UNICEUB – CENTRO UNIVERSITÁRIO DE BRASÍLIA

FATECS – FACULDADE DE TECNOLOGIA E CIÊNCIAS SOCIAIS APLICADAS

CURSO DE ENGENHARIA DA COMPUTAÇÃO

FERNANDO ULHÔA PIMENTEL

ACIONAMENTO DE INTERRUPÇÃO EM SEMÁFORO PARA PEDESTRES POR BOTÃO NO SOLO

Orientadora: Prof. M.C. Maria Marony Sousa Farias Nascimento.

BRASÍLIA / DF

JULHO, 2012

FERNANDO ULHÔA PIMENTEL

ACIONAMENTO DE INTERRUPÇÃO EM SEMÁFORO PARA PEDESTRES POR BOTÃO NO SOLO

Monografia apresentada ao UniCEUB – Centro Universitário de Brasília para o curso de Engenharia da Computação, como requisito parcial para obtenção do grau de Engenheiro de Computação.

Orientadora: Prof. M.C. Maria Marony Sousa Farias Nascimento.

BRASÍLIA / DF

JULHO, 2012

FERNANDO ULHÔA PIMENTEL

ACIONAMENTO DE INTERRUPÇÃO EM SEMÁFORO PARA PEDESTRES POR BOTÃO NO SOLO

Monografia apresentada ao UniCEUB – Centro Universitário de Brasília para o curso de Engenharia da Computação, como requisito parcial para obtenção do grau de Engenheiro de Computação.

Orientadora: Prof. M.C. Maria Marony Sousa Farias Nascimento.

BRASÍLIA

JUNHO, 2012

RESUMO

Trata-se de uma proposta de automação do acionamento de um semáforo de pedestres, por meio de uma interface no solo sensível ao pressionamento. O objetivo é facilitar o uso da faixa de pedestres, aumentando sua segurança, acessibilidade e higiene. Ao ser acionada pela presença do pedestre tem-se a certeza que o mesmo se encontra a espera, fazendo com que o sinal seja fechado para os veículos, permitindo novamente o trânsito quando o pedestre alcançar o outro lado da pista. Para implementação deste projeto foi utilizado um *hardware* conectado a um micro-controlador PIC16F628A, programado para ler as entradas e executar os comandos de acordo com a lógica atribuída.

Palavras-chave: automação, PIC e semáforo.

ABSTRACT

This is a proposal for an automated pedestrian traffic light, employing a pressure-sensitive touch interface placed on the ground. The aim is to facilitate the use of the pedestrian crosswalk, increasing its safety, accessibility and hygiene. When triggered by the presence of pedestrians, the traffic light is closed to vehicles, allowing the traffic to flow again only when the pedestrians have reached the other side of the street. To implement this project, a hardware has been connected to a PIC 16F628A micro-controller programmed to read the input signals and execute the commands according to the assigned logic.

Keywords: automation, PIC and traffic light.

Este projeto e monografia são dedicados especialmente, aos meus pais, meus irmãos, a minha grande família, aos professores e a todos aqueles que tiveram paciência e acreditaram em mim durante todo o tempo.

AGRADECIMENTOS

Agradeço a Deus por me permitir estar vivo e viver este momento.

Agradeço aos meus pais por estarem ao meu lado na superação das dificuldades e por serem os melhores pais que alguém poderia ter. Amo vocês!!!

Agradeço aos meus irmãos por estarmos unidos.

Agradeço à minha namorada Taissa Iorrana pela compreensão, apoio e força.

Aos meus amigos e parentes.

E a todos que estiveram comigo nestes tempos difíceis, colegas de sala, do trabalho, ao José Carlos, Sergio, Samir entre muitos outros, muito obrigado.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS

LED – Diodo Emissor de Luz

LCD – Display de Cristal Líquido

GND – Fio Terra

Vcc – Fio de Alimentação

BE – Botão de Entrada

BS – Botão de Saída

S.VD.C – Sinal Verde para Carros

S.VM.C – Sinal Vermelho para Carros

S.AM.C – Sinal Amarelo para Carros

S.VD.P – Sinal Verde para Pedestres

S.VM.P – Sinal Vermelho para Pedestres.

Buzzer – Auto falante para alarme sonoro

RAM – Memória de Acesso Randômico

ROM – Memória apenas de leitura

EPROM - Memória somente de leitura programável e que se pode apagar

DETRAN-DF – Departamento de trânsito do Distrito Federal

DF – Distrito Federal

16F628A – Micro-Controlador PIC 16F628A

Kit2 – PicKit2 Gravador de Micro-controlador PIC

CAPÍTULO 1 – INTRODUÇÃO

1.1 – Introdução ao Tema Proposto

Novas formas de controle do tráfego estão sendo usadas em todo o mundo. Já se fala até em uma internet dos carros onde todos estarão conectados entre si por um sistema de controle distribuído, que dará vazão inteligente ao fluxo. Internet dos carros promete fim da irritação no trânsito CVIS (*Cooperative Vehicle-Infrastructure System*) (SITE: INOVAÇÃO TECNOLÓGICA, 2010) [1].

Vê-se nas inovações que vem surgindo, a busca de melhorias na forma com que interagimos com os sistemas, mas também encontrar sistemas mais eficazes capazes de executar as tarefas com menores chances de erros poupando tempo e recursos.

1.2 – Motivação

Ao dirigir em vias comerciais ou em algumas vias de acesso a universidades, pode-se constatar que muitas vezes, o semáforo é acionado para a parada de veículos, sem que haja pedestres para atravessar a via, impedindo desta forma o trânsito dos carros desnecessariamente.

Mesmo de madrugada quando raramente há pedestres encontra-se nestas vias os semáforos nas faixas de pedestre em vermelho aos veículos.

Em todo o mundo o transporte continua sendo um grande problema e no Brasil não é diferente, não há um transporte público eficiente na maioria de suas cidades. Dessa forma, as pessoas são obrigadas a adquirir seus próprios veículos, sejam eles carros ou motos. Como consequência, as ruas e estacionamentos tornam-se insuficientes para todo o fluxo existente, gerando morosidade, periculosidade e ineficiência. Na ausência de soluções definitivas, um grande desafio da administração pública será a melhoria do fluxo do trânsito.

> Outras considerações, tais como o tempo requerido para o cruzamento de pedestres e condições físicas na interseção, também afetam a sincronização do semáforo.

1. [1] Site Inovação Tecnológica (*http://www.inovacaotecnologica.com.br)*

> Após a seleção inicial de uma duração de ciclo..., devem-se fazer revisões frequentes e estudos do semáforo em operação, para obter a programação mais eficiente.

Como disposto no (DNER, 1971) citado acima, a duração do ciclo deve ser analisada caso a caso; e frequentes estudos devem ser realizados para garantir um sistema cada vez mais eficiente.

> As estatísticas de trânsito sofreram melhoras significativas durante os 14 anos de implementação da faixa, principalmente na diminuição dos atropelamentos, que era um de seus objetivos.
>
> Entretanto, ainda continuam ocorrendo acidentes fatais neste equipamento de segurança, demonstrando, cada vez mais, a importância da continuidade na implementação de melhorias na faixa de pedestre.

No (DETRAN-DF, Acidentes com Morte na Faixa de Pedestre, 2011) fala-se também na importância de melhorias continuas na implementação das faixas de pedestres.

Neste trabalho, é apresentada uma proposta para melhorar a vazão do trânsito de carros e pessoas, utilizando a automação para otimizar o uso das faixas de pedestres, de forma a permitir um fluxo seguro e a satisfação dos pedestres com o sistema utilizado, ao mesmo tempo que assegura o menor tempo possível à continuidade do fluxo de veículos.

1.3 – Objetivo Geral do Trabalho

Desenvolver um protótipo que tem como objetivo principal dar vazão ao tráfego evitando que o semáforo para pedestres seja acionado sem necessidade e que permaneça por muito tempo fechado aos veículos após a passagem dos pedestres.

1.4 – Objetivos Específicos

Fazer uso de uma interface no solo sensível ao pressionamento para garantir a presença do pedestre no local garantindo assim que o semáforo somente opere quando existir de fato alguém à espera de passagem. Ao concluir a travessia o pedestre acionará do outro lado da via outra interface que poderá liberar novamente o trânsito aos veículos permitindo assim uma economia no tempo em que o trânsito é interrompido.

1.5 – Metodologias

Visando o desenvolvimento deste projeto, múltiplas ideias foram surgindo e sendo adaptadas com o passar das pesquisas e do tempo; chegando a atual abordagem focada na eficiência e simplicidade do uso.

Após analisar o código de trânsito e algumas abordagens já em teste no Brasil, como mostrado na Figura 1.1 (Prefeitura de Vitória, 2008).

Figura 1.1 – Faixa da vida prefeitura de Vitória ES

Verifica-se a possibilidade de simplificar a primeira abordagem onde os botões ficariam como mostrado na Figura 1.2.

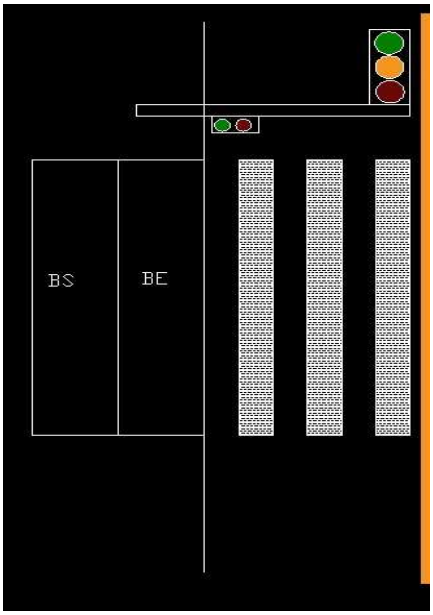


Figura 1.2 – Ideia inicial de topologia; determinar a direção do pedestre pela sequência de acionamento dos botões.

Alterando a configuração do posicionamento dos botões, criando como nas portas do metrô ou na faixa da Figura 1.1 sentidos de mão e contramão para a travessia da faixa de pedestres e com isso obter uma economia significativa no código fonte e no tempo de implementação.

Assim foi estabelecido no projeto o sentido de travessia dos pedestres que vão pela direita e voltam pela esquerda utilizando mão e contramão como no trânsito de veículos, assim se pode definir de forma mais simples quem está iniciando a travessia ou quem a está concluindo.

Na Figura 1.3 tem-se uma primeira visão da ideia proposta e da disposição dos elementos do projeto. BE representa o Botão de Entrada e BS o Botão de Saída. A forma, tamanho ou estrutura dos botões não serão abordadas nesse trabalho que tem foco no sistema proposto.

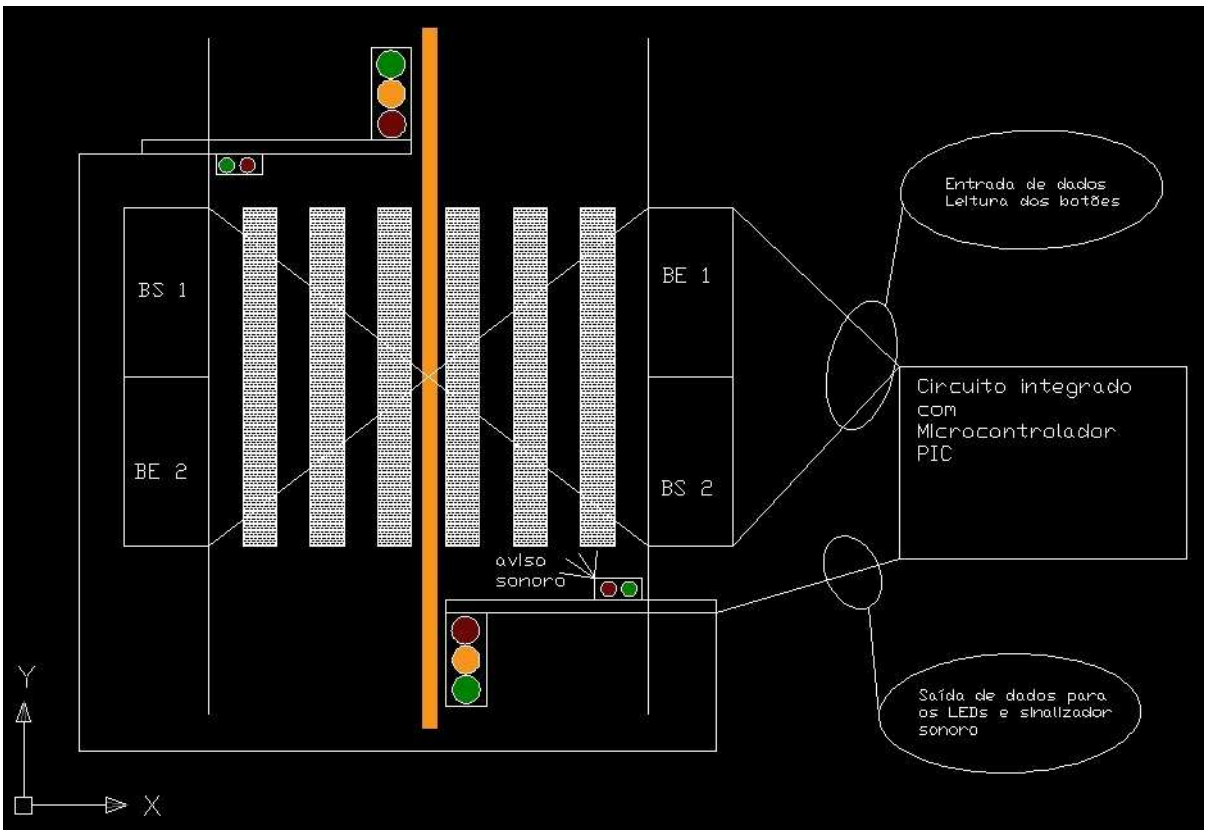


Figura 1.3 – Topologia geral

Esse projeto utiliza conteúdos de diversas disciplinas cursadas na Engenharia da Computação, como: Microcontroladores e Microprocessadores; Circuitos Eletrônicos; Arquitetura de Computadores; Linguagem e Técnicas de Programação I e II; Controle e Servomecanismo; Física I, II, III e IV; Mecânica; Lógica Digital; Desenho Técnico; Gerência de Projetos; sendo, portanto, indicador do conhecimento adquirido com as mesmas.

1.6 – Estrutura da Monografia

A estrutura desta monografia consiste de sete capítulos tratando os seguintes assuntos:

Capítulo 1 – Capítulo introdutório, onde é feita a apresentação do projeto, introdução ao tema proposto, a motivação do projeto, os objetivos, metodologias, topologia geral do projeto, e esta descrição da estrutura da monografia.

Capítulo 2 – Capítulo de apresentação do problema, sistema atual versus solução proposta de alteração na interface.

Capítulo 3 – Capítulo de Apresentação do Referencial Teórico, Abordagem ao Código Nacional de Trânsito e ao Micro-Controlador Usado.

Capítulo 4 – Capítulo de Descrição do *Hardware* e *Software*, O Micro-Controlador PIC16F268A, Kit de gravação PicKit2 e CCS C Compiler.

Capítulo 5 – Capítulo de Implementação, Modelagem do Sistema, Fluxograma geral do Projeto, Elaboração do Circuito, Elaboração e Gravação do Código Fonte, Montagem do Circuito e Montagem da Maquete.

Capítulo 6 – Capítulo dos Resultados Obtidos, Simulações e Problemas Encontrados.

Capítulo 7 – Considerações finais, Conclusão e Propostas para Trabalhos Futuros.

CAPÍTULO 2 – APRESENTAÇÃO DO PROBLEMA

Atualmente as faixas de pedestre que contam com sinalização semafórica, utilizam botões nos postes para a solicitação de passagem pelo pedestre. No sistema atual, caso o pedestre solicite a passagem através do botão no poste, o sistema somente responderá a sua solicitação após dois minutos da última vez que o trânsito de veículos foi obstruído (tempo do último ciclo), caso nenhuma solicitação de passagem seja feita, o sistema libera passagem aos pedestres a cada cinco minutos[2]. Facilmente se vê uma falha neste sistema, pois não garante que no momento da interrupção do trânsito o pedestre ainda se encontre a espera, o botão no poste pode não ser acessível a certos deficientes físicos em cadeiras de rodas elétricas e temos ainda a questão da higiene por ser um botão exposto ao publico pode transmitir doenças.

Tornar este sistema mais eficaz exigiu pensar na causa do problema e em seguida numa solução que fosse simples, viável e de baixo custo. O que encaminhou o foco do trabalho em um sistema que use outra interface homem - máquina.

Visando melhorar o funcionamento dos semáforos, tornando a interface mais inteligente e objetiva, facilitando o uso e aumentando a segurança do usuário, além de melhorar a fluidez do trânsito, propõe-se modificar a forma de acionamento do mesmo. No sistema atual, por muitas vezes o semáforo impede o trânsito de veículos, mesmo não tendo pedestres para atravessar a pista. O projeto aqui apresentado torna possível programar as atividades, fazendo com que as ações ocorram de maneira automática, fazendo uso de botões no solo para identificar a presença do pedestre na lateral da pista. Os botões estão presentes nos dois lados da pista e assim é possível identificar além da solicitação de passagem, a saída do pedestre da faixa.

[2] Dados obtidos em pesquisa de campo seguida de questionamento junto ao DETRAN-DF.

CAPÍTULO 3 – REFERENCIAL TEÓRICO

A elaboração desse projeto exigiu, além do conhecimento acadêmico e das pesquisas em *hardware* e *software*, uma análise do código de trânsito brasileiro, em busca das informações necessárias à sua execução, pois, qualquer sistema criado deve necessariamente se preocupar com a legalidade dos seus procedimentos, tornando-se dentro da legalidade um sistema aplicável.

3.1 – Código de Trânsito Brasileiro

Diz a lei 9.503/97 do código de trânsito Brasileiro, trecho referente à travessia de pedestres na faixa e em locais com semáforo (FRANCO, 2004).

> Art. 69. (...) III- (...) b - uma vez iniciada a travessia de uma pista, os pedestres não deverão aumentar o seu percurso, demorar-se ou parar sobre ela sem necessidade.
> Art. 70. Os pedestres que estiverem atravessando a via sobre as faixas delimitadas para este fim terão prioridade de passagem, exceto em locais com sinalização semafórica (...).
>
> Parágrafo único. Nos locais onde houver sinalização semafórica de controle de passagem será dada preferência aos pedestres que não tenham concluído a travessia, mesmo em caso de mudança do semáforo liberando a passagem dos veículos.
>
> Art. 71. O órgão ou entidade com circunscrição sobre a via manterá, obrigatoriamente, as faixas e passagens de pedestres em boas condições de visibilidade, higiene, segurança e sinalização.

A penalidade decorrente do não cumprimento do disposto nos artigos citados a seguir decorrentes do desrespeito para com pedestres no uso das faixas destinadas a eles correspondem a infrações gravíssimas; como dito no (HONORATO,1998).

> Art. 214. Deixar de dar preferência de passagem a pedestre e a veículo não motorizado:
>
> I – que se encontre na faixa a ele destinada;
>
> II – que não haja concluído a travessia mesmo que ocorra sinal verde para o veículo;
>
> III – portadores de deficiência física, crianças, idosos e gestantes:
>
> Infração – gravíssima;
>
> Penalidade – multa.

Tendo a sua disposição um sistema eficaz pretende-se que o pedestre o use corretamente com a consciência de não atravessar a pista sem aguardar sua vez, pois, além de não demorar, ele fará uma travessia segura.

3.2 – Micro-Controlador PIC

O PIC é fabricado pela Microchip Technology Inc. (http://www.microchip.com); que iniciou seus negócios com o Brasil em 1990 selecionando a Artimar Ltda como parceira e representante exclusiva dos produtos no mercado nacional.

Temos no (SOUZA, 2008) uma boa definição do que é um micro-controlador, parte desta definição está descrita a seguir.

> Em poucas palavras poderíamos definir o microcontrolador como um "pequeno" componente eletrônico, dotado de uma "inteligência" programável, utilizado no controle de processos lógicos.
>
> O controle de processos deve ser entendido como o controle de periféricos, tais como: LEDs, Botões, displays de segmentos, displays de cristal líquido (LDC), resistências, relês, sensores diversos (pressão, temperatura, etc.) e muitos outros.
>
> São chamados de controles lógicos, pois a operação do sistema baseia-se nas ações lógicas que devem ser executadas, dependendo do estado dos periféricos de entrada e/ou saída.
>
> "pequeno", pois em uma única pastilha de silício encapsulada (popularmente chamada de CI ou CHIP), temos todos os componentes necessários ao controle de um processo.

Sendo um micro-controlador, contém internamente um circuito completo com memórias voláteis e não-voláteis, processamento, *timer*, portas de entrada e saída entre outras funcionalidades, sendo utilizados em aplicações que não necessitam armazenar grandes quantidades de dados, acrescentando automação à aplicação.

3.3 – Linguagem de Programação C

A opção por uso da linguagem C se fez por afinidade do autor com esta linguagem e após conclusão de que para o referido projeto usar outra linguagem como o Assembly geraria um código muito maior por não ter a mesma flexibilidade da linguagem C. Mesmo sendo uma linguagem de mais alto nível, não oferece perda significativa na velocidade de processamento para este projeto.

CAPÍTULO 4 – DESCRIÇÃO DO HARDWARE E SOFTWARE

Neste capitulo tem-se uma descrição dos principais componentes de *hardware* e os programas utilizados para elaboração e gravação do *software* no micro-controlador.

4.1 – Explorando o Micro-Controlador PIC 16F628A

O micro-controlador PIC 16F268A, mostrado na Figura 4.1 e descrito a seguir.

Figura 4.1 – PIC 16F628A

Suas principais características são:

- CPU RISC de alta performance:
- Possui 18 pinos;
- Velocidades de operação em DC de 20 MHz;
- Ampla faixa de tensão de operação. (2.0 - 5.5V);
- Capacidade de interrupção;
- Pilha de oito níveis para chamadas de desvios;
- Modos de endereçamentos diretos e indiretos;
- SET (conjunto) de 35 instruções;
- Instruções de ciclo único, exceto as que causam desvios;
- Oscilador interno de 4Mhz calibrado de fábrica para ± 1%;
- Memória de programa flash de 2048 *words* (32 bits);
- 224 bytes de memória RAM para dados;

- 128 bytes de memória EEPROM para dados;

- Instruções de 14 bits com 200ns de tempo de execução;

- Dados de oito bits por endereço de memória;

- 15 registradores especiais;

- 16 portas que podem ser configuradas para entrada e/ou saída;

- Modo *SLEEP* (dormindo, poupança de energia);

- *Pull-ups* programáveis no PORTB;

- Comparador interno;

- Outras características especiais como programação *in-circuit* serial, proteção por código, *watchdog timer*, módulo CCP, USART,...

4.1.1 – Diagrama de blocos, arquitetura interna

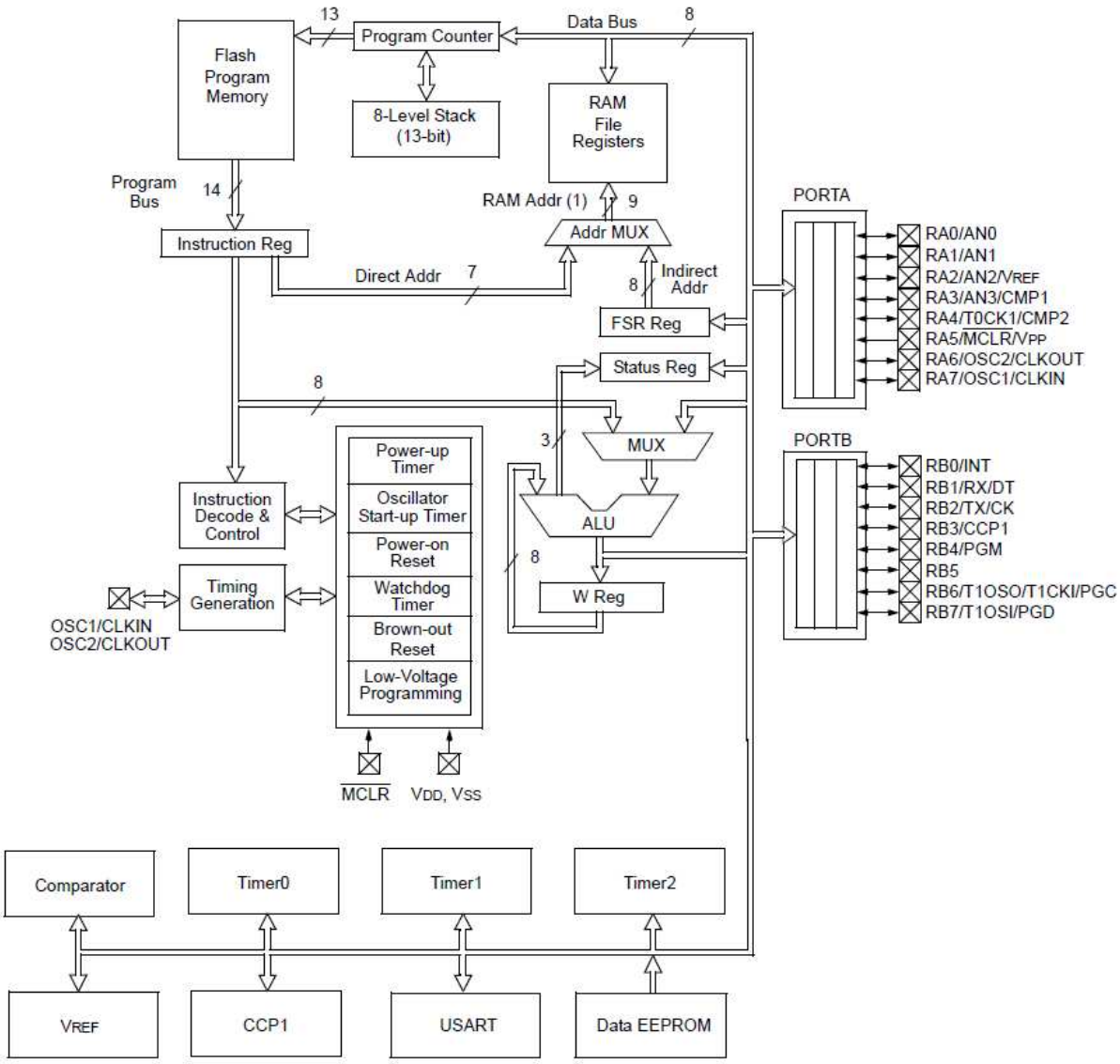


Figura 4.2 – Diagrama de blocos, arquitetura interna

4.1.2 – Pinagem

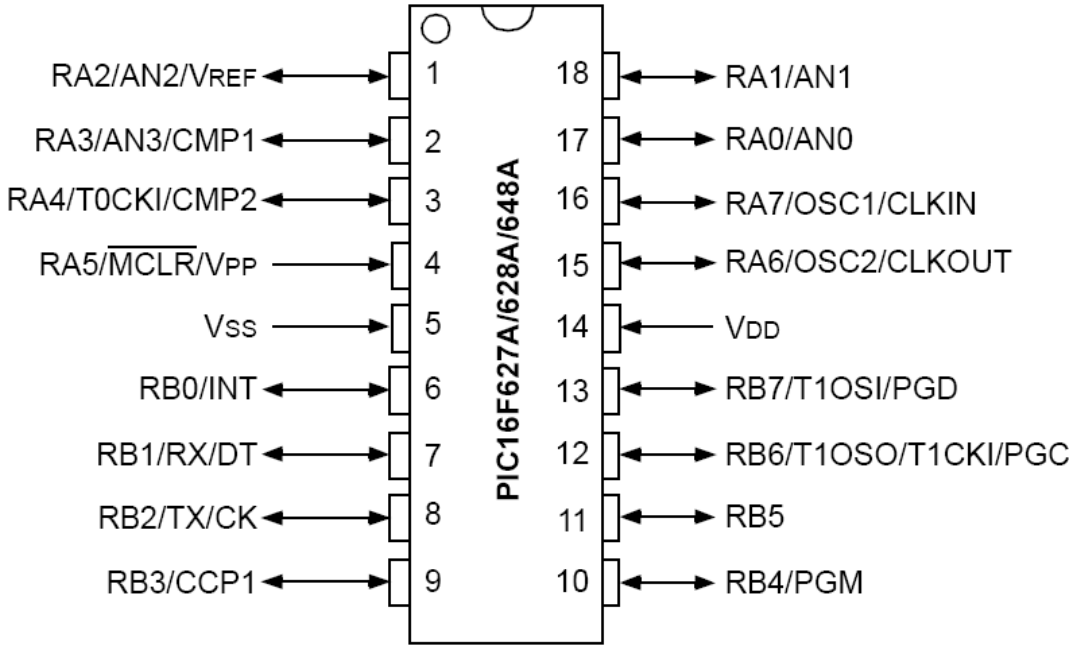


Figura 4.3 – Pinagem PIC 16F628A

No próximo capítulo nas figuras 5.2 e 5.3 é possível ver quais pinos do micro-controlador foram usados neste projeto.

No Quadro 4.1 temos a descrição das funções de cada pino.

Quadro 4.1 – Descrição da pinagem do PIC 16F628A

Pino	Função	Tipo	Descrição
1	RA2/AN2/Vref	Entrada/Saída	PORTA bit 2 / Entrada do comparador analógico / Saída da tensão de referência
2	RA3/AN3/CMP1	Entrada/Saída	PORTA bit 3 / Entrada do comprador analógico / Saída comparador 1
3	RA4/T0CKI/ CMP1	Entrada/Saída	PORTA bit 4 / Entrada de *clock* externo do *timer* 0 / saída do comparador 2. *Esse pino possui saída com **dreno aberto***
4	RA5/MCLR/VPP	Saída	PORTA bit 5 / Reset CPU / Tensão de programação
5	VSS	Alimentação	Terra (negativo)
6	RB0/INT	Entrada/Saída	PORTB bit 0 / entrada de interrupção externa
7	RB1/RX/DT	Entrada/Saída	PORTB bit 1 / Recepção USART (modo assíncrono) / Dados (modo síncrono)
8	RB2/TX/CK	Entrada/Saída	PORTB bit 2 / Transmissão USART (modo assíncrono) / *Clock* (modo síncrono)
9	RB3/CCP1	Entrada/Saída	PORTB bit 3 / Entrada ou saída do módulo CCP
10	RB4/PGM	Entrada/Saída	PORTB bit 4 / Entrada de programação **LVP ***
11	RB5	Entrada/Saída	PORTB bit 5
12	RB6/T1OSO/ T1CKI/PGC	Entrada/Saída	PORTB bit 6 / Entrada do oscilador do TMR1 / Entrada de *clock* do TMR1 / *Clock* na programação ICSP*
13	RB7/T1OSI/PGD	Entrada/Saída	PORTB bit 7 / Entrada do oscilador do TMR1 / Dados na programação ICSP
14	VDD	Alimentação	Alimentação positiva (2.0V a 5.5V)

15	RA6/OSC2/ CLKOUT	Entrada/Saída	PORTA bit 6 / Entrada para cristal oscilador / Saída de *clock*
16	RA7/OSC2/ CLKIN	Entrada/Saída	PORTA bit 7 / Entrada para cristal oscilador / Entrada de *clock* externo
17	RA0/AN0	Entrada/Saída	PORTA bit 0 / Entrada do comparador analógico
18	RA1/AN1	Entrada/Saída	PORTA bit 1 / Entrada do comparador analógico

***LPV** – Baixa tensão de programação.

*** Dreno aberto** – Uma fonte de alimentação externa deve ser aplicada ao pino.

***ICSP** – Programação *in-circuit*.

4.1.3 – Organização da memória

No mapa de memória RAM do PIC16F628A, se localizam os registradores de uso especial (SFR) e de propósito geral (GPR) e seus respectivos endereços como apresentado na Figura 4.4 a seguir.

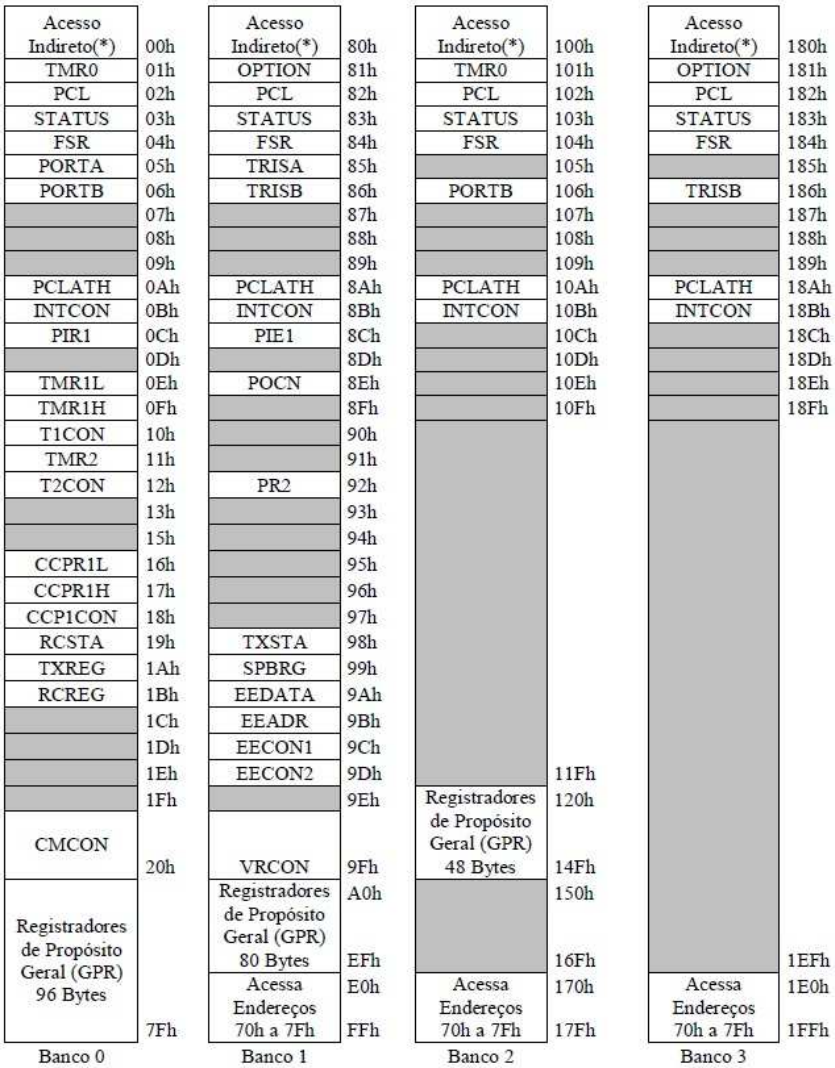


Figura 4.4 – Mapa de memória RAM do PIC16F628A

O PIC 16F628A possui memória de programa do tipo FLASH. Esse tipo de memória permite um mínimo de 1000 ciclos de gravação/apagamento das informações, sendo indicada tanto para a fase de desenvolvimento e teste quanto para aplicações práticas. Os dispositivos FLASH são indicados pela letra "F". A seguir na Figura 4.5 temos o mapa da memória de programa do PIC 16F628A.

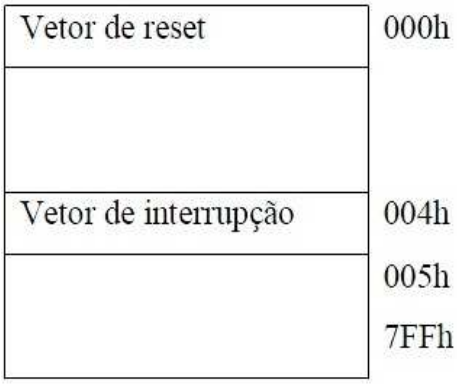


Figura 4.5 – Memória de programa

4.2 – Kit de Gravação PICKIT 2

Trata-se de uma ferramenta para intermediar a comunicação do computador com o micro-controlador tornando simples a gravação do programa a ser executado no micro-controlador.

Figura 4.6 – PICKit 2

Composto de um cabo USB, um circuito micro-controlado e um conector do tipo ICSP *(In-Circuit Serial Programming*) capaz de programar via serial os micro-controladores PIC conforme mostrado na Figura 4.6. O kit contém também um CD com Software que identifica o micro-controlador e faz a gravação do programa através de uma porta USB.

4.3 – CCS C Compiler

Este programa oferece ferramentas para o desenvolvimento e depuração de programas embutidos e em execução nos micro-controladores PIC. O CCS C Compliler foi utilizado para a escrita e compilação do código fonte desenvolvido na linguagem C, detalhado no próximo capítulo.

O programa compila códigos gravados com a extensão '.c'. Após compilação são gerados oito arquivos com o mesmo nome no mesmo local do arquivo '.c', com as seguintes extensões: '.cof', '.err', '.hex', '.lst', '.pjt', '.sta', '.sym', '.tre'. O programa além de compilar, mostra avisos e erros e a previsão do uso da memória RAM e ROM do micro-controlador. Após a compilação, o arquivo '.hex' será utilizado na simulação no programa PROTEUS, e na gravação do micro-controlador PIC.

O CCS C Compiler é um programa desenvolvido pela empresa *Custom Computer Service* (CCS), e está disponível para ser baixado no site: (http://www.ccsinfo.com).

CAPÍTULO 5 – IMPLEMENTAÇÃO

A implementação do projeto foi dividida em etapas que vão auxiliar no entendimento do todo.

São elas:

- Modelagem do sistema;
- Elaboração dos circuitos;
- Montagem dos circuitos;
- Elaboração do código fonte para o micro-controlador PIC;
- Montagem da maquete;

5.1 – Modelagem do Sistema

Como visto na topologia geral do projeto (Figura 1.3), a apresentação do projeto envolve o uso de uma maquete para simular o funcionamento da faixa de pedestres. Na apresentação serão utilizados botões para simular a presença dos pedestres nas laterais da via. Os resultados desta interação podem ser vistos na maquete através das luzes dos semáforos e na sinalização sonora.

Os botões de entrada e saída foram dispostos um ao lado do outro paralelamente à via. Partindo do princípio disposto na legislação de que o pedestre ao efetuar a travessia deve fazê-la em sentido perpendicular ao eixo da via, não devendo aumentar seu percurso, demorar-se ou parar sobre a via sem necessidade.

> A faixa de pedestre é respeitada no Distrito Federal. Após os 14 anos da implementação da faixa, a pesquisa apontou que, na maioria dos casos, em torno de 80% das travessias, o condutor parou o veículo para o pedestre atravessar.
> Apesar da comprovada eficiência das faixas para travessia de pedestres, as quais contribuem para sua autoestima e garantia de seus direitos constitucionais como cidadão, há que se priorizar as atividades de educação, além das de fiscalização e de engenharia, de forma a se buscar a situação ideal de respeito pela totalidade dos condutores de veículos.

A legislação pode não ser sempre cumprida como mencionado acima na (DETRAN-DF, Pesquisa de Respeito à Faixa de Pedestre, 2011), porem campanhas educativas podem fazer com que os pedestres usem de forma correta um sistema que eles julgam eficaz. Atualmente

existe em Brasília e em algumas outras cidades uma grande aceitação e respeito às faixas de pedestre devido a campanhas educativas que inseriram uma consciência tanto nos motoristas quanto nos pedestres.

A construção da maquete foi feita em uma base de isopor, onde a topologia pode ser claramente vista, com a faixa de pedestre, os semáforos, a via, os botões e ao lado o circuito micro-controlado construído.

5.1.1 – Fluxograma geral do sistema

O sistema fica ligado e em espera até que o pedestre fique parado sobre o botão de entrada por cinco segundos, então o sistema verifica quanto tempo se passou desde a última vez que o trânsito foi interrompido. Caso tenham se passado sessenta segundos da última liberação do tráfego, o sistema interrompe novamente o trânsito para a passagem do pedestre. No momento em que o pedestre alcança o outro lado da via e pressiona o botão de saída, gera-se uma nova interrupção no sistema que irá liberar novamente o fluxo de veículos. Caso aconteça de outro pedestre pressionar o botão de entrada antes de o primeiro pressionar o botão de saída as interrupções são bloqueadas e o sistema adiciona mais dez segundos antes de iniciar a sequência de liberação do trânsito aos veículos. Caso nenhum botão seja pressionado, depois de quinze segundos a sequência de liberação do trânsito é iniciada automaticamente.
No fluxograma Figura 5.1 são mostrados todos os possíveis caminhos do sistema e as ações tomadas de acordo com cada hipótese de uso.

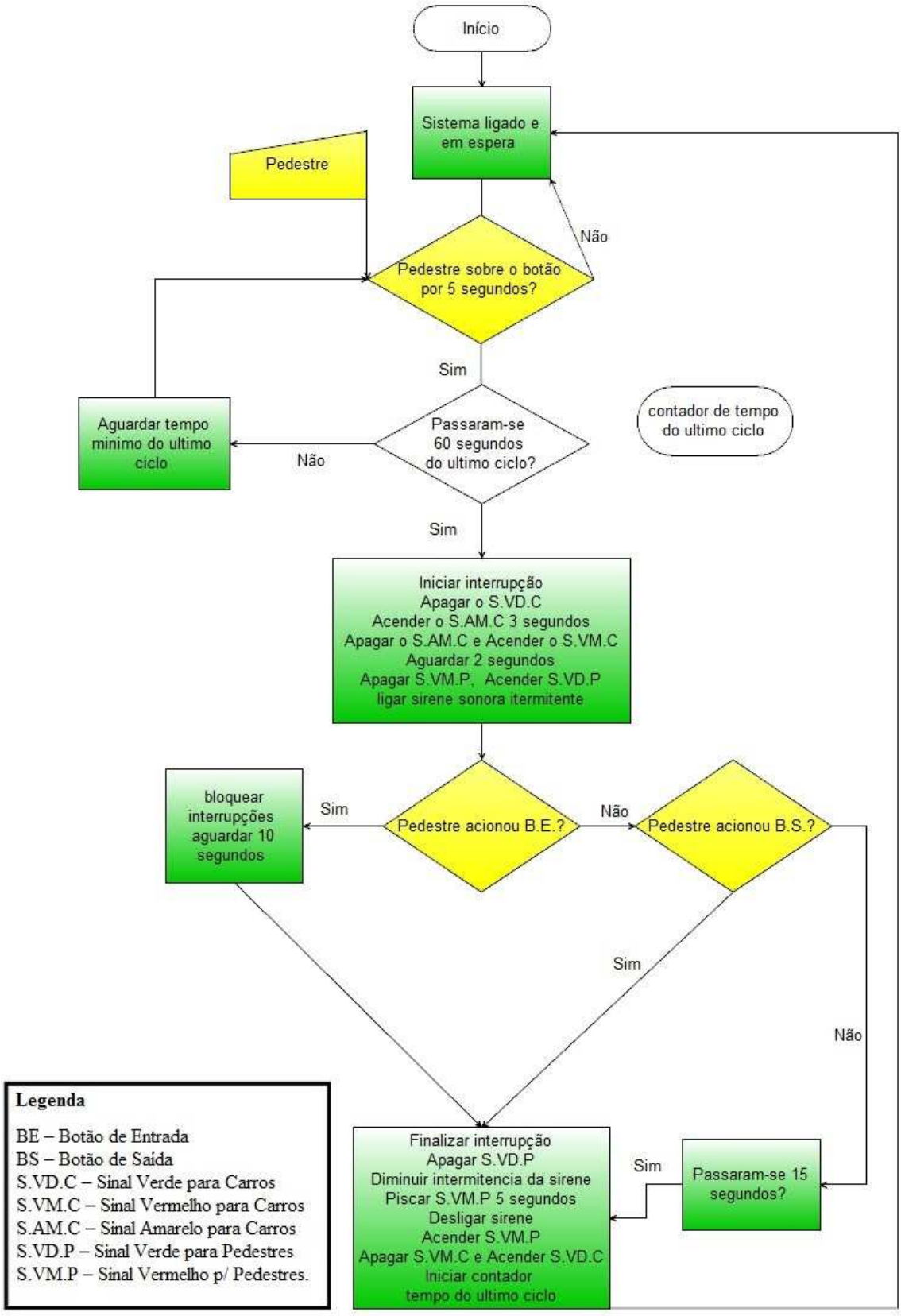


Figura 5.1 – Fluxograma do projeto

5.2 – Elaboração do Circuito

Para realização desse projeto adotou-se a utilização de um micro-controlador PIC dispositivo inteligente estudado no curso que suporta diferentes linguagens de programação além de possuir diversas características que o torna extremamente eficiente para inúmeros usos.

5.2.1 – PROTEUS ISIS 7 Professional

Depois de elaborado o fluxograma pode-se dar início à elaboração do circuito virtual utilizando o programa Proteus ISIS 7 Profissional, no qual foi possível escolher os componentes e definir como seriam ligados ao micro-controlador.

A Figura 5.2 mostra a ilustração do circuito virtual onde foram feitos os primeiros testes.

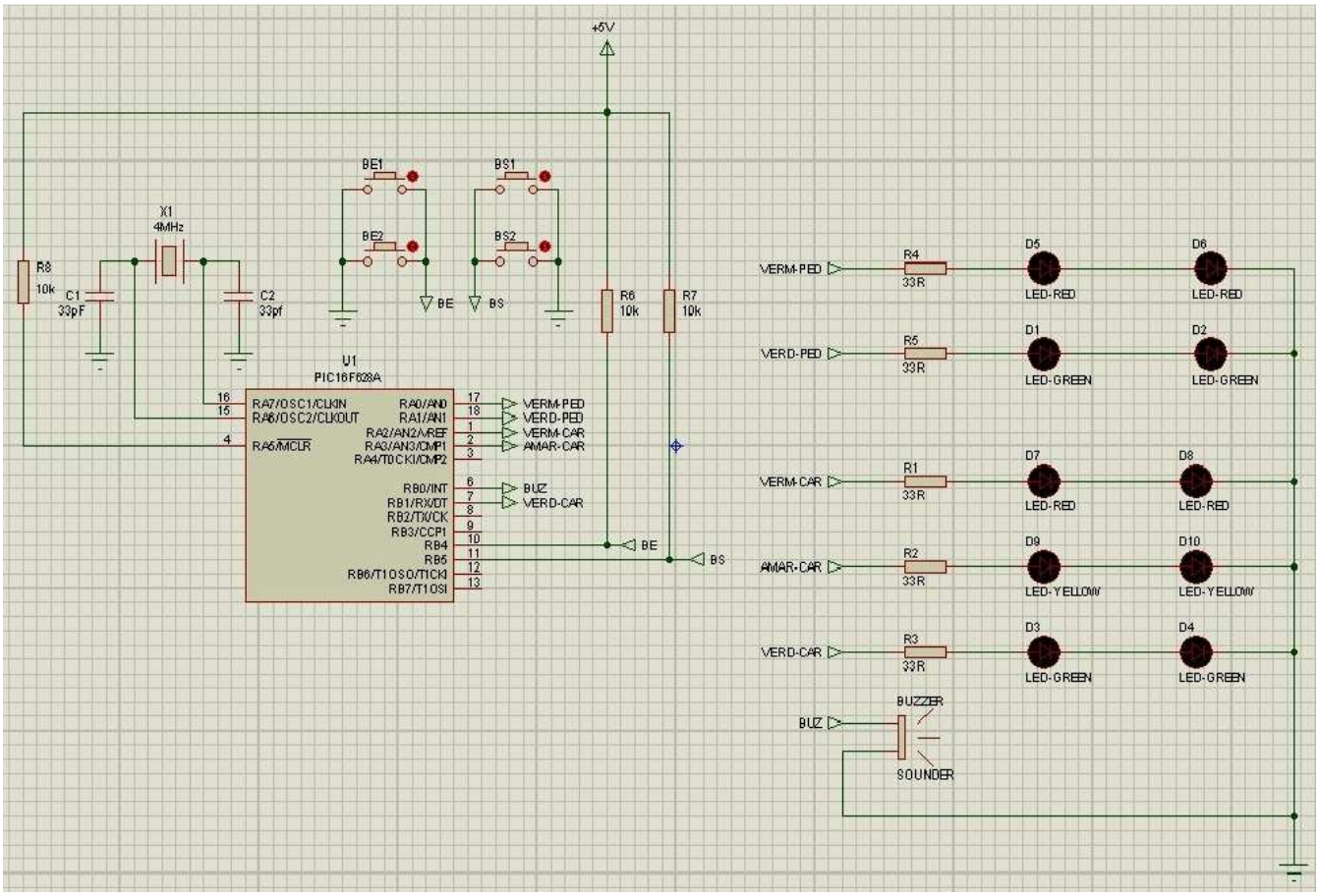
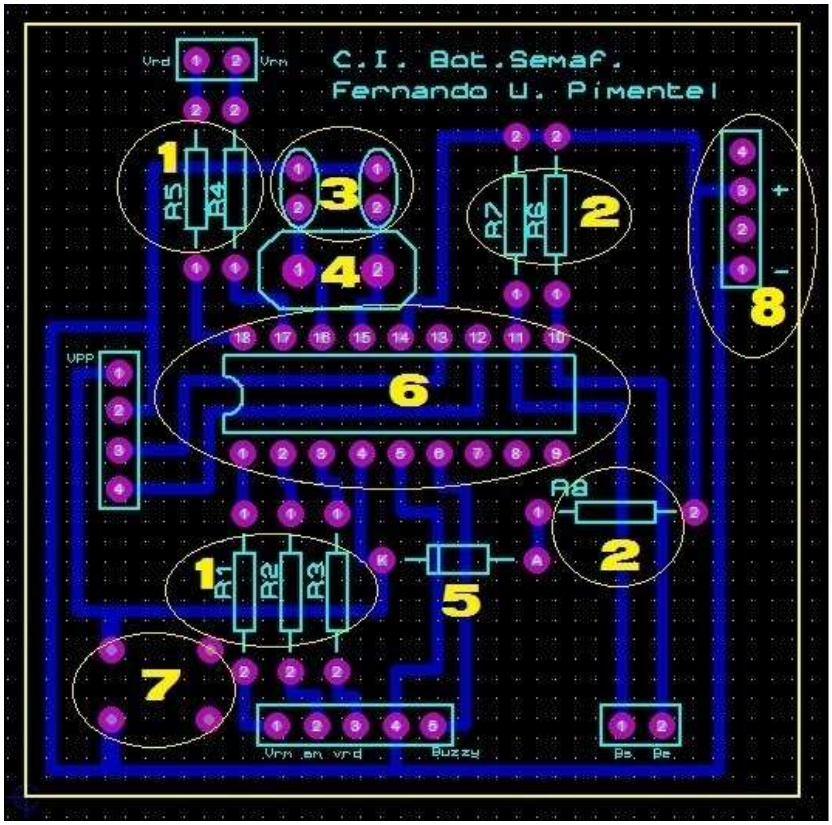


Figura 5.2 – Circuito virtual ISIS Proteus

5.2.2 – PROTEUS ARES 7 Professional

Os componentes utilizados neste projeto e suas ligações físicas são mostrados na Figura 5.3 do circuito construído no ARES Proteus 7 acrescido da legenda.



1 - Resistores de 220R;
2 - Resistores de 10K;
3 - Capacitores de 33pF;
4 - Cristal Ocilador de 4MHz;
5 - Diodo;
6 - Micro-Controlador;
7 - Botão Reset;
8 - Barra de pinos;

Figura 5.3 – Circuito ARES Proteus, componentes

5.3 – Elaboração, Descrição, Compilação e Gravação do Código Fonte

A linguagem C foi a usada no projeto para controle dos semáforos e interpretação das leituras feitas pelos botões. Através de sua lógica tomará as decisões sequenciais estabelecidas no programa inserido no micro-controlador. Como visto no capítulo anterior para escrita do programa e compilação do código foi utilizado o CCS C Compiler e para gravação do código no micro-controlador foi usado o kit de gravação PICKit 2 da Robótica Simples.

5.3.1 – Descrição do código fonte

É apresentado a seguir uma sucinta descrição do código fonte e suas etapas.

A Figura 5.4 mostra o cabeçalho do código, definição do micro-controlador, do *clock* usado alem das definições das variáveis que estão associadas a cada pino usado.

```
1
2  /*************************************Projeto Semáforo*************************************************
3
4  Nome do Arquivo: projeto semaforo.c
5  Versão: 1.0
6  Descrição:
7
8  Autor:
9  Compilador: PIC COMPILER Versão 3.43.
10 Ambiente de simulação: Proteus 7.7 SP2 toolsuite ISIS Professional.
11 Microcontrolador utilizado: PIC16F628A.
12 Data:
13
14 ******************************************************************************************************/
15
16 #include <16f628a.h>   //Inclui os arquivos com as definições do PIC16F628A.
17 #fuses XT, NOWDT, NOPROTECT, PUT, BROWNOUT, MCLR
18 #use delay(clock=4000000)   //Cristal oscilador = 4MHz
19
20 /*********************************Definições dos pinos no PIC*********************************/
21
22 #define   Sem_Verm_Ped   PIN_A0    //Pino de saida p/ o Semáforo Vermelho de Pedestre.
23 #define   Sem_Verd_Ped   PIN_A1   //Pino de saida p/ o Semáforo Verde de Pedestre.
24 #define   Sem_Verm_Car   PIN_A2   //Pino de saida p/ o Semáforo Amarelo dos Carros.
25 #define   Sem_Amar_Car   PIN_A3   //Pino de saida p/ o Semáforo Amarelo dos Carros.
26 #define   Sem_Verd_Car   PIN_B1   //Pino de saida p/ o Semáforo Verde dos Carros.
27 #define   Buzzer         PIN_B0   //Pino de saida p/ o Buzzer.
28 #define   Bot_BE         PIN_B4   //Pino de entrada p/ os push button BE1 e BE2.
29 #define   Bot_BS         PIN_B5   //Pino de entrada p/ os push button BS1 e BS2.
30
```

Figura 5.4 – Cabeçalho e definição das variáveis associadas aos pinos

Na Figura 5.5 estão as variáveis definidas na memora RAM.

```
30
31 /*********************************Definições das variáveis na RAM*******************************/
32
33 int Cont_Fim=00; //Declara e inicializa a variável p/ timer = 5s rotina de saida
34 int Conta_Seg=00; //Declara e inicializa a variável p/ timer = 15s.
35 int Cont_Be=00; //Declara e inicializa a variável p/ timer = 5s rotina de espera do pedestre sobre o BE
36 int Cont_Sesse=00; //Declara e inicializa a variável p/ timer = 60s rotina de espera do ultimo ciclo
37 int Block=00; //Declara e inicializa a variável p/ bloqueio dos botoes
38
```

Figura 5.5 – Variáveis na memora RAM

Na Figura 5.6 se encontra a função de interrupção final, momento em que o semáforo executa os comandos finais, piscando a luz vermelha aos pedestres e reduzindo a intermitência do sinal sonoro, antes de retornar ao estado inicial do sistema.

```
/************************************ Função Interrupção ***************************************/

void Fim()
{
   int Cont_Fim=00;
      while(true)   //Enquanto verdadeiro...
      {
        Cont_Fim --;
          if (Cont_Fim > 4)
             {
               Cont_Fim = 4;
             }
            output_low(Sem_Verd_Ped); // durante este tempo o led vermelho pisca e aciona o Buzzer intermitentemente.
            output_high(Sem_Verm_Ped);// dura 1 segundo que é o tempo do loop
            output_high(Buzzer);
            delay_ms(250);
            output_low(Sem_Verm_Ped);
            output_low(Buzzer);
            delay_ms(250);
            output_high(Sem_Verm_Ped);
            output_high(Buzzer);
            delay_ms(250);
            output_low(Sem_Verm_Ped);
            output_low(Buzzer);
            delay_ms(250);
            output_high(Sem_Verm_Ped);
          if(Cont_Fim == 0)
             {
                output_low(Sem_Verm_Car);
                output_high(Sem_Verd_Car);
               return;
             }
      }
}
```

Figura 5.6 – Função FIM

Na Figura 5.7 é mostrado o temporizador de quinze segundos no qual o trânsito de veículos é interrompido e o dos pedestres é liberado. Primeiramente, executa-se a sequência de interrupção do trafego e em seguida a liberação da via aos pedestres acionando também a sinalização sonora. Como visto no Fluxograma, caso outro pedestre acione o BE (botão de entrada), o contador de tempo é acrescido em dez segundos e as interrupções são bloqueadas; ou seja, este procedimento só pode ser feito uma única vez. Caso algum pedestre acione o BS (botão de saída) antes que outro acione o botão de entrada, a função FIM vista anteriormente é chamada. Caso nenhum botão seja pressionado após quinze segundos chama-se automaticamente a função FIM.

```
/********************************** Função temporizador 15 segundos ****************************/

void Tempo_15()
{
  int Conta_Seg=00; //Declara e inicializa a variável p/ timer = 15s.
  int Block=00; //Declara e inicializa a variável p/ bloqueio dos botoes
   output_low(Sem_Verd_Car);   //Semáforo verde dos carros apagado.
   output_high(Sem_Amar_Car);   //Aciona semáforo amarelo dos carros.
   delay_ms(3000);   //Aguarda 3 segundos.

   output_low(Sem_Amar_Car);   //Semáforo amarelo dos carros apagado.
   output_high(Sem_Verm_Car);   //Aciona semáforo vermelho dos carros.
   delay_ms(2000);   //Aguarda 2 segundos.

   output_low(Sem_Verm_Ped);   //Semáforo vermelho de pedestre apagado.
   output_high(Sem_Verd_Ped);   //Aciona semáforo verde de pedestre.
   delay_ms(100);   //Tempo p/ estabilização dos pinos do microcontrolador.

    while(true)
    {
       Conta_Seg --;
       if (Conta_Seg > 14)
           {
             Conta_Seg = 14;
           }
        if(!input(Bot_BE) && Block == 0){
           Conta_Seg = Conta_Seg + 9;
           Block = 1; // bloquear botões
        }
        if(!input(Bot_BS) && Block == 0){
           Fim(); // chamar interrupçao de fechamento
           return;   //retorna.
        }
             output_high(Buzzer); //durante este tempo de 1 segundo o Buzzer é ligado e desligado intermitentemente.
             delay_ms(200);
             output_low(Buzzer);
             delay_ms(150);
             output_high(Buzzer);
             delay_ms(200);
             output_low(Buzzer);
             delay_ms(150);
             output_high(Buzzer);
             delay_ms(200);
             output_low(Buzzer);
             delay_ms(100);

        if (Conta_Seg == 0)   //Se valor de Conta_Seg = 0 ...
           {
              Fim(); // chamar interrupçao de fechamento
              return;   //retorna.
           }
        //Decrementa segundos.
    }
}
```

Figura 5.7 – Função TEMPO_15 travessia dos pedestres

A Figura 5.8 mostra o código relativo à solicitação de passagem do pedestre, onde o mesmo deve permanecer sobre o botão de entrada por cinco segundos antes que seja encaminhada interrupção chamando a função TEMPO_15 vista anteriormente. No *While* enquanto o botão estiver pressionado conta-se cinco segundos e então chama-se a função seguinte.

```
/************************************ Função BotaoBE 5s ***************************************/
void BotaoBE()
{
int Cont_Be=00; //Declara e inicializa a variável p/ timer = 5s.
   while(!input(Bot_BE)) // Verifica se o BE está acionado
   {
    Cont_Be --;   //Decrementa segundos.
      if (Cont_Be > 4)
       {
         Cont_Be = 4;
       }
       if (Cont_Be == 0)   //Se valor de Cont_Cinc = 0 ...
          {
            Tempo_15();
            return;   //retorna.
          }
     delay_ms(1000);   //Decrementa contador de segundos a cada 1s.
    }
}
```

Figura 5.8 – Função BOTAOBE

Já a Figura 5.9 descreve o contador de tempo do último ciclo, é um timer de sessenta segundos que toda vez que o programa é iniciado ou reiniciado conta este tempo antes de ser possível interromper novamente o trânsito. Com este *timer* é possível configurar o sistema para não interromper o trânsito toda vez que um pedestre ficar sobre o botão de entrada, pois terá que aguardar o fim deste tempo para uma nova interrupção, sendo muito útil nos casos em que há um fluxo muito grande de pedestres no local. Este *timer* foi alterado para dez segundos a fim de facilitar a visualização e a apresentação do programa em funcionamento.

```
/************************************ Função Timer_60 ***************************************/

void Timer_60()
{
int Cont_Sesse=00; //Declara e inicializa a variável p/ timer = 60s
while(true)   //Enquanto verdadeiro...
{
   Cont_Sesse --;   //Decrementa segundos.
   if (Cont_Sesse > 9)
    {
      Cont_Sesse = 9;
    }
      if(Cont_Sesse == 0)   //Se valor de Cont_Seg = 0 ...
      {
         BotaoBE();
         return;
      } //fim do if disparando a interrupção
   delay_ms(1000);   //Decrementa contador de segundos a cada 1s.
}  //fim do while
}  //fim void Semaforo.
```

Figura 5.9 – Função TIMER_60

Por fim na Figura 5.10 o programa principal onde se inicia o sistema com o semáforo verde aos veículos e vermelho aos pedestres. Chamando em seguida a função TIMER_60 que dará o tempo mínimo do último ciclo.

```
169  /************************************Programa principal*****************************************/
170
171  void main()
172  {
173     do
174     {
175        output_low(Sem_Verm_Car);
176        output_low(Sem_Amar_Car);
177        output_low(Buzzer);   //Desliga Buzzer.
178        output_low(Sem_Verd_Ped);
179        output_high(Sem_Verm_Ped);   //Semáforo vermelho pedestre aceso.
180        output_high(Sem_Verd_Car);   //Semáforo verde carros aceso.
181
182              Timer_60();
183
184     }while(1);
185  }
186
187  /*************************************************************************************************/
```

Figura 5.10 – Função MAIN

5.3.2 – Compilação e gravação do código fonte

Com o código já escrito o próximo passo é sua compilação, para isso tem-se na aba Compiler a opção Compiler ou pode-se usar a tecla de atalho F9 no programa CCS C Compiler. Neste procedimento são gerados oito arquivos com o mesmo nome porem extensões diferentes entre eles o '.hex' que será usado no simulador do ISIS Proteus e no PICKit2 para a gravação no micro-controlador. Após o reconhecimento do micro-controlador pelo programa do PICKit2, deve-se importar o arquivo HEX na aba FILE a opção IMPORT HEX, onde busca-se o arquivo a ser gravado no micro-controlador PIC. Na Figura 5.11 é mostrada uma visão do programa ao importar o arquivo Hex e da sua gravação no micro-controlador.

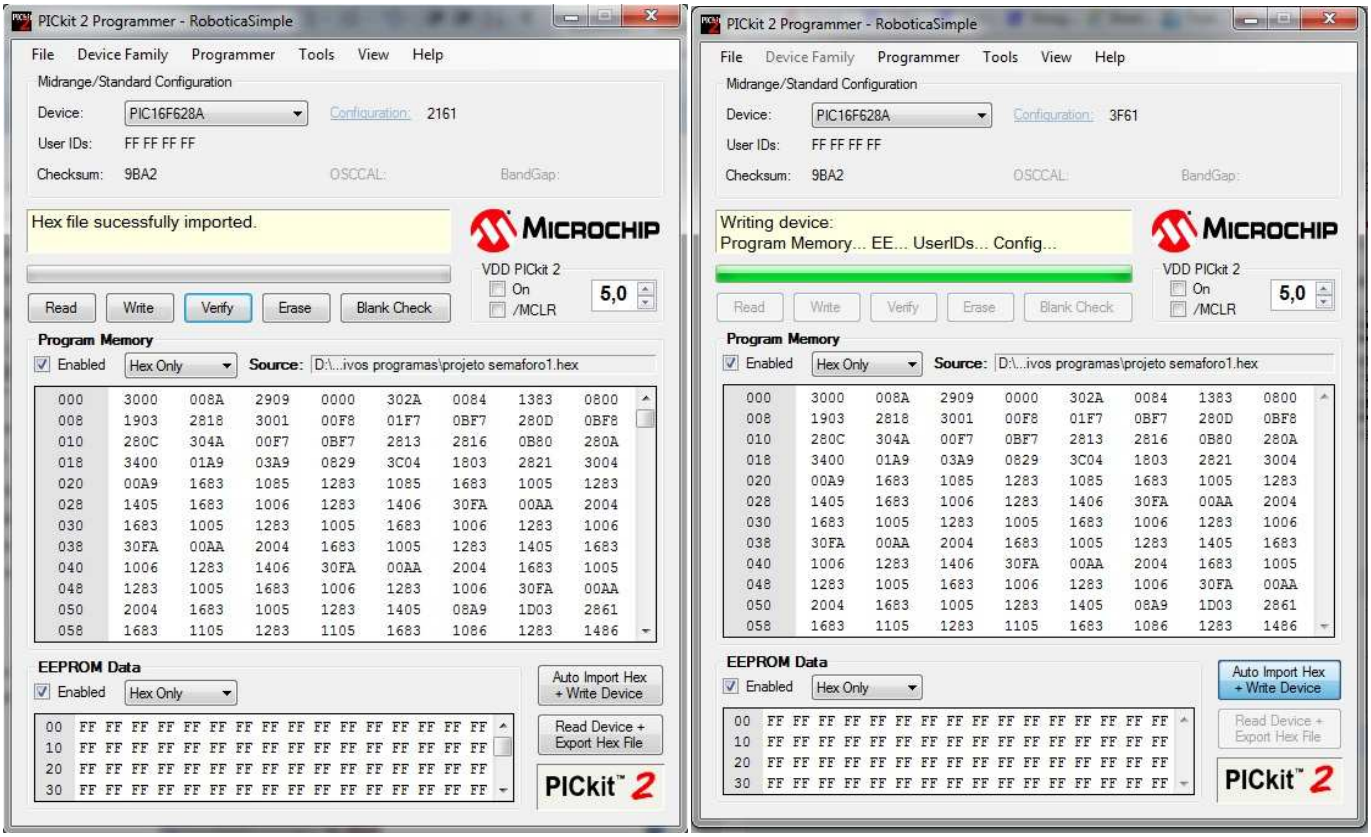


Figura 5.11 – PICKit2 importação e gravação do .HEX

5.4 – Montagem do Circuito

Após elaborado o circuito utilizando o Proteus, o passo seguinte foi sua construção física e para isso utilizou-se uma placa de fenolite cobreada de face única, e os componentes descritos anteriormente neste capítulo na Figura 5.3 (p. 32).

Na Figura 5.12 são mostradas as etapas da confecção do circuito físico deste projeto feito sobre a placa de fenolite cobreada de face única.

Figura 5.12 – Confecção do circuito impresso

O primeiro passo é imprimir preferencialmente em papel fotográfico o circuito criado no ARES Proteus, em seguida após limpar devidamente a face de cobre da placa com uma esponja de aço, coloca-se sobre esta face o circuito impresso, aplica-se calor com um ferro de passar roupas, fazendo com que o circuito impresso deixe o papel e se fixe sobre o cobre. Em seguida coloca-se a placa em água com sabão e cuidadosamente esfregar com os dedos para remover o papel restando somente o circuito impresso na placa de cobre. Após remover todo o papel verifica-se se o circuito está intacto caso ocorra alguma imperfeição nas trilhas deve-se cobri-las com uma caneta de escrever em CDs para que nestes locais o ácido não atue. Feito isso a placa é mergulhada em uma solução de água com percloreto de Ferro onde todo cobre exposto será corroído restando somente as trilhas impressas na placa. Aplica-se tiner para remover a tinta das trilhas e agora o circuito está pronto para ser montado; Furos são feitos onde entrarão os componentes e após sua colocação no lugar serão soldados e testados.

Para evitar que as partes de cobre fiquem expostas à corrosão é bom aplicar sobre a placa já soldada e testada um verniz ou tinta evitando o contado do cobre com o oxigênio.

5.5 – Montagem da Maquete

Buscando melhorar a aparência da apresentação levando-a ao foco de sua utilização prática foi montada uma maquete em isopor para demonstração do funcionamento do sistema com uma visão do todo. Como visto na topologia do projeto Figura 1.3 (p.16) que relata uma visão geral de como foi idealizado o projeto, agora aqui se vê a maquete na Figura 5.13 que representa mais claramente a proposta e mostra a solução aplicada em escala reduzida.

Lembrando que o foco deste projeto está no sistema desenvolvido que utilizará uma interface qualquer sensível ao pressionamento no solo a fim de garantir a presença do pedestre na lateral da pista.

A escolha do uso de isopor foi por tornar mais fácil os processos de cortar, moldar, montar e furar.

A maquete feita sobre uma placa de isopor que tem 50 x 30 centímetros foram usados canudinhos de plástico para a simulação dos postes, foi usada tinta acrílica para pintura das calçadas em tom de cinza, as caixas de luzes dos semáforos foram feitas de papel com plástico colorido e isopor, papel camurça foi usado para simular o asfalto e a faixa de pedestre, a faixa amarela de separação de mão e contramão foi feita com fita adesiva, setas indicam a direção de travessia dos pedestres, os botões de entrada são verdes e os de saída brancos.

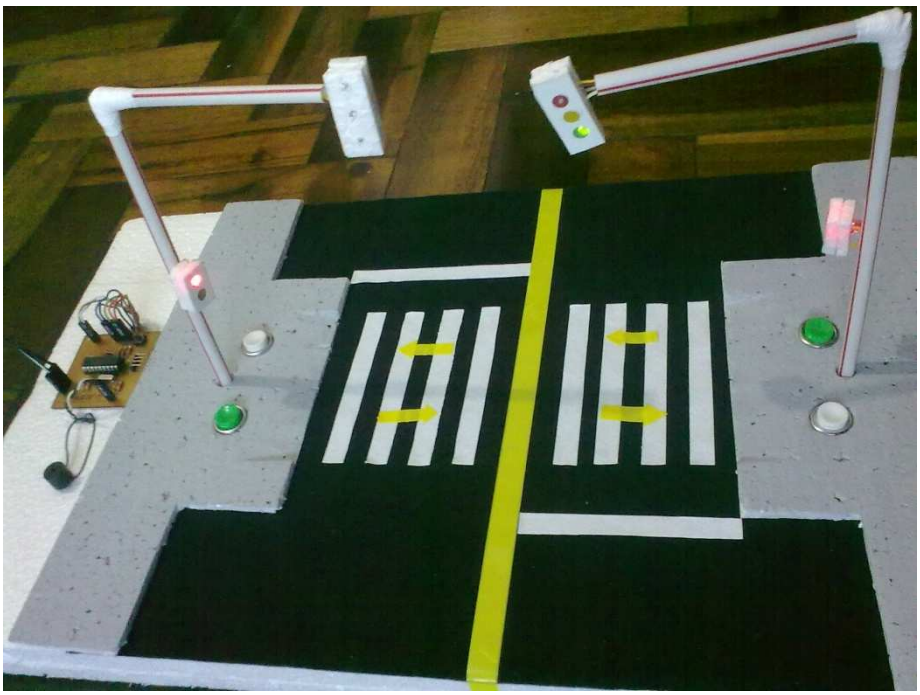


Figura 5.13 – Maquete

CAPÍTULO 6 – RESULTADOS OBTIDOS

Foi construída para este projeto uma placa de circuito impresso, micro-controlada, autônoma, a fim de controlar um semáforo e as interrupções no fluxo dos veículos para passagem dos pedestres. Leds coloridos mostram o estado atual dos semáforos tanto para os veículos quanto para os pedestres. Além das luzes, uma sinalização sonora é emitida para informar a interrupção do trânsito, orientando os pedestres com deficiência visual.

6.1 – Simulações

Nesta fase do trabalho as funções do sistema foram exaustivamente testadas a fim de encontrar possíveis erros em sua aplicação prática.

Os testes, tanto no ambiente virtual quanto na placa construída, foram bem sucedidos e tanto o programa quanto o circuito se comportaram de forma satisfatória, tento respaldo tecnológico e prático para poder ser usado em uma aplicação real.

6.2 – Problemas Encontrados

Alguns pequenos problemas foram encontrados nos testes realizados, como o tempo indicado na linguagem C através do comando ‘delay_ms’ é adicionado ao tempo de processamento do micro-controlador, desta forma não se pode obter uma contagem exata do tempo decorrido em cada parte do sistema e nos eventos provocados ao mesmo, porem a diferença é insignificante para a apresentação e o bom funcionamento geral do sistema.

CAPÍTULO 7 – CONSIDERAÇÕES FINAIS

7.1. – Conclusão

Nesse trabalho foi desenvolvido um protótipo de uma faixa de pedestres a fim de apresentar o sistema proposto de forma clara e objetiva, simulando o ambiente real de sua aplicação prática.

Focado na elaboração de uma forma mais acessível, eficaz, higiênica e segura de interromper o trânsito nas faixas de pedestres com semáforo, o presente projeto trouxe da automação uma mudança na interface e no controle dos tempos do semáforo para reduzir os erros e frustrações do sistema atualmente em vigor no Brasil.

Todos os tempos envolvidos podem ser facilmente alterados para a sua aplicação prática em diferentes locais, dependendo da quantidade de pistas e da quantidade de pedestres em cada local.

Os objetivos do trabalho foram alcançados proporcionando uma economia no tempo em que o semáforo interrompe o trânsito dos veículos por ter a opção de liberar novamente o trânsito na conclusão da travessia do pedestre e fazendo com que este somente seja interrompido quando existe de fato alguém a espera para efetuar a travessia da via.

7.2. – Sugestões para Trabalhos Futuros

Os pontos que poderiam melhorar este projeto são:

- Acrescentar um display com a informação dos segundos se passando em cada etapa do processo.
- Usar sensores para verificar se os carros pararam e/ou medir o fluxo de veículos e/ou se estão parados sobre a faixa.
- Acrescentar uma placa ou tapete impermeável ao botão aumentando sua superfície.
- Efetuar pesquisas práticas para dimensionar o tamanho e localização da interface no solo.

REFERÊNCIAS BIBLIOGRÁFICAS

DETRAN-DF, Acidentes com Morte na Faixa de Pedestre - Distrito Federal, 1997 - 2010 (Inf. Nº 1/2011), Disponível em: <http://www.detran.df.gov.br/sites/200/240/00000756.pdf> Acesso em 20 Abr. 2012.

DETRAN-DF, Pesquisa de Respeito à Faixa de Pedestre - Distrito Federal, 2010 (Inf. Nº 2/2011) Disponível em: <http://www.detran.df.gov.br/sites/200/240 /00000785.pdf> Acesso em 20 Abr. 2012.

DNER, Departamento Nacional de Estradas de rodagem. Ministério dos Transportes do Brasil, Manual Inter-Americano de sinalização rodoviária e urbana, XI Congresso Rodoviário Pan-Americano, Quito 1971.

FRANCO, Paulo. Código de trânsito anotado. 2° edição São Paulo: Editora J.H. Mizuno, 2004.

Internet dos carros promete fim da irritação no trânsito. Inovação Tecnológica. Publicado em: 17 Ago. 2010. Disponível em: <http://www.inovacaotecnologica.com.br/noticias/noticia.php?artigo=internet-dos-carros&id=010150100817> Acesso em 10 Mai. 2012.

HONORATO, Cássio. Alterações introduzidas pelo novo código de trânsito brasileiro: (lei n° 9.503, de 23/09/97). São Paulo, 1998.

PEREIRA, Fabio. Microcontroladores PIC Programação em C. 7ª edição São Paulo: Editora Érica Ltda, 2009.

Prefeitura aplica material antiderrapante nas faixas de pedestre. Prefeitura de Vitória. Publicado em: 2008. Disponível em: <http://legado.vitoria.es.gov.br/diario/2008/0731/faixapedestre.asp> Acesso em 10 Mai. 2012.

SOUZA, David. Desbravando o PIC – ampliado e atualizado para PIC 16F628A. 12ª edição. São Paulo: Editora Érica Ltda, 2008.

APÊNDICE A – PROGRAMA INSERIDO NO MICRO-CONTROLADOR

```
/************************************Projeto Semáforo*****************************************

Nome do Arquivo: projeto semaforo.c
Versão: 1.0
Descrição:

Autor:
Compilador: PIC COMPILER Versão 3.43.
Ambiente de simulação: Proteus 7.7 SP2 toolsuíte ISIS Professional.
Microcontrolador utilizado: PIC16F628A.
Data:

*************************************************************************************************/

#include <16f628a.h>   //Inclui os arquivos com as definições do PIC16F628A.
#fuses XT, NOWDT, NOPROTECT, PUT, BROWNOUT, MCLR
#use delay(clock=4000000)   //Cristal oscilador = 4MHz

/************************************Definições dos pinos no PIC*********************************/

#define   Sem_Verm_Ped   PIN_A0    //Pino de saída p/ o Semáforo Vermelho de Pedestre.
#define   Sem_Verd_Ped   PIN_A1   //Pino de saída p/ o Semáforo Verde de Pedestre.
#define   Sem_Verm_Car   PIN_A2   //Pino de saída p/ o Semáforo Amarelo dos Carros.
#define   Sem_Amar_Car   PIN_A3   //Pino de saída p/ o Semáforo Amarelo dos Carros.
#define   Sem_Verd_Car   PIN_B1   //Pino de saída p/ o Semáforo Verde dos Carros.
#define   Buzzer         PIN_B0   //Pino de saída p/ o Buzzer.
#define   Bot_BE         PIN_B4   //Pino de entrada p/ os push button BE1 e BE2.
#define   Bot_BS         PIN_B5   //Pino de entrada p/ os push button BS1 e BS2.

/************************************Definições das variáveis na RAM*****************************/

int Cont_Fim=00; //Declara e inicializa a variável p/ timer = 5s rotina de saida
int Conta_Seg=00; //Declara e inicializa a variável p/ timer = 15s.
int Cont_Be=00; //Declara e inicializa a variável p/ timer = 5s rotina de espera do pedestre sobre o BE
int Cont_Sesse=00; //Declara e inicializa a variável p/ timer = 60s rotina de espera do último ciclo
int Block=00; //Declara e inicializa a variável p/ bloqueio dos botoes

/************************************** Função Interrupção ***************************************/

void Fim()
{
  int Cont_Fim=00;
     while(true)   //Enquanto verdadeiro...
```

```
    {
     Cont_Fim --;
      if (Cont_Fim > 4)
        {
         Cont_Fim = 4;
        }
       output_low(Sem_Verd_Ped); // durante este tempo o led vermelho pisca e aciona o Buzzer intermitentemente.
       output_high(Sem_Verm_Ped);// dura 1 segundo que é o tempo do loop
       output_high(Buzzer);
       delay_ms(250);
       output_low(Sem_Verm_Ped);
       output_low(Buzzer);
       delay_ms(250);
       output_high(Sem_Verm_Ped);
       output_high(Buzzer);
       delay_ms(250);
       output_low(Sem_Verm_Ped);
       output_low(Buzzer);
       delay_ms(250);
       output_high(Sem_Verm_Ped);
      if(Cont_Fim == 0)
        {
          output_low(Sem_Verm_Car);
          output_high(Sem_Verd_Car);
         return;
        }
    }
}
/********************************** Função temporizador 15 segundos ***************************/

void Tempo_15()
{
  int Conta_Seg=00; //Declara e inicializa a variável p/ timer = 15s.
  int Block=00; //Declara e inicializa a variável p/ bloqueio dos botoes
   output_low(Sem_Verd_Car);   //Semáforo verde dos carros apagado.
   output_high(Sem_Amar_Car);   //Aciona semáforo amarelo dos carros.
   delay_ms(3000);   //Aguarda 3 segundos.

   output_low(Sem_Amar_Car);   //Semáforo amarelo dos carros apagado.
   output_high(Sem_Verm_Car);   //Aciona semáforo vermelho dos carros.
   delay_ms(2000);   //Aguarda 2 segundos.

   output_low(Sem_Verm_Ped);   //Semáforo vermelho de pedestre apagado.
   output_high(Sem_Verd_Ped);   //Aciona semáforo verde de pedestre.
   delay_ms(100);   //Tempo p/ estabilização dos pinos do microcontrolador.
```

```
   while(true)
   {
     Conta_Seg --;
     if (Conta_Seg > 14)
        {
         Conta_Seg = 14;
        }
     if(!input(Bot_BE) && Block == 0){
        Conta_Seg = Conta_Seg + 9;
        Block = 1; // bloquear botões
     }
     if(!input(Bot_BS) && Block == 0){
        Fim(); // chamar interrupçao de fechamento
        return;   //retorna.
     }
          output_high(Buzzer); //durante este tempo de 1 segundo o Buzzer é ligado e desligado intermitentemente.
          delay_ms(200);
          output_low(Buzzer);
          delay_ms(150);
          output_high(Buzzer);
          delay_ms(200);
          output_low(Buzzer);
          delay_ms(150);
          output_high(Buzzer);
          delay_ms(200);
          output_low(Buzzer);
          delay_ms(100);

     if (Conta_Seg == 0)   //Se valor de Conta_Seg = 0 ...
        {
          Fim(); // chamar interrupçao de fechamento
          return;   //retorna.
        }
     //Decrementa segundos.
   }
}

/************************************ Função BotaoBE 5s ***************************************/
void BotaoBE()
{
int Cont_Be=00; //Declara e inicializa a variável p/ timer = 5s.
   while(!input(Bot_BE)) // Verifica se o BE está acionado
   {
    Cont_Be --;   //Decrementa segundos.
      if (Cont_Be > 4)
       {
```

```
      Cont_Be = 4;
     }
     if (Cont_Be == 0)   //Se valor de Cont_Cinc = 0 ...
       {
        Tempo_15();
        return;   //retorna.
       }
   delay_ms(1000);   //Decrementa contador de segundos a cada 1s.
  }
}

/************************************* Função Timer_60 ***************************************/

void Timer_60()
{
int Cont_Sesse=00; //Declara e inicializa a variável p/ timer = 60s
while(true)   //Enquanto verdadeiro...
{
  Cont_Sesse --;   //Decrementa segundos.
  if (Cont_Sesse > 9)
   {
    Cont_Sesse = 9;
   }
    if(Cont_Sesse == 0)   //Se valor de Cont_Seg = 0 ...
    {
      BotaoBE();
      return;
    } //fim do if disparando a interrupção
  delay_ms(1000);   //Decrementa contador de segundos a cada 1s.
}   //fim do while
}   //fim void Semaforo.

/************************************* Programa principal ***************************************/

void main()
{
  do
  {
    output_low(Sem_Verm_Car);
    output_low(Sem_Amar_Car);
    output_low(Buzzer);   //Desliga Buzzer.
    output_low(Sem_Verd_Ped);
    output_high(Sem_Verm_Ped);   //Semáforo vermelho pedestre aceso.
    output_high(Sem_Verd_Car);   //Semáforo verde carros aceso.
            Timer_60();
  }while(1);
```

```
}
/*****************************************************************************************/
```