PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA
ENGENHARIA DE COMPUTAÇÃO

PROJETO METAL COLLECTOR PLUS

CURITIBA
2009

FERNANDO P. GOMES
JOÃO HENRIQUE DESZAUNET
RAPHAEL RAMOS NOGUEIRA

PROJETO METAL COLLECTOR PLUS

Este projeto será apresentado às disciplinas do Curso de Engenharia de Computação do Centro de Ciências Exatas e de Tecnologia da Pontifícia Universidade Católica do Paraná, como parte integrante da nota do segundo semestre. A finalidade deste projeto é integração das diversas disciplinas do curso. Professores orientadores: Gil Marcos Jess e Afonso Ferreira Miguel.

CURITIBA
2009

SUMÁRIO

1. INTRODUÇÃO..... 04
1.1 JUSTIFICATIVA..... 05
1.2 METODOLOGIA..... 06
1.3 RESPONSABILIDADES..... 07
2. OBJETIVOS..... 08
3. NÃO ESTÁ INCLUSO NO ESCOPO DO PROJETO..... 09
4. EQUIPE DE DESENVOLVIMENTO..... 10
5. PROJETO..... 11
5.1 DIAGRAMA DE BLOCOS DO PROJETO..... 12
5.2 DIFICULDADES ENCONTRADAS..... 13
6. CONCLUSÃO..... 14
7. ANEXOS
7.1 DICIONÁRIO DE TERMOS TÉCNICOS..... 15
7.2 CÓDIGO FONTE DO SOFTWARE..... 29

1. Introdução

O projeto Metal Collector Plus consiste em uma plataforma com uma haste conectada a um eletroímã, o qual é capaz de separar metais ferrosos de metais não ferrosos ou não metais. A idéia surgiu de uma discussão entre os integrantes do grupo, onde todos concordaram em usar uma plataforma elevada com uma haste vinculada a estrutura contendo um eletroímã na ponta, a maior discussão foi qual seria a utilidade de projeto.

A primeira idéia foi um Jogo de Xadrez, onde as peças seriam movidas pelo eletroímã, mas por não ter uma grande atuação no cotidiano foi descartada, A segunda idéia foi um Guindaste de Containers , a qual por já existir uma grande tecnologia e estar em desenvolvimento muito mais avançados do que nossos conhecimentos foi brevemente descartada, até que surgiu uma idéia a qual seria muito útil para separação metais ferrosos, os quais podem ser reciclados e transformados em outros produtos. Como já dizia Lavoisier: “Na natureza nada se cria nada se perde tudo se transforma.”.

1.1. Justificativas

O projeto Metal Collector Plus visa a idéia de combinar reciclagem com tecnologia. Por se tratar de um coletor de metais, pode ser usado na coleta de metais ferrosos nos meios mais adversos, os quais não podem ser separados manualmente, como por exemplo em ferros velhos e lixões. O projeto também será capaz de separar metais como bronze, latão e ferro para que possa ser feita uma reciclagem precisa. Este sistema será inteiramente controlado por computador, cedendo uma maior facilidade ao usuário.

No mercado existem guindastes que executam essa função, mas por serem controlados manualmente por controles, não tem muita precisão. Por esta maneira os controles do projeto serão estabelecidos em computador, os quais terão uma exata precisão.

1.2. Metodologia

O projeto se constituiu em:

• Confecção dos circuitos e eletroímãs que constituirão o projeto.

- Para circuitos foi usado o software Eagle.
- Os eletroímãs foram feitos manualmente.

•Confecção da estrutura física (parte mecânica)

•Desenvolvimento do Software.

- O software foi desenvolvido em linguagem de programação C++, usando o compilador Visual Studio.

Ao decorrer do projeto, o grupo utilizou-se de alguns equipamentos para auxiliar na criação do projeto em si. Foram esses: osciloscópio, multímetro, fonte de alimentação, protoboard, prensa, ferro de solda, picstart, computador e notebook. Estes equipamentos foram essenciais para desenvolver e concluir nosso projeto.

Tais equipamentos em exceção do notebook eram de propriedade da PUC e podem ser utilizados por seus alunos gratuitamente. Utilizamos os equipamentos das seguintes maneiras:

• Osciloscópio: usamos para exibir os formatos de ondas gerados pelos componentes, para saber se havia alguma falha no circuito ou algum ruído indesejável, os quais podem causar problemas ao longo do projeto.

• Multímetro: utilizamos para medir as correntes, tensões, resistências, verificar continuidade ou descontinuidade entre as trilhas do circuito na placa, entre outros.

• Fonte de alimentação: esta foi utilizada para alimentar precisamente todo o circuito que produzimos e utilizamos desde as placas até os motores.

• Protoboard: para que não houvesse erros depois de prontas as placas, todos os circuitos foram inicialmente montados em protoboard par a que pudessem ser testados e/ou corrigidos.

• Prensa: utilizada para transferir o circuito impresso em folha de transparência para a placa de cobre na qual foi montado os circuitos.

• Ferro de solda: usado para soldar os componentes nas placas.

• PICSTART: utilizado para programar os PIC`s (microcontroladores) utilizados em nossas placas. Cada PIC foi programado importando um arquivo específico para cada função. Estes arquivos, fornecidos pelo professor Afonso Ferreira Miguel, estão no formato HEX (*.hex)* e são importados para dentro do microcontrolador através um programador d e PIC.

• Computador e notebook: utilizados para criar o software, programar os microcontroladores, desenhar os circuitos, entre outros. Esta ferramenta foi indispensável na maior parte de desenvolvimento do projeto.

1.3. Responsabilidades

A maior responsabilidade coube aos integrantes do grupo, os quais com muita determinação e pró-atividade respeitaram o cronograma, seguiram o foco do projeto e concluíram em tempo.

A responsabilidade dos professores coube a orientar quando preciso e administrar o andamento do projeto.

Pudemos contar com as estruturas da Pontifícia Universidade Católica do Paraná – PUCPR, sendo uma das responsabilidades essenciais, pois são nos laboratórios com os devidos equipamentos que conseguimos levar o projeto adiante.

2. Objetivos

O objetivo principal é proporcionar ao usuário manipulação do Metal Collector Plus através de um software no qual distância deverá ser percorrida no eixo horizontal e um vertical para que assim seja ativado um eletroímã o qual irá coletar metais ferrosos.

3. Não está incluso no escopo do projeto

Não foi nesse projeto:
- Câmeras;
- Joystick;
- Processo de reciclagem dos metais coletados;
- Controles sem fio;

4. Equipe de Desenvolvimento

A equipe de desenvolvimento contou com Fernando P. Gomes, João Henrique Deszaunet e Raphael Ramos Nogueira (líder da equipe). Em nosso cronograma decidimos as tarefas responsáveis por cada integrante da equipe. As tarefas para cada integrante foram distribuídas da seguinte maneira:

Fernando P. Gomes:

1. Confecção de Módulos;
2. Programação em C++;
3. Testes;

João Henrique Deszaunet:

1. Plano de trabalho;
2. Implementação da estrutura matricial;
3. Programação em C++;
4. Documentação do projeto;

Raphael Ramos Nogueira:

1. Confecção de placas;
2. Implementação da estrutura matricial;
3. Programação em C++;
4. Vídeo;

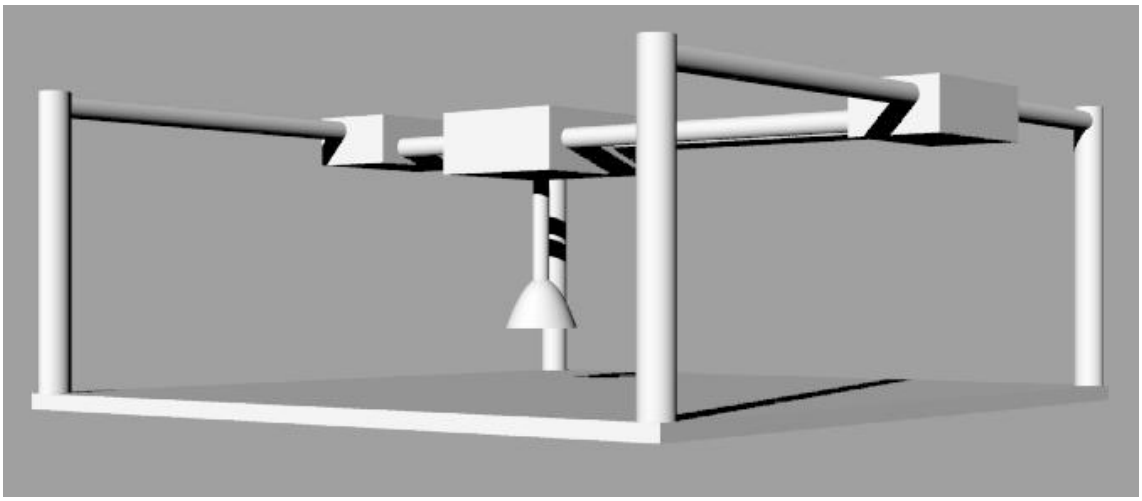
5. Projeto

O projeto Metal Collector Plus e constituído principalmente em três blocos: eletrônica, software e a estrutura.

A eletrônica se baseia em um conversor RS-232, que tem como função receber dados via a porta serial do computador e transformá-las em dados TTL para assim poder enviar comandos para os módulos de controle. Os módulos M0 recebem os dados TTL para controlar os motores de passo responsáveis pelo movimento da estrutura em X e Y, porém antes dessas informações do controle chegar aos motores essas passaram por etapas de potência. O módulo M2 recebe os dados para ativar o eletroímã, mas antes disso os dados passam por um amplificador de corrente e por uma etapa de potência assim, podendo ativar o eletroímã.

O software foi desenvolvido em linguagem C++ , foi usado o suporte a porta serial do Visual Studio 2008, contendo uma interface gráfica com teclas de controle para acionamento de motores, do eletroímã e da interface de comunicação serial.

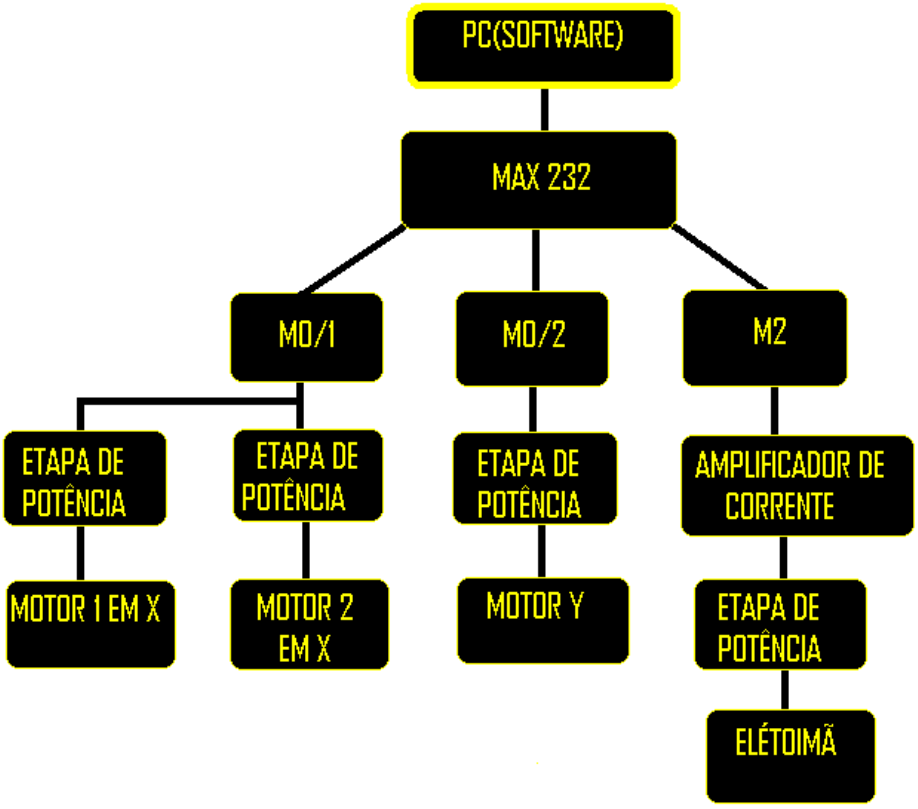
A estrutura foi montada com base na figura abaixo:



Para isso desmontamos três scanners para fazer a estrutura matricial. Para sustentar a estrutura mandamos fazer sob medida os quatro pilares laterais, após isso levamos em uma oficina a qual soldou as barras. Alinhamos e fixamos em um compensado a estrutura. Após isso levamos novamente para a oficina para soldar os motores de passo.

Depois disso foi só integrar as três partes e testar para que o projeto ficasse em perfeitas condições de funcionamento.

5.1 DIAGRAMA DE BLOCOS DO PROJETO



5.2 DIFICULDADES ENCONTRADAS

Ao longo do desenvolvimento do projeto nos deparamos com algumas dificuldades, as quais puderam ser superadas a tempo para que o projeto fosse concluídos com êxito, algumas dessas dificuldades foram:

Problemas de alinhamento na estrutura mecânica, começaram com a fixação dos 4 pinos de sustentação, mais foram facilmente resolvidos, depois o alinhamento da correia no motor Y, o qual apresentou um problema e teve q ser tocado, para solucionar o problema fizemos uma adaptação na estrutura fazendo uma nova base de massa(Durepox), a qual ficou firme alinhada.

As correias também foram um problema, por não apresentarem o tamanho em que precisávamos, assim tivemos que reduzir a correia em Y e aumentar as duas correias em X.

6. Conclusão

O Projeto Integrado tendo como objetivo principal a integração das diversas disciplinas do curso, relacionando assim teoria com prática. Pode-se dizer que o objetivo do mesmo foi alcançado com êxito no projeto. Foram desenvolvidas diversas habilidades durante o projeto, as quais ajudaram a contornar os problemas e superar as dificuldades. O projeto serviu para observarmos os vários conceitos passados em sala de aula na prática, pois o conceito passa apenas uma noção básica sobre o assunto, sendo inestimável o aprendizado na realização deste projeto, e dando novas idéias de planejamento e execução para os próximos projetos desenvolvidos.

Nesse projeto pudemos comprovar a frase dita por Flávio nosso professor de circuitos, “Se fizer certo não tem erro!” e fazendo certo conseguimos concluir o projeto com êxito.

7. ANEXOS

7.1 ANEXOS I – DICIONÁRIO DE TERMOS TÉCNICOS

Para facilitar a leitura para quem possa a vir a fazer leitura do mesmo, foi feito este dicionário básico contendo alguns dos termos usados neste projeto. Este dicionário serve apenas para dar uma noção básica sobre do que se tratam cada palavra em especifico.

CAPACITOR

Um capacitor ou condensador é um componente que armazena energia num campo elétrico, acumulando um desequilíbrio interno de carga elétrica. Os formatos típicos consistem em dois eletrodos ou placas que armazenam cargas opostas. Estas duas placas são condutoras e são separadas por um isolante ou por um dielétrico. A carga é armazenada na superfície das placas, no limite com o dielétrico. Devido ao fato de cada placa armazenar cargas iguais, porém opostas, a carga total no dispositivo é sempre zero.

CAPACITÂNCIA

A propriedade que estes dispositivos têm de armazenar energia elétrica sob a forma de um campo eletrostático é chamada de capacitância ou capacidade (C) e é medida pelo quociente da quantidade de carga (Q) armazenada pela diferença de potencial ou tensão (V) que existe entre as placas:

Pelo Sistema Internacional de Unidades (SI), um capacitor tem a capacitância de um Farad (F) quando um Coulomb de carga causa uma diferença de potencial de um volt (V) entre as placas. O farad é uma unidade de medida considerada muito grande para circuitos práticos, por isso, são utilizados valores de capacitâncias expressos em microFarads (μF), nanoFarads (nF) ou picoFarads (pF). A equação acima é exata somente para valores de Q muito maiores que a carga do elétron (e = 1,602 × 10-19 C). Por exemplo, se uma capacitância de 1 pF fosse carregada a uma tensão de 1 μV, a equação perderia uma carga Q = 10-19 C, mas isto seria impossível já que seria menor do que a carga em um único elétron.

Entretanto, as experiências e as teorias recentes sugerem a existência de cargas fracionárias.

A capacitância de uma capacitor de placas paralelas constituído de dois eletrodos planos idênticos de área A separados à distância constante d é aproximadamente igual a:

$$C = \epsilon_0 \epsilon_r \frac{A}{d}$$

Onde:

C é a capacitância em Farads
E0 é a permissividade eletrostática do vácuo ou espaço livre

ENERGIA

A energia (no SI, medida em Joules) armazenada em um capacitor é igual ao trabalho feito para carregá-lo. Considere um capacitor com capacitância C, com uma carga +q em uma placa e -q na outra. Movendo um pequeno elemento de carga dq de uma placa para a outra contra a diferença de potencial V = q/C necessita de um trabalho dW:

$$dW = \frac{q}{C}dq$$

Nós podemos descobrir a energia armazenada em um capacitor integrando essa equação. Começando com um capacitor descarregado (q=0) e movendo carga de uma placa para a outra até que as placas tenham carga +Q e -Q, necessita de um trabalho W:

$$W_{carregando} = \int_0^Q \frac{q}{C}dq = \frac{1}{2}\frac{Q^2}{C} = \frac{1}{2}CV^2 = E_{armazenada}$$

Capacitores Comuns

Apresenta-se com tolerâncias de 5 % ou 10 %.

Capacitores são freqüentemente classificados de acordo com o material usados como dielétrico. Os seguintes tipos de dielétricos são usados:

Cerâmica (valores baixos até cerca de 1 µF)
C0G or NP0 - tipicamente de 4,7 pF a 0,047 uF, 5 %. Alta tolerância e performance de temperatura. Maiores e mais caros X7R - tipicamente de 3300 pF a 0,33 uF, 10 %. Bom para acoplamento não-crítico, aplicações com timer.
Z5U - tipicamente de 0,01 uF a 2,2 uF, 20 %. Bom para aplicações em bypass ou acoplamentos. Baixo preço e tamanho pequeno.

Poliestireno (geralmente na escala de picofarads).

Poliéster (de aproximadamente 1 nF até 1000000 µF).

Polipropileno (baixa perda. alta tensão, resistente a avarias).

Tântalo (compacto, dispositivo de baixa tensão, de até 100 µF aproximadamente).

Eletrolítico (de alta potência, compacto mas com muita perda, na escala de 1 µF a 1000 µF)

Propriedades importantes dos capacitores, além de sua capacitância, são a máxima tensão de trabalho e a quantidade de energia perdida no dielétrico. Para capacitores de alta potência a corrente máxima e a Resistência em Série Equivalente (ESR) são considerações posteriores. Um ESR típico para a maioria dos capacitores está entre
0,0001 ohm e 0,01 ohm, valores baixos preferidos para aplicações de correntes altas.

Já que capacitores têm ESRs tão baixos, eles têm a capacidade de entregar correntes enormes em circuitos curtos, o que pode ser perigoso. Por segurança, todos os capacitores grandes deveriam ser descarregados antes do

manuseio. Isso é feito colocando-se um resistor pequeno de 1 ohm a 10 ohm nos terminais, isso é, criando um circuito entre os terminais, passando pelo resistor.

Capacitores também podem ser fabricados em aparelhos de circuitos integrados de semicondutores, usando linhas metálicas e isolantes num substrato. Tais capacitores são usados para armazenar sinais analógicos em filtros chaveados por capacitores, e para armazenar dados digitais em memória dinâmica de acesso aleatória (DRAM). Diferentemente de capacitores discretos, porém, na maior parte
do processo de fabricação, tolerâncias precisas não são possíveis (15 % a 20 % é considerado bom).

Resistor

Um resistor (chamado de resistência em alguns casos) é um dispositivo elétrico muito utilizado em eletrônica, com a finalidade de transformar energia elétrica em energia térmica (efeito joule), a partir do material empregado, que pode ser, por exemplo, carbono.

Um resistor ideal é um componente com uma resistência elétrica que permanece constante independentemente da tensão ou corrente elétrica que circular pelo dispositivo. Os resistores podem ser fixos ou variáveis. Neste caso são chamados de potenciômetros ou reostatos. O valor nominal é alterado ao girar um eixo ou deslizar uma alavanca.

O valor de um resistor de carbono pode ser facilmente determinado de acordo com as cores que apresenta na cápsula que envolve o material resistivo, ou então usando um ohmímetro.

Alguns resistores são longos e finos, com o material resistivo colocado ao centro, e um terminal de metal ligado em cada extremidade. Este tipo de encapsulamento é chamado de encapsulamento axial. A fotografia a direita mostra os resistores em uma tira geralmente usados para a pré formatação dos terminais.

Resistores usados em computadores e outros dispositivos são tipicamente muito menores, freqüentemente são utilizadas tecnologia de montagem superficial (Surface-mount technology), ou SMT, esse tipo de resistor não tem perna de metal. Resistores de potência maior são feitos mais robustos para dissipar calor de maneira mais eficiente, mas eles seguem basicamente a mesma estrutura.

Os resistores são sim como parte de um circuito elétrica e incorporada dentro de dispositivos microeletrônicos ou semicondutores. A medição crítica de um resistor é a resistência, que serve como relação de voltagem para corrente é medida em ohms, uma unidade SI. Um componente tem uma resistência de 1 ohm se uma voltagem de 1 volt no componente fazer com que percorra, pelo mesmo, uma corrente de 1 Ampère, o que é equivalente à circulação de 1 Coulomb de carga elétrica, aproximadamente 6.241506 x 1018 elétrons por segundo.

Qualquer objeto físico, de qualquer material é um tipo de resistor. A maioria dos metais são materiais condutores, e opõe baixa resistência ao fluxo de corrente elétrica. O corpo humano, um pedaço de plástico, ou mesmo o vácuo têm uma resistência que pode ser mensurada. Materiais que possuem

resistência muito alta são chamados isolantes ou isoladores A relação entre tensão, corrente e resistência, através de um objeto é dada por uma simples equação, Lei de Ohm:

Onde V é a voltagem em volts, I é a corrente que circula através de um objeto em Ampères, e R é a resistência em ohms. Se V e I tiverem uma relação linear -- isto é, R é constante -- ao longo de uma gama de valores, o material do objeto é chamado de ôhmico. Um resistor ideal tem uma resistência fixa ao longo de todas as freqüências e amplitudes de tensão e corrente.

Materiais supercondutores em temperaturas muito baixas têm resistência zero.

Isolantes (tais como ar, diamante, ou outros materiais não-condutores) podem ter

resistência extremamente alta (mas não infinita), mas falham e admitem que ocorra um grande fluxo de corrente sob voltagens suficientemente altas.

A resistência de um componente pode ser calculada pelas suas características físicas. A resistência é proporcional ao comprimento do resistor e à resistividade do material (uma propriedade do material), e inversamente proporcional à área da secção transversal. A equação para determinar a resistência de uma seção do material é:

Onde é a resistividade do material, é o comprimento, e é a área da secção transversal. Isso pode ser estendido a uma integral para áreas mais complexas, mas essa fórmula simples é aplicável a fios cilíndricos e à maioria dos condutores comuns. Esse valor está sujeito a mudanças em altas freqüências devido ao efeito skin, que diminui a superfície disponível da área.

Resistores padrões são vendidos com capacidades variando desde uns poucos miliôhms até cerca de um gigaôhms; apenas uma série limitada de valores, chamados valores preferenciais, estão isponíveis. Na prática, o componente discreto vendido como "resistor" não é um resistor perfeito como definido acima.

Resistores são freqüentemente marcados com sua tolerância (a variação máxima esperada da resistência marcada). Em resistores codificados com cores, uma faixa mais à direita demonstra uma tolerância de 10%, uma faixa dourada significa 5% de tolerância, uma faixa vermelha marca 2% e uma faixa marrom significa 1% de tolerância. Resistores com tolerância menores, também chamados de resistores de precisão, também estão disponíveis.

Um resistor tem uma voltagem e corrente máximas de trabalho, acima das quais a resistência pode mudar (drasticamente, em alguns casos) ou o resistor pode se danificar fisicamente (queimar, por exemplo). Embora alguns resistores tenham as taxas de voltagem e corrente especificadas, a maioria deles são taxadas em função de sua potência máxima, que é determinada pelo tamanho físico. As taxas mais comuns para resistores de composição de carbono e filme de metal são 1/8 watt, ¼ watt e 1/2 watt. Resistores de filme de metal são mais estáveis que os de carbono quanto a mudanças de temperatura e a idade. Resistores maiores são capazes de

dissipar mais calor por causa de sua área de superfície maior.

Resistores dos tipos wire-wound e sand-filled são usados quando se necessita de taxas grandes de potência, como 20 Watts.

Além disso, todos os resistores reais também introduzem alguma indutância e capacitância, que mudam o comportamento dinâmico do resistor da equação ideal.

CORRENTE ELÉTRICA

Na Física, corrente elétrica é o fluxo ordenado de partículas portadoras de carga elétrica. Sabe-se que, microscopicamente, as cargas livres estão em movimento aleatório devido a agitação térmica. Apesar desse movimento desordenado, ao estabelecermos um campo elétrico na região das cargas, verificase um movimento ordenado que se apresenta superposto ao primeiro. Esse movimento recebe o nome de movimento de deriva das cargas livres.

Raios são exemplos de corrente elétrica, bem como o vento solar, porém a mais conhecida, provavelmente, é a do fluxo de elétrons através de um condutor elétrico, geralmente metálico.

O símbolo convencional para representar a intensidade de corrente elétrica (ou seja, a quantidade de carga Q que flui por unidade de tempo t) é o I, original do alemão Intensität, que significa intensidade.

$$I = \frac{\Delta Q}{\Delta t}$$

A unidade padrão no SI para medida de intensidade de corrente é o ampère. A corrente elétrica é também chamada informalmente de amperagem. Embora seja um termo válido, alguns engenheiros repudiam o seu uso.

MICROCONTROLADOR

Um microcontrolador (também denominado MCU ou µC) é um computador num chip, contendo um processador, memória e funções de entrada/saída. É um microprocessador que enfatiza a alta integração, em contraste com os microprocessadores de uso geral (do tipo usado em computadores pessoais). Além dos componentes lógicos e aritméticos usuais dum microprocessador de uso geral, o microcontrolador integra elementos adicionais tais como memória RAM, EEPROM ou Memória flash para armazenamento de dados ou programas, dispositivos periféricos e interfaces de E/S que podem ir de um simples pino digital do componente a uma interface USB ou Ethernet nos mais avançados (como o ARM LPC2368).

Com freqüências de clock de poucos MHz ou ainda mais baixas microcontroladores são considerados lentos se comparados aos microprocessadores modernos, mas isso é perfeitamente adequado para aplicações típicas. Eles consomem relativamente pouca energia (miliwatts), e geralmente possuem a capacidade de "hibernar" enquanto aguardam que aconteça algum evento interessante provocado por um periférico, tal como o pressionar dum botão, que os colocam novamente em atividade. O consumo de energia enquanto estão "hibernando" pode ser de nanowatts, tornando-os ideais para aplicações de baixa energia e que economizem bateria.

De forma oposta aos microprocessadores, onde se super dimensiona ao máximo tendo como limite o preço que o usuário deseja investir, a escolha do microcontrolador é feita pelo projetista do equipamento. É erro de projeto super dimensionar. Cada desperdício será multiplicado pelo numero de equipamentos fabricados (às vezes milhões). Por isso existem duas linhas de pesquisa paralelas, mas opostas uma criando microcontroladores mais capazes, para atender produtos de mais tecnologia como os novos celulares ou receptores de

TV digital e outra para criar microcontroladores mais simples e baratos, para aplicações elementares (como um chaveiro que emite sons).

De forma diferente da programação para microprocessadores, que em geral contam com um sistema operacional e um BIOS, o programador ou projetista que desenvolve sistemas com microcontroladores tem que lidar com uma gama muito grande de desafios, fazendo muitas vezes todo o processo construtivo do aparelho: BIOS, firmware e circuitos.

Microcontrolador PIC

Os PIC (**PICmicro**) são uma família de microcontroladores fabricados pela Microchip Technology, que processam dados de 8 bits e de 16 bits, mais recentemente 32, com extensa variedade de modelos e periféricos internos, com arquitetura Harvard e conjunto de instruções RISC (conjuntos de 35 instruções e de 76 instruções), com recursos de programação por Memória flash, EEPROM e OTP.

Os microcontroladores PIC têm famílias com núcleos de processamento de 12 bits, 14 bits e 16 bits e trabalham em velocidades de 0kHz (ou DC) a 48MHz, usando ciclo de instrução mínimo de 4 períodos de clock, o que permite uma velocidade de no máximo 10 MIPS. Há o reconhecimento de interrupções tanto externas como de periféricos internos. Funcionam com tensões de alimentação de 2 a 6V e os modelos possuem encapsulamento de 6 a 100 pinos em diversos formatos (SOT23, DIP, SOIC, TQFP, etc)

TRANSISTOR

O transistor (ou transistor) é um componente eletrônico que começou a se popularizar na década de 1950 tendo sido o principal responsável pela revolução da eletrônica na década de 1960, e cujas funções principais são amplificar e chavear sinais elétricos. O termo vem de transfer resistor (resistor de transferência), como era conhecido pelos seus inventores.

O processo de transferência de resistência, no caso de um circuito analógico, significa que a impedância característica do componente varia para cima ou para baixo da polarização pré-estabelecida. Graças à esta função, a corrente elétrica que passa entre coletor e emissor do transistor varia dentro de determinados parâmetros pré-estabelecidos pelo projetista do circuito eletrônico; esta variação é feita através da variação de corrente num dos terminais chamado base, que conseqüentemente ocasiona o processo de amplificação de sinal.

Entende-se por "amplificar" o procedimento de tornar um sinal elétrico mais fraco em mais forte. Um sinal elétrico de baixa intensidade, como os sinais gerados por um microfone, é injetado em um circuito eletrônico (transistorizado por exemplo), cuja função principal é transformar este sinal fraco gerado pelo microfone em sinais elétricos com as mesmas características mas com potência suficiente para excitar os alto-falantes, a este processo todo se dá o nome de ganho de sinal.

CIRCUITO INTEGRADO

Um circuito integrado, também conhecido por chip, é um dispositivo microeletrônico que consiste de muitos transistores e outros componentes interligados capazes de desempenhar muitas funções. Suas dimensões são extremamente reduzidas, os componentes são formados em pastilhas de material semicondutor.

A importância da integração está no baixo custo e alto desempenho, além do tamanho reduzido dos circuitos aliado à alta confiabilidade e estabilidade de funcionamento. Uma vez que os componentes são formados ao invés de montados, a resistência mecânica destes permitiu montagens cada vez mais robustas a choques e impactos mecânicos, permitindo a concepção de portabilidade dos dispositivos eletrônicos.

No circuito integrado completo ficam presentes os transistores, condutores de interligação, componentes de polarização, e as camadas e regiões isolantes ou condutoras obedecendo ao seu projeto de arquitetura.

No processo de formação do chip, é fundamental que todos os componentes sejam implantados nas regiões apropriadas da pastilha. É necessário que a isolação seja perfeita, quando for o caso. Isto é obtido por um processo chamado difusão, que se dá entre os componentes formados e as amadas com o material dopado com fósforo, e separadas por um material dopado com boro, e assim por diante.

Após sucessivas interconexões, por boro e fósforo, os componentes formados ainda são interconectados externamente por uma camada extremamente fina de alumínio, depositada sobre a superfície e isolada por uma camada de dióxido de silício.

DIFERENÇA DE POTENCIAL

Pode-se definir a diferença de potencial entre dois pontos como a variação entre os potenciais elétricos desses dois pontos.

MICROCONTROLADOR

Um microcontrolador (também denominado MCU ou µ C) é um computador num chip, contendo um processador, memória e funções de entrada/saída. É um microprocessador que enfatiza a alta integração, em contraste com os microprocessadores de uso geral (do tipo usado em computadores pessoais). Além dos componentes lógicos e aritméticos usuais dum microprocessador de uso geral, o microcontrolador integra elementos adicionais tais como memória RAM, EEPROM ou Memória flash para armazenamento de dados ou programas, dispositivos periféricos e interfaces de E/S que podem ir de um simples pino digital do componente a uma interface USB ou Ethernet nos mais avançados (como o ARMLPC2368).

Com freqüências de clock de poucos MHz ou ainda mais baixas microcontroladores são considerados lentos se comparados aos microprocessadores modernos, mas isso é perfeitamente adequado p ara aplicações típicas. Eles consomem relativamente pouca energia (miliwatts), e geralmente possuem a capacidade de "hibernar" enquanto aguarda que aconteça algum evento interessante provocado por um periférico, tal como o

pressionar dum botão, que os colocam novamente em atividade. O consumo de energia enquanto estão "hibernando" pode ser de nano watts, tornando-os ideais para aplicações de baixa energia e que economizem bateria.

De forma oposta aos microprocessadores, onde se super dimensiona ao máximo tendo como limite o preço que o usuário deseja investir, a escolha do microcontrolador é feita pelo projetista do equipamento. É erro de projeto super dimensionar. Cada desperdício será multiplicado pelo número de equipamentos fabricados (às vezes milhões). Por isso existem duas linhas de pesquisa paralelas, mas opostas uma criando microcontroladores mais capazes, para atender produtos de mais tecnologia como os novos celulares ou receptores de TV digital e outra para criar microcontroladores mais simples e baratos, para aplicações elementares (como um chaveiro que emite sons).

De forma diferente da programação para microprocessadores, que em geral contam com um sistema operacional e um BIOS, o programador ou projetista que desenvolve sistemas com microcontroladores tem que lidar com uma gama muito grande de desafios, fazendo muitas vezes todo o processo construtivo do aparelho: BIOS, firmware e circuitos.

RS 232

RS-232 (também conhecido por EIA RS-232C ou V.24) é um padrão para troca serial de dados binários entre um DTE (terminal de dados, de Data Terminal equipment) e um DCE (comunicador de dados, de Data Communication equipment). É comumente usado nas portas seriais dos PCs.

Hoje, o protocolo de comunicação RS-232 vem sendo, gradualmente, suprimido pelo USB para comunicação local. O protocolo USB é mais rápido, possui conectores mais simples de usar e tem um melhor suporte por software. Por isso muitas placas-mãe, destinadas ao uso em escritórios ditas "livre de legados" (legacy-free) são produzidas sem circuitos RS-232. Mesmo assim, esse protocolo continua sendo utilizado em periféricos para pontos de venda (caixas registradoras, leitores de códigos de barra ou fita magnética) e para a área industrial (dispositivos de controle remoto). Por essas razões, computadores para estes fins continuam sendo produzidos com portas RS-232, tanto on-board ou em placas para barramentos PCI ou barramento ISA. Como alternativa, existem adaptadores para portas USB, que podem ser utilizados para conectar teclados ou mouses PS/2, uma ou mais portas seriais e uma ou mais portas paralelas.

No protocolo de comunicação RS-232, caracteres são enviados um a um como um conjunto de bits. A codificação mais comumente usada é o "start-stop assíncrono" que usa um bit de início, seguido por sete ou oito bits de dados, possivelmente um bit de paridade, e um ou dois bits de parada sendo, então, necessários 10 bits para enviar um único caractere. Tal fato acarreta a necessidade em dividir por um fator de dez a taxa de transmissão para obter a velocidade d e transmissão. A alternativa mais comum ao "start-stop assíncrono" é o HDLC. O padrão define os níveis elétricos correspondentes aos níveis lógicos um e zero, a velocidade de transmissão padrão e os tipos de conectores.

O padrão especifica 20 diferentes sinais de conexão, e um conector em forma de D é comumente usado. São utilizados conectores machos e fêmeas - geralmente os conectores dos cabos são machos e os conectores de

dispositivos são fêmeas - e estão disponíveis adaptadores m-m e f-f. Há também os chamados "null modems" para conectar unidades utilizando-se ambas como terminais de dados (ou modems). Para configuração e diagnóstico de problemas com cabos RS-232 pode-se utilizar uma "breakout box". Este dispositivo possui um conector macho e um conector fêmea e deve ser anexado em linha. Além disso, possui luzes para cada pino e meios de interconectar os pinos com diferentes configurações.

A maioria dos pinos são inutilizados pela maioria dos dispositivos sendo, então, comum que máquinas economizem espaço e dinheiro, utilizando conexões menores. A segunda geração dos IBM PC AT foi disponibilizada com um conector em forma de D com apenas 9 pinos, tornando-se o padrão.

Grandes partes dos dispositivos utilizam conectores de 25 pinos. Conseqüentemente, cabos com 9 pinos em uma extremidade e 25 em outra são comuns. O Apple Macintosh utilizava um sistema similar, mas posteriormente mudou para um novo conector com apenas 8 pinos, menos que o necessário para um modem.

Os cabos para RS-232 podem ser construídos com conectores disponíveis em qualquer loja de eletrônicos. Os cabos podem ter d e 3 a 25 pinos. Cabos "Flat RJ" (cabos de telefone) podem ser usados com conectores RJ-RS232 e são os de mais fácil configuração. A razão pela qual é possível criar uma interface mínima com apenas três fios é que todo sinal RS-232 utiliza o mesmo fio terra para referência. O uso de circuitos desbalanceados deixa o RS-232 a problemas devido a diferenças de potencial entre os sinais de terra dos dois circuitos. Este padrão também tem um pobre controle dos tempos de picos e descidas do sinal, levando a potenciais problemas de comunicação.

O RS-232 é recomendado para conexões curtas (quinze metros ou menos). Os sinais variam de 3 a 15 volts positivos ou negativos, valores próximos de zero não são sinais válidos. O nível lógico um é definido por ser voltagem negativa, a condição de sinal é chamada marca e tem significado funcional de OFF (desligado). O nível lógico zero é positivo, a condição de sinal é espaço, e a função é ON (ligado). Níveis de sinal +-5, +-10, +- 12 e +-15 são vistos comumente, dependendo da fonte elétrica disponível.

Marca e espaço são termos herdados das teletypewriters. O modo de comunicação nativo destas eram simples séries de circuitos d e corrente contínua que são interrompidos, muito similar aos telefones que possuíam as "rodas de discagem" que interrompiam o sinal telefônico. A condição de marca é quando o circuito está fechado e a condição de espaço, quando o circuito está aberto. O início de um caractere é sinalizado por um espaço e os bits de parada são marcas. Quando a linha é interrompida, a teletypewriter entra num ciclo contínuo, mas nada é impresso porque tudo o que é recebido são zeros, o caractere NULL.

Os dispositivos RS-232 podem ser classificados em DTE e DCE. Essa classificação permite definir quais fios irão mandar e/ou enviar sinais de dados. De qualquer modo, estas definições nem sempre seguidas. Normalmente é necessário consultar a documentação ou testar as conexões com uma "breakout box" para determinar os sinais necessários.

O sinal de terra tem a função de aterrar as outras conexões e é necessário. Se os equipamentos estiverem muito longe, com diferentes fontes de eletricidade, o terra se degradará entre os dois dispositivos e a comunicação irá falhar, sendo esta uma condição difícil de traçar. Em conectores de 25

pinos, o pino sete geralmente é o terra (pino 1 e terra do chassis são raramente usados). Neste mesmo conector, os pinos dois e três são os pinos de transmissão e recepção, um dispositivo deve enviar no 2 e receber no 3; o outro deve ser o contrário (se não, essa inversão deve ser feita no fim do cabo, como num cabo para null modem, também chamado crossover). No caso de desenvolver cabos para uma conex ão, pode-se testá-lo com uma breakout box qual pino está transmitindo.

Estritamente falando, apenas um dispositivo precisa estar transmitindo (se não for necessária comunicação duplex ou um handshake), por exemplo, uma impressor simples que não responde seu estado para o computador. Necessariamente, deve-se utilizar tanto o pino TX quanto o pino RX.

Outros handshakes podem ser necessários por um ou por outro dispositivo. Por exemplo, o pino 20 é comumente usado para indicar "dispositivo pronto". Os pinos também podem ser curto-circuitados. Por exemplo, um pino que pergunte "você está pronto?" que parte do dispositivo A pode ser ligado diretamente no pino referente à resposta "estou pronto" no dispositivo A se o dispositivo A não transmitir tal sinal. Os pinos normalmente utilizados para handshake são os pinos 20, 8, 4 e 6.

Há várias configurações de software para conexões seriais. As mais comuns são velocidade e bits de paridade e parada. A velo cidade é a quantidade de bits por segundo transmitida de um dispositivo para outro. Taxas comuns de transmissão são 300, 1200, 2400, 9600, 19200, etc. Tipicamente ambos os dispositivos devem estar configurados com a mesma velocidade, alguns dispositivos, porém, podem ser configurados para auto detectar
a velocidade.

Paridade é um método de verificar a precisão dos dados. Paridade é normalmente nula (não usada), mas pode ser par ou ímpar. Paridade funciona modificando os dados, em cada byte enviado. Paridade nula é simples, os dados não são modificados. Na paridade par, os dados são acomodados de modo que o número de bits 1 (isto é, sua contagem em um byte) seja um número par; isto é feito definindo o bit de paridade (geralmente os bits mais ou menos significativo) como 0 ou 1. Na paridade impar, o número de bits 1 é um número impar. A paridade pode era usada pelo receptor para detectar a transmissão de erros - se um byte foi recebido com o número errado de bits 1, então ele deve estar corrompido. Se a paridade estiver correta então não deve haver erros, ou então há um número par de erros. Bits de parada são enviados no fim de cada byte transmitido com o intuito de permitir que o receptor do sinal se sincronize. Existe uma convenção para a notação se uma configuração de software de uma conexão serial, esta notação é da forma D/P/S. Sendo que a configuração mais comum é a 8/N/1 que especifica que são transmitidos 8 bits de dados, paridade nula e um bit de parada. O número de bits de dados pode ser 7, 8 ou (às vezes) 9. Paridade pode ser nula (N), ímpar (O) ou par (E); o bit de paridade é emprestado dos bits de dados, então 8/E/1 significa que um dos oito bits de dados é utilizado como bit de paridade. Pode haver 1, 1,5 ou 2 bits de parada (1,5 era utilizado em teletypewriters baudot de 60 palavras por minuto).

Outras configurações definem quando pinos enviam sinais de "handshake", ou outra checagem de integridade dos dados. Combinações comuns são RTS/CTS, DTR/DSR, ou XON/XOFF (que não usam pinos no conector, mas caracteres especiais no fluxo dos dados). O caractere XON diz

ao receptor que o remetente do caractere está pronto para receber mais dados. O caractere XOFF diz ao receptor para parar de enviar caracteres. O XON/XOFF está em desuso, e é preferível que se utilize o controle de fluxo RTS/CTS. XON/XOFF é um método "em banda" que funciona entre dois pontos, mas ambos devem suportar o protocolo, e há uma confusão em potencial no início. Pode ser feito numa interface com três fios. RTS/CTS foi desenvolvido com o intuito de permitir que a teletypewriter e o modem coordenassem ligações half-duplex onde apenas um modem pode transmitir por vez. O terminal deve "levantar" o sinal Pronto Para Enviar e esperar que o modem responda com Envie os Dados. RTS/CTS é um "handshake" no nível do hardware, mas tem suas vantagens. Uma teletypewriter ASR tinha um leitor de fita de papel. Os caracteres eram enviados quando a fita era lida (ASR vem de Automatic Send Receive,envia e recebe automaticamente). Quando a máquina recebia um caractere XOFF, ela desligava a leitora de fita e ao receber um XON a religava. O sistema remoto poderia enviar um XOFF quando era necessário que o remetente diminuísse sua velocidade. Nos sistemas, originalmente, as mensagens eram previamente preparadas na fita d e papel para que o tempo de transmissão fosse minimizado. Largura de banda era muito escaça e cara. Em alguns minicomputadores antigos, a fita de papel era a única maneira de efetuar guardar e restaurar dados e programas.

RECEPTOR ELÉTRICO

O receptor elétrico é o dispositivo que transforma a energia elétrica em outra forma de energia, exceto em elétrica, sendo exemplos de receptores, a lâmpada, o chuveiro, um motor elétrico, etc.

TRANSISTOR

O transistor (ou transistor) é um componente eletrônico que começou a se popularizar na década de 1950 tendo sido o principal responsável pela revolução da eletrônica na década de 1960, e cujas funções principais são amplificar e chavear sinais elétricos. O termo vem de transfer resistor (resistor de transferência), como era conhecido pelos seus inventores.

O processo de transferência de resistência, no caso de um circuito analógico, significa que a impedância característica do componente varia para cima ou para baixo da polarização pré-estabelecida. Graças à esta função, a corrente elétrica que passa entre coletor e emissor do transistor varia dentro de determinados parâmetros pré-estabelecidos pelo projetista do circuito eletrônico; esta variação é feita através da variação de corrente num dos terminais chamado base, que conseqüentemente ocasiona o processo de amplificação de sinal. Entende-se por "amplificar" o procedimento de tornar um sinal elétrico mais fraco em mais forte. Um sinal elétrico de baixa intensidade, como os sinais gerados por um microfone, é injetado em um circuito (transistorizado), cuja função principal é transformar este sinal fraco gerado pelo microfone em sinais elétricos com as mesmas características, mas com potência suficiente para excitar os alto-falantes, a este processo todo se dá o nome de ganho de sinal.

REGULADOR DE TENSÃO

O regulador de tensão é projetado para manter u ma tensão constante através da carga por meio do ajuste da corrente. O capacitor de desvio elimina qualquer ruído de freqüência da carga (o regulador de tensão está monitorando a carga, então isso levaria a flutuações de correntes indesejadas). Ele recebe u ma tensão maior do que a desejada, e a regula, reduzindo-a até a tensão desejada. Cada regulador possui uma tensão pré determinada, ou seja, se desejada a tensão de 5V, utiliza-se o regulador 7805, se desejada a tensão de 12V, utiliza-se o regulador 7812, entre outros.

MOTOR DE PASSO

O motor de passo é um transdutor que converte energia elétrica em movimento controlado através de pulsos, o que possibilita o deslocamento por passo, onde passo é o menor deslocamento angular.

Com o passar dos anos houve um aumento na popularidade deste motor, principalmente pelo seu tamanho e custo reduzidos e também a total adaptação por controles digitais.

Outra vantagem do motor de passos em relação aos outros motores é a estabilidade. Quando quisermos obter uma rotação específica de certo grau, calcularemos o número d e rotação por pulsos o que nos possibilita uma boa precisão no movimento.

Os antigos motores passavam do ponto e, para voltar, precisavam da realimentação negativa. Por não girar por passos a inércia destes é maior e assim são mais instáveis.

Definições para Motores a Passo

Antes de explicarmos os tipos de motores e o funcionamento em si, definiremos algumas outras expressões a fim de tornar o texto mais claro.

Rotor = É denominado rotor o conjunto eixo-imã que rodam solidariamente na parte móvel do motor.

Estator = Define-se como estator a trav e fixa onde as bobinas são enroladas.

Parâmetros Importantes

Graus por Passo = sem dúvida a característica mais importante ao se escolher o motor, o número de graus por passo está intimamente vinculado com o número de passos por volta. Os valores mais comuns para esta característica, também referida como resolution, são 0.72,1.8, 3.6, 7.5, 15 e até 90 graus.

Momento de Frenagem = momento máximo com o rotor bloqueado, sem perda de passos.

Momento (Torque) = efeito rotativo de uma força, medindo a partir do produto da mesma pela distância perpendicular até o ponto em que ela atua partindo de sua linha de ação.

Taxa de Andamento = regime de operação atingido após uma aceleração suave.

Momento de Inércia=medida da resistência mecânica oferecida por um corpo à aceleração angular.

Auto- Indutância= determina a magnitude da corrente média em regimes pesados de operação, de acordo com o tipo de enrolamento do estator: relaciona o fluxo magnético com as correntes que o produzem.

Resistências Ôhmicas = determina a magnitude da corrente do estator com o rotor parado.

Corrente máxima do estator =determinada pela bitola do fio empregado nos enrolamentos.

"Holding Torque"=é mínima potência para fazer o motor mudar de posição parada.

Torque Residual = é a resultante de todos os fluxos magnético presente nos pólos do estator.

Resposta de Passo = é tempo que o motor gasta para executar o comando.

Ressonância = como todo material, o motor de passos tem sua freqüência natural. Quando o motor gira com uma freqüência igual a sua, ele começa a oscilar e a perder passos.

Tensão de trabalho=normalmente impresso na própia chassi do motor, a tensão em que trabalha o motor é fundamental na obtenção do torque do componente. Tensões acima do estipulado pelo fabricante em seu datasheet costumam aumentar o torque do motor, porém, tal procedimento resulta na diminuição da vida útil do mesmo. Destaca-se que a tensão de trabalho do motor não necessariamente deve ser a tensão utilizada na lógica do circuito. Os valores normalmente encontrados variam de +5V à +48V.

Tipos de Motores de Passo

Relutância Variável=Apresenta um rotor com muitas polaridades construídas a partir de ferro doce, apresenta também em estator laminado. Por não possuir imã, quando energizado apresenta torque estático nulo. Tendo assim baixa inércia de rotor não pode ser utilizado como carga inercial grande.

Imã Permanente=Apresenta um rotor de material alnico ou ferrite e é magnetizado radialmente devido a isto o torque estático não é nulo.

Híbridos=É uma mistura dos dois anteriores e apresenta rotor e estator multidentados. O rotor é de imã permanente e magnetizado axialmente. Apresenta grande precisão (3%), boa relação torque/tamanho e ângulos pequenos (0,9 e 1,8 graus).

Para que o rotor avance um passo é necessário que a polaridade magnética de um dente do estator se alinha com a polaridade magnética oposta de um dente do rotor.

ELÉTROIMÃ

O eletroímã ou electroíman é um dispositivo que utiliza corrente elétrica para gerar um campo magnético, semelhantes àqueles encontrados nos ímãs naturais. É geralmente construído aplicando-se um fio elétrico espiralado ao redor de um núcleo de ferro, aço, níquel ou cobalto ou algum material ferromagnético.

Quando o fio é submetido a uma tensão, o mesmo é percorrido por uma corrente elétrica, o que gerará um campo magnético na área circunvizinha a essa espira através da Lei de Biot-Savart. A intensidade do campo e a distância que ele atingirá a partir do eletroímã dependerão da intensidade da corrente aplicada e do número de voltas da espira.

A passagem de corrente elétrica por um condutor produz campos magnéticos nas suas imediações e estabelece um fluxo magnético no material ferromagnético envolto pelas espiras do condutor. A razão entre a intensidade do fluxo magnético concatenado pelas espiras e a corrente que produziu esse fluxo é a indutância.

O pedaço de ferro apresenta as características de um ímã permanente, enquanto a corrente for mantida circulando, e o campo magnético pode ser constante ou variável no tempo dependendo da corrente utilizada (contínua ou alternada). Ao se interromper a passagem da corrente o envolto pelas espiras pode tanto manter as características magnéticas ou não, dependendo das propriedades do mesmo.

7.3 CÓDIGO FONTE DO SOFTWARE

Form1.cpp

```
#pragma once
#include "Windows.h"

namespace ferpa2 {

	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;

	/// <summary>
	/// Summary for Form1
	///
	/// WARNING: If you change the name of this class, you will need
to change the
	///          'Resource File Name' property for the managed
resource compiler tool
	///          associated with all .resx files this class depends
on.  Otherwise,
	///          the designers will not be able to interact properly
with localized
	///          resources associated with this form.
	/// </summary>
	public ref class Form1 : public System::Windows::Forms::Form
	{
	public:
		Form1(void)
		{
			InitializeComponent();
			//
			//TODO: Add the constructor code here
			//
		}

	protected:
		/// <summary>
		/// Clean up any resources being used.
		/// </summary>
		~Form1()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::Button^  button1;
	private: System::IO::Ports::SerialPort^  serialPort1;
	private: System::Windows::Forms::Button^  button2;
	private: System::Windows::Forms::Button^  button3;
	private: System::Windows::Forms::Button^  button4;
	private: System::Windows::Forms::Button^  button5;
	private: System::Windows::Forms::Button^  button6;
	private: System::Windows::Forms::Button^  button7;
	private: System::Windows::Forms::Button^  button8;
	private: System::Windows::Forms::Button^  button9;
```

```
	private: System::ComponentModel::IContainer^  components;
	protected:

	private:
		/// <summary>
		/// Required designer variable.
		/// </summary>


#pragma region Windows Form Designer generated code
		/// <summary>
		/// Required method for Designer support - do not modify
		/// the contents of this method with the code editor.
		/// </summary>
		void InitializeComponent(void)
		{
			this->components = (gcnew
System::ComponentModel::Container());
			this->button1 = (gcnew
System::Windows::Forms::Button());
			this->serialPort1 = (gcnew
System::IO::Ports::SerialPort(this->components));
			this->button2 = (gcnew
System::Windows::Forms::Button());
			this->button3 = (gcnew
System::Windows::Forms::Button());
			this->button4 = (gcnew
System::Windows::Forms::Button());
			this->button5 = (gcnew
System::Windows::Forms::Button());
			this->button6 = (gcnew
System::Windows::Forms::Button());
			this->button7 = (gcnew
System::Windows::Forms::Button());
			this->button8 = (gcnew
System::Windows::Forms::Button());
			this->button9 = (gcnew
System::Windows::Forms::Button());
			this->SuspendLayout();
			//
			// button1
			//
			this->button1->Location = System::Drawing::Point(61,
54);
			this->button1->Name = L"button1";
			this->button1->Size = System::Drawing::Size(152, 41);
			this->button1->TabIndex = 0;
			this->button1->Text = L"X- Ir";
			this->button1->UseVisualStyleBackColor = true;
			this->button1->Click += gcnew
System::EventHandler(this, &Form1::button1_Click);
			//
			// serialPort1
			//
			this->serialPort1->BaudRate = 1200;
			this->serialPort1->PortName = L"COM4";
			//
			// button2
			//
			this->button2->Location = System::Drawing::Point(335,
311);
```

```
                this->button2->Name = L"button2";
                this->button2->Size = System::Drawing::Size(106, 23);
                this->button2->TabIndex = 1;
                this->button2->Text = L"habilitar";
                this->button2->UseVisualStyleBackColor = true;
                this->button2->Click += gcnew
System::EventHandler(this, &Form1::button2_Click);
                //
                // button3
                //
                this->button3->Location = System::Drawing::Point(61,
148);
                this->button3->Name = L"button3";
                this->button3->Size = System::Drawing::Size(152, 40);
                this->button3->TabIndex = 2;
                this->button3->Text = L"X- Voltar";
                this->button3->UseVisualStyleBackColor = true;
                this->button3->Click += gcnew
System::EventHandler(this, &Form1::button3_Click);
                //
                // button4
                //
                this->button4->Location = System::Drawing::Point(61,
245);
                this->button4->Name = L"button4";
                this->button4->Size = System::Drawing::Size(152, 43);
                this->button4->TabIndex = 3;
                this->button4->Text = L"X- Parar";
                this->button4->UseVisualStyleBackColor = true;
                this->button4->Click += gcnew
System::EventHandler(this, &Form1::button4_Click);
                //
                // button5
                //
                this->button5->Location = System::Drawing::Point(335,
54);
                this->button5->Name = L"button5";
                this->button5->Size = System::Drawing::Size(162, 41);
                this->button5->TabIndex = 4;
                this->button5->Text = L"Y- Ir";
                this->button5->UseVisualStyleBackColor = true;
                this->button5->Click += gcnew
System::EventHandler(this, &Form1::button5_Click);
                //
                // button6
                //
                this->button6->Location = System::Drawing::Point(335,
148);
                this->button6->Name = L"button6";
                this->button6->Size = System::Drawing::Size(162, 40);
                this->button6->TabIndex = 5;
                this->button6->Text = L"Y- Voltar";
                this->button6->UseVisualStyleBackColor = true;
                this->button6->Click += gcnew
System::EventHandler(this, &Form1::button6_Click);
                //
                // button7
                //
                this->button7->Location = System::Drawing::Point(335,
245);
                this->button7->Name = L"button7";
```

```
                this->button7->Size = System::Drawing::Size(162, 43);
                this->button7->TabIndex = 6;
                this->button7->Text = L"Y- Parar";
                this->button7->UseVisualStyleBackColor = true;
                this->button7->Click += gcnew
System::EventHandler(this, &Form1::button7_Click);
                //
                // button8
                //
                this->button8->Location = System::Drawing::Point(603,
54);
                this->button8->Name = L"button8";
                this->button8->Size = System::Drawing::Size(149, 41);
                this->button8->TabIndex = 7;
                this->button8->Text = L"Imã - Ligar";
                this->button8->UseVisualStyleBackColor = true;
                this->button8->Click += gcnew
System::EventHandler(this, &Form1::button8_Click);
                //
                // button9
                //
                this->button9->Location = System::Drawing::Point(603,
245);
                this->button9->Name = L"button9";
                this->button9->Size = System::Drawing::Size(149, 43);
                this->button9->TabIndex = 8;
                this->button9->Text = L"Imã - Voltar";
                this->button9->UseVisualStyleBackColor = true;
                this->button9->Click += gcnew
System::EventHandler(this, &Form1::button9_Click);
                //
                // Form1
                //
                this->AutoScaleDimensions = System::Drawing::SizeF(6,
13);
                this->AutoScaleMode =
System::Windows::Forms::AutoScaleMode::Font;
                this->ClientSize = System::Drawing::Size(805, 346);
                this->Controls->Add(this->button9);
                this->Controls->Add(this->button8);
                this->Controls->Add(this->button7);
                this->Controls->Add(this->button6);
                this->Controls->Add(this->button5);
                this->Controls->Add(this->button4);
                this->Controls->Add(this->button3);
                this->Controls->Add(this->button2);
                this->Controls->Add(this->button1);
                this->Name = L"Form1";
                this->Text = L"Form1";
                this->ResumeLayout(false);

            }
#pragma endregion
            //Funcao para X ir
     private: System::Void button1_Click(System::Object^  sender,
System::EventArgs^  e) {
                        serialPort1->Write("\r");
                        Sleep(200);
                        serialPort1->Write("fgay.run(1,350)\r");
                        Sleep(200);
                 }
```

```
                    //Funcao para habilitar o projeto
      private: System::Void button2_Click(System::Object^  sender,
System::EventArgs^  e) {
                          serialPort1->Open();
                          Sleep(200);
                    }
                   //Funcao para X parar
      private: System::Void button4_Click(System::Object^  sender,
System::EventArgs^  e) {
                          serialPort1->Write("\r");
                          Sleep(200);
                          serialPort1->Write("fgay.free()\r");
                          Sleep(200);
                    }
                    //Funcao para X voltar
      private: System::Void button3_Click(System::Object^  sender,
System::EventArgs^  e) {
                           serialPort1->Write("\r");
                          Sleep(200);
                          serialPort1->Write("fgay.run(0,350)\r");
                          Sleep(200);
             }
                    //Funcao para Y ir
private: System::Void button5_Click(System::Object^  sender,
System::EventArgs^  e) {
                         serialPort1->Write("\r");
                          Sleep(200);
                          serialPort1->Write("fgay2.run(1,350)\r");
                          Sleep(200);
             }
private: System::Void button6_Click(System::Object^  sender,
System::EventArgs^  e) {
                         serialPort1->Write("\r");
                          Sleep(200);
                          serialPort1->Write("fgay2.run(0,350)\r");
                          Sleep(200);
             }
private: System::Void button7_Click(System::Object^  sender,
System::EventArgs^  e) {
                         serialPort1->Write("\r");
                          Sleep(200);
                          serialPort1->Write("fgay2.free()\r");
                          Sleep(200);
             }
private: System::Void button8_Click(System::Object^  sender,
System::EventArgs^  e) {
                   serialPort1->Write("\r");
                   Sleep(200);
                   serialPort1->Write("hover.io0.set(1)\r");
                   Sleep(200);
             }
private: System::Void button9_Click(System::Object^  sender,
System::EventArgs^  e) {
                    serialPort1->Write("\r");
                   Sleep(200);
                   serialPort1->Write("hover.io0.set(0)\r");
                   Sleep(200);
             }
};
}
```

Ferpa2.cpp

```
// ferpa2.cpp : main project file.

#include "stdafx.h"
#include "Form1.h"

using namespace ferpa2;

[STAThreadAttribute]
int main(array<System::String ^> ^args)
{
	// Enabling Windows XP visual effects before any controls are
created
	Application::EnableVisualStyles();
	Application::SetCompatibleTextRenderingDefault(false);

	// Create the main window and run it
	Application::Run(gcnew Form1());
	return 0;
}
```

AssemblyInfo.cpp

```
#include "stdafx.h"

using namespace System;
using namespace System::Reflection;
using namespace System::Runtime::CompilerServices;
using namespace System::Runtime::InteropServices;
using namespace System::Security::Permissions;

//
// General Information about an assembly is controlled through the
following
// set of attributes. Change these attribute values to modify the
information
// associated with an assembly.
//
[assembly:AssemblyTitleAttribute("ferpa2")];
[assembly:AssemblyDescriptionAttribute("")];
[assembly:AssemblyConfigurationAttribute("")];
[assembly:AssemblyCompanyAttribute("")];
[assembly:AssemblyProductAttribute("ferpa2")];
[assembly:AssemblyCopyrightAttribute("Copyright (c)  2009")];
[assembly:AssemblyTrademarkAttribute("")];
[assembly:AssemblyCultureAttribute("")];

//
// Version information for an assembly consists of the following four
values:
//
//      Major Version
//      Minor Version
//      Build Number
//      Revision
//
// You can specify all the value or you can default the Revision and
Build Numbers
// by using the '*' as shown below:

[assembly:AssemblyVersionAttribute("1.0.*")];
```

```
[assembly:ComVisible(false)];

[assembly:CLSCompliantAttribute(true)];

[assembly:SecurityPermission(SecurityAction::RequestMinimum,
UnmanagedCode = true)];
```