CURSO DE ENGENHARIA ELÉTRICA

Angélico Loreto Teixeira

DESENVOLVIMENTO DE UM SISTEMA MICROCONTROLADO COMO ALTERNATIVA AO MÉTODO ELETROMECÂNICO DE PARTIDA COMPENSADA SEQUENCIAL DE MOTORES ELÉTRICOS

Santa Cruz do Sul
2019

Angélico Loreto Teixeira

DESENVOLVIMENTO DE UM SISTEMA MICROCONTROLADO COMO ALTERNATIVA AO MÉTODO ELETROMECÂNICO DE PARTIDA COMPENSADA SEQUENCIAL DE MOTORES ELÉTRICOS

Trabalho de conclusão apresentado ao Curso de Engenharia Elétrica da Universidade de Santa Cruz do Sul como requisito parcial para obtenção do título de bacharel em Engenharia Elétrica
Orientador: Prof. Me. Adriano José Bombardieri

Santa Cruz do Sul
2019

A Deus pelo dom da vida.
A minha família pelo apoio incondicional.
Aos meus amigos que tornaram essa jornada mais agradável.

AGRADECIMENTOS

Agradeço aos meus pais Maria Conceição Loreto Teixeira e José Cláudio de Vargas Teixeira pelo apoio. Aos professores da Universidade de Santa Cruz do Sul pelos ensinamentos transmitidos. Aos colegas de curso pela amizade e troca de experiências, especialmente ao colega Jessé Carlos dos Santos, pela sugestão do projeto deste trabalho. Aos companheiros de viagem no trajeto Cachoeira do Sul - Santa Cruz do Sul, pela amizade e companheirismo ao longo desses 200 mil quilômetros percorridos. Agradecimento especial ao Prof. Me. Adriano José Bombardieri pelos conhecimentos transmitidos e orientação na realização deste trabalho. Ao governo federal brasileiro por me possibilitar estudar em uma universidade privada através de seus programas de incentivo. Por fim, agradeço aos professores da Universidade Federal de Santa Maria – Campus Cachoeira do Sul, pelo incentivo e parceria de longa data.

Muito obrigado!

"A tarefa não é tanto ver aquilo que ninguém viu, mas pensar o que ninguém ainda pensou sobre aquilo que todo mundo vê."

Arthur Schopenhauer

RESUMO

Este trabalho apresenta o desenvolvimento de um sistema microcontrolado como alternativa ao método eletromecânico de partida compensada sequencial. Assim, constitui-se como uma possível solução aos grandes e complexos circuitos de comando eletromecânicos. Para a elaboração do projeto, foi realizado um levantamento das necessidades. Na sequência, desenvolveu-se um algoritmo capaz de atender as demandas do projeto e esse, foi embarcado em um microcontrolador de uso comercial. Após, confeccionou-se uma placa de circuito impresso capaz de oferecer robustez e confiabilidade. Portanto, ao término do trabalho houve implementação de *firmware* e *hardware* dedicados.

A solução abordada neste trabalho, permite ao usuário definir o número máximo de motores a serem acionados e o tempo de partida entre eles. Por fim, realizou-se um comparativo entre o sistema eletromecânico e o sistema eletrônico afim de verificar as vantagens e desvantagens.

Palavras Chave: Acionamento de Motores Elétricos. Sistemas Embarcados. Partida Compensada Sequencial.

ABSTRACT

This work presents the development of a microcontrolled system as an alternative to the sequential compensated starting electromechanical method. Thus, it is a possible solution to large and complex electromechanical control circuits. To prepare the project, a survey of needs was carried out. In the sequence, an algorithm was developed able to attend the demands of the project and this one, was embarked in a microcontroller of commercial use. After that, a printed circuit board was created capable of offering robustness and reliability. Therefore, at the end of the work there was implementation of dedicated firmware and hardware.

The solution addressed in this work, allows the user to define the maximum number of motors to be triggered and the start time between them. Finally, a comparison was made between the electromechanical system and the electronic system in order to check the advantages and disadvantages.

Keywords: Electric Motors Activation. Embedded Systems. Sequential Compensating Match.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

PCI	Placa de Circuito Impresso
CNI	Confederação Nacional da Indústria
TAP	Derivação de um transformador
CV	Cavalo-vapor
CPU	Unidade de Processamento Central, (do inglês, *Central Processing Unit*)
ROM	Memória somente de Leitura, (do inglês, *Read-Only Memory*)
PROM	Memória Programável somente de Leitura (do inglês, *Programmable Read-Only Memory*)
EPROM	Memória Programável apagável somente de Leitura, (do inglês, *Erasable Programmable Read-Only Memory*)
EEPROM	Memória Programável somente de Leitura apagável eletricamente, (do Inglês, *Electrically-Erasable Programmable Read-Only Memory*)
RAM	Memória de Acesso Aleatório, (do Inglês, *Random-Access Memory*)
SRAM	Memória de Acesso Aleatório Estático, (do inglês, *Static Random-Access Memory*)
DRAM	Memória de Acesso Aleatório Dinâmico, (do inglês, *Dynamic Random-Access Memory*)
RISC	Computador com Conjunto Reduzido de Instruções, (do inglês, *Reduced Instruction Set Computer*)
PCI	Placa de Circuito Impresso
PTH	Componente soldado por inserção, (do inglês, *Pin Through-Hole*)
SMT	Tecnologia de Montagem em Superfície, (do inglês, *Surface-Mount Technology*)
SMD	Componente soldado por inserção, (do inglês, *Surface-Mount Device*)
IMT	Tecnologia de Montagem por Inserção, (do inglês, *Insertion-Mount Technology*)
OLED	Diodo Emissor de Luz Orgânico, (do inglês, *Organic Light-Emitting Diode*)
OPV	Dispositivo fotovoltaico Orgânico, (do inglês, *Organic Photovoltaic*)

OTFT	Transistor de Filme Fino de Carbono, (do inglês, *Organic Thin-Film Transistor*)
EMC	Compatibilidade Eletromagnética, (do inglês, *Electromagnetic Compability*)
EMI	Interferência Eletromagnética, (do inglês, *Electromagnetic Interference*)

SUMÁRIO

1 INTRODUÇÃO

As chaves de partida são muito utilizadas em instalações elétricas industriais. Uma dessas chaves de partida denomina-se compensadora. De acordo com (FRANCHI, 2008, p. 170):

> "[...] a chave de partida compensadora pode ser usada para partir motores sob carga, os quais podem ser de tensão única e possuírem apenas três cabos."

De modo geral, pode-se dizer que uma chave de partida é constituída de um circuito de comando e circuito de força. Conforme Franchi (2008), uma chave de partida é constituída basicamente por três tipos de circuitos: principal (ou força), de comando e auxiliar. Esse trabalho tem como enfoque a substituição do circuito de comando eletromecânico por um microcontrolado.

Existem várias definições de microcontroladores. Contudo, como lembra (SOUZA, 2005), um microcontrolador pode ser definido como um pequeno dispositivo eletrônico com uma certa inteligência programável. Além disso, com o advento da Quarta Revolução Industrial (Indústria 4.0) microcontroladores serão essenciais para a implementação de sistemas inteligentes e de baixo custo.

Essa obra consiste na análise da partida compensada sequencial e substituição da lógica de comando eletromecânico por um sistema microcontrolado utilizando um microcontrolador de uso comercial. Para implementação do sistema embarcado, será desenvolvido um algoritmo capaz de atender aos requisitos de projeto da aplicação. Por fim, foi realizado a comparação do sistema eletromecânico com o sistema microcontrolado.

1.1 Área e limitação do tema

Este trabalho está alicerçado nas áreas de eletrônica, acionamento de motores elétricos e computação. O tema discorrido envolve análise da partida compensada sequencial de motores elétricos. Após, será realizado o levantamento de necessidades para implementação de um sistema embarcado capaz de substituir os convencionais sistemas de partida eletromecânicos. Para realização deste trabalho, será necessário o uso de vários conhecimentos obtidos ao longo do curso de Engenharia Elétrica, principalmente disciplinas como: máquinas elétricas, sistemas de acionamentos, circuitos eletrônicos, instrumentação e controle, entre outras. Para

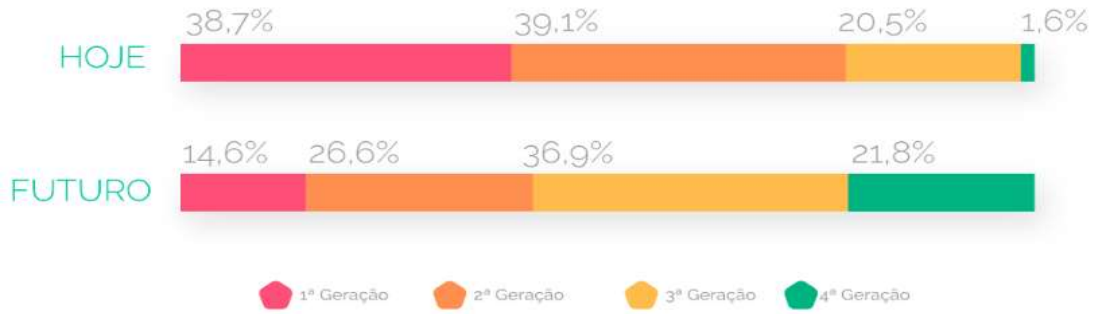
validação dos resultados, serão realizados testes em ambientes que retratem o mais próximo possível as situações reais de aplicação.

1.2 Contextualização

A utilização de chaves de partida em sistemas de acionamento de motores elétricos é extremamente utilizada. As aplicações abrangem os mais variados segmentos da economia: agricultura, indústria, saneamento básico, entre outros.

Segundo uma pesquisa realizada pela Confederação Nacional da Indústria (CNI, 2017), o número de indústrias brasileiras que esperam digitalizar seus processos produtivos mudará de 1,6% atualmente para 21,8% em dez anos. Portanto, existe um grande mercado para soluções inteligentes capazes de atender aos desafios da Indústria 4.0. A Figura 1 a seguir, apresenta o estágio de tecnologia da indústria brasileira no ano de 2017 e sua perspectiva para o ano de 2027.

Figura 1 - Estágio de Tecnologia da Indústria Brasileira



Fonte: Adaptado de PORTAL DA INDÚSTRIA (2017).

Assim, demonstra-se que a utilização de chaves de partida combinadas com sistemas eletrônicos e sistemas informáticos são uma tendência no país. Logicamente que para o desenvolvimento de produtos industriais sofisticados com elevado nível de integração, seria necessário a utilização de outros recursos tecnológicos que não estão contemplados nesse trabalho.

1.3 Justificativa

Grande parte dos sistemas industriais que utilizam a partida compensada sequencial fazem uso de lógica de comando eletromecânico. Todavia, sistemas eletromecânicos possuem desvantagens como: custo elevado dos componentes, complexibilidade na montagem de sistemas maiores e perda de tempo na montagem dos quadros de comando.

Contudo, o surgimento de dispositivos eletrônicos cada vez mais baratos, precisos e robustos, possibilita a realização de uma mesma tarefa aplicando outra tecnologia. Existem vantagens ao se aplicar circuitos eletrônicos, por exemplo: velocidade na montagem do quadro de comando pelo eletricista, possibilidade de configuração do número de motores, tempo de partida e integração com sistemas informatizados. Portanto, a solução proposta necessita a aplicação de conhecimentos na área de engenharia elétrica justificando o desenvolvimento deste trabalho.

1.4 Objetivos

Nesta seção serão apresentados primeiramente o objetivo geral e, posteriormente, os objetivos específicos.

1.4.1 Objetivo geral

O principal objetivo desse trabalho consiste em desenvolver um sistema eletrônico embarcado capaz de substituir a tradicional lógica de comando eletromecânica presente na partida compensada sequencial de motores elétricos.

1.4.2 Objetivos específicos

- Analisar a partida compensada sequencial;
- Levantar os requisitos para implementação do sistema eletrônico;
- Implementar algoritmo capaz de atender as necessidades do projeto;

- Desenvolver placa de circuito impresso para embarcar o sistema de comando;
- Comparar o desempenho do sistema embarcado com o sistema convencional;

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo será apresentado uma revisão bibliográfica utilizando livros, monografias, dissertações, artigos científicos entre outras fontes de pesquisa. Serão apresentados conceitos sobre microcontroladores, sistemas embarcados, tipos de partidas de motores elétricos, entre outros.

2.1 Tipos de chaves de partida

De acordo com FRANCHI (2008), a partida de motores elétricos é um dos momentos mais críticos pois ocorre a mudança do momento de inércia. Em outras palavras, o motor sairá de uma condição de repouso e passará para uma condição de movimento. Contudo, para que ocorra essa mudança no momento de inércia o motor necessita de um valor de corrente maior do que quando em regime permanente. Essa corrente extra denomina-se corrente de pico e geralmente varia de 6 a 8 vezes maior do que o valor da corrente nominal do motor. Esses valores maiores de corrente de partida acabam resultando em grandes problemas para concessionárias de energia ou mesmo em sistemas alternativos como geração distribuída ou geradores a diesel.

Além do mais, esses valores excessivos de corrente podem fazer com que os dispositivos de proteção ao longo do sistema elétrico venham a atuar e por consequência interromper o fornecimento de energia. Por isso, a redução da corrente de pico é algo imprescindível para o bom funcionamento do sistema elétrico como um todo. Outro fato importante a destacar é que a corrente de partida do motor é diretamente proporcional a tensão aplicada sobre o motor. Por isso, as chaves de partida que serão apresentadas a seguir se trata de dispositivos eletromecânicos que tem como foco reduzir a tensão aplicada ao motor e por consequência, ocorrerá a diminuição da corrente de pico durante a partida. Esse evento de submissão do motor a um valor de tensão menor ocorre por um período suficiente para que o motor atinja um valor de rotação próximo ao valor nominal e por fim, o motor submetido ao valor da tensão nominal da rede de alimentação. Apesar da partida de motores ser um evento relativamente rápido acaba gerando grandes problemas ao sistema elétrico como um todo, desde queda no valor de tensão, redução da qualidade da energia elétrica, entre outros.

2.1.1 Partida Direta

Segundo FRANCHI (2008), a partida direta de motores elétricos trifásicos é a mais utilizada para motores com potência até 5CV. Para potências maiores devem ser utilizadas outras chaves de partida. A partida direta consiste na aplicação das três fases, porém deve-se tomar cuidado com os níveis de tensão de linha para não causar danos irreversíveis ao motor elétrico. A partida direta deve ser utilizada para motores de pouca potência e sem necessidade de aceleração progressiva. Esse tipo de partida apresenta vantagens como conjugado de partida elevado, baixo custo e partida rápida. Entre as desvantagens estão queda de tensão no sistema de alimentação, os componentes elétricos devem ser superdimensionados, além das correntes de pico elevadas. Além disso, na partida direta a corrente de partida é diretamente proporcional à tensão de alimentação e diminui à medida que a velocidade aumenta. Nesse tipo de partida o conjugado varia proporcionalmente ao quadrado da tensão de alimentação.

2.1.2 Partida Estrela-Triângulo

Segundo FRANCHI (2008), esse tipo de partida é realizado utilizando um valor menor de tensão nas bobinas do motor durante a partida. A sequência de fechamento das bobinas consiste na seguinte maneira: o motor parte em estrela, com uma tensão de 58% do valor nominal, após transcorrer um determinado período é desfeito o fechamento em estrela e realizado o fechamento em triângulo. Esse tipo de chave de partida, estrela-triângulo, reduz em cerca de 33% o valor da corrente de partida. Deve ser aplicada em situações com baixo conjugado de partida. Geralmente utiliza-se essa partida para acionar máquinas que partem a vazio. Essa característica do conjugado deve-se ao fato de que o conjugado de partida é proporcional a quarado da tensão de alimentação. A estabilização da velocidade do motor ocorre quando o conjugado do motor e resistente se equilibram normalmente entre 75% e 80% da velocidade nominal. Para trocar o fechamento do motor de estrela para triângulo utiliza-se um temporizador. Um fato muito importante a ser destacado é que para esse tipo de partida o motor deve ser capaz de suportar dois níveis diferentes de tensão. No Brasil geralmente utiliza-se motores de 220/380V e 380/660V. Por isso, esse tipo de motor possui no mínimo seis bornes de ligação. Uma informação interessante é que para a

mudança do fechamento estrela para o fechamento triângulo é necessário esperar um período que varia de 30 a 100 ms, dessa forma evita-se o fechamento simultâneo de contatores que poderia levar a um curto-circuito entre as fases. Existe um temporizador especialmente projetado para utilização na partida estrela-triângulo.

Ainda segundo FRANCHI (2008), entre as vantagens desse tipo de partida pode-se destacar baixo custo em relação a outros tipos de chaves de partida para motores de grande potência, um espaço menor dentro do quadro de comando, sem limite máximo de manobras por intervalo de tempo. Entre as desvantagens pode-se destacar a necessidade de um motor de seis terminais para a ligação, valor da tensão de linha deve coincidir com o valor de tensão da ligação triângulo do motor. Além disso, caso o motor não atinja valores próximos a 90% da rotação nominal ao comutar para o fechamento triângulo o valor da corrente de pico será o mesmo de uma partida direta.

2.1.3 Partida Compensadora

Segundo FRANCHI (2008), essa chave de partida alimenta as bobinas do motor com uma tensão menor que a tensão de linha, devido a utilização de um autotransformador em série com as bobinas. Após a partida, as bobinas do motor estarão submetidas a tensão nominal. Geralmente, para aplicação da partida compensada deve-se utilizar vários componentes elétricos, tais como: autotransformador com fechamento em estrela, três contatores, relé de sobrecarga, fusíveis de ação retardada, entre outros componentes. Devido a utilização de grandes quantidades de componentes a partida compensada possui um custo mais elevado. Os autotransformadores, são formados por três bobinas e ao longo de cada bobina, existem derivações chamadas de TAPS. Geralmente, são empregados TAPS de 50, 65 e 80% da tensão aplicada na fase. Os autotransformadores também possuem sondas térmicas no seu bobinado para monitorar o crescimento da temperatura. Caso a temperatura aumente consideravelmente, o autotransformador deixará de funcionar até que sejam reestabelecidos valores aceitáveis de temperatura. Evitando assim, a sua destruição.

Ainda segundo FRANCHI (2008), o autotransformador realiza o rebaixamento da tensão de alimentação das bobinas do motor e por conseguinte a redução do conjugado de partida. A escolha do TAP tem relação direta ao conjugado de partida

do motor. Portanto, para maiores conjugados de partida será necessário o uso de um TAP mais elevado resultando em correntes de partida maiores. Geralmente, utiliza-se a partida compensada em motores com potência superior a 15CV. Basicamente pode-se dividir a partida compensada em três estágios:

Estágio 1: Ocorre o fechamento do autotransformador em estrela e após o motor é energizado. Como o motor está em série com o autotransformador, será aplicada uma tensão reduzida sobre o primeiro. Esse valor, deve ser determinado pelo projetista e irá depender onde o projeto será aplicado.

Estágio 2: Ocorre a abertura da ligação estrela. Convém ressaltar que isso deve ocorrer quando o motor já atingiu cerca de 90% da velocidade nominal de rotação.

Estágio 3: Ocorre a aplicação do valor da tensão de linha sobre o motor. Dessa forma, o motor poderá atingir seus valores nominais de rotação.

Entre as vantagens da chave de partida compensada pode-se citar: ao retirar o motor da série com o autotransformador e colocá-lo no valor da tensão nominal da rede não ocorrerá o desligamento do motor e, portanto, não haverá correntes de pico elevadas. Possibilidade de variar entre diferentes níveis de tensão pré-estabelecidos agregando maior flexibilidade ao sistema. O valor da tensão da rede pode ser igual ao valor da tensão de ligação estrela ou triângulo do motor. O motor necessita apenas de três bornes de ligação.

Entre as desvantagens da chave de partida compensada pode-se citar: maior custo devido a necessidade de compra de um autotransformador, número limitado de manobras por intervalo de tempo visto que o autotransformador se aquece a cada partida, necessidade de um espaço maior para alocar o autotransformador dentro do quadro de comando. A Figura 2 abaixo, apresenta um autotransformador sendo este, um dispositivo essencial para realização da partida compensada.

Figura 2 - Autotransformador utilizado na partida compensadora

Fonte: Website Etna Transformadores.

2.2 Sistemas embarcados

Como lembram ALMEIDA, MORAES e SERAPHIM (2016), sistemas embarcados consistem em sistemas eletrônicos microprocessados. Entre suas principais características estão a possibilidade de poderem ser programados, conferindo assim flexibilidade durante o desenvolvimento do produto. Porém, os sistemas embarcados diferem dos computadores convencionais, pois os primeiros ao serem programados executaram sempre as mesmas tarefas. Contudo, os computadores podem realizar tarefas de entretenimento e após, se tornarem estações de trabalho bem mais flexíveis durante a utilização. Os sistemas embarcados estão presentes em vários segmentos: agricultura, telecomunicações, automação industrial e residencial, entre outros segmentos. Além disso, sistemas embarcados possuem maiores limitações quando comparados a computadores convencionais. Além disso, para serem programados necessitam de um conhecimento tanto de hardware como de software e linguagens específicas de programação.

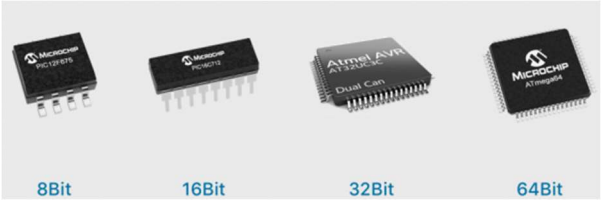
2.2.1 Microcontroladores

Segundo GIMENEZ (2019), microcontroladores são dispositivos eletrônicos fabricados com material semicondutor e alocados no invólucro de um circuito integrado (CI). Esses dispositivos possuem todas as partes essenciais presentes em um microcomputador. São elas: microprocessador (CPU), memórias não-voláteis (ROM, PROM, EPROM, EEPROM), memórias voláteis (RAM, SRAM, DRAM, *flash* RAM), portas de entrada e saída (portas de comunicação paralela ou serial, conversores analógicos/digitais, conversores digitais / analógicos, etc). O conceito principal do microcontrolador é possuir todos os recursos necessários que um microcomputador possui, porém, com suas respectivas limitações. Geralmente são utilizados em aplicações onde não se requer grande quantidade de armazenamento de dados, como sistemas de automação residencial, eletrodomésticos, automação industrial, entre outros.

Além disso, microcontroladores possuem um custo relativamente baixo e trabalham com arquitetura de processador *RISC (Reduced Instruction Set Computer),* que traduzido ao português seria “Computador com número reduzido de instruções”. Uma das principais características dos microcontroladores que os diferenciam dos

tradicionais circuitos integrados é a possibilidade de programá-los conforme a necessidade da aplicação. Dessa forma, são considerados componentes eletrônicos extremamente flexíveis justificando sua imensa utilização. A Figura 3 abaixo, apresenta o aspecto físico de um microcontrolador em diferentes tipos de encapsulamentos.

Figura 3 - Microcontroladores em diferentes encapsulamentos



Fonte: Website IIOT World.

2.2.2 Timer

Segundo HUANG (2005), em sistemas digitais o tempo é representado por um contador de tempo. Existem muitas aplicações que é muito difícil ou quase impossível a implementação sem o uso de uma função *timer* (ou temporizador). Exemplos onde o uso do temporizador é essencial: gravação e comparação do tempo de início de um evento, geração de interrupção periódica, medição do período e largura de pulso, medição do ciclo de trabalho e frequência de sinais periódicos, contagem de eventos, referências de tempo, geração de formas de onda que estão baseadas no ciclo de trabalho, entre outras aplicações. Geralmente os microcontroladores atuais possuem mais de um timer sendo que esses podem variar em quantidade de bits dentro de um mesmo microcontrolador.

2.2.3 Interrupção

Conforme lembram PEREZ e ARENY (2009), uma requisição de interrupção ou somente interrupção, consiste em um evento interno ou externo que quando ocorre faz o microcontrolador interromper a execução do corrente programa e passa a executar um outro programa. Após a realização da interrupção solicitada o microcontrolador retorna a executar a rotina que estava acontecendo antes do disparo da interrupção. Além disso, uma interrupção poderá ser executada a qualquer

momento a partir da ocorrência de determinadas condições. Portanto, não é possível prever quando ocorrerá uma interrupção. Uma interrupção pode ocorrer devido a solicitação de um evento interno ou externo. A interrupção interna pode ser gerada pelo módulo de entrada ou saída de um microcontrolador, pela memória ou pela CPU. Interrupções externas são originadas por periféricos e chegam ao microcontrolador através de um ou mais pinos. Além disso, os microcontroladores possuem recursos para receber e processar essas requisições de interrupção.

2.2.4 Algoritmo

Segundo CORMEN (2014), o conceito de algoritmo consiste em um conjunto de instruções ou etapas que sejam capazes de possibilitar a execução de uma determinada tarefa. Esse conceito de algoritmo sempre esteve presente no cotidiano das pessoas, pois etapas para realizar uma determinada tarefa são coisas pertinentes ao dia a dia. Contudo, quando se refere a algoritmos computacionais uma das características que o distingue é a precisão e por consequência rejeição de resultados não esperados. Além disso, um fato interessante é que para um mesmo problema podem existir um número muito grande de possíveis soluções. Por isso, um algoritmo deve ser capaz de resolver problemas em um período aceitável.

2.2.5 Firmware

Segundo GIMENEZ (2019), o *firmware* de um microcontrolador consiste no programa *(software)* que é armazenado em uma memória não-volátil (ROM, PROM, EPROM, EEPROM) de um equipamento. O objetivo ao desenvolver o firmware pelo projetista consiste em definir as condições de operação do microcontrolador conforme os requisitos de projeto para determinada aplicação.

Além disso, como lembra SUN et al. (2015), *firmware* geralmente é considerado parte do hardware, ao invés de ser considerado parte do software, porque ele reside dentro do hardware. Geralmente em um dispositivo de memória Flash ou ROM. Contudo, quando ele está sendo escrito em uma linguagem de programação durante seu desenvolvimento, as ferramentas e metodologias que são utilizadas apontam

claramente que se trata de um software, mesmo que muitas vezes esteja intrinsecamente ligado ao hardware (parte física).

2.2.6 Hardware

Segundo GIMENEZ (2019), o *hardware* consiste no conjunto de componentes eletrônicos utilizados, placa de circuito impresso (PCI), cabos, conectores, sensores, atuadores. Portanto, o hardware é toda a parte física do equipamento. O hardware pode ou não possuir flexibilidade. Quanto maior a flexibilidade, velocidade de processamento, memória, entre outras especificações aumentam seu custo. Para atender aos requisitos de projeto, muitas vezes é necessário desenvolver um firmware que estará embarcado no hardware e será responsável por atender as necessidades da aplicação. A Figura 4 a seguir, apresenta a imagem de uma parte do hardware de um computador.

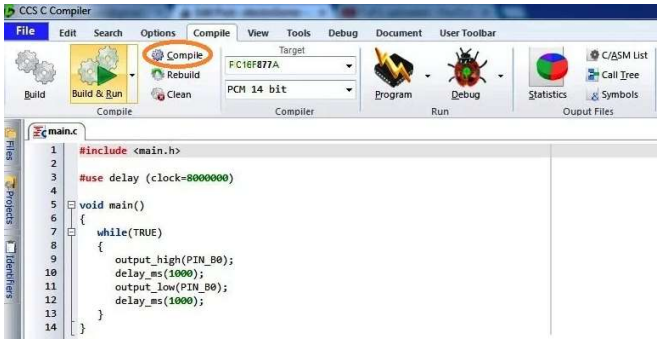
Figura 4 - Parte do hardware de um computador

Fonte: Website SG Computers.

2.2.7 Compilador

Segundo AHO, SETHI e ULLMAN (1990), os compiladores podem ser entendidos como o tradutor do código-fonte que foi escrito em uma determinada linguagem de programação e necessita ser traduzido para a linguagem do computador. Somente assim, o computador será capaz de executar o programa. Além disso, um dos objetivos ao se utilizar um compilador é pela sua capacidade de relatar erros no programa-fonte durante o processo de tradução. A Figura 5 a seguir, apresenta a interface de um compilador para microcontroladores.

Figura 5 - Interface de um compilador para microcontroladores

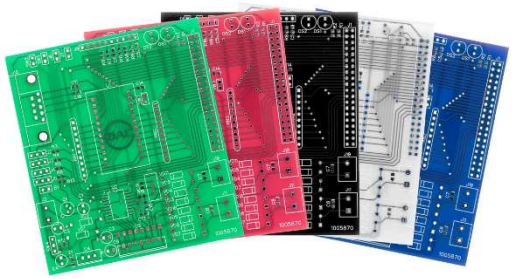


Fonte: Website Electrosome.

2.3 Placa de Circuito Impresso (PCI)

Segundo WAZLAWICK (2016), a invenção da placa de circuito impresso ocorreu em 1936. O invento é creditado ao engenheiro austríaco Paul Eisler. O conceito básico dessa invenção, consiste na impressão de contatos elétricos (material condutor) em uma placa rígida (material isolante) para ligação de componentes eletrônicos. Antes dessa invenção, os componentes eletrônicos eram soldados diretamente entre si. Contudo, esse método era completamente manual, tinha grande possibilidade de apresentar erros e não possibilitava nenhuma forma de automatização. No começo esse invento não teve grande aceitação pelo mercado uma vez que a mão de obra era extremamente barata e abundante. Contudo, durante a Segunda Guerra Mundial esse cenário social mudou completamente e o invento de Eisler passou a ser estratégico. A Figura 6 a seguir, apresenta placas de circuito impresso fabricadas em diferentes cores.

Figura 6 - Placa de circuito impresso em diferentes cores



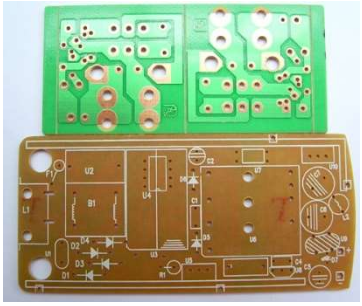
Fonte: Website Electronics-lab.

2.3.1 Classificação das Placas de Circuito Impresso (PCI)

2.3.1.1 Face única

As placas de face única também são referenciadas pelo seu termo em inglês *Single Layer.* Conforme abordado por SCHROEDER (1998), placas de circuito impresso de face única ainda são amplamente devido ao seu baixo-custo. Sendo empregadas principalmente em produtos de alto volume de produção. Já que uma única camada está disponível para a realização de roteamento, as trilhas de sinal não podem passar por cima de outra a menos que se utilize *jumpers*, pequenos pedaços de condutor utilizados para desviar trilhas em uma placa de circuito impresso. Enquanto o uso de múltiplos jumpers não é incomum, os custos associados crescem rapidamente até o ponto que uma placa de dupla face ou múltiplas faces torna-se mais vantajoso tanto no ponto de vista econômico quanto em tempo de montagem. Placas de face única geralmente são fabricadas com o uso de processos e equipamentos menos sofisticados que outros tipos de placas. Entre as desvantagens das placas de face única pode-se citar a limitação quanto ao seu uso em ambientes sujeitos a vibrações e impactos. Isso deve-se ao fato de que os componentes são apoiados por um fino anel de cobre preenchido com estanho em sua superfície. Sob estresse repetitivo, o adesivo vinculado entre o cobre e a lâmina da placa (substrato) pode se separar. Assim, a ilha pode se desprender da trilha a qual estava associada. A Figura 7 a seguir, apresenta uma placa de circuito impresso de face única.

Figura 7 - Placa de circuito impresso de face única



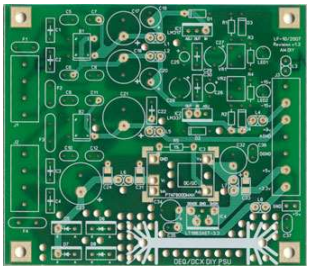
Fonte: Website JVD PCB.

2.3.1.2 Face Dupla

As placas de face dupla também são referenciadas pelo seu termo em inglês *Double Layer.* Conforme abordado por SCHROEDER (1998), placas de circuito

impresso de face dupla podem ser encontradas nos mais variados segmentos como automobilísticos, eletrodomésticos, equipamentos militares, industrial, entre outros. Placas de face dupla permitem que para um mesmo componente tipo PTH exista uma ilha em cima e outra embaixo, conferindo, portanto, maior resistência mecânica contra impacto e vibrações. Uma densidade maior de componentes pode ser alocada em uma placa de face dupla em comparação a uma de face única. Quando existe a possibilidade de roteamento em face dupla e utilização dos furos de via, consiste em um furo que atravessa o substrato e conecta duas trilhas que estão em lados diferentes, assim os jumpers podem ser eliminados possibilitando um aumento de produtividade para as indústrias de manufatura eletrônica. A Figura 8 a seguir, uma placa de circuito impresso de face dupla.

Figura 8 - Placa de circuito impresso em face dupla



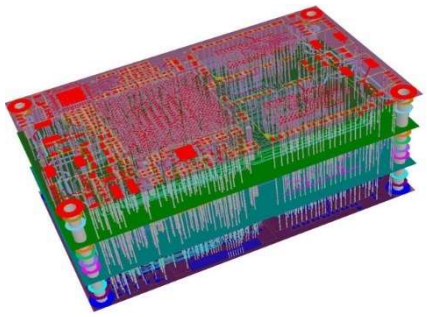
Fonte: Website AWDIY.

2.3.1.3 Faces Múltiplas

As placas de faces múltiplas (ou multicamadas) também são referenciadas pelo seu termo em inglês *Multi-layer.* Conforme abordado por KHANDPUR (2005), placas de circuito impresso com multicamadas possuem mais do que duas camadas condutoras (camadas de cobre) sobrepostas uma sobre a outra. As camadas de cobre são unidas por uma camada de resina. Placas multicamadas representam o tipo mais complexo de placas de circuito impresso. Possuem um custo relativamente alto, devido a complexibilidade durante o processo fabril, baixo volume de produção e dificuldade de retrabalho. A necessidade do uso de placas de multicamadas se justifica devido ao aumento da densidade de componentes e circuitos integrados. Aumentando assim, a concentração do número de linhas de interconexão. Entre as desvantagens do uso deste tipo de placa estão problemas como ruído, capacitância parasita, cross-talk, etc.

Ainda segundo KHANDPUR (2005), placas de circuito impresso multicamadas são fabricadas através do empilhamento de dois ou mais circuitos uma sobre a outra e deve estabelecer uma configuração confiável de interconexões pré-determinadas entre si. O processo de fabricação de placas multicamadas é diferente do processo convencional. Além disso, as placas multicamadas são aplicadas para fabricação de computadores, equipamentos militares e em projetos os quais peso e volume são considerações primordiais. Também, são muito utilizadas em aplicações de circuitos que trabalham com altas velocidades porque quando se tem à disposição mais do que duas camadas são possíveis criar grandes planos de terra e de alimentação evitando problemas de cross-talk, interferências eletromagnéticas, entre outros. A Figura 9 a seguir, apresenta a imagem de uma placa de face múltipla.

Figura 9 - Projeto de placa de circuito impresso de múltiplas faces



Fonte: Website Well PCB.

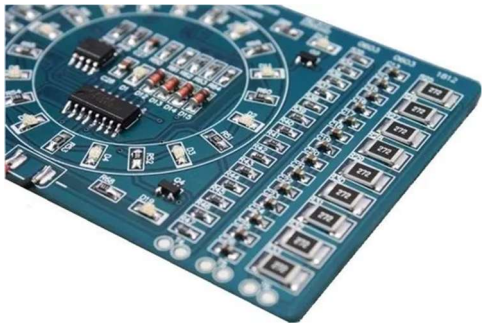
2.4 Classificação de Placas quanto ao Processo de Montagem

2.4.1.1 SMT

De acordo com TRAISTER (1990), a sigla em inglês *SMT (Surface Mount Technology)* que traduzida ao português significa Tecnologia de Montagem em Superfície engloba a possibilidade de processos de montagem de placas eletrônicas totalmente automatizado. Contudo em indústrias de menor porte ainda é comum ver montagem de componentes SMD de forma manual. O conceito básico do SMT é a alocação de componentes sobre a superfície de um substrato. Além disso, essa tecnologia de montagem de componentes eletrônicos possibilita a redução do tamanho de placas eletrônicas em comparação aos componentes convencionais. Assim, vem possibilitando o desenvolvimento equipamentos menores e portáteis.

Ainda de acordo com TRAISTER (1990), a sigla em inglês *SMD (Surface Mounted Devices)* que traduzida ao português significa Dispositivos Montados em Superfície refere-se aos componentes eletrônicos. Frequentemente existe uma certa confusão em relação aos dois termos. Portanto, SMT refere-se aos procedimentos e técnicas utilizadas na montagem dos componentes sobre superfície, enquanto o termo SMD, refere-se aos componentes eletrônicos em si e que estão englobados dentro do termo SMT. A Figura 10 a seguir, apresenta uma placa de circuito impresso montada com componentes SMD.

Figura 10 - Montagem utilizando componentes SMD

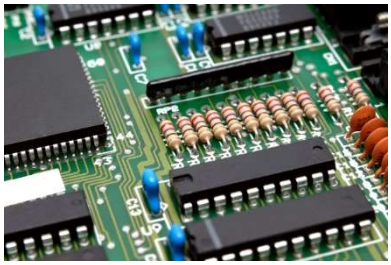


Fonte: Website Ventro.

2.4.1.2 IMT

Segundo VISWANADHAM e SINGH (1998), a sigla em inglês *IMT (Insertion-Mount Technology)* que traduzida ao português significa Tecnologia de Montagem por Inserção engloba as técnicas utilizadas para a fabricação de placas de circuito impresso com componentes eletrônicos inseridos por inserção. Além disso, a sigla em inglês *PTH (Pin Through-Hole)* que traduzida literalmente ao português, significa Pino-no-Buraco refere-se aos componentes eletrônicos que são empregados através da tecnologia de montagem IMT. Cabe ressaltar, os componentes tipo PTH são frequentemente chamados pela literatura técnica de componentes eletrônicos convencionais pois, foram os primeiros a serem inventados e estão a mais tempo no mercado. A Figura 11 a seguir, apresenta uma placa de circuito impresso montada com componentes PTH.

Figura 11 - Montagem utilizando componentes PTH



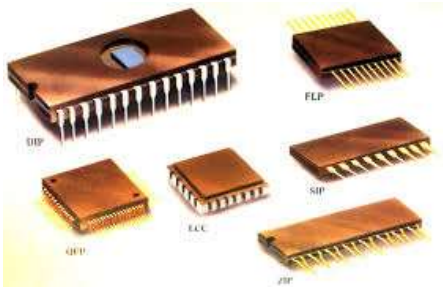
Fonte: Website Wikisis.

2.4.2 Encapsulamentos

Segundo HESTER e HARRISON (2009), circuitos eletrônicos modernos utilizam componentes e interconexões muito pequenas e finas. Isso faz com que sejam susceptíveis a riscos durante a manipulação e exposição a condições típicas de operação. Por isso, é necessário prover proteção a esses componentes tão delicados. Assim, para alcançar esse nível de proteção são utilizados uma ampla faixa de materiais para encapsulamento os quais podem conferir proteção por muitos anos. Basicamente o termo encapsulamento refere-se a carcaça do componente eletrônico. Além disso, componentes diferentes podem ser alocados em um mesmo tipo de encapsulamento ou invólucro. Logo, sempre é necessário observar a numeração do componente eletrônico e após procurar pela folha de dados do componente *(Datasheet)* nos motores de busca da internet possibilitando conhecer em detalhes as características de um determinado componente.

Conforme exposto por CUI (2016), para dispositivos eletrônicos inorgânicos, sem a presença de carbono, tais como chips de circuito integrado é preciso protegê-los através de um encapsulamento ou invólucro contra danos físicos ou químicos. Para dispositivos eletrônicos orgânicos, tais como diodos emissores de luz (OLED), dispositivos fotovoltaicos orgânicos (OPV) e transistores de filme fino de carbono (OTFT), o encapsulamento é utilizado não somente para prevenir contra danos externos mas também prolongar o tempo de vida desses dispositivos orgânicos, uma vez que muitos desses materiais possuem tendência ao decaimento e perda das suas propriedades elétricas em contato com minúsculos traços de água e oxigênio. Conforme exposto acima, o invólucro dos componentes eletrônicos é essencial para o seu correto funcionamento. A Figura 12 a seguir, apresenta os diversos tipos de encapsulamentos utilizados em componentes eletrônicos.

Figura 12 - Diferentes tipos de encapsulamentos



Fonte: Website Chipsetc.

2.5 Compatibilidade Eletromagnética (EMC)

A sigla em inglês *EMC (Electromagnetic Compability)* traduzida ao português significa Compatibilidade Eletromagnética. Conforme MONTROSE (1999), no mercado internacional de produtos eletrônicos existe a necessidade de atendimento a regulações e requisitos conforme solicitação de agências governamentais, organizações privadas de padronização ou conselhos voluntários. Existem normas que devem ser atendidas para venda de produtos tanto nacionais ou importados na América do Norte, União Europeia e entre outros países ao redor do mundo. A compatibilidade eletromagnética refere-se à capacidade de um produto coexistir em um ambiente onde existe a presença de *EMI (do inglês, Electromagnetic Interference)* que traduzindo ao português significa Interferência Eletromagnética. Quando um produto atende a requisitos de compatibilidade eletromagnética ele não deve causar ou sofrer degradação funcional, danos ou apresentar erros de funcionamento nesse ambiente inóspito. A compatibilidade eletromagnética compreende duas áreas principais, emissões e imunidades. Portanto, um equipamento que atenda aos requisitos de EMC deve ser imune a EMI e radiar o mínimo possível de perturbações eletromagnéticas para não afetar o funcionamento de outros equipamentos.

3 METODOLOGIA

Segundo GIL (2008), para que um conhecimento possa ser considerado científico, torna-se necessário identificar as operações mentais e técnicas que possibilitam a sua verificação. Portanto, definir qual método utilizar para chegar ao conhecimento desejado.

De acordo com FERRARI (1974), o método científico é um traço característico da ciência, constituindo-se em instrumento básico que ordena, inicialmente, o pensamento e traça os procedimentos do cientista ao longo do caminho até atingir o objetivo específico preestabelecido.

A proposta deste trabalho visa a melhoria de um processo existente através da criação de um novo processo tecnologicamente mais aprimorado e que pode comprovar ou não a eficácia da solução proposta. Assim, esse trabalho é de natureza aplicada, que conforme lembram (PRODANOV; DE FREITAS, 2013) objetiva gerar conhecimentos para aplicações práticas voltadas à solução de problemas específicos.

Após as definições acima apresentadas, este trabalho busca comparar o método existente com o método proposto. Dessa forma, será possível estabelecer os procedimentos que podem garantir a obtenção dos resultados esperados.

3.1 Método existente

Atualmente as chaves de partida compensadora utilizam componentes eletromecânicos tanto para realizar a lógica de comando quanto a de força. Convém ressaltar que, para temporização se utilizam temporizadores eletrônicos. Vale lembrar que há cerca de duas décadas atrás, era comum encontrar temporizadores pneumáticos que se baseavam na compressão de um diafragma, para contagem de tempo.

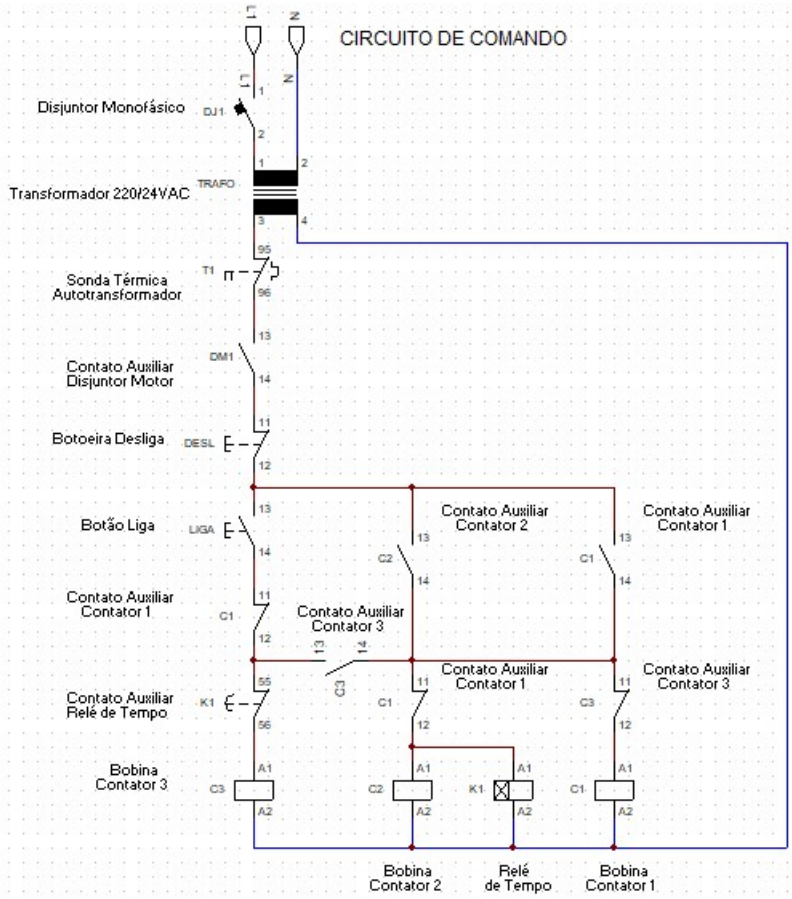
3.1.1 Circuito de Comando

A Figura 13 a seguir, apresenta o circuito de comando de uma partida compensada. Inicialmente chega o cabo alimentador de FASE passa por um disjuntor termomagnético que está em série com o primário do transformador 220/24Vac e a outra extremidade do transformador é conectada ao neutro. No secundário do

transformador uma extremidade será o neutro dos contatores e na outra a fase do circuito de comando. Convém ressaltar, o uso do transformador rebaixador torna a instalação elétrica mais segura devido ao nível menor de tensão elétrica e confere isolação galvânica. Dando continuidade, a fase (24Vac) é conectada em série com a sonda térmica que está inserida dentro do autotransformador, sendo ela responsável por monitorar a temperatura interna desse dispositivo. Ainda em série, há o contato auxiliar do disjuntor motor, esse impede a energização do circuito de comando. Tanto a sonda térmica quanto o contato auxiliar do disjuntor motor estão relacionados diretamente com a proteção termomagnética do circuito de força.

Ainda em série, estão as botoeiras de liga e desliga que são responsáveis respectivamente pela partida ou parada do motor. Conforme já mencionado, o relé de tempo é responsável pela temporização do tempo em que o autotransformador estará em série com o motor. Os outros contatos auxiliares presentes no circuito de comando são responsáveis pelo intertravamento da chave de partida de modo a evitar condições que poderiam colocar em risco a integridade do circuito.

Figura 13 - Circuito de Comando para uma chave Compensadora

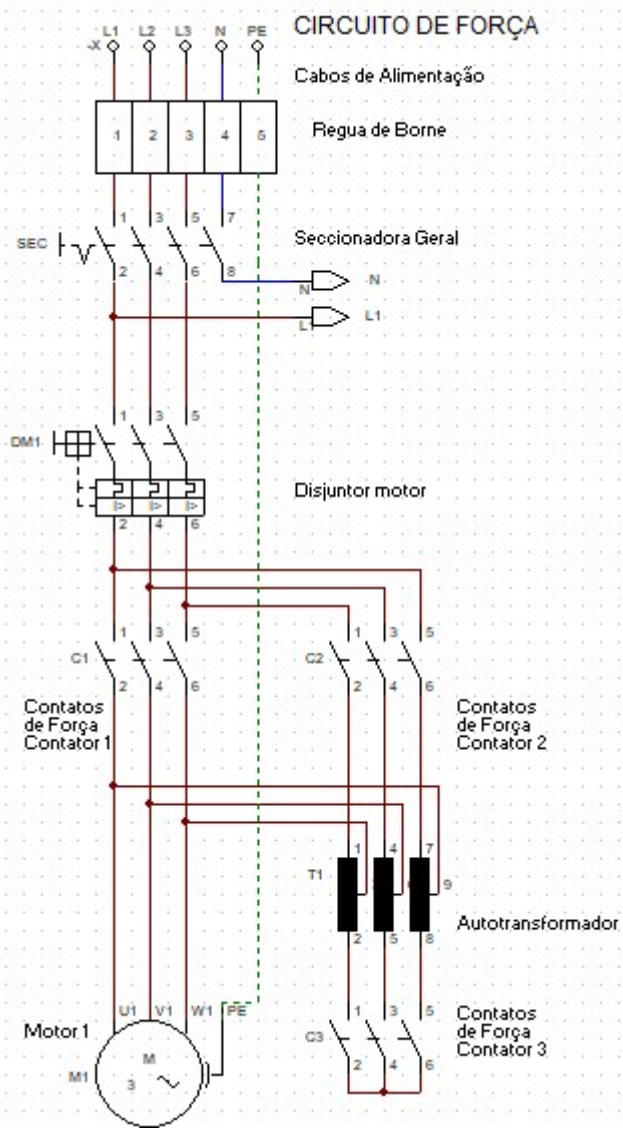


Fonte: Autor.

3.1.2 Circuito de Força

A Figura 14 a seguir, apresenta o circuito de força para uma partida compensada. Inicialmente chegam na régua de borne as três fases (R,S,T), neutro e terra. Após, os cabos saem da régua de borne e chegam na seccionadora geral responsável pela desenergização completa do quadro durante eventuais manutenções. Após a seccionadora, sairá um circuito de fase e neutro para utilização no circuito de comando. Além disso, também saem da seccionadora as três fases em direção ao disjuntor motor. Desse último, sairá concomitantemente as três fases para os contatos de força do contator 1, responsável pelo fechamento do motor após a partida aplicando sob ele tensão de linha também para os contatos de força do contator 2. A saída dos contatos de força do contator 1 chegam nos bornes do motor. Já a saída dos contatos de força do contator 2 chegam em uma das extremidades do autotransformador. A outra extremidade do autotransformador será conectada aos contatos de força do contator 3. As extremidades desse último, serão curto-circuitadas de modo a garantir o fechamento em estrela do autotransformador. Além disso, no centro do autotransformador sairá um cabo de cada bobina em direção ao motor. A escolha dos TAP'S dependerá do conjugado inerte do motor, sendo que para cargas mais pesadas o TAP deverá ser de um valor maior e para cargas mais leves pode-se utilizar um TAP de menor valor.

Figura 14 - Circuito de Força para uma chave Compensadora



Fonte: Autor.

3.1.3 Análise do Circuitos de Comando e de Força

Ao analisar os circuitos de comando e de força, percebe-se que existem vários dispositivos de proteção em série, para desarme em caso de anomalias, além de contatos intertravados que mais do que somente desarmar o sistema, garante que ele opere em “malha-fechada”, ou seja, a ação seguinte dependerá da anterior. Ademais, caso um contator abrir por algum motivo o sistema será desarmado. Dessa forma, o método proposto deverá levar em conta essa “confiabilidade” que o método existente apresenta.

3.2 Método proposto

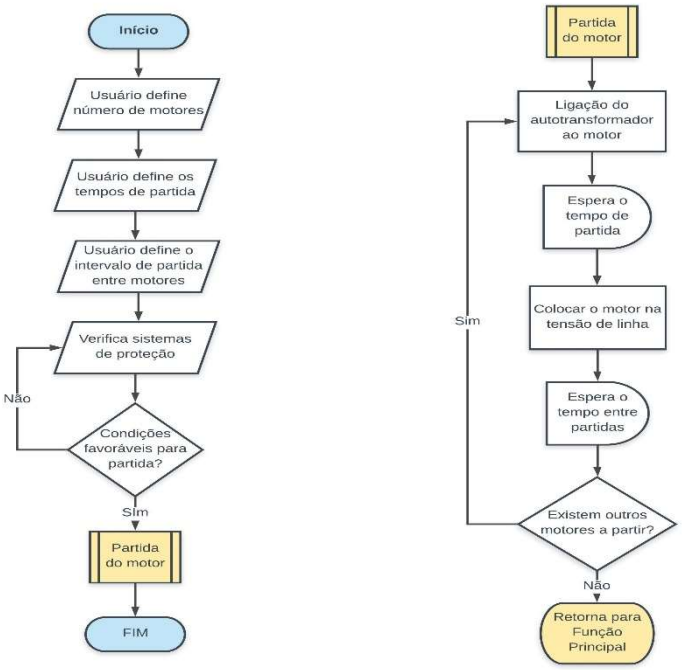
Conforme exposto no item *Objetivos Específicos*, esse trabalho visa o desenvolvimento de um sistema embarcado para substituir a tradicional lógica de comando eletromecânica com contatores, presente em chaves de partida compensadora. Convém ressaltar que não ocorreria a substituição de botoeiras, sinalizadores, dispositivos de proteção e contatores de força.

Esse método proposto visa a diminuição de custos, redução do tempo dispendido com manutenção e maior flexibilidade do sistema como um todo. Para isso, será desenvolvido um *firmware* que substituirá a lógica de contatores de comando e será embarcado em um microcontrolador. Feito isso, será desenvolvida uma placa de circuito impresso que possibilitará a escolha do número de motores utilizados que varia de 1 a 10, tempo destinado a partida de cada motor e compatibilidade com as atuais chaves de partida compensada. Além do mais, deverá ser projetados sistema de proteção contra ruídos eletromagnéticos, surtos de tensão, proteção contra sobrecorrente, drivers para acionamento dos contatores de força, entre outras eventuais necessidades que possam surgir durante o desenvolvimento do projeto.

3.2.1 Algoritmo de Comando

A Figura 15 a seguir, apresenta os procedimentos que deverão ser realizados pelo algoritmo de controle a ser embarcado em um microcontrolador. Inicialmente, percebe-se que o usuário insere os valores que atendam às necessidades da aplicação. Após, ocorre a verificação os dispositivos responsáveis pelo sistema de proteção da chave de partida. Caso não existam problemas então as partidas serão iniciadas. O autotransformador será conectado em série ao motor durante um período pré-estabelecido pelo usuário. Feito isso, o autotransformador irá esperar outro período para evitar que ocorra sobreaquecimento. E caso ainda existam outros motores a serem partidos, será seguido o mesmo procedimento de partida com seus respectivos tempos pré-estabelecidos pelo usuário. A seguir, a figura apresenta o fluxograma para o algoritmo de controle:

Figura 15 - Fluxograma do Algoritmo de Controle



Fonte: Autor.

3.2.2 Projeto da PCI

Para o projeto da placa de circuito impresso (PCI), utilizou-se o software *Eagle*[1] para desenvolvimento e análise de desempenho da placa de circuito impresso. Para a escolha do software de desenvolvimento, levou-se em conta critérios como interface do ambiente de projeto, disponibilidade de versão educacional, presença no mercado de trabalho, entre outros fatores.

[1] Disponível em: <https://www.autodesk.com/products/eagle/overview>. Acesso em: 5 jun. 2019.

4 DESENVOLVIMENTO

Este capítulo, está dividido em duas seções, de modo a permitir uma melhor compreensão: Implementação e Resultados e discussões.

4.1 Implementação

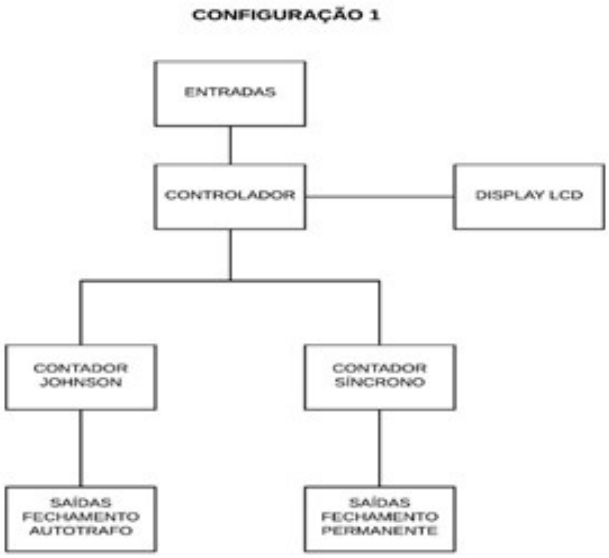
4.1.1 Simulação da Chave Compensadora

Para a implementação do projeto, inicialmente foram levantadas as necessidades inerentes a aplicação. Para tanto, realizou-se a simulação no software CAD_eSimu de uma chave compensadora para partida de dez motores trifásicos. Possibilitando assim, uma melhor compreensão a respeito da lógica de comando eletromecânica e as respectivas interações entre os seus componentes. O APÊNDICE A, apresenta detalhadamente a chave de partida compensada.

4.1.2 Representação em Diagrama de Blocos

A seguir, elaborou-se uma representação em diagrama de blocos a respeito de duas possíveis configurações equivalentes a lógica de comando eletromecânica, utilizando para isso, componentes eletrônicos. Sendo essas, apresentadas a seguir na Figura 16 e Figura 17:

Figura 16 - Diagrama de blocos da primeira configuração



Fonte: Autor.

Figura 17 - Diagrama de blocos da segunda configuração

Fonte: Autor.

Na sequência, será descrito o funcionamento de cada bloco:

- Entradas: *push buttons* (chave táctil sem retenção) utilizados para configuração dos parâmetros do controlador. Esses por sua vez, foram alocados da seguinte maneira: incremento, decremento, *reset* (reinício), *start* (início), *set/ok* (configura/confirma).
- Controlador: utilizou-se o microcontrolador PIC 18F4520 da empresa *Microchip*, que possui uma grande quantidade de pinos de entrada e saída, capacidade de processamento, entre outros fatores.
- Display de Cristal Líquido (do inglês, *Liquid Crystal Display*): esse tipo de display apresenta um ótimo custo-benefício. Além disso, é amplamente utilizado em equipamentos de baixo-custo. Nesse projeto, utilizou-se um *display* denominado JHD 162A, trata-se de um modelo composto de 2 linhas e 16 colunas (16x2).
- Contador Johnson: utilizou-se um circuito integrado da série 40xx, especificamente o CD4017BE da empresa *Texas Instruments.* O objetivo principal ao escolher esse circuito integrado, deve-se a capacidade de ligar apenas uma saída a cada pulso de *clock*, desligando as demais. Assim, percebeu-se que esse circuito integrado poderia ser usado para realizar o fechamento do autotransformador com o motor. Após o término do tempo pré-estabelecido pelo usuário, ocorreria a inserção de um pulso de *clock* para deslocar o “bit de acionamento para um estágio seguinte”

e assim sucessivamente, até que todas as partidas fossem inicializadas com o autotransformador. Convém destacar, como existe um intervalo de tempo entre as partidas, as saídas teriam de ser conectadas de alternadamente, ou seja, quando uma saída fosse utilizada a seguinte não poderia ser utilizada. Possibilitando assim, a existência desse “intervalo de tempo” entre os acionamentos;

- Contador síncrono: utilizou-se vários *flip-flops* tipo D com *preset* e *clear*. Trata-se especificamente do circuito integrado 74HC74AP da empresa Toshiba. O objetivo principal ao escolher esse circuito integrado, deve-se a sua capacidade de deslocar bits, ou seja, um após o outro. Diferentemente do contador anterior, nesse caso o objetivo é deslocar a cada pulso de *clock* o nível alto (ligado), contudo, também mantendo o nível alto (ligado) nas saídas anteriores. Assim, não ocorreria o desligamento dos motores que já foram colocados no fechamento em regime permanente;
- Saídas de fechamento: tanto para o fechamento com autotrafo (temporário) quanto para o fechamento em regime permanente, utilizou-se drives de saída a relé com uso de acoplamento óptico entre estágios (optoacoplador), transistor de junção bipolar (TBJ), diodo de roda-livre em paralelo a bobina do relé e diodo emissor de luz (LED), para sinalização ao usuário sobre o estado atual da saída visto que, na maioria das vezes os motores não se encontram no mesmo local onde ocorre o acionamento.

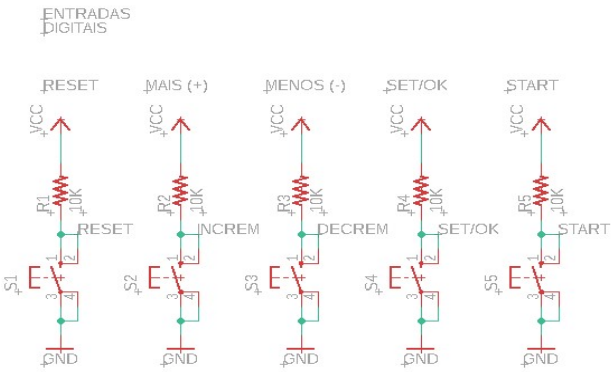
4.1.3 Esquemático eletrônico

Na sequência, realizou-se o desenvolvimento do esquemático eletrônico com seus respectivos componentes a serem comandados pelo algoritmo de controle *(firmware).* Como havia sido proposto duas configurações de *hardware,* elas serão apresentadas a seguir.

4.1.3.1 Configuração 1

Para o desenvolvimento desse esquemático, utilizou-se o software *Eagle* pertencente a empresa *Autodesk.* A Figura 18 a seguir, apresenta as entradas digitais do sistema.

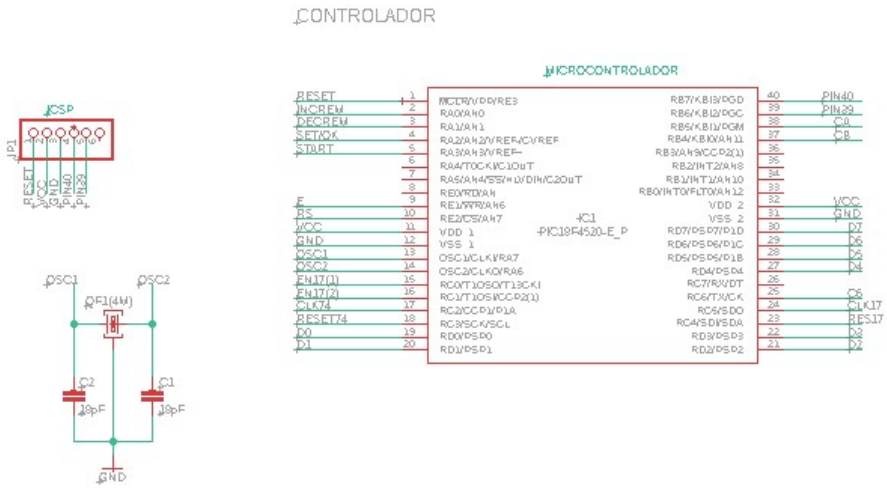
Figura 18 - Entradas digitais



Fonte: Autor.

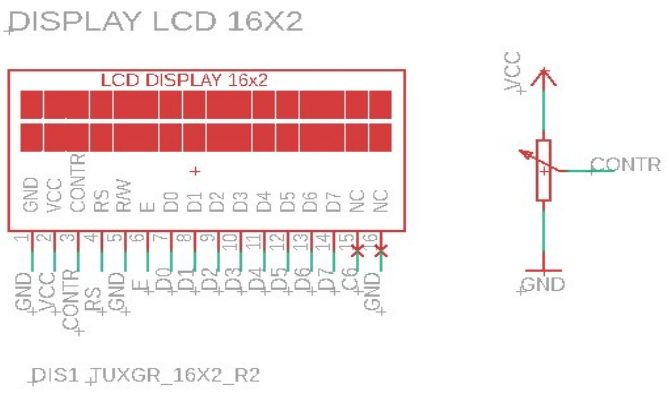
A Figura 19 e Figura 20 a seguir, apresentam respectivamente o Controlador e o Display LCD.

Figura 19 - Controlador



Fonte: Autor.

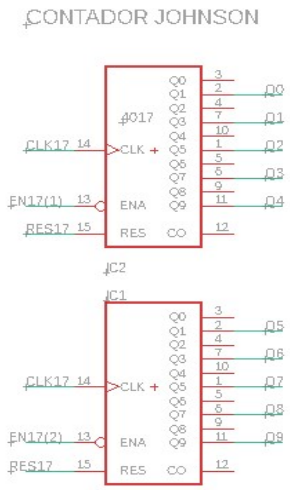
Figura 20 - Display LCD



Fonte: Autor.

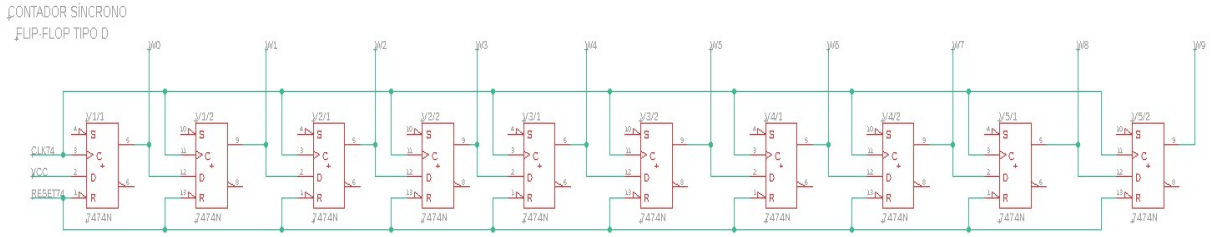
A Figura 21 e Figura 22 a seguir, apresentam respectivamente o Contadores Johnson e Contadores Síncronos.

Figura 21 - Contadores Johnson



Fonte: Autor.

Figura 22 - Contadores Síncronos



Fonte: Autor.

Para realizar o acionamento dos contatores, utilizou-se drivers a relé com 8 canais de saída. A Figura 23 a seguir, apresenta esse dispositivo:

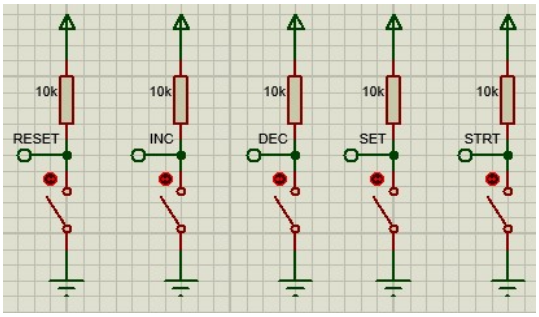
Figura 23 - Driver a relé com 8 canais de saída

Fonte: Autor.

4.1.3.2 Configuração 2

Para o desenvolvimento dessa configuração, utilizou-se o software *Proteus* pertencente a empresa *Labcenter Electronics.* A Figura 24 a seguir, apresenta as entradas digitais.

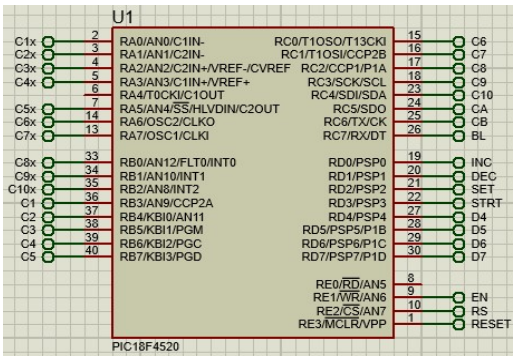
Figura 24 - Entradas digitais



Fonte: Autor.

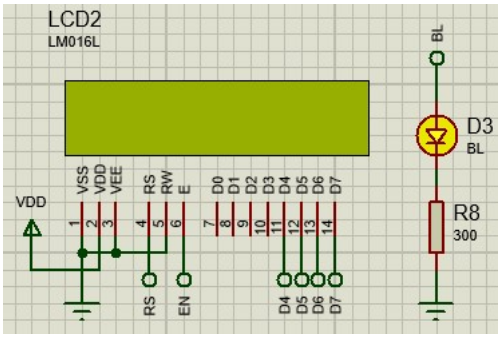
A Figura 25 e Figura 26 a seguir, apresentam respectivamente o Controlador e Display LCD.

Figura 25 - Controlador



Fonte: Autor.

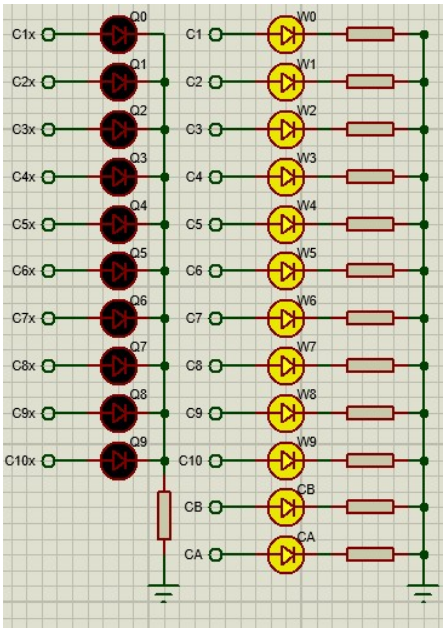
Figura 26 - Display LCD



Fonte: Autor.

A Figura 27 a seguir, apresenta as saídas para autotransformador e regime permanente.

Figura 27 - Saídas para Autotransformador e Regime Permanente



Fonte: Autor.

4.1.4 Algoritmo de controle

Para implementação do algoritmo de controle, foram realizados testes utilizando compiladores baseados em linguagem C, sendo esses: compilador *CCS*, da empresa *Custom Computer Services*, além do compilador MPLAB, da empresa *Microchip*.

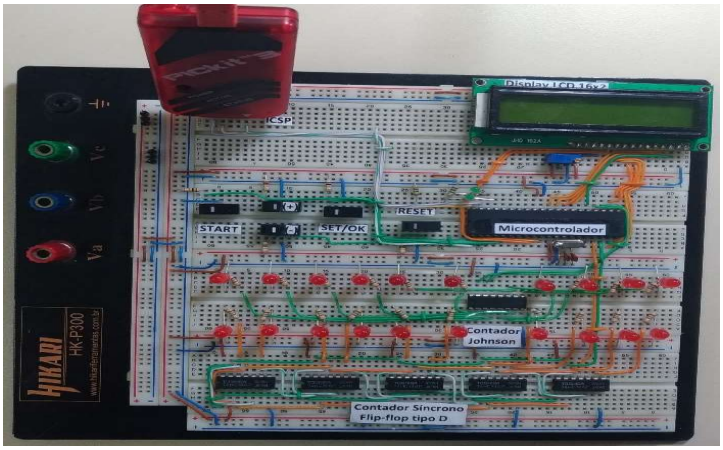
Porém, visando simplificar as coisas escolheu-se a Configuração 2, como *hardware* a ser controlado e o compilador MPLAB, devido a experiências anteriores em cadeiras do curso de Engenharia Elétrica. Assim, o arquivo referente a configuração do microcontrolador está no APÊNDICE B e o arquivo referente ao código-fonte do projeto está no APÊNDICE C.

4.1.5 Montagem do projeto em *Protoboard*

Para a validação do projeto e obtenção dos resultados, realizou-se a montagem do circuito em protoboard (ou matriz de contatos). A Figura 28 a seguir, apresenta a montagem do circuito na configuração que utiliza os contadores: Johnson e Síncrono. Para facilitar a realização de testes, utilizou-se o gravador PIC Kit 3 da empresa

Microchip, de modo que permitisse a gravação via ICSP (do inglês, In Circuit Serial Programming). Assim, evita-se o ato de inserir e retirar o microcontrolador repetidas vezes, pois isso, poderia ocasionar a ruptura física dos pinos ou possibilidade de descargas eletrostáticas durante a manipulação do componente.

Figura 28 - Montagem do circuito em protoboard

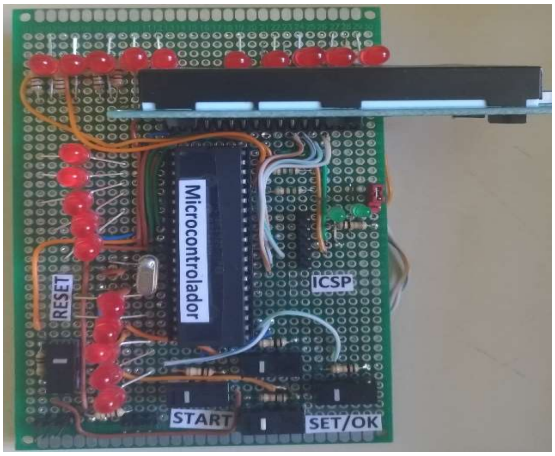


Fonte: Autor.

4.1.6 Montagem do projeto em Placa Universal

Realizou-se a montagem do circuito em placa universal (ou placa padrão). A Figura 29 a seguir, apresenta a montagem do circuito na configuração que não utiliza nenhum dos contadores. Para facilitar a realização de testes, utilizou-se a gravação via ICSP com o gravador PIC Kit 3 da empresa Microchip, novamente visando eliminar os possíveis riscos inerentes a inserção e retirada do chip repetidas vezes.

Figura 29 - Montagem do circuito em placa universal

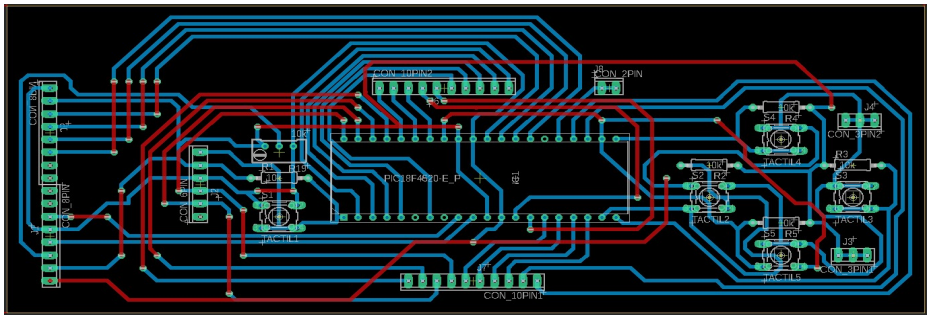


Fonte: Autor.

4.1.7 Desenvolvimento e montagem da Placa de Circuito Impresso

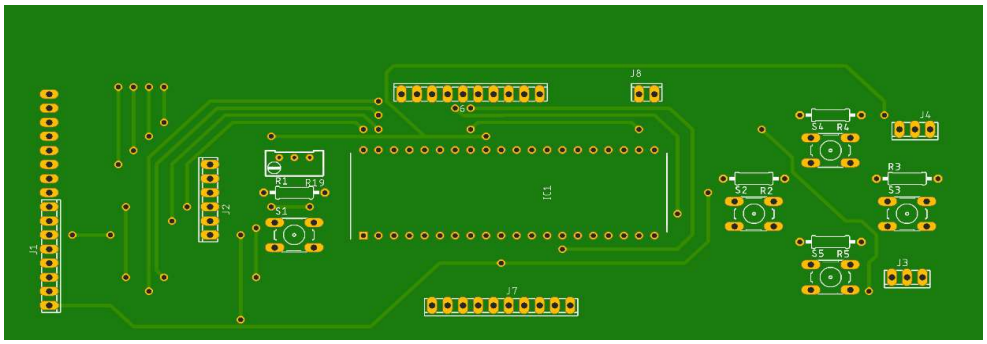
Na sequência, realizou-se o desenvolvimento e montagem da Placa de Circuito Impresso (PCI). Para o desenvolvimento do *layout* da placa, utilizou-se o *software Eagle*. Como havia somente a disponibilidade de placa de face única *(single layer)*, o roteamento da PCI levou em consideração esse fator. Surgindo assim, a necessidade de inserção de *jumpers.* A Figura 30 e Figura 31 a seguir, apresentam as diferentes etapas envolvidas no desenvolvimento da placa de circuito impresso.

Figura 30 - Layout da Placa de Circuito Impresso



Fonte: Autor.

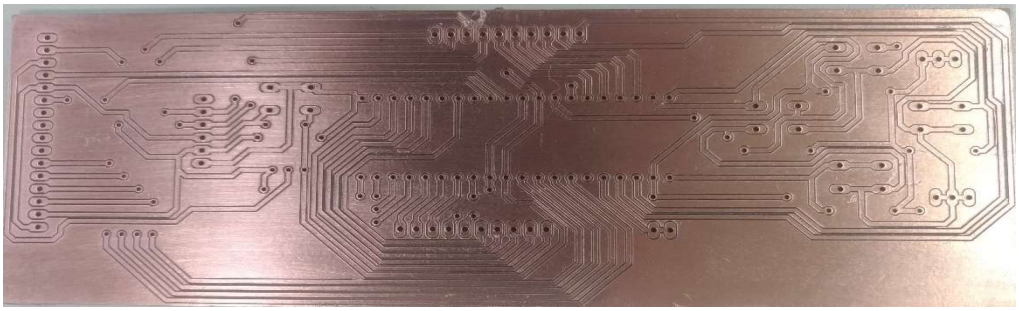
Figura 31 - Pré-visualização da Placa de Circuito Impresso



Fonte: Autor.

A confecção da Placa de Circuito Impresso, foi realizada em uma fresadora que utiliza Comando Numérico Computadorizado (CNC) para retirada do cobre presente na superfície da placa. A Figura 32 a seguir, apresenta a placa de circuito impresso após o processo de fresagem.

Figura 32 - Placa de Circuito Impresso após a fresagem



Fonte: Autor.

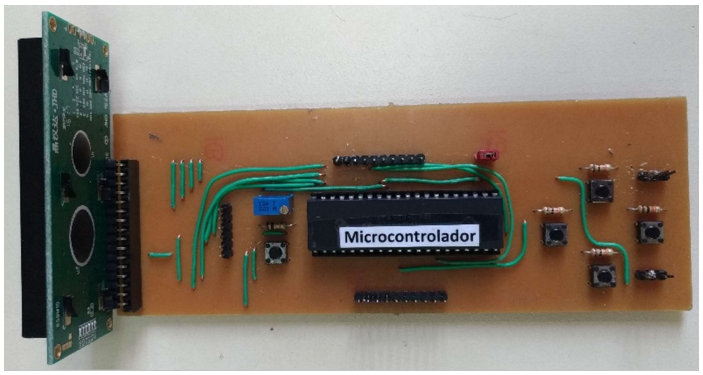
Na sequência, realizou-se a inserção e soldagem dos componentes eletrônicos na PCI, objetivando a conexão elétrica entre os elementos do sistema, conforme apresentado na Figura 33.

Figura 33 - Processo de soldagem dos componentes

Fonte: Autor.

A Figura 34 a seguir, apresenta o processo de montagem da PCI finalizado.

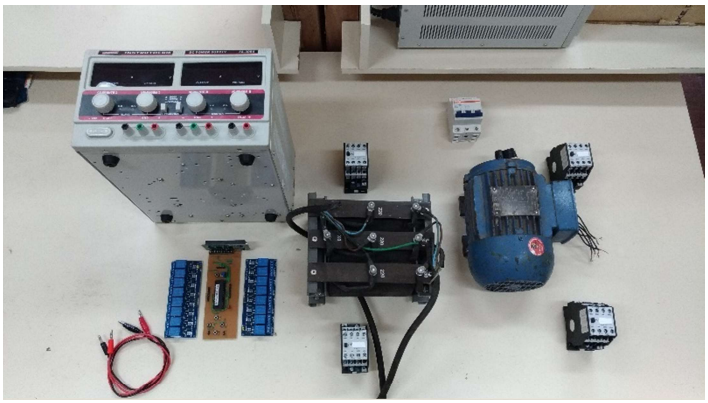
Figura 34 - Processo de montagem da PCI finalizado



Fonte: Autor.

A Figura 35 a seguir, apresenta os materiais a serem utilizados na implementação do projeto.

Figura 35 - Materiais a serem utilizados na implementação do projeto

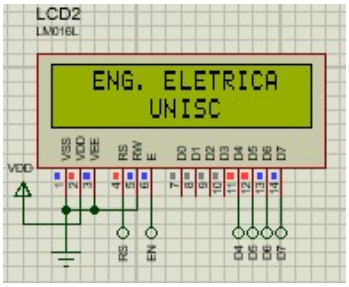


Fonte: Autor.

4.2 Resultados e discussões

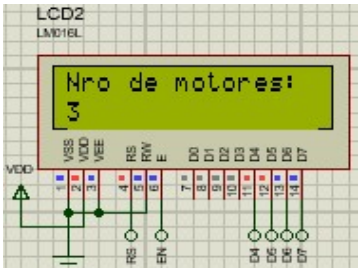
As imagens a seguir, compreendidas da Figura 36 à Figura 44, apresentam os resultados obtidos em simulação durante a configuração, para partida compensada sequencial de 3 motores trifásicos. Durante essa configuração, parametrizou-se que o tempo de partida de cada motor, durante o fechamento com o autotransformador seria de 1 minuto. Além disso, o intervalo de tempo entre cada partida, também seria de 1 minuto.

Figura 36 - Tela inicial



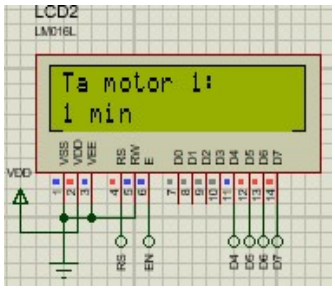
Fonte: Autor.

Figura 37 - Inserção do número de motores



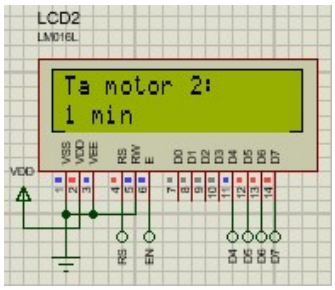
Fonte: Autor.

Figura 38 - Tempo de acionamento do primeiro motor com o autotransformador em série



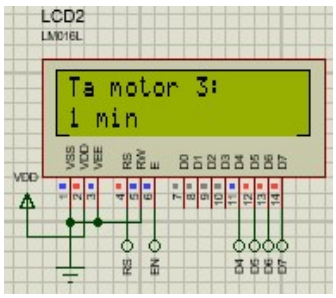
Fonte: Autor.

Figura 39 - Tempo de acionamento do segundo motor com o autotransformador em série



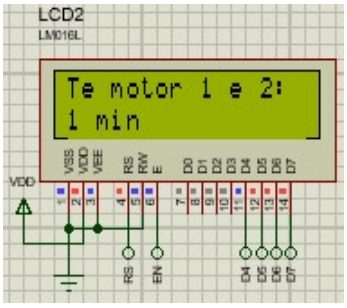
Fonte: Autor.

Figura 40 - Tempo de acionamento do terceiro motor com o autotransformador em série



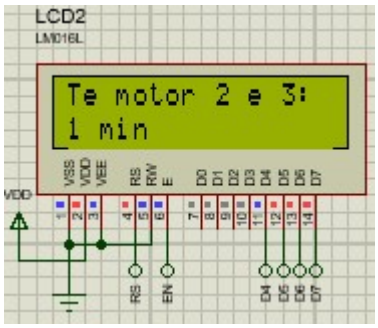
Fonte: Autor.

Figura 41 - Intervalo de tempo entre a partida do motor 1 e motor 2



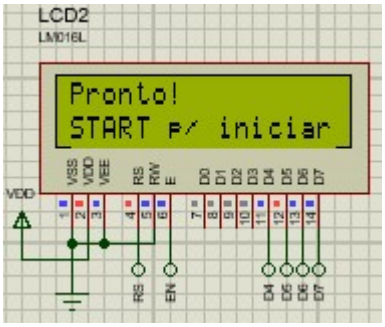
Fonte: Autor.

Figura 42 - Intervalo de tempo entre a partida do motor 2 e motor 3



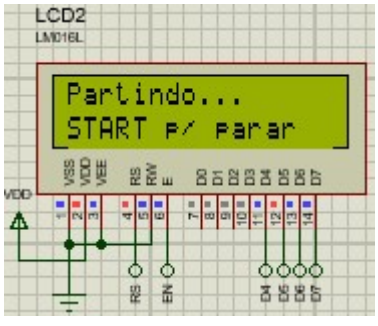
Fonte: Autor.

Figura 43 - Configuração finalizada e aguardando pressionar a tecla START



Fonte: Autor.

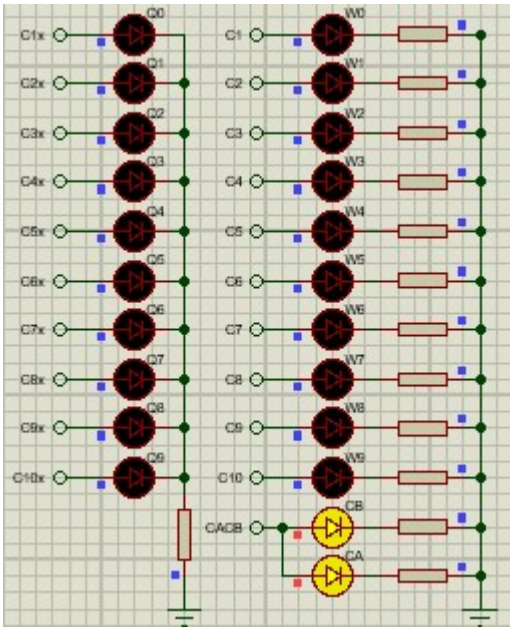
Figura 44 - Início da partida dos motores



Fonte: Autor.

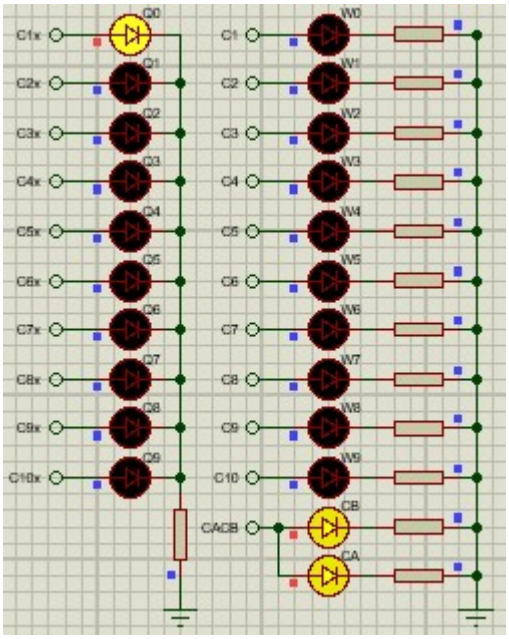
As imagens a seguir, compreendidas da Figura 45 à Figura 51, apresentam as diferentes etapas de sinalização durante a partida dos motores.

Figura 45 - Sinalização de fechamento do autotransformador



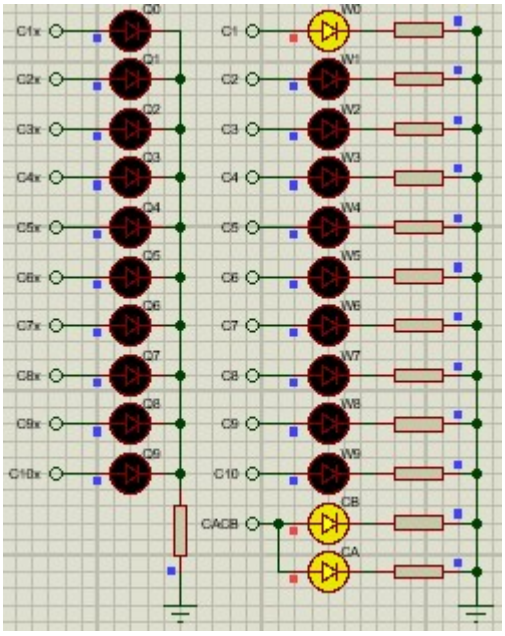
Fonte: Autor.

Figura 46 - Sinalização da ligação série do autotransformador com o primeiro motor (regime transitório)



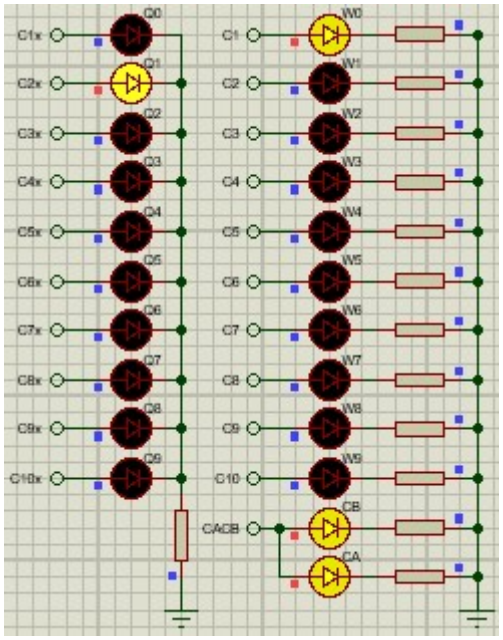
Fonte: Autor.

Figura 47 - Sinalização de fechamento do primeiro motor em regime permanente



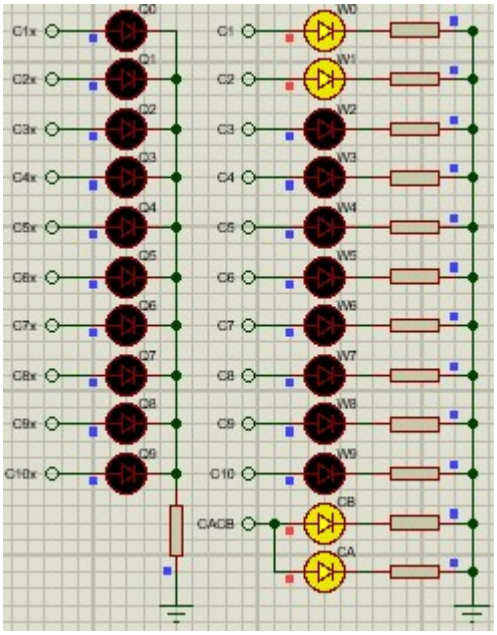
Fonte: Autor.

Figura 48 - Sinalização de fechamento do primeiro motor em regime permanente e ligação série do autotransformador com o segundo motor



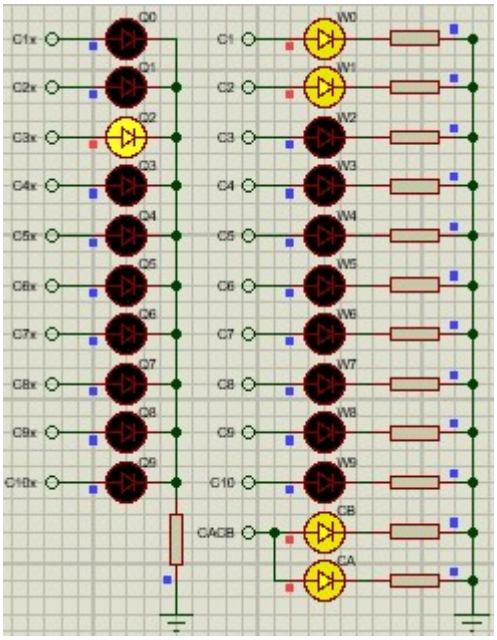
Fonte: Autor.

Figura 49 - Sinalização de fechamento do primeiro e segundo motores em regime permanente



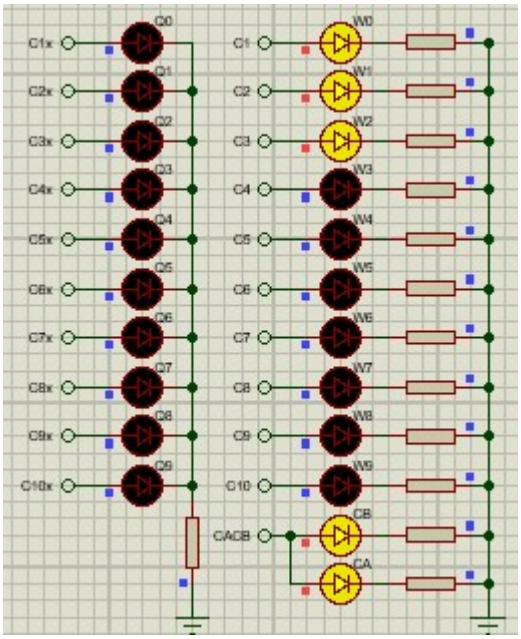
Fonte: Autor.

Figura 50 - Sinalização de fechamento do primeiro e segundo motores em regime permanente e ligação série do autotransformador com o terceiro motor



Fonte: Autor.

Figura 51 - Sinalização de fechamento do primeiro, segundo e terceiro motores em regime permanente



Fonte: Autor.

Em relação a implementação física do projeto, não houve resultados positivos que comprovassem o funcionamento do projeto. Atribui-se o resultado obtido a inobservância de determinados procedimentos durante a implementação. Sendo esses:

- Medição de continuidade na protoboard, *jumpers*, placa universal, entre outros. Essa não verificação, possibilita o surgimento de mau contato durante a realização dos testes;

- Minimização de ruídos e crosstalk provenientes do próprio meio físico (protoboard e placa universal), onde realizou-se a montagem do projeto;

- Processo de soldagem inadequada dos componentes eletrônicos;

- Pouco espaçamento entre trilhas e PAD’s na placa de circuito impresso. Dessa forma, se eleva consideravelmente a possibilidade de erros e ligações indesejadas, durante a soldagem dos componentes eletrônicos;

Assim, o conjunto desses fatores contribuíram para que a implementação física do projeto não obtivesse o êxito esperado.

5 CONCLUSÃO

O presente trabalho, apresentou as técnicas utilizadas para o desenvolvimento de um sistema embarcado, para partida compensada sequencial de motores trifásicos. Através do método proposto nesta obra, foi possível verificar em detalhes o funcionamento da partida compensada e propor um método alternativo. Convém ressaltar que foram propostas diferentes configurações de hardware, a fim de explicitar as diferentes possibilidades de projeto.

Os resultados obtidos em simulação, indicam que o projeto é capaz de atender aos requisitos da aplicação. Contudo, cabe ressaltar que não se obteve resultados durante a implementação física do projeto, devido a inobservância de determinados procedimentos de natureza prática.

Por fim, torna-se evidente que com procedimentos que agreguem confiabilidade durante a implementação física do projeto, pesquisa de mercado consumidor, adequações no custo de produção e no processo produtivo, seria possível desenvolver um produto em escala comercial com preços competitivos.

5.1 Sugestões para trabalhos futuros

Tendo em vista, possíveis trabalhos decorrentes dessa primeira implementação ou melhorias nesse projeto, será proposto sugestões em três segmentos distintos.

A primeira sugestão, propõe a implementação deste projeto com um microcontrolador de menor custo, em contrapartida, seria necessário implementar *hardware* externo, conforme proposto na configuração 1.

A segunda sugestão, propõe a integração do equipamento com sistemas de supervisão, acesso remoto, histórico de partidas, entre outras possiblidades. Dessa forma, possibilitaria o alinhamento ao conceito de Internet das Coisas e Indústria 4.0.

Por fim, a terceira sugestão propõe o desenvolvimento de um sistema de controle capaz de realizar a comutação automática dos TAP’s do autotransformador, em função do conjugado de partida. Assim, caso ocorram variações no conjugado de partida, automaticamente mudariam os tempos de comutação do autotransformador. Possibilitando assim, o sistema trabalhar em seu ponto ótimo e evitando as indesejáveis correntes de pico, queda de tensão, entre outros problemas.

6 REFERÊNCIAS

6.1 Livros

AHO, A.; SETHI, R.; ULLMAN, J. *Compiladores: Princípios, técnicas y herramientas*. 1. ed. México: Addison Wesley Iberoamericana, 1990.

ALMEIDA, R.; MORAES, C.; SERAPHIM, T. *Programação de Sistemas Embarcados: Desenvolvendo Software para Microcontroladores em Linguagem C*. 1. ed. Rio de Janeiro: Editora Elsevier, 2016.

CORMEN, Thomas H. *Desmistificando Algoritmos*. 1. ed. Rio de Janeiro: Editora Elsevier, 2014.

CUI, Zheng. *Printed Electronics: Materials, Technologies and Applications*. 1. ed. China: Higher Education Press, 2016.

FERRARI, Alfonso T. *Metodologia da Ciência*. 3. ed. Rio de Janeiro: Kennedy, 1974.

FRANCHI, Claiton M. *Acionamentos Elétricos*. 4. ed. São Paulo: Editora Érica, 2008.

GIL, Antônio C. *Métodos e Técnicas de Pesquisa Social*. 6. ed. São Paulo: Editora Atlas, 2008.

GIMENEZ, Salvador P. *8051 Microcontrollers: Fundamental Concepts, Hardware, Software and Applications in Electronics*. São Paulo: Editora Springer, 2019.

HESTER, R.; HARRISON, R. *Electronic Waste Management: Design, Analysis and Application.* United Kingdom: RSC, 2009.

HUANG, Han W. *PIC Microcontroller: An Introduction to Software and Hardware Interfacing.* United States of America: Delmar Learning, 2005.

KHANDPUR, R. S. *Printed Circuit Boards: Design, Fabrication, Assembly and Testing*. 1. ed. New Delhi: Tata McGraw-Hill, 2005.

MONTROSE, Mark I. *EMC and the Printed Circuit Board: Design, Theory, and Layout Made Simple*. 1. ed. United States of America: IEEE Press, 1999.

PEREZ, F.; ARENY, R. *Microcontrollers: Fundamentals and Applications with PIC.* United States of America: CRC Press, 2009.

PRODANOV, C.; FREITAS, E. *Metodologia do Trabalho Científico: Métodos e Técnicas de Pesquisa e do Trabalho Acadêmico.* 2 ed. Novo Hamburgo: Feevale, 2013.

SCHROEDER, Chris. *Printed Circuit Board Design Using AutoCAD*. 1. ed. United States of America: Newnes, 1998.

SUN, J. et al. *Embedded Firmware Solutions: Development Best Practices for the Internet of Things.* 1. ed. United States of America: Apress Open, 2015.

TRAISTER, John E. *Design Guidelines for Surface Mount Technology*. 1. ed. California: Academic Press, 1990.

VISWANADHAM, P.; SINGH, P. *Failure Modes and Mechanisms in Electronic Packages.* United States of America: Springer Science + Business Media Dordrecht, 1998.

WAZLAWICK, Raul S. *História da Computação*. 1. ed. Rio de Janeiro: Editora Elsevier, 2016.

6.2 Imagens

AWDIY

Disponível em: <http://www.awdiy.com/index.php?page=deq-psu-schematics>. Acesso em: 05 jun. 2019.

Chipsetc
Disponível em: <http://www.chipsetc.com/integrated-circuit-package-types.html>. Acesso em: 05 jun. 2019.

Electronics-lab
Disponível em: <http://www.electronics-lab.com/how-to-get-free-coupons-for-your-next-pcb-project-using-pcbway/>. Acesso em: 05 jun. 2019.

Electrosome
Disponível em: <https://electrosome.com/getting-started-pic-ccs-c/>. Acesso em: 05 jun. 2019.

Etna Transformadores
Disponível em: <https://etnatransformadores.com.br/transformadores-etna/auto-transformador/ >. Acesso em: 05 jun. 2019.

IIOT World
Disponível em: <https://iiot-world.com/connected-industry/a-guide-for-selecting-the-right-microcontroller-for-your-iot-project/>. Acesso em: 05 jun. 2019.

JVD PCB
Disponível em: <http://jvdpcb.com/en/product.aspx?id=8>. Acesso em: 05 jun. 2019.

Portal da Indústria
Disponível em: <http://www.portaldaindustria.com.br/cni/canais/industria-2027/noticias/industria-40-saltara-de-16-para-218-das-empresas-em-uma-decada-diz-pesquisa-da-cni/>. Acesso em: 3 abr. 2019.

SG Computers
Disponível em: <https://sg-computers.com/en/server-motherboard/gigabyte-mx31-bs0-motherboard-detail >. Acesso em: 05 jun. 2019.

Ventro

Disponível em: <https://ventro.com.br/produto/Kit-Sequencial-De-Leds-4017-Ne555--Placa-Teste-De-Solda-Smd-em-Bras%C3%ADlia_MLB711818888>. Acesso em: 05 jun. 2019.

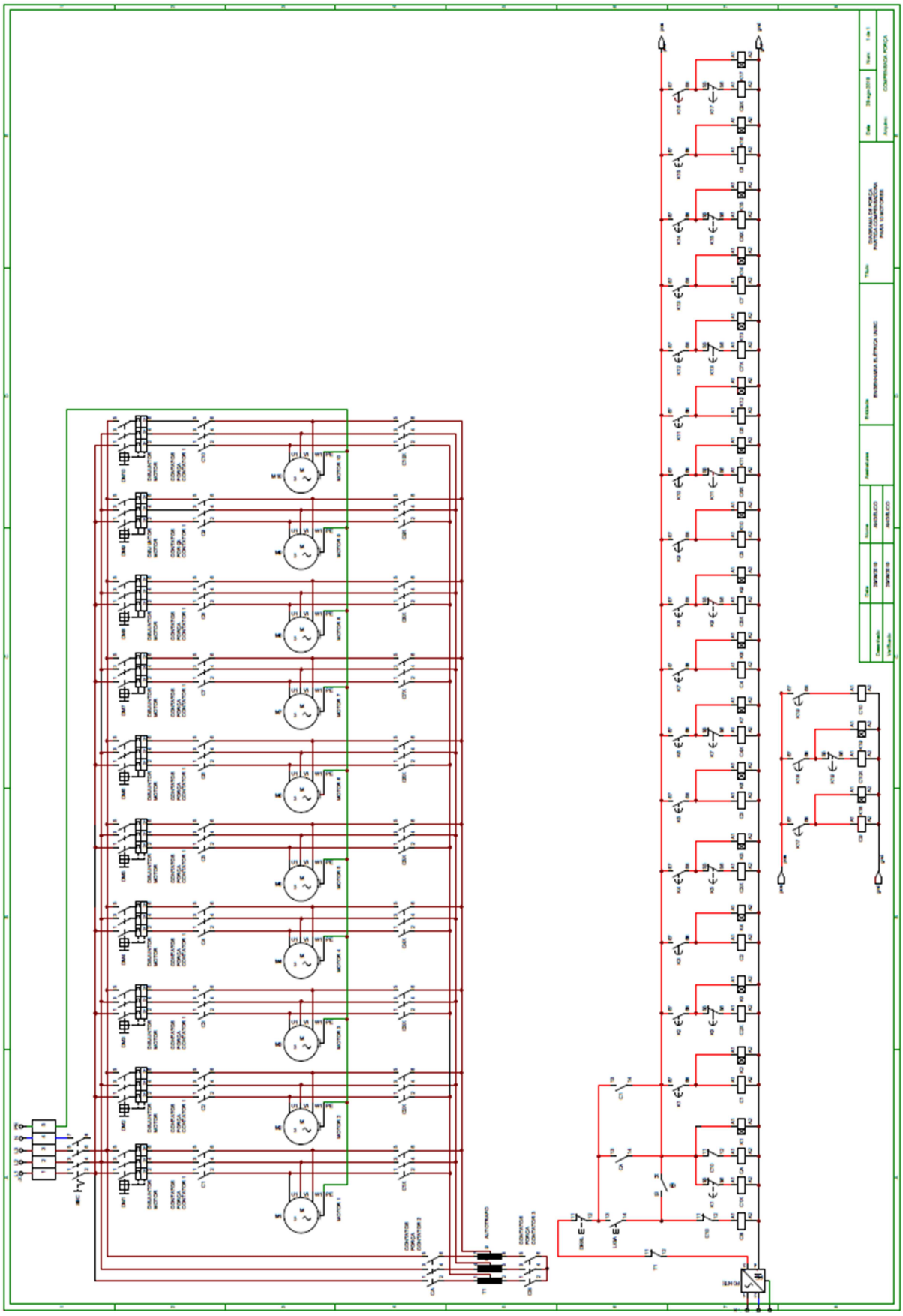
Well PCB

Disponível em: <https://www.wellpcb.com/sites/default/files/Multilayer%20PCB%20Pool%209.jpg>. Acesso em: 05 jun. 2019.

WIKISIS

Disponível em: <http://wiki.sjs.org/wiki/index.php/History_of_Computers_-_PCB>. Acesso em: 05 jun. 2019.

7 APÊNDICE A



8 APÊNDICE B

C:\Users\Angélico PC\Desktop\TCC-Angelico-20191010T140927Z-001\TCC-Angelico\TCC-Angelico-PIC18.X\config.h

```
// PIC18F4520 Configuration Bit Settings

// 'C' source line config statements

// CONFIG1H
#pragma config OSC = INTIO67        // Oscillator Selection bits (HS oscillator)
#pragma config FCMEN = OFF      // Fail-Safe Clock Monitor Enable bit (Fail-Safe Clock Monitor disabled)
#pragma config IESO = OFF       // Internal/External Oscillator Switchover bit (Oscillator Switchover mode disabled)

// CONFIG2L
#pragma config PWRT = OFF       // Power-up Timer Enable bit (PWRT disabled)
#pragma config BOREN = SBORDIS  // Brown-out Reset Enable bits (Brown-out Reset enabled in hardware only (SBOREN is disabled))
#pragma config BORV = 3         // Brown Out Reset Voltage bits (Minimum setting)

// CONFIG2H
#pragma config WDT = OFF        // Watchdog Timer Enable bit (WDT disabled (control is placed on the SWDTEN bit))
#pragma config WDTPS = 32768    // Watchdog Timer Postscale Select bits (1:32768)

// CONFIG3H
#pragma config CCP2MX = PORTC   // CCP2 MUX bit (CCP2 input/output is multiplexed with RC1)
#pragma config PBADEN = ON      // PORTB A/D Enable bit (PORTB<4:0> pins are configured as analog input channels on Reset)
#pragma config LPT1OSC = OFF    // Low-Power Timer1 Oscillator Enable bit (Timer1 configured for higher power operation)
#pragma config MCLRE = ON       // MCLR Pin Enable bit (MCLR pin enabled; RE3 input pin disabled)

// CONFIG4L
#pragma config STVREN = ON      // Stack Full/Underflow Reset Enable bit (Stack full/underflow will cause Reset)
#pragma config LVP = OFF        // Single-Supply ICSP Enable bit (Single-Supply ICSP disabled)
#pragma config XINST = OFF      // Extended Instruction Set Enable bit (Instruction set extension and Indexed Addressing mode disabled (Legacy mode))

// CONFIG5L
#pragma config CP0 = OFF        // Code Protection bit (Block 0 (000800-001FFFh) not code-protected)
#pragma config CP1 = OFF        // Code Protection bit (Block 1 (002000-003FFFh) not code-protected)
#pragma config CP2 = OFF        // Code Protection bit (Block 2 (004000-005FFFh) not code-protected)
#pragma config CP3 = OFF        // Code Protection bit (Block 3 (006000-007FFFh) not code-protected)

// CONFIG5H
#pragma config CPB = OFF        // Boot Block Code Protection bit (Boot block (000000-0007FFh) not code-protected)
#pragma config CPD = OFF        // Data EEPROM Code Protection bit (Data EEPROM not code-protected)

// CONFIG6L
#pragma config WRT0 = OFF       // Write Protection bit (Block 0 (000800-001FFFh) not write-protected)
#pragma config WRT1 = OFF       // Write Protection bit (Block 1 (002000-003FFFh) not write-protected)
#pragma config WRT2 = OFF       // Write Protection bit (Block 2 (004000-005FFFh) not write-protected)
#pragma config WRT3 = OFF       // Write Protection bit (Block 3 (006000-007FFFh) not write-protected)

// CONFIG6H
#pragma config WRTC = OFF       // Configuration Register Write Protection bit (Configuration registers (300000-3000FFh) not write-protected)
#pragma config WRTB = OFF       // Boot Block Write Protection bit (Boot block (000000-0007FFh) not write-protected)
#pragma config WRTD = OFF       // Data EEPROM Write Protection bit (Data EEPROM not write-protected)

// CONFIG7L
#pragma config EBTR0 = OFF      // Table Read Protection bit (Block 0 (000800-001FFFh) not protected from table reads executed in other blocks)
#pragma config EBTR1 = OFF      // Table Read Protection bit (Block 1 (002000-003FFFh) not protected from table reads executed in other blocks)
#pragma config EBTR2 = OFF      // Table Read Protection bit (Block 2 (004000-005FFFh) not protected from table reads executed in other blocks)
#pragma config EBTR3 = OFF      // Table Read Protection bit (Block 3 (006000-007FFFh) not protected from table reads executed in other blocks)

// CONFIG7H
#pragma config EBTRB = OFF      // Boot Block Table Read Protection bit (Boot block (000000-0007FFh) not protected from table reads executed in other blocks)

// #pragma config statements should precede project file includes.
// Use project enums instead of #define for ON and OFF.

#include <xc.h>

#define _XTAL_FREQ 8000000UL // Crystal frequency used to calculate delay functions
```

9 APÊNDICE C

C:\Users\Angélico PC\Desktop\TCC-Angelico-20191010T140927Z-001\TCC-Angelico\TCC-Angelico-PIC18.X\main.c

```
#include "config.h"
#include <pic18f4520.h>

/************************ Macros */
#define C1X_PIN LATAbits.LA0    // C1x pin
#define C2X_PIN LATAbits.LA1    // C2x pin
#define C3X_PIN LATAbits.LA2    // C3x pin
#define C4X_PIN LATAbits.LA3    // C4x pin
#define C5X_PIN LATAbits.LA5    // C5x pin
#define C6X_PIN LATAbits.LA6    // C6x pin
#define C7X_PIN LATAbits.LA7    // C7x pin
#define C8X_PIN LATBbits.LB1    // C8x pin
#define C9X_PIN LATBbits.LB2    // C9x pin
#define C10X_PIN LATBbits.LB3   // C10x pin

#define C1_PIN LATBbits.LB4     // C1 pin
#define C2_PIN LATBbits.LB5     // C2 pin
#define C3_PIN LATBbits.LB6     // C3 pin
#define C4_PIN LATBbits.LB7     // C4 pin
#define C5_PIN LATCbits.LC0     // C5 pin
#define C6_PIN LATCbits.LC1     // C6 pin
#define C7_PIN LATCbits.LC2     // C7 pin
#define C8_PIN LATCbits.LC3     // C8 pin
#define C9_PIN LATCbits.LC4     // C9 pin
#define C10_PIN LATCbits.LC5    // C10 pin
#define CACB_PIN LATCbits.LC6   // Auto-tranformer CA/CB pin
#define CX_C_DELAY 2            // Q to W delay in ms
#define AT_DELAY 1000           // Auto-tranformer power delay in ms

#define BTN_INC_PIN PORTDbits.RD0   // Button increment pin
#define BTN_DEC_PIN PORTDbits.RD1   // Button decrement pin
#define BTN_SET_PIN PORTDbits.RD2   // Button set pin
#define BTN_STRT_PIN PORTDbits.RD3  // Button start pin
#define BTN_DELAY 250               // Button debounce delay in ms

#define LCD_EN_PIN LATEbits.LE1     // LCD EN pin
#define LCD_RS_PIN LATEbits.LE2     // LCD RS pin
#define LCD_PORT LATD               // LCD data port
#define LCD_CMD_CLEAR 0x01          // LCD clear display command
#define LCD_CMD_SHIFT_L 0x18        // LCD shift left command
#define LCD_CMD_SHIFT_R 0x0C        // LCD shift right command
#define LCD_CMD_LINE1 0x80          // LCD first line command
#define LCD_CMD_LINE2 0xC0          // LCD second lins command

/************************ Function declarations */
void delay_ms(unsigned int ms);
void CLK_init(void);
void IO_init(void);
void LCD_init(void);
void INTE_init(void);
void LCD_write(unsigned char type, char data);
void LCD_write_cmd(char cmd);
void LCD_write_c(char c);
void LCD_write_s(const char *s);
```

```
55 void motors_run(void);
56 void motors_stop(void);
57 void menu_clear_update(void);
58 void menu_error(void);
59 void menu_boot(void);
60 void menu_n_motors(void);
61 void menu_n_motors(void);
62 void menu_s_time(void);
63 void menu_w_time(void);
64 void menu_ready(void);
65 void menu_run(void);
66 void menu_running(void);
67
68 /************************ Global variables */
69 unsigned int s_time[10] = {1, 1, 1, 1, 1, 1, 1, 1, 1, 1}; // Starting time
70 unsigned int w_time[9] = {1, 1, 1, 1, 1, 1, 1, 1, 1}; // Waiting time
71 unsigned char n_motors = 1; // Number of motors
72 unsigned char menu = 0; // Menu index
73 unsigned char menu_motor_i = 0; // Motor index for menus
74 unsigned char menu_update_flag = 1; // Flag used to update menu text
75
76 /**
77  * Main
78  */
79 int main(void) {
80     CLK_init(); // Initialize clock
81     IO_init(); // Initialize I/O
82     LCD_init(); // Initialize LCD
83     motors_stop(); // Reset motors
84
85     // Main loop
86     while (1) {
87         // Menu
88         switch (menu) {
89             case 0: menu_boot(); // Boot menu
90                 break;
91             case 1: menu_n_motors(); // Set the number of motors
92                 break;
93             case 2: menu_s_time(); // Set the starting time for each motors
94                 break;
95             case 3: menu_w_time(); // Set the waiting time between motors
96                 break;
97             case 4: menu_ready(); // Wait for start command
98                 break;
99             case 5: menu_run(); // Run starting sequence
100                break;
101            case 6: menu_running(); // Motor running
102                break;
103            default: menu_boot(); // Boot menu
104                break;
105        }
106    }
107
108    return (EXIT_SUCCESS);
109 }
110
111 /**
112  * Variation of the __delay_ms function to accept a variable as parameter
```

```
113  * @param ms Time in milliseconds
114  */
115 void delay_ms(unsigned int ms) {
116     while (ms > 0) {
117         __delay_us(994); // Tuned for more precision
118         ms--;
119     }
120 }
121
122 /**
123  * Clock initialization
124  */
125 void CLK_init(void) {
126     /**
127      * REGISTER 2-2: OSCCON REGISTER
128      */
129     OSCCONbits.IRCF = 7; // 8MHZ
130     OSCCONbits.SCS = 0; // Use internal clock as primary oscillator
131     while (!OSCCONbits.IOFS); // Wait clock stable
132 }
133
134 /**
135  * I/O initialization
136  */
137 void IO_init(void) {
138     /**
139      * REGISTER 19-1: ADCON0 REGISTER
140      */
141     ADCON0bits.ADON = 0; //  A/D converter module is disabled
142
143     /**
144      * REGISTER 19-2: ADCON1 REGISTER
145      */
146     ADCON1 = 0x0f; // All pins as digital I/O
147
148     /**
149      * REGISTER 20-1: CMCON REGISTER
150      */
151     CMCONbits.CM = 7; // Disable Comparator
152
153     /**
154      * 10.1 PORTA, TRISA and LATA Registers
155      * Outputs to C1x-C7x
156      */
157     TRISA = 0;
158
159     /**
160      * 10.2 PORTB, TRISB and LATB Registers
161      * Input to INT0
162      * Outputs to C8x-C10 and C1-C4
163      */
164     TRISB = 0x01;
165
166     /**
167      * 10.3 PORTC, TRISC and LATC Registers
168      * Outputs to C5-C10 and CA_CB
169      */
170     TRISC = 0;
```

```
171
172     /**
173      * 10.4 PORTD, TRISD and LATD Registers
174      * Inputs to INC, DEC, SET, STRT
175      * Outputs to LCD D4-7
176      */
177     TRISD = 0x0f;
178
179     /**
180      * 10.5 PORTE, TRISE and LATE Registers
181      */
182     TRISEbits.RE1 = 0; // Output to LCD EN
183     TRISEbits.RE2 = 0; // Output to LCD RS
184 }
185
186 /**
187  * LCD initialization
188  */
189 void LCD_init(void) {
190     unsigned char i = 0;
191     char LCD_INIT_CMDS[8] = {
192         0x03, // Initialization command
193         0x03, // Initialization command
194         0x03, // Initialization command
195         0x02, // Initialization command
196         0x28, // Mode 4-bit, 2 lines, 5x7 dots
197         0x0c, // Display on, cursor off
198         0x01, // Clear display
199         0x06 // Entry mode
200     };
201
202     LCD_RS_PIN = 0;
203     LCD_EN_PIN = 0;
204     __delay_ms(15); // LCD Power on delay
205
206     // Send initialization commands
207     for (i = 0; i < sizeof (LCD_INIT_CMDS); i++) {
208         LCD_write_cmd(LCD_INIT_CMDS[i]);
209     }
210 }
211
212 /**
213  * External interrupt initialization
214  */
215 void INTE_init(void) {
216     /**
217      * REGISTER 9-2: INTCON2 REGISTER
218      */
219     INTCON2bits.INTEDG0 = 0; // Set interrupt detection to falling edge for INT0
220
221     /**
222      * REGISTER 9-1: INTCON REGISTER
223      */
224     INTCONbits.INT0F = 0; // Clear External Interrupt Flag bit for INT0
225     INTCONbits.INT0IE = 1; // Enable External Interrupt for INT0
226     INTCONbits.GIE = 1; // Enable Global Interrupt
227 }
228
```

```
229 /**
230  * Interrupt function for dealing with external interrupt
231  */
232 void __interrupt() ISR(void) {
233     Reset(); // Reset the microcontroller, thus stopping motors
234 }
235
236 /**
237  * Write data on LCD
238  * @param type Type of data: 0 - command, 1 - char
239  * @param data Data to be sent to LCD
240  */
241 void LCD_write(unsigned char type, char data) {
242     LCD_RS_PIN = type; // Set data type
243
244     LCD_PORT = (LCD_PORT & 0x0f) | (0xf0 & data); // Higher part of data
245     // Send data to LCD
246     LCD_EN_PIN = 1;
247     NOP();
248     LCD_EN_PIN = 0;
249     __delay_ms(1);
250
251     LCD_PORT = (LCD_PORT & 0x0f) | (data << 4); // Lower part of data
252     // Send data to LCD
253     LCD_EN_PIN = 1;
254     NOP();
255     LCD_EN_PIN = 0;
256     __delay_ms(1);
257 }
258
259 /**
260  * Write data with command on LCD
261  * @param cmd Data with command
262  */
263 void LCD_write_cmd(char cmd) {
264     LCD_write(0, cmd); // Command
265 }
266
267 /**
268  * Write data with char on LCD
269  * @param c Data with char
270  */
271 void LCD_write_c(char c) {
272     LCD_write(1, c); // Char
273 }
274
275 /**
276  * Write data with string on LCD
277  * @param s Data with string
278  */
279 void LCD_write_s(const char *s) {
280     while ((*s) != 0) {
281         LCD_write_c(*s);
282         s++;
283     }
284 }
285
286 /**
```

```
287 * Run motor power sequence
288 */
289 void motors_run(void) {
290   unsigned char i = 0;
291
292   INTE_init(); // Enable external interrupt for START button
293
294   CACB_PIN = 1; // Turn on auto-transformer CA/CB pin
295
296   __delay_ms(AT_DELAY); // Auto-transformer power on delay
297
298   for (i = 0; i < n_motors; i++) {
299     // Apply waiting time for motor [i]
300     if (i != 0) // No waiting time for the first motor
301       delay_ms(w_time[i - 1]*1000);
302
303     // Turn on C[i]x
304     if (i == 0)
305       C1X_PIN = 1;
306     else if (i == 1)
307       C2X_PIN = 1;
308     else if (i == 2)
309       C3X_PIN = 1;
310     else if (i == 3)
311       C4X_PIN = 1;
312     else if (i == 4)
313       C5X_PIN = 1;
314     else if (i == 5)
315       C6X_PIN = 1;
316     else if (i == 6)
317       C7X_PIN = 1;
318     else if (i == 7)
319       C8X_PIN = 1;
320     else if (i == 8)
321       C9X_PIN = 1;
322     else
323       C10X_PIN = 1;
324
325     // Apply starting time for motor [i]
326     delay_ms(s_time[i]*1000);
327
328     // Turn off C[i]x
329     if (i == 0)
330       C1X_PIN = 0;
331     else if (i == 1)
332       C2X_PIN = 0;
333     else if (i == 2)
334       C3X_PIN = 0;
335     else if (i == 3)
336       C4X_PIN = 0;
337     else if (i == 4)
338       C5X_PIN = 0;
339     else if (i == 5)
340       C6X_PIN = 0;
341     else if (i == 6)
342       C7X_PIN = 0;
343     else if (i == 7)
344       C8X_PIN = 0;
```

```
345         else if (i == 8)
346             C9X_PIN = 0;
347         else
348             C10X_PIN = 0;
349
350         // Apply CX to C delay (to avoid short circuits)
351         __delay_ms(CX_C_DELAY);
352
353         // Turn on Cx
354         if (i == 0)
355             C1_PIN = 1;
356         else if (i == 1)
357             C2_PIN = 1;
358         else if (i == 2)
359             C3_PIN = 1;
360         else if (i == 3)
361             C4_PIN = 1;
362         else if (i == 4)
363             C5_PIN = 1;
364         else if (i == 5)
365             C6_PIN = 1;
366         else if (i == 6)
367             C7_PIN = 1;
368         else if (i == 7)
369             C8_PIN = 1;
370         else if (i == 8)
371             C9_PIN = 1;
372         else
373             C10_PIN = 1;
374     }
375 }
376
377 /**
378  * Stop motors
379  */
380 void motors_stop(void) {
381     CACB_PIN = 0;
382     C1X_PIN = 0;
383     C2X_PIN = 0;
384     C3X_PIN – 0;
385     C4X_PIN = 0;
386     C5X_PIN = 0;
387     C6X_PIN = 0;
388     C7X_PIN = 0;
389     C8X_PIN = 0;
390     C9X_PIN = 0;
391     C10X_PIN = 0;
392     C1_PIN = 0;
393     C2_PIN = 0;
394     C3_PIN = 0;
395     C4_PIN = 0;
396     C5_PIN = 0;
397     C6_PIN = 0;
398     C7_PIN = 0;
399     C8_PIN = 0;
400     C9_PIN = 0;
401     C10_PIN = 0;
402 }
```

```
403
404 /**
405  * Helper function used to clear and update the LCD
406  */
407 void menu_clear_update(void) {
408     LCD_write_cmd(LCD_CMD_CLEAR); // Clear LCD
409     menu_update_flag = 1; // Set menu update  flag
410 }
411
412 /**
413  * Helper function used to show error message
414  */
415 void menu_error(void) {
416     menu_clear_update(); // Clear menu and set menu update flag
417
418     // Line 1
419     LCD_write_cmd(LCD_CMD_LINE1);
420     LCD_write_s("Erro!");
421
422     __delay_ms(1000);
423
424     menu_clear_update(); // Clear menu and set menu update flag
425 }
426
427 /**
428  * Menu with boot information
429  */
430 void menu_boot(void) {
431     // Line 1
432     LCD_write_cmd(LCD_CMD_LINE1);
433     LCD_write_s("  ENG. ELETRICA");
434
435     // Line 2
436     LCD_write_cmd(LCD_CMD_LINE2);
437     LCD_write_s("      UNISC");
438
439     __delay_ms(1000);
440
441     menu = 1; // Go to menu_n_motors()
442     menu_motor_i = 0; // Reset motor index
443     menu_clear_update(); // Clear menu and set menu update flag
444 }
445
446 /**
447  * Menu for selecting the number of motors
448  */
449 void menu_n_motors(void) {
450     unsigned char d, u;
451
452     if (menu_update_flag) { // Avoid updating menu unnecessarily
453         // Line 1
454         LCD_write_cmd(LCD_CMD_LINE1);
455         LCD_write_s("Nro de motores:");
456         d = n_motors / 10;
457         u = n_motors - d * 10;
458
459         // Line 2
460         LCD_write_cmd(LCD_CMD_LINE2);
```

```
461         if (d)
462             LCD_write_c(d + '0');
463         LCD_write_c(u + '0');
464 
465         menu_update_flag = 0; // Menu updated
466     }
467 
468     // Button handling
469     if (!BTN_INC_PIN) {
470         __delay_ms(BTN_DELAY); // Software debounce
471         if (n_motors < 10) {
472             n_motors++; // Increase number of motors
473             menu_clear_update(); // Clear menu and set menu update flag
474         }
475     }
476 
477     if (!BTN_DEC_PIN) {
478         __delay_ms(BTN_DELAY); // Software debounce
479         if (n_motors > 1) {
480             n_motors--; // Decrease number of motors
481             menu_clear_update(); // Clear menu and set menu update flag
482         }
483     }
484 
485     if (!BTN_SET_PIN) {
486         __delay_ms(BTN_DELAY); // Software debounce
487         menu = 2; // Go to menu_s_time()
488         menu_motor_i = 0; // Reset motor index
489         menu_clear_update(); // Clear menu and set menu update flag
490     }
491 
492     if (!BTN_STRT_PIN) {
493         __delay_ms(BTN_DELAY); // Software debounce
494         menu_error();
495     }
496 }
497 
498 /**
499  * Menu for selecting starting time for each motor
500  */
501 void menu_s_time(void) {
502     unsigned char d, u;
503 
504     if (menu_update_flag) { // Avoid updating menu unnecessarily
505         // Line 1
506         LCD_write_cmd(LCD_CMD_LINE1);
507         LCD_write_s("Ta motor ");
508         d = (menu_motor_i + 1) / 10;
509         u = (menu_motor_i + 1) - d * 10;
510         if (d)
511             LCD_write_c(d + '0');
512         LCD_write_c(u + '0');
513         LCD_write_c(':');
514 
515         // Line 2
516         LCD_write_cmd(LCD_CMD_LINE2);
517         d = (s_time[menu_motor_i]) / 10;
518         u = (s_time[menu_motor_i]) - d * 10;
```

```
519         if (d)
520             LCD_write_c(d + '0');
521         LCD_write_c(u + '0');
522         LCD_write_s(" min");
523 
524         menu_update_flag = 0; // Menu updated
525     }
526 
527     // Button handling
528     if (!BTN_INC_PIN) {
529         __delay_ms(BTN_DELAY); // Software debounce
530         if (s_time[menu_motor_i] < 10) {
531             s_time[menu_motor_i]++; // Increase starting time for motor
532             menu_clear_update(); // Clear menu and set menu update flag
533         }
534     }
535 
536     if (!BTN_DEC_PIN) {
537         __delay_ms(BTN_DELAY); // Software debounce
538         if (s_time[menu_motor_i] > 1) {
539             s_time[menu_motor_i]--; // Decrease starting time for motor
540             menu_clear_update(); // Clear menu and set menu update flag
541         }
542     }
543 
544     if (!BTN_SET_PIN) {
545         __delay_ms(BTN_DELAY); // Software debounce
546         if ((menu_motor_i + 1) < n_motors) {
547             menu_motor_i++; // Go to next motor
548             menu_clear_update(); // Clear menu and set menu update flag
549         } else {
550             menu = 3; // Go to menu_w_time()
551             menu_motor_i = 0; // Reset motor index
552             menu_clear_update(); // Clear menu and set menu update flag
553             if (n_motors == 1) // If there is only 1 motor
554                 menu = 4; // Go to menu_ready()
555         }
556     }
557 
558     if (!BTN_STRT_PIN) {
559         __delay_ms(BTN_DELAY); // Software debounce
560         menu_error();
561     }
562 }
563 
564 /**
565  * Menu for selecting waiting time between motors
566  */
567 void menu_w_time(void) {
568     unsigned char d, u;
569 
570     if (menu_update_flag) { // Avoid updating menu unnecessarily
571         // Line 1
572         LCD_write_cmd(LCD_CMD_LINE1);
573         LCD_write_s("Te motor ");
574         d = (menu_motor_i + 1) / 10;
575         u = (menu_motor_i + 1) - d * 10;
576         if (d)
```

```
577             LCD_write_c(d + '0');
578         LCD_write_c(u + '0');
579         LCD_write_s(" e ");
580         d = (menu_motor_i + 2) / 10;
581         u = (menu_motor_i + 2) - d * 10;
582         if (d)
583             LCD_write_c(d + '0');
584         LCD_write_c(u + '0');
585         LCD_write_c(':');
586
587         // Line 2
588         LCD_write_cmd(LCD_CMD_LINE2);
589         d = (w_time[menu_motor_i]) / 10;
590         u = (w_time[menu_motor_i]) - d * 10;
591         if (d)
592             LCD_write_c(d + '0');
593         LCD_write_c(u + '0');
594         LCD_write_s(" min");
595
596         menu_update_flag = 0; // Menu updated
597     }
598
599     // Button handling
600     if (!BTN_INC_PIN) {
601         __delay_ms(BTN_DELAY); // Software debounce
602         if (w_time[menu_motor_i] < 9) {
603             w_time[menu_motor_i]++; // Increase waiting time for motor
604             menu_clear_update(); // Clear menu and set menu update flag
605         }
606     }
607
608     if (!BTN_DEC_PIN) {
609         __delay_ms(BTN_DELAY); // Software debounce
610         if (w_time[menu_motor_i] > 1) {
611             w_time[menu_motor_i]--; // Decrease waiting time for motor
612             menu_clear_update(); // Clear menu and set menu update flag
613         }
614     }
615
616     if (!BTN_SET_PIN) {
617         __delay_ms(BTN_DELAY); // Software debounce
618         if ((menu_motor_i + 1) < n_motors - 1) {
619             menu_motor_i++;
620             menu_clear_update(); // Clear menu and set menu update flag
621         } else {
622             menu = 4; // Go to menu_ready()
623             menu_motor_i = 0; // Reset motor index
624             menu_clear_update(); // Clear menu and set menu update flag
625         }
626     }
627
628     if (!BTN_STRT_PIN) {
629         __delay_ms(BTN_DELAY); // Software debounce
630         menu_error();
631     }
632 }
633
634 /**
```

```
635  * Menu shown while awaiting for start button
636  */
637 void menu_ready(void) {
638     if (menu_update_flag) { // Avoid updating menu unnecessarily
639         LCD_write_cmd(LCD_CMD_LINE1);
640         LCD_write_s("Pronto!");
641 
642         LCD_write_cmd(LCD_CMD_LINE2);
643         LCD_write_s("START p/ iniciar");
644 
645         menu_update_flag = 0; // Menu updated
646     }
647 
648 
649     // Button handling
650     if (!BTN_INC_PIN || !BTN_DEC_PIN || !BTN_SET_PIN) {
651         __delay_ms(BTN_DELAY); // Software debounce
652         menu_error();
653     }
654 
655     if (!BTN_STRT_PIN) {
656         menu = 5; // Go to menu_run()
657         menu_motor_i = 0; // Reset motor index
658         menu_clear_update(); // Clear menu and set menu update flag
659     }
660 }
661 
662 /**
663  * Menu shown while run sequence is executed
664  */
665 void menu_run(void) {
666     if (menu_update_flag) { // Avoid updating menu unnecessarily
667         // Line 1
668         LCD_write_cmd(LCD_CMD_LINE1);
669         LCD_write_s("Partindo...");
670 
671         // Line 2
672         LCD_write_cmd(LCD_CMD_LINE2);
673         LCD_write_s("START p/ parar");
674 
675         menu_update_flag = 0; // Menu updated
676     }
677 
678     motors_run(); // Run sequence
679 
680     menu = 6; // Go to menu_running()
681     menu_motor_i = 0; // Reset motor index
682     menu_clear_update(); // Clear menu and set menu update flag
683 }
684 
685 /**
686  * Menu shown after run sequence is executed
687  */
688 void menu_running(void) {
689     if (menu_update_flag) { // Avoid updating menu unnecessarily
690         // Line 1
691         LCD_write_cmd(LCD_CMD_LINE1);
692         LCD_write_s("Pronto");
```

```
693
694         // Line 2
695         LCD_write_cmd(LCD_CMD_LINE2);
696         LCD_write_s("START p/ parar");
697
698         menu_update_flag = 0; // Menu updated
699     }
700
701     // Button handling
702     if (!BTN_INC_PIN || !BTN_DEC_PIN || !BTN_SET_PIN) {
703         __delay_ms(BTN_DELAY); // Software debounce
704         menu_error();
705     }
706 }
```