UNICEUB – CENTRO UNIVERSITÁRIO DE BRASÍLIA

FATECS – FACULDADE DE TECNOLOGIA E CIÊNCIAS SOCIAIS APLICADAS

CURSO DE ENGENHARIA DA COMPUTAÇÃO

RENAN OLGUINS MARTINS

CIRCUITO CONTROLADOR DE DISPONIBILIDADE DE VAGAS EM ESTACIONAMENTO

BRASÍLIA / DF

2º SEMESTRE 2010

RENAN OLGUINS MARTINS

CIRCUITO CONTROLADOR DE DISPONIBILIDADE DE VAGAS EM ESTACIONAMENTOS

Trabalho de conclusão de curso apresentado como parte das atividades para obtenção do título de Bacharelado em Engenharia de Computação, do curso de Engenharia de Computação da Faculdade de Tecnologia e Ciências Sociais Aplicadas de Brasília – UniCEUB.

BRASÍLIA / DF

2º SEMESTRE 2010

DEDICATÓRIA

Este trabalho é dedicado às pessoas que acreditaram no meu trabalho e no meu potencial, em especial aos meus pais Antônio Carlos e Vania, à minha avó Maria José, aos meus irmãos, à minha namorada e aos grandes amigos.

AGRADECIMENTO

Agradeço primeiramente a Deus, por me abençoar com a sabedoria e entendimento, aos meus pais pelos esforços incansáveis para formação de meu caráter, a minha namorada pelo apoio constante, aos meus irmãos, avós, a minha família, e em caráter especial à professora Maria Marony que supervisionou e me orientou sabiamente durante o desenvolvimento deste e aos meus amigos Ronie Paulucio e Pedro Henrique que me ajudaram bastante no início do meu projeto final.

A todas as pessoas que, de alguma forma, contribuíram positivamente para que este projeto fosse concluído com sucesso.

“Only the knowledge that make us better is useful.”

- Sócrates

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E UNIDADE DE MEDIDAS

ASCII – **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange

CDVE– Controlador de Disponibilidade de Vagas em Estacionamento

CMOS – **C**omplementary **M**etal **O**xide **S**emiconductor

EPROM – **E**rasable **P**rogrammable **R**ead-**O**nly **M**emory

I2C - ***I**nter-**I**ntergrated **C**ircuit*

IDE - ***I**ntegrated **D**evelopment **E**nvironment*

ISP - ***I**n-**S**ystem-**P**rogramming*

LED – **L**ight **E**mitting **D**iode

RAM – **R**andom **A**ccess **M**emory

RS232 – **R**ecommended **S**tandard **232**

SBUF –**S**erial Data **Buf**fer

SCON – **S**erial **C**ontrol

SQL – **S**tructured **Q**uery **L**anguagem (Linguagem Estruturada de Consulta)

TTL – **T**ransistor-**T**ransistor **L**ogic

UART – **U**niversal **A**ssynchronous **R**ecevier **T**ransmitter

USB – **U**niversal **S**erial **B**US (Barramento Universal Serial)

UNIDADES DE MEDIDA

A	Ampère, unidade de corrente elétrica
Hz	Hertz, unidade de frequência
Ω	Ohm, unidade de resistência elétrica
V	Volts, unidade de tensão elétrica
W	Watt, unidade de potência

RESUMO

O projeto descrito neste trabalho propõe uma solução alternativa para a diminuição dos problemas encontrados em estacionamentos. Com o uso de tecnologias de detecção, identificação automática, é especificado no presente projeto como elaborar um sistema integrado que apresente maior conforto e segurança para os usuários e empreendedores nos estacionamentos dotados de tal sistema. O objetivo deste projeto é utilizar sensores *reed-switch* (sensores magnéticos) para identificar os veículos que param e saem de uma vaga no estacionamento, de modo a registrar esses dados em um banco de dados gerando relatórios. Em seguida, apresentar por meio de um painel (monitor), quais são as que estão ocupadas e disponíveis. Para detecção dos estados das vagas, é utilizado um sistema microcontrolado conectado a um computador central, onde é feito o gerenciamento do circuito. Este projeto mostra também um protótipo implementado do referido sistema em escala reduzida com equipamentos e metodologias de caráter acadêmico, porém que comprovam a eficácia na utilização das tecnologias indicadas e o seu grande potencial comercial.

Palavras-chave: Kit 8051, sensores, Placa 8051, Controle de Vagas, Reed-Switch, microcontrolador, AT89s8253, ímãs, programa CDVE.

ABSTRACT

The project described in this paper proposes an alternative solution to reducing the problems encountered in parking lots. With the use of detection technology, automatic identification, it is specified in this project as developing an integrated system that offers greater comfort and safety for users and entrepreneurs in the parking lots equipped with such a system. The objective of this project is to use sensors Reed Switch (magnetic sensors) to identify vehicles that stop and leave a parking space in order to record these data in a database, generating reports. Then, through a display panel (monitor), which are those that are busy and available. For detection of states of vacant slots, is used a micromachined system connected to a central computer, where it is done the management of the circuit. This project also shows a prototype of such a system implemented on a small scale with equipment and methodology of an academic nature, but of proven effectiveness in the use of technology and given its great commercial potential.

Key words: Kit 8051, sensors, Plate 8051, control of parking slots, Reed-Switch, microcontroller, AT89s8253, magnets, CDVE Program.

1. INTRODUÇÃO

É fato, a frota de veículos licenciados no Distrito Federal (DF) já passa de um milhão e não pára de crescer, isso sem levar em consideração os veículos que estão licenciados em outros estados, porém trafegam no DF. Como conseqüência já é praticamente impossível encontrar vagas em alguns estacionamentos onde o fluxo é maior. No DF, existe um veículo para cada 2,3 habitantes. Sendo que, a taxa de motorização é a relação entre o número de carros e a quantidade de habitantes, expressa na tabela 1.1:

Quadro 1.1 - Crescimento da frota de carros no DF.

	Crescimento anual da frota: (%)	Frota atual/2009: (em milhões)	Projeções /2010: (em milhões)	Projeções /2020: (em milhões)
Distrito Federal	8,2	1.086.015	1.175.068	2.584.248
Belo Horizonte	7,6	1.146.091	1.233.193	2.565.384
São Paulo	6,2	6.525.033	6.929.585	12.645.972
Curitiba	5,1	1.116.018	1.172.934	1.928.853
Rio de Janeiro	3,8	2.200.465	2.284.082	3.316.534

Fonte: CorreioWeb/2009

Com base nestas informações foi observado que: apesar de estarmos cercados de tecnologias por todos os lados, o ramos de gerenciamento de vagas em estacionamento pouco evoluiu.

A ideia é utilizar sensores nas vagas para detectar se há ou não veículos em cada uma destas. No projeto real (aplicação prática) os sensores seriam os infra-vermelhos que detectariam presença do veículo ou sensores de bobina aberta que associadas a um circuito comparador poderá detectar qualquer objeto com massa férrea (igual ou superior ao pré-estabelecido no circuito) que seja alocado na vaga. Para demonstração a nível acadêmico optou-se por um *reed-switch* que nada mais é do que um dispositivo eletrônico que ao ser aproximado de um campo magnético ele fecha os contatos funcionando como uma chave (com campo magnético conduz, sem o campo não conduz eletricidade).

Para tratar estas informações colhidas pelos sensores foi utilizado um circuito eletrônico onde o principal componente é um dispositivo da família 8051, no qual estão as instruções de verificação de presença. Os sensores quando detectam a presença de um campo magnético com os ímãs envia a informação ao microcontrolador que através das instruções inseridas nele, é capaz de tratar esses dados e logo em seguida envia para um computador.

No computador estas informações são armazenadas em um programa específico podendo ser consultada a qualquer momento. Tais, informações poderão ser usadas para tomadas de decisões, relatórios gerenciais, estatísticas, verificação qual o horário de pico, assim como a demanda de cada vaga (quanto tempo a vaga fica ocupada ou desocupada, rotatividade). Além disso, o computador disponibilizará informações em tempo real para os gestores do equipamento, orientando quais vagas encontram-se preenchidas ou não, podendo o gestor verificar o uso por datas e horas específicas. Com esses dados, seus diretores terão uma maior autonomia sobre o fluxo geral, como: tarifas promocionais em horários de menor demanda aumentando assim o incentivo aos motoristas utilizarem esta e não outra rede de estacionamento. Como se pode observar na figura 1.1, a topologia do projeto:

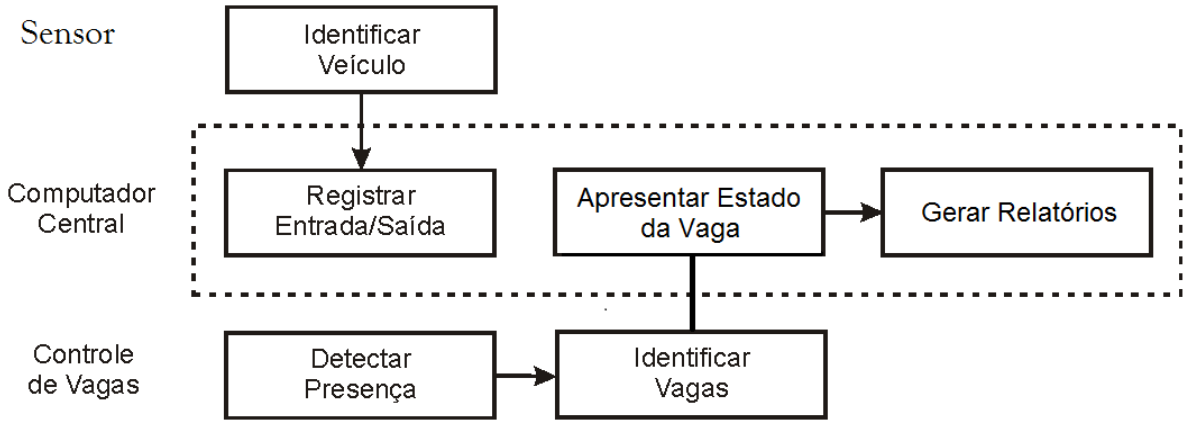

Figura 1.1 – Diagrama de blocos da topologia do projeto. (Fonte: Autor)

Assim que o motorista entrar no estacionamento, ele é informado por meio de *leds* quais vagas disponíveis no momento. Cada vaga possui um sensor de presença do tipo *reed-switch* conectado a um microcontrolador, como mostrado na figura 1.2, onde é

executado a identificação do veículo pelos sensores, que verifica os estados das vagas e transmite as alterações destes estados ao computador central, onde é feita a gestão e geradas as informações para os gerenciadores do programa.

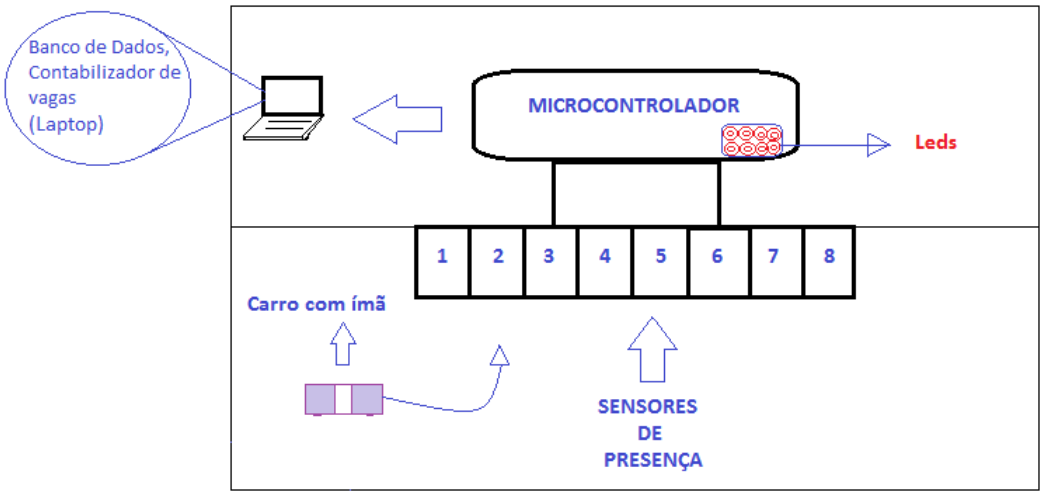

Figura 1.2 - Visão Geral do Projeto. (Fonte: Autor)

Para a implementação deste projeto, foi necessário adotar soluções de cunho acadêmico e demonstrativo, devido aos custos elevados de uma implementação com equipamentos ideais para um ambiente real. Para isso, este projeto utiliza apenas um kit 8051 com um microcontrolador AT89s8253, oito sensores *reed-switch*, para identificar o estado da vaga, gerir e posteriormente armazenar em um banco de dados.

1.1. Motivação

Atualmente, devido à falta de planejamento, a procura por vagas está bem acirrada, pois o tempo dos motoristas é limitado, e exige a busca por soluções automatizadas que busque diminuir esse temo de procura. Não obstante, quem nunca teve o desprazer de ter passado alguns preciosos minutos circulando em diversos setores de um estacionamento a procura de uma vaga para poder estacionar? - Seria muito mais fácil se ao chegar em um estacionamento houvesse painéis e monitores orientando em tempo real em quais setores do estacionamento há vagas disponíveis, podendo assim o motorista ir direto a esta, de modo a

evitar voltas desnecessárias. Tanto a desorganização, a falta de investimento e estrutura, como também a contagem manual das vagas denotam falhas no atual modelo de controle em estacionamentos, mesmo com vários projetos patenteados relacionados ao tema, proporcionalmente é baixo o número de implementações. Estes problemas afetam os usuários e gestores em vários aspectos e, como proposta, a implantação de um projeto que automatiza certas tarefas se faz bastante oportuno, como no caso dos estacionamentos rotativos.

Pode-se ainda levar em consideração o fator econômico. Imagine-se chegando a um shopping, logo na entrada do estacionamento tivesse um monitor indicando que em um determinado setor a vaga “n” está desocupada. Isso certamente resultaria num ganho de tempo por parte do consumidor, o que poderia proporcionar um grande aumento nas vendas. É bem provável que grandes centros comerciais já tenham perdido muito dinheiro por deixar de vender em suas lojas, cinemas e restaurantes, uma vez que clientes em potencial acabam ficando menos tempo circulando nas lojas por passar mais tempo procurando vagas para estacionar.

A motivação para elaboração deste projeto é, além de abordar uma solução para minimizar os problemas, aplicar conceitos técnicos adquiridos ao longo do curso de Engenharia de Computação. Ou seja, utilizar tecnologias e metodologias de engenharia para solucionar ou aprimorar métodos empregados em uma dada situação problemática.

1.2. Objetivos

Este projeto tem por objetivo o intuito de utilizar um software com a finalidade de controlar um estacionamento, sendo possível assim gerenciar a rotatividade de cada vaga individualmente ou em bloco, fato que poderá ser de grande utilidade para planejamento de ampliações, promoções, etc. Além disso, o software em associação a um hardware desenvolvido no kit 8051 e demais componentes eletrônicos, torna possível o gerenciamento das vagas pelo computador, como também a fixação de monitores e painéis eletrônicos em pontos estratégicos, e através destes painéis informar ao motorista onde encontrar a vaga mais próxima.

O objetivo principal é apresentar uma proposta de melhoria na organização de estacionamentos que haja, de forma automática, a gestão das vagas e que as que estão

disponíveis sejam indicadas para o motorista. Além disso, o registro dos eventos de entrada e saída de veículos é feita também de forma automática, através da leitura dos dados colhidos pelos sensores por meio da comunicação com fio ao computador, gerando relatórios das informações necessárias, tais como, por exemplo: ampliação, promoções relâmpago (em horários de menor fluxo), sistemas de tarifação, monitoração, rastreamento, ponto eletrônico, programas de fidelidade.

Para a implementação física do projeto, foi construído um protótipo em escala reduzida, de caráter acadêmico, com tecnologias semelhantes às propostas para ambientes reais, porém com características mais adequadas a um projeto final, como simulação de sensores de presença, algoritmo com número limitado de oito (8) vagas e equipamento de comunicação. Ao final do projeto, pode-se destacar as seguintes metas propostas:

- Implementar e desenvolver um software que auxilie a gestão de vagas utilizando sensores com fiação;

- Armazenar os registros de entrada e saída dos identificadores em banco de dados;

- Estruturar os circuitos para simular os sensores de presença, com leds e a comunicação com o microcontrolador;

- Desenvolver programa no Kit 8051 para a identificação de sinal proveniente da presença de veículo em vaga;

- Desenvolver programa de gestão de vagas e apresentação das vagas em tela;

1.3. Metodologia

Para o desenvolvimento desta monografia e implementação do projeto, foi utilizada um método de estudo de pesquisas bibliográficas em livros especializados em circuitos elétricos e eletrônicos, linguagens de programação, banco de dados, sensores industriais e sites especializados na internet, para auxílio na implementação e consulta de equipamentos. Para a implementação do protótipo, foram escolhidos os seguintes equipamentos:

- Kit de desenvolvimento para o microcontrolador 8051 para a identificação de vagas e comunicação com o programa localizado no computador central;
- Cabo de comunicação RS232 (PC/Kit) para a comunicação entre o notebook e o protótipo;
- Notebook e microcontrolador como dispositivos de entrada, contendo programa gestor, sistema de registro (incluindo banco de dados) e apresentação do estado das vagas.

1.4. Estrutura

Este trabalho está dividido em seis capítulos, organizado de tal forma que sejam apresentadas as tecnologias envolvidas e, posteriormente, a aplicação e implementação destas em estacionamentos. Sendo que a INTRODUÇÃO aborda a motivação do projeto, descreve os objetivos do mesmo, além da sua metodologia. Este capítulo reserva ainda a estrutura que traz por partes toda a organização escrita da monografia, proporcionando ao leitor um acompanhamento linear sobre os objetos de estudo.

O segundo capítulo, REFERENCIAL TEÓRICO, apresenta os principais conceitos envolvidos neste trabalho, faz menção às tecnologias utilizadas e equipamentos do circuito controlador e seus aspectos conceituais e físicos, bem como os aspectos relevantes para a adoção destas tecnologias para o propósito deste projeto.

O capítulo 3, DESCRIÇÃO DO HARDWARE, trata do modelo do Kit didático escolhido assim como o tipo. Constam ainda neste capítulo, os dados técnicos que o microcontrolador contém e cada um dos componentes físicos apresentados no hardware, como também seus respectivos custos.

O capítulo 4, IMPLEMENTAÇÃO, designa todos os itens do projeto, a estrutura eletrônica, o desenvolvimento dos softwares e a implementação final, ou seja, um passo-a-passo sobre a engenharia do projeto. São destacados os dados dos equipamentos utilizados, os processos da implantação e as dificuldades encontradas.

No capítulo 5, TESTES E RESULTADOS, são apresentados uma série de resultados para os variados testes realizados no sistema, bem como considerações sobre os métodos utilizados, os parâmetros tentados e apresentação de métodos alternativos em caso de falhas no sistema.

No capítulo 6, CONCLUSÃO, é feita a conclusão sobre o projeto, a relação entre o objetivo esperado e o alcançado e considerações finais, além de sugestões para trabalhos futuros.

2. REFERENCIAL TEÓRICO

Este capítulo aborda as características dos microcontroladores, sensores, suas propriedades físicas, o histórico e as aplicações.

Após a concessão de fabricação dada pela Intel a outros fabricantes para a reprodução e atualização desse chip, as autorizadas desenvolveram uma grande quantidade de chips derivados do 8051. Em 1955, o mercado de microcontroladores ultrapassava US$ 10 bilhões, a considerar em maior parte às máquinas de 8 bits como o 8051. A expectativa é que o mercado atinja mais de US$ 20 bilhões e a família 8051 deve se responsabilizar por dez por cento disso. (Nicolosi, 2002)

Com todos os componentes básicos em um só chip (CPU, RAM, ROM e I/O), estes permitem alta integração, alta confiabilidade e poder. Os engenheiros de desenvolvimentos têm inúmeras opções específicas, como controle e atuação direta com vários tipos de sensores, atuadores, comunicação entre vários processadores, pois possuem internamente uma poderosa interface de comunicação serial. É por isso que se encontra abundantemente 8051 no mercado, desde o teclado de seu computador, controle remoto de sua televisão, no controle de seu carro, na bomba de gasolina, até em aparelhos médicos,etc. Atualmente, a ATMEL dá excelentes contribuições tecnológicas a esta família. (Nicolosi, 2002)

Partindo deste princípio, foi construído um protótipo utilizando um microcontrolador da família 8051. A solução desenvolvida para o presente trabalho é ilustrada na Figura 2.1

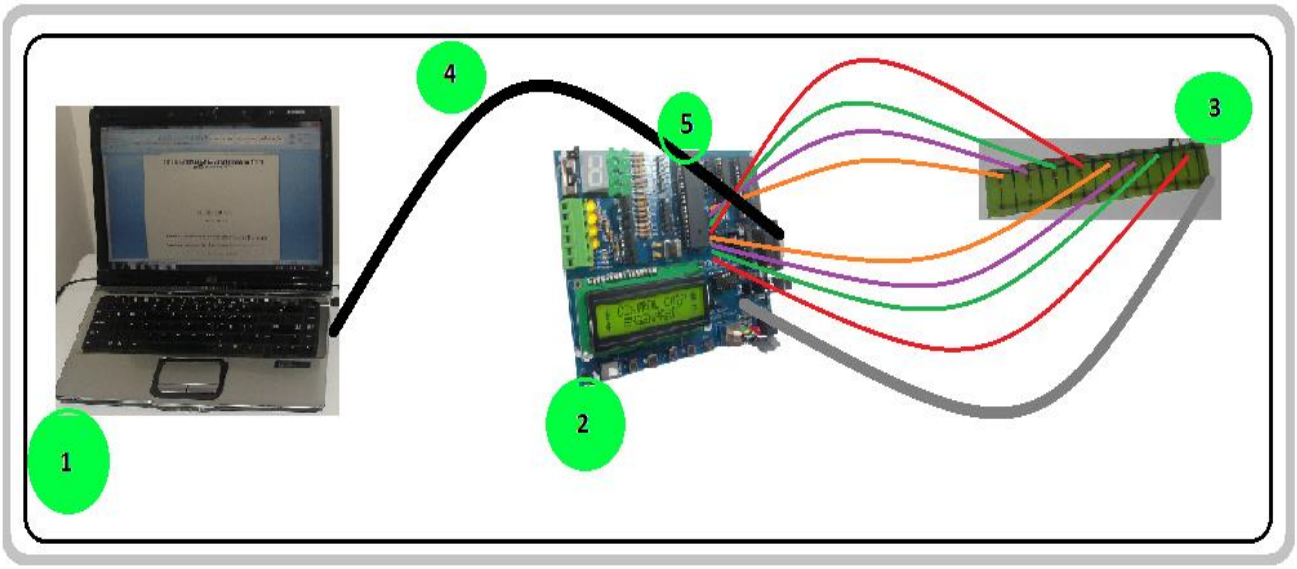

Figura 2.1 - Diagrama Geral do Projeto (Autor)

Onde:

1 – Laptop

2 – placa 8051

3 – Sensores reed-switch

4 – Cabo rs 232

5 - Microcontrolador AT89S8253

Além do microcontrolador, foi utilizado também um sensor magnético. Este tem a finalidade de verificar a presença do automóvel. Outros componentes adotados no projeto foram os *leds* e um cabo RS232, assim como o *laptop* que serve tanto como fornecedor de energia como também como saída das informações.

Os leds, integrados a placa principal, têm a função de indicar quando o sensor detecta a presença de um carro em sua respectiva posição. Caso o sensor verifique a ausência de veículo, o circuito continua aberto e prossegue-se a verificação. O cabo RS232 faz a comunicação entre a porta serial do hardware e a USB do *notebook*.

2.1 - SENSORES

Sensor é um conjunto de elementos que trabalham de uma forma característica e que agem sobre outro elemento deste sistema, atuando para alertar, corrigir, acionar etc. Assim, estes elementos possuem funções específicas, e são classificados de acordo com sua participação no conjunto.

Na prática, um sensor é um elemento capaz de detectar (sentir) uma alteração de energia de um determinado meio e, de acordo com esta alteração, é também capaz de representar esta alteração em outro meio. O exemplo mais primordial de um sensor é a pele humana. A nossa pele é caracterizada por ser o mais evidente órgão sensorial do homem pelo sentido do tato e, por isso, é uma espécie de sensor, que, conceitualmente percebe as alterações que ocorrem no ambiente ao seu redor como o ar ou outros corpos, e transmite esta percepção para interpretação por parte do cérebro.

Analogamente ao exemplo acima, um diagrama pode descrever o funcionamento genérico de um sistema de sensores. A Figura 2.2 representa, por diagrama de blocos, os efeitos físicos (que são relacionados a alguma grandeza física específica), e as formas de energia de cada tipo de efeito sensibilizando o sensor e, este por sua vez, traduz o efeito físico para um sinal elétrico de saída que alimentará um sistema de tratamento e/ou análise do ocorrido efeito físico. Isto é, o sensor é um sistema completo que produz um sinal elétrico de saída proporcional à grandeza sendo medida.” (WERNECK, 96)

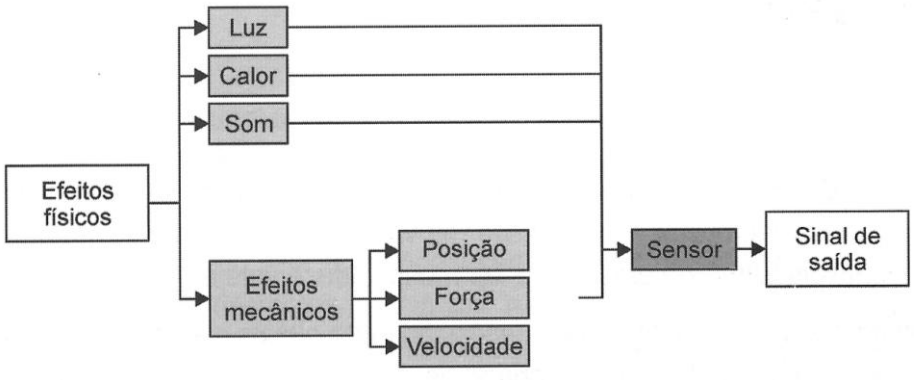

Figura 2.2 – Efeitos físicos como entrada de um sensor. (THOMAZINI; ALBUQUERQUE, 2007)

Sensores são os elementos que possuem uma característica de perceber as alterações energéticas de um meio. Ou seja, é um dispositivo sensível a alguma forma de energia.

2.1.1 - Sensores Reed-Switch

Os *reed-switches* ou interruptores de lâminas consistem em dispositivos formados por um bulbo de vidro no interior do qual existem lâminas flexíveis feitas de materiais que podem sofrer a ação de campos magnéticos. O bulbo de vidro é cheio com um gás inerte de modo a evitar a ação corrosiva do ar sobre as lâminas, o que afetaria o contato elétrico em pouco tempo. Abaixo foi descrito o funcionamento do sensor reed-switch:

Na sua versão mais simples, como à do presente projeto, temos duas lâminas, montadas conforme mostra a figura 2.3.

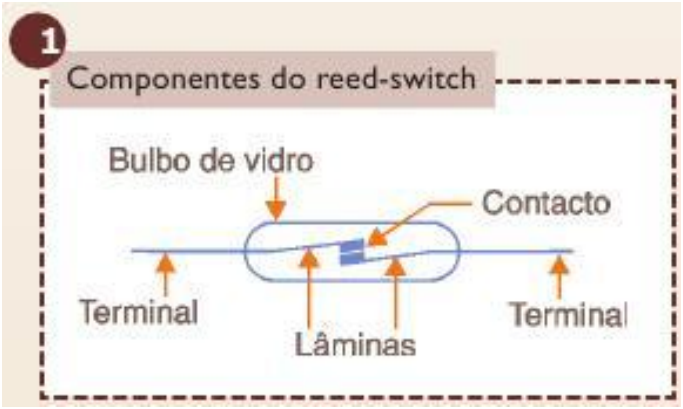

Figura 2.3 – Componentes do reed-switch. (Fonte: www.mecatronicaatual.com.br)

Nas condições normais, as lâminas estão separadas e nenhuma corrente pode circular através do componente. Ele opera como uma chave aberta. Aproximando-se um ímã permanente do dispositivo (confira na figura 2.4), a ação do campo magnético faz com que as lâminas se magnetizem e com isso se atraiam, unindo-se. Nestas condições, o contato elétrico é fechado.

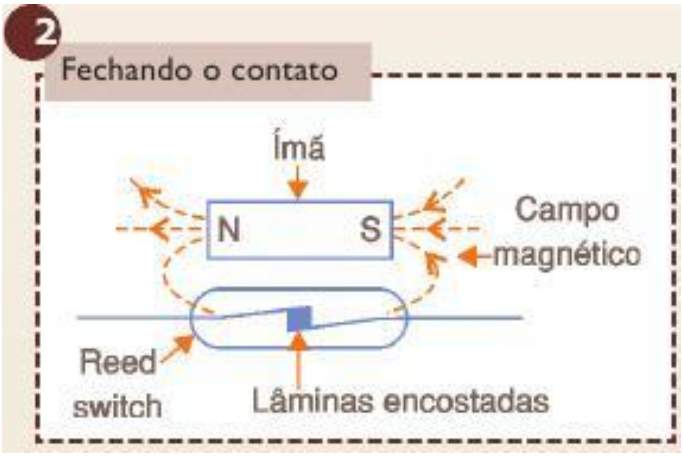

Figura 2.4 – Funcionamento do reed-switch. (Fonte: www.mecatronicaatual.com.br)

Em outras palavras, o *reed-switch* abre e fecha seus contatos conforme a ação de um campo magnético externo.

É importante observar que para termos uma ação apropriada das lâminas fechando os contatos, o campo magnético precisa ser corretamente orientado. Se o campo não magnetizar as lâminas de modo que elas se atraiam, não há a atuação da chave. Na figura 2.5 são indicadas as posições corretas que devem ser usadas para que ímãs permanentes acionem um reed-switch.

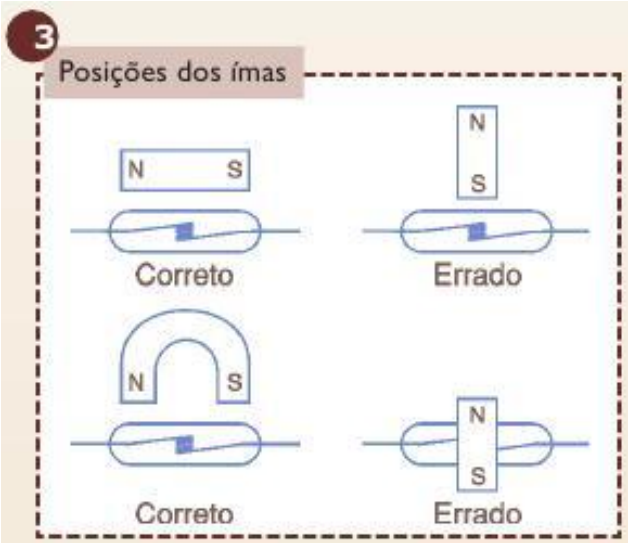

Figura 2.5 – Posições dos ímãs. (Fonte: www.mecatronicaatual.com.br)

O sensor magnético *reed-switch* é de fácil utilização, manuseio e barato. O engenheiro pode elaborar muitos dispositivos interessantes usando essas chaves como sensores, comutadores ou relés. Essa é apenas uma das muitas aplicações existentes para esses úteis componentes. (http://www.mecatronicaatual.com.br)

2.2 - Microncontroladores

No início da década de 1980, a INTEL, que já tinha criado o microcontrolador 8048, lançou uma nova família, herdeira do 8048, intitulada 8051, de "8 Bits". Inicialmente ela lançou o "8051" com ROM interna programável de fábrica, o "8751" com EPROM interna programável pelo usuário e o "8031" com necessidade de ter "chips" de EPROM externa. Após alguns anos, a família se expandiu para o "8052", que tem um timer a mais que o 8051 e suas variações: com EPROM (8752) e sem ROM/EPROM (8032), e uma versão especial do 8052, chamado 8052 - Basic, que tem dentro de sua ROM um interpretador Basic que permite programá-lo também em linguagem Basic. Esta é a família chamada MCS-51. Desde então, o 805l tem sido um dos mais populares microcontroladores, possibilitando ter uma vasta família no mercado, sendo hoje produzido por mais de 30 fabricantes, com mais de 600 variações de chips. [Nicolosi e Bronzeri,2005]

As diferentes versões de chips podem apresentar conversores analógico-digital (A/D), variação com relação à quantidade de memória (RAM / ROM internas), modulação PWM e memória Flash (que possibilita a reprogramação mediante um sinal elétrico), conversores analógico-digital (A/D) e digital-analógico (D/A), comunicação SPI (SPI - Serial Program Interface), comunicação serial I²C (Acces Bus), tecnologia de interconexão para redes locais (Ethernet), CAN (Control Area Network), memórias EEPROM que podem ser utilizadas pelo programa em tempo real, e clock de 252 a lOOMHz. Tudo em apenas um chip. (Nicolosi e Bronzeri,2005)

Hoje em dia, também encontramos microcontroladores derivados da família 8051 que trabalham com 16 bits, com alta performance, sendo aplicados em processamento digital de sinais e controle de sistemas em tempo real. (Nicolosi e Bronzeri,2005)

Alguns dos microcontroladores de 16 bits possuem grande quantidade de memória, maior número de canais de conversores analógico-digitais, grande número de “ports”, “ports” seriais, alta velocidade aritmética e lógica de operação, e um poderoso set de instruções com capacidade de processamento de sinais. (Nicolosi e Bronzeri,2005)

Basicamente, o 80C5 l (80"C"5 l - Tecnologia CMOS), observe o "C" no código ao lado, que é de tecnologia “CMOS” a qual possui baixo consumo de energia, fabricado pela Intel, é considerado o hardware básico da família de microcontroladores 805 l. Porém, com o avanço tecnológico na elaboração de componentes semicondutores e a grande procura por esses componentes, existe hoje um número expressivo de fabricantes no mercado, sendo a variedade de modelos apenas conseqüência da quantidade de fabricantes. Dentre eles temos: Mitsubishi, Siemens, Philips, Intel, Toshiba, Atmel, National, Texas, Sharp e outros. São mais de 600 variações da família básica. (Nicolosi e Bronzeri,2005)

Alguns representantes da família Atmel de microcontroladores de 8 e 16 bits Baseiam-se no microcontrolador C251, uma versão mais completa do 80C51. Outros representantes, como o AT89C5ICC03, possuem uma interface CAN .(Control Area Network), possibilitando comunicações a uma distância de aproximadamente 40 metros, com alta velocidade. Além dessa interface, esse microcontrolador é provido da capacidade de reprogramação sem a necessidade de retirarmos o chip do circuito, o que é feito por software. Os pacotes trazidos pelo sistema CAN são traduzidos em código binário a fim de reprogramarmos a memória Flash. Outros microcontroladores apresentam um sistema similar UART (Universal Assynchronous Recevier Transmitter), porém a distância para comunicação é menor. [Nicolosi e Bronzeri,2005]

Além da UART, existe uma outra forma de comunicação serial denominada de "ACCESS BUS", também conhecida como I²C. Ele foi criado pela Philips e tem sido adotado como padrão. Possui também grande importância no cenário industrial. Com esse protocolo é possível transferir dados, de forma bidirecional, entre o sistema central (mestre) e um sistema periférico (escravo), ou para outros sistemas mestre. Por ter sua interface dentro do chip, não ocupa espaço na placa, além de possibilitar conexão com vários dispositivos ao mesmo tempo por meio de apenas dois pinos físicos. [Nicolosi e Bronzeri,2005]

Tipicamente o 8051 é um microcontrolador que trabalha com palavras de 8 bits, possui alta performance e baixo custo. Um representante básico de família, contém internamente:

• 64 Kbytes de memória para DATA (RAM) e CODE (ROM).

• 256x8 bytes de memória RAM, dividida entre área de uso geral

e registradores especiais.

• Dois timers/counters de 16 bits.

• Uma porta serial programável (UART).

• Interface para memória externa com capacidade de 64 Kbytes

de endereçamento externo para ROM e 64 Kbytes de

endereçamento externo para RAM.

• Quatro portas de I/O.

• Seis possibilidades de interrupções com dois grupos de

prioridades. [Nicolosi e Bronzeri,2005]

Os microcontroladores são componentes que em um único chip, possuem vários elementos como, uma CPU, memórias ROM e RAM, PWM, temporizadores/contadores, conversores AD, canais de comunicação e conversores analógico-digital. (MOHR, 2004)

Sistemas baseados em microprocessadores utilizam vários componentes para implementar as funções citadas anteriormente, sendo assim os microcontroladores levam vantagem na implementação de sistemas mais baratos e mais compactos, do que os baseados em microprocessadores, que por sua vez, possuem CPUs mais poderosas, conjunto de instruções menos limitadas, freqüênciade clock mais alta e costuma ter mais memória de endereçamento. (MOHR, 2004)

Disposto disto é verificado que as aplicações para estes dois tipos de sistemas são bem diferentes, os sistemas controlados por microcontrolador deve ser com menor custo e de

complexidade menor, já um sistema controlado por microprocessador deve exigir uma alta capacidade de processamento.

O microcontrolador utilizado para implementação deste projeto (Figura 2.6) é o AT89S8253, fabricado pela Atmel Corporation, este circuito integrado faz parte da família 8051 de microcontroladores, sendo assim possui o mesmo conjunto de instruções desta família.

Figura 2.6 – Microcontrolador ATMEL AT89S8253. (Autor)

2.2.1. – A Família 8051

Os microcontroladores da família 8051 surgiram no início da década de 80, sucessor do 8048, lançadas pela INTEL, sendo bastante aceito pelo mercado. Nos dias de hoje, existem vários fabricantes com autorização para fabricação dos microcontroladores da família 8051. [NICOLOSI, 2000]

As características principais desta família são: [NICOLOSI, 2000] [MOHR, 2004]

- CPU de 8 bits;
- 64 Kbytes de endereçamento de programa (ROM externa);
- 64 Kbytes de endereçamento de memória de dados (RAM externa);
- 4 Kbytes de memória de programa (ROM interna);

- 128 bytes de memória de dados (RAM interna) e 128 bytes referentes aos registradores especiais;

- 2 timers/contadores de 16 bits;

- 4 portas de I/O;

- 1 interface serial;

- Processador para operação em bits (Booleano);

- 3 entradas de interrupção interna com dois níveis de prioridade;

- 2 entradas de interrupção externa com dois níveis de prioridade;

- Instrução direta de divisão e multiplicação;

- Ciclos típicos de instrução de 1 e 2 µs a 12MHz.

A família do 8051 destaca-se das demais pelo motivo dela ser fabricada por várias empresas, e sendo assim, existe uma grande concorrência pelo mercado consumidor deste tipo de microcontrolador, o tornando mais acessível no mercado.

2.2.1.1. – ARQUITETURA BÁSICA DOS MICROCONTROLADORES 8051/8052

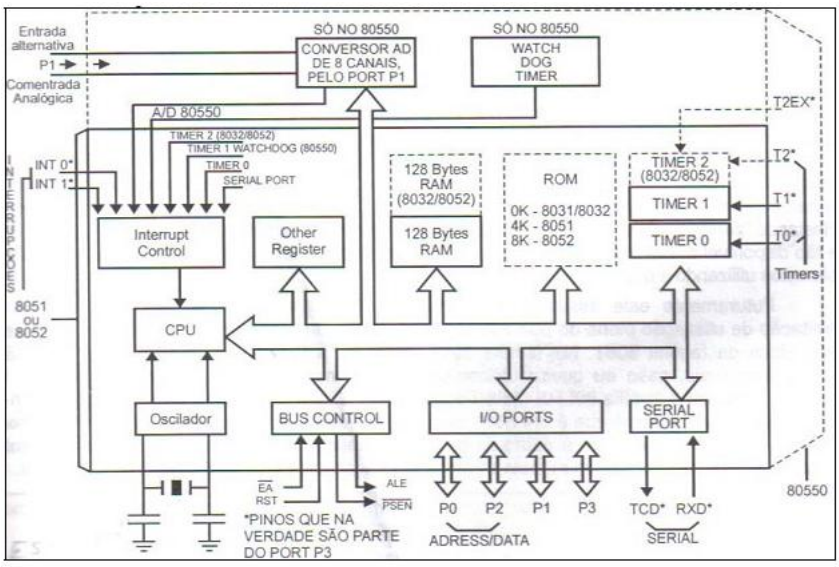

Figura 2.7 – Arquitetura Interna dos microcontroladores da família 8051 – (Nicolosi, 2005)

As portas PO, P1, P2 e P3, cada uma com oito linhas, são destinadas à comunicação externa. PO e P2 se destinam a gerenciar as vias de dados e endereços da comunicação do microcontrolador com a ROM, RAM ou periféricos tipo "I/O mapeado". P1 e P3 se destinam às vias de comunicação externa, sendo tipicamente usadas para interface com o mundo externo. [Nicolosi, 2005]

Além disto, a porta P3 tem funções especiais por onde se comunicam os periféricos internos, que existem nos microcontroladores: timers (2 deles no 8051, 3 deles no 8052), serial (1 unidade) e A/D (de 8 canais, somente no 80550, que neste caso usa a porta P1 como 8 entradas analógicas, além de ter normalmente os timers e serial do 8051). Os asteriscos na figura acima estão justamente representando que estes pinos não são separados dos ports P1 e P3. [Nicolosi, 2005]

Por exemplo, TXD* e RXD* são os pinos P3.0 e P3.1, isto é, os pinos "zero" e "um" da porta P3. O mesmo acontece com T2EX*, T2*, T1* e TO*. Eles fazem parte das “portas” P3 e P1. Logo, a porta P3 não é tão disponível assim como aparenta. A porta P1 é plenamente utilizável como porta de 8 vias nos chips da família 8051. Na família 8052 os pinos P1.0 e P1.1 são compromissáveis, caso seja necessário utilizar o terceiro timer que este chip tem disponível. O mesmo se aplica à família 80550, (este é igual ao 8051, só que a porta P1 é compatível com a utilização do A/D interno) que é um 8051 com um A/D de 8 canais dentro do chip de 40 pinos. Na figura 2.8 é apresentada a visão externa do chip. [Nicolosi, 2005]

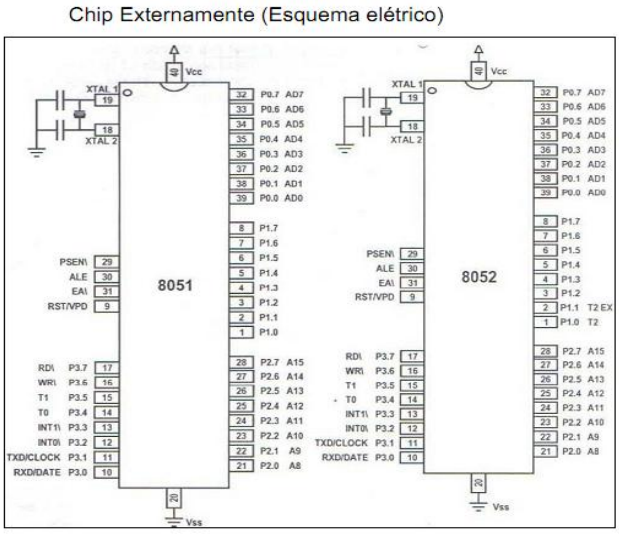

Figura 2.8 - Desenho externo do chip (família 8051 e 8052) - (Nicolosi,2005)

2.2.1.2. – Pinagem no 8051

Os pinos do microcontrolador têm a função de interagir com o resto do circuito, essa interação é feita por meio de sinais elétricos. A Figura 2.9 representa a configuração da pinagem do 8051.

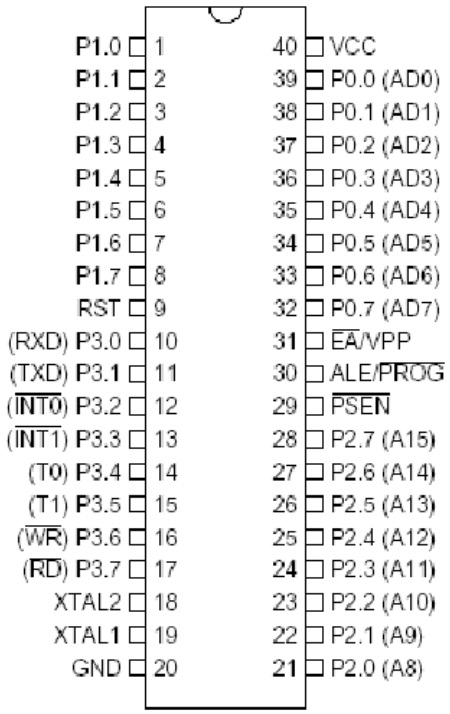

Figura 2.9 – Pinagem do Microcontrolador 8051 – Fonte: Datasheet AT89S52

Os pinos do microcontrolador descrito têm funções diferentes entre si, baixo são descritos o símbolo, a numeração da pinagem, o nome do pino e sua descrição e principais funções. [NICOLOSI, 2000] [MOHR, 2004]

VCC – pino 40 – Fonte de Alimentação – Entrada do dispositivo da fonte de alimentação.

VSS – pino 20 – Terra – Entrada do terra no circuito (GND).

P0.0 a P0.7 – pinos 39 a 32 – Porta 0 ou barramento de endereços menos significativos e dados multiplexados no tempo, quando se utiliza memória ROM e/ou RAM externa – A porta 0 é uma porta de entrada e saída bidirecional de 8 bits de dreno aberto.

Operando como uma porta de saída, cada pino pode absorver oito entradas LS TTL. Escrevendo o 1 lógico nos pinos da porta 0, eles flutuam, e esses estados podem ser utilizados como entradas de alta impedância. A porta zero é também o barramento de endereços menos significativos, multiplexados no tempo com o barramento de dados durante o acesso a uma memória de programa ou de dados externa. Nessa aplicação, utilizam-se pull-ups internos ao escrever o 1 lógico e pode-se fornecer ou absorver até oito entradas LS TTL. A porta 0 também recebe os bytes de códigos durante a programação da EPROM e envia os bytes de códigos durante a verificação do programa gravado na ROM e EPROM.

P1.0 a P1.7 – pinos 1 a 8 – Porta 1 é uma porta de entrada e saída bidirecional de 8 bits com pull-ups internos. Os buffers de saída podem fornecer ou absorver quatro entradas LS TTL. Os pinos da porta 1, que têm o 1 lógico em suas saídas, são levados a 1 lógico pelos pull-ups internos e, nesse caso, podem ser utilizados como entrada. Com a porta 1 funcionando como entrada, seus pinos, que são externamente levados para zero lógico, fornecerá corrente devido aos seus pulls-ups internos. A porta 1 também recebe o byte de endereço menos significativo durante a programação da EPROM e durante a verificação do programa gravado da ROM e EPROM.

P2.0 a P2.7 – pinos 21 a 28 – Porta 2 ou barramento de endereços mais significativos quando se utiliza a memória ROM e/ou RAM externa – A porta 2 é uma porta de entrada e saída bidirecional de 8 bits com pull-ups internos. Os buffers de saída podem fornecer ou absorver quatro entradas LS TTL. Os pinos da porta 2, que têm o 1 lógico em suas saídas, são levados a 1 lógico pelos pull-ups internos e, nesse caso, podem ser utilizados como entrada. Com a porta 2 funcionando como entrada, seus pinos, que são externamente levados para zero lógico, fornecerá corrente devido aos seus pull-ups internos. A porta 2 também emite o byte do endereço mais significativo durante a programação da EPROM, durante a busca da memória de programa externa e durante o acesso à memória de dados externa que utiliza 16 bits de endereçamento. Nessa aplicação, a porta utiliza pull-ups internos quando emite 1 lógico. Durante o acesso à memória de dados externa que utiliza endereçamento de 8 bits, a porta 2 emite o conteúdo do registrador de função especial P2. A porta 2 também recebe o byte do endereço mais significativo durante a programação da EPROM e durante a verificação da ROM e EPROM.

P3.0 a P3.7 – pinos 10 a 17 – Porta 3 ou pinos de recepção e transmissão

serial, interrupção externa 1 e 0, entrada de clock externa do timer 0 e 1, sinal de escrita e leitura de memória RAM externa – A porta 3 é uma porta de entrada e saída bidirecional de 8 bits com pull-ups internos. Os buffers de saída podem fornecer ou absorver quatro entradas LS TTL. Os pinos da porta 3 que têm 1 lógico em suas saídas, são levadas a 1 lógico pelos pull-ups internos e, nesse caso, podem ser utilizados como entrada. Com a porta 3 funcionando como entrada, seus pinos, que são externamente levados para 0 lógico, fornecerá corrente devido aos seus pull-ups internos.

RST – pino 9 – Reset – Entrada de Reset. Um nível lógico alto nesse pino por dois ciclos de máquina: enquanto o oscilador está sendo executado, reseta o dispositivo (inicializa alguns registradores internos com valores predefinidos pelo fabricante).

ALE/PROG\ – pino 30 – Address Latch Enable/PROG\ (pulso habilitador de captura de endereço) – Pulso de saída que indica a um dispositivo externo que ele deve captar o sinal de endereço no barramento de endereço e os dados que estão multiplexados no tempo. Esse pino também serve como entrada do pulso de programação durante a programação da EPROM. Em operação normal, o sinal de ALE é emitido a uma ração constante de 1/6 da freqüência do oscilador e pode ser utilizado para fins de clock ou temporizador externo. Observe que um pulso de ALE é emitido a cada acesso à memória de dados externa.

PSEN\ – pino 29 – Program Store Enable (pulso habilitador de armazenamento de programa) – É o pulso de leitura para a memória de programa externa. Quando o dispositivo está executando códigos da memória de programa externa, PSEN\ é ativado duas vezes a cada ciclo de máquina, exceto quando existe acesso à memória de dados externa.

EA\ /VPP – pino 31 – External Access Enable Programming Supply Voltage – EA\deve ser ligado a VSS para habilitar o dispositivo a buscar códigos da memória de programa externa no endereço inicial de 0000h até FFFFh. EA\ deve ser ligado a VSS para a execução do programa contido na memória ROM/EPROM interna. Se o Security Bit (bit de segurança) na EPROM é programado, o dispositivo não buscará códigos de qualquer local de memória de programa externo. Esse pino também recebe a fonte de alimentação de programação de 21 V durante a programação da EPROM.

XTAL1 – pino 19 – Entrada do amplificador oscilador inversor.

XTAL2 – pino 18 – Saída do amplificador oscilador inversor.

2.3. - Linguagem Assembly

Uma explicação básica sobre a linguagem Assembly se faz necessária, visto que esta é a linguagem utilizada no desenvolvimento do programa de controle do projeto. A escolha desta linguagem primeiramente foi feita pelo fato de ter sido a linguagem estudada juntamente com o microcontrolador 8051 na disciplina de Microprocessadores e Microcontroladores.

O Assembly é normalmente confundido com a linguagem de máquina, mas não é a mesma coisa. É uma linguagem escrita por códigos alfanuméricos, também chamados mnemônicos, que facilitam o entendimento dos programadores e a linguagem de máquina deve ser compreendida pelas máquinas. A semelhança mais evidente entre a linguagem de máquina e a linguagem Assembly é que cada instrução desta segunda linguagem corresponde exatamente a uma instrução de máquina. O que não ocorre com as linguagens de alto nível. (SICA, 2006)

O termo Assembly é um termo da língua inglesa que significa montagem, construção. Assembler é um termo também da língua inglesa, significa montador.

Estes dois termos são bastante confundidos, mas é importante ressaltar que eles se referem a coisas diferentes. O Assembly é a linguagem de programação em si, já o Assembler é o seu compilador, ou melhor, é o programa responsável pela tradução da linguagem para a máquina. (Zelenovsky e Mendonça, 2005)

O nível mais baixo de programação é denominado de linguagem de máquina, onde as instruções e os dados são utilizados em nível binário, é a linguagem que o microcontrolador reconhece. Não seria de fácil compreensão para o ser humano, pois ele se perde ao tentar ler os grupos de valores binários e tentar interpretá-los como uma instrução. O Assembly é formado por mnemônicos, que são códigos mais fáceis para o ser humano ler, decorar, operar e utilizar no desenvolvimento de programas. (Nicolosi, 2000)

Uma pequena parte do algoritmo deste projeto utiliza a linguagem Assembly. A representação do Assembly é feita por mnemônicos, que são códigos alfanuméricos utilizados para tornar o entendimento dos programadores mais fáceis, sendo assim esta linguagem não é uma linguagem de máquina, como muitas vezes é confundida. Porém, uma instrução em linguagem Assembly é correspondente a uma instrução em linguagem de máquina, tornando-

se uma grande vantagem sobre as linguagens de alto nível que tem seus programas mais extensos. (Gimenez, 2002)

Apesar da linguagem de programação Assembly ser parecida com a linguagem de máquina, até em número de instruções, ela não é uma linguagem de máquina como já foi dito. Sendo assim, após ser elaborado o programa em linguagem em Assembly, temos que compilá-lo utilizando o Assembler, que é o aplicativo responsável em transformar a linguagem Assembly em linguagem de máquina.

Este projeto utiliza essa linguagem como base para o desenvolvimento do programa inserido no protótipo do controlador da disponibilidade das vagas.

3. DESCRIÇÃO DO HARDWARE

O hardware utilizado para construção do protótipo do presente projeto é composto por:

- um kit didático 8051 CMXV2-32K;
- oito sensores reed-switch;
- oito carrinhos equipados com imãs acoplados.

Neste capítulo é descrito o desenvolvimento da maquete e do programa de controle do projeto final. São apresentados os detalhes acerca da construção do programa de controle do estacionamento e do sensor de posição, detalhando as funcionalidades utilizadas para o controle de cada um. Também é detalhada a construção da maquete e toda a sua estrutura.

3.1 . - Kit CMXV2-32K

O kit adquirido é um kit didático utilizado para o desenvolvimento de projetos eletrônicos microcontrolados, o qual possui vários componentes eletrônicos integrados, facilitando os projetos utilizando microcontroladores, bem como o seu aprendizado. O kit dará a você a possibilidade de aplicar na prática absolutamente todos os conceitos envolvendo microcontroladores 8051. O kit é a unidade central de processamento deste protótipo e pode ser visualizado na Figura 3.1.

Figura 3.1 – Kit CMVX2-32K – Fonte: Manual CMVX2-32k

O kit é constituído de um pacote integral (Pack-ControlChip-CMXV.exe) com todos os softwares, além de fonte, cabo e periféricos, assim como todas as informações necessárias para a imediata exploração do kit, sem a necessidade de buscar nenhuma informação, equipamento ou componente extra.

Principais Características do kit:

- Gravação in circuit (usuário grava seu programa direto no kit)
- Memória de programa / Dados de 32Kbytes (Volátil)
- Freqüência de trabalho de aproximadamente 12MHz
- Led indicativo de circuito energizado
- Leds indicativos de modo do kit (Load/Run)
- Acesso ao drive de motor de passo por intermédio de bornes (Chave de fenda comum)

Porém dentre os componentes acima não foi utilizada a gravação in circuit porque a porta que foi responsável no projeto pela leitura dos sensores é a mesma da gravação in circuit, isto é, a porta P2. Sendo a gravação feita externamente. Já os leds indicativos de circuito energizado foram utilizados para uma melhor visualização da disponibilidade das vagas.

Composição do kit :

- 1 Placa microcontrolada CMXV2-32K
- 1 CD ROM com todo o material necessário p/ exploração do kit
- 1 Cabo de comunicação RS232 (PC/KIT)

Periféricos embarcados na placa CMXV2-32K:

- Microcontrolador de 8 Bits (8051) da Atmel AT89S52
- Display LCD 16x2
- RS232 totalmente disponível ao usuário do kit. (Modo Run)

- 8 Leds
- Display de 7 segmentos
- Botões
- Driver para Step Motor
- Acesso aos PORTS do uC, p/ expansão a circuitos externos (Protoboards etc...)
- Memória externa de 32kBytes
- Acesso aos pinos de interrupções externas INT0 e INT1 por Pinhead

Contudo, somente são utilizados neste projeto os componentes a seguir:

Cabo de alimentação de 12V, comunicação serial RS-232, microcontrolador de 8 Bits (8051) da Atmel AT89S52, placa microcontrolada e os 8 leds, que também vem embutido no kit.

DETALHAMENTO DA PLACA

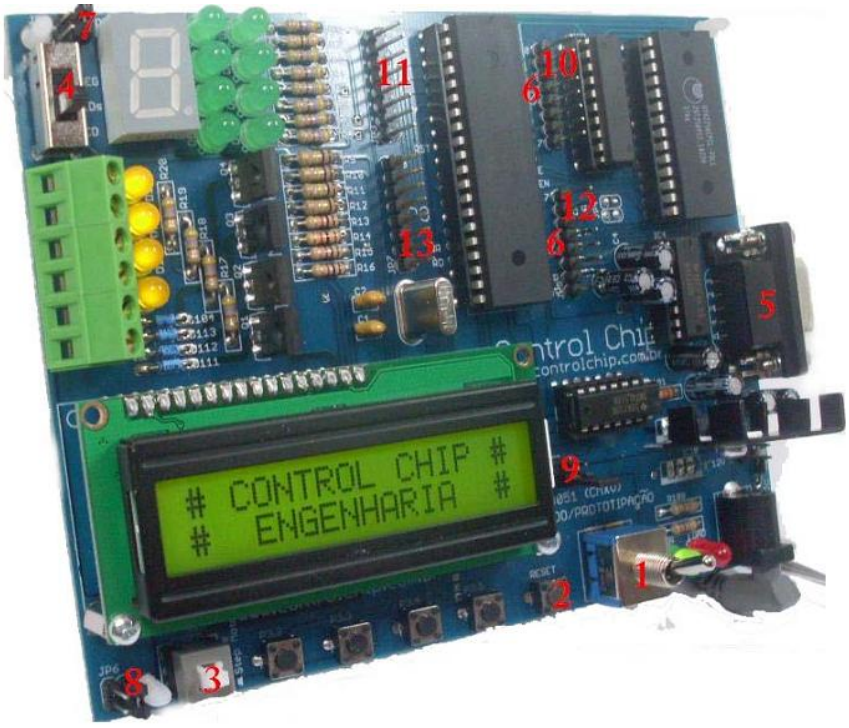

Figura 3.2 – Placa 8051. Fonte: Manual CMXV2-32k

1) CHAVE “LOAD/RUN”

2) BOTÃO “RESET”

3) CHAVE S1 (Seleciona Step Motor ou Botões)

4) CHAVE S2 “3 Posições” (Seleção entre Leds , 7 Segmentos ou Display LCD)

5) CONECTOR CON1 (DB9 FEMEA)

6) OS PORTs P0 e P2 do 8051

7) JUMPER JP2

8) JUMPER JP6

9) DISPLAY DE LCD 16x2

10) CONECTOR JP1

11) CONECTOR JP3

12) CONECTOR JP5

13) CONECTOR JP7

No entanto, são utilizados neste projeto os componentes a seguir: chave Load/Run, o botão RESET, a chave S2, fonte de alimentação é feita pela USB do computador, comunicação serial RS-232 e o conector DB-9 fêmea.

Dentre os vários componentes do kit e funções citados anteriormente, pode-se destacar as portas P0 e P2, pois o kit possui 32Kb de memória de programa/dados, totalmente disponível ao usuário, tanto para dados quanto para programa e as portas P0 e P2 são encarregadas pelo acesso (Dados/Endereçamento) a memória externa, onde foi feita através da P2 a ligação através de fiação com os sensores magnéticos. Como mostra a figura 3.3:

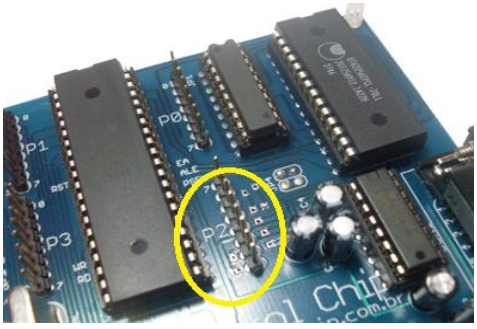

Figura 3.3 – Porta P2 da Placa 8051. Fonte: Manual CMXV2-32k

A CHAVE “LOAD/RUN”, em modo Load: o kit estará “aguardando” a transferência do programa do usuário para em seguida executá-lo e enquanto isso acendendo os LEDs e enviando mensagens pela porta serial. E como o programa foi gravado externamente e não *in circuit*, a chave deverá estar na posição *Load* para carregar o programa sempre que se ligar o kit.

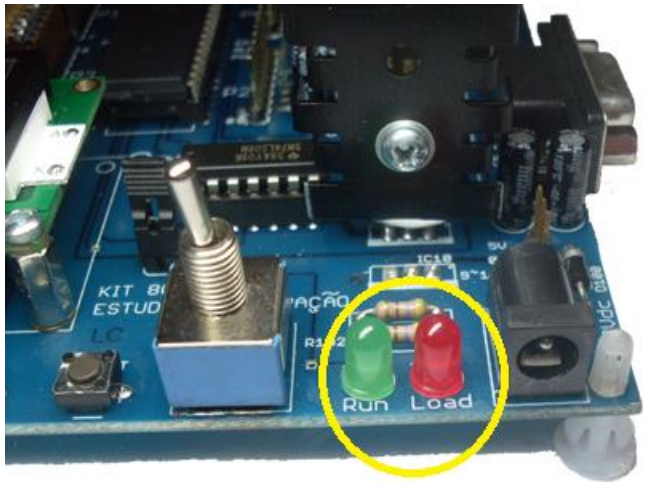

Figura 3.4. – Chave Load/Run da Placa 8051. Fonte: Manual CMXV2-32k

3.1.1. Microcontrolador da Atmel AT89S8253

Por ser um representante da família 8052, esse microcontrolador apresenta maior quantidade de memória RAM e mais um timer/counter de 16 bits que a família 805l básica. Um dos pontos que o torna compatível com o 805l é a região em que estão alojados os registradores especiais, trabalhando assim com palavras de 8 bits. Suas principais características são:

• 8 Kbytes de memória Flash - reprogramável pelo sistema serial de download de programas no formato binário (SPI - Serial Program Interface).

• 2 Kbytes EEPROM.

• Operação estática: OHz até 24MHz.

• 382x8-bit de Memória RAM Interna, sendo 256x8 bytes para os registradores especiais e as regiões A, B e C, e mais 127x8 bytes de extensão, acessados de forma indireta.

• 32 l/O ports programáveis.

• Três 16 bits timer/counters.

• Seis interrupções vetoradas.

• Porta serial programável - UART.

• SPI - Interface serial para reprogramação.

• Dois DPTRs - Data Pointers.

• Por meio da interface serial SPI, após compilarmos o programa, podemos transferí-lo para a região de memória CODE, que nesse microcontrolador é composta por 8 Kbytes de memória Flash. [Nicolosi e Bronzeri, 2005]

O microcontrolador AT89S8253 possui características idênticas ao microcontrolador AT89S8252 exceto a quantidade de memória flash. O AT89S8253 possui 4 KB a mais de memória *flash*, ou seja, 12KB.

Devido principalmente as facilidades que este tipo de memória proporciona, por apresentar as mesmas características citadas em parágrafos anteriores e possuir uma relação custo benefício melhor que o AT89S8252 este será o microcontrolador utilizado para a realização do projeto.

Utilizar o microcontrolador AT89S8253 não causa prejuízo algum para a implementação do projeto e ainda possibilita a vantagem de possuir 4KB a mais de memória flash em caso de necessidade e prevendo uma possível expansão. Apesar de pertencer a família 8052 este chip é perfeitamente compatível com a família 8051 conforme informado acima.

O sistema apresentado é extremamente flexível, pois todas as portas do microcontrolador estão disponíveis ao usuário o que facilita estender as características da placa de acordo com a necessidade.

Esta placa contém um microcontrolador de 8 bits (AT89S8253) que possui internamente 5 interrupções mascaráveis, 2 tempos (timers) de 16 bits programáveis e um

canal serial, que já tem dentro dele uma Flash de 8KB e EEPROM de 2KB. O diagrama apresentado na figura 3.5 demonstra os principais componentes e fluxo de dados:

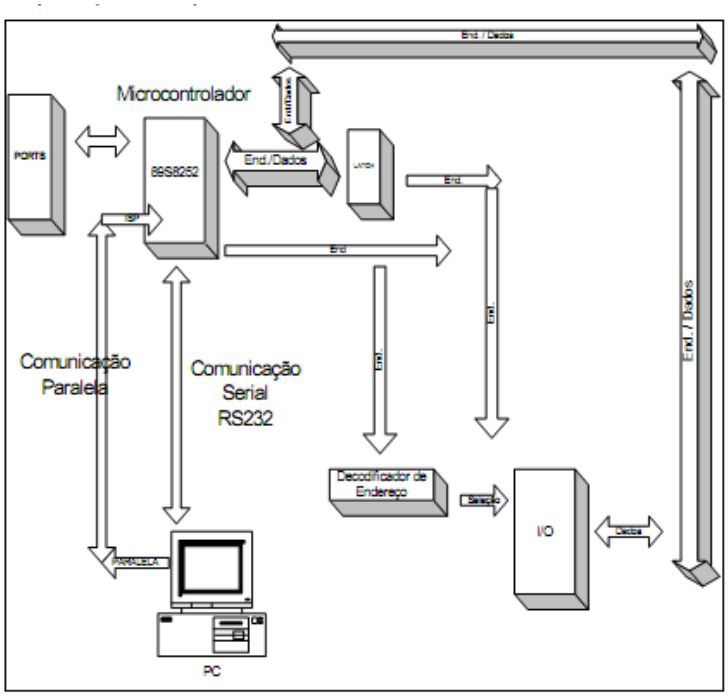

Figura 3.5 - Fluxo de dados do kit de gravação do microcontrolador AT89s8253.

(Fonte: Gimenez, 2002)

O microcontrolador utilizado no kit é o 8051 da Atmel AT89S8253, que contém funções importantes frequentemente utilizadas em projetos eletrônicos envolvendo microcontroladores. A família de Microcontroladores 8051 é ainda a mais usada em todo o mundo, devido ao fato de que são baratas, possuem uma variedade de periféricos e porque qualquer engenheiro conhece e sabe utilizá-la (devido a sua longa jornada no mercado), sendo assim, mesmo se tratando de um microcontrolador de 8 bits, atende ainda uma considerável parte da demanda do mercado.

O microcontrolador AT89S8253 como qualquer outro microcontrolador da família 8051, nada mais é que um microcomputador implementado em um único circuito integrado, no qual estão as unidades básicas de um computador (Gimenez,2002).

Sua importância se dá ao fato da necessidade de controlar e organizar os bits recebidos pelo componente encoder uma vez que eles estão no formato paralelo de 8 bit e necessitam ser transmitidos ao computador via serial.

Conforme informado anteriormente, o microcontrolador utilizado neste projeto foi o AT89s8253 da Atmel. Este é um power-baixo e de alto desempenho CMOS microcontrolador 8-bit com bytes 8K no sistema de memória flash programável. O dispositivo é fabricado com alta densidade da Atmel tecnologia de memória não volátil e é compatível com o conjunto de instruções indústria-padrão 80C51 e pinagem. O Flash no chip de memória permite que o programa a ser reprogramado no sistema ou por um não-volátil Grammer pró-memória convencional. Ao combinar um processador de 8 bits versátil, com sistema de Flash programável em um chip monolítico, o AT89S8253 Atmel é um microcontrolador poderoso que fornece uma solução altamente flexível e rentável para muitas aplicações de controle embarcado.

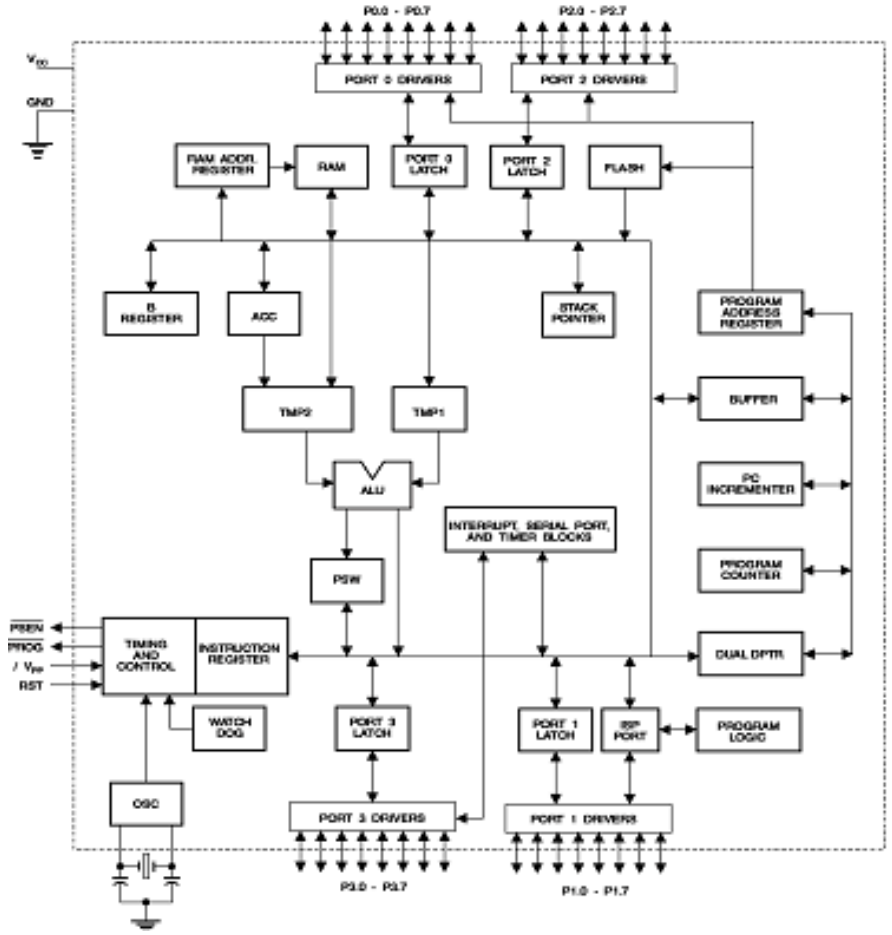

Figura 3.6 – Diagrama de blocos do microcontrolador. (Fonte: Gimenez, 2002)

3.2. REED-SWITCH

Os sensores magnéticos fabricados pela ECP (Eletromatic Controle e Proteção) operam na mesma freqüência do receptor 433,92 Mhz. Sua corrente de consumo é de 10 mA em operação. Uma bateria tipo pilha modelo A23 de 12 V alimenta o circuito.

O mecanismo de disparo deste tipo de sensor é bem simples, porém, muito eficiente, evitando acionamentos do alarme por falhas dos mesmos. Segundo o fabricante, este sensor é uma cápsula de vidro que contém terminais de metal sobrepostos e levemente afastados. Quando o ímã está próximo da parte com o sensor o suficiente para que seu campo magnético exerça influência sobre os contatos, de forma correta, haverá o contato entre os dois, fechando o circuito. Caso haja separação das partes ocorre a separação dos contatos, ocorrendo o envio do sinal ao microcontrolador. Na figura 3.7 é mostrado o sensor e o seu respectivo ímã.

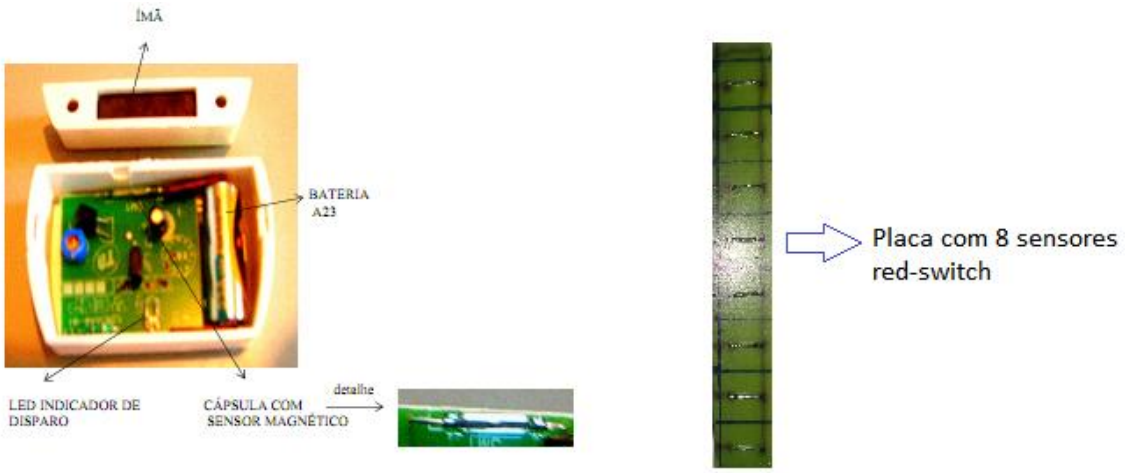

Figura3.7 - Conjunto do sensor magnético, ímã e mais a placa com os 8 sensores utilizada no protótipo. (Autor)

3.3. CARROS COM ÍMÃS

Os carrinhos próprios para maquete utilizados no desenvolvimento do protótipo são facilmente encontrados em qualquer loja de brinquedos ou feira de comércio e possui um custo relativamente baixo, correspondente a R$ 2,00. Já os ímãs encontram-se disponíveis em lojas especializadas, normalmente em materiais elétricos e eletrônicos, estes adquiridos na Contato Eletrônica Ltda. Localizada em Brasília ao preço de R$ 5,00 cada peça. Os mesmos podem ser visualizados na figura 3.8:

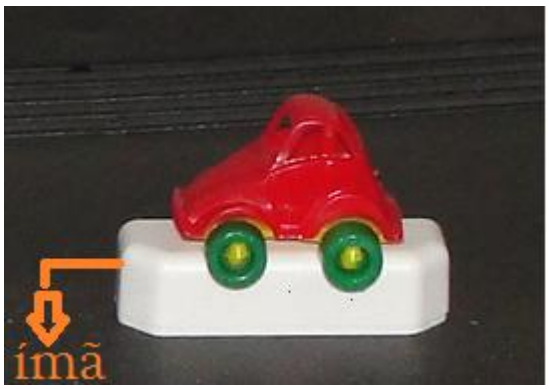

Figura 3.8- Carrinho com ímã acoplado. (Autor)

4. - IMPLEMENTAÇÃO DO PROJETO

Neste capítulo são descritos os processos de implementação do projeto do sistema de identificação automática de veículos e vagas em estacionamentos. São apresentados como foram utilizadas as ferramentas e tecnologias para elaboração do projeto.

Por se tratar de um projeto de cunho acadêmico, muitas das tecnologias sugeridas para a implementação num ambiente real foram substituídas ou simuladas de tal forma que o foco do projeto não seja prejudicado ou diminuído, que é oferecer um mecanismo de identificação de vagas e veículos.

4.1 - Protótipo

Inicialmente foi adquirida uma placa de fenolite com dimensões de 20x4cm, após isso, foi desenhado o circuito na placa, corroendo-se o hipercloreto. Depois, fez-se a furação com a broca de 1mm para a colocação dos fios, sendo de um lado os sensores e do outro conectados à porta do microcontrolador por meio de fiação. Em seguida, escolheu-se na porta P2 do microcontrolador para que ele possa receber o sinal do imã pelos sensores através da comunicação com a placa microcontrolada. Feita todas as ligações elétricas passou-se à programação.

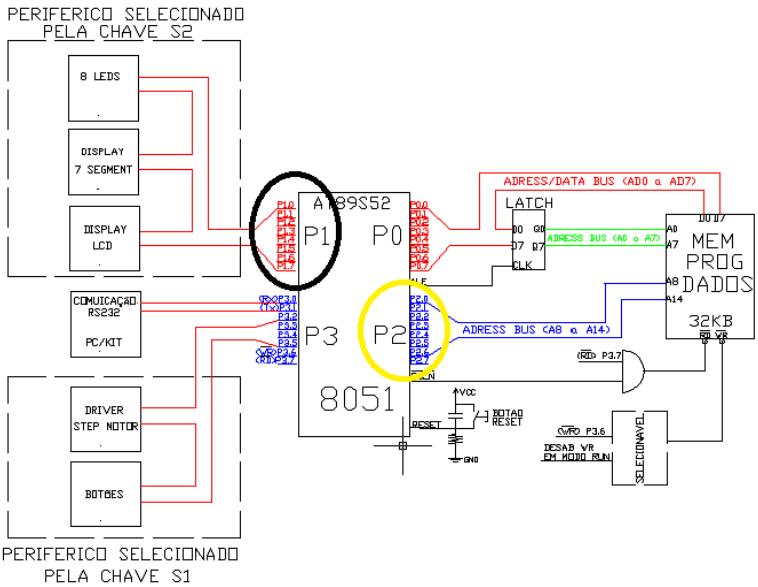

Figura 4.1 - Arquitetura Simplificada do Kit CMVXV2-32K- Fonte: Manual CMVX2-32k

Onde na figura 4.1 através a porta P2 do microcontrolador 8051 destacada em amarelo, foi feita a ligação do pino P2.0 ao sensor 0 com o fio laranja, do pino P2.1 com o sensor 1 com o fio roxo, do pino P2.2 com o sensor 2 com o fio verde, do pino P2.3 com o sensor 3 com o fio vermelho, do pino P2.4 com o sensor 4 com outro fio laranja, do pino P2.5 com o sensor 5 com outro fio roxo, do pino P2.6 com o sensor 6 com outro fio verde, do pino P2.7 com o sensor 7 com outro fio vermelho. E, por fim um fio cinza que faz a ligação elétrica com os demais sensores e é ligado na Vcc da placa com energia entre 9-12v.

O aviso luminoso gerado pelos leds foi feito através do programa em assembly utilizando os pinos correspondentes à porta P2 na porta P1. Como exposto na tabela seguinte:

A comunicação das chaves de acionamento com o AT89s8253 é feita por fios comuns como detalhado acima de laboratórios eletrônicos. As chaves, de 0 a 7 são ligadas na PORTA P2 do microcontrolador, na disposição de acordo com o Quadro 4.1.

Quadro 4.1 – Ligação dos sensores, pinos e leds do AT89S8253.

Sensor (vaga)	Pino da Porta P2	Posição do Led
0	P2.0	D1
1	P2.1	D2
2	P2.2	D3
3	P2.3	D4
4	P2.4	D5
5	P2.5	D6
6	P2.6	D7
7	P2.7	D8

(Fonte: Autor)

As ligações detalhadas acima podem ser visualizadas na figura 4.2, onde se tem as ligações do microcontrolador por meio de fios, no quadrinho maior destacado em amarelo, e a ligação do fio cinza, destacado no quadrinho menor em amarelo, no pino IC10 que é um regulador de tensão da placa 8051. E o cabo branco ligado plugado no JP8, circulado em laranja, faz a ligação com o circuito externo (laptop nesse caso).

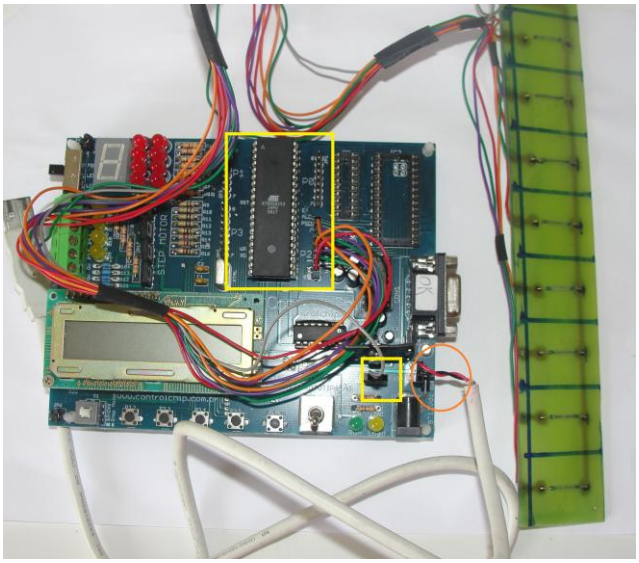

Figura 4.2 – Ligações do Microcontrolador com os Sensores. (Autor)

4.1.1 - Fluxograma do Programa Assembly

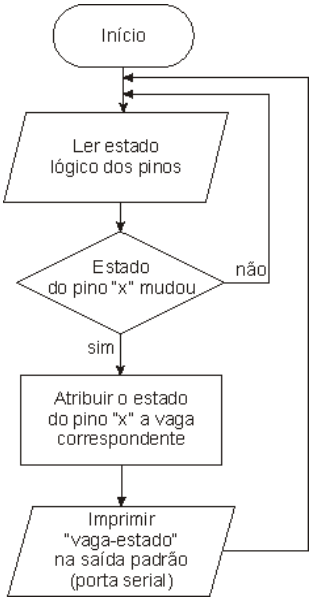

Figura 4.3 – Fluxograma do Programa em Assembly. (Autor)

O fluxograma resume a idéia geral do programa controlador do estacionamento. O código funciona da maneira mais simples possível e faz jus aos objetivos fixados na proposta inicial deste projeto.

Primeiramente são estabelecidas as configurações para gravação no microcontrolador. Depois é feito a definição e inicialização das variáveis e constantes internas. As entradas e saídas são configuradas e, em seguida, é realizada a definição das funções utilizadas no programa. O programa é inicializado ao pressionar o botão reset na placa do microcontrolador e então se inicia a rotina principal. Onde o estado do pino que é enviado pela serial é de led ligado ou desligado

4.1.2 - Programação do Kit 8051

O código do kit 8051 foi desenvolvido no compilador assembler Jen's File Editor. Foi utilizada a linguagem assembly. Com a programação devidamente concluída, compilou-se o programa para a verificação de eventuais erros, onde o compilador transforma o código em um arquivo de instruções seqüenciais de extensão *.bin.

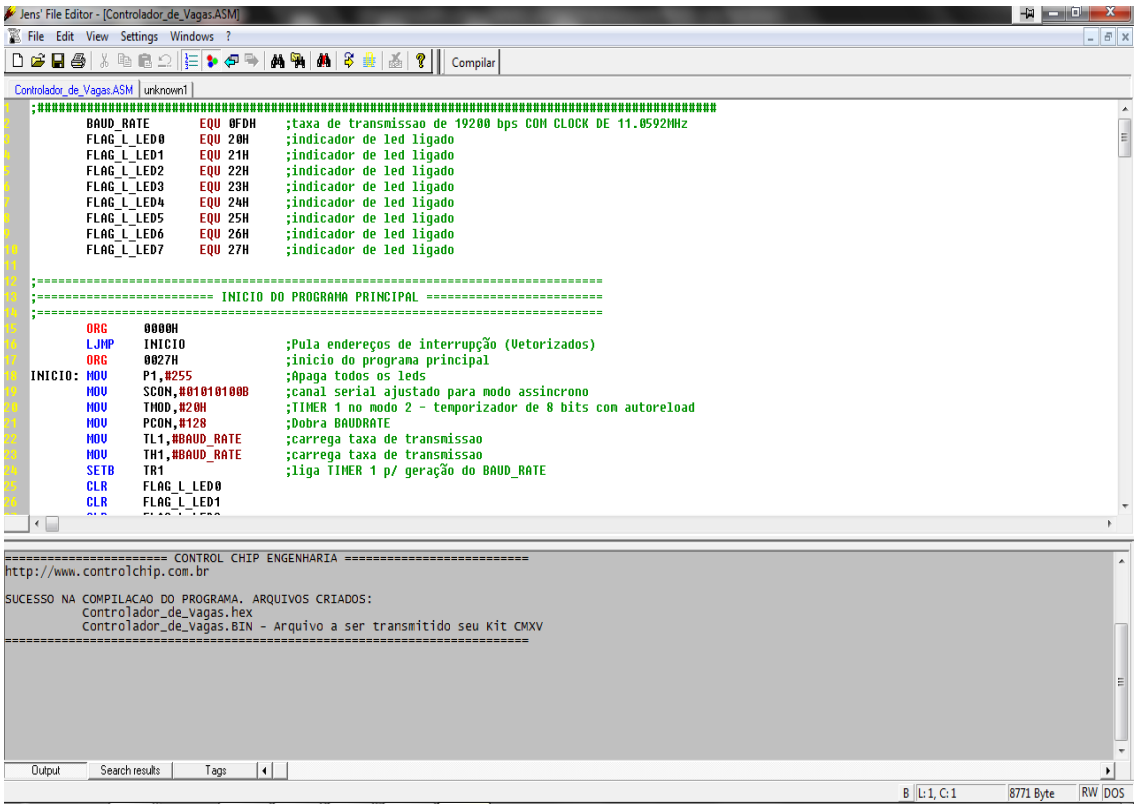

Figura 4.4 – Compilação do software do Controlador de Vagas. (Autor)

Feito isso, é preciso gravar na memória do 8051 e finalmente carregar para o microcontrolador. Depois de ter compilado o programa e tendo gerado o arquivo com extensão .bin já é possível gravar este programa no microcontrolador. Para a execução desta tarefa foi escolhido o programa Grava, versão 2.11 também disponível no CD anexo ao kit do microcontrolador. Neste programa é possível selecionar o chip utilizado dentre uma lista de chips compatíveis, conforme Figura 4.5.

Figura 4.5 – Início da gravação do AT89S8253. (Autor)

Para gravar o arquivo no chip do microcontrolador, basta clicar no botão Gravar que pode ser visto na Figura 4.5. E selecionar o arquivo com extensão .bin para ser gravado no chip, o detalhe da tela de seleção é visto na Figura 4.6.

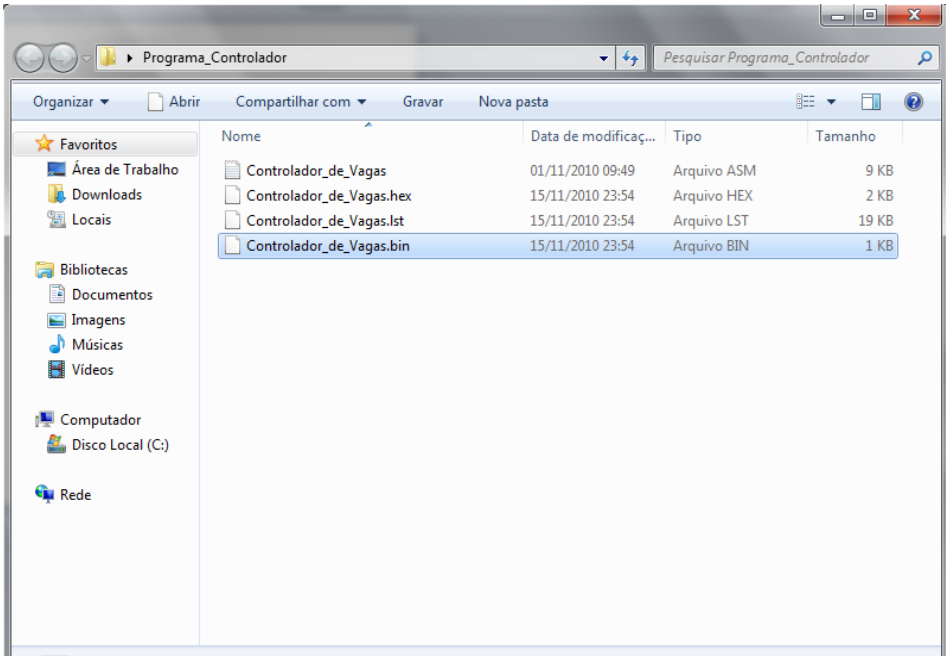

Figura 4.6 – Programa Controlador em binário. (Autor)

Após a escolha do arquivo basta aguardar a conclusão da gravação conforme Figura 4.7.

Figura 4.7 – Fim da gravação do AT89S8253. (Autor)

4.2 - PROGRAMA CDVE

Foi desenvolvido um software na linguagem C ++ com o nome 'CDVE-Controlador de Disponibilidade de Vagas em Estacionamento' na plataforma C++ Builder 6.0 Enterprise. Este programa é executado em plano de fundo (execução contínua) e possui as seguintes funções:

- Receber a entrada de dados na porta serial;

- Gravar os estados das vagas em banco de dados;

- Enviar para a tela a informação do painel.

Como o sistema microcontrolado envia como saída para a porta serial apenas os eventos (alteração dos estados das vagas), o programa gestor recebe na porta serial essas informações e realiza um tratamento, para identificar quais vagas sofreram alteração e as grava no banco de dados. Logo em seguida, este programa exibe em tela as vagas disponíveis, de acordo os estados atuais de cada vaga no banco de dados.

A Figura 4.8 apresenta o fluxograma da função principal do programa.

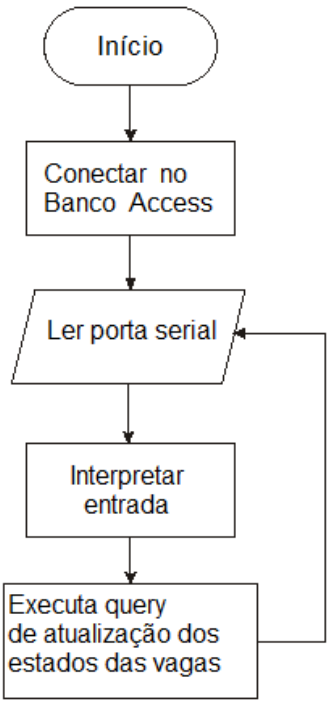

Figura 4.8 – Fluxograma CDVE (Autor)

Após a gravação em banco de dados e exibição do resultado em tela, o programa retorna para o início, onde é feita a leitura da porta serial. Como o sistema foi desenvolvido usando o programa C++Builder 6.0 Enterprise, a porta serial pode ser lida como um arquivo, neste caso, os dados enviados à porta serial são armazenado em ".txt".

Os códigos-fonte para o programa 'CDVE' e a geração do banco de dados das vagas encontram-se a seguir.

4.2.1 -DESENVOLVIMENTO DO CDVE

O código abaixo é o principal responsável por alimentar o banco de dados. Ele recebe a sequência de caracteres da porta serial, analisa essa sequência de caracteres, verifica se ela é valida, processa-a e altera o banco de dados. Criando novos registros, atualizando registros antigos e ligando de desligando os botões-lâmpada do software.

```
//Evento onChange é disparado quando o protótipo envia uma sequência
//de caracteres para o computador
void __fastcall TForm1::Memo1Change(TObject *Sender)
{
    //Variáveis
    String Texto; //Vai receber o valor lido
    int Vaga; //Vai receber a vaga
    TDateTime Hora; //Vai receber a hora do evento
    Boolean booLigado;
    Boolean booTeste1;
    Boolean booTeste2;
    Boolean booTeste3;

    Texto = Memo1->Text;
    //Compara o primeiro caracter com L e guarda o resultado  em booTeste
    booTeste1 = Texto.SubString(1,1) == 'L' ? True : False;
    booTeste2 = Texto.Length() >= 15 ? True : False;
    booTeste3 = Texto.Length() <= 18 ? True : False;
```

```
MemLog->Lines->Append(DateToStr(Date()) + (String)" " + (String)TimeToStr(Time()) +
(String)">" + (String)Texto);
        Memo1->Lines->Clear();

        if (booTeste1 && booTeste2 && booTeste3) {
            Vaga = StrToInt(Texto.SubString(6,1));
            booLigado = Texto.SubString(8,1) == 'L' ? True : False;
         Hora = StrToDateTime(DateToStr(Date()) + (String)" " + (String)TimeToStr(Time()));
```

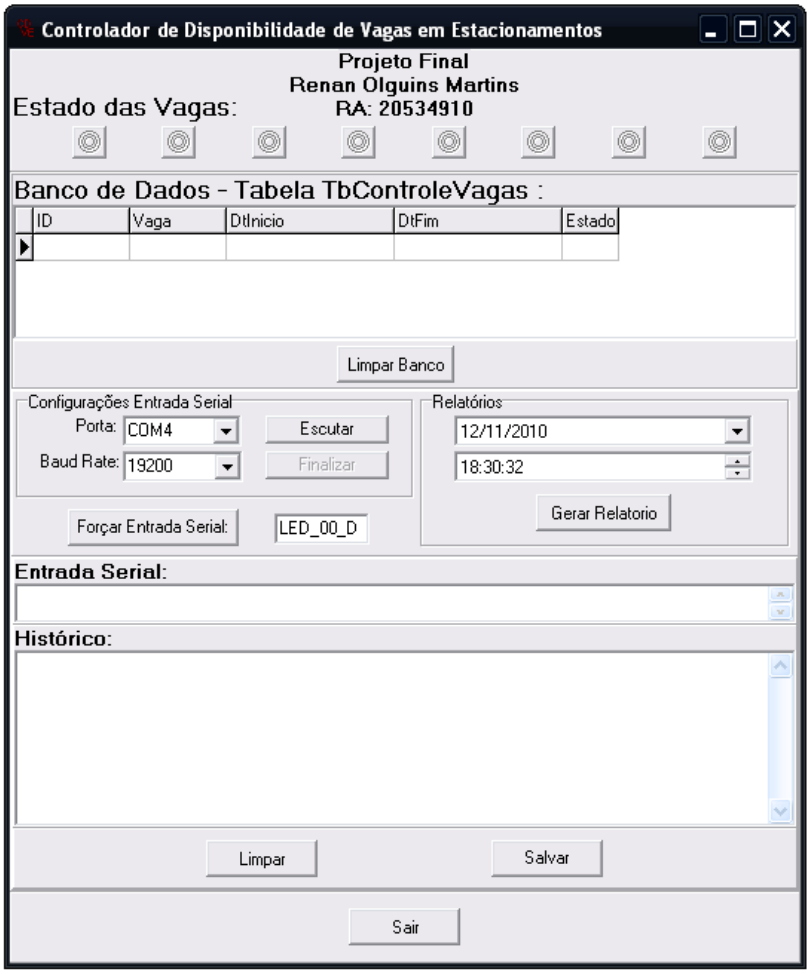

Figura 4.9 – Interface Gráfica do programa CDVE (Autor)

Após a codificação acima responsável pela validação dos caracteres enviados pela serial, abaixo será mostrado a rotina, comentada, para alimentar o Banco de Dados:

```
//Rotinas para alimentar o banco de dados
    if (!booLigado){ //Mandei Desligar
        if (VerificaEstadoTrue(Vaga)) {//Se a vaga estiver true = ocupada = com carrinho
            // É porque já está com DtInicio
            //Preenche DtFim e
            //Põe Estado para False
            ADOQry->Edit();
            ADOQry->FieldByName("DtFim")->AsDateTime = Hora;
            ADOQry->FieldByName("Estado")->AsBoolean = False;
            ADOQry->Post();
            SwitchLampadas(Vaga,False);
        }
        else // Já estava desligado = sem carrinho = false
        {
            //Não faz nada
        }
    }
    else
    { //Mandei Ligar
        if (VerificaEstadoTrue(Vaga)) {// Já estava ligado = true = com carrinho
            //Não faz nada
        }
        else // Não estava ligado ainda = Nao tava ocupado = sem carrinho
        {
            //Cria Novo Registro
            //Seta Estado para True
            //Seta DtInicio para a hora atual
            ADOQry->Append(); //Novo registro
            ADOQry->FieldByName("Vaga")->AsInteger = Vaga; //Grava a vaga
            ADOQry->FieldByName("DtInicio")->AsDateTime = Hora; // Grava a hora
            ADOQry->FieldByName("Estado")->AsBoolean = True; //Grava o estado para true =
ocupado
            ADOQry->Post(); //Escreve o registro
            SwitchLampadas(Vaga,True); //Liga o botao-lampada de numero "vaga"
        }
    }
```

Posteriormente a verificação do status da vaga de ocupada ou desocupada, à confirmação dos horários de uso, abaixo seguirá a função responsável pelo acionamento e desligamento dos botões-lâmpada do programa:

```
//Essa função auxiliar liga ou desliga os botões-lâmpada de determinada vaga.
SpeedButton1 é um botão.
A situação ligado ou desligado é na verdade a propriedade Enabled (Habilitado)
O estado é um dos parâmetros da função = booEstado.
A vaga também é um desses parâmetros = nVaga;

void   __fastcall TForm1::SwitchLampadas(int nVaga,Boolean booEstado){
    switch (nVaga) {
        case 0:{
            SpeedButton1->Enabled = booEstado;
        } break;
        case 1:{
            SpeedButton2->Enabled = booEstado;
        } break;
        case 2:{
            SpeedButton3->Enabled = booEstado;
        } break;
        case 3:{
            SpeedButton4->Enabled = booEstado;
        } break;
        case 4:{
            SpeedButton5->Enabled = booEstado;
        } break;
        case 5:{
            SpeedButton6->Enabled = booEstado;
        } break;
        case 6:{
            SpeedButton7->Enabled = booEstado;
        } break;
        case 7:{
            SpeedButton8->Enabled = booEstado;
        } break;
    }
}
```

E por fim, apresentada abaixo, a rotina, também comentada, responsável pela geração de relatórios, onde estes são armazenados em .txt para uma melhor visualização do usuário do programa.

```
void __fastcall TForm1::BtnRelatorioClick(TObject *Sender)
{
    TDateTime dtPesquisa;
    int i;
    if (DlgSave->Execute()){
        //O Relatório sai em formato texto e deve ser salvo em algum lugar
```

```
        DateTimePicker2->Date = DateTimePicker1->Date;
        dtPesquisa = DateTimePicker2->DateTime;

        ADOQryRel->Close();
        ADOQryRel->SQL->Clear();
        ADOQryRel->SQL->Append("Select * from TbControleVagas");
        ADOQryRel->SQL->Append("Where [DtInicio]  <= " + (String)DateTimeForSQL(dtPesquisa));
        ADOQryRel->SQL->Append("AND [DtFim] >= " +  (String)DateTimeForSQL(dtPesquisa));
        ADOQryRel->Open();

        //ShowMessage(ADOQryRel->SQL->Text);

        // Com a pesquisa feita prepara o relatorio
        MemR->Lines->Clear();
        MemR->Lines->Append("Relatório de Vagas");
        MemR->Lines->Append("Relatório Gerado em:");
        MemR->Lines->Append("Data: " + (String) DateToStr(Date()));
        MemR->Lines->Append("Hora: " + (String) TimeToStr(Time()));
        MemR->Lines->Append("----------");
        MemR->Lines->Append("Situação do Estacionamento em");
        MemR->Lines->Append("Data: " + (String) DateToStr(DateTimePicker1->Date));
        MemR->Lines->Append("Hora: " + (String) TimeToStr(DateTimePicker2->Time));
        MemR->Lines->Append("");

    if  (ADOQryRel->RecordCount > 0){

        ADOQryRel->First();
        for (i=0;i<ADOQryRel->RecordCount;i++)
        {
        MemR->Lines->Append("Vaga " + (String) IntToStr(ADOQryRel->FieldByName("Vaga")-
>AsInteger) + (String) " Ocupada");
        ADOQryRel->Next();
        }
    }
    else
        MemR->Lines->Append("Nessa data e hora todas as vagas estavam desocupadas");



    MemR->Lines->SaveToFile(DlgSave->FileName);
    }
            }
```

4.2.2. Banco de Dados Access

Foi utilizado um banco de dados chamado BDControleVagas.mdb cujo arquivo deve estar sempre junto ao executável . O banco de dados consiste em uma única tabela chamada tbControleVagas que possui as seguintes colunas:

ID - Chave primária – Autoincremento – É apenas um campo que serve de código para cada registro. É uma chave única. Não tem nenhum significado tão importante no programa.

Vaga - Guarda o número da vaga;

DtInicio - Guarda a data e a hora de inicio em que o carro foi estacionado = sensor foi ligado pela primeira vez;

FtFim - Guarda a data e a hora de inicio em que o carro saiu = sensor desligado. Estado = Guarda o estado atual da vaga → False = Desocupado True = Ocupado. Encontra-se no funcionamento do programa CDVE descrito no item acima, juntamente com o código-fonte para a geração do banco de dados 'tbControleVagas'.

tbControleVagas

ID	Vaga	DtInicio	DtFim	Estado	Adicionar Novo Campo
62	0	12/11/2010 19:35:15	13/11/2010 09:48:43	☐	
63	1	12/11/2010 19:35:17	12/11/2010 19:35:26	☐	
64	2	13/11/2010 19:35:18	12/11/2010 19:35:24	☐	
65	2	12/11/2010 19:35:26	12/11/2010 19:35:29	☐	
66	3	12/11/2010 19:35:27	12/11/2010 19:35:53	☐	
67	5	12/11/2010 19:35:32	12/11/2010 19:35:56	☐	
68	0	12/11/2010 19:35:36	12/11/2010 19:35:38	☐	
69	0	12/11/2010 19:35:39	12/11/2010 19:35:41	☐	
70	4	12/11/2010 19:35:50	12/11/2010 19:35:54	☐	
71	0	13/11/2010 09:48:56		☐	
72	1	13/11/2010 09:49:10		☐	
73	7	13/11/2010 09:49:46		☐	
* (Novo)	0			☐	

Registro: 4 de 12 Sem Filtro Pesquisar

Figura 4.10 – tbControleVagas do CDVE (Autor)

4.3. Funcionamento do Protótipo

Com os equipamentos adequados em mãos, é necessário fazê-los interagir. Para que este conjunto identifique a disponibilidade das vagas, o papel do microcontrolador é de fundamental importância, pois o mesmo é a central lógica e de controle do protótipo. O atmel estabelece e comanda todos os recursos necessários para o funcionamento adequado do protótipo, de acordo com a programação inserida em sua memória. Com isso, como se pode ver na figura 4.11 tem-se o funcionamento do protótipo com dois sensores ocupados e pode-se ver os dois leds correspondentes acesos também com a chave no modo load.

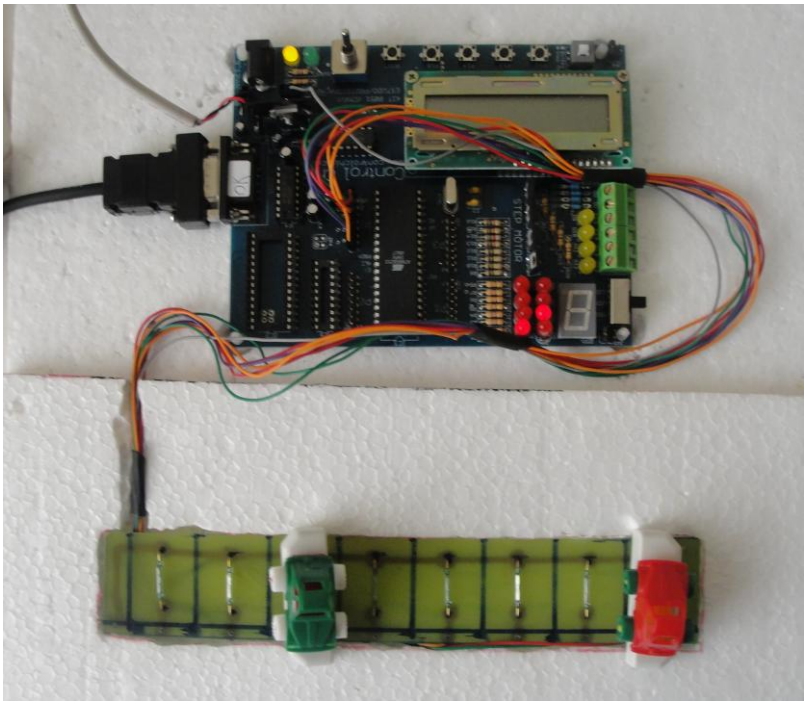

Figura 4.11 – Protótipo em funcionamento. (Autor)

Concomitantemente, apareceu no programa CDVE a seguinte tela mostrada na figura 4.12. Com os leds do programa CDVE acionados representando os leds reais que seriam avisos luminosos para os usuários indicando que a vaga esta ocupada.

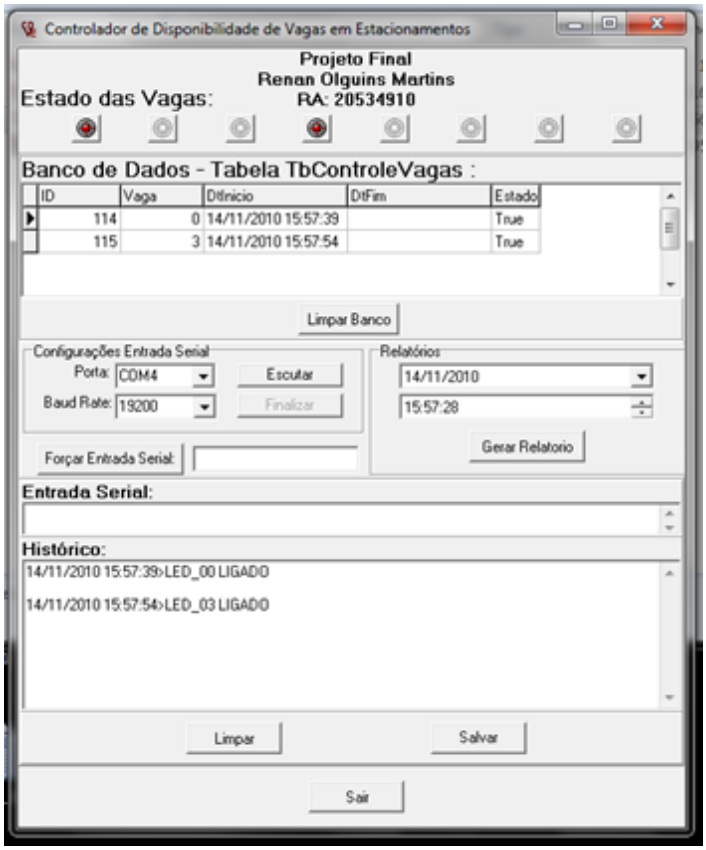

Figura 4.12 – Programa CDVE em funcionamento 1. (Autor)

O programa desenvolvido permite que o microcontrolador controle o funcionamento dos *leds* da placa, através do sinal de entrada medido pelo sensor magnético. Este sinal é enviado ao atmel, após o sensor identificar o campo magnético gerado pelo ímã acoplado no carrinho, é mostrado no *led* de posição correspondente à do sensor por acender a lâmpada e se manter continuamente até a retirada do carro. Concomitantemente, o microcontrolador envia pela porta serial, através de sua programação em *assembly* inserida, qual a posição do led que está aceso e o seu status (ligado ou desligado). A partir desses dados o programa CDVE filtra e trabalha essas informações para poder gerar relatórios de entrada e saída de veículos assim como tempo de uso das vagas. Esse relatório sendo possível por uma tabela no Access.

Deste modo, quando o automóvel estaciona na vaga é detectado pelo sensor, que gera um aviso luminoso pelo led de posição correspondente e envia pela porta serial a posição do(s) led(s) e seu status (ligado ou desligado). Na Figura 4.13 é ilustrada a informação enviada do protótipo ao computador pela porta serial.

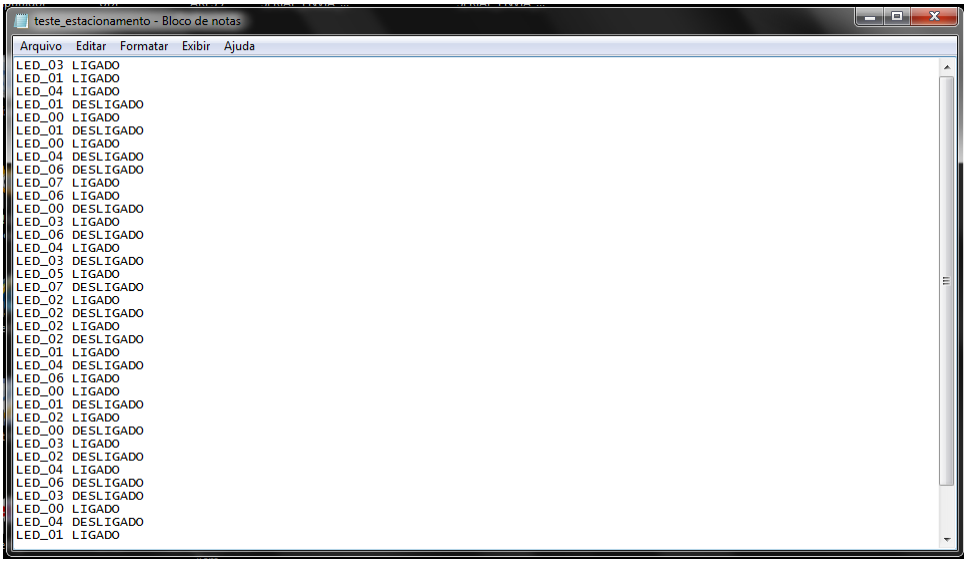

Figura 4.13 – Dados enviados do protótipo para a porta serial. (Autor)

Com isso, esses dados foram posteriormente trabalhados pelo programa CDVE para simulação e controle dos dados de entrada e saída de veículo. Este procedimento é repetido infinitamente, a partir do momento em que o protótipo é ligado no computador. Na figura 4.14 é verificado a interface gráfica do programa CDVE em funcionamento.

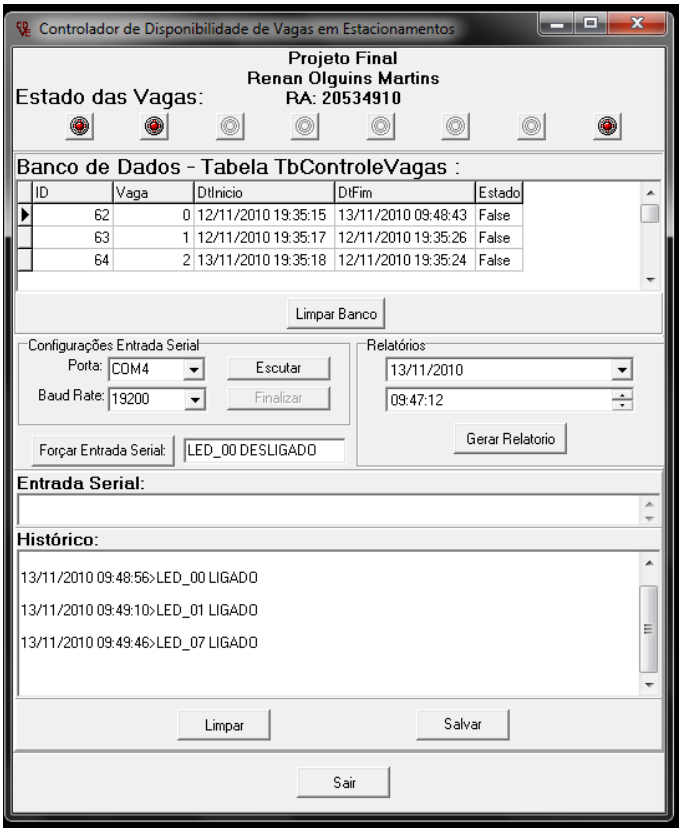

Figura 4.14 – Programa CDVE em funcionamento 2. (Autor)

Após explicado todo o funcionamento, na próxima figura é mostrado o protótipo em funcionamento, todo interligado e já com sua maquete pré-projetada. Primeiramente, observa-se a placa microcontrolada em funcionamento com os sensores e carrinhos com ímãs que foram embutidas na maquete. E, posteriormente, seguindo a ordem de montagem o protótipo completo em perfeito funcionamento, conforme figura 4.15.

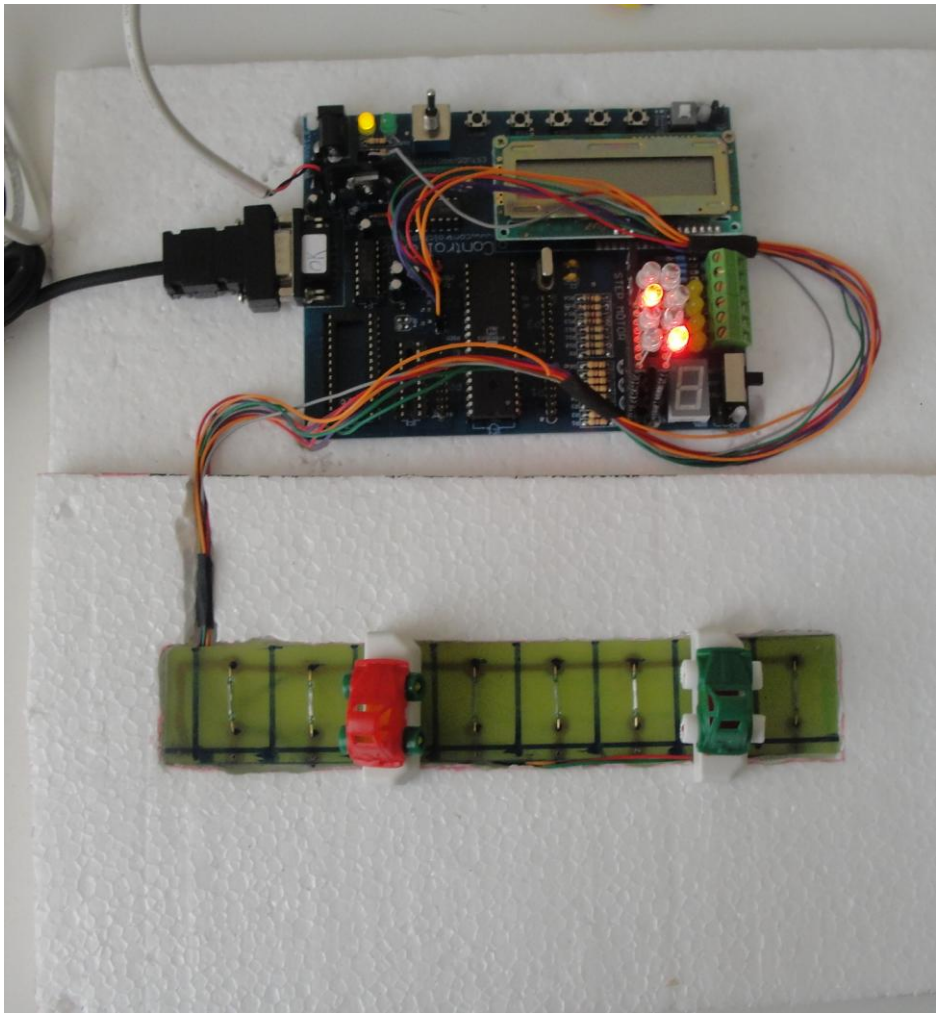

Figura 4.15 – Protótipo interligado e montado (Autor)

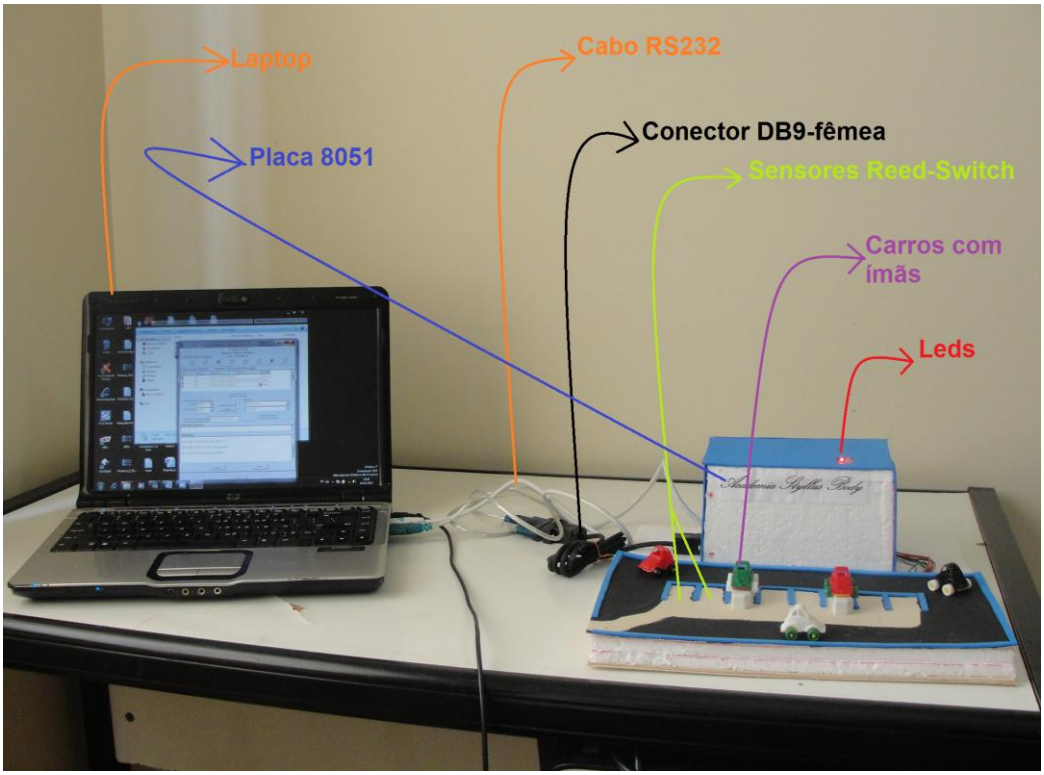

Figura 4.16 – Protótipo montado e funcionando com a localização dos recursos. (Autor)

Na figura 4.16 é mostrado a todos os equipamentos do protótipo funcionando com os recursos implementados. Sem esquecer que os leds foram adaptados em forma de "canudos" para que fosse visto na demonstração, assim como os sensores reed-switch estão sobrepostos pelo piso do estacionamento, como também a placa 8051esta inserida na caixa de isopor, como vistos na figura 4.15. Já na tela do computador o programa mostrou a seguinte tela, visualizada na figura 4.17:

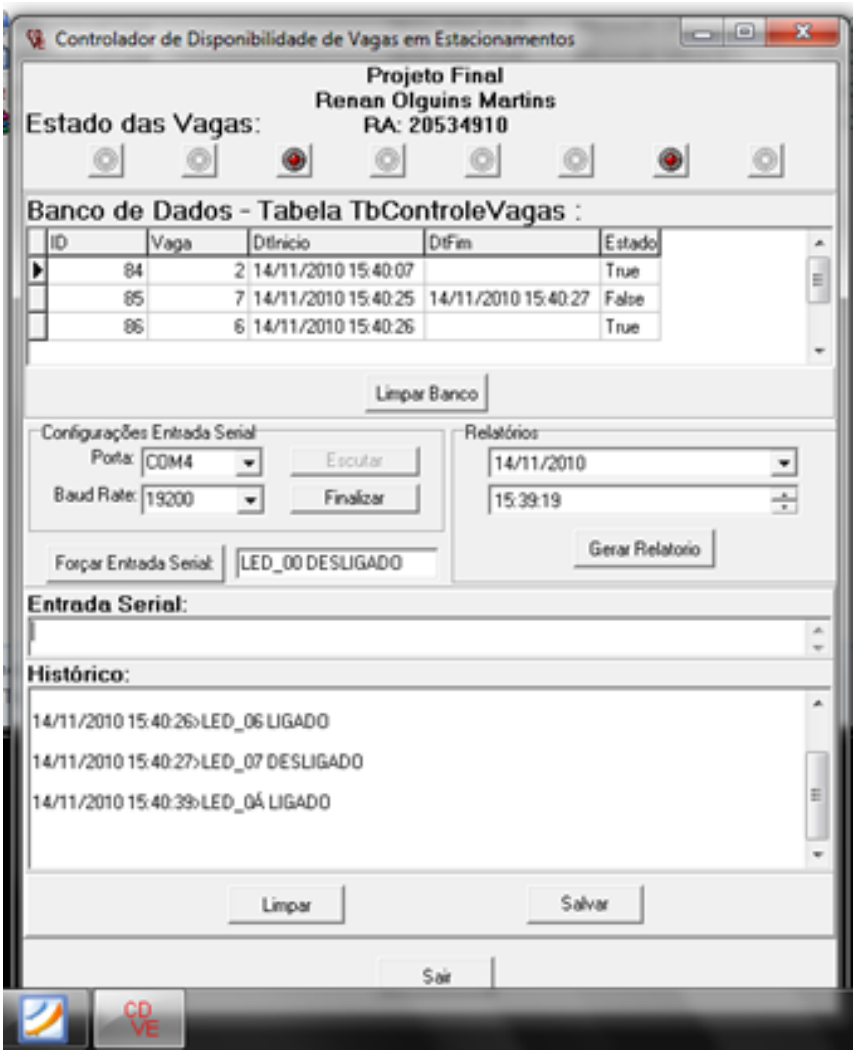

Figura 4.17 – CDVE. (Autor)

Onde pode-se ver os botões acesos em vermelhos simbolizando os leds para o gestor. O banco de dados sendo alimentado em cada evento novo, verificado no "ID". As configurações da porta COM4 e a baudrate de 19200, como estabelecidas no programa controlador. O botão escutar ativado, pois os dados estavam sendo transmitidos pela serial. Posteriormente, pressiona-se o botão "Gerar Relatório" para verificar o uso em determinado período.

4.4 Dificuldades encontradas

Para a realização deste projeto, as dificuldades encontradas foram de vários tipos e intensidades, umas de fácil resolução, outras mais complexas.

A primeira dificuldade foi a definição do escopo do projeto, como seria a estrutura do sistemas e as tecnologias específicas a serem utilizadas. Com o avanço nos estudos e tentativas de implementação, idéias foram surgindo e aos poucos, a forma final foi sendo definida.

Para a realização deste projeto, várias dificuldades tiveram que ser superadas. Primeiramente a aquisição de um kit 8051 que atendesse às especificações do projeto, não foi possível no mercado local e sua aquisição se deu através da internet.

A escolha dos equipamentos foi uma dificuldade de rápida resolução, porém na fase de testes tive problema com o ímã. O programa responsável pela leitura serial foi trocado por duas vezes, no entanto só foi solucionado quando mudei o ímã circular por um retangular de duas pontas.

Uma das maiores dificuldades foi a implementação dos circuitos simuladores e sua comunicação com o microcontrolador 8051. Com o auxílio do manual de instruções do kit CMVX2-32k e de estudos sobre a família 8051 e sua programação assembly, foi possível estabelecer o circuito e implementar o código.

A escolha dos softwares de compilação dos programas do microcontrolador também representou uma certa dificuldade. Após alguma pesquisa e indicações da orientadora, foi possível definir o ambiente de trabalho para programação no AT89s8253.

A compreensão dos códigos e a estruturação de exemplo do CDVE também se tornou uma dificuldade, que logo foi superada através de estudos mais aprofundados da linguagem C++ Builder.

5. TESTES E RESULTADOS OBTIDOS

A implementação deste projeto exige que sejam executados uma série de testes tanto de funcionalidade como de eficácia. São analisados uma série de situações específicas que envolvem o comportamento dos motoristas e as possíveis conseqüências para a organização do estacionamento.

Após a montagem do protótipo a partir do hardware especificado, possibilitando a identificação de automóveis e o devido controle adequado do *reed-switch*, iniciaram-se os testes, levando-se em conta os objetivos inicialmente propostos.

O quadro a seguir representam essas situações e o resultado obtido pela resposta das identificações automáticas.

O quadro 5.1 mostra os testes das possíveis situações no programa de saída CDVE. A relação do ímã e veículo, neste ponto é automática, ou seja, ao se referir a veículo, se entende a presença do campo magnético no sensor.

Quadro 5.1 – Testes no CDVE.

Situação	Registro
Veículo estaciona na vaga	Registro da hora, data e evento (entrada)
Veículo sai da vaga	Registro da hora, data e evento (saída)
Veículo permanece na vaga por tempo indeterminado	Sem ação. Último registro: Entrada
Veículo nunca mais retorna à vaga	Sem ação. Último registro: Saída

(Fonte: Autor)

Todos os testes realizados para a busca do objetivo foram analisados de acordo com a proposta inicial do projeto e mostraram-se satisfatórios ao controle das vagas, uma

análise por meio de relatórios de tempos escolhidos aleatoriamente permitindo a um gerenciamento eficaz por meio desse controle de dados.

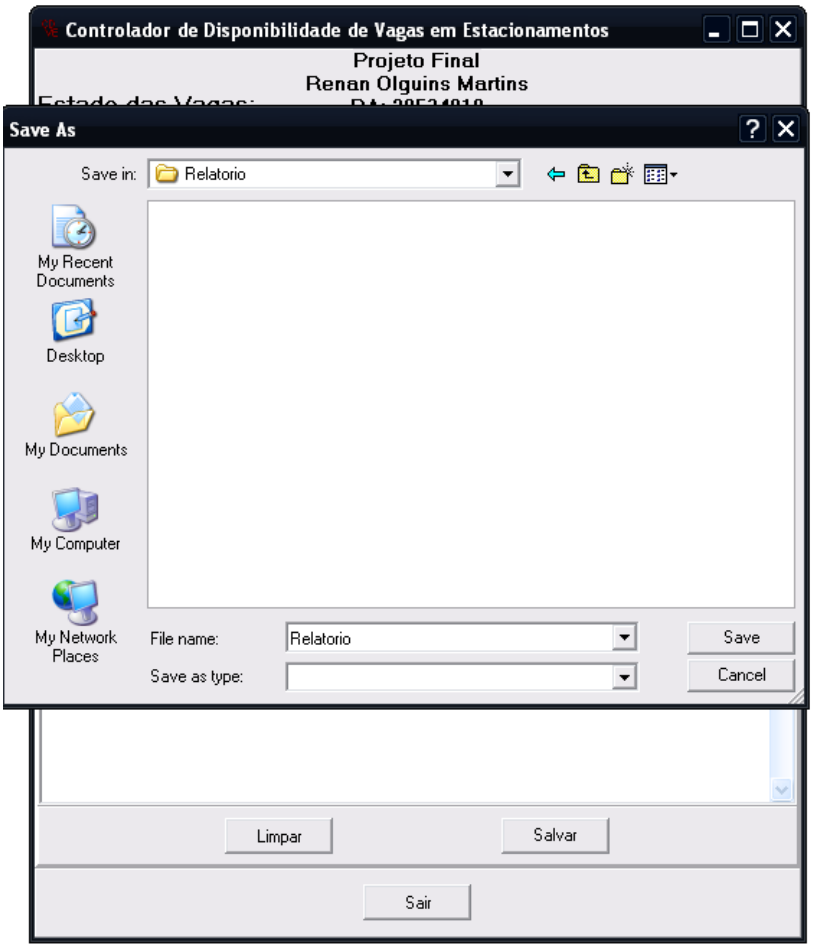

Figura 5.1 – Gerar relatório no CDVE (Autor)

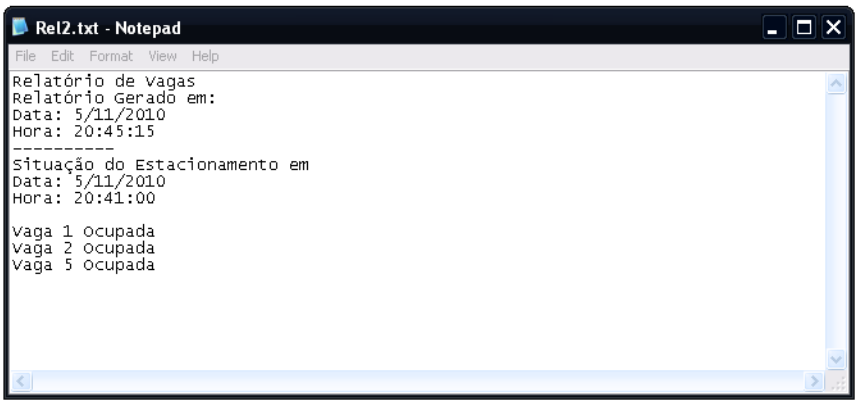

Figura 5.2 – Relatório do CDVE (Autor)

A proposta inicial definiu o controle de vagas em um estacionamento através de sensores, avisando se esta ocupada ou não. Como também um relatório de verificação das vagas através de um tempo escolhido. De forma de que fosse visualizada em uma maquete de um estacionamento com as vagas e os carrinhos para demonstração como se pode ver na figura 5.3.

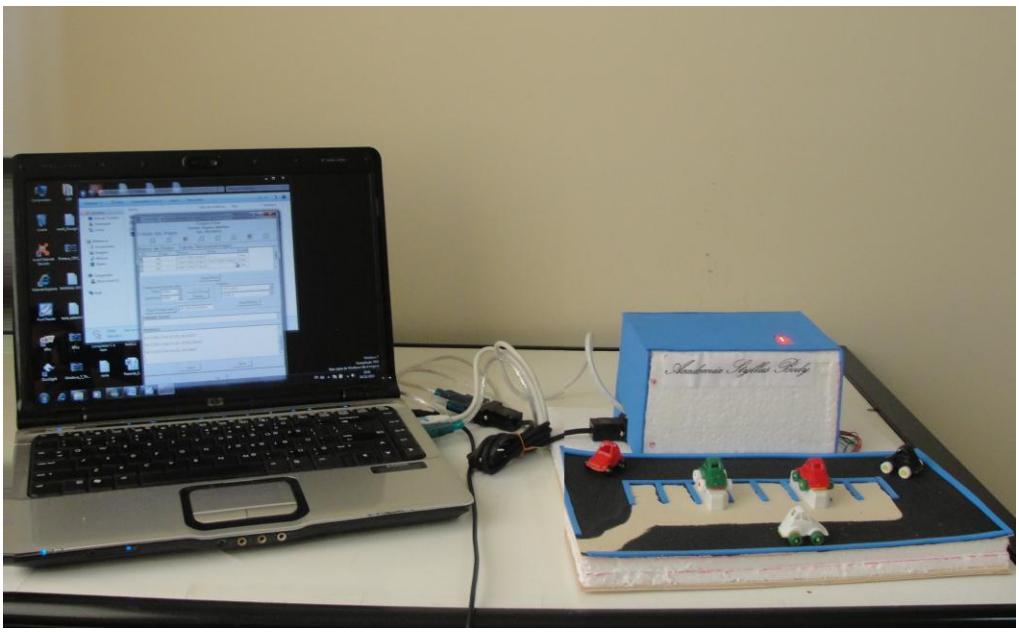

Figura 5.3 – Protótipo Final. (Autor)

6. CONCLUSÃO

Este projeto teve como finalidade o desenvolvimento e a construção de um estacionamento microcontrolado, onde o protótipo identifica a presença do veículo e informa a base de dados para que se possa ter um relatório para efeitos de estatística ou estudo de viabilidade de acordo com a faixa de tráfego de automóveis periodicamente no estacionamento em estudo.

Com o exposto e o realizado, conclui-se que os objetivos propostos foram atingidos, pois o protótipo desenvolvido é capaz de identificar a presença do veículo na vaga e gerir, através de um software, os dados, como: o fluxo diário por data, o tempo de uso e dados de entrada e saída de veículos.

O programa se comportou da maneira esperada, fazendo o controle correto do reed-switch, a partir do sinal de entrada capturado e enviado pelo sensor de identificação. O microcontrolador de 8 Bits (8051) da Atmel AT89S8253 analisou e interpretou corretamente a programação gravada em sua memória, fazendo com que acendesse o LED da placa no instante em que o sensor de presença identificasse o veículo, mostrando qual vaga foi ocupada e através de um programa que possibilita a transmissão da porta serial do protótipo para a USB do laptop, envie esses dados a um software que as contabiliza em um banco de dados para futuros relatórios.

O circuito controlador de vagas em estacionamento implementado é um projeto acadêmico, desenvolvido de forma artesanal, porém com grande potencial comercial, no entanto, necessita de ajustes e um maior investimento para ser comercializado de forma otimizada. Entretanto espera-se que, de alguma maneira, este trabalho possa trazer benefícios para os empresários do setor de comércio, como também em estacionamentos públicos, exemplificados nos estacionamentos rotativos presentes em várias cidades do nosso país, onde há um controle manual por parte do governo das vagas no centro da cidade, o que poderia ser feito de forma automatizada.

A tecnologia dos sensores magnéticos para identificar presença e posição é largamente utilizada no meio industrial e já é uma tecnologia consolidada e madura, que traz uma grande precisão para os sistemas em que é empregado. No contexto veicular, o seu uso

seria inviável, pois, há soluções melhores para o ambiente real, como o infra-vermelho e o sensor de bobina aberta, porém academicamente o emprego do reed-switch foi o mais aconselhável pelo uso de maquete, isopor e outros materiais que prejudicariam a sensibilidade dos sensores efetivamente comerciais.Assim como a sua ligação poderia ser sem fiação, dependendo do caso. Além de apresentar as principais tecnologias envolvidas, este trabalho sugere uma integração, conseguida através do emprego de outras tecnologias não menos importantes para o contexto, mas que desempenham um papel coadjuvante, como o sistema microcontrolado que utiliza a família 8051 com o assembly e as aplicações desenvolvidas em linguagem C++ para o sistema CDVE (Gestor).

6.1. - Integração de disciplinas

Para a realização deste projeto, foi indispensável a utilização de conhecimentos que não foram explicitados, como por exemplo, as configurações necessárias para a compilação e execução dos programas desenvolvidos, porém estes conhecimentos não podem ser desconsiderados e devem ser levados em consideração para a implantação do sistema proposto.

É possível relacionar as disciplinas do curso de Engenharia de Computação que mais tiveram impacto de acordo os conhecimentos utilizados:

- Microcontroladores e microprocessadores;
- Sistemas Distribuídos;
- Linguagem Técnica de Programação I e II;
- Circuitos Eletrônicos;
- Circuitos e Máquinas Elétricas;
- Lógica Digital;
- Física III e IV;

Entre os conhecimentos externos, adquiridos fora do ambiente de graduação, pode-se destacar:

- Tópicos avançados de programação em C++ Builder;

- Estudos aprofundados sobre o Atmel 8051 e linguagem assembly;

6.2 - Propostas de projetos futuros

Neste ponto é interessante ressaltar que existe uma enorme variedade de aplicações das tecnologias expostas para outros contextos. Com a mesma topologia (ou com algumas diferenças), é possível que um engenheiro ou projetista desenvolva sistemas totalmente diferentes e/ou mais completos. A seguir, são apresentadas propostas para projetos futuros.

O circuito controlador de disponibilidade em vagas de estacionamento descrito neste projeto, apesar de completamente funcional e dentro dos objetivos propostos, aceita diversos aperfeiçoamentos.

Como proposta de trabalhos futuros, pode-se citar a melhoria deste protótipo através do acréscimo de catracas, caso necessite, utilizando outras tecnologias, sem fiação, um painel com *leds* para o gestor, e um para o usuário visualizar em cada vaga. Como também um circuito de iluminação com direcionamento a vaga disponível, onde o gestor, caso deseje possa cadastrar mensalistas, diaristas, funcionários, etc.

Além disso, pode-se utilizar o sensor na entrada no caso de um estacionamento fechado que possa contabilizar já na entrada para controle geral. Pode-se também acrescentar mais pisos, contendo vagas delimitadas aos portadores de deficiência e idosos, onde caso o carro posicionado na vaga pré-delimitada ao uso específico seja ocupada por outrem gere um sinal sonoro para a evacuação da mesma ou aplicação da sanção devida.

Outra função interessante a ser acrescida, seria possibilitar a integração com atuadores, como travas para portas e veículos, motores e outros tipos de acionamentos.

Pode-se ainda desenvolver uma maneira para a integração de detectores para controle de ambientes, como com a utilização de câmeras para segurança e identificação pela alteração dos pixels na imagem.

REFERÊNCIAS BIBLIOGRÁFICAS

- ALBUQUERQUE, Pedro U. B.; THOMAZINI, Daniel. **Sensores Industriais – Fundamentos e Aplicações.** 4ª ed. São Paulo: Érica, 2007.

-WERNECK, Marcelo Martins Werneck. **Transdutores e Interfaces.** 1ª ed. Rio de Janeiro: LTC, 1996.

-GIMENEZ, P. S. **Microcontroladores 8051.** 1ª ed. Prentice Hall, 2002.

-NICOLOSI, Denys E. C. **Laboratório de Microcontroladores Família 8051: Treino de Intruções, Hardware e Software.** São Paulo: Editora Érica,2000

-NICOLOSI, Denys Emílio Campion. **Microcontrolador 8051 Detalhado**. São Paulo: Érica, 2000.

-SICA, Carlos. **Sistemas Automáticos com Microcontroladores 8031/8051.** São Paulo: Novatec, 2006.

-JANDL JUNIOR, Peter. **Introdução ao C++**. 1ª ed. São Paulo: Futura, 2003.

-MANZANO, José Augusto N. G. **Estudo Dirigido C++ Builder 6**, Ed. Érica Ltda, 2003.

-MATEUS, César Augusto. **Guia Prático C++ Builder 5**. 2ª. ed. Érica Ltda., 2002.

-ZELENOVSKY, Ricardo; Alexandre Mendonça. **Microcontroladores: Programação e Projeto com a Família 8051.** Rio de Janeiro: MZ, 2005.

-NICOLOSI, DENYS E. C. – **MICROCONTROLADOR 8051 DETALHADO** – ED. ÉRICA, 2004.

-MOHR, Hari Bruno : Werner Kraus Jr. : Raimes Moraes; Artigo Acadêmico; Universidade Federal de Santa Catarina; **A família de microcontroladores 8051.** janeiro/2004. Disponível em (agosto 2009).

-NUNES, Paulo; Artigo: **Conceito de Fluxograma**. Disponível em http://www.knoow.net/cienceconempr/gestao/fluxograma.htm. Acessado em agosto de 2010

- KIT 8051: **"Manual completo Kit CMXV2-32K Plus 6"**. Disponível no CD da ControlChip Engenharia Ltda.

- CORREIOWEB: **"Aumento na frota de veículos do DF"**. Disponível em http://www.correiobraziliense.com.br/app/noticia182/2009/07/03/cidades,i=123434/FROTA+DE+VEICULOS+DO+DF+E+UMA+DAS+MAIS+NOVAS.shtml. Acessado em agosto de 2009

- SENSOR REED-SWITCH: **"Apostila do sensor reed-switch"**. Disponível em http://www.if.ufrgs.br/mpef/mef004/20061/Cesar/SENSORES-Reed-switch.html - acessado em setembro de 2010

- SENSOR REED-SWITCH: **"Funcionamento do sensor reed-switch".** Disponível em http://www.reed-sensor.com/Notes/General_Reed_Switch_Theory.htm. Acessado em setembro de 2010

- C++ BUILDER: **"Curso de C++ Builder".** Disponível em http://www.dicasbcb.com.br/downloads/curso%20basico%20c++builder%20dicasbcb.pdf. Acessado em outubro de 2010

- SENSOR MAGNÉTICO: **"Curso sobre automação utilizando sensores".** Disponível em http://www.mecatronicaatual.com.br/secoes/leitura/51. Acessado em outubro de 2010

APÊNDICE A

Código-fonte em assembly do microcontrolador

```
;#############################################################################
####################
    BAUD_RATE      EQU 0FDH ;taxa de transmissao de 19200 bps COM CLOCK DE
11.0592MHz
    FLAG_L_LED0    EQU 20H          ;indicador de led ligado
    FLAG_L_LED1    EQU 21H          ;indicador de led ligado
    FLAG_L_LED2    EQU 22H          ;indicador de led ligado
    FLAG_L_LED3    EQU 23H          ;indicador de led ligado
    FLAG_L_LED4    EQU 24H          ;indicador de led ligado
    FLAG_L_LED5    EQU 25H          ;indicador de led ligado
    FLAG_L_LED6    EQU 26H          ;indicador de led ligado
    FLAG_L_LED7    EQU 27H          ;indicador de led ligado

;=============================================================
==============
;======================== INICIO DO PROGRAMA PRINCIPAL
========================
;=============================================================
==============
    ORG    0000H
    LJMP   INICIO           ;Pula endereços de interrupção (Vetorizados)
    ORG    0027H            ;inicio do programa principal
INICIO: MOV    P1,#255          ;Apaga todos os leds
    MOV    SCON,#01010100B    ;canal serial ajustado para modo assincrono
    MOV    TMOD,#20H        ;TIMER 1 no modo 2 - temporizador de 8 bits com
autoreload
    MOV    PCON,#128        ;Dobra BAUDRATE
```

```
    MOV    TL1,#BAUD_RATE     ;carrega taxa de transmissao
    MOV    TH1,#BAUD_RATE     ;carrega taxa de transmissao
    SETB   TR1                ;liga TIMER 1 p/ geração do BAUD_RATE
    CLR    FLAG_L_LED0
    CLR    FLAG_L_LED1
    CLR    FLAG_L_LED2
    CLR    FLAG_L_LED3
    CLR    FLAG_L_LED4
    CLR    FLAG_L_LED5
    CLR    FLAG_L_LED6
    CLR    FLAG_L_LED7
    MOV    P2,#255

INICI: ACALL  ENTRA0          ;rotina rensponsavel pela leitura dos sensores(entrada do
carro)
INIC:  ACALL  SAIDA0          ;rotina rensponsavel pela leitura dos sensores(saida do
carro)
INI:   MOV    R7,#0           ;variavel de inicializacao do timer
    ACALL  TIMER_1            ;rotina de tempo de 2s
    SJMP   INICI              ;retorna o laço para sempre verificar a leitura dos sensores
;************************************************
;Rotina de acionamento do led atraves da presença do ima (carro entrando)
L_LED0: JB     FLAG_L_LED0,ENTRA1 ;desvia para proxima verificação
    SETB   FLAG_L_LED0            ;ativa a flag de indicação de led aceso
    CLR    P1.0                   ;acende led 0
    MOV    R1,#30H                ;joga o valor '0' para o registrado para ser
utilizado na comunicacao serial
    JMP    PULO                   ;desvio para envio da msg de led aceso pela serial
L_LED1: JB     FLAG_L_LED1,ENTRA2
    SETB   FLAG_L_LED1
    CLR    P1.1
    MOV    R1,#31H
```

```
        JMP     PULO
L_LED2: JB      FLAG_L_LED2,ENTRA3
        SETB    FLAG_L_LED2
        CLR     P1.2
        MOV     R1,#32H
        JMP     PULO
L_LED3: JB      FLAG_L_LED3,ENTRA4
        SETB    FLAG_L_LED3
        CLR     P1.3
        MOV     R1,#33H
        JMP     PULO
L_LED4: JB      FLAG_L_LED4,ENTRA5
        SETB    FLAG_L_LED4
        CLR     P1.4
        MOV     R1,#34H
        JMP     PULO
L_LED5: JB      FLAG_L_LED5,ENTRA6
        SETB    FLAG_L_LED5
        CLR     P1.5
        MOV     R1,#35H
        JMP     PULO
L_LED6: JB      FLAG_L_LED6,ENTRA7
        SETB    FLAG_L_LED6
        CLR     P1.6
        MOV     R1,#36H
        JMP     PULO
L_LED7: JB      FLAG_L_LED7,INIC
        SETB    FLAG_L_LED7
        CLR     P1.7
        MOV     R1,#37H
PULO:   ACALL   TIMER_1
        ACALL   TIMER_1
```

```
        ACALL   TIMER_1
        ACALL   TIMER_1
        ACALL   TIMER_1
        ACALL   LED             ;envia pela serial 'LED_R1'o registrador é carregado com o valor
da posição do led
        ACALL   LIGA            ;envia pela serial 'LIGAdo'
        JMP     INIC
;************************************************
;Rotina de leitura dos sensores(entrada do carro)
ENTRA0: JNB     P2.0,L_LED0
ENTRA1: JNB     P2.1,L_LED1
ENTRA2: JNB     P2.2,L_LED2
ENTRA3: JNB     P2.3,L_LED3
ENTRA4: JNB     P2.4,L_LED4
ENTRA5: JNB     P2.5,L_LED5
ENTRA6: JNB     P2.6,L_LED6
ENTRA7: JNB     P2.7,L_LED7
        RET
;************************************************
;Rotina de leitura dos sensores(saida do carro)
SAIDA0: JB      P2.0,D_LED0
SAIDA1: JB      P2.1,D_LED1
SAIDA2: JB      P2.2,D_LED2
SAIDA3: JB      P2.3,D_LED3
SAIDA4: JB      P2.4,D_LED4
SAIDA5: JB      P2.5,D_LED5
SAIDA6: JB      P2.6,D_LED6
SAIDA7: JB      P2.7,D_LED7
        RET
;************************************************
;Rotina de desligamento do led atraves da retirada do ima (carro saindo)
D_LED0: JNB     FLAG_L_LED0,SAIDA1;desvia para proxima verificação
```

```
        CLR     FLAG_L_LED0             ;desativa a flag de indicação, led apagado
        SETB    P1.0                    ;apaga led 0
        MOV     R1,#30H                 ;joga o valor '0' para o registrado para ser
utilizado na comunicacao serial
        JMP     PULO1                   ;desvio para envio da msg de led apagado pela serial
D_LED1: JNB     FLAG_L_LED1,SAIDA2
        CLR     FLAG_L_LED1
        SETB    P1.1
        MOV     R1,#31H
        JMP     PULO1
D_LED2: JNB     FLAG_L_LED2,SAIDA3
        CLR     FLAG_L_LED2
        SETB    P1.2
        MOV     R1,#32H
        JMP     PULO1
D_LED3: JNB     FLAG_L_LED3,SAIDA4
        CLR     FLAG_L_LED3
        SETB    P1.3
        MOV     R1,#33H
        JMP     PULO1
D_LED4: JNB     FLAG_L_LED4,SAIDA5
        CLR     FLAG_L_LED4
        SETB    P1.4
        MOV     R1,#34H
        JMP     PULO1
D_LED5: JNB     FLAG_L_LED5,SAIDA6
        CLR     FLAG_L_LED5
        SETB    P1.5
        MOV     R1,#35H
        JMP     PULO1
D_LED6: JNB     FLAG_L_LED6,SAIDA7
        CLR     FLAG_L_LED6
```

```
        SETB    P1.6
        MOV     R1,#36H
        JMP     PULO1
D_LED7: JNB     FLAG_L_LED7,PULO2
        CLR     FLAG_L_LED7
        SETB    P1.7
        MOV     R1,#37H
PULO1:  ACALL   TIMER_1
        ACALL   TIMER_1
        ACALL   TIMER_1
        ACALL   TIMER_1
        ACALL   TIMER_1
        ACALL   LED
        ACALL   DESL
PULO2:  JMP     INI
;************************************************
;Rotina de envio da palavra 'LED_0r1'
LED:    MOV     A,#'L'
        ACALL   ENVIA
        MOV     A,#'E'
        ACALL   ENVIA
        MOV     A,#'D'
        ACALL   ENVIA
        MOV     A,#'_'
        ACALL   ENVIA
        MOV     A,#'0'
        ACALL   ENVIA
        MOV     A,R1
        ACALL   ENVIA
        MOV     A,#' '
        ACALL   ENVIA
        RET
```

```
;************************************************
;Rotina de envio da palavra 'LIGADO ou DESLIGADO'
DESL:   MOV     A,#'D'
        ACALL   ENVIA
        MOV     A,#'E'
        ACALL   ENVIA
        MOV     A,#'S'
        ACALL   ENVIA
LIGA:   MOV     A,#'L'
        ACALL   ENVIA
        MOV     A,#'I'
        ACALL   ENVIA
        MOV     A,#'G'
        ACALL   ENVIA
        MOV     A,#'A'
        ACALL   ENVIA
        MOV     A,#'D'
        ACALL   ENVIA
        MOV     A,#'O'
        ACALL   ENVIA
        MOV     A,#0DH;enter
        ACALL   ENVIA
        MOV     A,#0AH;enter
        ACALL   ENVIA
        RET
;************************************************
;Rotina para envio de bits pela serial caregada no buffer
ENVIA:  MOV     SBUF,A
ESPERA: JNB     TI ,ESPERA          ;AGUARDA TERMINO DA TRANSMISSÃO
        CLR     TI
        RET
;************************************************
```

```
;Rotina de tempo de 2s aproximadamente
TIMER_1:        CJNE    R7,#0,INICIO_TIMER_1
                MOV     R7,#1
INICIO_TIMER_1: MOV     R5,#250
                MOV     R6,#1
REPETE_TEMPO:   NOP     ;1uS
                NOP     ;1uS
                NOP     ;1uS
                NOP     ;1uS
                NOP     ;1uS
                NOP     ;1uS
                NOP     ;1uS
                NOP     ;1uS
                NOP     ;1uS
                NOP     ;1uS
                NOP     ;1uS
                NOP     ;1uS
                NOP     ;1uS
                NOP     ;1uS
                NOP     ;1uS
                NOP     ;1uS
                NOP     ;1uS
                NOP     ;1uS
                NOP     ;1uS
                NOP     ;1uS
                NOP     ;1uS
                NOP     ;1uS
                NOP     ;1uS
                DJNZ R5,REPETE_TEMPO
                MOV R5,#250
                DJNZ R6,REPETE_TEMPO
                DJNZ R7,INICIO_TIMER_1
                RET

END
```

APÊNDICE B

Comunicação Serial

Os codigos abaixo foram adaptados de
http://maikel.galeon.com/serie/bcbcomm.html#EXAMPLE

Eles são responsáveis pela comunicação serial entre o computador e o protótipo.

Não usei o código abaixo, ele serviria para transmitir caracteres para o protótipo.

```
void __fastcall TForm1::Memo1KeyPress(TObject *Sender, char &Key)
{
  // TRANSMITS ANYTHING TYPED INTO THE MEMO AREA.

  TransmitCommChar(hComm, Key);

  // THIS PREVENTS TYPED TEXT FROM DISPLAYING GARBAGE ON THE SCREEN.
  // IF YOU ARE CONNECTED TO A DEVICE THAT ECHOES CHARACTERS, SET
  // Key = 0 WITHOUT THE OTHER STUFF.

  if(Key != 13 && (Key < ' ' || Key > 'z')) Key = 0;
}
```

O escutar fica verificando se chegou dados na porta serial

```
void __fastcall TForm1::BtnEscutarClick(TObject *Sender)
{
  char chPorta[5];
  char chConf[15];
```

```
strcpy(chPorta,CBPorta->Text.c_str());
strcpy(chConf,CBBaud->Text.c_str());
strcat(chConf,",N,8,1");


DCB dcbCommPort;

// OPEN THE COMM PORT.
// REPLACE "COM2" WITH A STRING OR "COM1", "COM3", ETC. TO OPEN
// ANOTHER PORT.

hComm = CreateFile(chPorta,
                GENERIC_READ | GENERIC_WRITE,
                0,
                0,
                OPEN_EXISTING,
                0,
                0);

// IF THE PORT CANNOT BE OPENED, BAIL OUT.

if(hComm == INVALID_HANDLE_VALUE){
     ShowMessage("Porta Inválida");
}
else
{
//Código adaptado de Fonte
// SET THE COMM TIMEOUTS IN OUR EXAMPLE.
GetCommTimeouts(hComm,&ctmoOld);
ctmoNew.ReadTotalTimeoutConstant = 100;
ctmoNew.ReadTotalTimeoutMultiplier = 0;
ctmoNew.WriteTotalTimeoutMultiplier = 0;
```

```
  ctmoNew.WriteTotalTimeoutConstant = 0;
  SetCommTimeouts(hComm, &ctmoNew);

  // SET BAUD RATE, PARITY, WORD SIZE, AND STOP BITS.
  // THERE ARE OTHER WAYS OF DOING SETTING THESE BUT THIS IS THE
EASIEST.
  // IF YOU WANT TO LATER ADD CODE FOR OTHER BAUD RATES, REMEMBER
  // THAT THE ARGUMENT FOR BuildCommDCB MUST BE A POINTER TO A
STRING.
  // ALSO NOTE THAT BuildCommDCB() DEFAULTS TO NO HANDSHAKING.

  dcbCommPort.DCBlength = sizeof(DCB);
  GetCommState(hComm, &dcbCommPort);
  BuildCommDCB(chConf,&dcbCommPort);
  SetCommState(hComm, &dcbCommPort);

  // ACTIVATE THE THREAD. THE FALSE ARGUMENT SIMPLY MEANS IT HITS
THE
  // GROUND RUNNING RATHER THAN SUSPENDED.
  Escutando = True;
  BtnEscutar->Enabled = !Escutando;
  BtnFinaliza->Enabled = Escutando;

  ReadThread = new TRead(false);
  }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::BtnFinalizaClick(TObject *Sender)
{
if (Escutando){ //Se estiver escutando, para de escutar
 // Esse código foi adaptado de Fonte
 // TERMINATE THE THREAD.
```

```
  ReadThread->Terminate();
  // WAIT FOR THREAD TO TERMINATE,
  // PURGE THE INTERNAL COMM BUFFER,
  // RESTORE THE PREVIOUS TIMEOUT SETTINGS,
  // AND CLOSE THE COMM PORT.
  Sleep(250);
  PurgeComm(hComm, PURGE_RXABORT);
  SetCommTimeouts(hComm, &ctmoOld);
  CloseHandle(hComm);
  // Fim do Código

  Escutando = False;
  BtnEscutar->Enabled = !Escutando;
  BtnFinaliza->Enabled = Escutando;

  }
}
```