CENTRO UNIVERSITÁRIO DE BRASÍLIA

FACULDADE DE TECNOLOGIA E CIÊNCIAS SOCIAIS APLICADAS

CURSO DE ENGENHARIA DA COMPUTAÇÃO

FILLIPE RIBEIRO GALIZA

AUTOMAÇÃO DE ROTINAS DE ALIMENTAÇÃO E ILUMINAÇÃO DE AQUÁRIOS DOMÉSTICOS

Orientadora: Profª. M.C. Maria Marony Sousa Farias

BRASÍLIA

2º SEMESTRE DE 2012

FILLIPE RIBEIRO GALIZA

AUTOMAÇÃO DE ROTINAS DE ALIMENTAÇÃO E ILUMINAÇÃO DE AQUÁRIOS DOMÉSTICOS

Trabalho apresentada ao UniCEUB – Centro Universitário de Brasília como pré-requisito para obtenção de Certificação de Conclusão do Curso de Engenharia de Computação. Orientadora: Profª. M.C. Maria Marony Sousa Farias Nascimento.

BRASÍLIA

2º SEMESTRE DE 2012

FILLIPE RIBEIRO GALIZA

AUTOMAÇÃO DE ROTINAS DE ALIMENTAÇÃO E ILUMINAÇÃO DE AQUÁRIOS DOMÉSTICOS

Trabalho apresentada ao UniCEUB – Centro Universitário de Brasília como pré-requisito para obtenção de Certificação de Conclusão do Curso de Engenharia de Computação. Orientadora: Profª. M.C. Maria Marony Sousa Farias Nascimento.

Este Trabalho foi julgado adequado para a obtenção do Título de Engenheiro de Computação, e aprovado em sua forma final pela Faculdade de Tecnologia e Ciências Sociais Aplicadas - FATECS.

_____

Prof. Abiezer Amarília Fernandez
Coordenador do Curso

Banca Examinadora:

_____

Profª. Maria Marony Sousa Farias, mestre em Engenharia Elétrica – UFPB – PB.
Orientadora

_____

Prof. Julimá Bezerra Junior, Mestre em Engenharia Elétrica – Instituto Militar de Engenharia Rio de Janeiro -RJ

_____

Profª. Irene de Azevedo Lima Joffily, Mestre em Estruturas e Construção Civil - UNB

_____

Prof. Flávio Antonio Klein, Mestre em Estatística - UNB

Dedico este trabalho a todos que me deram forças e suportaram comigo por todos esses anos a loucura que é se tornar um engenheiro.

AGRADECIMENTOS

A minha família, por toda dedicação que tiveram em me educar e fazer com que me tornasse a pessoa que sou hoje, me dando apoio por todos os momentos da minha vida. Agradecimento em especial ao meu pai, Milton da Costa Galiza Filho, por me ajudar diretamente na montagem do protótipo e por todo o apoio.

A minha namorada Gabrielle Velloso Portes, que por várias vezes me ajudou e me acompanhou noites adentro fazendo trabalhos, ouvindo reclamações, lamentos, suportou estresses e frustrações, mas que sempre esteve lá para me colocar de volta no caminho certo, me dando todo o apoio que eu precisei para que hoje pudesse chegar ao final deste caminho.

Aos amigos que dividiram o calor da batalha de todos os dias durante os longos cinco anos, enfrentando trabalhos, provas dificílimas, e que sem o apoio de todos como um grupo, nunca teríamos conseguido conquistar mais este desafio. E em especial aos amigos permitiram que este projeto fosse possível: Paulo Henrique, Jefferson Santos, Matheus Santana, José Carlos Cruz, Manuella Tereza, Vinicius Monteiro Bezerra dentre outros.

Aos professores que me passaram ensinamentos diversos não só sobre como a engenharia funciona e suas aplicabilidades, mas também preparações para a vida que teremos que seguir a partir de agora. Agradeço em especial à professora Maria Marony e o professor Francisco Javier por terem me orientado neste projeto final.

E a todos que participaram direta e indiretamente para que eu conseguisse concluir meu projeto.

“For long you’ll live and high you’ll fly
And smiles you’ll give and tears you’ll cry
And all you touch and all you see
Is all your life will ever be.”
PinkFloyd

RESUMO

Este trabalho apresenta um dispositivo de controle de alimentação e iluminação de aquários domésticos baseando-se nos intervalos necessários entre cada operação. O projeto integra um motor de passo, microcontrolador PIC16F877A, relé para acionar a iluminação, e um grupo de botões utilizados para a programação e personalização do sistema pelo usuário. Controlado através de linguagem de programação C, o dispositivo permite a configuração dos intervalos entre as rotinas de alimentação e iluminação utilizando-se do *timer* nativo do microcontrolador para realizar a contagem entre as operações e uma máquina de estados criada para a mudança entre os modos de configuração. A alimentação será feita a partir de um compartimento contendo o alimento sendo girado pelo motor de passo, e a iluminação acionada por meio de um relê, ambos ocorrendo ao estouro do *timer*. O protótipo é composto por uma lâmpada para a iluminação, uma caixa contendo a placa e um compartimento de alimentação que será ligado a um pequeno aquário para simular uma situação real.

Palavras chave: Aquário, microcontrolador, automação, motor de passo, timer, alimentação e iluminação.

ABSTRACT

This paper presents a power control device and lighting for home aquariums based on the required intervals between each operation. The project integrates a stepper motor, a PIC16F877A microcontroller, a relay to trigger lighting, and a group of buttons used for programming and user customization of the system. Controlled by programming language C, the device allows the configuration of the intervals between the feeding routines and illumination using native timer the microcontroller to perform the count between operations and a state machine designed for switching between configuration modes. The feeding is done from a compartment containing the food being rotated by the stepper motor and lighting systems activated by a relay, both occurring at the overflow of the timer. The prototype consists of a lamp for illumination, a box containing the card slot and a supply that is connected to a small tank to simulate a real situation.

Keywords: Aquarium, microcontroller, automation, stepper motor, timer, feeding and lighting.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

A/D	Analógico / Digital
CPU	Central Processing Unit
EEPROM	Erasable Electronically Programmable Read Only Memory
I/O	Input/Output
ICSP	In Circuit Serial Programming
LCD	Liquid Cristal Display
LED	Light-Emitting Diode
PH	Potencial Hidrogeniônico
PIC	Programmable Interface Controller
RAM	Random Access Memory
ROM	Read-Only Memory
SPI	Serial Peripheral Interface
USART	Universal Synchronous Asynchronous Receiver Transmitter

CAPÍTULO 1 - INTRODUÇÃO

1.1 Introdução ao Tema Proposto

Programar tarefas diárias rotineiras de uma forma automática é o novo conceito que está em ascensão no mercado de tecnologia mundial. A automação residencial é um ramo com grandes quantias de investimento, despertando a curiosidade de muitos e oferecendo soluções para os mais diversos tipos de questões.

Grande parte do espaço conquistado pela automação provém da facilidade fornecida. Tarefas rotineiras e cansativas se tornam simples operações e instruções quando controlados por máquinas e aparatos específicos, possibilitando para o seu usuário maior conforto e comodidade. Segurança doméstica, controle de iluminação, janelas automatizadas, e inclusive cuidados com seus animais de estimação, tema abordado neste trabalho, são algumas opções de serviços que podem ser automatizados.

As possibilidades da automação são inúmeras, logo, o crescimento desse tipo de tecnologia e o interesse para com o mesmo são inevitáveis. Assim, cada vez mais processos e serviços serão automatizados, desde empregados domésticos robotizados até maior conforto ao gerenciar aplicativos eletrônicos e aparelhos dentro de casa.

1.2 Motivação

Animais dos mais sensíveis para serem criados, peixes são altamente dependentes do habitat que criamos para eles. Mudanças na ambientação, no PH da água, nível de bactérias, oxigenação do aquário, iluminação, quantidade de alimento, são fatores que alteram a adaptação do peixe e devem ser constantemente verificados por seus donos. Atenção e cuidado são essenciais, pois descuidos podem ocasionar lesões, doenças e até a morte do animal.

Visando a todas essas pequenas rotinas que a criação de peixes requer, a automatização se torna óbvia, do ponto de vista da facilidade de não haver a

preocupação com fatores pequenos como esses, porém importantes, deixando o dono livre para simplesmente poder apreciar a beleza ornamentária do aquário.

O projeto abrange alguns desses detalhes, controle das rotinas de alimentação e iluminação, tornando possível a ausência do dono por um curto período de tempo sem que este tenha que se preocupar em estar arriscando a vida de seu animal de estimação.

1.3 Objetivos

O projeto tem como objetivo geral criar um ambiente automatizado através de um dispositivo para efetuar o controle da alimentação e iluminação de um aquário doméstico por meio da programação de seus intervalos. O usuário poderá programar por meio de um display, as configurações definindo os horários em que o peixe receberá a comida e iluminação. O compartimento de alimento será ajustável, permitindo maior controle da quantidade que será despejada no aquário, adequando-se para todos os tipos de ração.

O objetivo específico deste trabalho é apresentar um protótipo de um aquário automatizado, que, por meio de um microcontrolador, possa facilitar a interação do usuário com o aquário, retirando algumas rotinas repetitivas que os cuidados com o animal exigem. Um motor de passo cuidará da rotação do dispositivo de alimento, e um display tornará possível a interação máquina-usuário, possibilitando a programação dos intervalos entre o funcionamento do motor e dos comandos de acionar ou desligar a iluminação.

1.4 Estrutura da Monografia

Esta monografia está dividida entre sete capítulos, incluindo a INTRODUÇÃO, que tratará da introdução do tema proposto, motivações e objetivos do projeto, metodologias de elaboração e pesquisa. Esta seção descreverá a estrutura em que serão dispostos os capítulos, e uma breve descrição de seu conteúdo, assim como uma introdução sobre o tema abordado.

No segundo capítulo, APRESENTAÇÃO DO PROBLEMA, é contextualizada a problemática que o projeto se propõe a resolver. Descreve melhor as motivações, explica o problema a ser resolvido e mostra as soluções já presentes no mercado. Por fim, os benefícios que o projeto apresenta, sua proposta e escopo.

No terceiro capítulo, REFERENCIAL TEÓRICO, trata dos conhecimentos básicos necessários para um melhor entendimento conceitual do projeto. São discutidos assuntos como automação, microcontroladores e aquarismo. É apresentada também uma visão geral do projeto.

No quarto capítulo, DESCRIÇÃO DO HARDWARE E SOFTWARE, detalha as especificações dos recursos utilizados no projeto tanto na parte de *hardware* como de *software*. É observado o funcionamento de cada aparato, suas qualidades técnicas visando um entendimento completo de como o projeto será feito e a escolha de cada componente. É um capítulo complementar ao terceiro capítulo, precedendo a IMPLEMENTAÇÃO.

No quinto capítulo, IMPLEMENTAÇÃO, são apresentadas as etapas necessárias para compreensão geral da implementação do projeto.

No sexto capítulo, RESULTADOS OBTIDOS, é abordado o resultado dos experimentos, testes e implementação do projeto. As simulações feitas e as funcionalidades do projeto e seus resultados.

No sétimo capítulo, CONSIDERAÇÕES FINAIS, são apresentadas a conclusão e as sugestões para trabalhos futuros.

CAPÍTULO 2 – APRESENTAÇÃO DO PROBLEMA

Este capítulo tem como finalidade detalhar as motivações do trabalho e apresentar de forma mais contextualizada o problema a ser resolvido pelo projeto proposto. É parte complementar do tópico – Motivação – presente no capítulo anterior.

Serão exploradas as dificuldades do aquarismo, a comodidade que é gerada pela automação e soluções já presentes no mercado que são utilizadas para os problemas em questão. E por fim, é apresentada a proposta do projeto com seus benefícios e restrições.

2.1 Aquarismo e suas Dificuldades

> "É como se eles fossem um quadro vivo, uma peça decorativa do ambiente. Trabalhamos com esse conceito", explica o aquarista Yuri Frota. "Buscamos a qualidade de vida do animal, fazendo as pessoas entenderem que ele não está ali porque quer, mas nem por isso é um objeto — tem sentimentos, precisa curtir e descansar", completa André Bicalho, há 18 anos no comércio da aquariofilia. Com essa consciência, os aquaristas afirmam ser possível mudar a ideia de que criar peixes é chato.(CORREIOWEB, 2011, grifo meu).

O parágrafo acima mostra a realidade do aquarismo nos dias de hoje. É muito comum ouvirmos falar dos benefícios e vantagens de criar cachorros e gatos, mas a criação de peixes sempre foi deixada de lado. Grande parte dessa exclusão é pelo fato de ser um *hobby* que exige muitos cuidados de seu dono e uma atenção diária que não permite negligências.

A criação de peixes necessita inicialmente da construção de um ambiente artificial que simule o habitat do animal. Esse ambiente necessita de cuidados tanto quanto qualquer ser vivo presente nele, pois pequenas alterações no ecossistema podem desencadear doenças e até a morte do aquário como um todo. Os cuidados incluem a checagem de diversos itens para garantir a qualidade de vida do peixe como podemos observar abaixo:

- Alimentar os peixes 2 a 3 vezes por dia.
- Fornecer rações com mais de 40% de proteínas.
- Iluminar o aquário de 8 a 12 horas por dia.
- Não utilizar cascalho ou pedras com pontas ou cortantes, pois podem ferir os peixes
- Respeitar a temperatura da água conforme o tipo de peixe.
- Trocar sempre a alimentação dos peixes para evitar doenças nutricionais.
- Nunca bater no vidro
- Lavar as mãos e braços antes de mexer no aquário
- Trocar 1/4 da água por semana.
- Distribuir a iluminação uniformemente por todo tanque.
- Alimenta os peixes antes de desligar a luz, diminuindo o risco dos peixes maiores se alimentarem dos menores.

Fonte: (http://www.clickinformacao.com.br)

Dentre estes diversos itens, serão aprofundados os dois itens pertinentes ao projeto: Alimentação e Iluminação.

2.1.1 Alimentação

A alimentação é fundamental para a vida do peixe. Apesar de parecer simples, há diversos tipos de regras ao escolher, medir, e dar a ração ao animal. Mesmo depois de escolhida a ração ideal, é preciso manter uma regularidade quanto à quantidade e horário para que possa manter-se uma dieta balanceada.

Em aquários com vários tipos de peixe, a alimentação pode ser ainda mais crítica, pois uma vez que fiquem sem alimento, os peixes maiores veem outros peixes como presas, podendo até matá-los para se alimentar. Comida excessiva ou diminuta pode também causar prejuízos à saúde do animal. É possível perceber a

necessidade de um maior controle e periodicidade na rotina de alimentação, podendo sanar esses tipos de ocorrências. (aquarismobrasileiro.com.br).

2.1.2 Iluminação

> A iluminação é outro fator de grande importância para quem quer ter peixes. Atualmente as opções são muitas. A maior novidade, no entanto, são as luzes LED. Com programação específica, elas podem simular os ciclos naturais, tais como a intensidade dos raios solares, as fases da lua para reprodução, e até mesmo as condições climáticas. (CORREIOWEB, 2011, grifo meu).

É possível observar com base no texto acima, a importância que a iluminação tem em um aquário. Ela ajuda o peixe a se ambientar, sentir-se em seu habitat natural, ajudar na reprodução dentre outros fatores. Em aquários plantados vemos ainda mais nitidamente essa necessidade de um controle mais detalhado da iluminação. (WATSON, 2009)

Aquários plantados exigem um cuidado mais intenso. Peixes e algas devem coexistir dentro de um aquário. Elas realizam fotossíntese baseando-se na quantidade de luz, e oxigenam a água. Quando ocorre o excesso de luz, algas de aquário podem crescer desregulamente, prejudicando o habitat do peixe e a beleza ornamentaria do aquário. (WATSON, 2009)

2.2 Comodidade da Automação

A tecnologia trouxe diversos avanços na vida cotidiana. Pequenas doses de facilidade foram introduzidas ao dia a dia, facilitando aos poucos tarefas que seriam mais complexas ou mais trabalhosas sem o apoio de aparatos modernos. Detalhes quase imperceptíveis como o acender de uma lâmpada ou colocar roupas para lavar na máquina mostram o quão essencial à automação se tornou na vida do ser humano.

Desde a Revolução Industrial, a automação vem lentamente tomando espaço na rotina diária do homem. Começando nas linhas de montagem, onde a automação industrial se tornou essencial para a produção, percebe-se a grande van-

tagem ao mecanizar processos repetitivos. Desde então, sempre que possível, a aplicação desse tipo de comodidade vem se tornando mais comum. (GROOVER, 2000)

Com a popularização da tecnologia e maior acessibilidade às soluções mais sofisticadas para problemas simples, a automação deixa as indústrias para solucionar problemas caseiros. A Domótica é o termo que define a utilização dessa tecnologia para meios residenciais. Os exemplos são inúmeros, sendo alguns citados abaixo:

- Segurança automatizada (Sensores, Alarmes, Travas Eletrônicas, Câmeras de Segurança)
- Controle de iluminação de ambientes
- Controle de temperatura
- Cuidados com animais de estimação automatizados (Alimentação, Iluminação dentre outras necessidades)
- Jardim automatizado (irrigação e iluminação)
- Dentre outras.

Fonte: (engenium.net)

A busca por comodidade é algo sempre presente, e as facilidades que são proporcionadas por esses dispositivos tornam a vida em si mais agradável, diminuindo o stress e a preocupação de quem os possui. Isso impulsiona o crescimento do mercado de serviços de automação, visando garantir que essa tecnologia se expanda e possa abranger cada vez mais áreas.

A Figura 2.1 Ilustra alguns exemplos comuns de soluções de automação residencial.

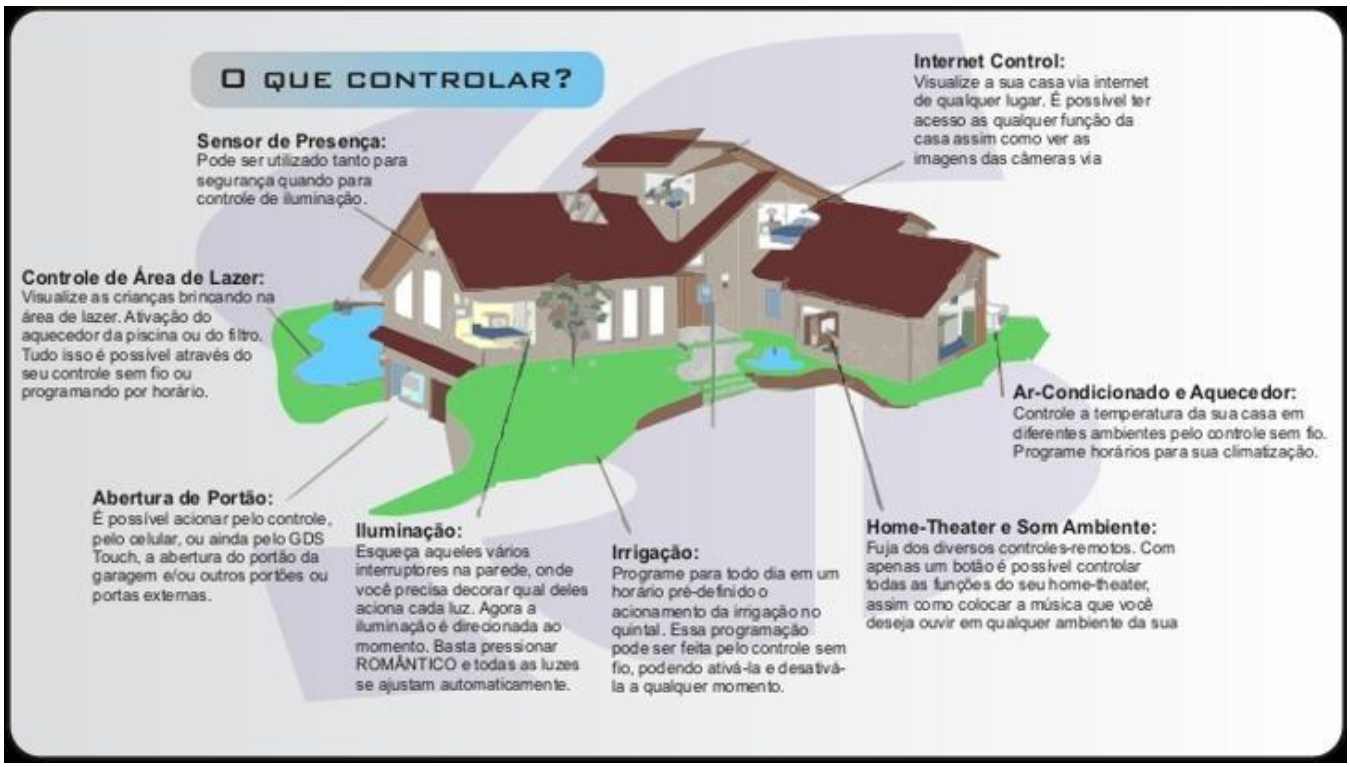

Figura 2.1 – Automação Residencial Exemplos
Fonte: gdsautomacao.com.br

2.3 Soluções Existentes

O aquarismo é uma das áreas que se beneficia da comodidade proporcionada pela automação. Por possuir diversas variáveis a serem controladas simultaneamente, ela é facilmente aplicada, pois retira preocupações que o usuário pudesse ter, evitando possíveis negligências ao se distrair de algum desses detalhes. Um aquário, mesmo pequeno, exige diversos cuidados como citados no tópico 2.1 deste capítulo, e a possibilidade de isenção da responsabilidade se torna agradável aos olhos de quem cultiva este *hobby*.

Existem no mercado aparelhos que facilitam a criação de peixes ornamentais e através dessas ferramentas diminuem a necessidade de manutenção tão periódica que um aquário exige. Desde aparelhos rudimentares como bombas de ar para oxigenar a água e filtros, até controles de temperatura, PH, alimentação e iluminação, são dos serviços hoje fornecidos por diversas empresas de automação doméstica.

2.4 Benefícios do Dispositivo Proposto e Suas Restrições

Esse projeto tem como finalidade apresentar um dispositivo de controle e automação de iluminação e alimentação de aquário, sendo utilizados equipamentos computacionais simples e de baixo custo.

Por se tratar de um dispositivo automatizado que será responsável por cuidar das rotinas diárias do aquário, o benefício principal seria a comodidade de poder viajar, se ausentar, ou mesmo não se preocupar com estes detalhes mais frequentes do aquarismo. Isso garante o bem estar do animal, por ser alimentado em horários precisos, e dá liberdade ao usuário.

O projeto se restringe às rotinas de alimentação e iluminação do aquário, não contemplando outros tipos de manutenção necessários para o bem estar do habitat do peixe como controle de PH, temperatura e etc. O alimentador deve ser preenchido com comida para que funcione, sendo a única preocupação do usuário, a vistoria do nível de comida existente no dispositivo. A iluminação ficará em cargo de uma lâmpada que será ativada ou desativada de acordo com a programação de intervalo.

A programação do projeto se restringe à escolha do intervalo de tempo entre as rotinas, sendo esse intervalo repetido até nova configuração ser feita. O projeto propõe a demonstração de suas funcionalidades por meio de um ambiente real de aquário onde o projeto deva automatizar suas funções.

CAPÍTULO 3 – REFERENCIAL TEÓRICO

Neste capítulo são abordados termos e conceitos teóricos necessários para um melhor entendimento do projeto e suas funcionalidades. Alguns desses temas, por serem extensos e visando o foco no projeto descrito neste trabalho, serão apresentadas somente as características e conceitos pertinentes.

Será apresentada também uma visão geral do projeto proposto. O entendimento deste capítulo é essencial para um melhor aproveitamento do restante do trabalho.

3.1 Aquarismo

> Não há dúvida de que os aquários de água doce e marinha portátil podem se tornar fonte de diversão e instrução infinitas, e ao mesmo tempo ser construídos de forma a ornamentar as salas em que são colocados. [...] A visão dos objetos em movimento, e das plantas aquáticas verdes sobre a superfície da água é sem dúvida confortante. [...] As travessuras de seus pequenos habitantes, e o pouco cuidado necessário para manter este mundo em miniatura em uma condição saudável, vai tirar a atenção de muitas horas de sofrimento e, inconscientemente, desenvolver um amor para as criaturas de Deus. Para as crianças, manter um aquário pode ser o meio de imperceptivelmente ensinar os sentimentos de humanidade para com os animais inferiores que até agora têm sido muito negligenciadas. (TAYLOR 1910, grifo e tradução Autor)

Desde antes de 1910, quando foi publicado o primeiro livro ocidental sobre o aquarismo, *The Aquarium It's Inhabitants, Structure and Management* (O Aquário, Seus Habitantes, Estrutura e Gerenciamento), já era possível perceber os benefícios que a atividade trazia. Não só como a criação de um animal de estimação, ou como ornamentação para diversos tipos de ambientes, mas a soma desses e outros fatores o tornaram um hobby muito procurado.

Na Figura 3.1 é mostrado um exemplo de aquário utilizado no século XVIII.

Figura 3.1: Aquário doméstico século XVIII.
Fonte: victoriana.com

O aquarismo, ou aquariofilia, é o ato de criar peixes, plantas e qualquer organismo aquático em aquários de vidro, tanques naturais ou artificiais, com o objetivo de ornamentação ou estudo dos mesmos. Foi de grande importância para a catalogação das espécies de organismos aquáticos no século XVI e um hobby que se integrou com os diversos tipos de criação animal em todo mundo.(BORGES JUNIOR, OLIVEIRA, 2009)

Para manter este tipo de atividade, é essencial ter conhecimento das necessidades e dificuldades de manutenibilidade de um aquário como comentado no tópico 2.1, e dos equipamentos que são utilizados para manter um ambiente saudável para o peixe. Abaixo temos alguns dos itens necessários para a montagem de um aquário simples de água doce:

- Aquecedor: Esse aparelho tem a função de aquecer a água quando há uma queda de temperatura, fazendo com que a água fique sempre aquecida, mantendo-a na mesma temperatura.
- Bombas de Circulação: Servem para fazer a movimentação da água internamente, ficando submersas e funcionando 24h por dia, melhorando a circulação de todo o aquário e consequentemente dando a estabilidade adequada.
- Baldes e Canecas: São necessários para transporte da água, medição de quantidade de água para reposição, coleta de água para testes e etc.
- *Chiller*: Ao contrário do aquecedor, este aparelho tem como função resfriar a água visando a estabilidade de temperatura do aquário
- Densímetro: Este aparelho tem a função de medir a densidade ou salinidade da água, que é o total da concentração de sólidos, gases e substâncias em suspenção dissolvidos na água.
- Filtro Biológico de Fundo: Consiste de um sistema de placas perfuradas e tubos ligados a uma bomba submersa ou pedra porosa, para que a água seja sugada através do cascalho de fundo.
- Redes: Utilizadas para capturar e movimentar peixes e outros animais do aquário
- *Timmers*: Tem a função de automatizar o ligamento e desligamento de alguns aparelhos do aquário.
- *Skimmers*: Fracionador de proteínas é um filtro que retira os poluentes da água, evitando a queda de qualidade da água.
- Termômetro: Funciona como medidor de temperatura do aquário

Fonte: (aqualandia.com.br)

A Figura 3.2 são mostrados exemplos de equipamentos básicos utilizados em um aquário doméstico

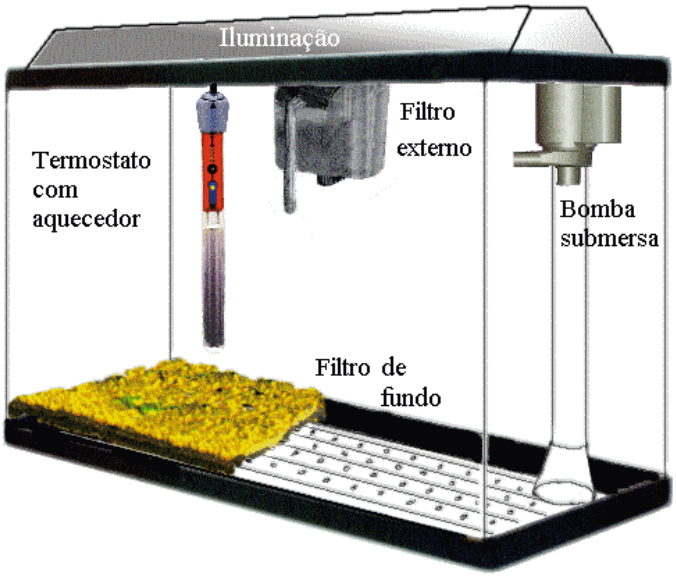

Figura 3.2 – Equipamentos de um aquário
Fonte: aquarioland2000.tripod.com

3.2 Automação

Segundo Moraes, Castrucci e Plinio (2001), a automação engloba qualquer sistema, apoiado em computadores, que substitua o trabalho humano e que vise a soluções rápidas e econômicas para atingir os complexos objetivos das indústrias e dos serviços (por exemplo, automação industrial, automação bancária).”.

A automação é o conceito de tornar atividades repetitivas em automáticas, utilizando equipamentos que efetuam coleta de dados e atuam nos processos, diminuindo a necessidade da interferência humana, resultando em maior velocidade nas operações, redução de erros e fidelidade de informações, que são elementos cruciais para um gerenciamento mais eficaz. É possível resolver problemas como menores custos de trabalho, menores perdas de materiais, maiores níveis de qualidade e controle das informações relativas ao processo, maior qualidade das informações e um melhor planejamento e controle.(MORAES, CASTRUCCI, PLINIO, 2001)

Por ser um conceito abrangente, a automação é aplicável em diversas áreas. Começou na indústria, resolvendo problemas de produtividade de fábricas e

linhas de montagem, até chegar à automação de rotinas cotidianas, bem próximas ao dia a dia moderno.

No Quadro 3.1, é feita uma comparação das vantagens trazidas pelo trabalho humano em comparação ao trabalho mecânico.

Quadro 3.1 – Trabalho Humano x Trabalho Maquinário

Trabalho Humano	Trabalho Maquinário
Perceber estímulos externos inesperados	Realizar tarefas repetitivas de modo consistente
Desenvolver novas soluções para problemas	Armazenar grandes quantidades de dados
Resolução de problemas abstratos	Resgatar dados da memoria de modo confiável
Adaptação à mudanças	Realizar várias tarefas simultaneamente
Generalizar por observações	Aplicar grandes forças
Aprender por experiência	Computar cálculos simples rapidamente
Realizar decisões complexas baseadas em dados incompletos	Realizar decisões rotineiras rapidamente

Fonte: (GROOVER, 2000)

3.2.1 Automação Residencial

> O objetivo da automação residencial é integrar iluminação, entretenimento, segurança, telecomunicações, aquecimento, ar condicionado e muito mais através de um sistema inteligente programável e centralizado. Como consequência fornece praticidade, segurança, conforto e economia para o dia a dia dos usuários. (ABREU, 2003).

A automação residencial é um ramo que está em constante crescimento juntamente ao avanço tecnológico e o aumento das possibilidades e diversas aplicações. Criada para aplicar os conceitos provenientes da automação industrial das fábricas num ambiente mais cotidiano, é responsável por grande parte da comodidade e conforto existentes no meio doméstico.(HARPER, 2003)

Soluções para segurança, entretenimento, eletrodomésticos, iluminação e ambientação são alguns dos exemplos mais comuns de aplicação da automação residencial como pode ser observado na Figura 3.3.

Figura 3.3 – Casa automatizada
Fonte: controlled.com.br

A princípio, por possuir poucas pessoas com o conhecimento para criar esse tipo de soluções, o custo de implantação da automação doméstica era elevado e de público restrito. Hoje em dia pode-se observar que é muito mais acessível e simples obter pequenos confortos automáticos dentro de casa

3.3 Microcontroladores

> O microcontrolador é um dispositivo semicondutor em forma de circuito integrado, que integra as partes básicas de um microcomputador microprocessador, memórias não-voláteis e voláteis e portas de entrada e saída. Geralmente, é limitado em termos de quantidade de memória, principalmente no que diz respeito à memória de dados, é utilizada em aplicações específicas, ou seja, naquelas que não necessitam armazenar grandes quantidades de dados, como automação residencial, automação predial, automação industrial e automação embarcada. (GIMENEZ, 2005, p. 4).

O microcontrolador é um circuito único integrado e programável, contemplando um CPU, memória de dados e programa, sistema de *clock*, portas, além de

outros periféricos como conversores A/D entre outros. Pode ser usado para controlar uma grande quantidade de aparelhos, coordenando suas funções e ações. (JUCA, 2010)

Na figura 3.4 é mostrado o microcontrolador utilizado no projeto.

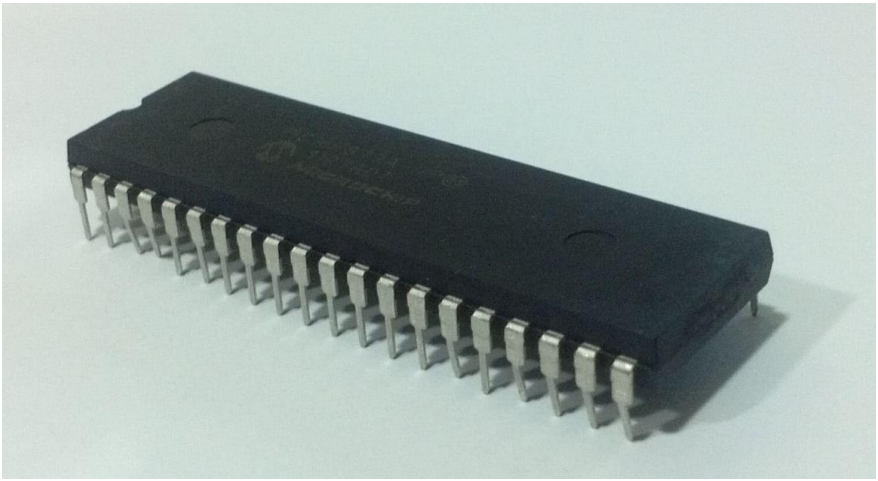

Figura 3.4 – Microcontrolador utilizado no projeto.
Fonte: Autor

Normalmente, são “embarcados” em algum outro aparelho, executando o controle de ações e respostas. O programa é registrado na memória ROM, fazendo com que o programa não mude depois de gravado no PIC, dedicando o microcontrolador à uma tarefa única, rodando o programa específico. Possui um dispositivo de entrada dedicado e normalmente um display de LCD ou LED para saída, facilitando a comunicação com o usuário e a sua aplicação no produto. (PEREIRA, 2003)

Além de receber entradas do usuário, o microprocessador também recebe informações do próprio produto no qual está embarcado, e controla o aparelho mandando sinais para diversos componentes do mesmo. É também, de certa forma, resistente, podendo ser aplicado em condições mais extremas onde computadores normais não conseguiriam. (GROOVER, 2001)

Na Figura 3.5 são ilustrados os componentes internos que compõe o funcionamento do microcontrolador.

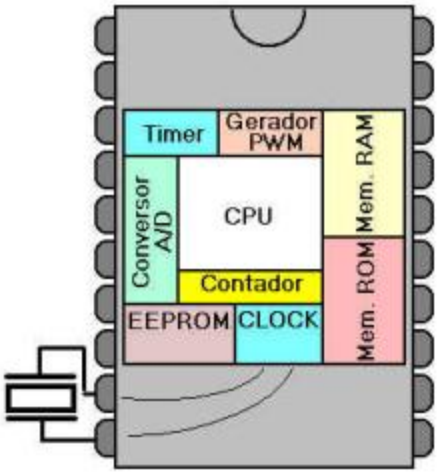

Figura 3.5 – Esquemático de um Microcontrolador
Fonte: eletronicagratis.blogspot.com.br

Um bom exemplo do funcionamento do microcontrolador seria observando um micro-ondas. Além de receber informações das teclas e exibi-las no *display*, controla o relé que liga ou desliga o funcionamento do aparelho. Pode-se observar a sua resiliência quando aplicado ao controle de motores de carros onde, dependendo da localização e clima, podem funcionar a temperaturas de -34ºC até 49ºC. Se for considerado o calor gerado pelo próprio motor, as temperaturas podem subir até 80ºC, sem que o controlador pare de funcionar ou perca suas capacidades. (GROOVER, 2001)

Podem ser fabricados por diversas marcas como *Microchip*, Intel, Atmel, Texas dentre outras. O modelo presente neste projeto, e discutido no tópico a seguir é o PIC pertencente à *Microchip Technology Inc*.

3.3.1 PIC

O PIC é um circuito integrado produzido pela *Microchip Technology Inc*., categorizado como um microcontrolador por possuir todos os circuitos necessários para o funcionamento de um sistema digital programável. Ele possui:

- CPU: Para interpretar as instruções de programa
- Memória EPROM: Para gravar permanentemente as instruções de programa

- Memória RAM: Para o armazenamento de variáveis utilizadas pelo código
- Barramentos de I/O: Para conexão de dispositivos externos, recepção de pulsos, sinais além de uma série de dispositivos auxiliares.

(JUCA, 2010)

Outra importante parte de um microcontrolador é seu *clock* ou circuito oscilador. Ele é responsável por ditar a velocidade em que as instruções serão executadas. Por estarem todos esses dispositivos num só circuito integrado, há uma maior facilidade e rapidez de aplicação poupando o custo e espaço que todos esses dispositivos trariam, vantagens de um sistema microcontrolado. (JUCA, 2010)

Os microcontroladores PIC se dividem em famílias de diversos modelos. Detalhamentos do PIC16F877A, modelo escolhido, estão presentes no capítulo 4.

3.4 Motor Elétrico

> Motor elétrico é a máquina destinada a transformar energia elétrica em energia mecânica. O motor de indução é o mais usado de todos os tipos de motores, pois combina as vantagens da utilização de energia elétrica - baixo custo, facilidade de transporte, limpeza e simplicidade de comando com sua construção simples, custo reduzido, grande versatilidade de adaptação às cargas dos mais diversos tipos e melhores rendimentos. O tipo mais comum de motor elétrico é o Motor de Indução. (www.coe.ufrj.br)

Na Figura 3.6 é mostrado um motor elétrico comum.

Figura 3.6 – Motor Elétrico
Fonte: dee.feb.unesp.br

Motores elétricos estão presentes em grande parte dos aparatos tecnológicos de hoje em dia. Eletrodomésticos, brinquedos, componentes para carros, bombas de aquário, computadores são exemplos de algumas de suas aplicações mais comuns. Como explicado no texto acima, transformam energia elétrica em mecânica e por sua versatilidade, é possível observar sua habilidade de se adaptar a várias utilizações diferentes. (coe.ufrj.br)

3.4.1 Motor de Passo

O motor de passo é um dispositivo eletro-mecânico que converte pulsos elétricos em movimentos angulares controlados chamados de “passos”. Cada passo consiste de pequenos incrementos na angulação do motor, motivo de sua precisão. É possível controlar exatamente o ponto que o motor deve ser acionado ou desligado, sua velocidade alterando a frequência de pulsos recebidos e a sequência de impulsos elétricos dita a direção de rotação do motor. (BRITES, SANTOS, 2008)

> O motor de passo é um motor com uma precisão muito grande do seu movimento. São utilizados onde é necessário o controle do número de rotações é muito importante, tais como em impressoras, drives de disquete e sistemas de automação industrial e robótica, pois, se não houvesse esse controle, o movimento contínuo poderia estragá-los. (MAXWELL BOHR, 2006).

Na figura 3.7 podemos observar um motor de passo comum de 4 fios.

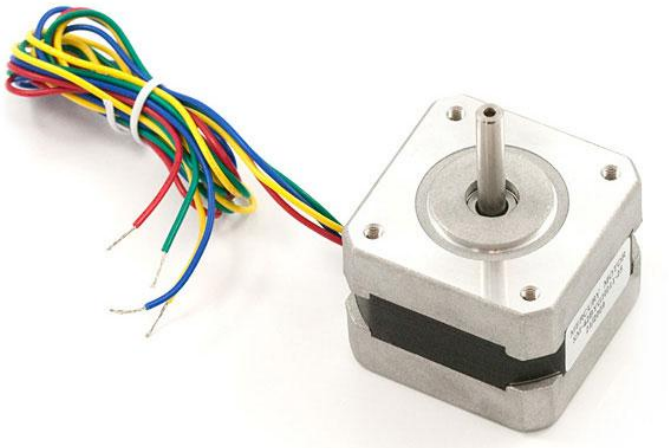

Figura 3.7 – Motor de passo
Fonte: multilogica-shop.com

É altamente empregado em automação, robótica dentre outras aplicações que requeiram o controle de variáveis como ângulo de rotação, velocidade, posição e sincronismo. Não é um motor de força ou velocidade, mas sua principal característica é a ampla gama de possibilidades possíveis com o seu funcionamento. (BRITES, SANTOS, 2008)

Serão mostradas suas especificações técnicas e o seu funcionamento detalhado no capítulo seguinte.

3.5 Linguagem de Programação

Uma linguagem de programação é um meio de expressar informações de modo compreensível tanto para o computador quanto para as pessoas. A sintaxe especifica como uma combinação de frases (podendo ser comandos, declarações, expressões e etc.) pode formar um programa. (HARPER 2012)

Este tipo de comunicação entre computadores e programadores é o que resulta em diversos sistemas com diferentes finalidades, sejam eles sistemas operacionais utilizados em computadores, ou mesmo programação de microcontroladores visando aplicações menores e mais pontuais.

No tópico a seguir será feita a descrição da linguagem de programação utilizada no projeto.

3.5.1 Linguagem C

Os primeiros dispositivos programáveis eram escritos em códigos chamados códigos de máquina, compostos por dígitos binários (combinações de “0” e “1”) inseridos por meio de um dispositivo de entrada de dados (teclado, leitora de cartões, fitas perfuradas ou discos magnéticos) para serem executados pela máquina. A programação Assembly foi criada para facilitar a programação, tornando-se mais acessível aos programadores, podendo executar operações como MOVLW 0X8C ao invés de longas linhas de código binário (PEREIRA, 2003).

Por ser uma linguagem de baixo nível, o Assembly não possui comandos, instruções ou funções além dos predefinidos no conjunto de instruções do processador utilizado. O programador então deve desenvolver rotinas e operações para contornar essas limitações impostas. Isto leva a criação de programas muito mais extensos e complexos com um fluxo muitas vezes difícil de ser seguido. (PEREIRA, 2003).

Visando a solução destes problemas, as linguagens de alto nível são criadas para permitir a programação utilizando comandos de alto nível que são posteriormente traduzidos para linguagem de baixo nível como *ASSEMBLY* ou diretamente para o próprio código de máquina do processador utilizado (PEREIRA, 2003).

Segundo Ritchie (1988, p. 8), C é uma linguagem de propósito geral que se baseia em economia de expressões, rica gama de operadores, e um controle moderno de curso de dados e estruturas. Sua falta de restrições, não sendo especializada em nenhuma área de aplicação específica, a torna muito mais conveniente e efetiva para diversas tarefas do que em comparação com linguagens mais poderosas.

A escolha de C para programação em microcontroladores se torna óbvia ao observar a grande velocidade na criação de novos projetos devido às facilidades de programação fornecidas pela linguagem, sua portabilidade que permite adaptar programas de um sistema para o outro com um mínimo de esforço e sua proximidade à linguagem de máquina, proporcionando extrema eficiência, o colocam como linguagem de alto nível mais eficiente atualmente disponível (PEREIRA, 2003).

3.6 Máquina de Estados Finitos

Uma máquina de estado é um modelo estrutural que representa o estado de algum símbolo ou conjunto de símbolos que podem receber entrada (*input*) para uma mudança que fará a alteração do estado atual para um novo estado, dentro de um determinado período de tempo. Um computador é um exemplo de máquina de

estado, mas também é possível descrever e modelar interações de sistemas. (DO-VICCHI, 2007).

Uma máquina de estado possui:

- Estado inicial descrito
- Conjunto de eventos de entrada
- Conjunto de eventos de saída
- Conjunto de estados

E utiliza funções que mapeiam:

- Estados e entradas para novos estados
- Estados para saídas

Uma máquina de estados finitos é aquela que possui um limite finito para seu número de estados. Ela introduz o conceito de estado como uma informação do seu histórico. Todos os estados serão uma representação de todas as situações em que a máquina possa se encontrar. Portanto, ela possui um tipo de “memória” do que ocorreu para que chegasse naquele estado presente. Enquanto a aplicação roda, o estado muda de tempo em tempo, e as entradas e saídas vão depender de qual estado a máquina se encontra. (WAGNER, 2006).

Podemos dividir a máquina de estados em dois tipos básicos de modelos que diferem no modo em que expressam sua saída (*output).* O modelo de Mealy baseia-se no estado atual assim como na entrada dos dados e o modelo de Moore será descrito a seguir.

3.7 Visão Geral do Projeto

O projeto proposto será constituído de um microcontrolador (PIC modelo 16F877A), um Motor de Passo, um relé de duas posições, um dispositivo de alimentação artesanal, um *display* de LCD, três botões para configurações e demais componentes eletrônicos, como resistores, capacitores, oscilador de cristal, dentre outros.

Um melhor detalhamento sobre o *hardware* e software que este projeto abrange e seu funcionamento será descrito no capítulo 4 – Descrição de *Hardware* e *Software*

CAPÍTULO 4 – DESCRIÇÃO DO HARDWARE E SOFTWARE

Este capítulo faz um detalhamento dos dispositivos utilizados neste projeto e seu funcionamento, abrangendo tanto a área de *hardware*, parte física, ou *software*, parte lógica.

4.1 Microcontrolador PIC 16F877A

O modelo de microcontrolador utilizado no projeto em questão é o 16F877A de 40 pinos, que possibilita a montagem de diversos dispositivos de entrada e saída. Neste capítulo, será feita a descrição detalhada de seu funcionamento e especificações técnicas.

4.1.1 Especificações

Segundo o seu *datasheet*, o PIC16F877A que será utilizado no projeto tem as seguintes especificações:

- 40 pinos
- 33 I/Os
- 8K x 14 bits de memória flash
- 256 bytes de memória EEPROM
- 368 bytes de memória RAM
- Dois módulos de Captura/Comparação/PWM (Controle por largura de pulso)
- 8 canais de A/D de 10 bits
- 2 comparadores
- 3 timers (2x8 e 1x16 bits)
- 1 USART

- Porta serial síncrona com SPI (*master mode*) e I²C (*master/slave*);
- Porta paralela com 8 bits de dados e sinais de controle externos (leitura e escrita);
- *Timer*/*Counter* programável e um *Watchdog Timer* embutidos, com seu próprio oscilador, para aplicações de Tempo Real críticas.
- 14 fontes de interrupção (internas e externas)

Sua grande quantidade de portas de entrada e saída facilitou a construção do projeto, tornando possível a conexão dos componentes necessários para o seu funcionamento. Seu *timer* e conversor AD embutidos foram primordiais na elaboração deste dispositivo.

4.1.2 Pinagem do PIC 16F877A

Na Figura 4.1 é mostrada a pinagem do PIC16F877A e o Quadro 4.1 o significado das nomenclaturas utilizadas na identificação desses pinos.

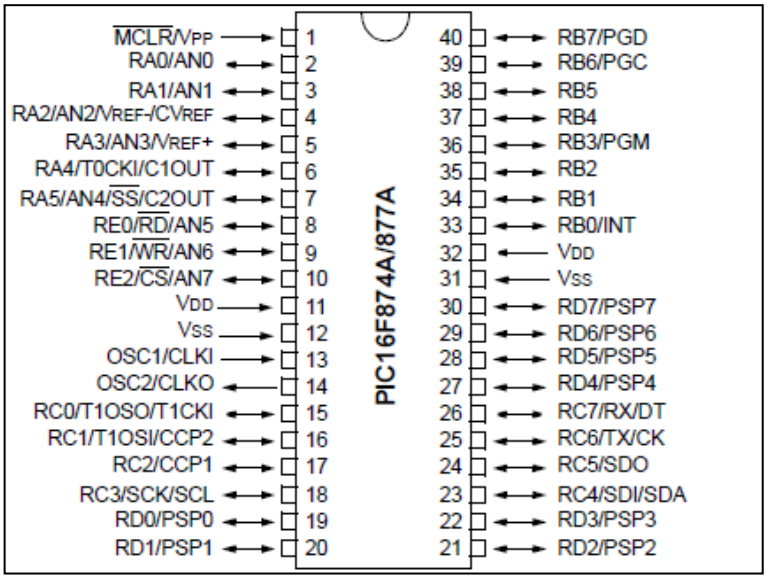

Figura 4.1 – Pinagem do PIC 16F877A
Fonte: sabereletronica.com.br

Quadro 4.1 –Pinos do PIC16F877A.

N°	Pino	Descrição
1	MCLR/Vpp	Master Clear (*reset*). O microcontrolador funciona quando este pino está em nível alto.
2	RA0 AN0	Entrada e saída digital. Entrada analógica.
3	RA1 AN1	Entrada e saída digital. Entrada analógica.
4	RA2 AN2 V_{REF-}/CV_{REF}	Entrada e saída digital. Entrada analógica. Tensão negativa de referência analógica.
5	RA3 AN3 V_{REF+}	Entrada e saída digital. Entrada analógica. Tensão positiva de referência analógica.
6	RA4 T0CKI C1OUT	Entrada e saída digital. *Open-drain* quando configurado como saída. Entrada externa do contador TMR0. Saída do comparador 1.
7	RA5 AN4 SS C2OUT	Entrada e saída digital. Entrada analógica. *Slave* para a comunicação SPI. Saída do comparador 2.
8	RE0 RD AN5	Entrada e saída digital. Controle de leitura da comunicação paralela. Entrada analógica.
9	RE1 WR AN6	Entrada e saída digital Controle de escrita da comunicação paralela. Entrada analógica.
10	RE2 CS AN7	Entrada e saída digital. Habilitação externa para comunicação paralela. Entrada analógica.
11/32	VDD	Alimentação positiva.
12/31	VSS	GND.
13	OSC1/CLKIN	Oscilador cristal ou entrada de osciladores externos.
14	OSC2/CLKOUT	Saída para oscilador cristal.
15	RC0 T1OSO T1CKI	Entrada e saída digital. Saída do oscilador externo para TMR1. Entrada de incremento para TMR1.
16	RC1 T1OSI CCP2	Entrada e saída digital. Entrada do oscilador externo para TMR1. Entrada do Capture2 ou Saída para Compare2/PWM2.
17	RC2 CCP1	Entrada e saída digital. Entrada do Capture1 ou Saída para Compare1/PWM1.
18	RC3 SCK SCL	Entrada e saída digital. Entrada/Saída do *clock* para comunicação SPI. Entrada/Saída do *clock* para comunicação I2C.
19	RD0 PSP0	Entrada e saída digital. Comunicação paralela.
20	RD1 PSP1	Entrada e saída digital. Comunicação paralela.
21	RD2 PSP2	Entrada e saída digital. Comunicação paralela.
22	RD3 PSP3	Entrada e saída digital. Comunicação paralela.
23	RC4 SDI DAS	Entrada e saída digital. Entrada de dados para comunicação SPI. Entrada/Saída de dados para comunicação I2C.
24	RC5	Entrada e saída digital.

	SDO	Saída de dados para comunicação SPI.
25	RC6 TX CK	Entrada e saída digital. Transmissão para comunicação assíncrona USART. *Clock* para comunicação síncrona USART.
26	RC7 RX DT	Entrada e saída digital. Recepção para comunicação assíncrona USART. Dados para comunicação síncrona USART.
27	RD4 PSP4	Entrada e saída digital. Comunicação paralela.
28	RD5 PSP5	Entrada e saída digital. Comunicação paralela.
29	RD6 PSP6	Entrada e saída digital. Comunicação paralela.
30	RD7 PSP7	Entrada e saída digital. Comunicação paralela.
33	RB0 INT	Entrada e saída digital. Interrupção externa.
34	RB1	Entrada e saída digital.
35	RB2	Entrada e saída digital.
36	RB3 PGM	Entrada e saída digital. Entrada para programação de baixa tensão.
37	RB4	Entrada e saída digital.
38	RB5	Entrada e saída digital.
39	RB6 PGC	Entrada e saída digital. *Clock* de programação serial ou pino de *in-circuit debugger*.
40	RB7 PGD	Entrada e saída digital. Dado de programação serial ou pino de *in-circuit debugger*.

Fonte: (SOUZA, 2005)

A pinagem escolhida no projeto pode ser observada na Figura 4.2. Essa configuração atende às necessidades de cada componente utilizado no projeto

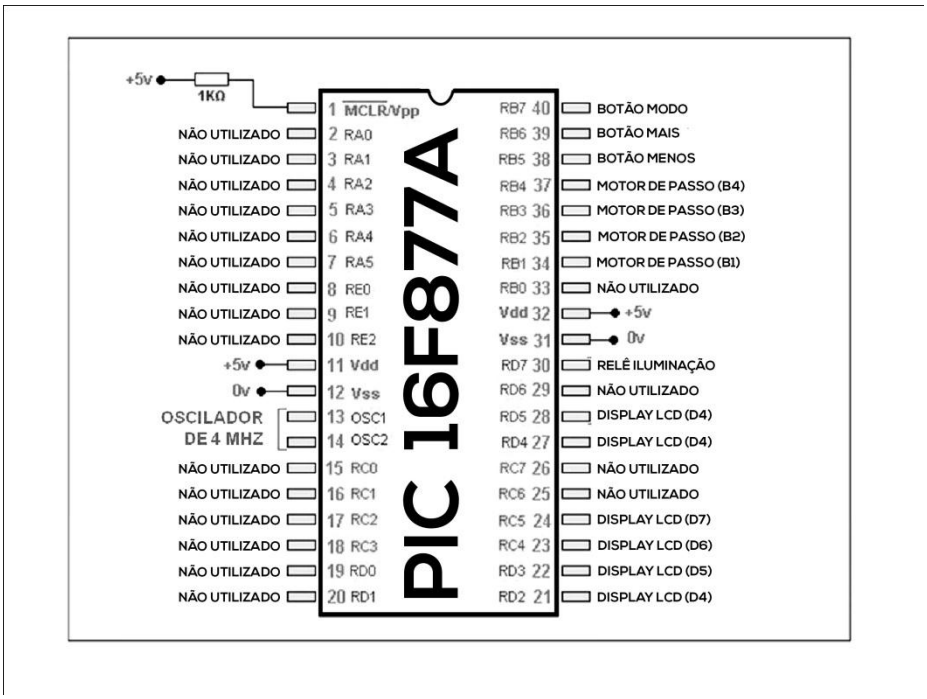

Figura 4.2 – Pinagem utilizada no projeto
Fonte: Autor

4.2 PIC C Compiler

Este *software* foi utilizado para a programação e compilação do código de programação do microcontrolador PIC. Seu trabalho é oferecer um ambiente amigável para a redação do código fonte, contendo bibliotecas específicas de eletrônica, facilitando o processo de construção do mesmo. Detalhamentos sobre o código escrito são feitos no capítulo seguinte. (ccsinfo.com)

O PIC C *Compiler* tem a função de auxiliar na criação e compilação do código em C para definir o funcionamento do microcontrolador. O *software* além da compilação faz a correção de erros de sintática e semântica do código, e previsão do uso da memória RAM e ROM do microcontrolador tornando mais intuitiva a programação, correção e teste antes da compilação e fornecendo uma visão geral de quanto o código utilizará dos recursos disponíveis pelo PIC. Ao compilar, são gerados arquivos de extensão ‘c’, ‘cof’, ‘err’, ‘hex’, ‘lst’, ‘pjt’, ‘sta’, ‘sym’, ‘tre’. O arquivo 'HEX' gerado é utilizado pelo PROTEUS para testes no circuito simulado, e para a gravação do programa no dispositivo. (ccsinfo.com)

4.3 Kit de Gravação PICkit 2

O kit de gravação PICkit 2 é uma ferramenta de desenvolvimento que permite a comunicação direta entre o computador e o PIC. Com ele, é possível gravar e apagar programas da memória do microcontrolador, sendo uma ferramenta indispensável. Ele é composto do conector ICSP para gravação em microcontroladores PIC de 8, 14, 18, 20, 28 e 40 pinos. (microchip.com)

O kit contempla o gravador, o cabo USB, conector ICSP e um CD que possui o *software* que fará o reconhecimento do microcontrolador.

4.3.1 Utilização do Kit de Gravação PICkit 2

Após compilar o programa com o *CCS C Compiler*, será utilizado o arquivo de extensão “HEX” com o kit de gravação, e gravá-lo no PIC16F877A. Além da gravação, o programa também permite apagar a memória do PIC para que sejam feitas alterações no programa ou para que seja inserido um programa diferente.

O programa mostra a mensagem “*Programming Successfull*” assim que termina a gravação, e emite um sinal sonoro indicando seu término.

4.4 Módulo LCD

O módulo LCD escolhido para o projeto foi o LCD 16X2, ou seja dezesseis caracteres divididos em duas linhas. Ele foi escolhido dentre os diversos modelos por possuir um número de caracteres suficientes para exibir as informações necessárias para a configuração do automatizador. A Figura 4.3 ilustra o módulo LCD utilizado no projeto.

Figura 4.3 – LCD utilizado no projeto
Fonte: Autor

Como se pode observar o módulo possui 16 pinos, o Quadro 4.2 apresenta a descrição dos pinos do módulo LCD e o interfaceamento no projeto.

Quadro 4.1– Pinos do LCD

Pino	Símbolo	Função	Interfaceamento no Projeto
16	K	Cátodo	NÃO UTILIZADO
15	A	Anodo	NÃO UTILIZADO
1	VSS	Terra	Aterrado
2	VDD	5 V	Fonte de Alimentação
3	V_O	Ajuste de contraste	Potenciômetro
4	R/S	Seleção de registro	Pino 27 do PIC16F877A
5	R/W	Leitura / Escrita	Aterrado
6	E	Inicia ciclo Leitura / Escrita	Pino 28 do PIC16F877A
7	DB0	Dado	NÃO UTILIZADO
8	DB1	Dado	NÃO UTILIZADO
9	DB2	Dado	NÃO UTILIZADO
10	DB3	Dado	NÃO UTILIZADO

11	DB4	Dado	Pino 21 do PIC16F877A
12	DB5	Dado	Pino 22 do PIC16F877A
13	DB6	Dado	Pino 23 do PIC16F877A
14	DB7	Dado	Pino 24 do PIC16F877A

FONTE: (ZANCO, 2010), Adaptada

4.5 Motor de Passo

O motor de passo visto no capítulo anterior é um dispositivo eletromecânico que converte pulsos elétricos em movimentos angulares controlados chamados de “passos”. O modelo utilizado no projeto é de 6 fios bipolar, que tem a função de rotacionar o dispositivo que conterá a ração que alimentará o aquário.

4.5.1 Funcionamento

O movimento básico é dado pelo uso de solenoides alinhados que quando recebem carga, se polarizam atraindo o rotor e fazendo-o se alinhar em seu eixo variando sua angulação, o chamado “passo”. O número de passos vai ser definido pela quantidade de alinhamentos possíveis entre o rotor e as bobinas, sendo que quanto mais bobinas, mais polos no rotor e consequentemente mais passos.(BRITES, SANTOS, 2008)

Se energizados uma bobina por vez, cada uma desloca o rotor levemente em seu eixo. As bobinas criam um campo magnético quando energizadas e juntamente com o magnetismo do rotor, forçam o alinhamento entre os dois, causando o deslocamento. Com a polarização adequada das bobinas, pode-se movimentar o rotor entre elas (meio passo ou “*half-step*”) ou alinha-lo com as mesmas (passo completo ou “*full-step*”). Seu funcionamento é bem simples, quando uma corrente circula pela bobina, esta cria um campo magnético que atrai o contato fechando ou abrindo circuito. Ao cessar a corrente da bobina o campo magnético também cessa e o contato volta a sua posição original. (BRITES, SANTOS, 2008)

4.6 Relê

Os relés são dispositivos eletromecânicos que controlam circuitos por meio de pequenas correntes ou tensões, funcionando como um interruptor

liga/desliga, fechando ou abrindo conexões. Ao receber corrente, a bobina cria um campo magnético que atrai o contato fechando ou abrindo o circuito e alternando de volta para o estado inicial quando esse campo é cessado.

4.5.1 Funcionamento

São amplamente utilizados em automação predial, sistemas de geração, transmissão e distribuição de energia elétrica e em máquinas e equipamentos em geral. São compostos por um eletroímã em forma de bobina; uma armadura metálica, que possa ser atraída pelo campo magnético criado pelo eletroímã; uma mola e um conjunto de contatos elétricos que serão abertos e fechados ou comutados conforme a configuração e o sinal recebido pelo circuito. (osetoreletrico.com.br)

Quando a corrente elétrica percorre a bobina e gera o campo magnético, a armadura é atraída por essa força e alterna a posição dos contatos dependendo da posição inicial do relé. Se a corrente é interrompida, o campo se anula e a mola faz com que o s contatos voltem à posição inicial. (osetoreletrico.com.br)

CAPÍTULO 5 – IMPLEMENTAÇÃO

Este é o capítulo principal do projeto, pois descreve como o mesmo foi feito, suas etapas para que fosse concluído; foi dividido em cinco tópicos para facilitar o entendimento:

- Modelagem do sistema;
- Elaboração dos circuitos;
- Elaboração do código fonte para o microcontrolador PIC;
- Montagem dos circuitos nas placas;
- Montagem do protótipo;

5.1 Modelagem do Sistema

A primeira etapa na implementação do projeto, foi a realização de um esboço do sistema definitivo, ou seja, a disposição em que se encontrariam os componentes em um ambiente real e que pudesse ser escolhida a melhor forma para dispor os componentes visando maior conforto ao usuário e garantindo melhor desempenho do projeto.

Para isso, foi pensado em como o resultado final seria acoplado a um aquário doméstico e como seria seu funcionamento. Esta etapa auxiliou na visualização do projeto como um todo, e fazer as alterações necessárias na ideia inicial para que pudesse se adequar melhor às necessidades do usuário. O projeto consiste de uma caixa de madeira contendo a placa do circuito, o compartimento de alimentação onde será acoplado o motor de passo e posicionado no aquário por meio de um encaixe na borda do mesmo e uma lâmpada que também será acoplada na borda do aquário, para melhor iluminação do ambiente.

Para montagem do protótipo, foi utilizado a maior quantidade de material reutilizável e reciclável possível, visando um melhor aproveitamento dos recursos já obtidos. A caixa de madeira que comporta a placa é de aproximadamente 10 por 20 centimetros; o dispositivo de alimentação para a demonstração foi feito com um tubo

de mini Cds vazio e a iluminação foi utilizado um iluminador próprio de aquário adaptado.

5.1.1 – Fluxograma Geral do Sistema

Após definição da disposição dos componentes, foi elaborado o fluxograma geral do sistema, conforme mostra a Figura 5.1. Este fluxograma foi fundamental na elaboração do código fonte, facilitando a observação de como seria a interação do usuário com o dispositivo.

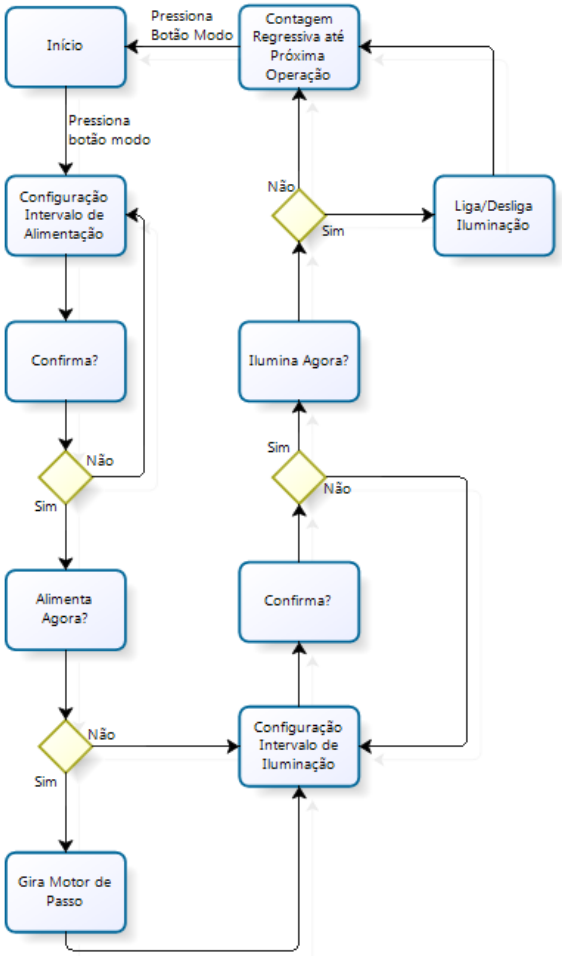

Figura 5.1 – Fluxograma do sistema
Fonte: Autor

5.2 Elaboração dos Circuitos

Após definido o funcionamento do sistema, como mostrado no fluxograma do tópico acima, o próximo passo foi a elaboração dos circuitos. Para isso, foi utilizado o programa Proteus ISIS 7 Profissional, qual, neste primeiro momento, foi possível definir quais seriam os componentes utilizados, e as quais pinos do microcontrolador PIC eles seriam interligados.

5.2.1 Proteus ISIS 7 Professional

O Proteus ISIS é um simulador de circuitos eletrônicos, que torna possível a simulação em tempo real do código programado junto ao *hardware* esquematizado com a ferramenta. O programa possui um banco de dados com uma série de modelos de componentes para serem adicionados como microcontroladores, resistores, displays, dentre outros, facilitando na hora da implementação e criação de um protótipo do projeto físico com complexidades diversas.

A primeira etapa para a criação da placa é realizar um estudo para decidir quais materiais serão necessários para que seja possível o funcionamento da placa como desejado. Para isso, é utilizado o *software* PROTEUS ISIS, onde é possível escolher componentes incluídos em uma grande biblioteca do programa, sendo possível fazer a busca pelo tipo de componente, material, e até fabricante.

A biblioteca de componentes do *software* pode ser observada na Figura 5.2

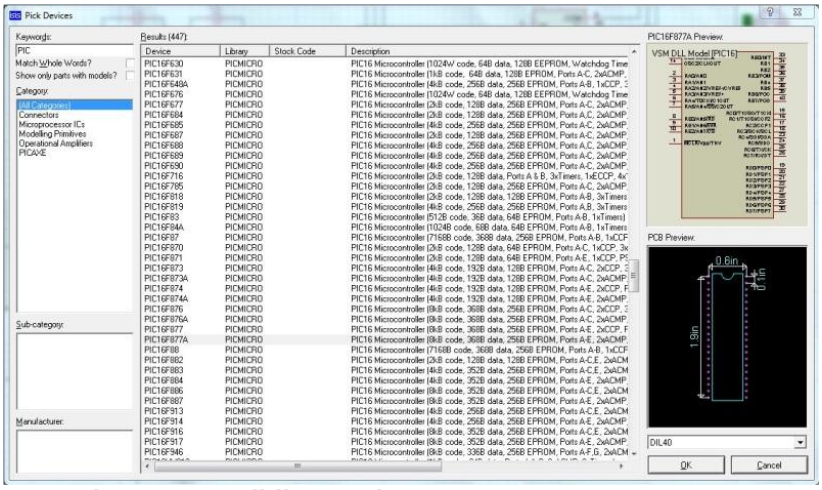

Figura 5.2 – Biblioteca de componentes ISIS
Fonte: Autor

Foi selecionado o microcontrolador PIC16F877A, um grupo de botões para que fosse possível fazer a configuração do sistema, um *display* LCD para retornar informações para o usuário, um motor de passo que girasse o dispositivo de alimentação e um LED de cor azul para que fosse feita a simulação do funcionamento do relé. Para que a criação do esquemático da placa seja feita, é necessário pensar também em todos os resistores, capacitores, cristal, fontes de energia e aterramentos que serão essenciais para o funcionamento correto do circuito.

Na Figura 5.3 é mostrado o esquema do circuito, utilizado como base para criação da placa.

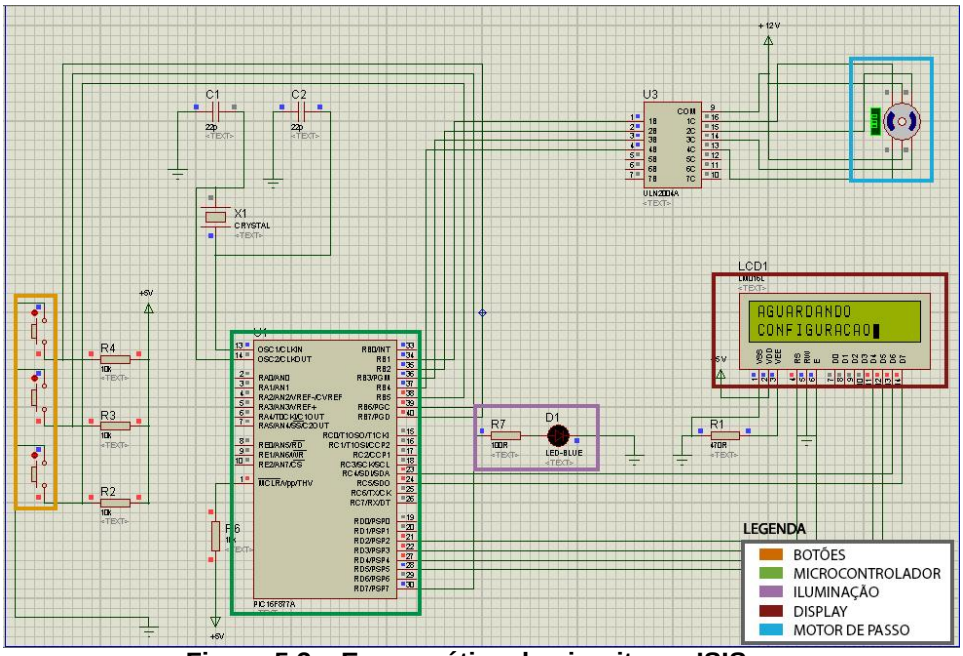

Figura 5.3 – Esquemática do circuito no ISIS
Fonte: Autor

Após a criação do modelo do circuito, o programa permite a gravação simulada do código fonte no microcontrolador PIC para que sejam feitos testes de funcionalidade do sistema. É necessário o arquivo “HEX” gerado *pelo CCS C Compiller* ao compilar o código fonte para que seja possível “abri-lo” com o microcontrolador. Depois de carregado o código, o circuito já poderá receber as instruções exatas e operar de acordo com o programado, possibilitando os testes.

A Figura 5.4 ilustra a tela de gravação do PIC, utilizada para realizar os testes no ambiente virtual.

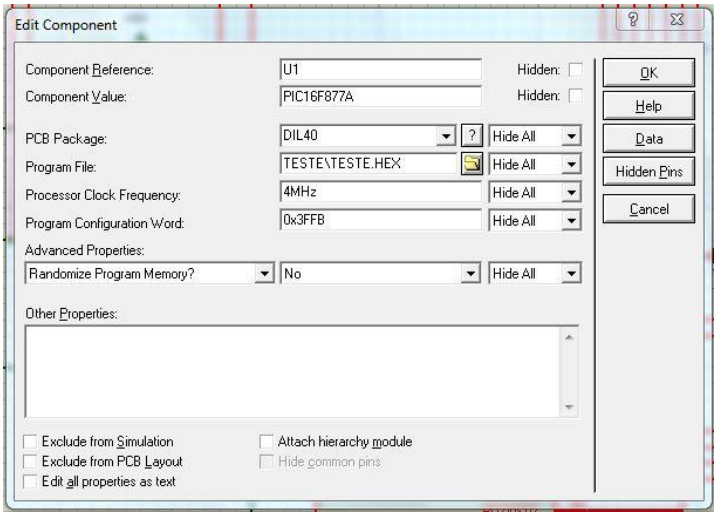

Figura 5.4 – Gravação do PIC ISIS
Fonte: Autor

Os testes são essenciais para observar quais componentes estão corretamente conectados, se o funcionamento deles corresponde ao desejado. No projeto, a escolha das portas de conexão no PIC foi feita com base na proximidade e facilidade de ligação, salvo a conexão dos botões que foi feita nas portas RB5, RB6 e RB7 por serem portas que permitem interrupção essencial para o funcionamento do projeto.

Mesmo o Proteus ISIS possibilitando testes em ambiente simulado, é aconselhado que testes sejam refeitos em ambiente físico para que seja possível garantir o nível de precisão desejado. Com isso, foi necessário a elaboração dos circuitos em um sistema físico para realização de novos testes

Após serem feitos os testes e decididos os componentes definitivamente, é possível passar os dados contidos no PROTEUS ISIS para que seja feito o desenho da placa utilizando o PROTEUS ARES.

5.3 Elaboração do Código Fonte para o Microcontrolador PIC

Com a modelagem da ideia do sistema e a elaboração do circuito, a próxima etapa é a elaboração do código fonte e testes utilizando o PROTEUS ISIS e gravação no microcontrolador. Para escrita das linhas de comando e compilação do código foi utilizado o programa PIC C *Compiler* e para gravação do código a ferra-

menta utilizada foi o kit de gravação PICkit 2 da Robótica Simples, que é composto por dispositivo e programa.

5.3.1 Escrita do Código Fonte

O código fonte foi escrito utilizando a linguagem C, por sua facilidade e alto nível de programação, vantagens sobre a programação em ASSEMBLY. A linguagem foi escolhida também pela grande biblioteca de exemplos e informações disponíveis em livros e *sites* pela *internet*.

> "... a utilização de uma linguagem de alto nível como C permite que o programador preocupe-se mais com a programação da aplicação em si, já que o compilador assume para si tarefas como o controle e localização das variáveis, verificação de bancos de memória, etc." (PEREIRA, 2007, p.18).

Tomando como base o fluxograma geral do sistema, e o modelo do circuito feito no PROTEUS ISIS contendo os componentes necessários para o funcionamento do projeto, o primeiro passo é definir as diretivas de pré-compilação, definições estas que são de extrema importância para a correta compilação do código.

Na Figura 5.5 são apresentadas as diretivas utilizadas na programação do código fonte do projeto e a frente de cada linha de comando, após duas barras conjuntas, um comentário explicando sua funcionalidade.

```
/*============================================================================
                          DIRETIVAS DE PRÉ-COMPILAÇÃO
============================================================================*/
#include <16f877A.h>                  //Diretivas do PIC16F877A
#use delay(clock=4000000)             //Cristal oscilador 4Mhz.
#fuses hs, nowdt, put                 //Fusíveis
#fuses brownout, nolvp, noprotect
#fuses nocpd, nodebug
#use rs232(baud=9600, xmit=PIN_C6)    //Transmissão a 9600 bps
                                      //Sem paridade, 8 bits de dados
                                      //Pino TX RC6.

#use fast_io(b)                       //Dá controle do I/O ao programador
#byte portb=0x06                      //Endereça o portb
#bit btnModo=portb.7                  //Define o botão modo no port B7
#bit btnMais=portb.6                  //Define o botão mais no port B6
#bit btnMenos=portb.5                 //Define o botão menos no port B5
#define MOT1          PIN_B1          //Define MOT1 como o pino B1 do motor
#define MOT2          PIN_B2          //Define MOT2 como o pino B2 do motor
#define MOT3          PIN_B3          //Define MOT3 como o pino B2 do motor
#define MOT4          PIN_B4          //Define MOT4 como o pino B2 do motor
#define LAMP          PIN_D7          //Define LAMP como o pino D7 do Relê
#define INI_TMR0      131             //Define o TMR0 com valor incial 131
#include <lcd-v2.0.h>                 //Biblioteca p/ o módulo LCD
```

Figura 5.5 – Diretivas Pré-Compilação
Fonte: Autor

O Quadro 5.1 apresenta a descrição de cada comando utilizado.

Quadro 5.1– Diretivas do código fonte.

Comando	Descrição
#include	Insere um arquivo texto externo a partir da posição atual.
#use delay	Informa ao compilador a velocidade de *clock* do sistema de destino.
#use rs232	Ativa o suporte à comunicação serial. Especificam: velocidade, pinos TX e RX.
#fuses	Programa as opções da palavra de configuração (*configuration word*) do PIC.
#device	Define o nome do processador utilizado.
#define	Substitui o identificador pelo texto especificado imediatamente depois dele.

FONTE: PEREIRA, 2007.

Após a definição de todas as diretivas necessárias para que o programa possa ser compilado, são definidas as variáveis utilizadas pelas funções. Para uma melhor manipulação dos eventos e possibilidades do programa, foram criadas variáveis separadas para a Alimentação e Iluminação do aquário, assim como diferenciação entre hora, dezena dos minutos e unidade dos minutos, para que ficasse mais- fácil a interação do usuário ao configurar o intervalo desejado e também para ajudar no entendimento da programação. O código em questão trabalha com variáveis inteiras.

O projeto baseia-se nas interrupções que o sistema recebe tanto para configuração do usuário por meio dos três botões (mais, menos e configuração) quanto por meio do *timer*. As interrupções são habilitadas por meio da instrução mostrada na figura 5.6.

```
//habilitação das interrupções usadas

   enable_interrupts(INT_RTCC);        //Timer
   enable_interrupts(INT_RB);          //Interrupção na portB
   enable_interrupts(GLOBAL);          //Interrupções globais
```

Figura 5.6 – Habilitando as interrupções
Fonte: Autor

Para a criação da interrupção via botões, foi criada uma máquina de estados para que fosse mais fácil a transição entre cada um dos estados e definir cada conjunto de funcionalidades disponíveis para cada botão dependendo de qual estado o sistema se encontrar. Foram criados doze modos para este projeto, cada modo sendo uma tela diferente do programa.

No Quadro 5.2 é mostrada a máquina de estados do projeto, com todas as telas exibidas e o funcionamento dos botões em cada modo.

Quadro 5.2 – Máquina de estados do projeto

	AÇÃO			
Estado	Display	Botão Modo	Botão Mais	Botão Menos
0	Aguardando Configuração	Estado seguinte	Nada	Nada
1	Alimentação _ : 00	Estado seguinte	Soma hora	Subtrai hora
2	Alimentação 0:_0	Estado seguinte	Soma minuto dezena	Subtrai minuto dezena
3	Alimentação 0:0_	Estado seguinte	Soma minuto unidade	Subtrai minuto unidade
4	Confirmar X:YZ S/N?	Nada	Confirma configuração	Nega configuração e volta para o modo 1
5	Alimentar Agora? S/N	Nada	Executa a alimentação e passa para o modo seguinte	Estado seguinte
6	Iluminação _: 00	Estado seguinte	Soma hora	Subtrai hora
7	Iluminação 0:_0	Estado seguinte	Soma minuto dezena	Subtrai minuto dezena
8	Iluminação 0:0_	Estado seguinte	Soma minuto unidade	Subtrai minuto unidade
9	Confirmar X:YZ S/N?	Nada	Confirma configuração	Nega configuração e volta para o modo 6
10	Iluminar Agora? S/N	Nada	Executa a iluminação e passa para o modo seguinte	Estado seguinte
11	Alim: x:xx Ilum y:yy	Retorna ao estado 1.	Nada	Nada

Fonte: Autor

Para a programação da interrupção do *timmer*, foi utilizado o próprio *timmer* presente no PIC16F877A. Parte da função do clock é mostrada na Figura 5.7.

```
#int_RTCC                                  //Função que controla o TMR0
                                           //e aciona interrupção a cada 1 minuto

void clock_isr(void){

contador++; //incrementação da variável contador
/*caso a variável atinja o valor de 3750, o equivalente a 1 segundo executa
    a instrução abaixo*/

if(contador >= 3750){ // 125 = 1s => 30s = 3750.
   //a cada 30 segundos
   contador = 0; //zera contador de interrupções.
   if(minuto){
      //a cada minuto
      if (modo == 11){           //Caso nenhum botão seja pressionado
         timerAlimentacao();      //em 30 segundos, volta para a tela
         timerIluminacao();       //de contagem regressiva
         mostrarTempoRestante();
      }
      minuto = 0;
   }else{
      minuto = 1;
   }
   //a cada 30 segundos
   //sair do modo de configuração

   if(modo!= 11){                 //Caso modo seja diferente de 11,
      //e não tenha sido feita configuração,
      //volta para tela inicial
```

Figura 5.7 – Função do clock
Fonte: Autor

O cristal escolhido para o projeto foi o de 4MGHZ, o que influencia no clock geral do microcontrolador. Utilizando-se de um prescaler de 64 bits para que seja dividido o sinal, temos uma frequência de entrada no timer 0 de 15625 Hz. Dividindo este sinal por 125, temos exatamente 125 Hz, e para que o programa divida o sinal por 125 para obtermos uma interrupção por segundo, basta iniciar a contagem com o valor 131 (256-125). Esse cálculo é necessário para calibrar exatamente o timer do microcontrolador para que funcione o mais precisamente possível. O projeto foi feito para interrupções a cada 30 segundos para facilitar a utilização da função de tempo esgotado caso não haja interação com o sistema durante esses 30 segundos.

A interrupção por meio do timer funciona para que ocorra a contagem regressiva e o usuário possa ter noção de quanto tempo falta para a execução da próxima operação. Assim que o contador mostrado no *display* chega a zero, indica que ocorreu o estouro do clock, acionando a interrupção e executando as funções programadas.

Há também uma interrupção por meio de um *timmer,* que foi criada com o objetivo de prever ocasional descuido do usuário ao apertar o botão de configuração sem a real intenção de configurar o dispositivo. Funciona para definir o tempo que o sistema pode ficar ocioso, ou seja, se deixado de lado por 30 segundos sem que nenhum botão seja apertado, o timer grava as configurações que tiverem sido confirmadas e volta para a tela da contagem regressiva.

O programa completo e comentado pode ser encontrado no ANEXO A.

5.3.2 Compilação e Gravação do Código Fonte

Com o código fonte escrito, o próximo passo é a sua compilação, com a utilização do PIC C *Compiler*, para que sejam gerados os arquivos necessários para a simulação e a gravação no PIC.

Na Figura 5.8 é mostrada a tela de compilação com sucesso, exibindo os arquivos gerados e a quantidade de memória utilizada pelo microcontrolador.

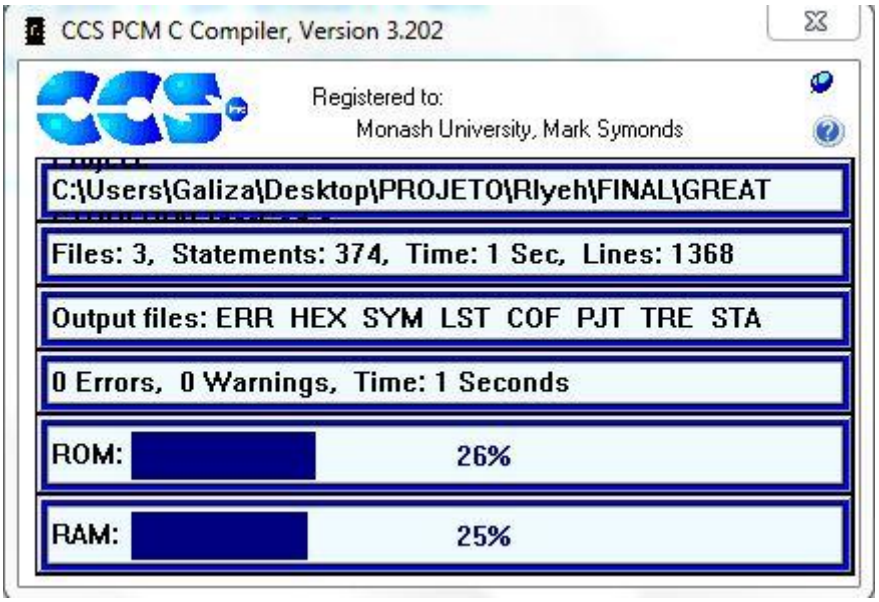

Figura 5.8 - Tela de compilação ISIS
Fonte: Autor

Pressionando o botão para compilar (F9), o compilador começa uma série de verificações de sintática, semântica e lógica no código, verificando possíveis erros que precisem ser corrigidos antes da compilação de fato. Caso haja algum problema, o programa indica o tipo de erro ocorrido assim como a linha em que esse

erro possivelmente ocorreu. Se não houve nenhum erro, o compilador gera oito arquivos com extensões diversas, sendo uma delas “HEX” que será utilizada tanto para a simulação quanto para a gravação no PIC.

A gravação no microcontrolador PIC é feita com auxílio do kit de gravação PICkit, as etapas de gravação seguem as descritas na seção – Utilização do Kit de Gravação PICkit - do capítulo anterior.

5.4 Montagem dos Circuitos nas Placas

As placas de circuito deste projeto foram feitas em placa de fenolite cobreada de face única. Neste tópico, será feita a descrição detalhada dos passos necessários para a sua confecção. Para a criação do *layout* da placa, é utilizado o *software* Proteus ISIS 7 Professional e Proteus ARES 7 Professional.

5.4.1 Proteus ARES

O programa Proteus ARES tem a finalidade de fazer o desenho esquemático que será impresso na placa, para que se possa utilizá-lo tanto como guia ao conectar os componentes, quanto ao queimar a placa com o Percloreto criando as trilhas de cobre.

O design do programa e seu funcionamento se assemelham ao ISIS, onde há uma tela onde serão dispostos os componentes que devem ser escolhidos através de uma lista contendo vários modelos e tipos para melhor se adequar ao projeto. É importante que todos sejam cuidadosamente escolhidos e posicionados pois a partir deste modelo que será feita a placa do circuito.

O objetivo do projeto foi fazer com que a placa fosse a menor possível, visando um melhor acabamento final. Para isso é preciso se atentar ao cruzamento das trilhas que ligam os dispositivos, pois nenhuma das trilhas de cobre pode se cruzar. Caso ocorra a necessidade, é utilizado um *jumper* que faz o papel de ponte, passando por cima da trilha.

Na Figura 5.9 é mostrado o esquema criado, mostrando o posicionamento de cada componente na placa e suas devidas conexões.

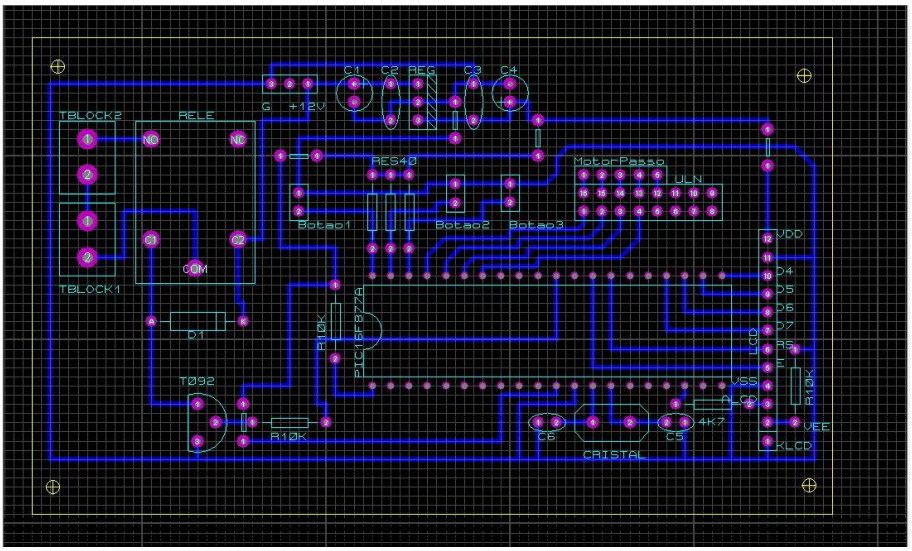

Figura 5.9 – Esquemático ARES
Fonte: Autor

5.4.3 Criação da Placa

A placa de fenolite comprada para o projeto é cobreada de face única, ou seja, ela possui uma de suas faces completamente coberta em cobre. Para que desta placa sejam feitas as trilhas dos circuitos, foi utilizado o resultado do programa ARES impresso em uma folha de papel couche para ser pressionado contra a placa e aquecido com ferro quente para que pudesse ser transferido o desenho impresso para a superfície da placa como mostra a Figura 5.10

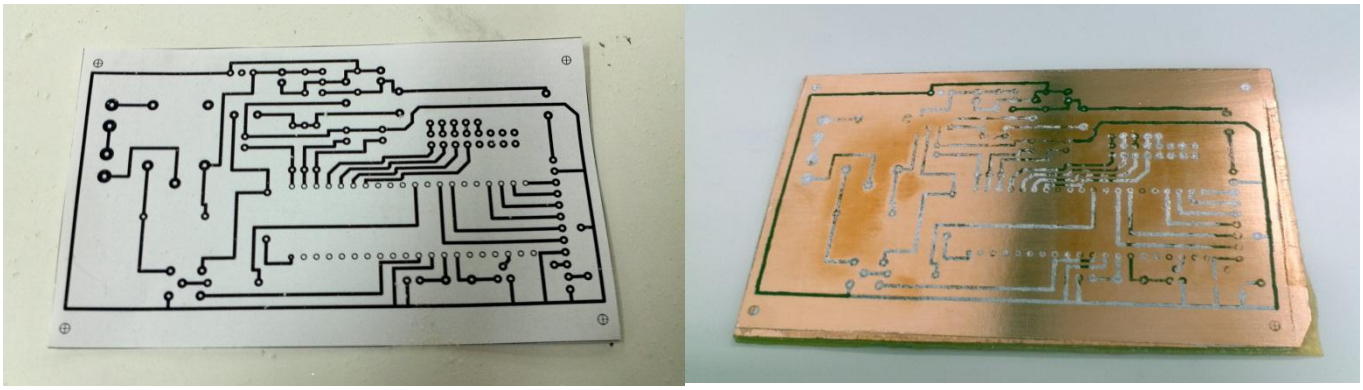

Figura 5.10 – Impressão ARES e placa Impressa
Fonte: Autor

A tinta marca a placa por meio do calor, e vai definir as áreas onde ficarão as trilhas de cobre. As imperfeições provenientes da impressão foram corrigidas utilizando caneta preta de CD. Impressa, a placa precisa ser mergulhada numa solução de Percloreto de Ferro que queimará o cobre da placa salvo os locais marcados para a trilha. A Figura 5.11 ilustra este processo em andamento.

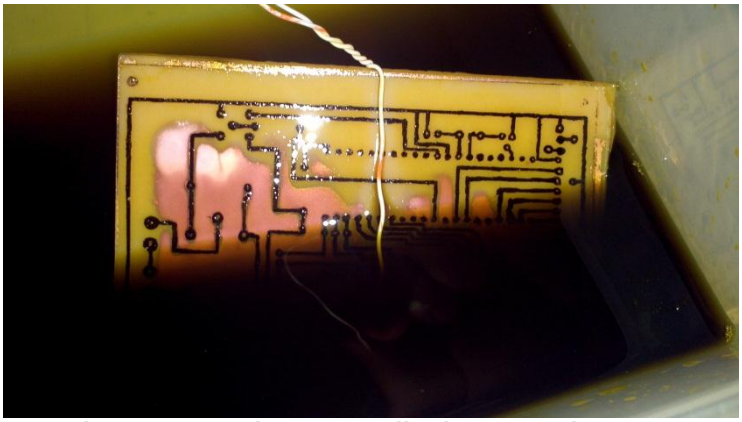

Figura 5.11 – Placa mergulhada no Percloreto
Fonte: Autor

Como resultado, as trilhas ficam bem definidas, prontas para conduzir energia entre os componentes da placa. O próximo passo é furar o local onde eles serão posicionados e soldá-los em seus respectivos lugares, soldando também a trilha para evitar oxidação do cobre. Na Figura 5.12 é possível observar a placa com as trilhas e componentes soldados.

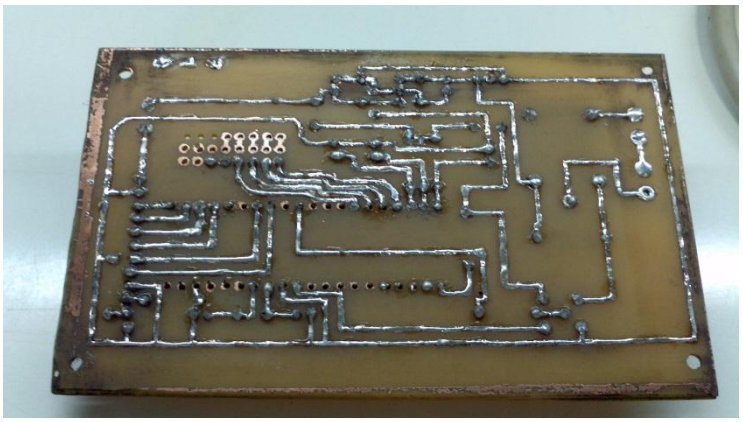

Figura 5.12 – Placa pronta e soldada
Fonte: Autor

5.5 Montagem do Protótipo

A maquete construída para a demonstração do projeto consiste de um aquário plantado de 40 cm x 20 cm x 30 cm no qual será acoplado o alimentador, e a iluminação será ligada diretamente na placa do projeto para que seja efetuado o seu controle.

Na Figura 5.13 é mostrada a placa e conexões do circuito, a iluminação e o aquário utilizado para testes e demonstração do projeto.

Figura 5.13 – Aquário, iluminação e circuito
Fonte: Autor

A descrição de como foi feita a confecção do dispositivo de alimentação será feita a seguir.

5.5.1 Dispositivo de Alimentação

O dispositivo de alimentação consiste de um tudo de mini CDs usado para conter a ração para peixes contendo furos em locais espaçados para que o alimento seja despejado quando o motor girar o dispositivo. O tubo contém três grupos de furos, para um melhor ajuste da quantidade que será despejada no aquário e pequenos adesivos que serão utilizados para tampar alguns dos buracos para que a dosagem seja mais precisa e de acordo com a necessidade do usuário.

O motor será acoplado ao tubo de CDs e será preso a uma haste de metal semi-flexível para que seja possível dobrar o dispositivo em direção ao centro do

aquário. A haste é presa a uma peça de metal em formato de “U” que será a forma que o dispositivo ficará preso ao aquário como mostra a Figura 5.14.

Figura 5.14 – Dispositivo de alimentação
Fonte: Autor

CAPÍTULO 6 – RESULTADOS OBTIDOS

Neste capítulo são apresentados os resultados obtidos para efetivamente solucionar o problema apresentado no Capítulo 2 – Apresentação do Problema – através da realização de simulações. São apresentados também alguns problemas encontrados.

6.1 Simulações

Esta fase do trabalho tem como objetivo o teste e comprovação do funcionamento do projeto. As simulações testarão todas as possibilidades de configuração e funcionalidades que o projeto possui para atender as necessidades do usuário. As metas dos testes são:

- Configuração da Alimentação e Iluminação;
- Teste do Dispositivo de Alimentação;
- Teste do Acionamento da Iluminação

Para iniciar os testes, primeiramente o protótipo e a iluminação foram ligados na tomada para ser realizada a configuração.

6.1.1 Configuração da Alimentação e Iluminação

Esta simulação tem como finalidade observar o funcionamento do dispositivo ao ser ligado, passando por todas as telas de configuração. Este teste foi executado em torno de 20 vezes, de forma a comprovar que o dispositivo está realizando todas as etapTas laborais do sistema. Os passos desta simulação foram:

- Ligar o interruptor do dispositivo e da Iluminação. Será mostrada uma mensagem no LCD, indicando que está ligado e aguardando configuração, conforme mostrado na figura 6.1;

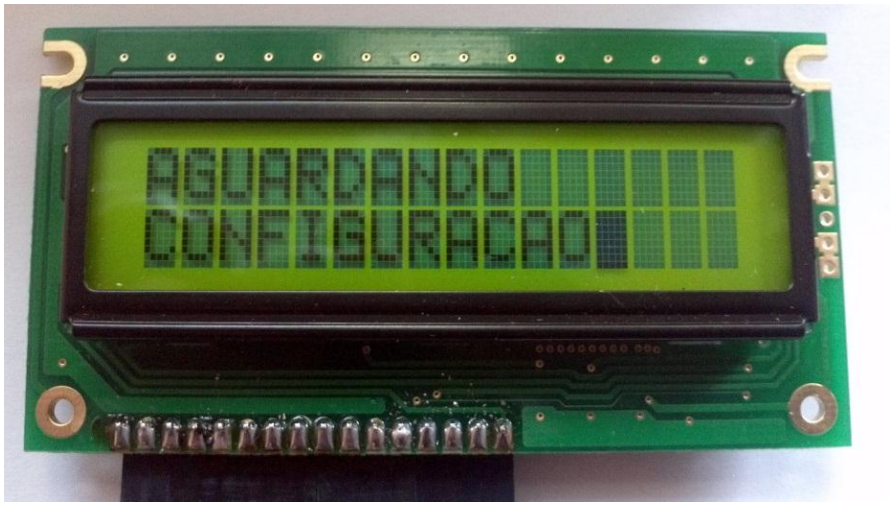

Figura 6.1 – Tela aguardando configuração
Fonte: Autor

- Ao pressionar a tecla de configuração (vermelha) será mostrada a tela de configuração do intervalo de alimentação, permitindo o ajuste da hora, dezena dos minutos e unidade dos minutos do intervalo pressionando a tecla de configuração para passar entre os campos de configuração como mostra a figura 6.2. Caso seja fornecido o horário 0:00, será considerado que o usuário não quer que o controle da alimentação seja executado;

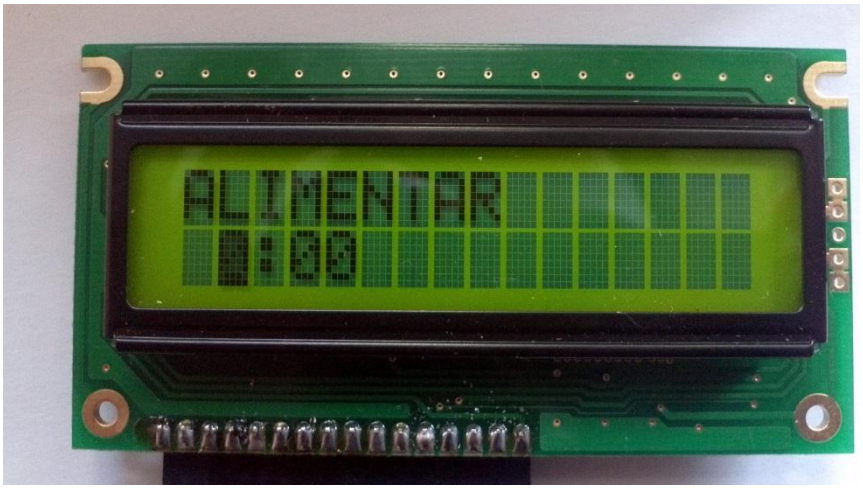

Figura 6.2 – Tela configuração do intervalo
Fonte: Autor

- Feita a configuração, será mostrada uma tela de confirmação, caso tenha sido feita alguma configuração errada, pressiona-se a tecla Menos/Não para voltar ao modo anterior e realizar ajustes, caso contrário, aceitando as configurações com o botão Mais/Sim levará à próxima tela. Na Figura 6.3 é mostrada a tela de configuração;

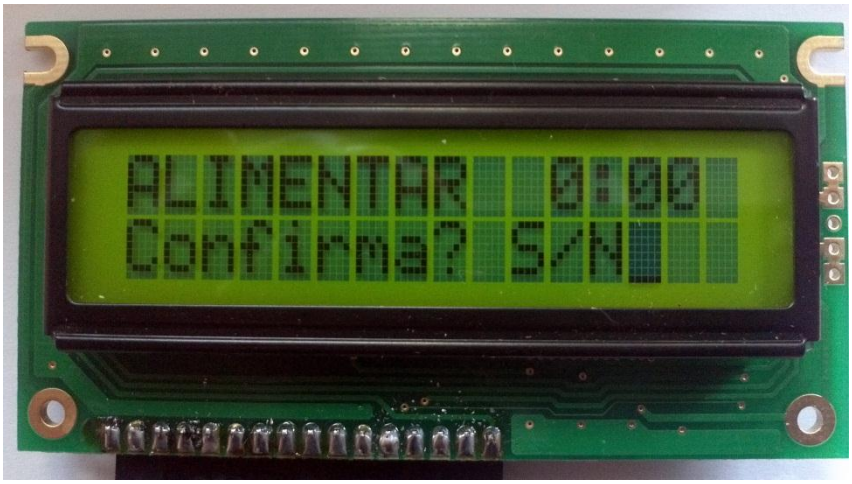

Figura 6.3 – Confirmação do intervalo de alimentação
Fonte: Autor

- É exibida uma tela de alimentação, perguntando se o usuário deseja que ela seja feita imediatamente ou somente após o intervalo programado como mostrado na figura 6.4 caso a resposta seja Sim, ou somente após o intervalo configurado caso contrário;

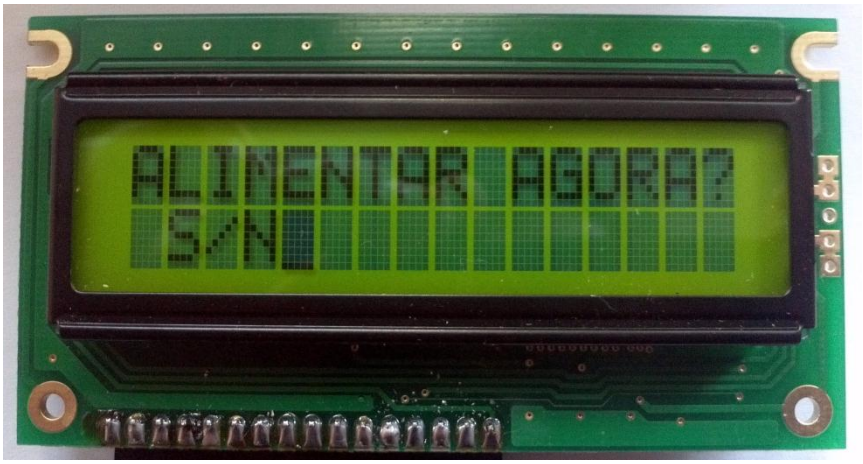

Figura 6.4 – Tela alimentação imediata
Fonte: Autor

- Começa a configuração da Iluminação, como mostra a figura 6.5, que funcionará de modo idêntico à alimentação, iniciando-se com a configuração do intervalo de iluminação. Caso seja fornecido o horário 0:00, será considerado que o usuário não quer que o controle da iluminação seja executado.

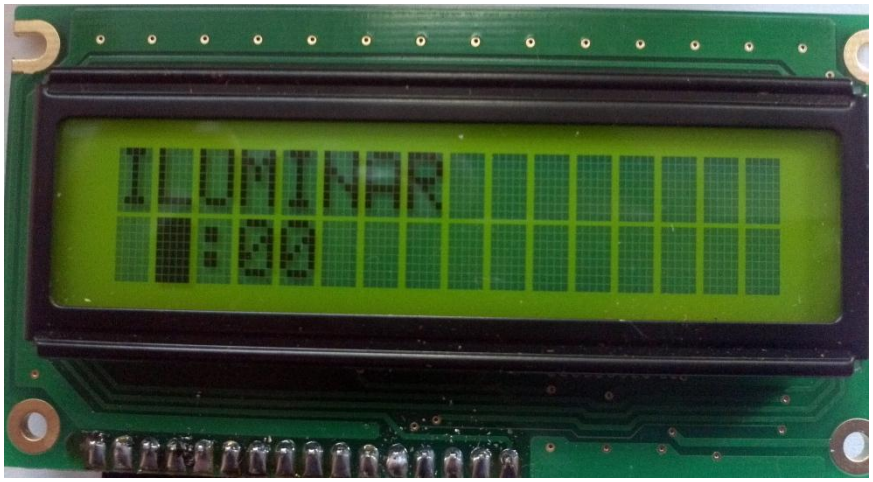

Figura 6.5 – Tela configuração do intervalo de iluminação
Fonte: Autor

- Tela de confirmação da configuração da iluminação previamente feita, com o mesmo funcionamento para reconfiguração apertando a tecla Menos/Não e confirmação apertando a tecla Mais/Sim como mostra a Figura 6.6.

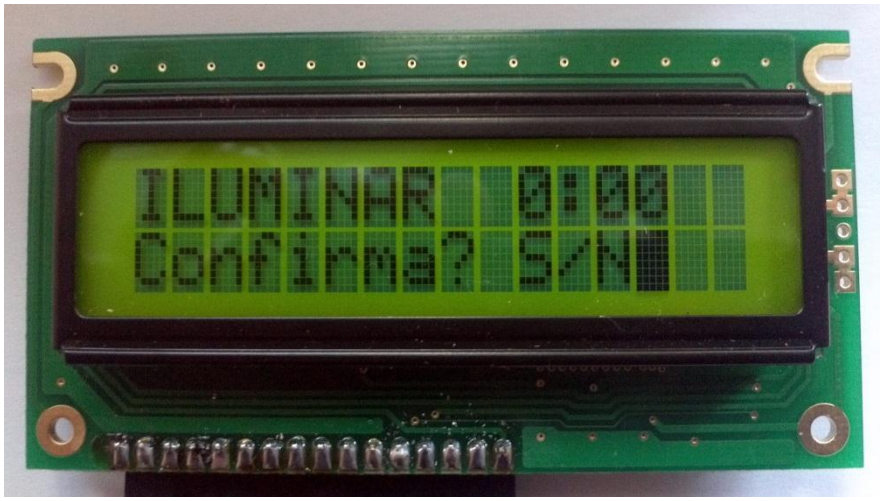

Figura 6.6 – Confirmação do intervalo de iluminação
Fonte: Autor

- A Figura 6.7 ilustra a ultima tela de configuração, pergunta se deseja alternar o estado da lâmpada imediatamente ou somente após o intervalo escolhido.

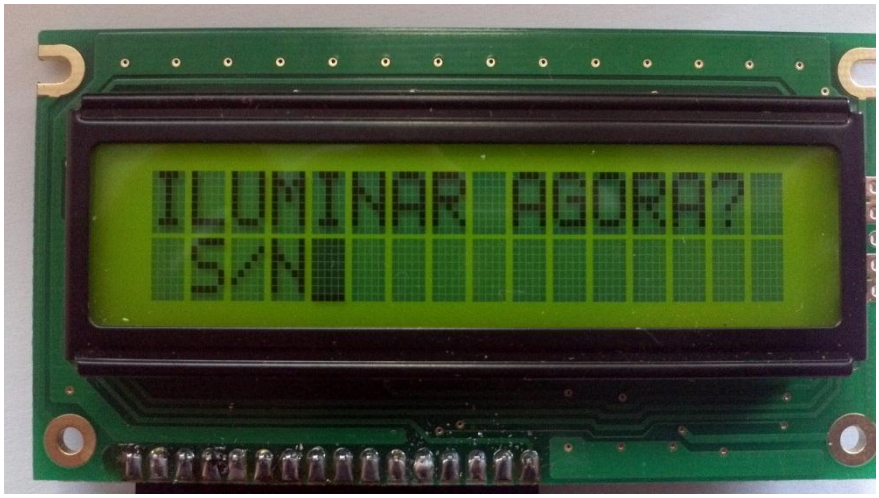

Figura 6.7 – Tela Iluminar imediatamente
Fonte: Autor

- Na Figura 6.8, é mostrada a tela de contagem regressiva, que ficará no display exibindo o tempo restante para que seja executada cada operação.

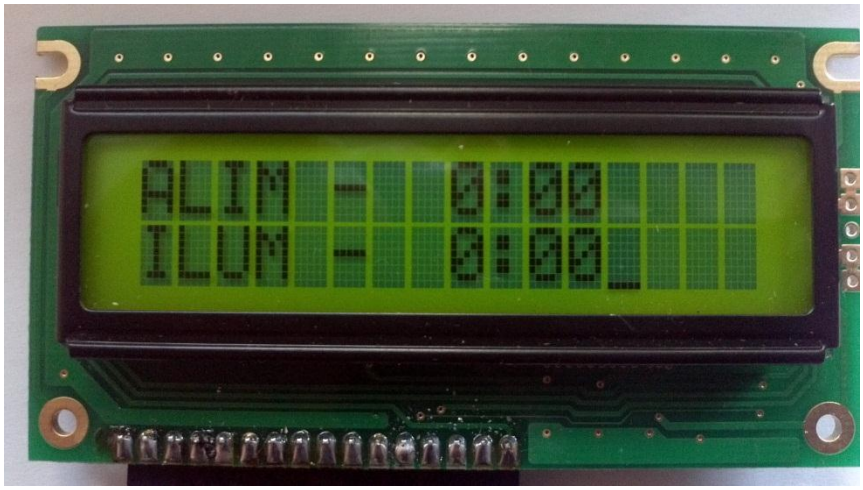

Figura 6.8 – Tela de contagem regressiva
Fonte: Autor

- O usuário pode pressionar novamente o botão de configuração caso queira fazer as alterações, iniciando novamente os passos descritos acima.
- Se o usuário apertar o botão de configuração sem querer, ou configurar parte do sistema e não desejar configurar o resto, o sistema voltará para a tela de contagem regressiva automaticamente em 30 segundos caso não seja apertado nenhum botão, ou para a tela de configuração, caso não tenha sido feito nenhuma.

Os resultados obtidos nesta primeira etapa do dispositivo demonstraram um funcionamento satisfatório. Este teste foi indispensável para ajustes finos como o que seria escrito em cada tela e o ajuste de *delay* nos botões. Não ocorreram erros no sistema, mostrando-se ser de fácil configuração e intuitivo para o usuário.

6.1.2 Teste do Dispositivo de Alimentação

O teste do dispositivo de alimentação tem como objetivo observar o seu funcionamento utilizando a ração de peixe e observar se a quantidade despejada é desejável, visando as alterações na quantidade de furos que o compartimento precisa possuir e ajustar essas variáveis à velocidade do giro do motor de passo que estará controlando o sistema. Para isso foram realizados cerca de 20 testes utilizando ração para peixes em grãos e observando a quantidade que seria despejada com cada quantidade de furos contidos no dispositivo.

Todos os testes se mostram consistentes com o esperado, pois foi possível a comprovação que o alimento do peixe cai em quantidade satisfatória podendo ser regulado dependendo da quantidade e tamanho dos peixes contidos no aquário. O adesivo escolhido para vedar os demais orifícios do dispositivo é suficiente para que não se desprendam com facilidade, e ainda permitem que sejam tirados e recolocados sem que a aderência seja perdida.

A alimentação foi testada com ração em grãos para peixe beta, por ser o tipo de ração mais comum e a mais aplicável no projeto pelo seu tamanho e seu peso leve evitando sobrecargas no motor de passo.

Na Figura 6.9 é ilustrado o teste do dispositivo de alimentação feito com a ração de peixe em grãos

Figura 6.9 – Teste realizado com a ração
Fonte: Autor

6.1.3 Teste da Iluminação

O teste da iluminação do projeto foi feito ligando um abajur específico com uma lâmpada fluorescente que não afetasse a temperatura do aquário com o calor gerado. A ligação da lâmpada foi conectada a um dos bornes, sendo a ligação do outro borne diretamente na tomada para energia. O projeto segue perfeitamente o fluxograma mostrado na figura, sendo configurado e correspondendo às informações fornecidas ao sistema.

O sistema foi testado como um todo, utilizando-se simultaneamente a iluminação e a alimentação, comprovando os testes feitos no PROTEUS ISIS.

Primeiramente é executada a alimentação do aquário, em seguida acionando o relé, alternando a iluminação. O projeto foi utilizado em situação real em um aquário plantado com a iluminação sendo alternada a cada 12 horas durante três dias.

6.2 Problemas Encontrados

Pode ser observado que os testes de funcionamento realizados no PROTEUS ISIS mostraram que o sistema funcionaria perfeitamente de acordo com o programado, mas houve algumas dificuldades de adaptação que uma simulação em software não é capaz de prever.

O funcionamento do motor de passo utilizado no projeto é realizado de forma diferente do modelo disposto no software. Tiveram de ser feitos diversos testes para que fosse descoberto o real funcionamento e como as informações deveriam ser mandadas para que fosse possível realizar o giro de 360º necessário no projeto.

Outro problema não previsto foi na construção da placa. Ao preparar as entradas na placa para driver de funcionamento do motor de passo, não foi considerado o espaço entre os furos na placa, o que tornou impossível a sua conexão. Para isso foi feita uma nova placa separada, que pudesse comportar o driver melhor.

Pequenos erros ao soldar o circuito, ao fazer os cabos necessários, pequenos ajustes na programação, queima de equipamentos por descuidos, também fizeram parte na construção deste projeto ajudando para que o mesmo chegasse em sua forma final.

O motor de passo utilizado no projeto, por ser um motor reciclado, não suporta grande quantidade de peso no alimentador, limitando a quantidade de comida que pode ser contida no dispositivo.

CAPÍTULO 7 – CONSIDERAÇÕES FINAIS

7.1 Conclusões

Neste trabalho foi desenvolvido um protótipo de aquário automatizado. O projeto aplica-se em aquários de pequeno porte, executando rotinas básicas de alimentação e iluminação. Para a resolução do problema proposto, foram necessários conhecimentos aplicados de microprocessadores, programação e circuitos elétricos.

O objetivo geral é a automação programável de rotinas cotidianas de um aquário. A correria da vida cotidiana traz uma constante falta de tempo e disposição para realizar tarefas básicas como cuidar como cuidar de um aquário com o cuidado que ele precisa. A exigência da obrigação acaba sendo um estorvo para o dono, e é exatamente este o problema que o trabalho se propõe a solucionar. A automação é aplicada visando maior conforto e liberdade para o dono e um cuidado mais atencioso às necessidades básicas do animal.

Para a sua concretização, foi necessário o desenvolvimento de duas placas: a placa principal e a placa contendo o driver e os pinos de entrada do motor de passo. Foi utilizado o PIC16F877A para controle dos componentes, um LCD para fornecer informações ao usuário, um motor de passo para girar o compartimento de alimentação e um relé para o acionamento da lâmpada.

Apesar dos problemas que surgiram ao longo do processo de elaboração deste projeto, o objetivo do trabalho foi alcançado. O dispositivo realiza com eficácia o controle e automação das rotinas de alimentação e iluminação do aquário, sendo de fácil configuração.

É possível concluir que os resultados obtidos cumpriram com as propostas e os objetivos planejados para este trabalho e o protótipo encontra-se em funcionamento e operando de acordo com a programação pretendida.

7.2 Propostas para Trabalhos Futuros

Aquarismo é uma área excelente para a automação, devido a necessidade de cuidado constante de todo um meio ambiente complexo e suas inúmeras

variáveis que tem que ser observadas e controladas como PH da água, alimentação, iluminação, temperatura e etc. O projeto atentou-se à forma mais simples de cuidado de um aquário, restando ainda diversas outras oportunidades em que a automação pode ser implementada para que o aquário se torne completamente independente. Algumas sugestões são:

- Automação do controle de PH do aquário;
- Controle de Temperatura;
- Tornar o projeto maior para se adequar às necessidades de aquários de maior porte;
- Aplicar um novo motor de passo, que tenha potencia para movimentar uma quantidade maior de ração no dispositivo de alimentação.

REFERÊNCIAS BIBLIOGRÁFICAS

ABREU, R. S. **Automação Residencial: um pouco de história.** Disponível em: http://www.aureside.org.br/publicacoes/download/automacao_residencial.zip. Acesso em: 29 de setembro, 2012.

AQUALANDIA. **Equipamentos (para Aquário).** Disponível em: http://www.aqualandia.com.br/site/aqualandia_equipamentos.html. Acesso em: 17 de outubro, 2012.

AQUARISMO BRASILEIRO. **Como cuidar do meu aquário - Iniciantes.** Disponível em: http://aquarismobrasileiro.com.br/art_anteriores/iniciantes.htm. Acesso em: 10 de outubro, 2012.

BORGES J, Hamilton J. OLIVEIRA, Renato A.B. **Aquário de Água Doce**. Disponível em: http://www.forumamordepeixe.com.br/download/aquariodeaguadocehistoricoeaulas.pdf. Acesso em: 15 de outubro, 2012.

BRAGA, N. C. **Tudo sobre relés**. Revista Saber Eletrônica. Disponível em: http://www.metaltex.com.br/tudosobrereles/tudo1.asp. Acesso em: 16 de novembro, 2012.

BRITES Felipe G. SANTOS, Vinicius P. A. **Motor de Passo**. Disponível em: http://www.telecom.uff.br/pet/petws/downloads/tutoriais/stepmotor/stepmotor2k81119.pdf. Acesso em: 10 de Novembro, 2012

CLICKINFORMAÇÃO. **Cuidados Com Aquário**. Disponível em: http://www.clickinformacao.com.br/animais/cuidados-com-aquario-montar-aquario.html Acesso em: 10 de Outubro, 2012.

COSTA, Helder. **Domótica – Edifícios Inteligentes**. Disponível em: http://www.engenium.net/344/domotica-edificios-inteligentes.html#78RPMj83XrfhdyE3.99. Acesso em: 15 de outubro, 2012.

COZZO,Reinaldo. **Relés e Contratores**. Disponível em: http://www.osetoreletrico.com.br/web/a-revista/edicoes/169-reles-e-contatores.html Acesso em: 13 de Novembro, 2012

DOVICCHI, João. **Estrutura de Dados:** http://www.inf.ufsc.br/~dovicchi/pos-ed/ebook/e-book_estrut_dados_dovicchi.pdf. Acesso em: 10 de Novembro 2012

FROTA, Yuri. **Quadro Vivo**. Disponível em: http://www.correiobraziliense.com.br/app/noticia/revista/2011/10/30/i275925/quadro-vivo.html. Acesso em: 09 de outubro, 2012.

GROOVER, Mikell P. **Automation, Production Systems and Computer Integrated Manufacturing Systems**. 2ª Ed. Prentice Hall, 2000.

MICROCHIP TECHNOLOGY INC. **PIC16F87XA.** DataSheet: 28/40/44-Pin Enhanced Flash Microcontrollers. 2003. Disponível em: http://ww1.microchip.com/downloads/en/devicedoc/39582b.pdf. Acesso em: 09 de setembro, 2012.

MICROCHIP TECHNOLOGY INC.. ***PICkit 2 Programmer/Debugger, User's Guide*.** 2008. LOCAL FALTANDO

PEREIRA, F. **Microcontroladores PIC:** Programação em C. 7ª. Ed. São Paulo: Érica, 2007.

SOUZA, D. J. de; LAVINIA, N. C. **Conectando o PIC 16F877A**: Recursos Avançados. 2ª. Ed. São Paulo: Érica, 2005.

ZANCO, W. da S. **Microcontroladores PIC com Base no PIC16F877A**: Técnicas de Software e Hardware para Projetos de Circuitos Eletrônicos. 2ª. Ed. São Paulo: Érica, 2010.

HARPER, Richard. **Inside the Smart Home**. Springer Verlag, Bristol, 2003.

JUCÁ, Sandro. **Apostila de Microcontroladores PIC e Periféricos**. Disponível em http://www.ebah.com.br/content/ABAAAAmX0AH/apostila-microcontroladores-pic-perifericos.html. Acesso em: 20 de outubro, 2012.

MORAES, Cícero. CASTRUCCI, Plinio. **Engenharia de Automação Industrial**. 2ª Ed. Editora LTC: FALTANDO LOCAL ,2007.

OLIVEIRA FILHO, João. **A Iluminação do Aquário**. Disponível em: http://pt.scribd.com/doc/22707747/Aquariofilia-A-Iluminacao-Do-Aquario. Acesso em: 10 de outubro, 2012.

PATSKO, Luiz F. **Tutorial Controle de Motor de Passo.** Disponível em: http://www.maxwellbohr.com.br/downloads/robotica/mec1000_kdr5000/tutorial_eletronica_-_motor_de_passo.pdf. Acesso em: 10 de Novembro 2012

TAYLOR, J.E. **The Aquarium: It's Inhabitants, Structure, and Management.** 1ª Edição. Edinburgh, 1910.

WEG. **Motores elétricos**. Disponível em: http://www.coe.ufrj.br/~richard/Acionamentos/Catalogo%20de%20Motores.pdf. Acesso em: 10 de Novembro, 2012.

APÊNDICE A - CÓDIGO FONTE DO DISPOSITIVO

```
/*=========================================================================
        PROJETO FINAL - Engenharia da Computação - UniCEUB
                2o. Semestre de 2012

                FILLIPE RIBEIRO GALIZA
                RA: 2081731/9

    AUTOMAÇÃO DE ROTINAS DE ALIMENTAÇÃO E ILUMINAÇÃO DE AQUÁRIO DOMÉSTICO
=========================================================================*/



/*=========================================================================
                        DIRETIVAS DE PRÉ-COMPILAÇÃO
=========================================================================*/
#include <16f877A.h>                    //Diretivas do PIC16F877A
#use delay(clock=4000000)               //Cristal oscilador 4Mhz.
#fuses hs, nowdt, put                   //Fusíveis
#fuses nobrownout, nolvp, noprotect
#fuses nocpd, nodebug
#use rs232(baud=9600, xmit=PIN_C6)      //Transmissão a 9600 bps
                                        //Sem paridade, 8 bits de dados
                                        //Pino TX RC6.

#use fast_io(b)                          //Dá controle do I/O ao programador
#byte portb=0x06                        //Endereça o portB
#bit btnModo=portb.7                    //Define o botão modo no port B7
#bit btnMais=portb.6                    //Define o botão mais no port B6
#bit btnMenos=portb.5                   //Define o botão mais no port B5
#define MOT1        PIN_B1              //Define MOT1 como o pino B1 do motor
#define MOT2        PIN_B2              //Define MOT2 como o pino B2 do motor
#define MOT3        PIN_B3              //Define MOT3 como o pino B2 do motor
#define MOT4        PIN_B4              //Define MOT4 como o pino B2 do motor
#define LAMP        PIN_D7              //Define LAMP como o pino D7 do Relê
#define INI_TMR0     131                //Define o TMR0 com valor inicial 131
#include <lcd-v2.0.h>                   //Biblioteca p/ o módulo LCD
/*=========================================================================
                        VARIÁVEIS
=========================================================================*/

int horaIluminacaoDefinitivo=0;
int minutoDezenaIluminacaoDefinitivo=0;
int minutoUnidadeIluminacaoDefinitivo=0;
int horaAlimentacaoDefinitivo=0;
int minutoDezenaAlimentacaoDefinitivo=0;
int minutoUnidadeAlimentacaoDefinitivo=0;
int passo=0;
int passovalor=0;
int modo=0;
```

```
int horaAlimentacao=0;
int minutoDezenaAlimentacao=0;
int minutoUnidadeAlimentacao=0;
int horaIluminacao=0;
int minutoDezenaIluminacao=0;
int minutoUnidadeIluminacao=0;
int botaomais=0;
int botaomenos=0;
long contador=0;
signed int horaIluminacaoRestante=0;
signed int minutoDezenaIluminacaoRestante=0;
signed int minutoUnidadeIluminacaoRestante=0;
signed int horaAlimentacaoRestante=0;
signed int minutoDezenaAlimentacaoRestante=0;
signed int minutoUnidadeAlimentacaoRestante=0;


/*=====================================================================
                    CHAMADA INICIAL DAS FUNÇÕES BÁSICAS
=====================================================================*/


void botaoModoApertado();           //Controla os acontecimentos
                                    //ao pressionar o botão configuração

void botaoMaisApertado();           //Controla os acontecimentos
                                    //ao pressionar o botão de MAIS/SIM

void botaoMenosApertado();          //Controla os acontecimentos
                                    //ao pressionar o botão de MENOS/NAO

void somaHoraAlimentacao();         //Soma a hora de 0 a 12 da Alimentação

void somaMinutoDezenaAlimentacao(); //Soma dezenas no minuto da Alimentação
void somaMinutoUnidadeAlimentacao();//Soma unidades no minuto da Alimentação
void somaHoraIluminacao();          //Soma a hora de 0 a 12 da Iluminação
void somaMinutoDezenaIluminacao();  //Soma dezenas no minuto da Iluminação
void somaMinutoUnidadeIluminacao(); //Soma unidades no minuto da iluminação
void subHoraAlimentacao();          //Subtrai a hora de 0 a 12 da Alimentação
void subMinutoDezenaAlimentacao();  //Subtrai dezena no minuto da Alimentação
void subMinutoUnidadeAlimentacao(); //Subtrai unidade no minuto da Alimentação
void subHoraIluminacao();           //Subtrai a hora de 0 a 12 da Iluminação
void subMinutoDezenaIluminacao();   //Subtrai dezena no minuto da Iluminação
void subMinutoUnidadeIluminacao();  //Subtrai unidade do minuto da Iluminação
void exibeAlimentacaoHora();        //Exibe a tela de configuração da hora
                                    //do intervalo de Alimentação

void exibeAlimentacaoDezena();      //Exibe a tela de configuração da dezena
                                    //dos minutos do intervalo de Alimentação

void exibeAlimentacaoUnidade();     //Exibe a tela de configuração da unidade
                                    //dos minutos do intervalo de Alimentação
```

```
void exibeAlimentacaoConfirmar();          //Exibe a tela confirmação Alimentação
void exibeAlimentacaoAgora();              //Exibe a tela de alimentação imediata
void confirmaAlimentacao();                //Confirma alimentação e salva variaveis
void alimentaAgora();                      //Chama a função girarMotor
void alimentarDepois();                    //Não executa alimentação e
                                           //aguarda estouro do timer

void cancelaConfigAlimentacao();           //Retorna p/ configuração da alimentação
void timerAlimentacao();                   //Contagem regressiva para alimentação
void setaTempoAlimRestante();              //Guarda as variáveis definitivas de
                                           //alimentação

void exibeIluminacaoHora();                //Exibe a tela de configuração da hora
                                           //do intervalo de Iluminação

void exibeIluminacaoDezena();              //Exibe a tela de configuração da dezena
                                           //dos minutos do intervalo de Iluminação

void exibeIluminacaoUnidade();             //Exibe a tela de configuração da unidade
                                           //dos minutos do intervalo de Iluminação

void exibeIluminacaoConfirmar();           //Exibe a tela confirmação da Iluminação
void exibeIluminacaoAgora();               //Exibe a tela iluminação imediata
void iluminarAgora();                      //Chama função acionarLampada
void iluminarDepois();                     //Não executa iluminação e
                                           //aguarda estouro do timer

void cancelaConfigIluminacao();            //Retorna p/ configuração da iluminação
void confirmaIluminacao();                 //Confirma iluminação e salva variáveis
void timerIluminacao();                    //Contagem regressiva para alimentação
void setaTempoIlumRestante();              //Guarda as variáveis definitivas de
                                           //iluminação

void acionarLampada();                     //Aciona o relê
void girarMotor();                         //Aciona o motor de passo
void inicializar();                        //Rotina de inicialização do dispositivo
void mostrarTempoRestante();               //Mostra o tempo restante para próxima
                                           //operação

void theFinalCountDown();                  //Zera o contador do modo e
                                           //chama função mostratemporestante


#int_RB                                    //Interrupção pelos botões

void RB_isr(void){

  if(!btnModo){                            //Caso qualquer um dos botões seja
    delay_ms(200);
    botaoModoApertado();                   //pressionado, chamar a função
  }else if(!btnMais){                      //correspondente
    delay_ms(200);
    botaoMaisApertado();
```

```
  }else if(!btnMenos){
    delay_ms(200);
    botaoMenosApertado();
  }

}

int minuto = 0;                                    //Flag que define se o contador
                                                   //contou 1 minuto ou 30 segundos

/*Função que controla o TMR0 do PIC com o contador configurado para acionar a
interrupção a cada 1 minuto*/

#int_RTCC                                           //Função que controla o TMR0
                                                   //e aciona interrupção a cada 1 minuto

void clock_isr(void){

   contador++;                                     //incrementação da variável contador
   /*caso a variável atinja o valor de 3750, o equivalente a 1 segundo executa
   a instrução abaixo*/

   if(contador >= 3750){ // 125 = 1s => 30s = 3750.
      //a cada 30 segundos
      contador = 0; //zera contador de interrupções.
      if(minuto){
         //a cada minuto
         if (modo == 11){                          //Caso nenhum botão seja pressionado
           timerAlimentacao();                     //em 30 segundos, volta para a tela
           timerIluminacao();                      //de contagem regressiva
           mostrarTempoRestante();
         }
         minuto = 0;
      }else{
         minuto = 1;
      }
       //a cada 30 segundos
       //sair do modo de configuração

      if(modo!= 11){                               //Caso modo seja diferente de 11,
                                                   //e não tenha sido feita configuração,
                                                   //volta para tela inicial

         if(horaAlimentacaoDefinitivo == 0
               && minutoDezenaAlimentacaoDefinitivo == 0
               && minutoUnidadeAlimentacaoDefinitivo == 0
               && horaIluminacaoDefinitivo == 0
               && minutoDezenaIluminacaoDefinitivo == 0
               && minutoUnidadeIluminacaoDefinitivo == 0){

            modo = 0;
            printf(lcd_escreve, "\fAGUARDANDO\nCONFIGURACAO");
         }else{ //fim if definitivos = 0
```

```
                theFinalCountDown();

            } //fim else modo = 11
        }
    }//fim if(contador >= 3750)
    //a cada interrupção
    set_timer0(131 - get_timer0());                 //Subtrai 131 do valor atual do tmr0
                                                    //e reescreve no tmr0

}



/*=====================================================================
                            FUNÇÃO PRINCIPAL
=====================================================================*/

void main() {

    inicializar();

    while(true){

    }

}


/*=====================================================================
                            FUNÇÕES
=====================================================================*/
/*=====================================================================
                        FUNÇÃO DE INICIALIZAÇÃO
=====================================================================*/


void inicializar(){
    output_b(0x00);
    set_tris_b(0b11100000);                     //Seta os pinos da porta B como entrada
    lcd_ini();                                  //Rotina de inicialização do display
    set_timer0(131);                            //Inicia o TMR0 com 131
    setup_timer_0(RTCC_INTERNAL | RTCC_DIV_64);   //Configurações TMR0 e prescaler
    setup_timer_1(T1_DISABLED);                  //Desabilita timers 1 e 2
    setup_timer_2(T2_DISABLED,0,1);
    setup_vref(FALSE);

//habilitação das interrupções usadas

    enable_interrupts(INT_RTCC);                //Timer
    enable_interrupts(INT_RB);                  //Interrupção na portB
    enable_interrupts(GLOBAL);                  //Interrupções globais
```

```
   lcd_escreve("\f");
   printf(lcd_escreve, "\fAGUARDANDO\nCONFIGURACAO");
}
/*=====================================================================
                          FUNÇÕES DOS BOTÕES
=====================================================================*/

//Função que controla os acontecimentos ao pressionar o botão de configuração

void botaoModoApertado(){
          minuto = 0;
          contador = 0;

  switch(modo){

     case 0:                                  //Não configurado ou
     case 11:                                 //contagem regressiva
     exibeAlimentacaoHora();                  //Avança para configuração hora
     break;
     case 1:                                  //Configuração hora
     exibeAlimentacaoDezena();                //Avança para config min (Dezena)
     break;
     case 2:                                  //Config min (Dezena)
     exibeAlimentacaoUnidade();               //Avança para config min (Unidade)
     break;
     case 3:                                  //Config min (Unidade)
     exibeAlimentacaoConfirmar();             //Avança para confirmar alimentação
     break;
     case 4:                                  //Aguardando confirmação ajuste hora
     break;
     case 5:                                  //Aguardando confirmação alimentação agora
     break;
     case 6:                                  //Configuração hora Iluminação
     exibeIluminacaoDezena();                 //Avança para config min (Dezena)
     break;
     case 7:                                  //Config minuto (Dezena)
     exibeIluminacaoUnidade();                //Avança para config minuto (Unidade)
     break;
     case 8:                                  //Config minuto (Unidade)
     exibeIluminacaoConfirmar();              //Avança para confirmar iluminação
     break;
     case 9:                                  //Aguardando confirmação ajuste hora
     break;
     case 10:                                 //Aguardando confirmação iluminar agora
     break;

     }
}

//Função que controla os acontecimentos ao apertar o botão Mais

void botaoMaisApertado(){
```

```
  minuto = 0;
  contador = 0;

  switch(modo){

    case 1:
    somaHoraAlimentacao();                  //Soma uma unidade na hora alimentação
    break;
    case 2:
    somaMinutoDezenaAlimentacao();          //Soma uma unidade na dezena do minuto da
    break;                                  //alimentação
    case 3:
    somaMinutoUnidadeAlimentacao();         //Soma uma unidade no minuto da
    break;                                  //alimentação
    case 4:
    confirmaAlimentacao();                  //Confirma configuração da alimentação
    break;
    case 5:
    alimentaAgora();                        //Chama função giraMotor
    break;
    case 6:
    somaHoraIluminacao();                   //Soma uma unidade na hora alimentação
    break;
    case 7:
    somaMinutoDezenaIluminacao();           //Soma uma unidade na dezena do minuto da
    break;                                  //alimentação
    case 8:
    somaMinutoUnidadeIluminacao();          //Soma uma unidade no minuto da
    break;                                  //Alimentação
    case 9:
    confirmaIluminacao();                   //Confirma configuração da iluminação
    break;
    case 10:
    iluminarAgora();                        //Chama função acionarLampada
    break;

    }
}

//Função que controla os acontecimentos ao apertar o botão Menos

void botaoMenosApertado(){

  minuto = 0;
  contador = 0;


  switch(modo){

    case 1:
    subHoraAlimentacao();                   //Subtrai uma unidade na hora alimentação
    break;
    case 2:
```

```
    subMinutoDezenaAlimentacao();          //Subtrai uma unidade na dezena do minuto
    break;                                 //da alimentação
    case 3:
    subMinutoUnidadeAlimentacao();         //Subtrai uma unidade no minuto da
    break;                                 //alimentação
    case 4:
    cancelaConfigAlimentacao();            //Cancela configuração e volta para a
    break;                                 //configuração da hora da alimentação
    case 5:
    alimentarDepois();                     //Cancela alimentação imediata
    break;
    case 6:
    subHoraIluminacao();                   //Subtrai uma unidade na hora iluminação
    break;
    case 7:
    subMinutoDezenaIluminacao();           //Subtrai uma unidade na dezena do minuto
    break;                                 //da iluminação
    case 8:
    subMinutoUnidadeIluminacao();          //Subtrai uma unidade no minuto da
    break;                                 //iluminação
    case 9:
    cancelaConfigIluminacao();             //Cancela configuração e volta para a
    break;                                 //configuração da hora da iluminação
    case 10:
    iluminarDepois();                      //Cancela iluminação imediata
    break;

  }
}



/*=========================================================================
                    FUNÇÕES DO BOTÃO CONFIGURAÇÃO
=========================================================================*/


void exibeAlimentacaoHora(){
  printf(lcd_escreve, "\fALIMENTAR\n%2d:%d%d",horaAlimentacao,
  minutoDezenaAlimentacao, minutoUnidadeAlimentacao);
  lcd_pos_xy(2,2);                         //Posiciona o cursor na posição da hora
  modo = 1;                                //Passa para o modo seguinte
}

void exibeAlimentacaoDezena(){
  printf(lcd_escreve, "\fALIMENTAR\n%2d:%d%d",horaAlimentacao,
  minutoDezenaAlimentacao, minutoUnidadeAlimentacao);
  lcd_pos_xy(4,2);                          //Posiciona o cursor na posição da dezena do minuto
  modo = 2;                                //Passa para o modo seguinte
}
```

```
void exibeAlimentacaoUnidade(){
  printf(lcd_escreve, "\fALIMENTAR\n%2d:%d%d",horaAlimentacao,
  minutoDezenaAlimentacao, minutoUnidadeAlimentacao);
  lcd_pos_xy(5,2);                          //Posiciona o cursor na posição da unidade do minuto
  modo = 3;                                 //Passa para o modo seguinte
}

void exibeAlimentacaoConfirmar(){
  printf(lcd_escreve, "\fALIMENTAR %2d:%d%d\nConfirma? S/N",
  horaAlimentacao, minutoDezenaAlimentacao, minutoUnidadeAlimentacao);
  modo = 4;                                 //Passa para o modo 4, anulando o botão modo
                                            //até que seja respondida a pergunta
}
void exibeAlimentacaoAgora(){
  printf(lcd_escreve, "\fALIMENTAR AGORA?\n S/N");
  modo = 5;                                 //Passa para o modo 5, anulando o botão modo
                                            //até que seja respondida a pergunta
}


void exibeIluminacaoHora(){
  printf(lcd_escreve, "\fILUMINAR\n%2d:%d%d",horaIluminacao,
  minutoDezenaIluminacao, minutoUnidadeIluminacao);
  lcd_pos_xy(2,2);                          //Posiciona o cursor na posição da hora
  modo = 6;
}

void exibeIluminacaoDezena(){
  printf(lcd_escreve, "\fILUMINAR\n%2d:%d%d",horaIluminacao,
  minutoDezenaIluminacao, minutoUnidadeIluminacao);
  lcd_pos_xy(4,2);                          //Posiciona o cursor na posição da dezena do minuto
  modo = 7;                                 //Passa para o modo seguinte
}

void exibeIluminacaoUnidade(){
  printf(lcd_escreve, "\fILUMINAR\n%2d:%d%d",horaIluminacao,
  minutoDezenaIluminacao, minutoUnidadeIluminacao);
  lcd_pos_xy(5,2);                          //Posiciona o cursor do LCD na posição da unidade
                                            //do minuto
  modo = 8;                                 //Passa para o modo seguinte
}


void exibeIluminacaoConfirmar(){
  printf(lcd_escreve, "\fILUMINAR %2d:%d%d\nConfirma? S/N",
  horaIluminacao, minutoDezenaIluminacao, minutoUnidadeIluminacao);
  modo = 9;                                 //Passa para o modo 9, anulando o botão modo
                                            //até que seja respondida a pergunta
}

void exibeIluminacaoAgora(){
  printf(lcd_escreve, "\fILUMINAR AGORA?\n S/N");
  modo = 10;                                //Passa para o modo 10, anulando o botão modo
```

```
                                        //até que seja respondida a pergunta
}

void theFinalCountDown(){
   //contagem regressiva
   mostrarTempoRestante();              //Passa para o modo 11 e exibe
   modo=11;                             //a contagem regressiva
   contador = 0;
}

void mostrarTempoRestante(){            //Exibe o tempo restante para a próxima operação
   printf(lcd_escreve, "\fALIM - %2d:%d%d", horaAlimentacaoRestante,
   minutoDezenaAlimentacaoRestante, minutoUnidadeAlimentacaoRestante);
   printf(lcd_escreve, "\nILUM - %2d:%d%d", horaIluminacaoRestante,
   minutoDezenaIluminacaoRestante, minutoUnidadeIluminacaoRestante);
}


/*=====================================================================
                        FUNÇÕES DO BOTÃO MAIS
=====================================================================*/



void somaHoraAlimentacao(){

  if (horaAlimentacao == 12)
    horaAlimentacao = 0;
  else
    horaAlimentacao++;                  //Soma uma unidade na hora na alimentação

  printf(lcd_escreve, "\fALIMENTAR\n%2d:%d%d",horaAlimentacao,
  minutoDezenaAlimentacao, minutoUnidadeAlimentacao);
  lcd_pos_xy(2,2);                      //Posiciona o cursor na posição da hora
}


void somaMinutoDezenaAlimentacao(){

  if (minutoDezenaAlimentacao == 5)
    minutoDezenaAlimentacao = 0;
  else
    minutoDezenaAlimentacao++;          //Soma uma unidade na dezena do minuto
                                        //da alimentação
  printf(lcd_escreve, "\fALIMENTAR\n%2d:%d%d",horaAlimentacao,
  minutoDezenaAlimentacao, minutoUnidadeAlimentacao);
  lcd_pos_xy(4,2);                      //Posiciona o cursor na posição da dezena
                                        //do minuto
}


void somaMinutoUnidadeAlimentacao(){
```

```
  if (minutoUnidadeAlimentacao == 9)
    minutoUnidadeAlimentacao = 0;
  else
    minutoUnidadeAlimentacao++;              //Soma uma unidade do minuto da alimentação

 printf(lcd_escreve, "\fALIMENTAR\n%2d:%d%d",horaAlimentacao,
 minutoDezenaAlimentacao, minutoUnidadeAlimentacao);
 lcd_pos_xy(5,2);                            //Posiciona o cursor na posição da unidade
                                             //do minuto
}

void somaHoraIluminacao(){

  if (horaIluminacao == 12)
    horaIluminacao = 0;
  else
    horaIluminacao++;                        //Soma uma unidade na hora da iluminação

  printf(lcd_escreve, "\fILUMINAR\n%2d:%d%d",horaIluminacao,
  minutoDezenaIluminacao, minutoUnidadeIluminacao);
  lcd_pos_xy(2,2);                            //Posiciona o cursor na posição da hora


}

void somaMinutoDezenaIluminacao(){

  if (minutoDezenaIluminacao == 5)
    minutoDezenaIluminacao = 0;
  else
    minutoDezenaIluminacao++;                //Soma uma unidade na dezena do minuto
                                             //da iluminação
  printf(lcd_escreve, "\fILUMINAR\n%2d:%d%d",horaIluminacao,
  minutoDezenaIluminacao, minutoUnidadeIluminacao);
  lcd_pos_xy(4,2);                           //Posiciona o cursor na posição da dezena
                                             //do minuto
}

void somaMinutoUnidadeIluminacao(){

  if (minutoUnidadeIluminacao == 9)
    minutoUnidadeIluminacao = 0;
  else
    minutoUnidadeIluminacao++;               //Soma uma unidade do minuto da iluminação

  printf(lcd_escreve, "\fILUMINAR\n%2d:%d%d",horaIluminacao,
  minutoDezenaIluminacao, minutoUnidadeIluminacao);
  lcd_pos_xy(5,2);                           //Posiciona o cursor na posição da unidade
                                             //do minuto
}

void confirmaAlimentacao(){
    horaAlimentacaoDefinitivo = horaAlimentacao;
```

```
    minutoDezenaAlimentacaoDefinitivo = minutoDezenaAlimentacao;
    minutoUnidadeAlimentacaoDefinitivo = minutoUnidadeAlimentacao;
    setaTempoAlimRestante();

    exibeAlimentacaoAgora();                    //Salva as variáveis provisórias para as
    clock_isr();                                //definitivas e aciona o clock


}

void setaTempoAlimRestante(){                   //Salva os valores para as variáveis a serem
                            //decrementadas
   horaAlimentacaoRestante=horaAlimentacaoDefinitivo;
   minutoDezenaAlimentacaoRestante=minutoDezenaAlimentacaoDefinitivo;
   minutoUnidadeAlimentacaoRestante=minutoUnidadeAlimentacaoDefinitivo;
}

void alimentaAgora(){
     printf(lcd_escreve, "\fALIMENTANDO\nAguarde...");
     girarMotor();                              //Aciona o motor de passo
     exibeIluminacaoHora();

}

void confirmaIluminacao(){
   horaIluminacaoDefinitivo = horaIluminacao;
   minutoDezenaIluminacaoDefinitivo = minutoDezenaIluminacao;
   minutoUnidadeIluminacaoDefinitivo = minutoUnidadeIluminacao;
   setaTempoIlumRestante();
   exibeIluminacaoAgora();                      //Salva as variáveis provisórias para as
   clock_isr();                                 //Definitivas e aciona o clock


}

void setaTempoIlumRestante(){                   //Salva os valores para as variáveis a serem
                                                //decrementadas
   horaIluminacaoRestante=horaIluminacaoDefinitivo;
   minutoDezenaIluminacaoRestante=minutoDezenaIluminacaoDefinitivo;
   minutoUnidadeIluminacaoRestante=minutoUnidadeIluminacaoDefinitivo;
}


void iluminarAgora(){                           //Aciona a lâmpada e exibe a
                                                //contagem regressiva

   acionarLampada();
   theFinalCountDown();


}
```

```
/*=====================================================================
                        FUNÇÕES DO BOTÃO MENOS
=====================================================================*/


void subHoraAlimentacao(){

  if (horaAlimentacao == 0)
    horaAlimentacao = 12;
  else
    horaAlimentacao--;                    //Soma uma unidade na hora na alimentação

  printf(lcd_escreve, "\fALIMENTAR\n%2d:%d%d",horaAlimentacao,
  minutoDezenaAlimentacao, minutoUnidadeAlimentacao);
  lcd_pos_xy(2,2);                        //Posiciona o cursor na posição da hora


}

void subMinutoDezenaAlimentacao(){

  if (minutoDezenaAlimentacao == 0)
    minutoDezenaAlimentacao = 5;
  else
    minutoDezenaAlimentacao--;            //Soma uma unidade na dezena do minuto
                                          //na alimentação
  printf(lcd_escreve, "\fALIMENTAR\n%2d:%d%d",horaAlimentacao,
  minutoDezenaAlimentacao, minutoUnidadeAlimentacao);
  lcd_pos_xy(4,2);                        //Posiciona o cursor na posição da dezena
                                          //do minuto

}

void subMinutoUnidadeAlimentacao(){

  if (minutoUnidadeAlimentacao == 0)
    minutoUnidadeAlimentacao = 9;
  else
    minutoUnidadeAlimentacao--;           //Subtrai uma unidade no minuto
                                          //da alimentação

  printf(lcd_escreve, "\fALIMENTAR\n%2d:%d%d",horaAlimentacao,
  minutoDezenaAlimentacao, minutoUnidadeAlimentacao);
  lcd_pos_xy(5,2);                        //Posiciona o cursor na posição da unidade
                                          //do minuto
}
```

```
void subHoraIluminacao(){

  if (horaIluminacao == 0)
    horaIluminacao = 12;
  else
    horaIluminacao--;                    //Soma uma unidade na hora na iluminação

  printf(lcd_escreve, "\fILUMINAR\n%2d:%d%d",horaIluminacao,
  minutoDezenaIluminacao, minutoUnidadeIluminacao);
  lcd_pos_xy(2,2);                       //Posiciona o cursor na posição da hora


}

void subMinutoDezenaIluminacao(){

  if (minutoDezenaIluminacao == 0)
    minutoDezenaIluminacao = 5;
  else
    minutoDezenaIluminacao--;            //Soma uma unidade na dezena do minuto
                                         //na iluminação

  printf(lcd_escreve, "\fILUMINAR\n%2d:%d%d",horaIluminacao,
  minutoDezenaIluminacao, minutoUnidadeIluminacao);
  lcd_pos_xy(4,2);                       //Posiciona o cursor na posição da dezena
                                         //do minuto


}

void subMinutoUnidadeIluminacao(){

  if (minutoUnidadeIluminacao == 0)
    minutoUnidadeIluminacao = 9;
  else
    minutoUnidadeIluminacao--;           //Soma uma unidade na unidade do minuto
                                         //na alimentação

  printf(lcd_escreve, "\fILUMINAR\n%2d:%d%d",horaIluminacao,
  minutoDezenaIluminacao, minutoUnidadeIluminacao);
  lcd_pos_xy(5,2);                       //Posiciona o cursor na posição da unidade
                                         //do minuto
}


void cancelaConfigAlimentacao(){
  exibeAlimentacaoHora();                //Cancela a configuração da alimentação
                                         //e retorna ao modo de configuração da hora
                                         //da alimentação
}

void cancelaConfigIluminacao(){
   exibeIluminacaoHora();                //Cancela a configuração da alimentação
```

```
                                        //e retorna ao modo de configuração da hora
                                        //da iluminação
}

void alimentarDepois(){                 //Cancela alimentação imediata e passa
                                        //para a configuração da iluminação
    exibeIluminacaoHora();

}

void iluminarDepois(){

   theFinalCountDown();                 //Cancela a iluminação imediata e passa
                                        //para a contagem regressiva para a próxima
}                                       //operação


/*=====================================================================
                        FUNÇÕES DE OPERAÇÃO
=====================================================================*/

void girarMotor(){                      //Gira o motor de passo 360º

  delay_ms(500);
  output_high(MOT3);
  output_low(MOT4);
  output_low(MOT1);
  output_low(MOT2);

  delay_ms(500);
  output_low(MOT3);
  output_high(MOT4);
  output_low(MOT1);
  output_low(MOT2);

  delay_ms(500);
  output_high(MOT3);
  output_low(MOT4);
  output_low(MOT1);
  output_low(MOT2);

  delay_ms(500);
  output_low(MOT3);
  output_high(MOT4);
  output_low(MOT1);
  output_low(MOT2);

  delay_ms(500);
  output_high(MOT3);
  output_low(MOT4);
  output_low(MOT1);
  output_low(MOT2);
```

```
delay_ms(500);
output_low(MOT3);
output_high(MOT4);
output_low(MOT1);
output_low(MOT2);

delay_ms(500);
output_high(MOT3);
output_low(MOT4);
output_low(MOT1);
output_low(MOT2);

delay_ms(500);
output_low(MOT3);
output_high(MOT4);
output_low(MOT1);
output_low(MOT2);

delay_ms(500);
output_high(MOT3);
output_low(MOT4);
output_low(MOT1);
output_low(MOT2);

delay_ms(500);
output_low(MOT3);
output_high(MOT4);
output_low(MOT1);
output_low(MOT2);

delay_ms(500);
output_high(MOT3);
output_low(MOT4);
output_low(MOT1);
output_low(MOT2);

delay_ms(500);
output_low(MOT3);
output_high(MOT4);
output_low(MOT1);
output_low(MOT2);

delay_ms(500);
output_high(MOT3);
output_low(MOT4);
output_low(MOT1);
output_low(MOT2);

delay_ms(500);
output_low(MOT3);
output_high(MOT4);
output_low(MOT1);
output_low(MOT2);
```

```
delay_ms(500);
output_high(MOT3);
output_low(MOT4);
output_low(MOT1);
output_low(MOT2);

delay_ms(500);
output_low(MOT3);
output_high(MOT4);
output_low(MOT1);
output_low(MOT2);

delay_ms(500);
output_high(MOT3);
output_low(MOT4);
output_low(MOT1);
output_low(MOT2);

delay_ms(500);
output_low(MOT3);
output_high(MOT4);
output_low(MOT1);
output_low(MOT2);

delay_ms(500);
output_high(MOT3);
output_low(MOT4);
output_low(MOT1);
output_low(MOT2);

delay_ms(500);
output_low(MOT3);
output_high(MOT4);
output_low(MOT1);
output_low(MOT2);

delay_ms(500);
output_high(MOT3);
output_low(MOT4);
output_low(MOT1);
output_low(MOT2);

delay_ms(500);
output_low(MOT3);
output_high(MOT4);
output_low(MOT1);
output_low(MOT2);

delay_ms(500);
output_high(MOT3);
output_low(MOT4);
output_low(MOT1);
```

```
   output_low(MOT2);

   delay_ms(500);
   output_low(MOT3);
   output_high(MOT4);
   output_low(MOT1);
   output_low(MOT2);   passo++;

}

void acionarLampada(){

       output_toggle(LAMP);                    //Aciona o pino que o relê está ligado


}



/*=====================================================================
                              FUNÇÕES TIMER
=====================================================================*/


void timerAlimentacao(){                       //Decrementador do tempo de Alimentação

   if(horaAlimentacaoRestante == 0
         && minutoDezenaAlimentacaoRestante == 0
         && minutoUnidadeAlimentacaoRestante == 0){

         }
   else{

         minutoUnidadeAlimentacaoRestante--;
         if (minutoUnidadeAlimentacaoRestante < 0){
           minutoUnidadeAlimentacaoRestante = 9;
           minutoDezenaAlimentacaoRestante--;
         }
         if (minutoDezenaAlimentacaoRestante < 0){
              minutoDezenaAlimentacaoRestante = 5;
              horaAlimentacaoRestante--;
         }
         if (horaAlimentacaoRestante == 0
               && minutoDezenaAlimentacaoRestante == 0
               && minutoUnidadeAlimentacaoRestante == 0){
             printf(lcd_escreve, "\fALIMENTANDO\nAguarde...");
             girarMotor();
             setaTempoAlimRestante();
         }
      }
}
```

```
void timerIluminacao(){                          //Decrementador do tempo de iluminação

   if  (horaIluminacaoRestante == 0
         && minutoDezenaIluminacaoRestante == 0
         && minutoUnidadeIluminacaoRestante == 0){
         }

   else
   {

         minutoUnidadeIluminacaoRestante --;

         if (minutoUnidadeIluminacaoRestante < 0){
           minutoUnidadeIluminacaoRestante = 9;
           minutoDezenaIluminacaoRestante --;
         }
         if (minutoDezenaIluminacaoRestante < 0){
           minutoDezenaIluminacaoRestante = 5;
           horaIluminacaoRestante --;
         }
         if (horaIluminacaoRestante == 0
               && minutoDezenaIluminacaoRestante == 0
               && minutoUnidadeIluminacaoRestante == 0){

            acionarLampada();
            setaTempoIlumRestante();
               }
   }
}
```