento de energia no chip; circuitos para o controle de inicialização (*reset*); alguns tipos de sensores; interface para LCD; e outros periféricos de acordo com o fabricante. A Figura 3 apresenta um diagrama esquemático de um microcontrolador típico (VILLAÇA, 2012).

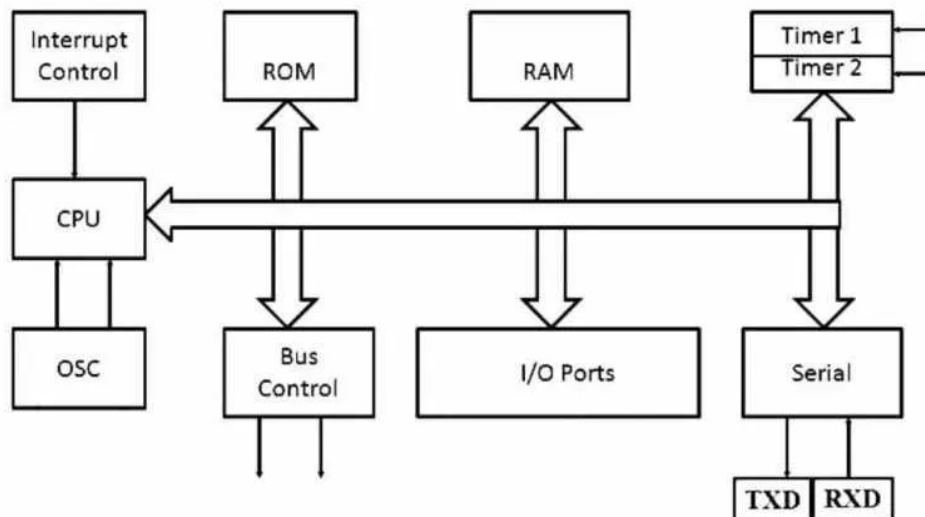


Figura 3: Diagrama esquemático de um microcontrolador.

Fonte: <https://descubrearduino.com/microcontrolador/>

2.2.1 MICROCONTROLADORES AVR

A história dos microcontroladores AVR começa em 1992 na *Norwegian University of Science and Technology* (NTNU), na Noruega, onde dois estudantes de doutorado, Alf-Egil-Boden e Vegard Wollan, defenderam uma tese sobre um microcontrolador de 8 bits com memória de programa flash e arquitetura RISC avançada. Dos seus nomes surgiu a sigla AVR: Alf – Vegard – RISC. Percebendo as vantagens do microcontrolador que haviam criado, Alf e Vegard refinaram o seu projeto por dois anos. Com o AVR aperfeiçoado, foram para a Califórnia (EUA) para encontrar um patrocinador para a ideia, a qual foi comprada pela Atmel, que lançou comercialmente o primeiro AVR em meados de 1997, o AT90S1200. Desde então, o aperfeiçoamento do AVR é crescente, juntamente a sua popularidade.

O AVR apresenta uma boa eficiência de processamento e um núcleo compacto (poucos milhares de portas lógicas). O desempenho do seu núcleo de 8 bits é superior ao da maioria das outras tecnologias de 8 bits disponíveis atualmente no mercado. Com uma

arquitetura RISC avançada, o AVR apresenta mais de uma centena de instruções e uma estrutura voltada à programação C, permitindo a produção de códigos compactos (menor número de bytes por funcionalidade). Também, apresenta inúmeros periféricos que o tornam adequado para uma infinidade de aplicações.

As principais características dos microcontroladores AVR são:

- Executam a maioria das instruções em 1 ou 2 ciclos de *clock* (poucas em 3 ou 4) e operam com tensões entre 1,8 V e 5,5 V, com velocidades de até 20 MHz. Estão disponíveis em diversos encapsulamentos;
- Alta integração e grande número de periféricos com efetiva compatibilidade entre toda a família AVR;
- Possuem vários modos para redução do consumo de energia e características adicionais (*picoPower*) para sistemas críticos;
- Possuem 32 registradores de propósito geral e instruções de 16 bits (cada instrução ocupa 2 bytes na memória de programa);
- Memória de programação flash programável *in-system*, SRAM e EEPROM;
- Facilmente programados e com *debug in-system* via interface simples, ou com interfaces JTAG compatível com 6 ou 10 pinos;
- Um conjunto completo e gratuito de *softwares*;
- Preço acessível.

Existem microcontroladores AVR específicos para diversas áreas, tais como: automotiva, controle de LCDs, redes de trabalho CAN, USB, controle de motores, controle de lâmpadas, monitoração de bateria, 802.15.4/ZigBee™ (comunicação sem fio, *wireless*) e controle por acesso remoto.

2.2.2 ATMEGA328

O ATmega328 será abordado por ser um microcontrolador utilizado na placa de prototipagem escolhido para o projeto que apresenta a maioria das características da família AVR e uma memória flash de maior tamanho comparado aos microcontroladores

AVR com o mesmo número de pinos. Outro fator decisivo, é que ele é utilizado atualmente na plataforma de desenvolvimento Arduino®, de fácil aquisição. O importante a saber é que, ao se programar esse microcontrolador, os conceitos de programação para qualquer outro da família AVR serão iguais, devido às similaridades entre os componentes. Na maioria dos casos a mudança se dá no uso e nome de alguns registradores e periféricos. Na dúvida, qualquer mudança de hardware e software pode ser resolvida com uma busca ao manual do fabricante.

As principais características do ATmega328 são:

- Microcontrolador de baixa potência, com arquitetura RISC avançada.
- 131 instruções, a maior parte executada em 1 ou 2 ciclos de *clock* (poucas em 3 ou 4 ciclos).
- 32 registradores de trabalho de propósito geral (8 bits cada). Alguns trabalham em par para endereçamentos de 16 bits.
- Operação de até 20 MIPS a 20 MHz.
- Multiplicação por hardware em 2 ciclos de *clock*.
- 32 kbytes de memória de programa flash de auto programação *In-System* (8 k, 16 k, nos respectivos ATmega88 e ATmega168).
- 1 kbytes de memória EEPROM.
- 2 kbytes de memória SRAM.
- Ciclos de escrita e apagamento: memória flash 10 mil vezes, EEPROM 100 mil vezes.
- Seção opcional para código de boot para programação *In-System* por *boot loader*.
- Bits de bloqueio para proteção contra a cópia do *firmware*.
- Tensão de operação: 1,8 - 5,5 V.
- Consumo de corrente a 1 MHz (1,8 V, 25 °C): modo ativo = 0,2 mA e modo *Power-down* = 0,1 µA.
- Possui os seguintes periféricos:
 - 23 entradas e saídas (I/Os) programáveis.
 - 2 Temporizadores/Contadores de 8 bits com *Prescaler* separado, com modo de comparação.
 - 1 Temporizador/Contador de 16 bits com *Prescaler* separado, com modo de comparação e captura.

- Contador de tempo real (com um cristal externo de 32,768 kHz conta precisamente 1 s).
- 6 canais PWM.
- 8 canais AD com resolução de 10 bits na versão TQFP (*Thin profile plastic Quad Flat Package*) e 6 canais na versão PDIP (*Plastic Dual Inline Package*).
- Interface serial para dois fios orientada a byte (TWI), compatível com o protocolo I2C.
- Interface serial USART.
- Interface serial SPI *Master/Slave*.
- *Watchdog Timer* com oscilador interno separado.
- 1 comparador analógico.
- Características especiais:
 - *Power-on Reset* e detecção *Brown-out* programável.
 - Oscilador interno RC (não há a necessidade do uso de cristal externo ou de outra fonte de *clock*).
 - Fontes de interrupções internas e externas (em todos os pinos de I/O).
 - Saída de *clock* em um pino de I/O (PB0).
 - Pull-up habilitáveis em todos os pinos de I/O 6 modos de *Sleep: Idle*, Redução de ruído do ADC, *Power-down*, *Power-save*, *Standby* e *Extended Standby*.
 - Medição de temperatura do encapsulamento.

2.3 SOFTWARE E HARDWARE

O uso de *softwares* é imprescindível quando se trabalha com microcontroladores. O código fonte (programa ou *firmware*) para o microcontrolador necessita ser escrito, compilado, depurado e gravado. Todas essas tarefas são realizadas com o suporte de *softwares* adequados. Assim, neste capítulo é feita uma introdução aos programas necessários para o trabalho com o AVR e de um programa de gravação para uso com a plataforma Arduino®. Também é sugerido o uso de um software de simulação, o Proteus®.

Para o trabalho exclusivamente com a plataforma Arduino® pode ser empregado o seu próprio *software* gratuito (www.arduino.cc). Ele não permite o *debug* do programa e

não é adequado ao desenvolvimento profissional de projetos. Entretanto, o seu ambiente integrado de desenvolvimento – IDE tem grande aceitação para desenvolvimentos rápidos e que não demandem muito conhecimento técnico e desempenho do microcontrolador.

2.4 ARDUINO®

O Arduino® é considerado na comunidade de desenvolvedores (ARDUINO,2014), um bom meio para quem está iniciando em sistemas embarcados, pois fornece o necessário para aplicar conceitos a serem explorados e desenvolver experimentalmente as próprias ideias.

Essa plataforma beneficia grande parte das pessoas com conhecimento em programação, mas sem domínio da eletrônica, que podem realizar experimentos práticos. E também pessoas que possuem conhecimento da eletrônica mas sem conhecimento de programação podem realizar experimentos de alto nível. Abrindo espaço para leigos que desejam criar um protótipo útil que pode chegar a resultados consideráveis (WERNECK, 2009).

Existem outras plataformas de microcontroladores criadas antes mesmo do Arduino® e mantendo o mesmo objetivo. Entretanto não possuíam a característica financeira e a facilidade do Arduino®, um exemplo é o BasicStamp®. O Arduino® pode ser utilizado no desenvolvimento de projetos autônomos ou iterativos, que respondem a sensores ou *switches* e controlam motores, outras plataformas, relés ou outros tipos de saída (RODRIGO, 2011).

Pelo fato de ser um hardware e software livre, existem diversos tipos de placas no mercado com seus esquemáticos eletrônicos disponíveis, caso o consumidor que deseja criar a própria placa possa construir ou mesmo modificar (ARDUINO,2014). A base de desenvolvimento Arduino®, é um sistema de computação físico que fornece ao usuário, diversos pinos de entrada ou saída, digitais e analógicos, que podem ser definidos na programação do microcontrolador utilizando o Arduino IDE.

No Arduino® é possível criar rotina para Windows, Mac OS X e Linux, pois o seu ambiente de desenvolvimento é multiplataforma. A linguagem aceita pelo Arduino IDE é muito parecida com a linguagem C++, porém mantém uma forma simplificada.

2.5 SHIELDS

2.5.1 LCD

Os módulos LCDs são interfaces de saída muito úteis em sistemas microcontrolados. Estes módulos podem ser gráficos ou a caractere (alfanuméricos). Os LCDs comuns, são especificados em número de linhas por colunas, sendo mais usuais as apresentações 16×2, 16×1, 20×2, 20×4, 8×2. Além disso, os módulos podem ser encontrados com luz de fundo (LEDs para iluminação de fundo), facilitando a leitura em ambientes escuros. Os LCDs mais comuns empregam o circuito integrado controlador HD44780 da Hitachi com interface paralela. Há no mercado também LCDs com controle serial, sendo que novos LCDs são constantemente criados. Na Figura 4 é apresentado um circuito de controle de um display 16x4.



Figura 4: Display LCD 16x4.

Fonte: <https://www.embarcados.com.br/modulo-de-display-lcd/>

2.5.2 MOTOR DE CORRENTE CONTÍNUA

Motor de corrente contínua é um motor elétrico alimentado por corrente contínua, que pode ser fornecida por bateria ou qualquer outra alimentação. A comutação pode ser através de escovas ou sem escovas. Em motores CC a velocidade pode ser controlada apenas variando a tensão, diferente dos motores de corrente alternada em que a velocidade é controlada pela frequência.

Os motores de corrente contínua são constituídos por um circuito indutor, circuito induzido e circuito magnético. Sendo constituído por elementos fixos e moveis, dá-se o nome de estator à parte fixa do motor e o nome rotor à parte móvel do mesmo. No caso do motor CC o circuito indutor encontra-se no estator e o circuito induzido no rotor. O circuito induzido é constituído por um enrolamento envolvendo um núcleo ferromagnético laminado, isto é, dividido em chapas entre si. Como pode ser visto na Figura 5.



Figura 5: Motor de corrente contínua.

Fonte: <https://www.mundodaeletrica.com.br/>

2.6 PONTE H

Ponte H é um circuito eletrônico do tipo *chopper* de classe E (um *chopper* de classe E converte uma fonte fixa de corrente contínua fixa em uma tensão de corrente contínua variável abrindo e fechando as chaves diversas vezes).

A ponte H tem como principal função o controle de velocidade e sentido de motores de corrente contínua, podendo também ser usado para controle da saída de um gerador CC ou como inversor monofásico. O termo Ponte H, é derivado da representação gráfica típica deste circuito, como mostrado na Figura 6.

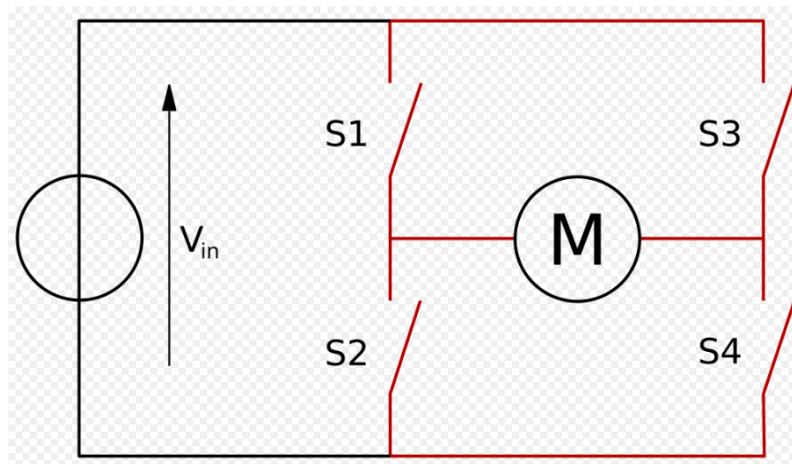


Figura 6: Diagrama de um circuito "Ponte H".

Fonte: https://pt.wikipedia.org/wiki/Ponte_H

Acionando-se em conjunto, as chaves S1 e S4, o terminal direito do motor fica com uma tensão mais positiva que o esquerdo, fazendo a corrente fluir da direita para a esquerda. Deste modo, o motor adquire sentido de giro que denotaremos por Sentido 1.

Acionando-se em conjunto as chaves S3 e S2, o terminal esquerdo do motor fica com uma tensão mais positiva que o direito, fazendo a corrente fluir da esquerda para a direita (no desenho, corrente convencional). Deste modo, o motor adquire sentido de giro que denotaremos por Sentido 2, que é inverso ao Sentido 1.

Ao acionar em conjunto as chaves S1 e S3 ou S2 e S4 provocamos um curto nos terminais do motor. Isso é necessário quando deseja-se frear um motor já em movimento ou aumentar a dificuldade de giro do eixo por um fator externo.

Um dos circuitos integrados mais utilizados, baseado em ponte H, é o L293D que contém duas pontes H completas. Com ele, existe a possibilidade de controlar dois motores DC bidirecionais ou quatro solenóides. O circuito integrado L293D pode ser visto na Figura 7.

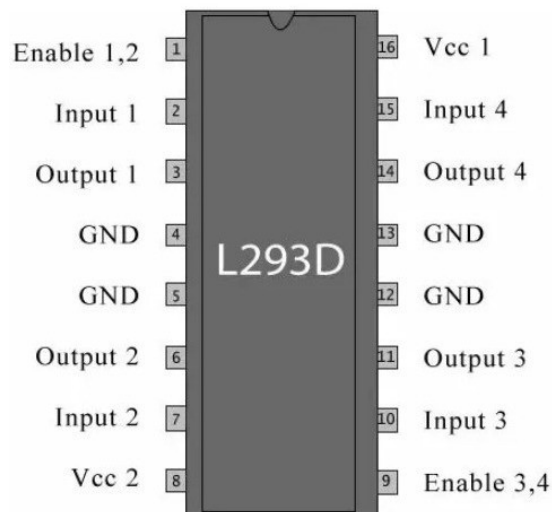


Figura 7: Circuito Integrado L293D.

Fonte: <https://produto.mercadolivre.com.br/>

2.7 CONTROLE DE PROCESSOS

2.7.1 CONTROLE DE MALHA FECHADA VS MALHA ABERTA

Sistemas de controle com realimentação é um sistema que estabeleça uma relação de comparação entre a saída e a entrada de referência, utilizando a diferença como meio de controle, é denominado sistema de controle com realimentação. Um exemplo poderia ser o sistema de controle de temperatura de um ambiente. Medindo-se a temperatura ambiente real e comparando-a com a temperatura de referência (temperatura desejada), o termostato ativa ou desativa o equipamento de aquecimento ou resfriamento, de modo que assegure que a temperatura ambiente permaneça em um nível confortável, independentemente das condições exteriores (OGATA, 2010).

Os sistemas de controle com realimentação não estão limitados à engenharia, podendo ser encontrados em várias outras áreas. O corpo humano, por exemplo, é um sistema de controle com realimentação extremamente desenvolvido. Tanto a temperatura corporal como a pressão sanguínea são mantidas constantes por meio da realimentação de ordem fisiológica. Nesse caso, a realimentação realiza uma função vital: faz que o corpo humano seja relativamente insensível a perturbações externas, permitindo seu perfeito funcionamento nos casos de mudanças no ambiente (OGATA, 2010). Um exemplo de sistema em malha fechada pode ser visto na Figura 8.

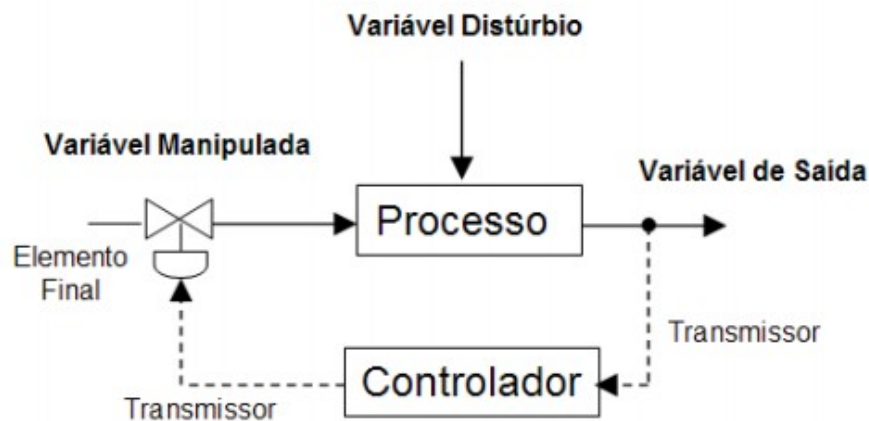


Figura 8: Exemplo de sistema em malha fechada.

Fonte: <http://monografias.ufrn.br:8080/>

Já os sistemas de controle de malha aberta são aqueles em que o sinal de saída não exerce nenhuma ação de controle no sistema. Isso quer dizer que, em um sistema de controle de malha aberta, o sinal de saída não é medido nem realimentado para comparação com a entrada. Um exemplo prático é o da máquina de lavar roupas. As operações de colocar de molho, lavar e enxaguar em uma lavadora são executadas em uma sequência baseada em tempo. A lavadora não mede o sinal de saída, isto é, não verifica se as roupas estão bem lavadas.

Em qualquer sistema de controle de malha aberta, a saída não é comparada com a entrada de referência. Assim, a cada entrada de referência corresponde uma condição fixa de operação. Dessa maneira, a precisão do sistema depende de uma calibração. Na presença de distúrbios, um sistema de controle de malha aberta não vai executar a tarefa desejada. Na prática, o sistema de controle de malha aberta somente poderá ser utilizado se a relação entre a entrada e a saída for conhecida e se não houver nenhum distúrbio interno ou externo. É claro que estes não são sistemas de controle realimentados. Observe que qualquer sistema de controle cujas operações são efetuadas em uma sequência baseada em tempo é um sistema de malha aberta. O controle de tráfego por meio de sinais, operado em função do tempo, é outro exemplo de controle de malha aberta (OGATA, 1995). Um exemplo de sistema em malha aberta pode ser visto na Figura 9.

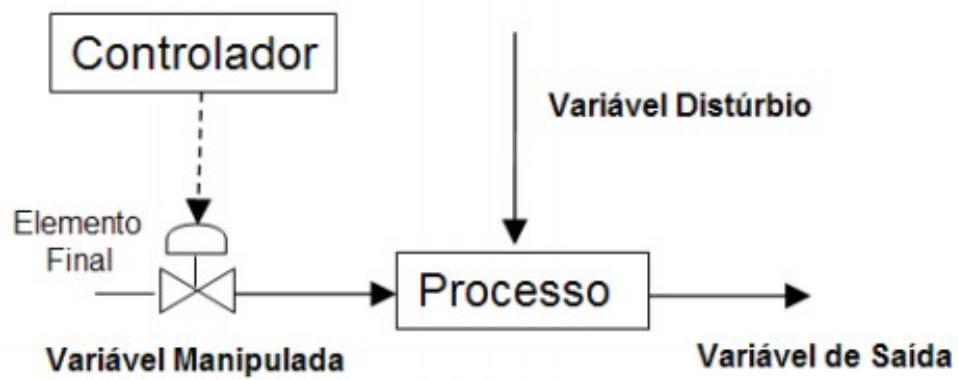


Figura 9: Exemplo de sistema em malha aberta.

Fonte: <http://monografias.ufrn.br:8080/>

CAPÍTULO III

METODOLOGIA

Em um projeto de sistemas embarcados, é necessário que haja medições precisas a respeito de tensões aplicadas em determinados terminais. Isto deve ser considerada pelo fato dos componentes envolvidos serem em boa parte de baixa potência e níveis de tensão elevados podem prejudicar seu funcionamento permanentemente caso ocorram, por exemplo, alguns erros de projeto ou de montagem. Portanto, além do cuidado com controle, é preciso que os *datasheets* dos dispositivos utilizados sejam cuidadosamente analisados, para que o sistema todo trabalhe da melhor maneira e evite-se prováveis erros.

3.1 AVALIAÇÃO DO SISTEMAS

A princípio foi realizado a avaliação do sistema proposto. Por se tratar de um sistema já implementado em estudos anteriores, o mesmo foi analisado afim de verificar se poderia ser aproveitado, e o que poderia ser substituído. Com isso, foram realizados os seguintes procedimentos:

- Teste com diferentes motores;
- Medição das dimensões da estrutura base para fixação do sistema;
- Teste com diferentes placas de prototipagem;
- Avaliação da melhor forma de indicar a frequência;
- Avaliar diferentes formas de realizar o controle do motor de corrente contínua.

3.1.1 MONTAGEM DO CIRCUITO

Após a análise do sistema proposto, tendo em vista os componentes que melhor se adequam ao projeto, foi realizado a montagem do circuito de controle para o motor de corrente contínua. No caso, foi utilizado o software Fritzing®, que permitiu criar o circuito, como visto na Figura 10.

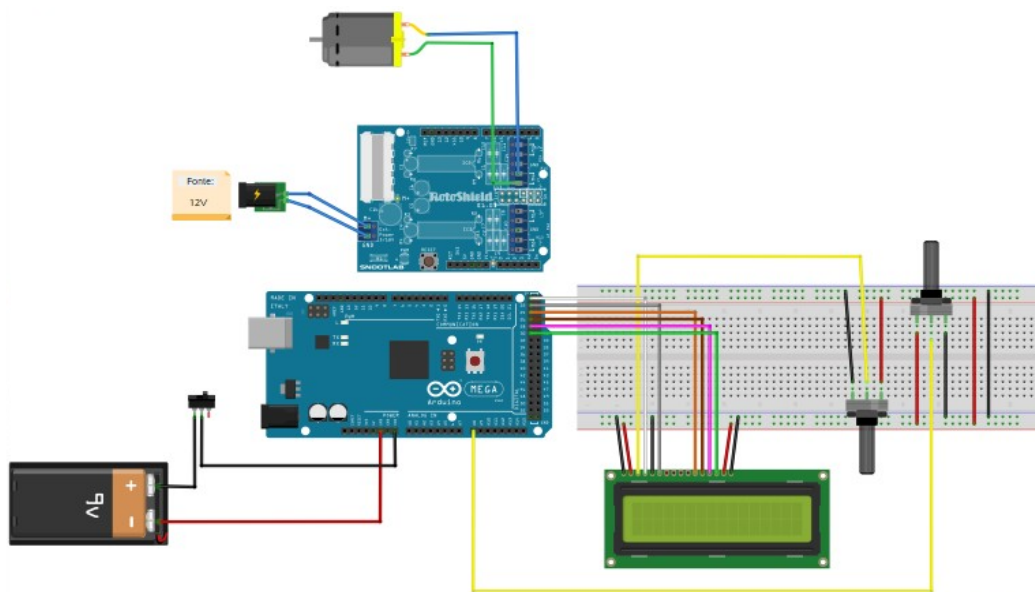


Figura 10: Montagem do circuito.

3.1.2 INSTALAÇÃO DO CIRCUITO NA BASE

Com a montagem do circuito, foi possível realizar a instalação do mesmo na base, constituída por uma calha com em perfil U confeccionada em aço carbono similar ao mostrado na Figura 11, com uma haste de sustentação para para a tocha de soldagem, como visto na Figura 12.



Figure 11: Perfil em U.

Fonte: <http://www.portoacoeferro.com.br/>

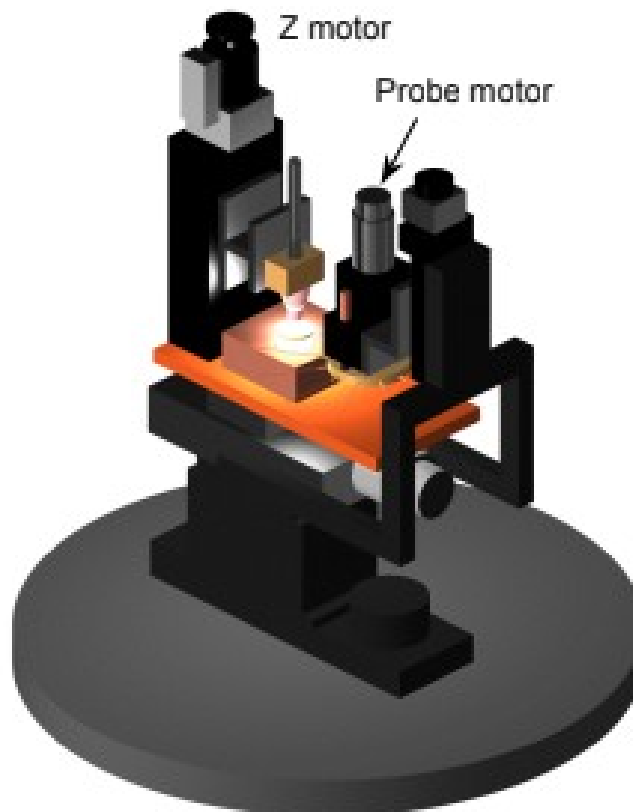


Figure 12: Modelo de estrutura de sustentação para o controlador do motor.

Fonte: Por Prof. Dr Louriel Oliveira Vilarinho

3.1.3 IMPLIMENTAÇÃO DO CÓDIGO

A implementação do código para o controle do motor foi realizada por meio compilador Arduino IDE, utilizando linguagem de programação C. Maiores informações sobre o código estão presentes no próximo capítulo.

Tendo em vista que o código foi gravado na placa, foi realizada a montagem completa do dispositivo proposto inicialmente. Por fim, sendo necessário os testes para a constatação do funcionamento do mesmo.

CAPÍTULO IV

DESCRIÇÃO DO PROJETO

4.1 HARDWARE

Para a execução do projeto elaborado, foram necessários diferentes módulos que operam em conjunto. Cada módulo executa diferentes funções, como fornecer corrente para o motor, servir de estrutura mecânica para o sistema, realizar processamento, informar a frequência do motor, entre outros.

4.1.1 MATERIAIS UTILIZADOS

Plataforma contendo:

- Um motor CC de 9V;
- Estrutura de aço-carbono com perfil em U;
- Haste para a fixação da tocha;
- Arduino® Mega 2560;
- Display de LCD 16x4;
- Uma bateria de 9V;
- Chave gangorra;
- Motor Shield L293D;
- Dois potenciômetros de 10K;

4.1.2 DESCRIÇÃO DO HARDWARE

O circuito de controle foi montado em uma estrutura no qual o eixo do motor possa movimentar as sondas, como visto na Figura 13, a frequência de rotação do eixo dependerá exclusivamente do PWM proveniente do posicionamento de um do potenciômetro, como visto na Figuras 14.



Figura 13: Estrutura montada.



Figura 14: Protótipo do sistema proposto.

O circuito ilustrado na Figura 15 consiste em uma placa de prototipagem do modelo Arduino® Mega 2560 R3, esta placa foi escolhida devido a facilidade na programação e por conter I/O's suficientes para as aplicações do projeto.

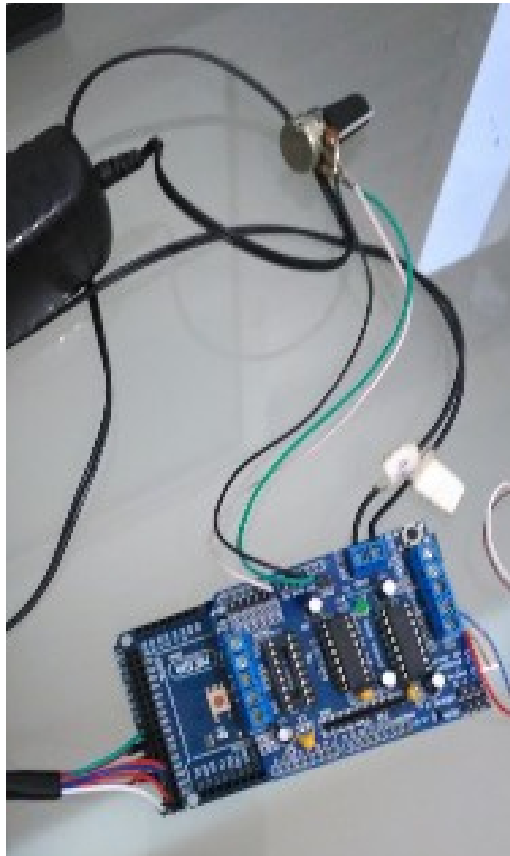


Figure 15: Placa de Prototipagem , e controlador.

O shield L293D, no qual funciona com base na teoria de ponte H, e é acoplado na placa de prototipagem, já o motor é ligado ao shield que é alimentado por uma fonte de 12 V, mostrada na Figura 16. Vale ressaltar que o shield e tem alimentação independente do arduino, o segundo tem uma alimentação de uma bateria de 9 V, que também realiza a alimentação do display de LCD 16x4. Um dos potenciômetros realiza o controle do PWM para o controle do motor, já o outro faz controle da iluminação do display de LCD 16x4.

4.2 SOFTWARE

O programa desenvolvido foi gravado no Arduino® Mega 2560, por meio do Arduino® IDE 1.0.5-r2, portanto esse será o responsável por realizar o processamento dos dados recebidos e enviados. As operações desempenhadas pela plataforma está diretamente associada com os softwares utilizados para elaboração do programa principal e também de configurações.

4.2.1 DESCRIÇÃO DO PROGRAMA

A seguir é indicado basicamente como o programa funciona. Inicialmente é realizado a criação das variáveis e objetos globais, assim como a inserção das bibliotecas utilizadas no *sketch*, como se trata de um sistema embarcado é necessário inicializar os pinos que serão utilizados pelos componentes e *shields*. Na sequência é realizado a criação dos blocos, o primeiro a ser criado é o *setup()*, onde são inicializadas as funções necessárias de cada um dos *shields*. O próximo bloco se trata do *delayPWM()*, responsável por fazer controle por modulação de pulso do motor de corrente contínua, a frequência de giro do eixo depende da modulação selecionada através do potenciômetro, como mostrado na Figura 16.

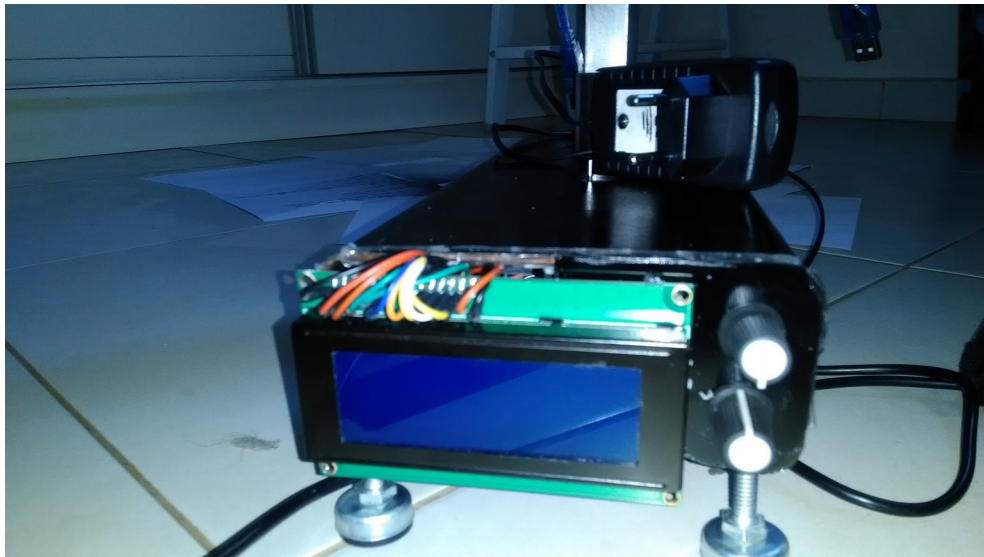


Figura 16: Potenciômetro para controle do motor.

Na subrotina (*loop*) é onde o programa principal se instala, descrito no fluxograma, nela é feita a chamada das variáveis dos pinos de alimentação do motor de acordo com o *PWM* mapeado no bloco *delayPWM()*. A frequência no qual o motor vai girar depende da resolução de oito bits do próprio arduino, dessa forma é feito um laço de *if-else* com os intervalos inseridos, que estão entre 0 e 255.

Já o último bloco do funcionamento do CI 74HC595, nele é realizada a inicialização, a preparação da transmissão e recepção de dados e seleciona o clock de trabalho da shield L293d acoplada ao Arduino®. O código pode ser observado nos anexos.

4.2.2 LINGUAGEM DE PROGRAMAÇÃO

Utilizando a linguagem do Arduino®, um programa típico é elaborado a partir de duas funções de estrutura principais: o "*void setup()*" e o "*void loop()*". A função "*void setup()*" é executada toda vez que a plataforma é ligada ou reiniciada. Esta função é utilizada para definir se os pinos definidos previamente são de entrada e saída, iniciar variáveis, definir bibliotecas, entre outras.

A outra função utilizada é o "*void loop()*" que, na maioria dos casos, é onde o programa principal se instala. Consiste em uma rotina de loop que executa o código elaborado de forma repetida no microcontrolador. Dentro da "*void loop()*", neste caso, são chamadas outras subrotinas que executam determinada parte do programa. Isto ocorre para manter uma organização dentro do código para que possa ser compreendido por outras pessoas que não o elaboraram e em caso de modificações, fique mais fácil de alterar a estrutura e parâmetros.

4.2.3 ARDUINO IDE “Integrated Development Environment”

O ambiente de desenvolvimento é composto por uma barra de ferramentas superior, com funções como *Verify*, *Upload*, *New*, *Open*, *Save* e *Serial Monitor*, mostrado na Figura 17. A opção *Verify*, procura erros de sintaxe no programa apresentado. A compilação e transferência para o microcontrolador são feitas pelo *Upload*. *New* cria uma nova aba em branco. *Open* abre um programa já salvo. *Save* grava o programa escrito com um nome e extensão “.ino” e a opção *Serial Monitor* é utilizada para fazer a comunicação com o Arduino® pela porta serial, além de monitorar dados que podem ser adquiridos com a placa.

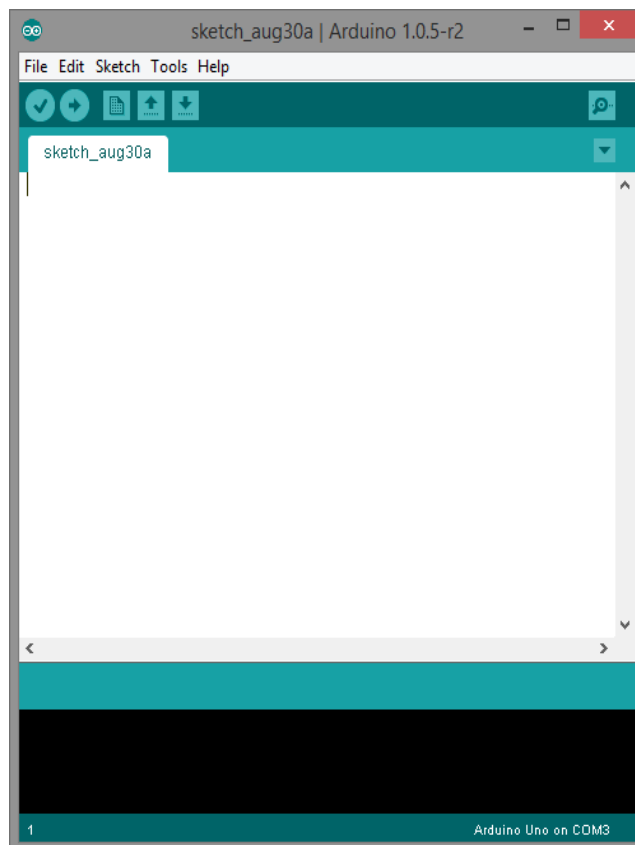


Figura 17: Interface do Arduino IDE.

Fonte: <https://www.intel.com.br/>

No trabalho, a utilização do *Serial Monitor* se mostrou prática, pelo fato do programa elaborado poder ser facilmente adaptado e testado, otimizando o processo metodológico. Através dele, foi possível realizar diversos testes de reconhecimento de erro, pela impressão de comandos estrategicamente inseridos no código.

4.2.4 Arduino MEGA 2560

O Arduino® Mega 2560, Figura 18, é uma placa microcontrolada de prototipagem, elaborada para trabalhar com o microcontrolador Atmega328. O ambiente de desenvolvimento possui 54 pinos de input ou output digitais (sendo que 6 deles podem ser usados para saída PWM), 16 entradas analógicas, 1 resonador cerâmico de 16MHz, conexão USB, conexão de energia e um botão de *reset*. A placa contém o necessário para dar suporte ao Atmega328, basta conectar o cabo USB ao computador para iniciar a programação através do IDE (ARDUINO, 2014).

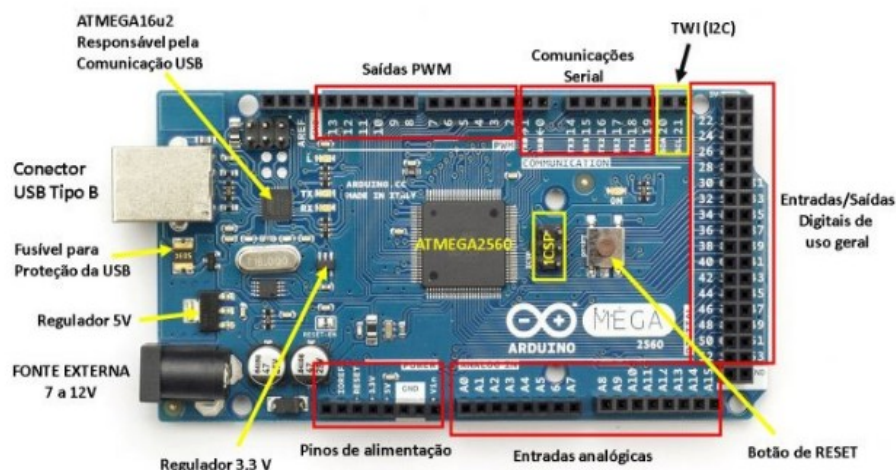


Figura 18: Arduino Mega 2560 R3.

Fonte: <https://www.4hobby.com.br/>

A Tabela 1 indica informações pertinentes a respeito da placa de desenvolvimento.

Tabela 1 – Características do Arduino Mega 2560 R3.

Características	Descrição
Microcontrolador	ATmega328
Tensão de operação	5 V
Tensão de entrada	7-12 V
Pinos de I/O digitais	54
Pinos de entrada analógica	16
Corrente DC por pino I/O	40 mA
Corrente DC por pino 3.3V	50 mA
Memória Flash	256 KB
Clock	16 MHz
SRAM	8 KB
EEPROM	4 KB

4.3 SHIELD L293D

O sistema embarcado Arduino® apresenta como principal componente o microcontrolador Atmega 328P. Segundo o datasheet do fabricante, este pode drenar cargas de saída de até 40 mA, o que torna inviável o controle direto da maioria dos motores DC. Por este motivo, torna-se necessária a utilização de driver para esta finalidade. Existem muitos Shields no mercado, prontos para utilização com Arduino, um deles é o Motor Shield L293D, que permite o controle de até 4 motores DC. O hardware

deste Shield consiste basicamente em dois circuitos integrados L293D (ponte H dupla com diodos de proteção internos) e um registrador de deslocamento da família TTL, o 74HC595, mostrado na Figura 19.

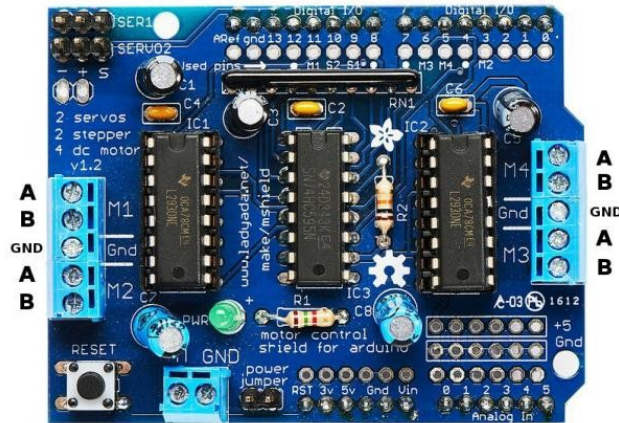


Figura 19: Shield L293D.

Fonte: <https://www.usinainfo.com.br/>

CAPÍTULO V

CONCLUSÃO

Ao fim do projeto, o protótipo montado funcionou conforme a proposta inicial, além da possibilidade de adquirir conhecimentos na área de programação, hardware e microcontroladores.

Como o desenvolvimento do projeto levou cerca de um ano, houve tempo suficiente para a realização da montagem de outros protótipos. Com isso, inúmeras características foram modificadas e melhoradas com o passar do tempo.

Entre as versões, vale ressaltar a questão do *display* utilizado para a indicação da frequência de rotação do eixo do motor. Inicialmente optou-se pela utilização de displays de 7 segmentos o que limitou a indicação dos dados, dessa forma foi realizado a substituição do mesmo para um display de LCD 16x4.

Outro ponto modificado entre as versões, foi a utilização de uma *shield* baseada no CI L293D (ponte H), o que excluiu a necessidade da confecção de uma placa impressa baseada no mesmo CI.

Tendo em vista que o projeto gerou resultados satisfatórios, é possível a realização das próximas etapas do projeto global, que englobam a realização da aquisição dos dados e os devidos estudos sobre o comportamento das zonas do arco elétrico (Zona anódica, catódica e coluna).

CAPÍTULO VI

TRABALHOS FUTUROS

Durante a execução dos testes finais, foram notados pontos com potencial para solução em trabalhos futuros. Sendo eles: a realização do controle em malha fechada do motor de corrente contínua, afim de ter maior confiabilidade dos dados coletados pelas sondas, devido a menor incerteza quando a velocidade do eixo do motor e a possibilidade de monitoramento da posição do mesmo.

Para isso, existe a necessidade da instalação de um encoder no motor, possibilitando a modelagem matemática por meio de função transferência, ou mesmo estados de espaço, que são técnicas de controle de processos mais confiáveis que a utilização de controle de malha aberta.

Além dos pontos citados anteriormente, existe a possibilidade da realização do controle do motor por meio de comunicação wireless (Bluetooth ou Wi-Fi), utilizando dispositivos móveis como celulares e computadores. Para a utilização deste método, é indicado shields como o módulo Bluetooth Hc-05 ou mesmo Wi-Fi ESP8266.

REFERÊNCIAS BIBLIOGRÁFICAS

- [1] VILARINHO, L. O. Characterisation of TIG Arcs Structures Using Experimental Techniques. 325p. 2002.
- [2] <https://www.filipeflop.com/blog/controle-motor-dc-arduino-motor-shield/>. Acessado em 07/05/2019.
- [3] RIBEIRO, A. M. Instrumentação. 360p. 2002.
- [4] JÚNIOR, P. A. Eletrônica Analógica - Amplificadores Operacionais e Filtros Ativos. 308p. 2003.
- [5] FRONIUS. Treinamento Expert CMT. 112p. 2011.
- [6] BRAGA, I. N. *Conheça as pontes H*. 2009. Acesso em 10/05/2019. Disponível em: <<http://www.newtoncbraga.com.br/index.php/robotica/5166-mec068a>>.
- [7] BRAGA, I. N. *O que é PWM*. 2009. Acesso em 10/05/2019. Disponível em: <<http://www.newtoncbraga.com.br/index.php/robotica/5169-mec071a>>.
- [8] ARDUINO. *Site Oficial*. 2014. Acesso em 15/05/2019. Disponível em: <<http://www.arduino.cc>>.
- [9] OGATA, K. Discrete-Time Control Systems. 2a edição. ed. [S.I.]: PEARSON PRENTICE HALL, 1995.
- [10] OGATA, K. Engenharia de Controle Moderno. 5a edição. ed. [S.I.]: PEARSON PRENTICE HALL, 2010.
- [11] ATMEL. ATmega328. 2017. Disponível em: <<https://www.microchip.com/>>.
- [12] DORF, R. C.; BISHOP, R. H. MODERN CONTROL SYSTEMS. 11a edição. ed. [S.I.]: PEARSON PRENTICE HALL, 2008.
- [13] CUNHA, A. F. O que são sistemas embarcados. *Saber Eletrônica-Ano*, 2007.
- [14] <https://www.filipeflop.com/blog/controle-motor-dc-arduino-motor-shield/>. Acessado em 10/06/2019.

- [15] <https://www.filipeflop.com/blog/motor-dc-arduino-ponte-h-l298n/>. Acessado em 10/06/2019.
- [16] THOMAZINI, D.; ALBUQUERQUE, B. U. P. Sensores Industriais - Fundamentos e Aplicações. 222p. 2011.
- [17] RODRIGO, L. Plataforma Arduino, Hardware e Visão Geral. 2011. Acessado em 10/06/2019 Disponível em: <<http://leandro-robotica.blogspot.com/>>.
- [18] http://www.fem.unicamp.br/~em621/aulas/aula1/ma_mf_real_hist.pdf. Acessado em 11/06/2019
- [19] http://monografias.ufrn.br:8080/jspui/bitstream/123456789/2332/3/Implementacaodecontrole_Monografia.pdf. Acessado em 11/06/2019
- [20] WERNECK, Pedro. Introdução ao Arduino. 2009. Disponível em: <<http://www.abereletronica.com.br/secoes/leitura/1307>>. Acesso em: 15/06/2019

APÊNDICE A

SKETCH DO CONTROLADOR

```
/*
 * Mapa do Sketch
 * \1- Cabeçalho (bibliotecas, definições, variáveis e objetos globais)
 * 2- setup()
 * 3- loop()
 * 4- ci74HC595Write()
 * 5- delayPWM()
 *
 */

//Carrega a biblioteca LiquidCrystal
#include <LiquidCrystal.h>

//-----
#define pinSH_CP 4 //Pino Clock DIR_CLK
#define pinST_CP 12 //Pino Latch DIR_LATCH
#define pinDS 8 //Pino Data DIR_SER
#define pinEnable 7 //Pino Enable DIR_EN

//-----
#define pinMotor1PWM 11

//-----
#define qtdeCI 1

//-----
#define bitMotor1A 2
#define bitMotor1B 3

//Define os pinos que serão utilizados para ligação ao display
```

```

LiquidCrystal lcd(18, 19, 20, 21, 22, 23);
byte valA0;

//-----
void ci74HC595Write(byte pino, bool estado);

//-----
void setup() {
    pinMode(pinSH_CP, OUTPUT);
    pinMode(pinST_CP, OUTPUT);
    pinMode(pinEnable, OUTPUT);
    pinMode(pinDS, OUTPUT);

    pinMode(pinMotor1PWM, OUTPUT);
    //Define o número de colunas e linhas do LCD
    lcd.begin(16, 4);

    digitalWrite(pinEnable, LOW);
    Serial.begin(9600);
}

//-----
void delayPWM(unsigned long tempo) {
    unsigned long inicio = millis();

    while ( (millis() - inicio) < tempo ) {
        valA0 = map(analogRead(A0), 0, 1023, 0, 255);
        analogWrite(pinMotor1PWM, valA0);

    }
}

//-----
void loop() {

```

```

float i = valA0;

//Serial.println("Motores A=HIGH B=LOW");
ci74HC595Write(bitMotor1A, HIGH);
ci74HC595Write(bitMotor1B, LOW);

pinMotor1PWM.run(FORWARD);
delayPWM(5000);
pinMotor1PWM.run(BACKWARD);
delayPWM(5000);
pinMotor1PWM.run(RELEASE);

if (valA0 <= 26){
    Serial.println("Frequência: 0 Hz ");
    //Limpa a tela
    lcd.clear();
    //Posiciona o cursor na coluna 3, linha 0;
    lcd.setCursor(4, 2);
    lcd.print(" Frequencia = 0 Hz ");
}

if ( valA0 >= 26 && valA0 < 51 ){
    Serial.println("Frequência: 0,5 Hz ");
    //Limpa a tela
    lcd.clear();
    //Posiciona o cursor na coluna 3, linha 0;
    lcd.setCursor(4, 2);
    lcd.print(" Frequencia = 0,5 Hz ");
}

if ( valA0 >= 51 && valA0 < 77 ){
    Serial.println("Frequência: 1 Hz ");
}

```

```

//Limpa a tela
lcd.clear();
//Posiciona o cursor na coluna 3, linha 0;
lcd.setCursor(4, 2);
lcd.print(" Frequencia = 1 Hz ");

}

if ( valA0 >= 77 && valA0 < 102 ){
    Serial.print("Frequência: 1,5 Hz ");
    //Limpa a tela
    lcd.clear();
    //Posiciona o cursor na coluna 3, linha 0;
    lcd.setCursor(4, 2);
    lcd.print(" Frequencia = 1,5 Hz ");

}

if (valA0 >= 102 && valA0 < 128){
    Serial.print("Frequência: 2,0 Hz ");
    //Limpa a tela
    lcd.clear();
    //Posiciona o cursor na coluna 3, linha 0;
    lcd.setCursor(4, 2);
    lcd.print(" Frequencia = 2,0 Hz ");

}

if (valA0 >= 128 && valA0 < 153){
    Serial.print("Frequência: 2,5 Hz ");
    //Limpa a tela
    lcd.clear();
    //Posiciona o cursor na coluna 3, linha 0;
    lcd.setCursor(4, 2);
    lcd.print(" Frequencia = 2,5 Hz ");

}

```

```

if (valA0 >= 153 && valA0 < 179){
    Serial.print("Frequência: 3,0 Hz ");
    //Limpa a tela
    lcd.clear();
    //Posiciona o cursor na coluna 3, linha 0;
    lcd.setCursor(4, 2);
    lcd.print(" Frequencia = 3,0 Hz ");

}

if (valA0 >= 179 && valA0 < 204){
    Serial.print("Frequência: 3,5 Hz ");
    //Limpa a tela
    lcd.clear();
    //Posiciona o cursor na coluna 3, linha 0;
    lcd.setCursor(4, 2);
    lcd.print(" Frequencia = 3,5 Hz ");

}

if (valA0 >= 204 && valA0 < 230){
    Serial.print("Frequência: 4,0 Hz ");
    //Limpa a tela
    lcd.clear();
    //Posiciona o cursor na coluna 3, linha 0;
    lcd.setCursor(4, 2);
    lcd.print(" Frequencia = 4,0 Hz ");

}

if (valA0 >= 230 && valA0 < 255){
    Serial.print("Frequência: 4,5 Hz ");
    //Limpa a tela
    lcd.clear();
    //Posiciona o cursor na coluna 3, linha 0;
    lcd.setCursor(4, 2);
    lcd.print(" Frequencia = 4,5 Hz ");
}

```

```

    }
}

//-----
void ci74HC595Write(byte pino, bool estado) {
    static byte ciBuffer[qtdeCI];

    bitWrite(ciBuffer[pino / 8], pino % 8, estado);

    digitalWrite(pinST_CP, LOW); //Inicia a Transmissão
    digitalWrite(pinDS, LOW);    //Apaga Tudo para Preparar Transmissão
    digitalWrite(pinSH_CP, LOW);

    for (int nC = qtdeCI-1; nC >= 0; nC--) {
        for (int nB = 7; nB >= 0; nB--) {

            digitalWrite(pinSH_CP, LOW); //Baixa o Clock
            digitalWrite(pinDS, bitRead(ciBuffer[nC], nB) );    //Escreve o BIT
            digitalWrite(pinSH_CP, HIGH); //Eleva o Clock
            digitalWrite(pinDS, LOW);    //Baixa o Data para Prevenir Vazamento
        }
    }

    digitalWrite(pinST_CP, HIGH); //Finaliza a Transmissão
}

```