CENTRO UNIVERSITÁRIO DE BRASÍLIA - UniCEUB
Faculdades de Tecnologia e Ciências Sociais Aplicadas - FATECS
Curso de Engenharia da Computação

ELEVADOR CONTROLADO VIA MICROCONTROLADOR

RAFAEL JOSÉ RAMOS
RA: 2021860/0

Brasília/DF. 2008

ELEVADOR CONTROLADO VIA MICROCONTROLADOR

Trabalho de conclusão de curso apresentado à banca examinadora do Centro Universitário de Brasília – UniCEUB como requisito parcial para a obtenção do Certificado de conclusão do curso de Engenharia da Computação.
Prof. José Julimá Bezerra Júnior

Brasília/DF.

2008

DEDICATÓRIA

Primeiramente quero agradecer Deus por ter me dado essa oportunidade única de estudar e me aprimorar em uma área que é minha paixão e por ter me dado força para trabalhar e me dedicar nessa árdua tarefa. Aos meus familiares que sempre estiveram presente em todo momento que precisei.

AGRADECIMENTOS.

Diversas pessoas me ajudaram durante esse processo, quero agradecer todas fizeram parte desse processo. Em especial quero agradecer ao meu pai Cacildo Gonçalves Ramos, que mesmo me empenhando em outra área sempre me deu força, a minha mãe Elizabet Laceur, pois sei que em suas preces sempre pede por mim. Ao professor José Julimá por estar sempre disposto em qualquer dúvida. A minha noiva Lena que por diversas vezes me deu força e mesmo se sacrificando me aconselhava o melhor, e me fez a correção da gramática e semântica junto comigo.

RESUMO

A questão da falta de um controle automático dos elevadores antigos é um problema desafiador. Hoje em dia ainda existem muitos elevadores que não utilizam nenhum tipo de automação dos recursos envolvidos no seu funcionamento. O objetivo deste foi encontrar uma solução fácil de ser instalada sem grande impacto na estrutura de um prédio. Com essa finalidade será demonstrado um circuito integrado a um microcontrolador que é responsável por empregar lógica aos acionamentos dos motores. Desta forma qualquer necessidade de novas implementações de uso ao elevador pode ser feita sem grandes impactos, com pequenas modificações eletrônicas e de software.

Palavras-chave: microcontrolador, circuito elétrico, otimização.

ABSTRACT

The problem of the lack of an automatic control of the old elevators is a challenging problem. Today, there are still many elevators that do not use any type of automation of the resources involved in this operation. The purpose of this was to find an easy solution to be installed without major impact on the structure of a building. For this purpose will be shown an integrated circuit to a microcontroller which will be responsible for employing the logic drives the engines. So no need to use new implementations of the elevator can be done without major impacts, with small modifications and electronic and software.

Keywords: microcontroller, electrical circuit, optimization.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

A/D	*Analog/Digital* (Analógico/Digital)
ACC	Acumulador
ALU	*Arithmeic-Logic Unit* (Unidade Lógica e Aritmética).
CI	Circuito Integrado
CPU	*Central Processing Unit* (Unidade Central de Processamento).
D/A	*Digital/Analog* (Digital/Analógico)
E/S	Entrada/Saída.
EPROM	*Erasable Programmable Read-only Memory* (memória somente de leitura programável eletronicamente).
HD	*Hard Disc*
I/O	*INPUT/OUTPUT(*Entrada/Saída*).*
IE	*Interrupt Enable* (Permitir a interrupção)
IP	*Interrupt Priority* (Prioridade de Interrupção)
MC	Microcontrolador
MP	Microprocessador
PC	*Personal Computer* (Computador Pessoal).
PSEN	*Program Store Enable* (Permitir Armazenar Programa)
RAM	*Random Access Memory* (memória de acesso aleatório).
ROM	*Ready Only Memory* (memória somente de leitura).
RST	*Reset* (Inicializar)
Vcc	*Collector Common Voltage* (Coletor Comum de Tensão)
FEM	Força Eletromotriz
FCEM	Força Contra Eletromotriz

1 INTRODUÇÃO

1.1 Motivação e posicionamento

Hoje diversos prédios se encontram com elevadores antigos e que não utilizam nenhum tipo de automação de seus recursos, sejam esses residenciais ou comerciais. Com isso diversos problemas tendem a aparecer, tais como: elevados tempos de espera, embarque de passageiros em trajetos contrários a do elevador, gasto indevido de energia elétrica com acionamento dos motores de forma inadequada e desgaste de diversas peças pelo funcionamento desnecessário do elevador.

Verificado a necessidade de facilitar o acesso a deficientes físicos, se faz cada vez mais necessário que estes equipamentos, muitas vezes essenciais para a locomoção destes em um edifício, utilizem esse tipo de lógica.

Uma solução simples desde problema baseia-se na utilização de circuitos elétricos e microcontroladores. Com isso a implantação desta de forma rápida e de baixo impacto à estrutura do prédio, visto que pode se utilizar os mesmos motores elétricos já existentes, tomando cuidado somente com o dimensionamento de cargas para cada. Com o emprego dos microcontroladores podemos inserir rotinas de ação, o que agregará mais inteligência à utilização dos recursos envolvidos em um elevador.

1.2 Objetivos e escopo

É apresentado neste trabalho, um circuito junto a um microcontrolador que é responsável pelos acionamentos dos motores de um elevador. Juntamente com ele, o código do software necessário para o correto funcionamento do microcontrolador.

Para a realização deste trabalho foi utilizado um elevador juntamente com sua caixa de corrida e contrapeso. Os amortecedores, porta de pavimento e outros elementos não foram reproduzidos neste modelo em escala reduzida, pois não são essenciais para demonstrar a lógica de funcionamento proposto neste trabalho,

podendo assim ser classificado com um elevador panorâmico. Verificado uma vez que este elevador apresenta somente quatro botões com a finalidade de chamada para cada andar, representando assim os botões internos e externos.

Toda a programação necessária foi feita em C, juntamente com a utilização do microcontrolador PIC18F452.

O funcionamento deste elevador se dá do seguinte modo: Ao acionar o botão de chamada e requisição do elevador, o microcontrolador fica responsável por fazer os acionamentos necessários para o atendimento a esta requisição.

O circuito também acionará os motores da porta, de maneira que o mesmo não se faça caso o elevador já esteja em movimento e só permita que a porta seja aberta com o elevador devidamente parado. Também será verificada a identificação de andar em um display. A Figura 1.1 ilustra a explanação supracitada.

Portanto este projeto tem como principal objetivo o desenvolvimento de um sistema microcontrolado que ficará responsável por fazer os acionamentos dos motores de um elevador.

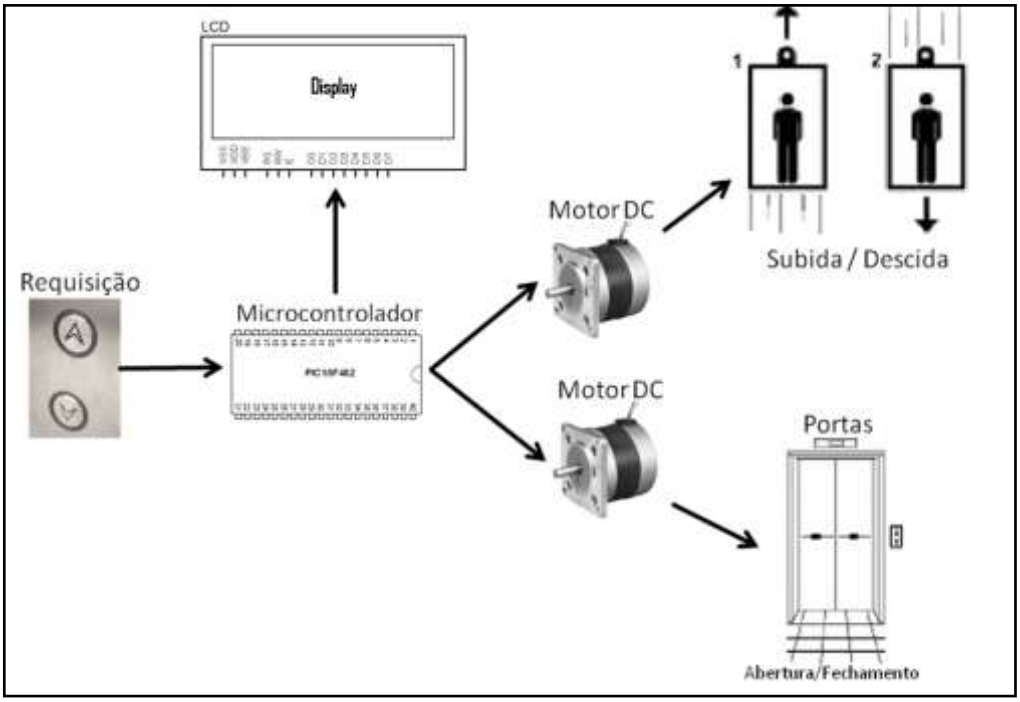


Figura 1.1 – Integração de periféricos ao microcontrolador.
Fonte: O Autor.

Portanto, pretende-se, neste trabalho, demonstrar o funcionamento de um elevador microcontrolado.

Sendo assim o microcontrolador é responsável por diversas funções, tais como:

- Armazenar a posição do elevador para que seja utilizada uma lógica em seu funcionamento.
- Enviar para o display o atual andar do elevador e em caso de movimento indicar qual o sentido (subida ou descida).
- Controlar o funcionamento dos motores de forma que os mesmos não funcionem ao mesmo tempo, visando assim a segurança de seus usuários.
- Indicar através de leds a posição do elevador e também do estado da porta.

2 REFERENCIAL TEÓRICO

2.1 Microcontrolador PIC 18F452

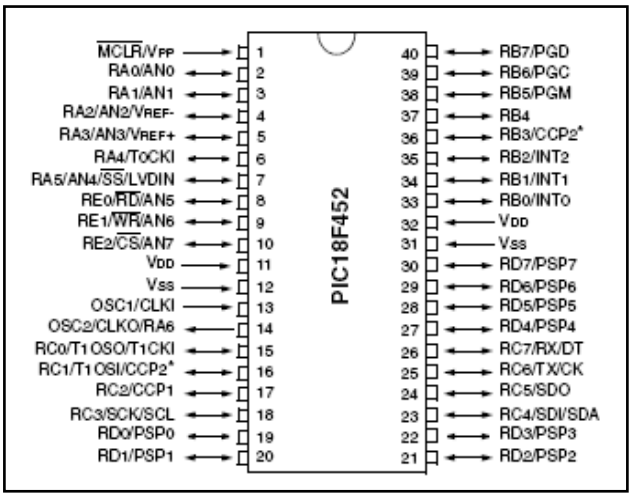


Figura 2.1 – Microcontrolador PIC 18F452 e sua pinagem.
Fonte: Manual Técnico PIC18FXX2 2007.

A figura 2.1, representa os pinos que o microcontrolador PIC 18F452 tem sendo este uma fonte de informação muito importante para a prototipagem.

2.3.1 Principais Características.

Além de responder a características anteriormente apresentadas na família PIC16XX, este por sua vez tem 16 bits de núcleo de processamento, podendo ainda trabalhar com dados de 8bits. Uma diferença circunstancial desta família de PIC's é possuir 75 (setenta e cinco) instruções em código de máquina enquanto os da família 16F possuem somente 35 (trinta e cinco).

Outra importante característica que devemos ressaltar é o fato deste microcontrolador ser otimizado para compilações na linguagem C e ainda apresentarem memória linear, enquanto os das famílias anteriores apresentavam banco de memórias. A velocidade de processamento das informações da família PIC 18FXX2 também se mostra maior do que nas anteriores. A memória de programa apresenta 32kbytes do tipo FLASH, o que permite que seja escrita, apagada e lida

através de corrente elétrica, não necessitando de nenhuma maneira especial de gravação, permitindo ainda em média 1000 regravações na mesma.

Este microcontrolador foi projetado para consumir pouca energia, consumindo até 0,6 mA, entre 2V a 5,5V, sendo o seu encapsulamento DIP com 40 pinos.

- AUSART (*addressable universal synchronous assynchronous receiver transmitter*).
- 8(oito) canais de conversão A/D a 10bits.
- MSSP (*máster synchronous serial port*) para SP1 e I2C.
- PSP (*parallel slave port*).
- 4 timers sendo estes 3 de 16 bits enquanto 1 de 8 bits.
- 2 módulos CCP (capture compare PWM).

.[Manual Técnico PIC 18XX2]

2.3.2 Clock.

Verificado que o clock que controla a velocidade de resposta do microcontrolador, alterando a freqüência em que o mesmo é operado. Este por ser obtido através de componentes denominado oscilador. Neste existe um oscilador interno, mas existe também a possibilidade de se trabalhar com um oscilador externo, sendo acoplando nas portas OCS1 e OCS2. O PIC consegue dividir este clock em quatro fases, que não tem dependência entre elas, estas por sua vez dão origem a um ciclo de máquina, ou ciclo de instrução. Conforme a Figura 2.2.

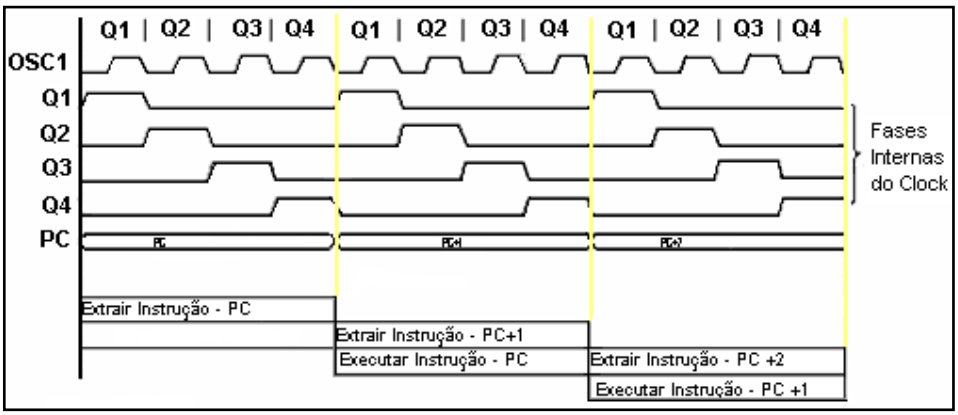


Figura 2.2 – Ciclo de máquina.
Fonte: Souza, 2003.

Como informado anteriormente o PIC tem a capacidade de executar *PIPELINE,* durante a leitura do código de instrução da memória de programa, é executado um ciclo de máquina, durante a decodificação e execução são feitas no ciclo seguinte. Sendo que ao mesmo tempo ele executa uma instrução, faz leitura do código da próxima instrução, quase que em todas as instruções, tendo como exceção a primeira instrução, ou nas instruções de desvio. Sendo gasto somente um ciclo de máquina para o seu processamento. [Souza, 2003]

2.3.3 Oscilador.

São possíveis quatro configurações diferentes no microcontrolador PIC para osciladores, sendo estas *LP*, *XT*, *HS* e *RC*. Para se configurar qual modo de oscilador será utilizado é feita a configuração por dois bits, do registrador 0X2007.

Para a utilização dos Osciladores nos modos LP, XT ou HS é necessário o acoplamento de cristais externos nas portas OSC1 e OSC2, mostrado na figura abaixo.

Existe um modo de oscilação que seria a RC, esta consiste em ligar um capacitor e uma resistência, junto a porta OSC1 para fazer de maneira a utilizar o oscilador interno do microcontrolador PIC18F452.

Visto que o valor dessa resistência é muito importante para o correto funcionamento do oscilador, sabe se que para um valor muito baixo o oscilador pode ficar instável, muitas vezes parando de oscilar. Enquanto para um valor muito elevado dessa resistência este oscilador acaba por se formar muito sensível ao ruído externo, sendo expressamente recomendado pela sua fabricante que este valor esteja entre 3kΩ e 100kΩ. Com a finalidade de ser aumentado sensivelmente a estabilidade, sendo também indicado o uso de uma capacitância acima de 20pF. [Manual Técnico PIC 18FXX2]

2.3.4 CPU, unidade central de processamento.

Esta por sua vez, liga todas as partes do microcontrolador por barramento de dados e de endereço, mostrado na figura 2.3 Tendo como principal função decodificar e executar as instruções do código em sua memória.

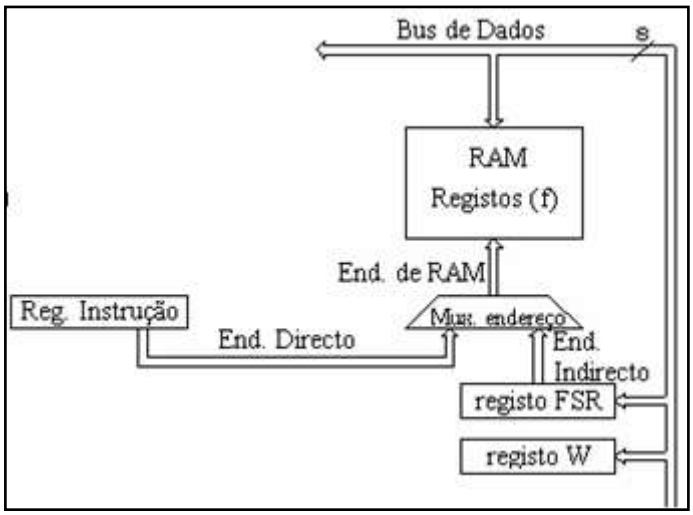


Figura 2.3 – Barramento de endereços e dados.
Fonte: Pereira, 2007.

Uma instrução é o mesmo que um conjunto de execuções que devem acontecer em tempo predeterminado, ou em condição específica. Devido esta necessidade a CPU tem que ter ligação direta com todos os outros componentes do microcontrolador, indo de memórias até as portas, dessa maneira podendo garantir rápida resposta utilizando o barramento de endereços e de dados.

Para os processos aritméticos, como adição, subtração, deslocamento de bit, é toda responsabilidade da ALU. [Vitor Amadeu de Souza]

2.3.5 Memória.

Basicamente o PIC 18F452 tem sua memória dividida em dois seguimentos as memórias de dados e a memória de programa. Sendo que cada tipo de memória tem um acesso exclusivo ao microcontrolador o que acaba por garantir o acesso simultâneo pelo microcontrolador e é esta característica que permite a ocorrência do PIPELINE.

Sendo que a memória de programa utiliza a tecnologia FLASH, com isso é possível fazer inúmeras gravações, cerca de 1000 vezes, com a finalidade de teste, e correções antes que o mesmo seja integrado definitivamente a um circuito. Esta memória por sua vez possui algumas alocações que seriam para o vetor de reset e de interrupções, conforme mostrado na figura abaixo.

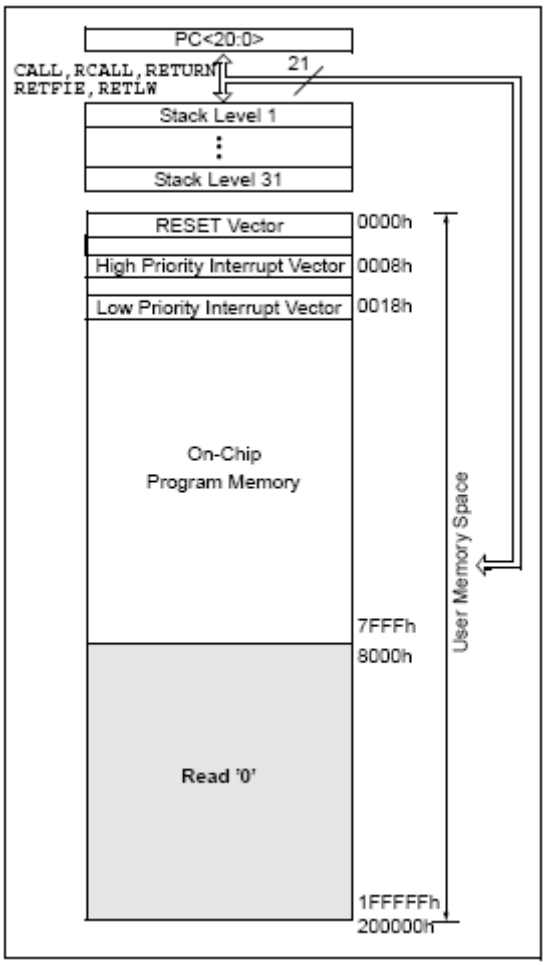


Figura 2.4 – Disposição da memória de programa no PIC 18F452.
Fonte: Manual Técnico PIC 18FXX2.

Vista distribuição da memória de programa, agora verificaremos como o mesmo trata a memória de dados, este tipo de memória se subdivide em dois tipos:

- Registradores de uso geral
- Registradores de uso específico.

A distribuição física da memória do microcontrolador em questão, PIC 18F452, constitui somente um bloco otimizando o seu uso com a linguagem de

programação C. Essa mesma distribuição em somente um bloco acaba por corrigir um problema encontrado em famílias anteriores desde microcontrolador, que tinham diversos blocos, onde várias vezes um bloco acabava por ser totalmente utilizado e outro ainda vazio, muitas vezes tendo que ser utilizado algum recurso avançado de programação para poder ser utilizado em outro banco, o que fazia com que o programador tivesse extremo trabalho no momento da programação, pois os valores tinham que ser passados manualmente, e o que acabava por dificultar sua programação em C.[Fábio Pereira, 2007]

2.3.6 Temporizador.

Os temporizadores, ou comumente conhecidos com Timers, exercem uma função muito importante para o funcionamento correto do microcontrolador, através deles é possível fazer a realização de uma tarefa tomando como grandeza o tempo, ou seja, fazendo a medição do tempo.

Para isso são utilizados registradores que tem incrementado o seu valor a cada clico de tempo. No PIC18F452 existem quatro temporizadores, TIMER0 este possui 8btis enquanto os outros TIMER1, TIMER2 e TIMER3 possuem 16bits. Visto a distribuição da memória de programa, agora verificaremos como mesmo trata a memória de dados, este tipo de memória se subdivide em dois tipos, em registradores de uso geral e em registradores de uso especifico. [Manual Técnico PIC18FXX2]

2.3.7 Interrupções

A interrupção se faz necessária quando precisamos interromper uma tarefa quando se atinge um determinado resultado, em outras palavras é uma maneira do microcontrolador realizar ações quando recebe uma resposta a um evento externo.

Geralmente, toda interrupção é diretamente associada a dois bits, um responsável por ligar ou desligar a interrupção, e o outro responsável por mostrar que uma interrupção aconteceu.

Sendo o PIC18F452 que possui múltiplas interrupções dessa maneira ainda podemos classificar cada interrupção com um grau diferente de prioridade o que acaba por dar mais autonomia e respostas ao dispositivo. No PIC 18F452 temos dez registradores responsáveis pelo acionamento dessas interrupções são eles:

- RCOM.
- INTCON.
- INTCON2
- INTCON3.
- PIR1, PIR2.
- PIE1, PIE2.
- IPR1, IPR2.

Na interrupção o programa armazena a posição atual no *stack*. Fazendo o direcionamento para o endereço 0004H, onde deve ser incluída pelo programa a rotina que fará o tratamento da interrupção. Para o correto funcionamento da interrupção é necessário que se conheça o motivo da interrupção, isto deve ser tratado no programa. Ainda se faz necessário que seja armazenado os registradores importantes, para não se perder o seu estado atual.

Conforme informada anteriormente também se faz muito importante para que se ocorram as interrupções que sejam setados em 1 o bit de interrupção global, pois se as mesmas se encontrarem em 0, qualquer interrupção não fará diferença. [Manual Técnico PIC18FXX2]

2.4 Motores de Corrente Contínua.

2.4.1Motor Elétrico.

O motor elétrico é um elemento de trabalho que contém a capacidade de converter a energia elétrica em energia mecânica de rotação. Sendo que em um motor elétrico se distingue especialmente em duas partes:

- *Estator,* sendo este o conjunto de elementos fixados à carcaça da máquina.
- *Rotor,* todos os outros elementos que estão fixados em torno do eixo, ou seja, internamente ao estator.

O rotor é basicamente formado pelos seguintes elementos:

- *Eixo de Armadura,* este responsável pela transmissão da energia mecânica para fora do motor, feitos pelo suporte dos elementos internos do rotor e pela fixação ao estator por meio de rolamentos e mancais.
- *Núcleo da Armadura,* geralmente compostas por laminas metálicas, isoladas umas das outras, com ranhuras axiais na sua periferia para a colocação dos enrolamentos da armadura.
- *Enrolamento da Armadura,* estas são bobinas isoladas entre si e ligadas eletricamente ao comutador.
- *Comutador,* um anel formado por segmentos de cobre, que também são isolados entre si, mas ligados eletricamente às bobinas do enrolamento da armadura.

Enquanto o estator é formado por:

- *Comutador,* um anel formado por segmentos de cobre, que também são isolados entre si, mas ligados eletricamente às bobinas do enrolamento da armadura.
- *Carcaça,* que é todo o suporte para o rotor.
- *Enrolamento de campo são* as bobinas responsáveis por gerar um campo magnético intenso entre os pólos.
- *Pólos, ou sapatas polares,* responsáveis por fazer a distribuição do fluxo magnético produzido pelas bobinas de campo.
- *Escovas,* barras que permanecem em contato permanente com o comutador, quase que exclusivamente feitas de carvão ou de grafite.

Basicamente os motores elétricos apresentam esta mesma formação de componentes, podendo variar conforme sua utilização em alguns aspectos. Como por exemplo, a bobina de armadura esta ligada diretamente ao estator ao invés do rotor, e podendo acontecer com outros componentes. Existem ainda motores elétricos com a ausência de escovas

Os motores elétricos ainda podem ser classificados de acordo com a sua característica de funcionamentos, então encontramos motores de corrente contínua, corrente alternada e motores especiais. Ainda sendo encontrando uma subclassificação dentre eles. [Material do Curso Pós Técnico em Automação Industrial – Motores – CEFET/SC]

- Motores de corrente contínua.
    - Motores Série.
    - Motores Paralelo.

  - Motores Composto ou Misto *Carcaça,* que é todo o suporte para o rotor.

- Motores de corrente alternada.
  - Motores síncronos.
  - Motores assíncronos.
- Motores Especiais.
  - Servomotores.
  - Motores de passo.
  - Motores Universais.

O fenômeno eletromagnético que permite o funcionamento do motor refere-se que sempre existe uma força mecânica no fio quando o mesmo conduz eletricidade, este contido em um campo magnético. Está descrita pela *lei da força de Lorentz,* sendo perpendicular ao fio e ao campo magnético.

A lei da força de Lorentz é a força atuante em uma partícula, carregada eletricamente, quando esta viaja em um campo eletromagnético. Sendo que associando as leis de Newton e as equações de Maxwell, encontraremos a equação da força de Lorentz, sendo apresentada sua equação na Figura 2.5.

$$F = q(E + V \times B)$$

Equação: 2.1 – Equação da força de Lorentz.

Onde os componentes desta são:

- *F*, força do campo eletromagnético.
- *E,* campo elétrico.
- *B,* campo magnético.
- *q,* carga elétrica.
- *v,* velocidade da carga.

Como este trabalho esta sendo montado sobre os motores de corrente contínua, segue na Figura 2.5 esquema de composição de um motor de corrente contínua de dois pólos.

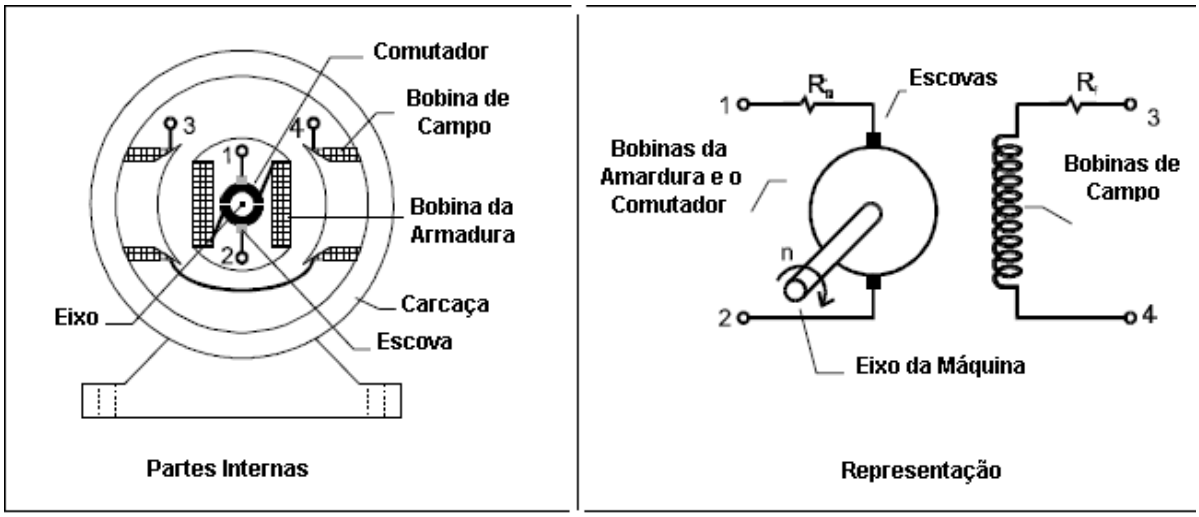


Figura 2.5 – Corte em motor elétrico de corrente continua de dois pólos.
Fonte: Material do Curso Pós Técnico em Automação Industrial – Motores – CEFET/SC

O motor de corrente contínua, apresenta quatro terminais acessíveis, sendo que dois destes são para as bobinas de armadura, que em motores menos potentes podem ter ímãs permanentes no lugar dessas bobinas. Sendo que para o funcionamento elementar de um motor de corrente contínua está baseado na força mecânica que age sobre o condutor quando esta sobre um campo magnético, quando sobre o mesmo circula uma corrente elétrica.

Como podemos verificar na figura 2.5 as forças da bobina 1, são iguais e opostas, o que acaba por anular qualquer força de rotação, mas sobre as bobinas 2, 3 e a bobina 4 encontra-se forças que acabam por impulsionar o rotor para girar, que leva com ela a bobina 1 para outra posição que lhe é propicia a exercer agora uma força de rotação, isso se torna um clico entre as bobinas sendo este principio que não deixa que as forças se anulem e que não haja movimento.

Os motores de corrente contínua também pode ter como principio de funcionamento a ação de forças magnéticas sobre o rotor, tais forças são geradas pela introdução do campo magnético criado pelas bobinas de campo como campo criado pelas bobinas da armadura, conforme exemplificado na figura 2.6. Onde

podemos então observar que o comutador possui a função de estar invertendo o sentido da corrente na bobina de armadura, quando estão em 90° e 270 °, podendo assim dar continuidade ao movimento de rotação.

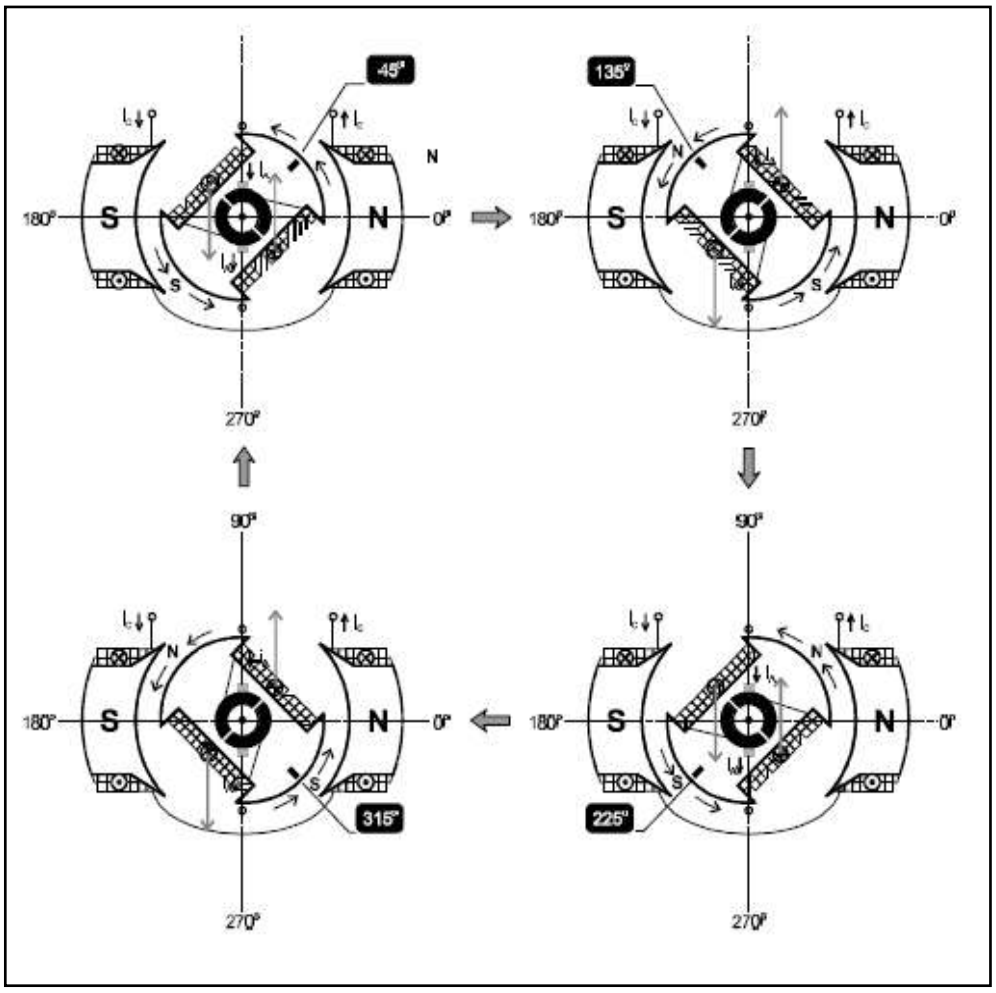


Figura 2.6 – Principio de funcionamento.
Fonte: Material do Curso Pós Técnico em Automação Industrial – Motores – CEFET/SC

Um fenômeno físico rege todo processo, já citado, mas ainda não explicada seria a FEM, (*Força Eletromotriz*). Conforme o mostrado na Figura 2.7, referência a ação motora, força, para o sentido do campo e da corrente da armadura. Essa força que é efetuada no condutor faz que o mesmo se movimente no sentido do campo magnético, formando assim uma força eletromotriz, que é induzida no condutor, motor. O sentido dessa FEM é oposta ao representado na figura, para os mesmo sentidos de movimento e de campo. Ao aplicar se esta FEM induzida ao condutador

pode ser verificada que a mesma vai se opuser ao sentido da circulação da corrente geradora desta força, por este motivo ela é denominada de *força contra-contraletromotriz* (FCEM*)*.

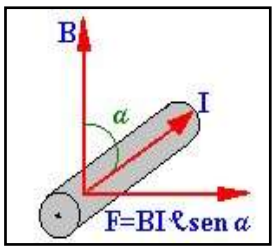


Figura 2.7 – Sentido de FEM.
Fonte: Motores Elétricos: Manutenção e Testes

2.5 Elevador história, componentes e funções.

2.5.1 História.

Um elevador tem como objetivo elevar objetos e pessoas, sendo assim um meio para o transporte vertical.

Como um dos primeiros elevadores que se tem notícia atualmente, foi criado por Vitrúvio no século I a.C. em Roma. Este por sua vez utilizava varias roldanas e eram movidas por tração humana, animal ou ainda pela força da água. Somente em 1853 o Elisha Graves Otis, este norte americano, o primeiro a utilizar um dispositivo para acionamento dos freios em caso de emergência.

Em 1889, foi a data aproximada do aparecimento dos primeiros elevadores elétricos. Uma curiosidade interessante, que levando em consideração a sua utilização comparada com os diversos meios de transporte existentes atualmente, o elevador se faz o mais seguro de todos eles com larga vantagem.

2.5.2 Componentes e suas funções.

Os atuais elevadores são formados por quatro partes, cada parte com diversos elementos, são:

- Casa de máquinas, na qual é composta pela máquina de tração, base integrada, quadro de comando, regulador de velocidade e quadro de força.

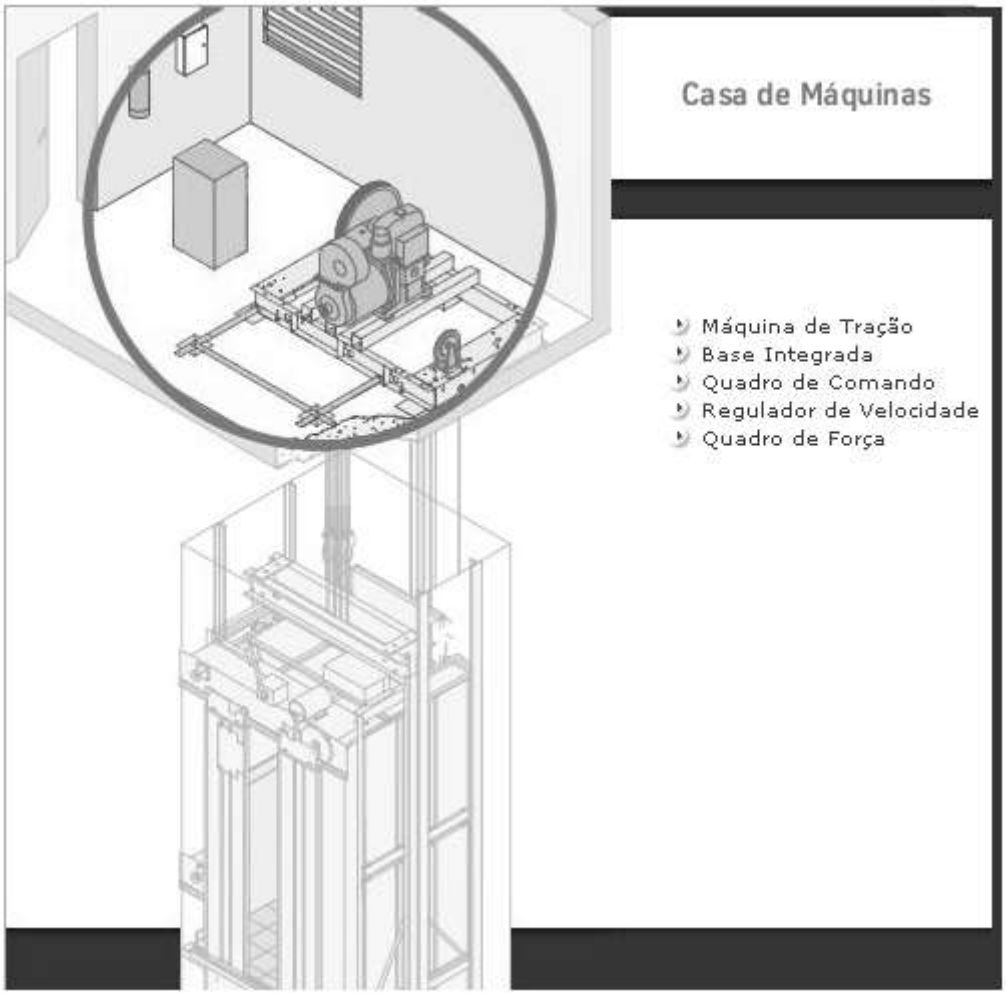


Figura 2.8 – Casa de Máquinas.
Fonte: http://www.thyssenkruppelevadores.com.br

- Caixa de corrida e poço, amortecedores, cabo de tração, guias, contrapeso e limites de percurso.

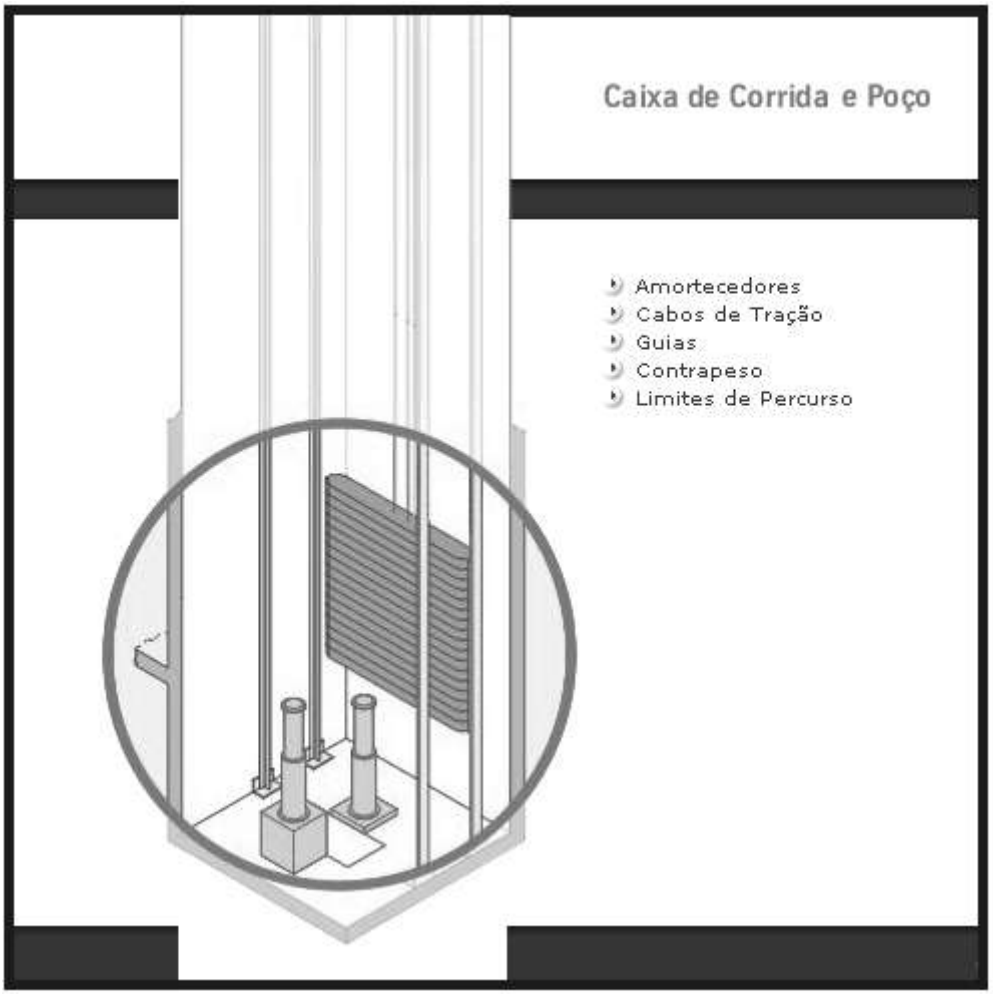


Figura 2.9 – Caixa de corrida e Poço.
Fonte: http://www.thyssenkruppelevadores.com.br

- Cabina, esta por sua vez é formada pela corrediça da cabina, painel de operação, indicador de posição, operador de portas, cortina(protetor de soleira, freio de segurança e portas da cabina.

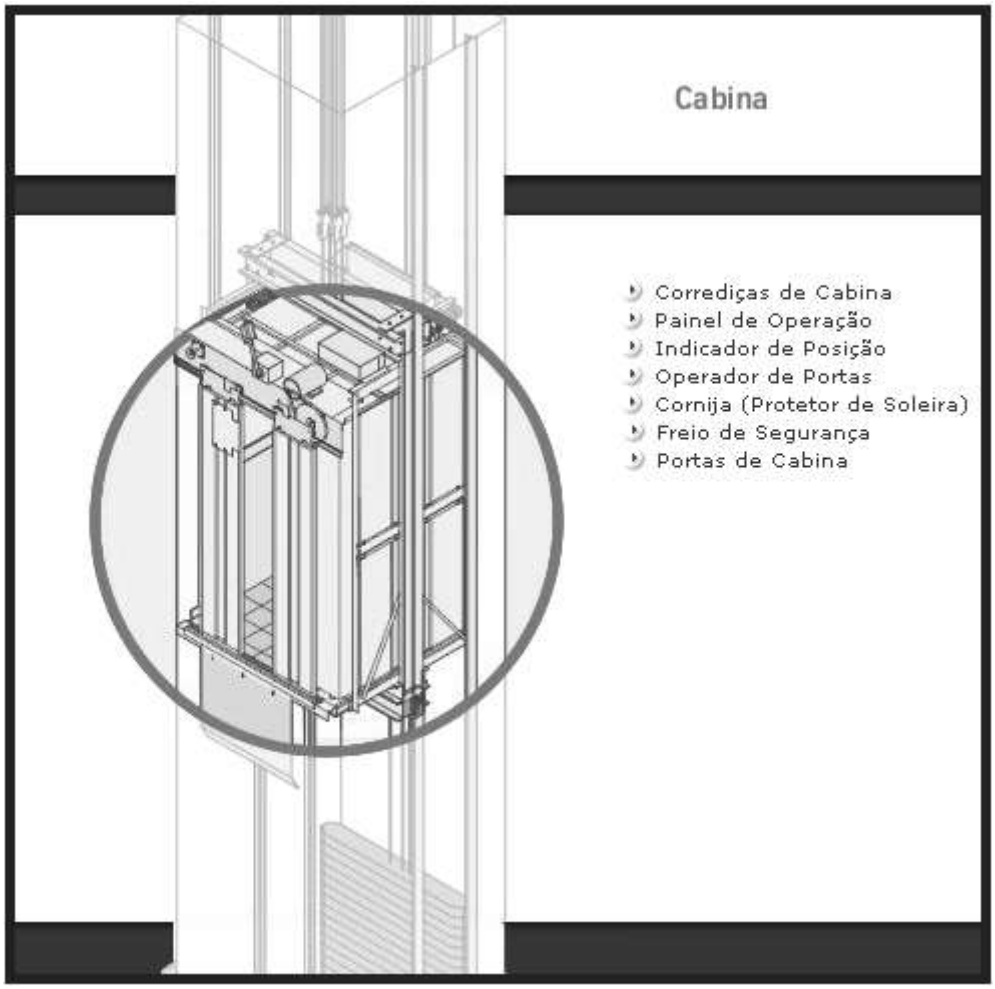


Figura 2.10 – Cabina.
Fonte: http://www.thyssenkruppelevadores.com.br

- Pavimento, formado por indicador de posição, botoeira e portas de pavimento.

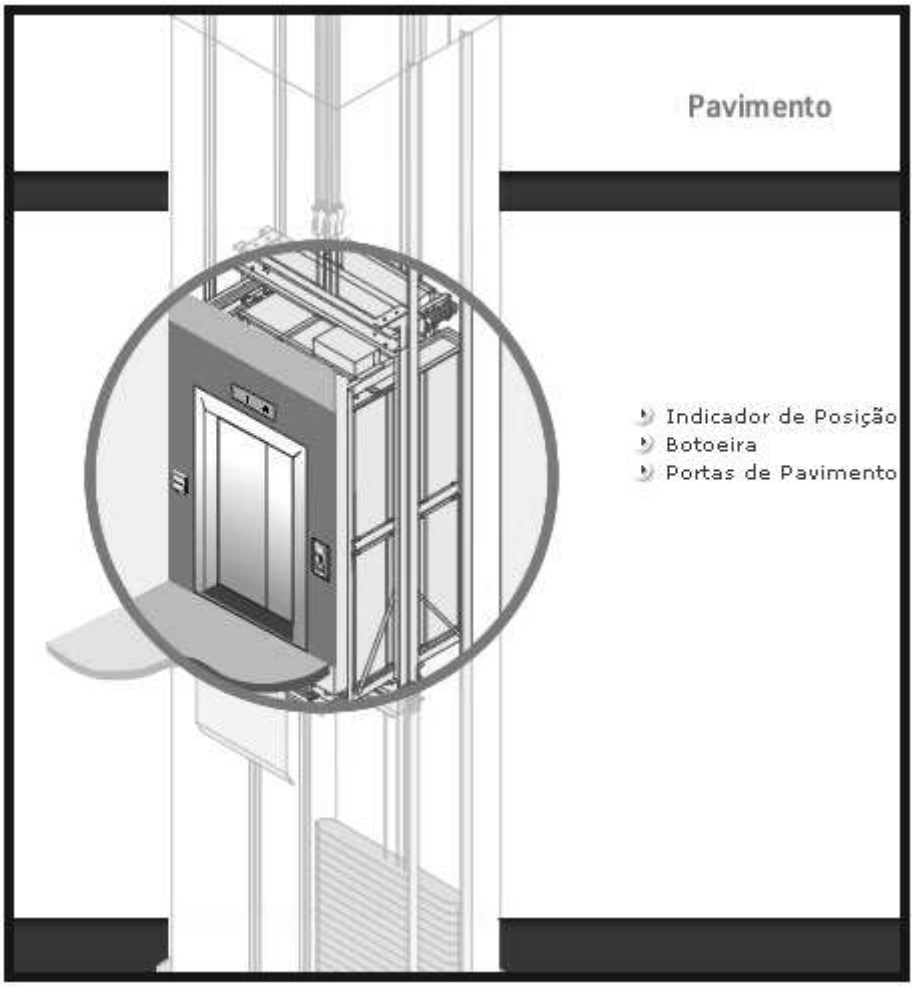


Figura 2.11 – Pavimento.
Fonte: http://www.thyssenkruppelevadores.com.br

Lembrando pode haver acréscimo ou redução de componentes, dependendo do critério de segurança adotado, mas continua muito semelhante à estrutura apresentada.

3. PROTÓTIPO HARDWARE E SOFTWARE.

3.1 Hardware.

O projeto do elevador controlado via microcontrolador, se baseia em um circuito capaz de desempenhar a função de um elevador com 4 (quatro) andares, onde cada andar poderá ser selecionado por 4 (quatro) botões, estes botões estão dispostos na placa de circuito impresso, representando os botões internos e externos do elevador. A sinalização de cada andar é por meio de leds(light emissor diode) assim como a sinalização de abertura e fechamento de porta

Neste projeto, para desempenhar a função de um elevador, foi todo desenvolvido utilizando como base o PIC 18F452.

O programa usado para a geração e simulação dos circuitos eletrônicos foi o Proteus em sua versão 7.21, conseqüentemente as imagens exibidas aqui fizeram parte do desenvolvimento do circuito no mesmo software. A imagem do layout do circuito foi gerada no Aeris, módulo do Proteus a fabricante desses dois softwares é a Labcenter Eletronics.

3.1.1 Alimentação do circuito.

Uma parte muito importante em qualquer equipamento eletrônico é sua alimentação, o mesmo se mostra crucial, pois como no caso deste projeto o PIC pode apresentar mau funcionamento e até travamento devido a alimentação elétrica irregular. A parte do circuito elétrico responsável pela a alimentação do circuito elétrico está representada na figura 3.1.

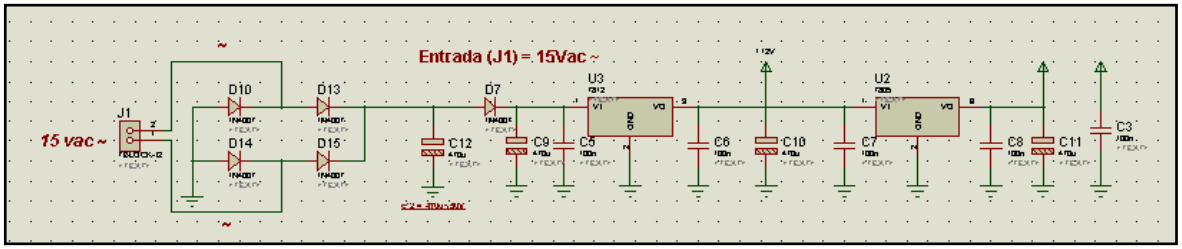


Figura 3.1 – Retificação e regulagem de alimentação.
Fonte: O Autor.

Inicialmente, o circuito é alimentado for uma fonte com 15V e 1A de saída. Como essa alimentação ainda não se mostra estável suficiente, e ainda podendo apresentar inversão de polarização, a qual poderia queimar diversos componentes e prejudicaria sistematicamente o funcionamento do elevador. Sua primeira parte é formada por um arranjo de diodos que tem por objetivo fazer uma retificação na entrada, verificado que a propriedade do diodo é permitir a passagem da corrente elétrica em somente uma direção, o mesmo acaba por garantir que independente da polarização de entrada a mesma é sempre garantida na saída desse arranjo de díodos.

Nesta parte do circuito ainda temos presente a utilização de dois reguladores de tensão, o primeiro reduz sua entrada de 15V para 12V em sua saída, onde estaremos ligando diretamente ao pino de alimentação dos relés dos motores, este com sua funcionalidade explicada adiante, enquanto o segundo recebe em sua entrada 12V e em sua saída temos 5V, responsáveis pela alimentação do microcontrolador ligando a porta VCC do mesmo e do display.

3.1.2 Botões internos e externos.

Toda parte de comando do elevador, uma vez feita toda sua implementação, tem como entrada de dados somente os botões do mesmo, por onde serão realizadas as requisições de serviço. Abaixo temos uma figura demonstrativa da ligação do mesmo dentro do circuito. A implementação de resistências em série aos botões acaba por garantir que existe circulação de corrente elétrica somente quando o botão estiver acionado, com a corrente adequada.

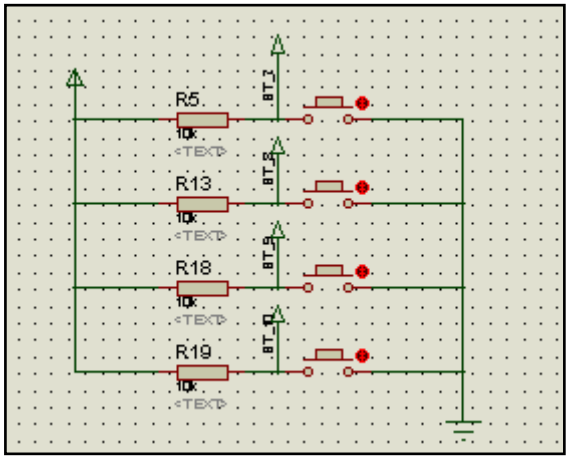


Figura 3.2 – Botões internos e externos.
Fonte: O Autor.

3.1.3 Controle do motor de movimentação.

Uma das partes mais importantes desse projeto é o controlo do motor de movimentação pelo microcontrolador. O microcontrolador por sua vez faz o acionamento de um relé que por sua vez fornece a alimentação necessária para o acionamento do motor, visto que o PIC pode fornecer um pouco mais de 5 V, mas o motor utilizado neste projeto necessita para correto funcionamento uma alimentação de 12 V.

Ainda verificado a necessidade de descer e subir, o mesmo motor tem um arranjo com a utilização de relés diferentes, este tem por objetivo fazer a inversão da polarização do motor, para que possa desenvolver os dois movimentos diferentes de subida e de descida. Apenas observando que cada relé é acionado por uma porta diferente do microcontrolador, e ainda temos em código a preocupação de nunca fazer o acionamento dos dois ao mesmo tempo. Abaixo a figura 3.3, retirada do Proteus(software de simulação) exemplificando o arranjo utilizado neste projeto.

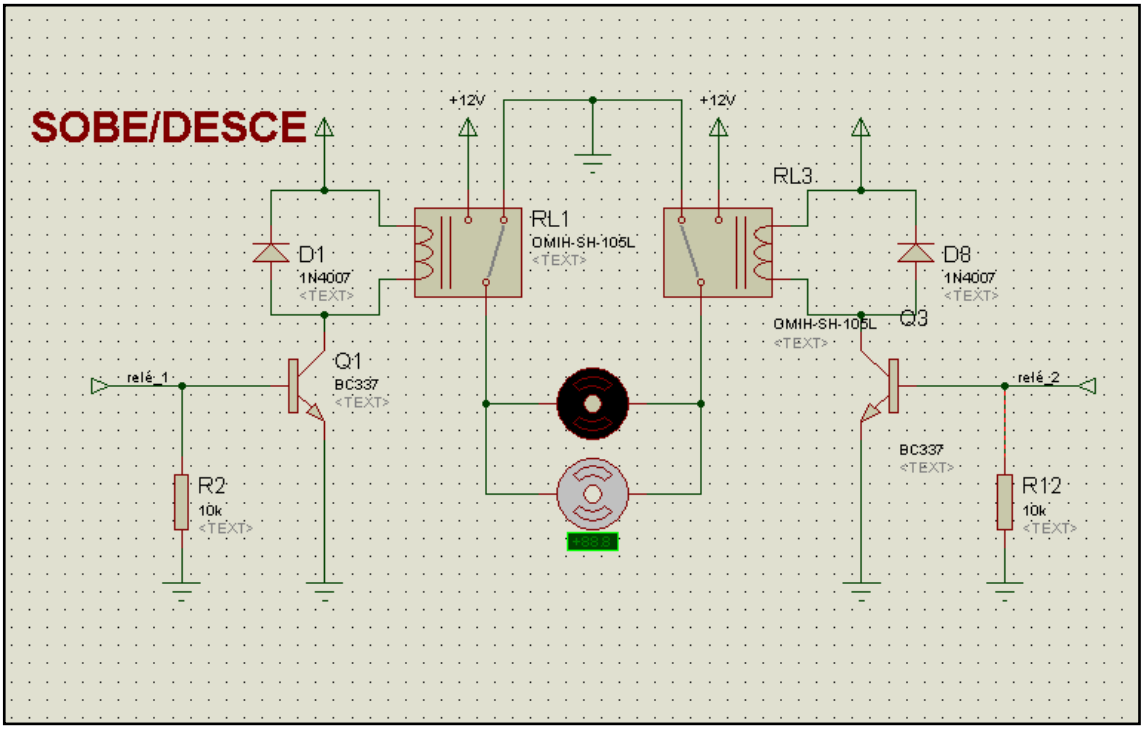


Figura 3.3 – Arranjo motor e reles para movimentação.
Fonte: O Autor.

3.1.4 Controle do motor de abertura e fechamento da porta.

Esta parte por sua vez, tem sua montagem física quase que idêntica a do motor de movimentação, exceto pela sua ligação em portas diferentes do microcontrolador. Seu caso de uso é bastante parecido com o motor de movimentação, ele também tem a necessidade de rotação nos dois sentidos motivo pelo qual o mesmo também utiliza o mesmo arranjo de reles. O mesmo também esta utilizando um motor de 12 V.

Observação importante neste projeto nunca será acionado os dois motores ao mesmo tempo. Segue na Figura3.4 o arranjo do mesmo.

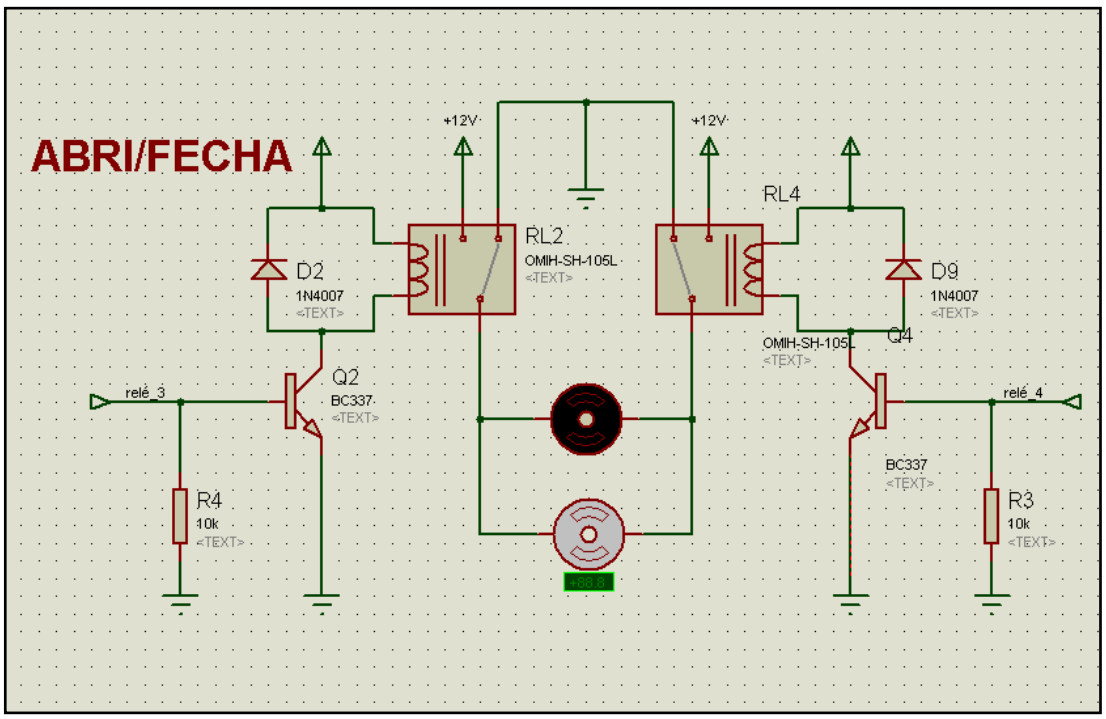


Figura 3.4 – Arranjo motor e relé de abertura e fechamento de porta.
Fonte: O Autor.

3.1.5 Indicador de nível e sinalizador de estado de porta e display.

Para uma melhor utilização de um elevador se faz necessário a indicação de nível, indica em qual andar se encontra o elevador, para desempenhar essa função estaremos utilizando quatro leds cada um indicando um andar. Essa identificação é feita pelo microcontrolador armazenado em memória, figura 3.5 tem a representação do esquema eletrônico dos leds de nível.

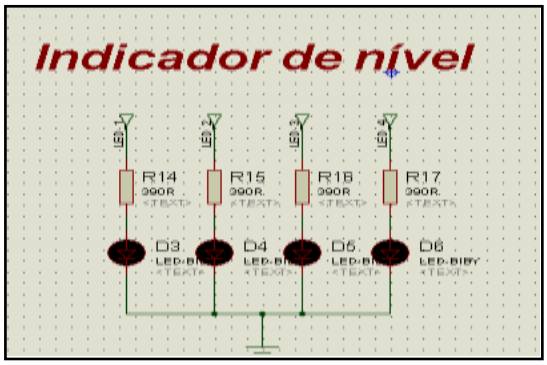


Figura 3.5 – Indicador de nível.
Fonte: O Autor.

Ainda com a finalidade de melhorar a relação de posicionamento, temos a necessidade de saber em qual sentido o elevador se movimenta, para isso temos um display de 2 linhas com 16 caracteres, este por sua vez recebe diretamente as informações do microcontrolador, tais como andar inicial e final, dessa informação conseqüentemente fica implícito em qual sentido o elevador esta se locomovendo. A figura 3.6 representa referente a ligação do display ao circuito.

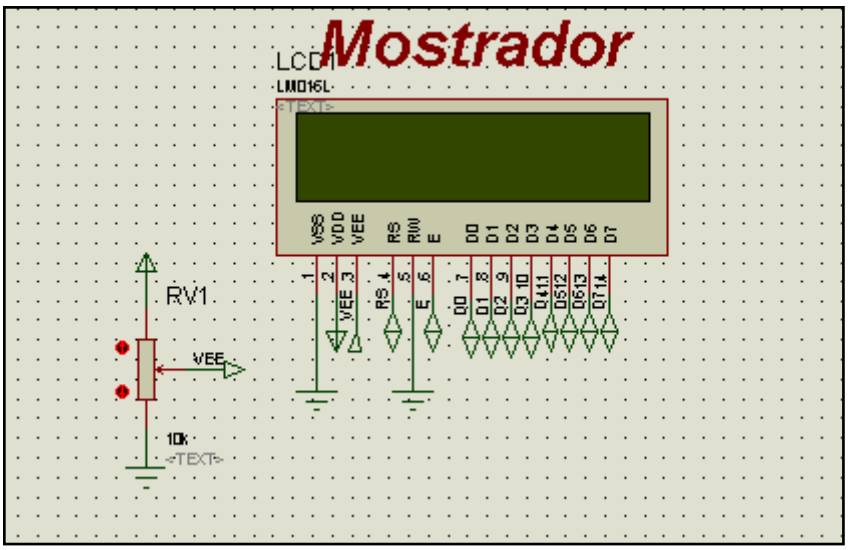


Figura 3.6 – Esquema de ligação do display ao circuito e microcontrolador.
Fonte: O Autor.

3.1.6 Microcontrolador PIC 18F452 e sua integração no circuito.

Tudo praticado neste projeto só foi possível devido a integração do PIC ao circuito, este atua como o cérebro do projeto, interagindo com situações externas e fazendo os ajustes necessários para a melhor utilização do elevador, logicamente que tudo respeitando sua programação.

Neste projeto foi utilizada a programação em C, verificado a melhor receptividade de códigos gerados nessa linguagem na família de 18F. Visando um melhor desempenho do mesmo para a temporização do mesmo está sendo utilizado um cristal de 4Mhz. Temos empregado nesse projeto um arranjo básico com um botão para realizar o reset do microcontrolador.

A comunicação entre o microcontrolador e seus periféricos é feita através das próprias portas do microcontrolador, essa característica acaba por viabilizar a

interação do microcontrolador com os diversos componentes dos circuitos e seus periféricos. Neste projeto temos a seguinte disposição de portas.

OSC1 e RA6(OSC2) utilizado para calibrar o temporizador.

RB0, botão 1.

RB1, botão 2.

RB2, botão 3.

RB3, botão 4.

RA0, led 1.

RA1, led 2.

RA2, led 3.

RA3, led 4.

RC0, rele 1.

RC1, rele 2.

RC2, rele 3.

RC3, rele 4.

RC6, led aberto.

RC7, led fechado.

RD0, pino display 0.

RD1, pino display 1.

RD2, pino display 2.

RD3, pino display 3.

RD4, pino display 4.

RD5, pino display 5.

RD6, pino display 6.

RD7, pino display 7.

Muito importante salientar que neste projeto foram utilizados as portas do microcontrolador somente como entrada e saída, sendo assim não foi utilizada nenhuma configuração especial dessas portas.

Na Figura 3.7 um esquema das ligações do microcontrolador.

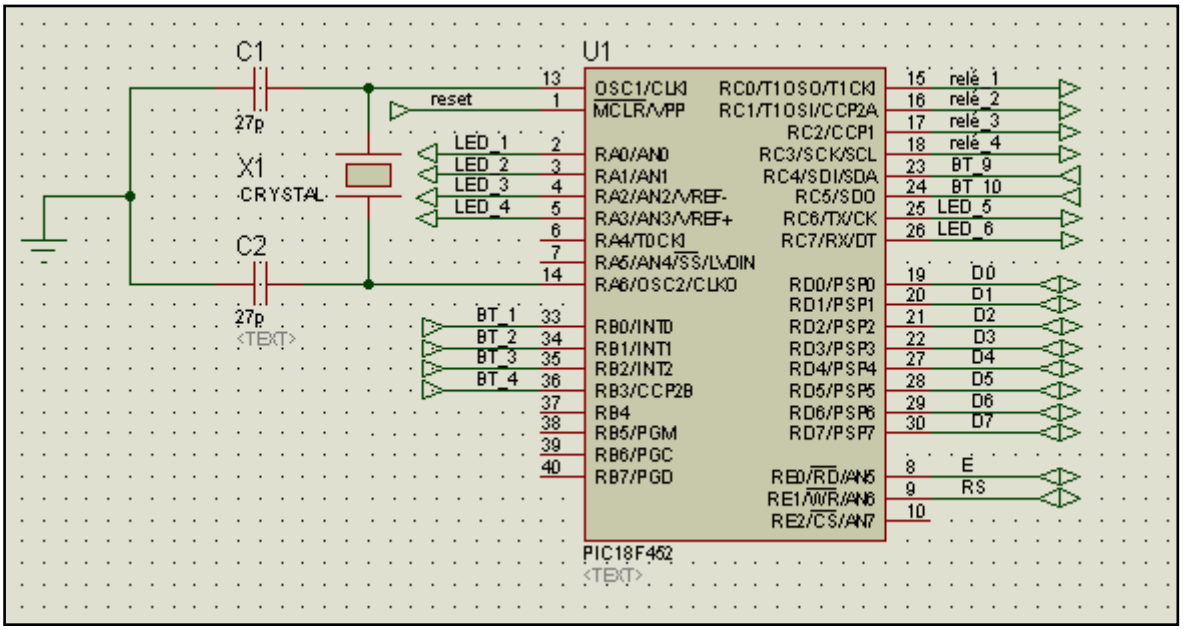


Figura 3.7 – Interligação de todos os pinos aos seus periféricos.
Fonte: O Autor.

3.2 Entrada, saída e processamento.

3.2.1 Entrada.

Este projeto acaba por encontrar como módulo de entrada para o microcontrolador:

Botões externos e internos, utilizados para selecionar andares tanto externamente como internamente. Os botões se encontram na PCB do circuito.

Botão de RESET, O botão de RESET é utilizado para reset manual da programação do microcontrolador, quando o mesmo apresenta avarias (mau

funcionamento). OBS.: Como foi utilizado o botão de reset a programação não possui WDT (Watch Dog Timer).

3.2.2 Processamento.

O processamento das informações é realizado pelo microcontrolador 18F452, no qual as interpretam e as enviam para os pinos de saída, conforme programação previamente especificada.

3.2.3 Saída

Os módulos de saída do microcontrolador são:

Relé de saída, existem 4(quatro) relés de saída: 2(dois) para subir e/ou descer o elevador enquanto os outros dois são utilizados para abrir e/ou fechar a porta do elevador. Lembrando que a esse relé ao receber o sinal do microcontrolador o mesmo faz o chaveamento necessário para liberar 12V para o motor elétrico.

Led’s para sinalização de andares: São utilizados para indicação do nível ao qual se encontra o elevador.

Led’s para sinalização de abertura e fechamento da porta do elevador: São utilizados para alertar quanto á abertura e/ou fechamento da porta do elevador.

Display LCD 16x2: Utilizado como IHM (Interface Homem Máquina). O display LCD mostra o nível atual do elevador e o nível no qual se deseja chegar.

3.3 Software.

Na programação em C temos com abertura de comentário /* e para fechamento */, os comentários por sua vez não são necessários para o

funcionamento do código em nível de hardware e software, mas este se torna um agente facilitador do entendimento do código.

Na primeira parte do código temos o local onde iniciamos as configurações do microcontrolador, essa configuração permanece durante todo o tempo que ocorre o processo no microcontrolador, além dessas características gerais nessa primeira parte temos a declaração de bibliotecas, estas têm como objetivo facilitar a programação, por já ter definido diversos tipos rotinas e funções com características diferentes para serem usadas sem a necessidade da programação dessas funções novamente.

As bibliotecas xcld.h e xcld.c têm por objetivo fazer a comunicação com o display, sendo necessário somente que seja declarado uma variável e fazer a chamada dessa variável pela sua função que a mesma será exibida na tela do display, essa comunicação é de 8 (oito) bits, disponível para download no site www.sabereletronica.com.br/downloads .

As primeiras configurações passadas para o microcontrolador nessa parte são a seleção da freqüência do oscilador de 4Mhz, desabilitar o watch dog timer visto esta ser incompatível com a função delay, desabilitar o brownout reset, habilitação do powerup(assim que ligado a eletricidade o mesmo já inicia o seus processos.)

Trata-se agora parte da definição do tempo do display, a comunicação pode ser feita entra o microcontrolador e o display em 15ms por segundo, este tempo é definido pela própria fabricante do display, esta rotina foi disposta pela fabricante do LCD no site citado acima, com a única observação de se declarar se a mesma vai ser feita de forma continua ou estática, a que foi escolhida nesse projeto foi estática, assim não podendo passar de 16 caracteres por linha, limite físico do display.

A função tela_inicial, esta é a primeira rotina a ser executada limpando tudo que aparece estiver armazenado ou exibido no display e exibindo a mensagem inicial desejada, lembrando que esta deve sempre respeitar o limite de 16 caracteres por linha.

O elevador deve sempre iniciar no quarto andar, pois como posição inicial assim que ligado o microcontrolador reconhece que este se encontra neste andar citado.

Neste momento o microcontrolador já esta fazendo a verificação das portas de entrada, esta parte será explicada mais a frente, assim que uma requisição chega ao microcontrolador, através do botão, o microcontrolador chama a função tela_andar que apaga o que estava sendo exibido no display e mostra em qual andar o elevador se encontra e para qual andar o elevador vai.

Depois de ligado o mesmo já está verificando se existe o acionamento de algum botão, caso o mesmo ocorra, ele já tem setado como posição atual, esse reconhecimento de posição atual se faz da seguinte maneira, se for a primeira vez que o elevador está sendo acionado o mesmo reconhece que se encontra no quarto andar, mas caso o mesmo já tenha sido acionado alguma vez, ele reconhece como posição o ultimo destino.Uma vez que ele tem o conhecimento de sua posição atual o mesmo verifica se o destino é maior ou menos que sua posição.

Caso o destino seja maior que a posição atual, para saber por quanto tempo ele tem que acionar o motor, ele diminui o andar de destino pelo andar da posição atual, esse resultado é multiplicado pelo tempo de subida de um andar. Enquanto se for o contrário o andar de destino seja menor que a posição atual, ele diminui a posição atual pelo destino, e também utilizando esse resultado para multiplicar pelo tempo de descida.

O tempo de descida e subida foi obtido como uma medida do tempo total que o elevador levou para percorrer o trajeto, com esse tempo o mesmo foi dividido pela quantidade de andar, sendo assim obtido o valor para subir um andar e também o valor de descer um andar, lembrando que os mesmos tem valores diferentes pela possível diferença de peso entre a cabine do elevador e o seu contrapeso.

Da mesma maneira da função anterior temos outra função responsável por fazer o elevador descer, para isso o mesmo utiliza o mesmo critério de avaliação da função passada e faz a desativação do relé (um)1 e depois faz a ativação do relé

2(dois0, depois sua rotina executada o mesmo faz a atualização dos leds indicativos de andar.

As funções motor_abre e motor_fecha ficam responsáveis por fazer o acionamento do motor da porta do elevador, estas duas funções utilizam um determinado tempo abrindo e fechando as portas do elevador. Uma parte importante dessa função é o cuidado na desativação dos reles responsáveis por ligar o motor de movimentação o que acaba por garantir que nesse momento o elevador ainda esta parado.

Qualquer destas funções utilizadas isoladamente não faria nenhum sentido. A função com o nome de "*padrão"* fica responsável por chamar o fechamento da porta, verifica a relação entre a posição e o destino, abre a porta e seta o valor de posição igual o valor do ultimo destino.

A função principal, o que liga todas essas funções tem como padrão na linguagem de programação C ser chamada de void_main. Esta função fica responsável por setar diversas outras características no microcontrolador, inicialmente toda as portas de saída do microcontrolador foram setadas em 0(zero). Tanto as de entrada como as de saída. Encontramos muitos detalhes dessa função em seus comentários no apêndice.

Na função acha_botao, fica responsável por fazer toda a parte de monitoriamente de acionamento dos botões. Uma vez esta rotina iniciada pelo microcontrolador, qualquer botão que for acionado iria inicia as rotinas necessárias para que a requisição ocasionada pelo acionamento do botão seja atendida. Como um exemplo que pode ser seguido para todos assim que o botão do primeiro andar for apartado ele vai chamar a função "*padrão*" tomando todas as ações necessárias" como a atualização do display e dos leds que indicam o andar e a situação da porta e os respectivos acionamentos dos motores.

4 CONCLUSÃO.

4.1 Resumo e analise dos principais resultados alcançados.

Tendo como objetivo principal deste projeto a construção de um circuito eletrônico capaz de gerar os recursos envolvidos na utilização do elevador isto com uma certa “inteligência” de gestão de recursos, minimizando o impacto sobre a estrutura de um prédio e maximizando o tempo de utilização do elevador e aumentando a vida útil de seus componentes, uma vez verifica que não será feito o acionamento de seus componentes de forma desordenada.

Verificado que os objetivos citados foram atendidos tanto sob o ponto de vista experimental quanto o de software ou seja, o modelo do sistema foi montado e operou satisfatoriamente.

A implantação desse projeto exigiu a construção de uma maquete em escala reduzida de um elevador e a confecção de uma placa de circuito impresso. Lembrando que nesta maquete não foram reproduzidos componentes de segurança, somente os necessários para demonstrar o funcionamento deste elevador. O mesmo foi bem sucedido no que foi proposto.

Ele acabou por reforçar diversos conhecimentos exigidos para a realização do mesmo, como circuito eletrônicos, microcontroladores e programação em C.

4.2 Críticas.

Mesmo o circuito se mostrando eficiente para o que foi proposto, o mesmo ainda necessita de algumas modificações para melhorar ainda mais a utilização do elevador. Em caso de prédios grandes o mesmo pode apresentar algumas deficiências em seu funcionamento por não apresentar nenhuma modificação comportamental do prédio, que se refere ao horário de funcionamento de cada andar, pois em um grande prédio é comum de mais de uma empresa, cada qual com horários diferentes. Outro problema que podemos encontrar com esse projeto seria a falta de comunicação entre mais de um elevador.

Ainda nesse projeto foi muito focado na parte de melhoria e maximização do tempo de utilização do elevador que acaba por não demonstrar poucas praticas de seguranças, como a falta de sensor de porta, utilização de tempo para abertura e fechamento de porta. Dentre outros.

4.3 Sugestões para trabalhos futuros.

Criação de um sistema de comunicação e/ou instalação de dois elevadores que com a melhoria de software trabalhariam em conjunto, como a existência de dois elevadores acaba por ocasionar mais de uma possibilidade de sentido no momento da requisição este novo software deve ficar responsável por fazer o elevador mais próximo e ainda que esteja em mesmo sentido atender a requisição.

Sistema de segurança, este fazendo o controle de velocidade do elevador, acionamento de freios de emergência e verificar se existe algum obstáculo a porta antes de acionar o fechamento das mesmas.

Sistema de verificação de passageiro, caso seja feito uma solicitação de em um botão interno o mesmo deve verificar antes de fechar a porta à existência de um passageiro dentro do elevador, caso se mostre negativo não completar a requisição. E caso a requisição seja de um botão externo o mesmo verifica constantemente a presença do passageiro aguardando o elevador, caso não detecte a presença do usuário o mesmo para a requisição.

5 REFERÊNCIAS BIBLIOGRÁFICAS

Almeida, J. E. – Motores Elétricos: Manutenção e Testes. 3ª Edição.

Boylestad, R. L. & Nashelsky L. – Dispositivos Eletrônicos e Teoria dos Circuitos. 6ª Edição.

Manual Técnico, Display WC1602A-1.

Manual Técnico, PIC16F1314 Data Sheet

Manual Técnico, PIC18FXX2 Data Sheet.

Material do Curso Pós Técnico em Automação Industrial – Motores – CEFET/SC

MIZRAHI, V. V. – **Treinamento em Linguagem C.** Módulo 1. São Paulo: Pearson Makron Books, 1990.

Pereira, F. – Microcontroladores PIC Programação em C. 6ª Edição São Paulo: Érica, 2007.

Saber Eletrônica – Motores elétricos - http://www.sabereletronica.com.br/ .Acessado 02/02/2008 .

Silva, R. A. – Programando Microcontroladores PIC: Linguagem C. Ensino Profissional

Sousa, D. J. & Lavinia N. C. – Conectando o PIC 16F877A Recursos Avançados, 2ª Edição São Paulo: Érica, 2005.

Sousa, D. J. – Desbravando o PIC 6ª Edição, Érica, 2003

Souza, V. A. – Projetando com os Microcontroladorse da Família Pic 18 uma Nova Percepção. Ensino Profissional.

Thyssenkrup Elevadors – Funcionamento e componentes de um elevador. http://www.thyssenkruppelevadores.com.br. Acessado em 15/10/2007.

Wikipédia – Microcontrolador PIC. Disponível em http://pt.wikipedia.org/wiki/PIC. Acessado em 20/09/2007.

APÊNDICE.

Código em C.

```
/*****************************************************************************************************************
Centro Universitário de Brasília – UniCEUB
 FATECS – Faculdade de tecnologia e ciências sociais aplicadas.
Autor: Rafael José Ramos
Projeto: Elevador controlado via microcontrolador.
*****************************************************************************************************************/

/*****************************************************************************************************************
                                                Descrição geral
*****************************************************************************************************************/
// Este software está preparado para demonstrar o controle de um elevador
//              através do microcontrolador PIC 18F452.
/*****************************************************************************************************************
                                                Definição do PIC
*****************************************************************************************************************/

#include <p18F452.h>            // Register definitions


/*****************************************************************************************************************
                                INCLUDES DAS FUNÇÕES DE PERIFÉRICOS DO PIC
*****************************************************************************************************************/
#include <pwm.h>                //PWM library functions
#include <adc.h>                //ADC library functions
#include <timers.h>             //Timer library functions
#include <delays.h>             //Delay library functions
#include <i2c.h>                //I2C library functions
#include <stdlib.h>             //Library functions
```

```
#include <usart.h>                    //USART library functions

/*****************************************************************************************************************
                                        Configurações para gravação
*****************************************************************************************************************/
#pragma config OSC = XT

#pragma config WDT = OFF

#pragma config LVP = OFF


/*****************************************************************************************************************
                                        Prototipagem de função
*****************************************************************************************************************/

void Fecha_Porta ();

void Desce ();

void Sobe ();

void Abre_Porta ();

void n_atual();

void n_destino();


/*****************************************************************************************************************
                                        ENTRADAS
*****************************************************************************************************************/

#define BT PORTBbits.RB0                //PORTA DO BOTÃO

                                                //1 -> PRESSIONADO

                                                //0 -> LIBERADO
```

```
#define BT1 PORTBbits.RB1          //PORTA DO BOTÃO
                                                    //1 -> PRESSIONADO
                                                    //0 -> LIBERADO

#define BT2 PORTBbits.RB2          //PORTA DO BOTÃO
                                                    //1 -> PRESSIONADO
                                                    //0 -> LIBERADO

#define BT3 PORTBbits.RB3          //PORTA DO BOTÃO
                                                    //1 -> PRESSIONADO
                                                    //0 -> LIBERADO

/*****************************************************************************************************************
                                        SAÍDAS
/*****************************************************************************************************************

#define     LD1     PORTAbits.RA0           //LED DE SINALIZAÇÃO
                                                        //1 -> LED ACESO
                                                        //0 -> LED APAGADO

#define     LD2     PORTAbits.RA1           //LED DE SINALIZAÇÃO
                                                        //1 -> LED ACESO
                                                        //0 -> LED APAGADO

#define     LD3     PORTAbits.RA2           //LED DE SINALIZAÇÃO
```

```
                                                        //1 -> LED ACESO
                                                        //0 -> LED APAGADO

#define         LD4     PORTAbits.RA3                   //LED DE SINALIZAÇÃO
                                                        //1 -> LED ACESO
                                                        //0 -> LED APAGADO

#define Porta_Open PORTCbits.RC0    //Abre a porta
#define Porta_Close PORTCbits.RC1    //Fecha a porta
#define Elevador_Up PORTCbits.RC2    //Elevador sobe
#define Elevador_Down PORTCbits.RC3    //Elevador desce
#define lcd PORTD
#define rs PORTCbits.RC4
#define enable PORTCbits.RC5
#define led_open PORTCbits.RC6
#define led_close PORTCbits.RC7

/*****************************************************************************************************************
Definição especial
*****************************************************************************************************************/

#define Padrao Fecha_Porta();if(Posicao > Destino){Motor_Desce =(Posicao - Destino);Desce
();}if(Posicao < Destino){Motor_Sobe = (Destino - Posicao);Sobe();}Abre_Porta ();Posicao = Destino;
#define escreve_comando rs = 0;enable = 1;Delay1KTCYx(1);enable = 0;Delay1KTCYx(1);
#define escreve_dado rs = 1;enable = 1;Delay1KTCYx(1);enable = 0;Delay1KTCYx(1);
```

```
#define lcd_atualiza lcd = 0x01;escreve_comando;lcd = 0x50;escreve_dado;lcd = 0x6F;escreve_dado;lcd = 0x73;escreve_dado;lcd = 0x69;escreve_dado;lcd = 0x63;escreve_dado;lcd = 0x61;escreve_dado;lcd = 0x6F;escreve_dado;lcd = 0x20;escreve_dado;lcd = 0x61;escreve_dado;lcd = 0x74;escreve_dado;lcd = 0x75;escreve_dado;lcd = 0x61;escreve_dado;lcd = 0x6C;escreve_dado;lcd = 0x20;escreve_dado;n_atual();next;

#define next lcd = 0xC3;escreve_comando;lcd = 0x44;escreve_dado;lcd = 0x65;escreve_dado;lcd = 0x73;escreve_dado;lcd = 0x74;escreve_dado;lcd = 0x69;escreve_dado;lcd = 0x6E;escreve_dado;lcd = 0x6F;escreve_dado;lcd = 0x20;escreve_dado;n_destino();


/*****************************************************************************************************************
                                        Definição de variáveis
*****************************************************************************************************************/

unsigned char Posicao = 4;

unsigned char Destino = 0;

unsigned char Motor_Desce = 0;

unsigned char Motor_Sobe = 0;

unsigned int  Time_Porta = 50 ;

unsigned int  Time_Elevador = 100;


/*****************************************************************************************************************
                                        Função Principal
/*****************************************************************************************************************

void main ()

{


        PORTA = 0X00;                   //Clear PORTA
```

```
PORTB = 0X00;                  //Clear PORTB

PORTC = 0X00;                  //Clear PORTC

PORTD = 0X00;                 //Clear PORTD

PORTE = 0x00;                  //Clear PORTE

LATA = 0X00;                   //Clear PORTA

LATB = 0X00;                   //Clear PORTB

LATC = 0X00;                  //Clear PORTC

LATD = 0X00;                  //Clear PORTD

LATE = 0x00;                  //Clear PORTE


TRISA = 0b00000000;            //CONFIG DIREÇÃO DOS PINOS PORTA

TRISB = 0b00001111;            //CONFIG DIREÇÃO DOS PINOS PORTB

TRISC = 0b00000000;            //CONFIG DIREÇÃO DOS PINOS PORTC

TRISD = 0b00000000;            //CONFIG DIREÇÃO DOS PINOS PORTD

TRISE = 0b00000000;            //CONFIG DIREÇÃO DOS PINOS PORTE


ADCON1 = 0b00000111;           //DESLIGA CONVERSORES A/D


/*****************************************************************************************************************
                                        Inicialização do LCD
*****************************************************************************************************************/


lcd = 0x01;

escreve_comando;

lcd = 0x38;

escreve_comando;
```

```
        lcd = 0x0C;
        escreve_comando;
        lcd = 0x06;
        escreve_comando;
        lcd = 0x01;
        escreve_comando;

/*****************************************************************************************************************
                                                Rotina Principa
*****************************************************************************************************************/

        lcd = 0x81;
        escreve_comando;

        lcd = 0x50;                     //P
        escreve_dado;
        lcd = 0x72;                     //r
        escreve_dado;
        lcd = 0x6F;                     //o
        escreve_dado;
        lcd = 0x6A;                     //j
        escreve_dado;
        lcd = 0x65;                     //e
        escreve_dado;
        lcd = 0x74;                     //t
        escreve_dado;
```

```
lcd = 0x6F;                    //o
escreve_dado;
lcd = 0x20;                    //
escreve_dado;
lcd = 0x46;                    //F
escreve_dado;
lcd = 0x69;                    //i
escreve_dado;
lcd = 0x6E;                    //n
escreve_dado;
lcd = 0x61;                    //a
escreve_dado;
lcd = 0x6C;                    //l
escreve_dado;

lcd = 0xC4;
escreve_comando;

lcd = 0x45;                    //E
escreve_dado;
lcd = 0x6C;                    //l
escreve_dado;
lcd = 0x65;                    //e
escreve_dado;
lcd = 0x76;                    //v
```

```
escreve_dado;
lcd = 0x61;                     //a
escreve_dado;
lcd = 0x64;                     //d
escreve_dado;
lcd = 0x6F;                     //o
escreve_dado;
lcd = 0x72;                     //r
escreve_dado;


while(1)
{
        //ClrWdt();

        if (BT)
        {
                Destino = 1;
                lcd_atualiza;
                LD1 = 1;
                Padrao;
                lcd_atualiza;
        }
        else LD1 = 0;

        if (BT1)
```

```
{
        Destino = 2;
        lcd_atualiza;
        LD2 = 1;
        Padrao;
        lcd_atualiza;
}
else LD2 = 0;

if (BT2)
{
        Destino = 3;
        lcd_atualiza;
        LD3 = 1;
        Padrao;
        lcd_atualiza;
}
else LD3 = 0;

if (BT3)
{
        Destino = 4;
        lcd_atualiza;
        LD4 = 1;
        Padrao;
```

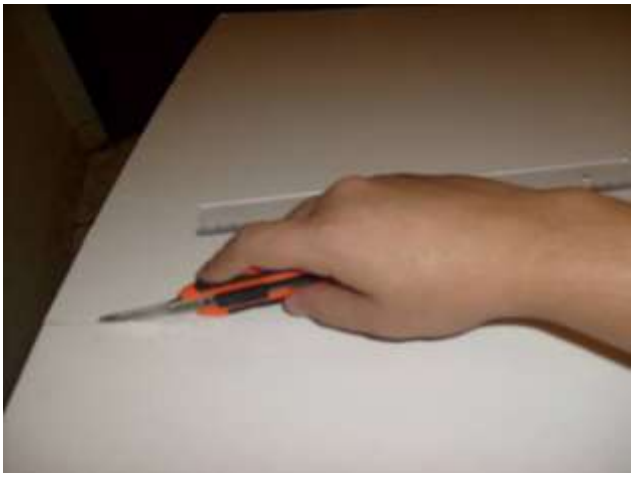
```
                    lcd_atualiza;
                }
                else LD4 = 0;

        }
}       // FIM DO PROGRAMA

/*****************************************************************************************************************
                                        Rotinas Auxiliares
*****************************************************************************************************************/

        void n_atual ()
        {
                switch(Posicao)
                {
                        case 1:
                        lcd = 0x31;
                        break;

                        case 2:
                        lcd = 0x32;
                        break;

                        case 3:
                        lcd = 0x33;
```

```
                break;


                case 4:
                lcd = 0x34;
                break;
        }
        escreve_dado;
}

void n_destino ()
{
        switch(Destino)
        {
                case 1:
                lcd = 0x31;
                break;


                case 2:
                lcd = 0x32;
                break;


                case 3:
                lcd = 0x33;
                break;
```

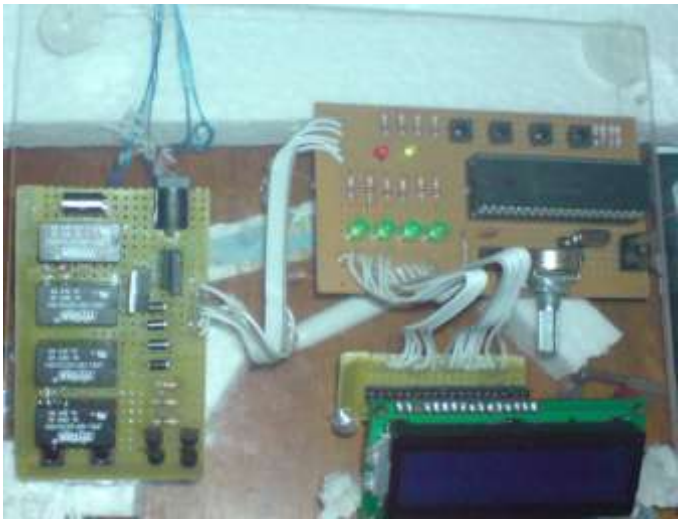
```
                case 4:
                lcd = 0x34;
                break;
        }
        escreve_dado;
}

void Fecha_Porta ()
{
        Porta_Close = 1;
        Delay10KTCYx(Time_Porta);
        Porta_Close = 0;
        led_close = 1;
        led_open = 0;
}

void Abre_Porta ()
{
        Porta_Open = 1;
        Delay10KTCYx(Time_Porta);
        Porta_Open = 0;
        led_open = 1;
        led_close = 0;
}
```

```
void Sobe ()
{
	Elevador_Up = 1;
	do
	{
		Delay10KTCYx(Time_Elevador);
		Motor_Sobe --;

	}while(Motor_Sobe != 0);
	Elevador_Up = 0;
}

void Desce ()
{
	Elevador_Down = 1;
	do
	{
		Delay10KTCYx(Time_Elevador);
		Motor_Desce --;

	}while(Motor_Desce != 0);
	Elevador_Down = 0;
}
```

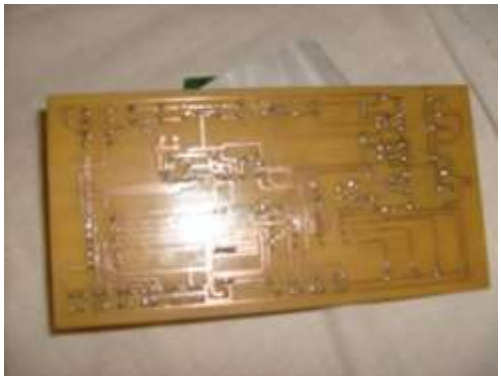
Montagem Maquete 1 Defeito.



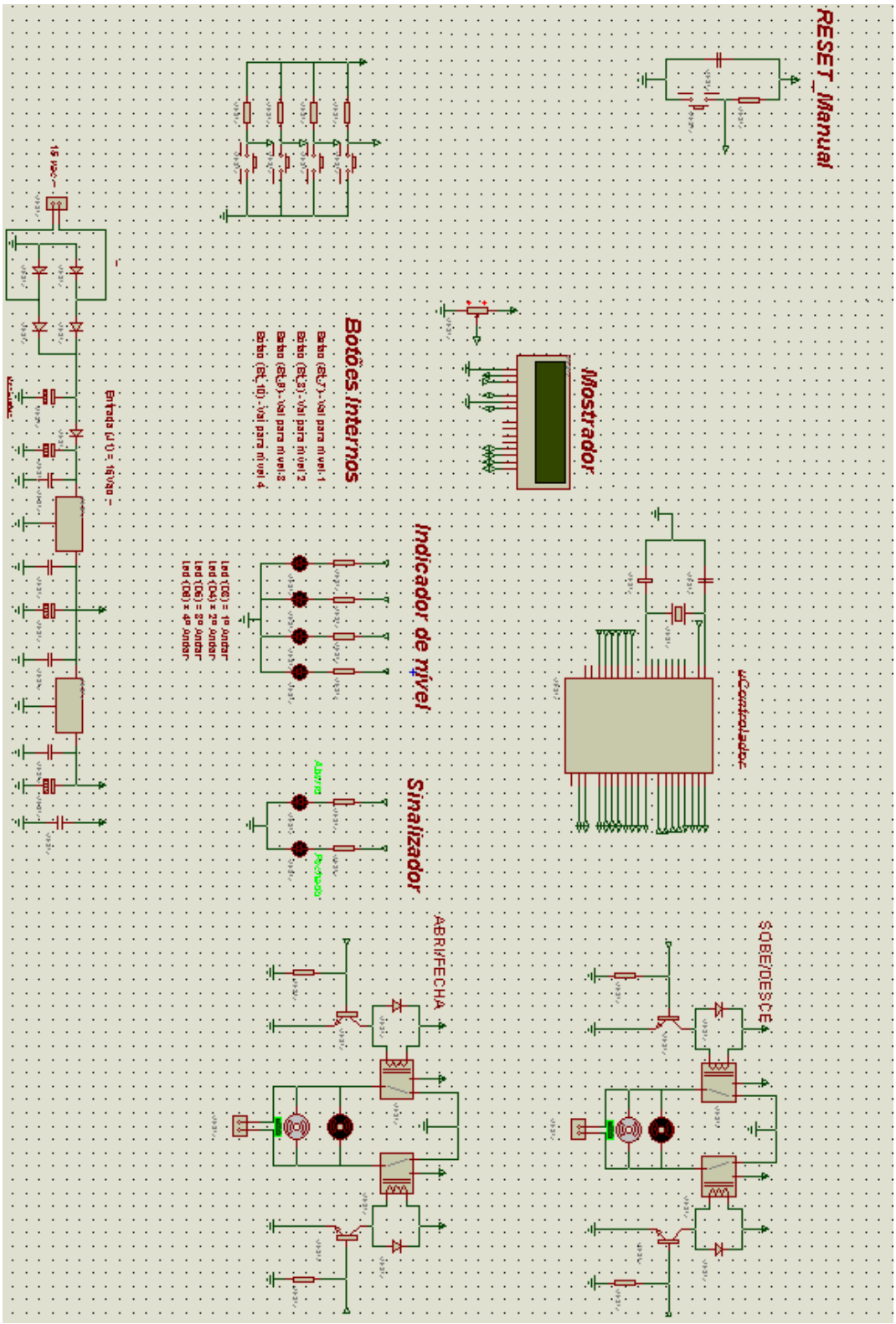
Maquete Final



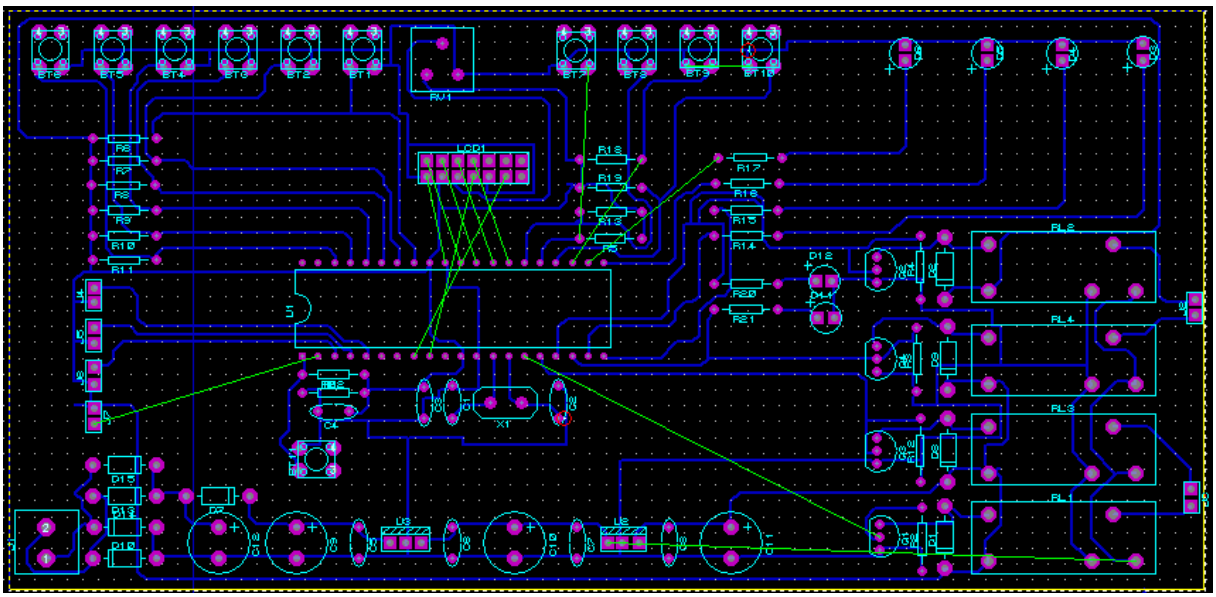
Montagem Circuito



EsquemaCircuito.



Layout PCB.


BT6
BT5
BT4
BT3
BT2
BT1
RV1
BT7
BT8
BT9
BT10
LCD1
R8
R7
R6
R5
R10
R11
R18
R19
R13
R12
R17
R16
R15
R14
R20
R21
D12
D11
RL2
RL4
RL3
RL1
R4
R3
R2
R1
C4
X1
D15
D14
D13
D2
C8
C9
U3
C7
C10
U5
C11