CENTRO UNIVERSITÁRIO DE BRASÍLIA -UniCEUB

CURSO DE ENGENHARIA DE COMPUTAÇÃO

Luciana Ferreira Melo

RFID em sistemas de segurança em prédios

Orientador: José Julimá Bezerra Junior

Brasília

Julho, 2010

Luciana Ferreira Melo

RFID em sistemas de segurança em prédios

Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB) como pré-requisito para a obtenção de Certificado de Conclusão de Curso de Engenharia de Computação.
Orientador: Prof. José Julimá Bezerra Junior

Brasília

Julio, 2010

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

_____

Prof. Abiezer Amarilia Fernandez
Coordenador do Curso

Banca Examinadora:

_____

Prof. José Julimá Bezerra Junior - Mestrado em Engenharia Elétrica - Instituto Militar de Engenharia - Rio de Janeiro - RJ
Orientador

_____

Prof. Miguel Arcanjo Bacellar Goes Telles Júnior - Doutorado em geologia - processamento de dados e análise ambiental - Universidade de Brasília - Brasília – DF

_____

Prof. Vera Lúcia Farini Alves Duarte - Mestrado Matemática - Universidade de Brasília - Brasília – DF

_____

Prof. Thiago de Miranda Leão Toribio
Mestrado em física teórica - Universidade de Brasília - Brasília - DF

Dedico este trabalho ao meu pai Floriano Melo por ser tão presente em minha vida me ensinando e guiando sempre para os melhores caminhos.

AGRADECIMENTOS

Agradeço primeiramente aos meus pais por terem me dado essa oportunidade. As minhas amigas por sempre me acompanharem. Aos meus irmãos do coração Gustavo e Walter. Aos monitores Thiago Rider, José Carlos e Wilson Gotti por terem me ajudado com seus conselhos e experiências. Ao Felipe Souto e Maria Luiza por terem acompanhado o desenvolvimento deste projeto. Ao meu professor orientador José Julimá. Ao instituto Omins pelo empréstimo de materiais. E ao meu namorado Felipe Dias.

RESUMO

Este projeto apresenta uma proposta de automação de garagens prediais usando a tecnologia RFID (identificação por rádio frequência). Para isso, um protótipo foi construído com o objetivo de simular a entrada e a saída de veículos de uma garagem. Os dispositivos RFID, servo motor, sensores de infravermelhos e chaves ópticas são os principais componentes desse protótipo. A tecnologia RFID é utilizada a fim de capturar a identificação dos veículos. O servo motor simula a cancela. Os sensores de infravermelhos informam a passagem dos carros pela cancela. As chaves ópticas verificam se a cancela está aberta ou fechada. Por fim, o computador é responsável pelo controle e pelo monitoramento do sistema.

Palavras Chave: identificação por rádio frequência, segurança, controle de acesso.

ABSTRACT

This project proposes an automation for garages in buildings by using RFID (radio frequency identification). For this, a prototype was built to simulate the entrance and the exit of vehicles in a garage. RFID devices, servo motor, infrared sensors and optical switches are key components of this prototype. As the RFID technology is utilized to capture vehicle's identification, the servo motor simulates the gate and the infrared sensors inform the passage of cars going through the gate. Thus, the optical key then verifies opening and closing of the gate. Finally, the computer is responsible for the monitoring and the control system.

Key-words: Radio frequency identification, security, access control.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

AABIC – Associação das Administradoras de Bens Imóveis e Condomínios

ANSI – American National Standards Institute

CPU – Central Processing Unit

DLL – Dynamic Link Library

DMA – Direct Memory Access

ECP – Enhanced Copabilities Port

EPC – Electronic Product Code

EPP – Enhanced Parallel Port

FIFO – First In First Out

GPS – Global Positioning System

HF – High Frequency

IBM – International Business Machines

LCD – Liquid Crystal Display

LED – Light Emitting Diode

LF – Low frequency

MIT – Massachusetts Institute of Technology

ms – Milisegundos

PWM – Pulse Width Modulation

RFID – Radio Frequency Identification

SGBD – Sistema Gerenciador de Banco de Dados

SPP – Standard Parallel Port

SQL – Structured Query Language

UHF – Ultra high frequency

USB – Universal Serial Bus

CAPÍTULO 1 - INTRODUÇÃO

1.1– Motivação e Posicionamento

A motivação para a realização deste projeto surgiu a partir da observação do crescente índice de falhas na segurança de prédios. No Distrito Federal o número de assaltos a residências cresceu em 2009. De acordo com dados da Secretaria de Segurança Pública, os roubos a residências no Distrito Federal subiram 40% de 2008 para 2009. Em uma reportagem a AABIC (Associação das Administradoras de Bens Imóveis e Condomínios) afirmou que muitos dos recentes arrastões em condomínios residenciais foram causados por falha humana. Porteiros que permitem a entrada de pessoas não autorizadas no local, seja pela garagem ou pelo portão principal, muitas vezes são os culpados por assaltos em prédios.

Com o intuito de amenizar este problema e aumentar a segurança, este projeto propõe, por intermédio de um protótipo, a construção de uma garagem automatizada. Nesta garagem, os carros são identificados por rádio frequência (RFID) e, só é permitida a entrada, se o veículo estiver devidamente cadastrado. Além disso, será capturado o dia e a hora de entrada e saída de todos os veículos.

O projeto se restringe a demonstrar essa solução em forma de protótipo, não se adequando à realidade por motivos financeiros. As questões de segurança aplicadas ao projeto também não são consideradas neste trabalho. Para este protótipo, são utilizados materiais de escala reduzida, como por exemplo, um servo motor para simular o fechamento e a abertura da cancela.

1.2 –Visão Geral do Projeto

O projeto simula um controle de acesso a prédios por intermédio de uma maquete. Na Figura 1.1 estão presentes os dispositivos eletrônicos do projeto que são controlados por um sistema de gerenciamento. Esta figura ilustra o posicionamento dos dispositivos eletrônicos utilizados na maquete.

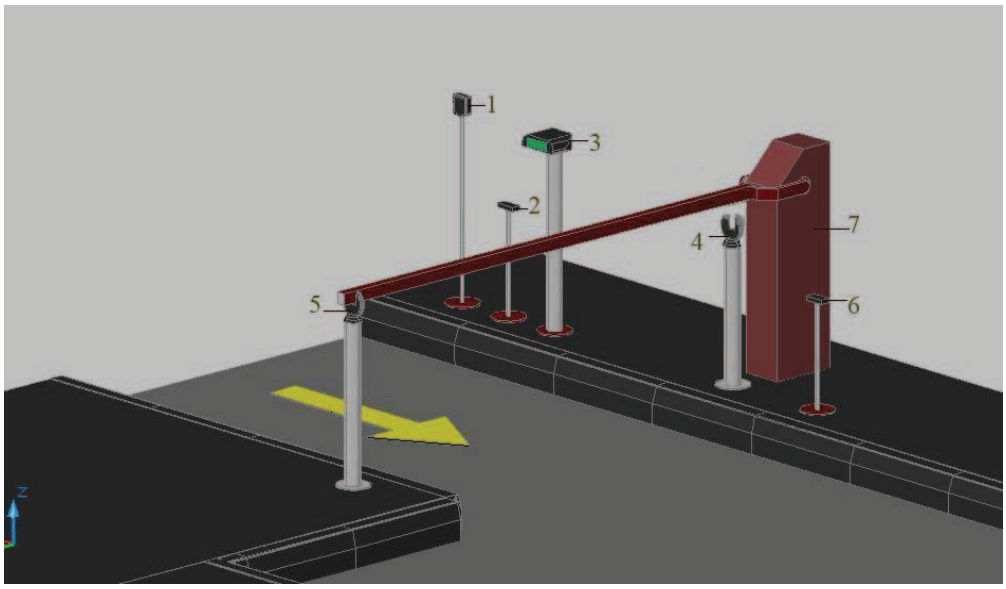

Figura 1.1 – Posicionamento dos dispositivos eletrônicos.

Na Figura 1.1 pode-se observar:

- 1 – Leitor RFID;
- 2 – Sistema infravermelho;
- 3 – Visor LCD;
- 4 – Chave óptica;
- 5 – Sistema infravermelho;
- 6 – Chave óptica;
- 7 – Servo Motor.

1.3 – Objetivos do Trabalho

O objetivo geral deste trabalho é apresentar um protótipo de uma garagem automatizada utilizando a tecnologia RFID (*Radio Frequency IDentification*). Esta garagem simula a entrada e saída de veículos em uma garagem. Para isto algumas tarefas precisam ser executadas, tais como:

- Disponibilizar o acesso ao condomínio apenas para pessoas autorizadas mediante etiqueta RFID acopladas nos carros e utilizando um leitor de RFID que emite um sinal de confirmação para o sistema de gerenciamento liberar o acesso através da cancela;
- Implementar sistemas de infravermelhos para informações sobre a passagem de carros pela cancela;
- Implementar sistemas de chaves ópticas para garantir se a cancela abriu e fechou corretamente;
- Implementar um sistema de mensagens via visor LCD (*Liquid Crystal Display*) para o usuário (motorista) ser informando se está autorizado ou não a entrar no estacionamento. Caso não esteja autorizado, informar-lhe o motivo;
- Programar um computador através do qual é implementada a lógica de acesso, liberação ou não da cancela, conforme sinal de confirmação do leitor RFID e dos infravermelhos;
- Capturar informações do veículo, que são armazenadas em um banco de dados restrito apenas ao administrador e aos vigias.

1.4 – Estrutura da Monografia

Além deste capítulo introdutório, esta monografia está estruturada em mais três capítulos e organizada da seguinte maneira:

- Capítulo 2 – Referencial Teórico e Tecnológico – Nesse capítulo é apresentado o referencial teórico e tecnológico que embasa o projeto. Primeiramente trata de detalhes da tecnologia RFID utilizada no protótipo. Em seguida apresenta uma visão geral sobre a porta paralela, os servos motores, os infravermelhos, as chaves ópticas e o visor LCD que também são utilizados na montagem da maquete.
- Capítulo 3 – Desenvolvimento do Projeto – O capítulo do desenvolvimento do projeto possui a visão e a topologia do projeto. Além de especificar os hardwares e softwares utilizados no protótipo. Esse capítulo também mostra os testes realizados e a simulação do projeto.
- Capítulo 4 – Conclusão – Esse capítulo marca o final da monografia concluindo-a e apresentando propostas para futuros trabalhos.

CAPÍTULO 2 – REFERENCIAL TEÓRICO E TECNOLÓGICO

2.1 – Identificação por Rádio Frequência (RFID)

2.1.1 – Histórico

O funcionamento e os mecanismos das ondas de rádio e a radiação eletromagnética só foram compreendidos pela humanidade nos séculos recentes, apesar de terem surgidos no momento da criação do universo. Cerca de 100 anos antes de cristo, os chineses foram os primeiros a observarem fenômenos magnéticos utilizando imãs naturais. Desde então pouco se pesquisou sobre ondas de rádio e radiação eletromagnética até meados do século XVIII desta era. Foi nesse século que ocorreu uma explosão de conhecimento empírico e de métodos para tratar matematicamente os avanços obtidos nessa área (SAKAMOTO, 2009).

Em 1864 o físico James Clerk Maxwell concluiu que a energia elétrica e magnética viajava em ondas transversas que se propagam em velocidade igual a da luz. Já em 1887 a teoria de Maxwell foi confirmada pelo alemão Heinrich R. Hertz. Credita-se a ele o feito de ser o primeiro a produzir e receber ondas de rádio. Ainda no século XIX foram feitas as primeiras transmissões telegráficas intercontinentais utilizando ondas de rádio. Porém esses dispositivos faziam apenas a transmissão de centelhas entre placas metálicas sintonizadas. Ou seja, a onda, em si, não portava informação. Sua presença ou ausência era codificada para ter algum significado (SAKAMOTO, 2009).

Foi apenas em 1906 que Ernst F. W. Alexanderson fez surgir as primeiras ondas de rádio contínuas. Todos os aspectos dessa onda são controlados e empregados para a transmissão da informação possibilitando a transmissão em maior volume e com menor ocorrência de erros (SAKAMOTO, 2009).

Por volta de 1922 surgiu o radar que permitiu avançar na criação das primeiras aplicações realmente de RFID. Ele emite ondas de rádio em todas as direções e a sua reflexão, quando captada, permite identificar a presença, a posição e velocidade de objetos. Os militares logo perceberam seu potencial, mantendo as novas descobertas em sigilo. No começo da segunda guerra mundial, os radares já eram amplamente empregados pelos dois lados oponentes, porém tinham uma grave deficiência. Ao alertar sobre a presença de um avião quando este ainda se encontrava a quilômetros de distância, os aparelhos de então não

tinham meios de identificar se o que se aproximava era amigo ou inimigo, sendo necessário contato visual ou via rádio para fazer a distinção. Ainda no período da guerra os ingleses descobriram uma maneira para suprir essa deficiência. Construíram o primeiro sistema ativo consistindo de um transmissor que, quando atingido por uma onda de radar, começava a transmitir um sinal que identificava a aeronave como amiga. Chamado de *transponder* esse aparelho foi instalado em todos os aviões da força aérea inglesa da época e é usado, em versões atuais, em todos os aviões. O *transponder* é então considerado como a primeira utilização da tecnologia RFID, pois se utilizam do mesmo método. A idéia e técnica necessárias ao surgimento do RFID já estavam lá, aguardando para serem notados (SAKAMOTO, 2009).

2.1.2 – Introdução

O RFID (*Radio Frequency Identification*) é um termo usado para as tecnologias que usam as ondas de rádio para identificar automaticamente pessoas ou objetos. Esta tecnologia permite capturar automaticamente dados, para identificação de objetos com dispositivos eletrônicos, conhecidos como etiquetas, *tags* ou *transponder*. Estes dispositivos emitem sinais de rádio frequência para leitores ou antenas, que captam as informações.

O RFID surgiu inicialmente como solução para sistemas de rastreamento e controle de acesso na década de 80. Juntamente com outros centros de pesquisa, o MIT (*Massachusetts Institute of Technology*) iniciou um estudo que utilizava tecnologias baseadas em rádio frequência para servir como modelo de referência ao desenvolvimento de novas aplicações de rastreamento e localização de produtos. Assim nasceu o Código Eletrônico de Produtos - EPC (*Electronic Product Code*). O EPC definiu uma arquitetura de identificação de produtos que utilizavam os recursos proporcionados pelos sinais de rádio frequência, chamada posteriormente de RFID (*Radio Frequency Identification*). A utilização da rádio frequência em vários processos foi incentivada pela necessidade da identificação de informações em produtos que estivessem em movimento ou em ambientes insalubres e também nas vezes que o código de barras não fosse viável (MATIAS, 2010).

Essa tecnologia provavelmente desencadeará uma revolução no futuro onde ela será a base para uma nova realidade na identificação de produtos e objetos (MATIAS, 2010).

2.1.3 – Funcionamento da Identificação por Rádio Frequência

A figura 2.1 ajuda a esclarecer o funcionamento da identificação por rádio frequência. Para o funcionamento estes sistemas são compostos por: etiqueta / *tranponder* / *tag* de identificação, dispositivos de leitura (antena e leitor) e um sistema de dados para permitir o acesso das informações. A comunicação desse tipo de sistema ocorre via ondas de rádio que levam as informações uni ou bidirecional. Este método de comunicação permite atribuir identidade única a cada etiqueta tornando-as únicas.

Existem dois tipos de sistemas: ativo e passivo. Os dois utilizam de ondas eletromagnéticas para a comunicação, porém o método de energização das etiquetas é diferente. Enquanto no sistema ativo cada etiqueta possui uma fonte interna de energia (bateria), o que permite alimentar continuamente seu circuito radiotransmissor, o sistema passivo se utiliza do sinal da onda eletromagnética incidente para sua energização. Com isso, sistemas passivos necessitam de sinais fortes e suas respostas são limitadas devido ao baixo nível de intensidade de energia. Essa diferença básica interfere diretamente no raio de comunicação do todo o sistema, na capacidade de ler várias etiquetas simultaneamente, na possibilidade de incluir sensores e registradores de dados, e em muitos outros parâmetros funcionais.

Figura 2.1 – Diagrama esquemático básico dos sistemas de RFID.

FONTE – Jornal de Representação Comercial sdr.

2.1.4 – Frequência de Transmissão e Recepção

A frequência tem um papel muito importante para a comunicação entre etiquetas e leitor. Quando vai ser implantado um sistema utilizando RFID, existem diversos fatores que devem ser observados quanto ao tipo de aplicação, normas nacionais e especificações. A frequência de um sinal é a grandeza que indica a velocidade de repetição de um fenômeno periódico. Para transmissões de rádio a frequência é uma das grandezas mais relevantes, pois é através dela que é possível estudar o comportamento dos sistemas em relação ao ambiente no qual o sistema será instalado. Cada um dos componentes de um sistema de RFID varia de acordo com a faixa de frequência definida para a solução do sistema, sendo que atualmente as aplicações para RFID operam principalmente nas seguintes faixas de frequências: LF, HF, UHF. Cada uma destas faixas de frequência possui comportamento e características diferentes, sendo que para cada aplicação de RFID deve ser avaliada qual a melhor faixa de frequência para a necessidade.

• LF (*Low Frequency*) – faixa de operação de 125 kHz até 134 kHz. São denominados sistemas baixa frequência;
• HF (*High Frequency*) – faixa de operação de 13,56 Mhz. São denominados sistemas de alta frequência;
• UHF (*Ultra High Frequency*) – faixa de operação de 860 MHz até 960 MHz. São denominados sistemas de UHF.
(*SANGHERA*, 2007*)*

2.1.5 – Componentes do RFID

Um sistema genérico RFID é composto por etiquetas inteligentes (também conhecidas como *tag* RFID ou t*ransponders*), por um ou mais leitores (também conhecidos como interrogadores ou *transceptors*), por uma antena ou bobina e por um sistema computacional que utiliza os dados captados pelos leitores RFID (THORNTON, 2006).

2.1.5.1 – Etiqueta, *Tag* ou *Transponder*

A etiqueta é também conhecida por *transponder* (*transmitter*+responder = *transponder*) devido a sua função, ou seja, ela recebe o sinal enviado pelo interrogador e responde com o seu ID e alguma outra informação, caso seja uma etiqueta com memória disponível (*SANGHERA*, 2007). O *transponder* é composto por três componentes básicos: antena, circuito integrado e encapsulamento como são ilustrados na figura 2.2. Da junção destes três componentes é formada a etiqueta que esta disponível em diversos formatos, tais como cartões, pastilhas, argolas e podem ser encapsulados com materiais como o plástico, vidro, epóxi, etc.

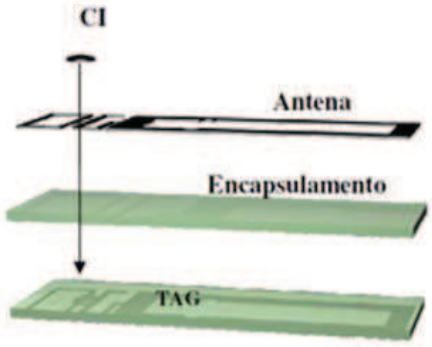

Figura 2.2 – Componentes do *transponder*.
FONTE*: (DOBKIN, 2007)*.

2.1.5.1.1 – Tipos de etiquetas

A frequência de operação e o tipo de etiquetas utilizadas fazem bastante diferença em relação ao desempenho de todo o sistema. O tipo de etiqueta é determinado por dois fatores: se a etiqueta é capaz de iniciar a comunicação e se a etiqueta tem fonte de energia própria. Baseados nesses dois fatores acima existem três tipos de etiquetas: passivas, semi-passivas e ativas (*SANGHERA*, 2007).

Passivas: as passivas são do tipo só leitura (*read only*), usados para curtas distâncias. Nestes, a capacidade de armazenamento varia entre 64 bits e 8 kilobits. É uma etiqueta que não tem fonte própria de energia, como uma bateria, e, portanto não é capaz de iniciar a

comunicação. Ela responde ao sinal enviado pelo leitor utilizando a energia contida no sinal. A Figura 2.3 mostra um exemplo de etiqueta passiva.

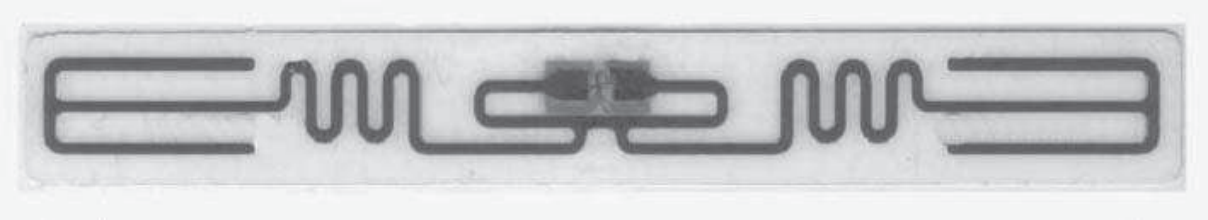

Figura 2.3 – Etiqueta passiva.
FONTE – (*DOBKIN, 2007*).

Semi-passivas: é um tipo de etiqueta que se utiliza da bateria para operação do *chip* da etiqueta e também se utiliza do sinal da leitora para reenviar o sinal. Esta etiqueta por necessitar da onda de rádio do leitor, tem seu campo também restrito à área da leitura.

Ativas: são alimentadas por uma bateria interna e permitem processos de escrita e leitura. Uma etiqueta deste tipo pode iniciar a comunicação mandando seu próprio sinal não dependendo do leitor para utilizar seu circuito interno e para criar sinais. E também não necessita de uma chamada de *wakeup* (acorda) do leitor (*SANGHERA*, 2007; *DOBKIN*, 2007). Em relação à sua operação, a etiqueta pode tanto ficar ligada o tempo todo quanto ser acordada assim que receber um sinal. Como a etiqueta ativa contém uma bateria, seu tamanho é maior que as etiquetas passivas. Já no que diz respeito à abrangência de leitura, a ativa é a de maior área disponível devido sua capacidade de iniciar a comunicação. Em consequência dessa característica, as ativas podem ser integradas a sistemas de posicionamento global (*Global Positioning System* – GPS) para determinação da posição de, por exemplo, um objeto.

Apesar dessas grandes vantagens trazidas com a utilização de etiquetas ativas, há também algumas desvantagens no seu uso. Devido sua maior complexidade, as etiquetas ativas são mais caras quando comparadas com as passivas e semi-passivas. Além disso, seu tempo de vida é limitado comparado às passivas devido a sua bateria interna. As etiquetas passivas devem estar contidas dentro do campo de atuação dos leitores para funcionarem, enquanto as ativas, devido sua capacidade de comunicação, não necessitam estar dentro do campo de leitura do interrogador (*SANGHERA*,2007; *DOBKIN*,2007).

Neste protótipo de baixa escala foram utilizadas etiquetas do tipo passiva. Primeiramente por não precisarem ser lidas a grandes distâncias e também por serem de tamanho e custo reduzidos.

2.1.5.1.2 – Técnicas de comunicação

A faixa de leitura do sistema varia com a técnica de comunicação usada, com a potência da antena e com o tipo de etiqueta utilizado nesse sistema. De acordo com *SANGHERA*, existem duas técnicas de comunicação. São elas: acoplamento indutivo e acoplamento difuso de Retorno.

Acoplamento Indutivo: Tanto o leitor quanto a etiqueta utilizam bobinas como antenas. Essas bobinas criam campos magnéticos. A variação no campo magnético é utilizada para transferir potência (e dados) entre o leitor e as etiquetas. Essa técnica limita a área de leitura porque ela só funciona no campo de atuação das bobinas. Dessa forma, o acoplamento indutivo requer que o leitor esteja próximo da etiqueta. Com isso, a distância de leitura é, por volta, de 30 cm para LF (Frequências Baixa) e de 1m para HF (Frequências Altas) (*SANGHERA*, 2007).

Acoplamento Difuso de Retorno**:** O acoplamento difuso de retorno geralmente é utilizado por etiquetas passivas operando em UHF (Frequências Ultra Altas) ou microondas. Como o acoplamento difuso de retorno trabalha além do campo de atuação das bobinas, ele permite maiores áreas de leitura. A maior área de atuação conseguida com esse tipo de acoplamento é devida ao uso de ondas eletromagnéticas para estabelecer a comunicação ao invés do campo magnético utilizado pelo acoplamento indutivo (*SANGHERA*, 2007).

2.1.5.1.3 – O comportamento das etiquetas na identificação de objetos ou produtos metálicos

Existe uma limitação para o uso da tecnologia RFID. E esta limitação esta na identificação de metais. Já que são utilizados campos magnéticos, é natural que o metal interfira negativamente no seu desempenho. Sobretudo há encapsulamentos especiais que contornam esta limitação, fazendo com que hoje se possa identificar automóveis, vagões de trens e contêineres, observando as limitações com relação às distâncias de leitura.

2.1.5.2 – Leitor de Etiquetas (*Transponders)*

Responsáveis por ler as informações das etiquetas, os leitores decodificam o sinal recebido através de *transceivers*. Para receber este sinal eles emitem frequências de rádio que são dispersas em vários sentidos. A etiqueta responde ao leitor com o conteúdo de sua memória. Este equipamento pode ler através de diversos materiais como cimento, vidro, plástico e madeira. Depois de decodificado o sinal, o leitor passa as informações para um computador poder realizar o processamento. A Figura 2.4 ilustra o funcionamento do leitor (interrogador).

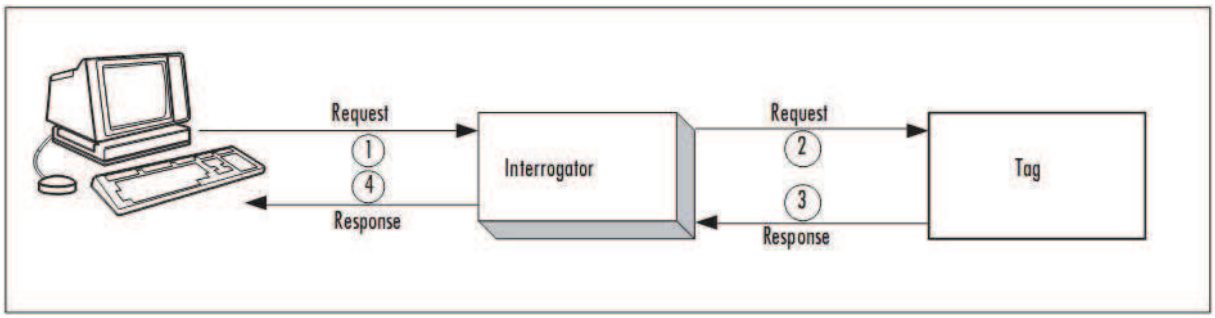

Figura 2.4 – Funcionamento do leitor (interrogador).
FONTE – (*SANGHERA*, 2007*)*.

2.1.5.2.1 – Tipos de leitores

Segundo *FINKENZELLER* (2003) existem vários tipos de interrogadores para atender aos requerimentos de várias aplicações. Todos esses tipos podem ser categorizados nessas duas classes abaixo:

·*Read-only* (somente leitura) – ler informações armazenadas (programadas) nas etiquetas é a principal função dos leitores. Todos os leitores que só têm capacidade de leitura são chamados de *read-only* (somente leitura).

·*Read and write* – leitores que podem escrever alguma informação nas etiquetas além do processo de leitura padrão, são denominados leitores *read and write* (leitura e escrita).

A etiqueta deve ser passível de escrita para que leitores *read and write* possam escrever informações nela.

2.1.5.3 – Antena de RFID

É responsável pela propagação do sinal gerado pelo leitor. É o meio que torna possível a comunicação da etiqueta e do leitor. Elas são fabricadas em vários tamanhos e diversos formatos. As antenas são encontradas ou acopladas nas etiquetas ou nas leitoras. Se acopladas nas etiquetas elas são conectadas ao seu microprocessador. Se acopladas nas leitoras podem estar conectadas ao próprio dispositivo ou distante das leitoras (*SANGHERA*, 2007*)*.

2.1.6 – Distância de Leitura

A distância da leitura é fundamental para o bom funcionamento do sistema. Ela depende de alguns fatores como tamanho e direção da antena, frequência utilizada, tipo de etiqueta, potência do leitor, entre outros. Essa distância deve ser determinada de acordo com cada tipo de aplicação não tendo assim uma única distância ideal. O Quadro 2.1 relaciona as frequências das ondas e suas aplicações em RFID e sua relação com a distância dos elementos do sistema.

Quadro 2.1 – Relação da distância de leitura com frequência.

Freqüência	Faixa máxima típica para identificadores passivos	Algumas aplicações típicas
LF	50 centímetros	Identificação de animais de estimação e leituras próximas de itens com alto conteúdo de água
HF	3 metros	Controle de acesso a prédios
UHF	9 metros	Caixas e caixotes
Microondas	> 10 metros	Identificação de veículos de todos os tipos

FONTE: (GLOVER; BHATT, 2007).

2.1.7 – Um Sistema RFID e uma Rede Sem Fio

Uma questão que gera duvida é se um sistema RFID interfere em uma rede sem fio. Na maior parte dos relatos, não. Pois as redes sem fios normalmente não usam a mesma faixa

de frequência que os sistemas de RFID. Com exceção de sistemas UHF, mas essa interferência, caso aconteça, não é maior do que a interferência de um telefone sem fio em uma televisão. Se for utilizada a mesma faixa de frequência para os dois sistemas, o que se deve fazer é evitar colocar o leitor muito próximo do aparelho que pode receber ou causar a interferência.

2.1.8 – Aplicações do Sistema RFID

Além do projeto proposto nesta monografia podem ser citadas inúmeras aplicações para o sistema RFID tais como:

- Catraca de transporte coletivo;
- Controle de produção, contagem de produtos e armazenamento;
- Caixas eletrônicos;
- Bilheterias de cinemas, boates e clubes;
- Controle de estoques;
- Apontamento automático em linhas de produção;
- Aplicações em identificação animal;
- Aplicação na saúde;
- Certificação de encomendas e documentos;
- Controle de acesso de veículos;
- Controle de acesso de pessoas em bares, boates, condomínios, clubes e estacionamentos;
- Controle de acesso a empresas;
- Controle de frotas e pedágios;
- Segurança em aeroportos.

2.1.9 – Porque a Solução RFID

A tecnologia RFID foi escolhida para este projeto porque atualmente esta sendo bastante estudada e implementada em várias áreas. Para este trabalho outros tipos de

tecnologias, como código de barras, poderiam ter sido escolhidas. Mas o RFID foi escolhido por apresentar várias vantagens como resistência mecânica alta, formatos variados, vida útil alta, custo de manutenção baixa, segurança, eliminação de erros humanos, redução de desperdício, entre outros. Foi escolhido também por ser a melhor solução para propostas de projeto futuro que é dada no final desta monografia.

Algumas vantagens do RFID sobre o código de barras são citadas no Quadro 2.2.

Quadro 2.2 – Vantagens de RFID sobre código de barras

Características	**RFID**	**Código de Barras**
Resistência mecânica	Alta	Baixa
Formatos	Variados	Etiquetas
Exige contato visual	Não	Sim
Vida útil	Alta	Baixa
Possibilidade de escrita e rescrita	Sim	Não
Leitura simultânea	Sim	Não
Dados armazenados	Alta	Baixa
Funções adicionais	Sim	Não
Segurança	Alta	Baixa
Custo inicial	Alto	Baixo
Custo de manutenção	Baixo	Alto
Reutilização	Sim	Não

2.2 – A Porta Paralela

A porta paralela é uma interface de comunicação entre um computador e um periférico. A IBM (*International Business Machines*) foi uma das primeiras empresas a desenvolver a porta paralela como uma maneira de ligar a impressora ao computador. Atualmente a porta paralela não é usada somente para impressora. Pode-se desenvolver um circuito eletrônico e acoplá-lo a essa porta e, através de um programa específico, enviar-lhe

sinais digitais para controlá-lo. Na comunicação em paralelo, grupos de bits são transferidos simultaneamente, em geral, byte a byte, através de diversas linhas condutoras dos sinais. Desta forma, como vários bits são transmitidos simultaneamente a cada ciclo, a taxa de transferência de dados (*throughput*) é alta. Há dois tipos de modelos de porta paralela a unidirecional e as bidirecionais (ROGERCOM, 2010).

A unidirecional SPP (*Standard Parallel Port*) pode chegar a uma taxa de transmissão de dados a 150KB/s. Ela comunica-se com a CPU (*Central Processing Unit*) utilizando um barramento de dados de 8 bits. Para a transmissão de dados entre periféricos são usado 4 bits por vez. Ela apresenta apenas 3 grupos de registros. O registro de dados, o registro de estado e o registro de controle (ROGERCOM, 2010).

Existem dois tipos de bidirecionais: EPP (*Enhanced Parallel Port*) e a ECP (*Enhanced Capabilities Port*).

A porta avançada EPP chega a atingir uma taxa de transferência de 2 MB/s. Para atingir essa velocidade, é necessário um cabo especial. Ela se comunica com a CPU utilizando um barramento de dados de 32 bits. Para a transmissão de dados entre periféricos são usado 8 bits por vez (ROGERCOM, 2010).

A porta avançada ECP tem as mesmas características que a EPP, porém, utiliza DMA (acesso direto à memória), sem a necessidade do uso do processador, para a transferência de dados. Utiliza também um buffer FIFO (*First In First Out*) de 16 bytes (ROGERCOM, 2010).

2.2.1 – Endereços da Porta Paralela

O computador nomeia as portas paralelas automaticamente, chamando-as de LPT1, LPT2, LPT3 e assim sucessivamente. A porta física padrão do computador é a LPT1 e, seus endereços de acesso são:

- 378h (*Data register*),
- 378+1h (*Status register*) e,
- 378+2h (*Control register*).

Às vezes pode estar disponível uma segunda porta paralela como LPT2, seus endereços são: 278h, 278+1h e 278+2h, com as mesmas funções dos endereços da porta LPT1 respectivamente. O Quadro 2.3 mostra o endereçamento dessas portas (ROGERCOM, 2010).

Quadro 2.3 – Endereçamento das portas paralelas.

Nome da Porta	*Endereço de memória*	*Endereço da Porta*		*Descrição*
LPT1	***0000:0408***	***378 hexadecimal***	***888 decimal***	***Endereço base***
LPT2	***0000:040A***	***278 hexadecimal***	***632 decimal***	***Endereço base***

2.2.2 – O Conector DB25

O DB25 é um tipo comum de conector, utilizado principalmente em computadores, e é por intermédio deste, que o cabo paralelo se conecta ao computador para poder enviar e receber dados.

No DB25, um pino está em nível lógico 0 quando a tensão elétrica no mesmo está entre 0 e 0,4v. Um pino se encontra em nível lógico 1 quando a tensão elétrica no mesmo está acima de 3.1 e até 5v. As Figuras 2.5 e 2.6 mostram os conectores fêmea e macho do padrão DB25, com 25 pinos, onde cada pino tem um nome que o identifica.

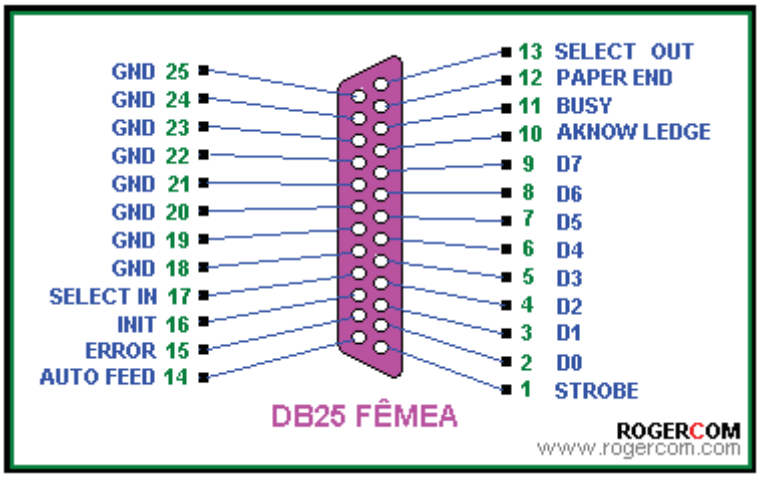

Figura 2.5 – Conector fêmea do DB25.

FONTE: (ROGERCOM).

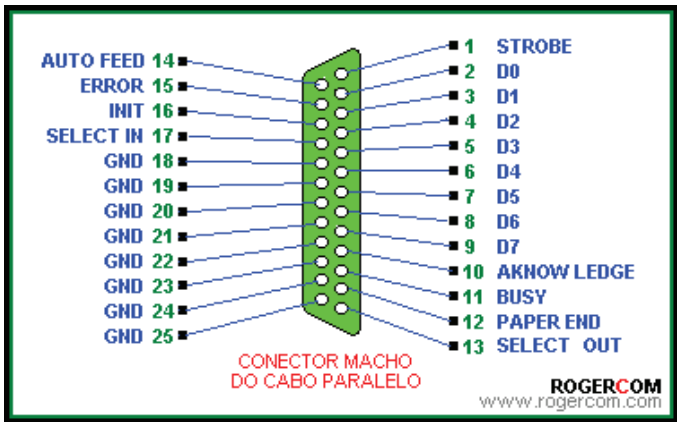

Figura 2.6 – Conector macho do DB25.

FONTE: (ROGERCOM).

2.2.3 – Pinagem

A porta paralela é formada por 17 linhas de comunicação e 8 linhas que se ligam ao terra. A pinagem desta porta é feita da seguinte forma:

- Os pinos de 2 a 9, são controlados pelo registro de DADOS, tendo como função enviar dados através da porta paralela;
- Os pinos 10, 11, 12, 13 e 15, são usados para troca de mensagens de estado da solução construída com o computador. O registro que os controla é o registro ESTADO;
- Os pinos 1, 14, 16 e 17, são usados para troca de mensagens de controle da solução construída com o computador. Estes são controlados pelo registro CONTROLE;
- Os pinos 18 a 25 são aterrados.

A Figura 2.7 ilustra a distribuição dos pinos de uma porta paralela.

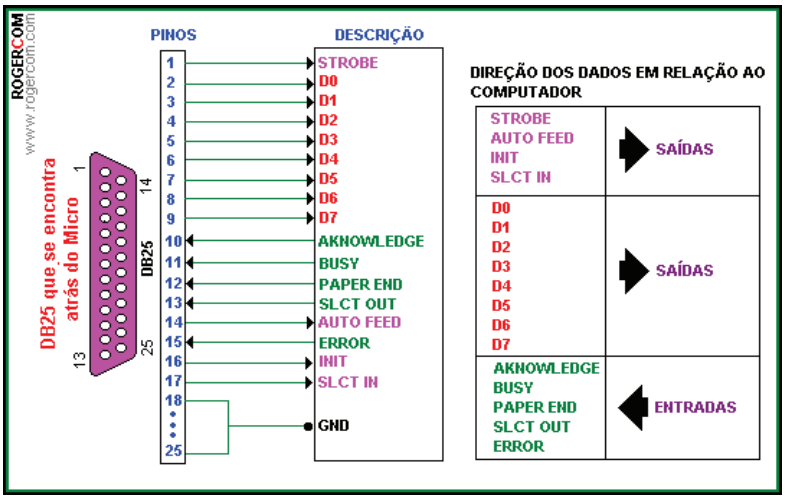

Figura 2.7 – Distribuição de pinagem da porta paralela.

FONTE: (ROGERCOM).

Existem pinos na porta paralela que trabalham com lógica invertida. Ou seja, para ativar estes pinos é preciso enviar um sinal lógico “0” e para desativar, mandar um sinal lógico “1”.

O Quadro 2.4 apresenta de forma resumida o que foi explicado acima mais as funções de cada pino.

Quadro 2.4 – Funções de cada pino.

Pino DB25	Nome	In/Out	Registro-bit	Invertido
1	nStrobe	Out	Controle-0	Sim
2	Data 0	In/Out	Data-0	Não
3	Data 1	In/Out	Data-1	Não
4	Data 2	In/Out	Data-2	Não
5	Data 3	In/Out	Data-3	Não
6	Data 4	In/Out	Data-4	Não
7	Data 5	In/Out	Data-5	Não
8	Data 6	In/Out	Data-6	Não
9	Data 7	In/Out	Data-7	Não
10	nAck	In	Status-6	Não
11	Busy	In	Status-7	Sim
12	Paper-Out	In	Status-5	Não
13	Select	In	Status-4	Não
14	Linefeed	Out	Controle-1	Sim
15	nError	In	Status-3	Não
16	nInitialize	Out	Controle-2	Não
17	nSelect-Printer	Out	Controle-3	Sim
18 - 25	Ground	-	-	-

2.3 – Servo Motor

Os servos motores possuem uma grande aplicabilidade e funcionalidade, que se estendem desde o setor da robótica de pequeno porte até as indústrias e seus dispositivos automáticos. Para quem precisa de movimentos de rotação e linear os servos motores são os motores elétricos mais indicados. São utilizados principalmente em aplicações de posicionamento. Recebem sinais elétricos que transformam em movimentos de rotação ou em deslocamentos lineares precisos. Existem vários modelos de servos motores, com diferentes tipos de motores e com diferentes processos de realimentação. A figura 2.8 mostra um exemplo de um servo motor.

Figura 2.8 – Servo motor.
FONTE – (*PyroElectro)*.

2.3.1 – Constituição

Estes servos motores são pequenos dispositivos constituídos basicamente por um pequeno motor, um circuito eletrônico de controle, um pequeno potenciômetro que roda com o eixo do servo, um conjunto de engrenagens e três condutores para ligação.
Segundo o autor FRANCISCO, o motor do servo, ao ser alimentado, faz rodar uma série de engrenagens que amplificam e transferem o binário do motor para o eixo. Neste estão ligados os dispositivos a movimentar. Com este processo, consegue-se uma força considerável à custa da redução da velocidade. Sendo os servos utilizados em operações de posicionamento, o

motor deve responder depressa às ordens recebidas, variando o seu binário e velocidade rapidamente. Estes pequenos motores são capazes de deslocar massas consideráveis, tornando-os poderosos para o seu tamanho. A figura 2.9 mostra os elementos que constituem o servo motor.

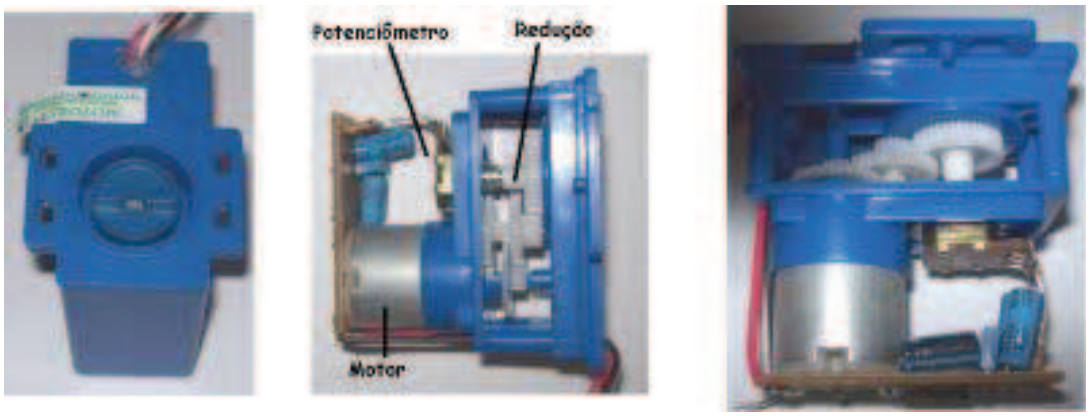

Figura 2.9 – Componentes de um servo motor.
FONTE – (FRANCISCO, 2004*)*.

2.3.2 – Principio de Funcionamento

O circuito eletrônico de controle e o potenciômetro formam um sistema interno de realimentação (*feedback*) para controle da posição do eixo do servo. O objetivo deste servo motor é permitir controlar a posição (ângulo) do disco. Para isto ele tem um controlador que aciona o motor. O motor é conectado ao disco através de um conjunto de engrenagens, para que várias voltas do motor correspondam a um pequeno ângulo do disco permitindo um ajuste fino da posição e aumentar o torque no disco. Para determinar a posição do disco, o potenciômetro está preso a ele. Cada ângulo vai corresponder a certa resistência, que é monitorada pelo controle. O circuito eletrônico compara o valor da resistência do potenciômetro com os impulsos que recebe pela linha de controle, ativando o motor para corrigir qualquer diferença que exista entre ambos. Isto é, o potenciômetro permite ao circuito de controle verificar a todo o momento a posição angular do eixo do servo. Se o eixo esta no ângulo correto, o motor não roda. Se o circuito verifica que o angulo não é o correto, o motor roda, no sentido adequado, até alcançar o ângulo correto. Aplicado e mantido o sinal, o servo matém a posição angular do eixo. Se o sinal mudar, o eixo do servo roda para nova posição angular. Caso não seja aplicado sinal, só a força de atrito mantém o servo na sua posição

angular. A tensão aplicada ao motor do servo é proporcional a distancia que o eixo necessita rodar. Logo, se o eixo precisa rodar muito, o motor roda à velocidade máxima. Se precisa rodar pouco, o motor roda a uma velocidade mais baixa. A esta propriedade chama-se controle proporcional. Em princípio o servo motor permite uma movimentação de 0 a 180 graus (FRANCISCO, 2004).

2.3.3 – Controle do Ângulo de Rotação

O ângulo de rotação do motor dos servos é determinado pela duração do impulso (tempo *ON*) que se aplica na entrada de comando, nesta aplica-se um sinal PWM (*Pulse Width Modulation* – Modulação por largura de impulso). Trata-se de uma onda em que se varia a duração do tempo T_{on}, mantendo o período da mesma fixo.
A largura mínima e máxima do impulso depende do tipo de servo. No entanto, e no caso geral, se o servo receber na sua entrada impulsos com a duração de:

- 1 ms, o seu eixo roda, no sentido anti-horário, até atingir o limite do intervalo de rotação, o que corresponde a 0 graus;
- 1,5 ms, o eixo roda até ficar estável no centro do intervalo de rotação, a que corresponde o ângulo de 90 graus;
- 2 ms, o servo roda no sentido horário, até atingir o outro limite do intervalo de rotação (180 graus ou um pouco mais).

Ou seja, impulsos entre 1 ms e 1,5 ms farão com que o servo rode para posições intermediarias entre 0 e 90 graus, enquanto impulsos entre 1,5 ms e 2 ms farão com que o servo rode para posições intermediarias entre 90 e 180 graus como ilustra a Figura 2.10.

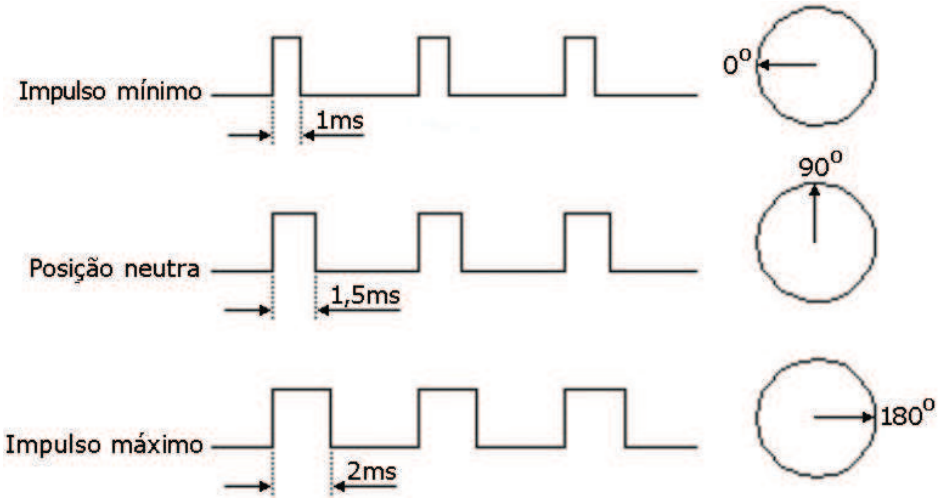

Figura 2.10 – Diagrama de controle do ângulo de rotação dos servos.
FONTE – (FRANCISCO, 2004).

Os impulsos, para que o servo funcione corretamente, devem ser aplicados a cada 20 ms (f=50 Hz), como mostra a Figura 2.11, mas valores entre 10 ms e 30 ms também são normalmente aceitáveis.

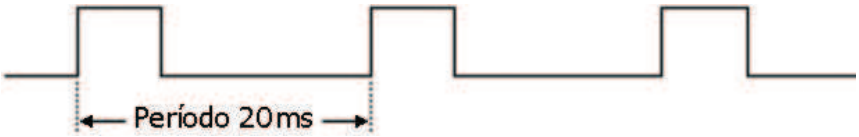

Figura 2.11 – Período dos impulsos.
FONTE – (FRANCISCO, 2004).

Se um mesmo impulso for aplicado ciclicamente na entrada do servo, o seu eixo se mantém na mesmo posição angular. Ao tentar-se rodar manualmente o eixo do servo, o circuito de controle detecta uma diferença entre o valor do potenciômetro e a amplitude dos impulsos e ativa o motor para tentar corrigir. Este processo é tão rápido que apenas se sente o servo a resistir à tentativa de se alterar a sua posição. Note-se que a duração dos impulsos e os valores angulares de rotação do eixo dependem do tipo de servo (FRANCISCO, 2004).

2.3.4 – Porque o Servo Motor

O servo motor foi escolhido para este protótipo por ser um motor de movimentos precisos e por ser utilizado principalmente em aplicações de posicionamento. Por possuir um potenciômetro mecanicamente ligado ao eixo de saída e um batente mecânico na engrenagem, esse tipo de motor é utilizado para controlar movimentos de 0° a 180°. Tendo em vista que o protótipo da cancela foi feita de alumínio, não precisando de um motor com torque de muitos quilos, o servo utilizado neste projeto foi um Hextronik HXT900 com torque de 1.60 kg/cm por ser de menor custo.

2.4 – Infravermelhos

Devido a sua simplicidade a utilização dos sinais de infravermelho é uma das soluções mais adequadas para problemas em robótica, em sistemas de segurança e aplicações industriais, na detecção de obstáculos e "presença". Suas vantagens são o baixo custo e a não necessidade de contato físico.

2.4.1– Tipos de Detecção e Aplicações

Os tipos de detecção podem ser divididos em dois grupos: detecção por reflexão e detecção por interrupção de feixe.

Detecção por reflexão: Nesse tipo de detecção, a luz emitida pelo emissor cria uma região ativa cuja presença de um objeto faz com que a luz seja refletida de volta para o receptor. Se o sinal alcançar um limite pré-definido, ativa o sensor indicando a presença do objeto. A Figura 2.12 mostra um exemplo de detecção por reflexão.

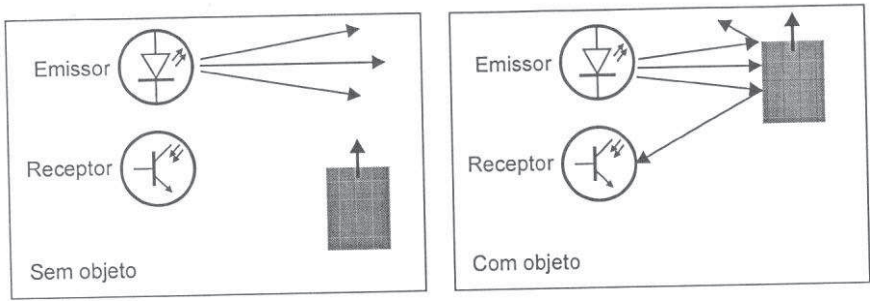

Figura 2.12 – Detecção por reflexão.
FONTE – (THOMAZINI, 2005; URBANO BRAGA, 2005).

A Figura 2.13 mostra uma aplicação onde o conjunto emissor-receptor auxilia no estacionamento de um veículo. Ao se aproximar a uma distância de poucos centímetros da parede o sistema avisa o motorista o ponto exato de parar.

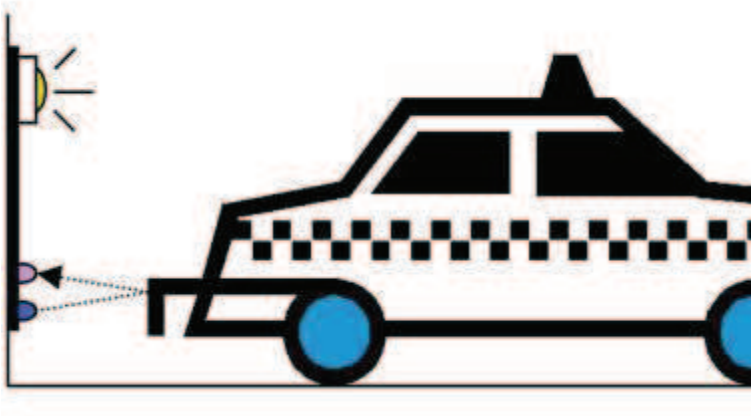

Figura 2.13 – Aplicação de uma detecção por reflexão.
FONTE – (MIGUEL, 2009).

Outra aplicação pode ser encontrada na detecção de presença. A Figura 2.14 mostra uma porta com emissor e receptor instalados nela. Quando uma pessoa passa pela porta, o feixe é refletido e um sistema de alarme pode reconhecer a presença e tomar as ações necessárias (MIGUEL, 2009).

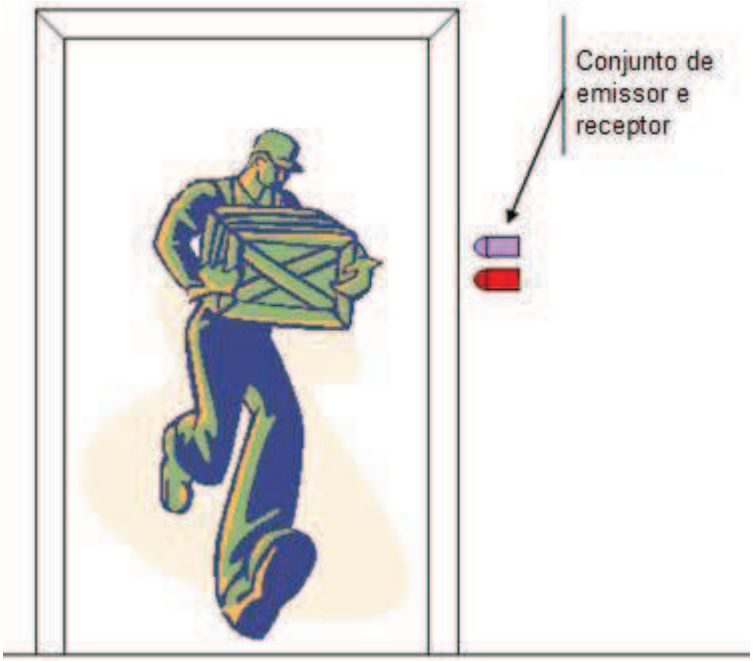

Figura 2.14 – Outra aplicação de uma detecção por reflexão.
FONTE – (MIGUEL, 2009).

Detecção por interrupção de feixe: Neste tipo de aplicação, ao serem alinhados, o emissor e o receptor criam entre si uma barreira de luz. O receptor, então, fica constantemente recebendo o feixe de infravermelho. A presença de um objeto interrompendo essa barreira faz com que o sensor seja ativado. A Figura 2.15 mostra este tipo de aplicação (THOMAZINI, 2005).

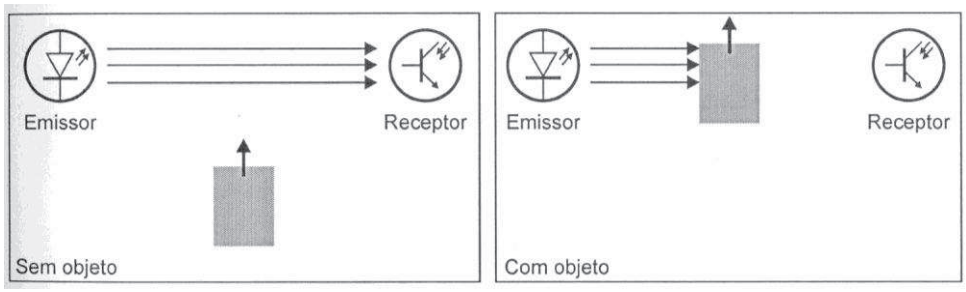

Figura 2.15 – Detecção por interrupção de feixe.
FONTE – (THOMAZINI, 2005; URBANO BRAGA, 2005).

Uma aplicação para esse tipo de técnica é a implementação de sistemas de segurança ou contadores de peças em uma esteira na linha de produção industrial. Além de detectar entrada e saída de pessoas e automóveis (MIGUEL, 2009).

2.5 – Visor LCD

O Visor LCD é dispositivo eletrônico-óptico modulado utilizado em vários aparelhos eletro-eletrônico com a finalidade de mostrar resultados ou informações. Alguns destes aparelhos são:

- Dispositivos de jogos;
- Relógios;
- Televisores;
- *Displays* em computadores de bordo de automóveis;
- Leitores de vídeo;
- Monitores para computadores;
- Calculadoras;
- Telefones;
- Painéis de instrumentos.

Tem baixo consumo de energia elétrica permitindo ser utilizado em equipamentos portateis, alimentados por bateria eletrônica. A Figura 2.16 mostra um exemplo de um *display* LCD.

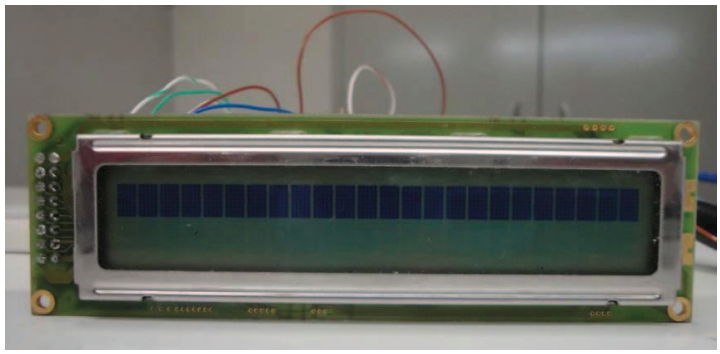

Figura 2.16 – Imagem de um *display* LCD.

2.5.1 – Funcionamento Básico do Visor LCD

O visor LCD é usado para exibir informações como texto, imagens e vídeos. É um *display* de cristal líquido (LCD - *liquid crystal display*). Consiste de um líquido polarizador da luz, eletricamente controlado, que se encontra comprimido dentro de camadas entre duas lâminas transparentes polarizadoras. Os eixos polarizadores das duas lâminas estão alinhados perpendicularmente entre si. Cada camada é provida de contatos elétricos que permitem que um campo elétrico possa ser aplicado ao líquido no interior (*WIKIPÉDIA*, 2010). A Figura 2.17 ilustra as diversas camadas de um LCD.

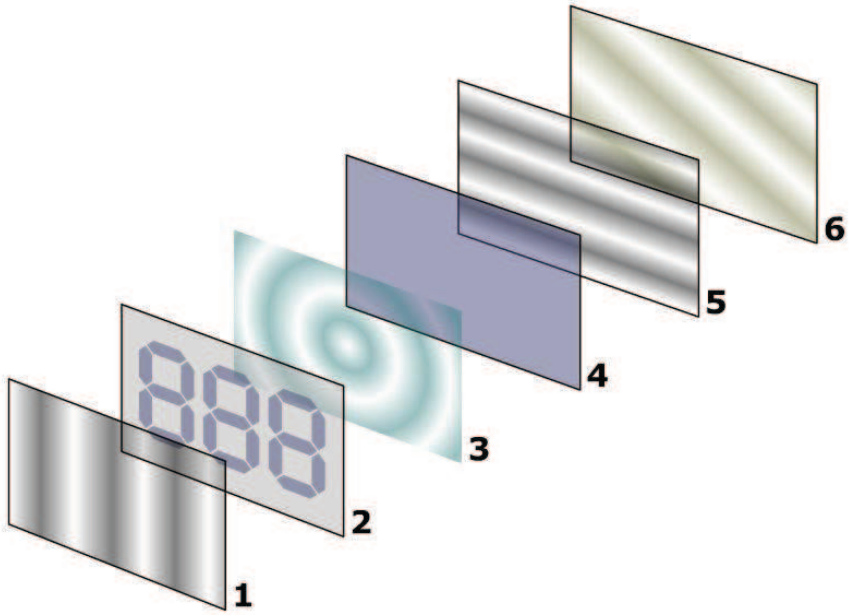

Figura 2.17 – Camadas de um LCD.
FONTE – (*WIKIPÉDIA*, 2010).

2.5.2 – Pinagem do LCD

O visor LCD Samsung da serie KS0063 utilizado neste projeto apresenta 14 pinos não possuindo os pinos 15 e 16 que são responsáveis pelo *LED backlight* (iluminação de fundo). Este *display* pode ser alimentado com 5 volts. O Quadro 2.5 apresenta sua estrutura de pinagem.

Quadro 2.5 – Pinagem de um LCD.

Pino	Símbolo	Detalhes
1	GND	Ground- Terra
2	Vcc	Supply Voltage +5V
3	Vo	Contrast adjustment
4	RS	0->Control input, 1-> Data input
5	R/W	Read/ Write
6	E	Enable
7 to 14	D0 to D7	Data
15	VB1	Backlight +5V
16	VB0	Backlight ground

FONTE: (ELECTROSOFTS).

2.6 – Chave Óptica

As chaves ópticas, atualmente, são importantes dispositivos usados nas aplicações de controle e transmissão de informações. Esses dispositivos envolvem o uso da luz como meio de transmissão de sinais de controle de informações. Seu funcionamento consiste basicamente em um emissor de luz (um *LED* infravermelho, por exemplo) e um receptor. A Figura 2.18 ilustra uma chave óptica.

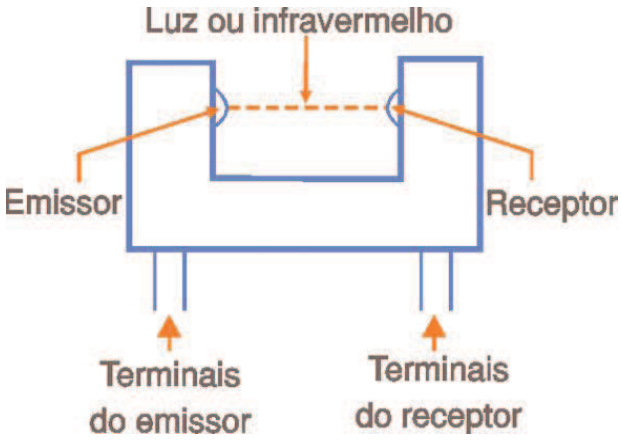

Figura 2.18 – Chave óptica.
FONTE: (SABERELETRONICA).

Seu acionamento se concretiza quando um objeto se interpõe ao feixe de luz que vai

do elemento emissor (*LED*) ao elemento receptor. A luz do elemento emissor (*LED*) incide no elemento sensor através de uma abertura. Quando um objeto interrompe o feixe de luz, um sinal de comando é produzido pelo receptor. As chaves ópticas podem ser usadas em aplicações em que se exija transferência de sinais entre circuitos de forma isolada. A segurança da transferência óptica garante a integridade dos equipamentos e de quem o manuseia.

2.7 – Linguagem de Programação

Basicamente uma linguagem de programação age como um tradutor entre o programador e o computador. A linguagem de programação serve para instruir o computador sem que o programador precise aprender a linguagem nativa da máquina.

Diferentes partes de um programa podem ser escritas em diferentes linguagens. Uma linguagem de programação pode ser convertida, ou traduzida, em código de máquina por compilação ou interpretação, que juntas podem ser chamadas de tradução.

Caso o método utilizado seja o que traduz todo o texto do programa para só depois executar o programa, então diz-se que o programa foi compilado e que o mecanismo utilizado para a tradução é um compilador. A versão compilada do programa tipicamente é armazenada, de forma que o programa pode ser executado um número indefinido de vezes sem que seja necessária nova compilação, o que compensa o tempo gasto na compilação. Isso acontece com linguagens como Pascal e C.

Há programas em que o texto só é traduzido à medida que vai sendo executado, como é o caso do Javascript, do Python e do Perl. Diz-se que o programa foi interpretado e que o mecanismo utilizado para a tradução é um interpretador neste processo de tradução de trechos seguidos de sua execução imediata. Programas interpretados são geralmente mais lentos do que os compilados, mas são também geralmente mais flexíveis, já que podem interagir com o ambiente mais facilmente (JAMSA, 1999).

2.7.1 – Linguagem C

Dennis Ritchie foi o responsável pelo nascimento da linguagem C. O C foi implementado pela primeira vez na década de 70. Essa linguagem nada mais é que a derivação da linguagem B criada por Ken Thompson. O B, por sua vez, veio da linguagem BCPL, inventada por Martin Richards.

O C é uma linguagem de programação genérica que é utilizada para a criação de programas diversos como processadores de texto, planilhas eletrônicas, sistemas operacionais, programas de comunicação, programas para a automação industrial, gerenciadores de bancos de dados, programas de projeto assistido por computador, programas para a solução de problemas da engenharia, física, química e outras ciências.

É uma linguagem padronizada pela ANSI. A ANSI é a sigla para *American National Standards Institute* e designa uma organização americana que tem a função de estabelecer quais normas desenvolvidas devem virar padrão. As empresas, grupos independentes, universidades desenvolvem novas tecnologias que são submetidas à análise do ANSI (APOSTILANDO).

2.7.2 – Vantagens da Linguagem C

A linguagem C foi selecionada tendo em vista as grandes vantagens trazidas por ela. Além de ser uma linguagem popular e de fácil aprendizado, a principal vantagem desta linguagem é o gerenciamento da sua memória. Ao contrário de muitas linguagens de programação, o C permite ao programador endereçar a memória de maneira muito parecida como seria feito em Assembly. Linguagens como o Java ou o Perl fornecem mecanismos que permitem que o programador faça o seu trabalho sem se ter de preocupar com a atribuição de memória ou com apontadores. Geralmente, isso é bom uma vez que é bastante tedioso lidar com a alocação de memória quando se escreve aplicações com algoritmos de alto nível. No entanto, quando for para lidar com tarefas de baixo-nível, como a de copiar um conjunto de bytes para uma placa de rede, torna-se altamente necessário um acesso direto à memória — algo que não é possível fazer apenas com Java. O C pode ser diretamente compilado em código de máquina, que é rapido e eficiente.

Concluindo, além dos fatores expostos acima, C é uma linguagem que qualquer Engenheiro de Computação deve saber e ter noção de seus conceitos, pois esta linguagem serviu como base para a criação de inúmeras outras. Seu conhecimento facilita bastante o entendimento de novas linguagens mais avançadas (JAMSA, 1999).

2.8 – Banco de Dados MySQL

O MySQL é um SGBD (sistema de gerenciamento de banco de dados), que utiliza a linguagem SQL (Linguagem de Consulta Estruturada, do inglês *Structured Query Language*) como interface. Esse SGBD possui licença dupla sendo uma delas de software livre. É atualmente um dos bancos de dados mais populares. Suas principais metas são velocidade, robustez e facilidade de uso. A base sob a qual o MySQL foi construído é formada por um conjunto de rotinas que foram utilizadas em ambiente de produção com alta demanda por muitos anos. Apesar de o MySQL estar sempre em desenvolvimento, este sistema já oferece um conjunto de funções altamente útil. Roda na maioria dos sistemas operacionais, incluindo Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64), e Windows (MILANI, 2008).

2.8.1 – Características do MySQL

Este popular sistema de gerenciamento de banco de dados possui várias caracteristicas. Dentre elas (WIKIPEDIA- 2009):

- Portabilidade (suporta praticamente qualquer plataforma atual);
- Compatibilidade (existem drivers ODBC, JDBC e .NET e módulos de interface para diversas linguagens de programação, como Delphi, Java, C/C++, Visual basicPython, Perl, PHP, ASP e Ruby)
- Excelente desempenho e estabilidade;
- Pouco exigente quanto a recursos de hardware;
- Facilidade de uso;
- É um Software Livre;
- Suporta Triggers;
- Replicação facilmente configurável;

- Interfaces gráficas (MySQL Toolkit) de fácil utilização cedidos pela MySQL Inc.

Por fim, o MySQL foi escolhido para este projeto por ser um software livre, de fácil utilização e de tempo de resposta considerado adequado para a solução criada.

CAPÍTULO 3 – DESENVOLVIMENTO DO PROJETO

Este capítulo apresenta a implementação, os testes e os resultados obtidos durante o desenvolvimento do projeto além da simulação do funcionamento. O item 3.1 e seus subtópicos referem-se ao entendimento geral do projeto. O item 3.2 trata da descrição dos dispositivos eletrônicos implementados. Já o item 3.3 do software responsável pelo funcionamento do protótipo. O item 3.4 apresenta os testes e resultados e o 3.5 a simulação.

3.1 – Desenvolvimento do Projeto

O projeto foi desenvolvido em quatro etapas. A primeira etapa foi o planejamento geral do projeto e seu estudo bibliográfico. A segunda etapa foi marcada pela redação da monografia e pela compras de materiais necessários para a montagem do protótipo. A terceira etapa se iniciou com os primeiros testes que serão mostrados ainda neste capítulo e finalizou com a montagem da maquete. Por fim, a quarta etapa foi a redação final desta monografia. A Figura 3.1 mostra essas quatro etapas.

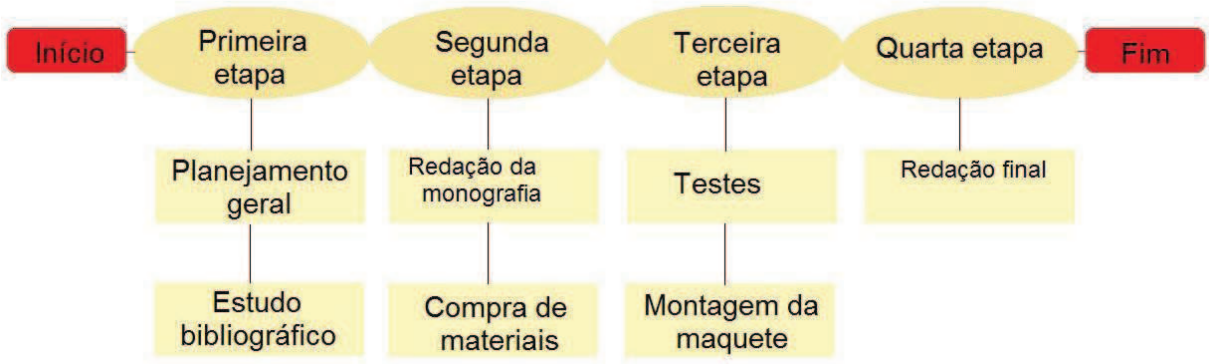

Figura 3.1 – Etapas do projeto.

3.1.1 – Estrutura Geral do Projeto

Este projeto apresenta um protótipo de uma garagem automatizada que simula a entrada e saída de veículos. A idéia desta garagem automatizada é garantir o acesso apenas de veículos autorizados e, além disso, deixar registrado esses acessos em um banco de dados. Neste protótipo encontram-se os seguintes dispositivos eletrônicos:

- Kit RFID (etiquetas e leitor *Thing Magic* M5e);
- Servo motor;
- Chaves ópticas;
- Sensores infravermelhos.

Estes dispositivos são controlados e monitorados por um sistema de gerenciamento que foi programado em linguagem C utilizando o software Dev C++. Para consulta de veículos autorizados e para o registro de acessos da garagem foi utilizado o banco de dados MySQL. A comunicação entre software e hardware é feita da seguinte forma:

- Cabo de rede – este cabo foi utilizado para a comunicação do banco de dados com o sistema de gerenciamento.
- Cabo USB – este cabo foi utilizado para a comunicação do sistema de gerenciamento com o leitor RFID.
- Cabo paralelo – este cabo integra os dispositivos restantes com o sistema de gerenciamento.

A Figura 3.2 ilustra esta forma de comunicação dos dispositivos e softwares do projeto.

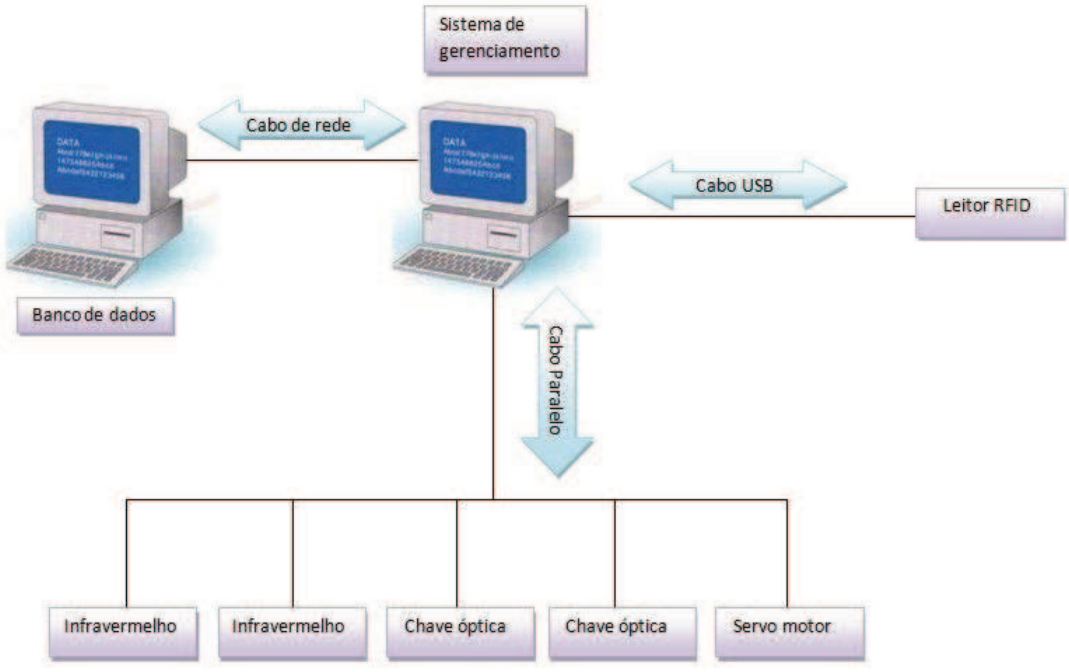

Figura 3.2 – Forma de comunicação dos dispositivos e softwares do projeto.

3.1.2 – Funcionamento Básico do Projeto

O funcionamento básico deste projeto pode ser explicado em cinco etapas de verificações. Estas etapas são verificações que constituem a lógica do sistema. A Figura 3.3 ilustra cada uma dessas etapas.

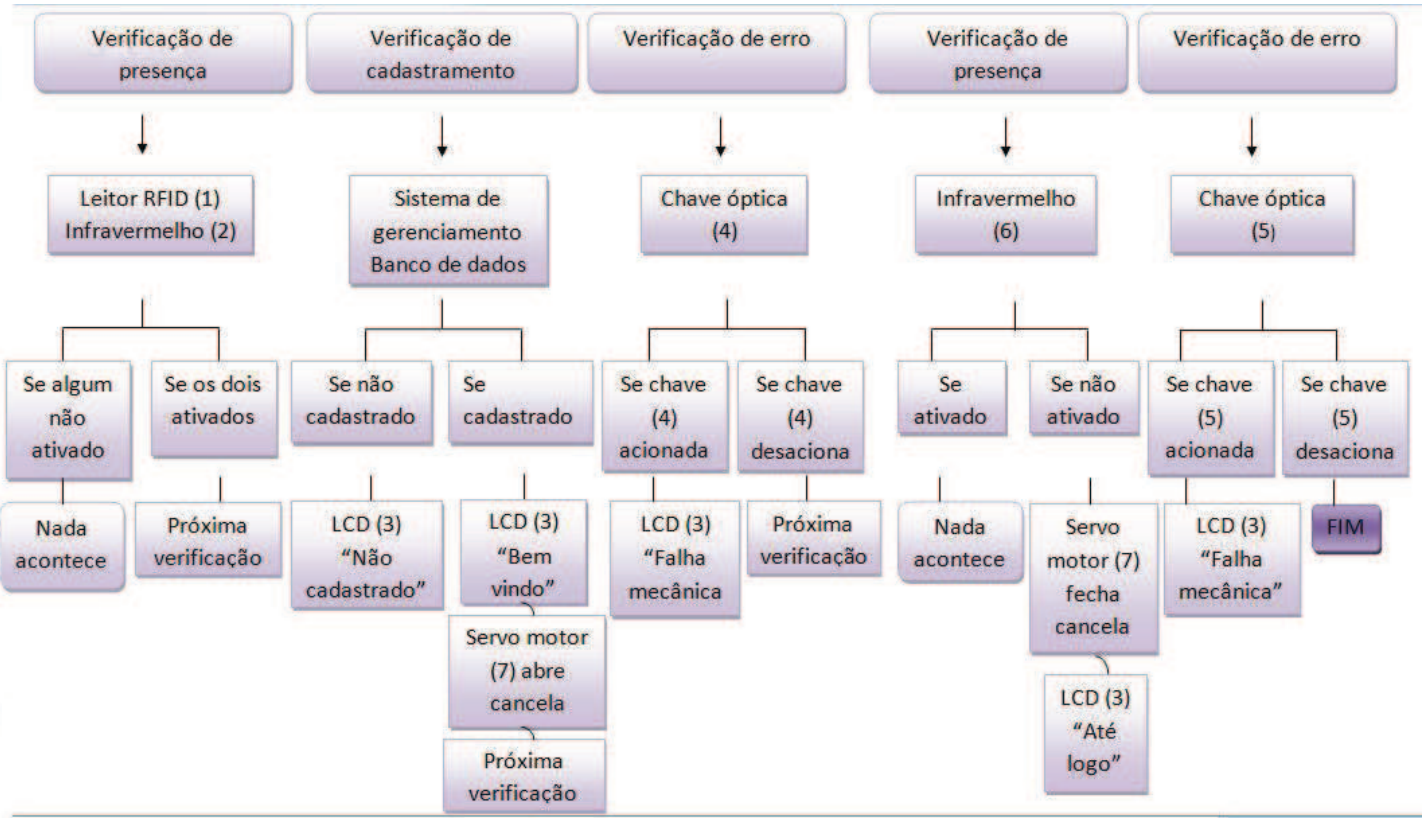

Figura 3.3 – Etapas de verificação.

Os números que aparecem ao lado dos dispositivos da Figura 3.3 servem de comparação com a Figura 1.1 do capítulo 1. Com essas figuras pode-se entender melhor o funcionamento do protótipo. Quando o carro com a etiqueta chegar perto da cancela se inicia a verificação de presença. Tanto o leitor RFID (1) quanto o primeiro sistema infravermelhos (2) irão identificá-lo. Caso não haja resposta positiva de um dos dois, absolutamente nada acontecer. Caso contrário, se iniciará a etapa de verificação de cadastramento.

Na etapa de verificação de cadastramento o sistema de gerenciamento realiza uma consulta no banco de dados para verificar se a etiqueta acoplada no veículo está cadastrada no sistema. Caso negativo, o LCD (3) exibe a mensagem ‘’Não cadastrado’’. Já se a etiqueta do veículo for cadastrada o sistema de gerenciamento envia um sinal para que o servo motor (7) gire 90 graus e abra a cancela e exibirá a mensagem “Bem vindo” no LCD(3).

Há também uma etapa de verificação de erro. Faz parte desta etapa a chave óptica (4). Essa chave é responsável por verificar se a cancela abriu totalmente. Para isso o feixe de luz emitido pela chave óptica precisa ser interrompido. Caso isto não aconteça, o sistema de gerenciamento notifica ao administrador do sistema que houve uma "Falha mecânica" e exibe a ocorrência da falha no LCD (3).

A próxima etapa de verificação é novamente de presença. Quem participa desta quarta etapa é o infravermelho (6). Caso este infravermelho esteja acionado é porque o veículo ainda esta passando por debaixo da cancela e então nada irá acontecer. Quando o veículo já passou pela cancela e já entrou na garagem esse infravermelho é desacionado. Com isso o sistema envia ao servo motor um sinal para que gire -90 graus e feche a cancela. Nesta etapa há a confirmação de que o carro entrou no estacionamento, com isso o sistema registra no banco de dados o dia, horário e o veículo que ingressou na garagem.

A última etapa de verificação é também de erro. Nela é a chave óptica (5) que faz parte desta etapa. Essa chave faz a verificação se a cancela está totalmente fechada. Para isto, o feixe de luz emitido na chave óptica precisa ser interrompido pela cancela. Caso isto não aconteça, o sistema de gerenciamento notifica ao administrador do sistema que houve uma "Falha mecânica" e exibe a ocorrência da falha no LCD (3). Com essa verificação que se tem o fim de todo o processo.

3.2 – Dispositivos Eletrônicos do Projeto

Os dispositivos eletrônicos utilizados neste projeto foram os seguintes:

- Servo motor;
- Chaves ópticas;
- Sistemas infravermelhos;
- Leitor RFID.

O leitor RFID é o único dispositivo deste protótipo que está conectado via porta USB como ilustrou a Figura 3.2 deste capítulo. Já os outros foram conectados via porta paralela. Esta comunicação entre a porta paralela e os dispositivos foi feita da seguinte forma como mostra a Figura 3.4 e a Quadro 3.1:

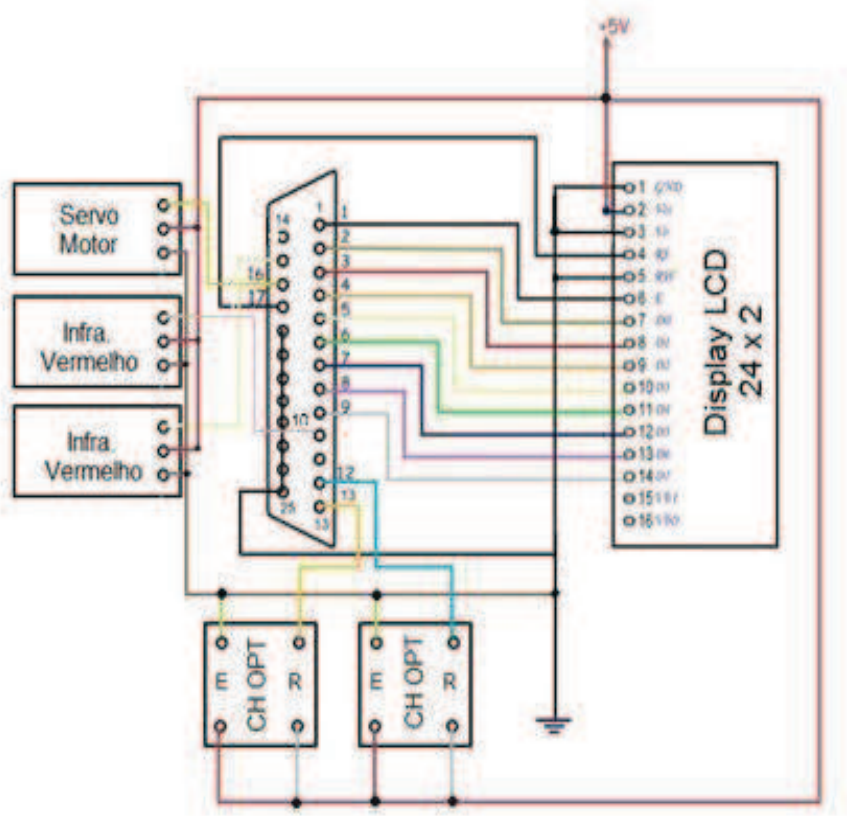

Figura 3.4 – Ligação da porta paralela com os dispositivos do projeto.

Quadro 3.1 – Legenda da Figura 3.4.

Pinos Porta Paralela	**Hardwares**
Pino1 - controle	LCD - Pino 6 – habilita/desabilita
Pino 2 ao 9 - dados	LCD - Pino 7 ao 14 – dados
Pino 10 – estado	Infravermelho (sinal)
Pino 11 – estado	DISPONÍVEL
Pino 12 – estado	Chave óptica (sinal)
Pino 13 – estado	Chave óptica (sinal)
Pino 14 – controle	DISPONÍVEL
Pino 15 – estado	Infravermelho (sinal)
Pino 16 – controle	Servo Motor (sinal)
Pino 17 – controle	LCD – Pino 4 – entrada controle/dados
Pinos 18 ao 25 – terra	TERRA

3.2.1 – Especificações dos Dispositivos Utilizados

Neste subitem são dadas as especificações dos dispositivos utilizados. Também são explicados os circuitos necessários para a montagem do protótipo.

3.2.1.1 – Servo Motor para Aeromodelo Hextronik HXT900

Para a simulação de uma cancela foi usado na maquete um servo para aeromodelo Hextronik de dimensões 31.8mm por 28 mm. Sua tensão de operação é de 5 volts. Seu torque chega a 1.60 kg/cm e sua velocidade pode chegar a 0.12 seg/60°. Por ser um protótipo de pequeno porte este servo motor responde perfeitamente pelas expectativas esperadas para o projeto. A Figura 3.5 é uma foto do servo motor utilizado.

Figura 3.5 – Servo motor utilizado.

3.2.1.2 – Chave Óptica

A chave óptica foi utilizada neste projeto para detectar se houve alguma falha com a cancela ao abrir e fecha-la. Foram postas duas chaves ópticas de forma que uma confirma se a cancela abriu totalmente e, a outra se fechou por completo. A Figura 3.6 mostra as posições que as chaves ópticas foram colocadas no protótipo.

Figura 3.6 – Posições das chaves ópticas no protótipo.

Para ligar o emissor da chave óptica é necessária uma tensão de 3.3 volts. De acordo com a lei de Ohms para que essa voltagem seja alcançada com uma corrente de 7.7mA é necessário um resistor de 220Ω. Já o resistor de 4.7k Ω foi usado na saída do receptor por convenção do *datasheet* da chave óptica. A Figura 3.7 mostra o esquemático deste circuito.

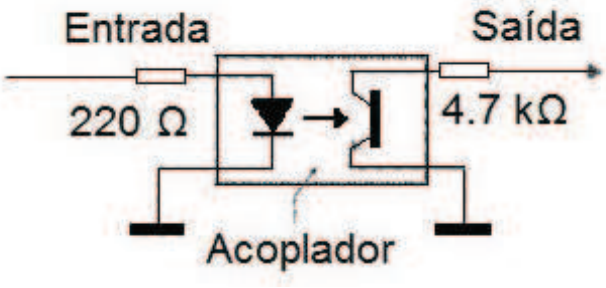

Figura 3.7 – Circuito chaves ópticas

3.2.1.3 – Sensores Infravermelhos

Para que se tenha certeza que há algum carro querendo entrar na garagem, foi desenvolvido um sensor infravermelho antes da cancela. Com a sinalização de que há um carro em frente a cancela, a cancela só se abrirá com a confirmação do leitor RFID.

Outro sensor infravermelho foi instalado após a cancela. Este serve para o sistema saber que o carro já passou pela cancela e pode ser fechada.

Para os sistemas infravermelhos utilizados neste protótipo foram utilizados dois circuitos: um emissor e um receptor. Estes circuitos foram postos um ao lado do outro de forma que a identificação do objeto é feita por detecção por reflexão. A Figura 3.8 ilustra o circuito do emissor infravermelho.

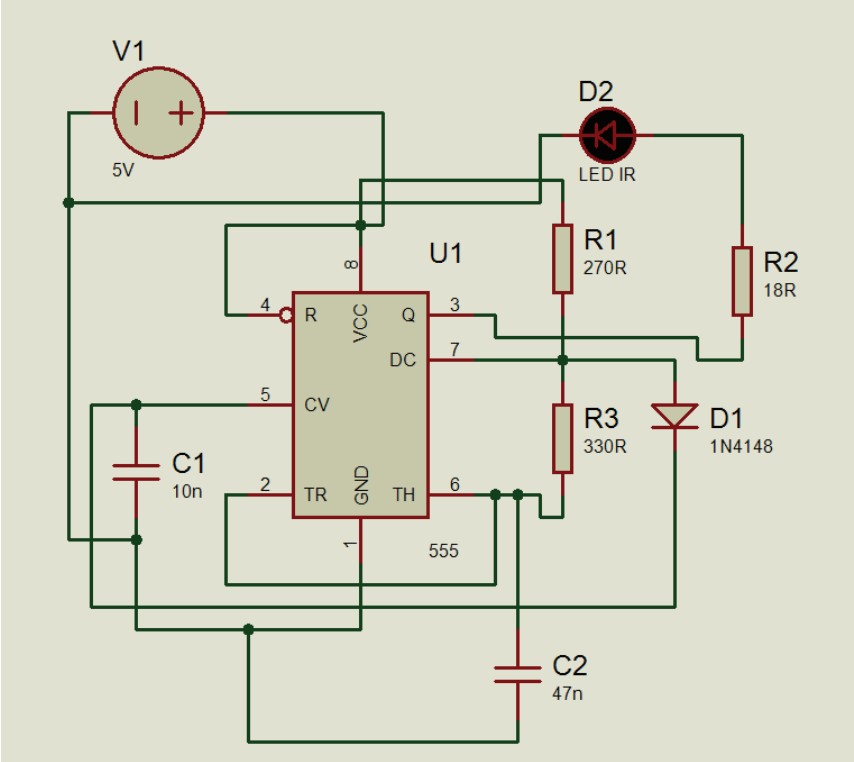

Figura 3.8 – Circuito emissor infravermelho.

Para o circuito emissor infravermelho foi utilizado um oscilador NE 555 configurado no modo astável gerando um sinal com frequência de 33kHz no pino de saída 3 de acordo com os resistores R1, R3 e capacitor C2 dada pela formula:

$$F = \frac{1}{T} = \frac{1.44}{[(R1 + 2 * R3) * C2]}$$

Onde F é a frequência, T o período, R1 e R3 as resistências e C2 o capacitor. O capacitor C1 é ligado ao pino 5 para filtragem de ruídos, sendo este valor fixo e de acordo com as especificações do fabricante constantes no *datasheet* dele. Este pino é usado para controle de tensão. O resistor R2 faz o ajuste da potência do emissor infravermelho definindo a distância que o sensor detecta o objeto.

A Figura 3.9 ilustra o oscilador NE 555 e seus pinos.

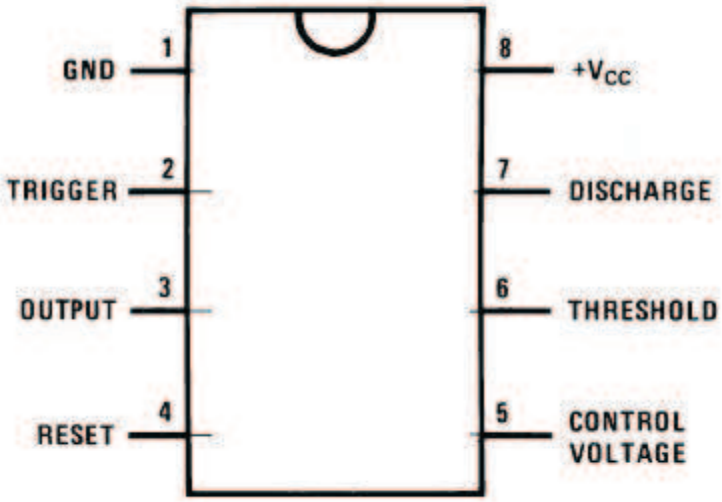

Figura 3.9 – NE 555 e seus pinos.

FONTE: (*Datasheet* NE 555.)

A Figura 3.10 mostra o circuito receptor infravermelho.

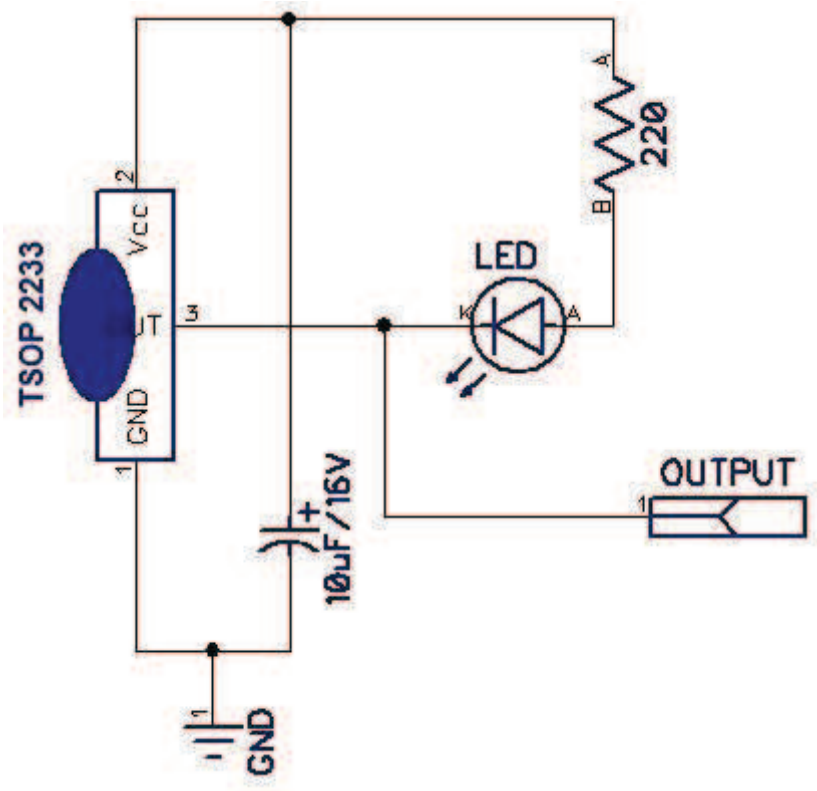

Figura 3.10 – Circuito receptor infravermelho.

Para o circuito receptor infravermelho foi utilizado o TSOP 2233 que detecta luz infravermelha na faixa de 33 KHz. Foi utilizado também um capacitor de 10µF para filtrar ruídos da fonte. O pino 1 é o terra. No pino 2 está ligado a fonte de 5 volts. Essa fonte manda

5 volts para o TSOP 2233 e 5 volts para o resistor 220 ohms. O resistor de 220 Ohms é necessário para que o LED se acenda a uma voltagem de 3.3 volts. Pela Lei de Ohm V= R*I (tensão = resistência * corrente), para que se chegue a uma voltagem de 3.3 volts com uma corrente de 7.7mA é necessário uma resistência de 220 Ohm.

Quando não há objeto detectado pelo TSOP 2233 a tensão de saída no pino 3 é de 3.3v. Assim, o catodo do LED está alimentado com 3,3 volts. O resistor 220 Ohms ocasiona uma queda de tensão de 1.7 volt desses 5 volts e fornece 3,3 volts para o anodo do LED. Assim não há diferença de potencial não acendendo o LED mantendo os pino 10 e 15 da porta paralela em nível alto.

Todavia, quando se detecta luz infravermelha na faixa de 33KHz, o TSOP 2233 baixa o nível do pino 3 de forma que o catodo do LED fique com tensão de 0 volts e o anodo (lado do resistor) fique com 3,3volts. Com isso há uma diferença de potencial e o LED acende. Assim é alterado o nível lógico da porta paralela para nível baixo dos pinos 10 e 15.

3.2.1.4 – Circuito de Alimentação

O circuito está alimentado com uma fonte de 16.5 volts. Todo o sistema trabalha com uma tensão de 5 volts. Para isso é necessário um ajuste da tensão de entrada de 16.5 volts para 5 volts. A Figura 3.11 mostra o circuito do regulador de tensão usado neste projeto.

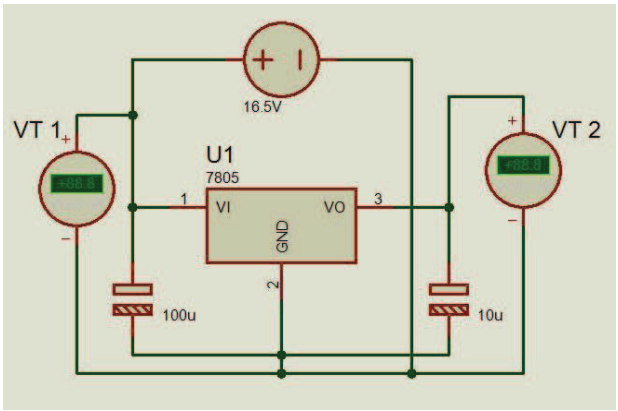

Figura 3.11 – Circuito regulador de tensão.

Foi utilizado o regulador de tensão KIA 7805A. Este regulador possui 3 pinos. O pino 2 é o terra, o pino 1 é a entrada que recebe os 16.5 volts e o pino 3 é a saída que alimenta o restante do circuito com 5 volts. Para filtrar ruídos foram usados os capacitores 100µF/25V e

10µF/25V de acordo com o *datasheet* do fabricante. Também foi adicionado um LED e um resistor para indicar que o circuito está ligado.

3.2.1.5 – O Leitor *Thing Magic* M5e

O leitor RFID foi emprestado pelo o Instituto Omnis que é uma instituição de direito privado, sem fins lucrativos, sediada em Brasília/DF. Dentre os modelos disponíveis no Instituto, o sugerido foi o *Thing Magic* M5e. O *Thing Magic* M5e é um leitor de mesa (RFID Desktop Reader) que utiliza a frequência UHF. Este leitor possui distância de leitura máxima de 31 cm devido à potência de sua antena. Levando em conta que o projeto é um protótipo a distância de 31 cm é suficiente. Este leitor também possui a vantagem de ler apenas uma etiqueta por vez. Estas características fazem o *Thing Magic* M5e se enquadrar ao esperado para a maquete de pequeno porte. Como já dito, este leitor se conecta ao computador via porta USB. Na Figura 3.12 o modelo *Thing Magic* M5e.

Figura 3.12 – Modelo *Thing Magic* M5e.
FONTE: (THINGMAGIC).

Para a comunicação do leitor RFID com as etiquetas ser bem sucedida, eles precisam compartilhar um mesmo protocolo de comunicação. O protocolo funciona como um conjunto de regras que define a forma de comunicação entre o leitor e a etiqueta.O leitor RFID (assim como as etiquetas utilizadas) utilizam o protocolo GEN2. Ou seja, quaisquer outras etiquetas que também utilizem o GEN2 podem ser lidas por este leitor.

3.3 – Software

Para fazer a integração dos dispositivos com o software foi usado a IDE (*Integrated Development Environment* – ambiente integrado para desenvolvimento) Dev – C++ que utiliza os compiladores gcc (compilador C) e g++ (compilador C++). Também foi utilizado o servidor de banco de dados MySQL Server 5.1 para o cadastro das etiquetas RFID. Para a integração do sistema com o leitor RFID foi utilizado trechos do código do software de demonstração do fabricante (ArbSer).

3.3.1 – Funções Principais Utilizadas no Programa

Este subitem apresenta e explica as principais funções utilizadas no programa gerado.

3.3.1.1 – Porta Paralela DB25

Para a comunicação com esta porta uma DLL (*Dynamic Link Library*) do Windows foi usada (inpout32). Esta DLL contém as funções que fazem controle de I/O (entrada e saída). Foram usadas duas funções principais que utilizam parte desta DLL para a comunicação com a porta paralela. A primeira do tipo void (outB), recebe dois parâmetros: o endereço da porta (short end) e o dado a ser enviado (short data). Esta função irá somente executar o que foi pedido e não retornará nenhum valor.

A segunda do tipo int (inpB) recebe um inteiro que contém o endereço da porta a ser lida. Esta função é executada e retorna os bits lidos da porta em decimal.
No anexo A encontram-se essas duas funções.

3.3.1.2 – *Display* LCD

As principais funções utilizadas para o *display* LCD foram retiradas da fonte ELECTROSOFTS e modificadas para a comunicação com a porta paralela. As funções utilizadas foram:

- lcd_init () – esta função inicializa e configura o *display* LCD para receber dados;
- lcd_puts (“palavra”) – esta função recebe uma string (palavra) e lê caracter por caracter enviando-o para o LCD através da função lcd_write ();

- lcd_write() – recebe um caracter e transmite para o *display*;
- lcd_goto() – recebe dois parametros (linha e coluna) movendo o cursor para a posição indicada;
- lcd_clear() – limpa o buffer do LCD.
- lcd_cursor() – configura o tipo de cursor a ser exibido no *display*.

No anexo B encontram-se essas funções utilizadas.

3.3.1.3 – RFID

Do programa ArbSer foram utilizadas 11 funções para a integração do leitor RFID com o restante do sistema, como ler etiquetas e enviar comandos ao leitor RFID:

- AbrePorta() – esta função inicia a comunicação com a porta USB;
- FechaPorta() – encerra comunicação com a porta USB;
- LerDados() – esta função captura as informações recebidas pela porta USB e armazena em uma variável (*buffer) juntamente com o tamanho desta mensagem (MaxBytes);
- enviaMsg() – esta função cria um pacote para a comunicação do computador com o leitor, armazenando a mensagem em um buffer antes de ser transmitido como mostra a Quadro3.2:

Quadro 3.2- Padrão de mensagem.

Cabeçalho	Tamanho da mensagem	Comando	Mensagem	Controle de erro
1 Byte	1 Byte	1 Byte	0 a 250 Bytes	2 Bytes

FONTE: (*M5eFamily development guide)*.

Passando os parâmetros numArgs (quantidade de códigos em hexadecimal a serem armazenados no buffer) e *args (códigos a serem armazenados) para a função, que concatena e forma a mensagem a ser transmitida.

- Ascii2Nibble() – recebe os códigos que foram armazenados no buffer e os converte para hexadecimal de 4 bits.

- Ascii2Byte() – recebe os códigos que foram armazenados no buffer e os converte para hexadecimal de 8 bits.
- SendData() – envia a mensagem armazenada no buffer para o leitor.
- MSG_receiveMsgObj() – cria uma estrutura de dados para o pacote recebido do leitor no seguinte formato como mostra a Quadro 3.3:

Quadro 3.3 – Estrutura contendo os dados recebidos do leitor.

Cabeçalho	Tamanho da mensagem	Comando	Estado	Dados	Controle de erro
1 Byte	1 Byte	1 Byte	2 Bytes	0 a 248 Bytes	2 Bytes

FONTE: (*M5eFamily development guide).*

- CRC_calcCrc8() – algoritmo de cálculo do controle de erro.
- Configura_RFID() – envia uma sequência de mensagens para o leitor RFID que configura várias de suas funções como protocolo, potência da antena entre outros.
- StartBuscaTag() – esta função envia ao leitor um comando para que procure etiquetas RFID durante um determinado tempo (1 segundo). A função armazena em uma variável a primeira etiqueta encontrada ou expira o tempo de procura (timeout).

Estas funções se encontram desenvolvidas no anexo C.

3.3.1.4 – Banco de Dados

No projeto foi utilizado o MySQL Server 5.1 instalado em uma máquina secundária para que não haja concorrência entre os processos do banco de dados e do sistema de gerenciamento. Os dados de conexão com o banco de dados são os seguintes:

```
#define HOST "192.168.200.1" //IP da máquina secundária
#define USER "usr_rfid" //Usuário com privilégios de conexão
#define PASS "usr_rfid" //Senha do usuário
#define DB "rfid" //Banco de dados
```

As funções utilizadas para realizar as *querys* são:

- consulta_bd() – esta função realiza uma consulta na tabela (tb_cadastro) verificando em cada um dos registros se a etiqueta RFID que foi lida pelo leitor esta cadastrada.

- cadastrar_tag() – esta função requer que alguma etiqueta RFID tenha sido lida e a cadastra no banco de dados juntamente com as informações como nome do utilizador e placa do carro.
- descadastra_tag() – esta função requer que alguma etiqueta tenha sido lida e a elimina completamente os registros referentes da etiqueta do banco de dados com confirmação do usuário.
- consulta_registros() – esta função faz uma consulta no banco de dados retornando na tela do administrador todas as etiquetas cadastradas no sistema.
- consulta_controle() – esta função faz uma consulta no banco de dados retornando na tela do administrador todos os registros de entrada e saída do estacionamento.

Estas funções utilizadas para realizar as *querys* encontram-se no apêndice 2.

3.3.1.5 – Controle do Servo Motor

Para o controle do motor HEXTRONIK HXT 900 foi utilizada a seguinte função:

- CtrlCancela () – esta função gera um pulso que é enviado ao servo motor controlando a abertura e fechamento da cancela.

Esta função se encontra junto com o programa principal no apêndice 1.

3.4 –Testes e Resultados

Após alguns estudos, foi decidido que para controlar o servo motor, os infravermelhos e as chaves ópticas iria ser usado um a porta paralela. Por isso os testes se iniciaram com o controle desta porta. A Figura 3.13 mostra o cabo construído para os primeiros testes.

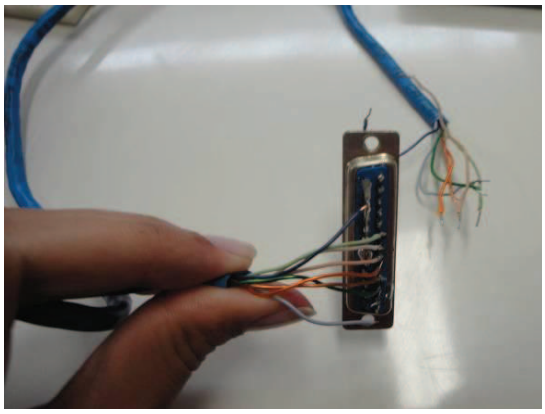

Figura 3.13 – Cabo paralelo construído para os primeiros testes.

Com este cabo o primeiro teste a ser executado foi o de controle de LEDs (*Light Emitting Diode*) como mostra a Figura 3.14.

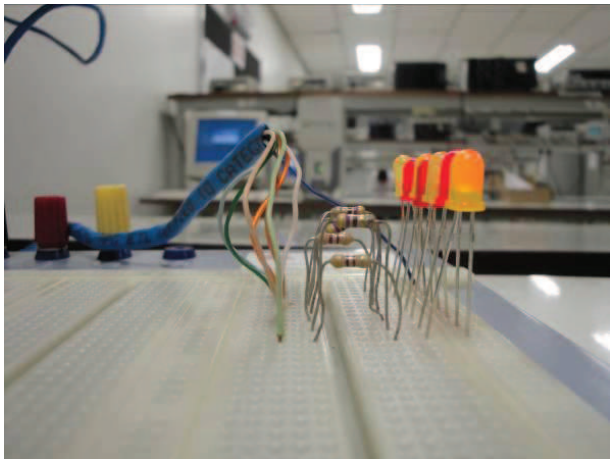

Figura 3.14 – Teste controle LEDs.

O próximo teste executado foi o controle do servo motor. Os primeiros testes não foram bem sucedidos. E após vários experimentos foi concluído que o primeiro motor testado era um servo modificado. Este servo motor não possuía o seu potenciômetro que é necessário para o controle de posição de se eixo. Isto não permitiu posições precisas como mostra a Figura 3.15.

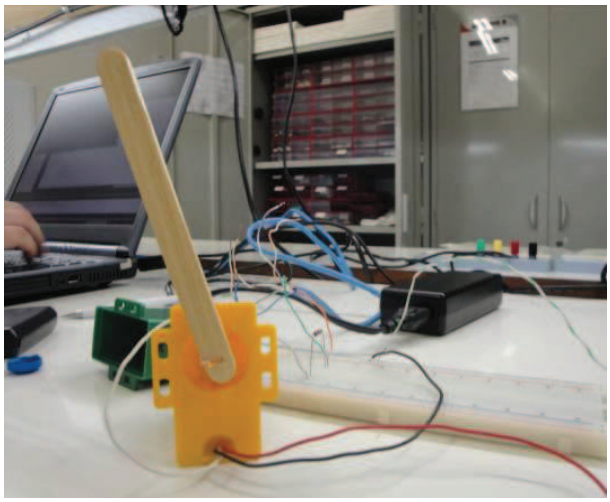

Figura 3.15 – Servo motor modificado.

Com a troca do motor modificado por outro servo os testes foram executados com sucesso como mostra a Figura 3.16.

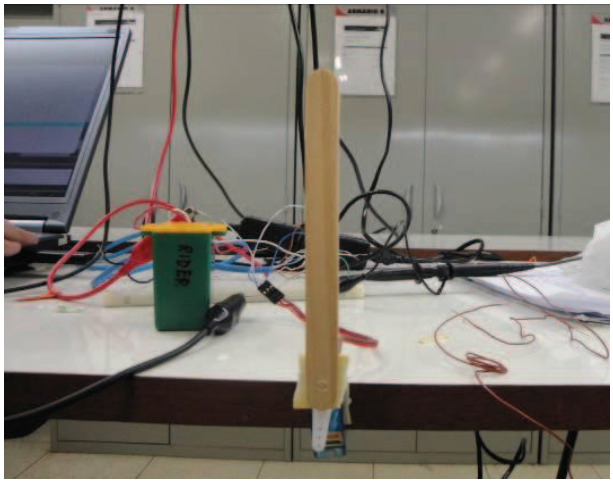

Figura 3.16 – Novo Servo motor.

Em seguida os circuitos para funcionamento do infravermelho foram testados com êxito. A Figura 3.17 mostra o teste com estes circuitos.

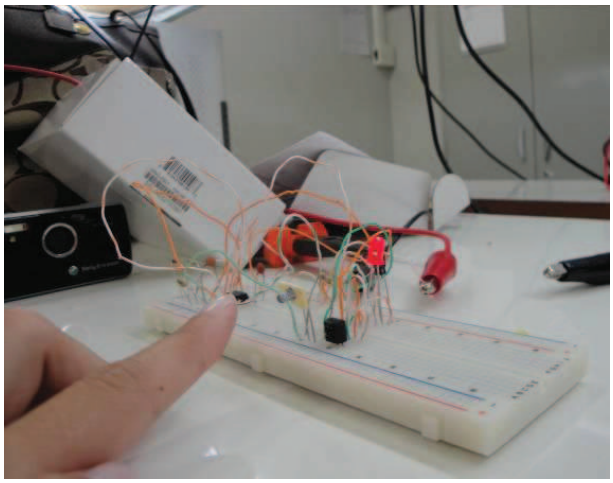

Figura 3.17 – Teste com infravermelho.

.

O próximo passo foi integrar o servo motor com o infravermelho de modo que o motor só funcionaria caso o infravermelho fosse acionado. Na Figura 3.18 o infravermelho é acionado por um objeto (multímetro) e o motor funciona (abrindo a cancela).

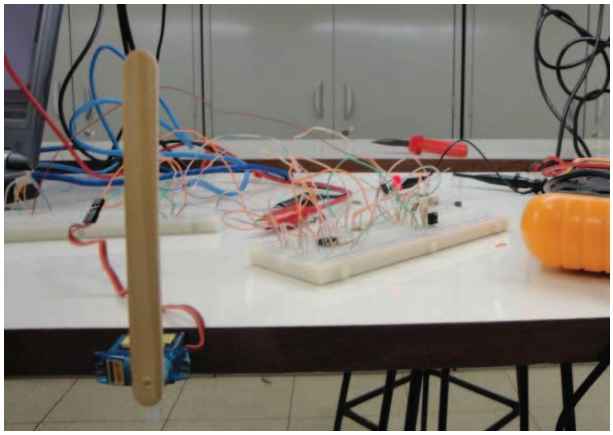

Figura 3.18 – Infravermelho acionado e cancela aberta.

A Figura 3.19 mostra o programa avisando que a cancela esta aberta.

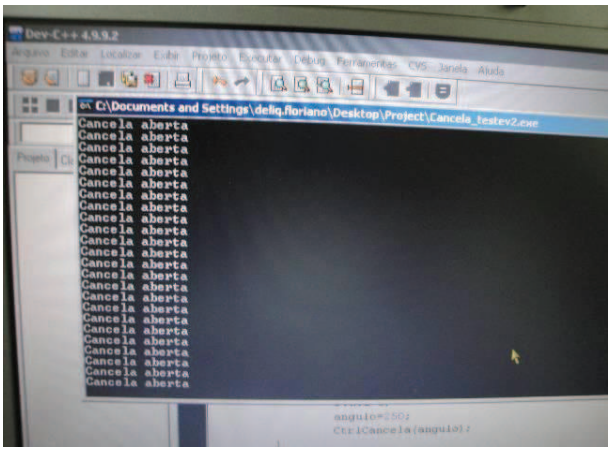

Figura 3.19 – Programa avisando que a cancela esta aberta.

Na Figura 3.20 o objeto (multímetro) é retirado da frente do infravermelho fechando a cancela.

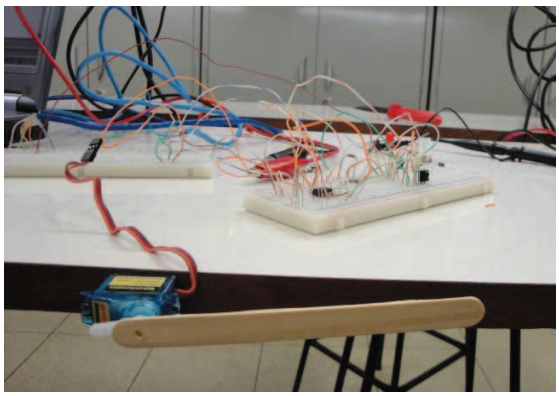

Figura 3.20 – Cancela fechada.

A Figura 3.21 mostra o programa avisando que a cancela esta fechada.

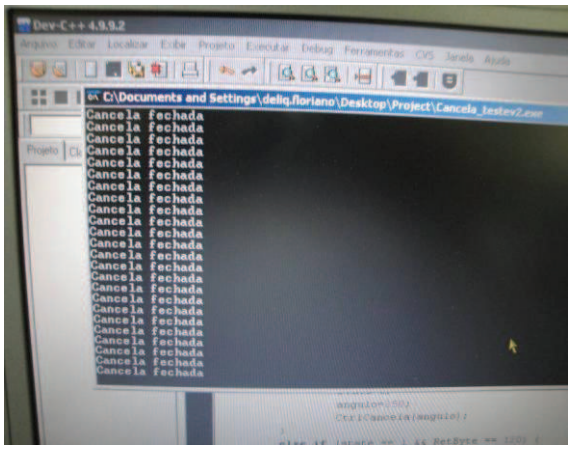

Figura 3.21 – Programa avisando que a cancela esta fechada.

A próxima etapa de testes foi acrescentar mais um infravermelho. Um desses infravermelhos fica antes do servo motor e o segundo fica depois dele. Com isto, o motor abre a cancela apenas quando o primeiro circuito de infravermelho é acionado. Após os dois circuitos infravermelhos desativados o motor fecha a cancela. As Figuras 3.22, 3.23, 3.24, 3.25 e 3.26 demonstram este processo.

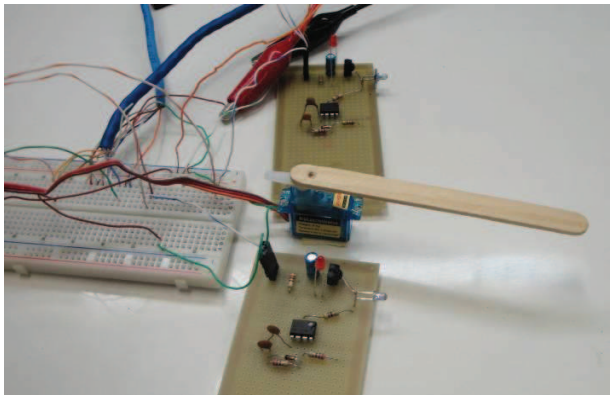

Figura 3.22 – Sistema com os dois infravermelhos.

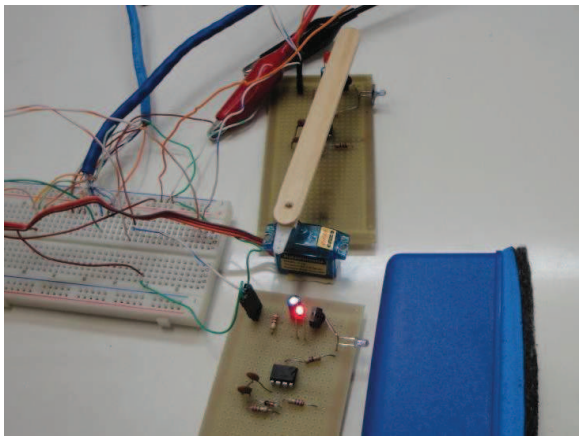

Figura 3.23 – Primeiro infravermelho sendo acionado.

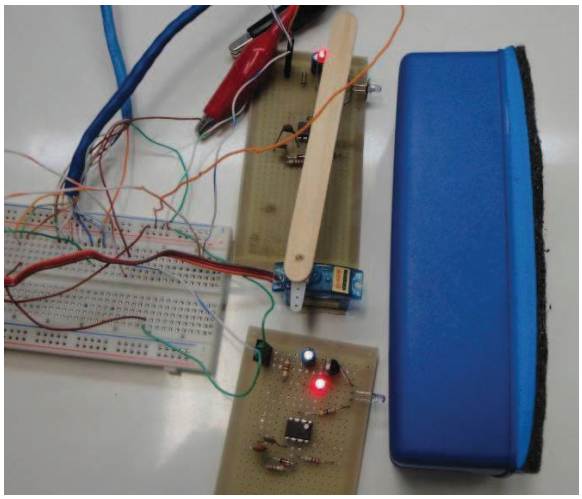

Figura 3.24 – Os dois infravermelhos acionados.

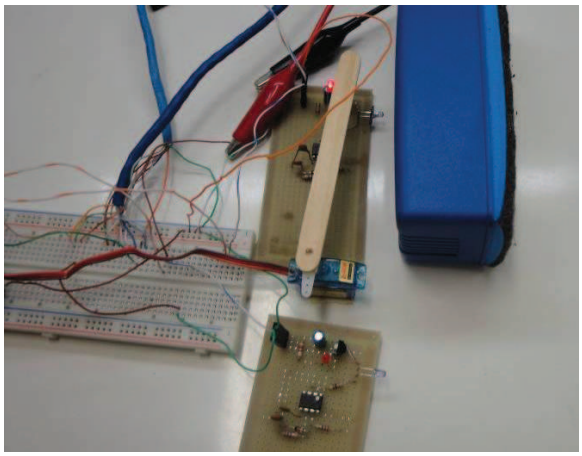

Figura 3.25 – Apenas o segundo infravermelho acionado.

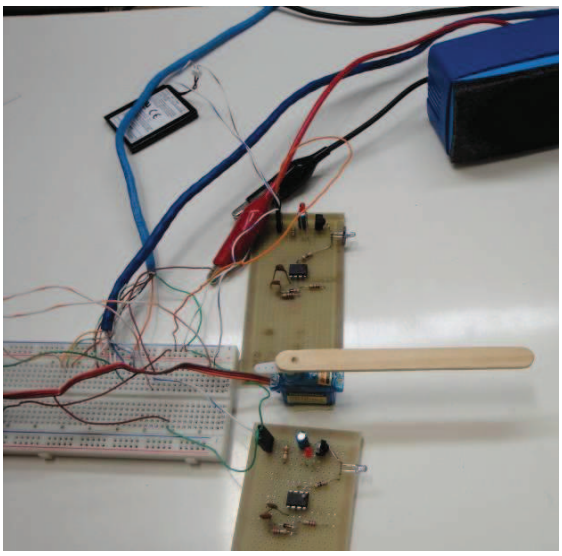

Figura 3.26 – Nenhum infravermelho acionado.

Logo em seguida foi feito o teste com o visor LCD. Primeiramente foi testado se ele ligava com o comando do programa como mostra a Figura 3.27.

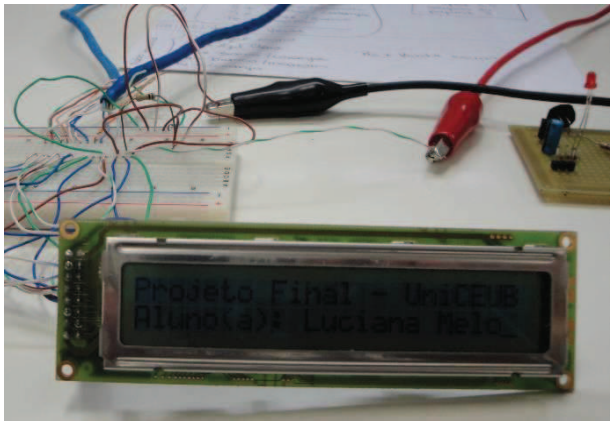

Figura 3.27 – LCD funcionando.

Depois foi feito a integração do LCD com o circuito do infravermelho e do servo motor. Com esta integração o LCD foi programado para mostrar mensagens como “Abrir cancela” e “Fechar cancela”. A Figura 3.28 ilustra esta interligação.

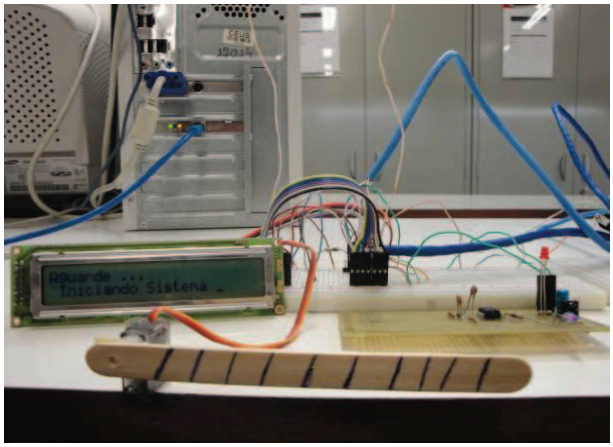

Figura 3.28 – Integração LCD com o infravermelho e o servo motor.

Os próximos testes foram com os circuitos das chaves ópticas. E por fim, os últimos testes foram feitos com o leitor RFID como mostra a Figura 3.29.

Figura 3.29 – Testes com o leitor RFID.

3.5 – Simulação

A simulação do projeto é uma maquete. Nesta maquete tudo que já foi testado é aplicado.

Para a montagem desta maquete o primeiro passo foi trocar os circuitos infravermelhos que estavam em protoboards por placas de fibra de vidro. A Figura 3.30 mostra este processo.

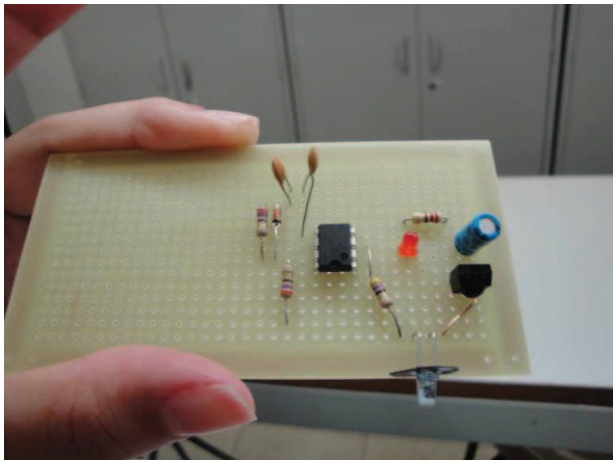

Figura 3.30 – Placa de fibra de vidro.

O cabo que tinha sido fabricado também foi trocado por novo cabo como ilustra a Figura 3.31.

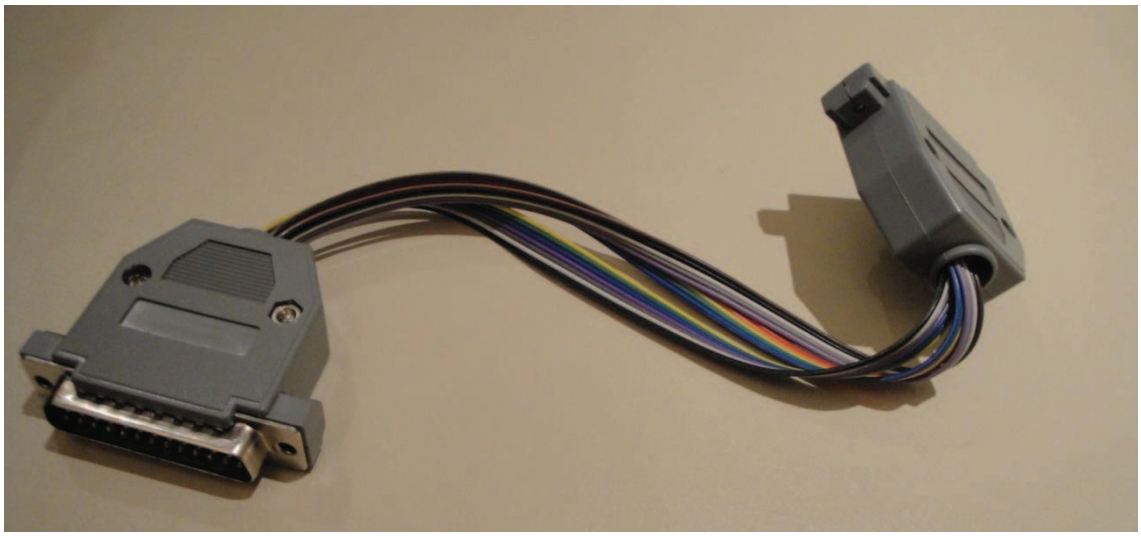

Figura 3.31 – Cabo novo.

O LCD, as chaves ópticas, os infravermelhos e o servo motor foram conectados aos pinos da porta paralela DB25. A Figura 3.32 a placa de alimentação e integração dos dispositivos eletrônicos com o DB25.

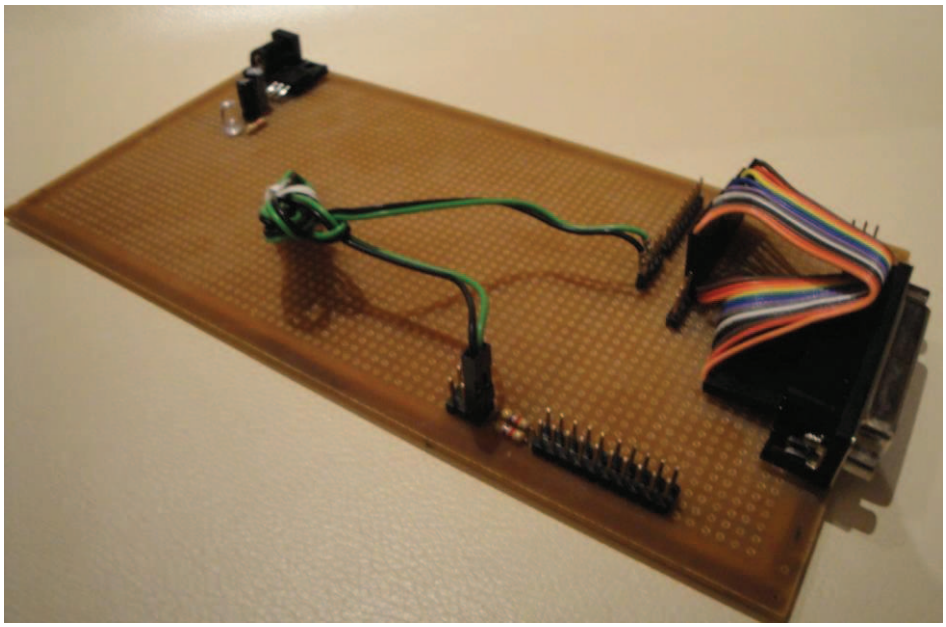

Figura 3.32 – Placa de alimentação e integração dos dispositivos eletrônicos.

Toda a estrutura física da maquete foi montada usando madeira e alumínio. A Figura 3.33 mostra este processo.

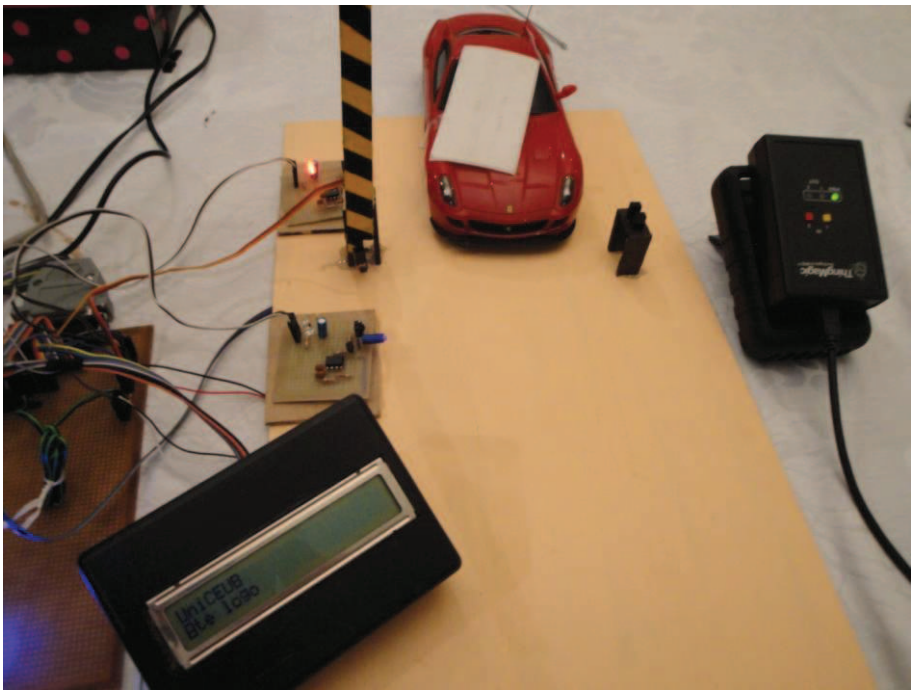

Figura 3.33 – Maquete de madeira e alumínio.

Após a montagem de toda a estrutura a simulação esta pronta para ser executada com sucesso.

CAPÍTULO 4 - CONCLUSÃO

4.1 – Conclusões

Este projeto apresenta uma proposta de automação de garagens usando a tecnologia RFID. Para isso foi construído um protótipo em escala reduzida como explicado neste trabalho. Para que este projeto seja utilizado em padrões reais basta adequá-lo à realidade trocando os materiais utilizados de baixa escala. Como exemplo, uma troca aconselhada seria um motor de indução no lugar do servo motor.

O objetivo geral do trabalho é amenizar a falta de segurança em prédios utilizando, para isto, a tecnologia RFID. Testes com etiquetas RFID não autorizadas e com veículos que não possuíam etiquetas RFID foram realizados. Em nenhum destes casos as entradas dos veículos foram autorizadas. Concluindo então que o sistema desenvolvido trará uma maior segurança para os usuários do condomínio. O controle do acesso utilizando o sistema RFID diminuirá consideravelmente a entrada de estranhos e de não autorizados. O registro de dados de quem e quando entrou ou saiu do prédio diminuirá furtos.

As tarefas propostas para a realização deste trabalho foram executadas com sucesso na montagem da maquete. As informações do veículo foram capturadas pelo sistema RFID, armazenadas em um de banco de dados e restritas aos administradores do sistema. A entrada de apenas autorizados foi realizada através de etiquetas e leitores RFID acoplados com um sistema de infravermelhos. A programação de um sistema foi feita através dos programas desenvolvidos em linguagem C integrando os sistemas mecânicos com o sistema RFID. Circuitos com chaves ópticas para garantir se a cancela abriu e fechou corretamente foram implementados funcionando perfeitamente. E para que o motorista do veículo e o administrador do prédio fossem avisados sobre os acontecimentos foi implementado um visor LCD que avisa caso tenha uma falha mecânica ou alguém não cadastrado tentando entrar prédio. Por fim, todos os registros de entradas e saídas do prédio são armazenados em um banco de dados. Alcançando assim todos os objetivos e tarefas propostas para este projeto.

Os resultados obtidos com o protótipo foram satisfatórios para os objetivos iniciais. A principal vantagem deste proposto é evitar ou até mesmo eliminar entradas indesejadas de estranhos. Porém este trabalho se limita a isto não considerando assim outros aspectos de segurança.

O modelo proposto neste trabalho trás varias vantagens apesar do seu alto custo inicial. O sistema RFID poderá ser incrementado no futuro trazendo novas soluções sem que para isso precise mudar de tecnologia ou aparelho. Vantagem essa que, por exemplo, algumas tecnologias, como código de barras, não conseguiriam. Mostrando assim que seu custo/benefício é considerável valendo o investimento.

4.2 - Sugestões para Trabalhos Futuros

Com os estudos e pesquisas para o desenvolvimento deste trabalho surgiram novas idéias para projetos futuros. A tecnologia RFID esta sendo bastante estudada e aplicada em diversas áreas por trazer muitas vantagens em comparação a outras tecnologias já existentes.

Não fugindo do tema "garagem automatizada" as sugestões para trabalhos futuros estão citadas a seguir:

- Levar a idéia para um patamar mais realista aplicando em verdadeiros portões ao invés de apenas simular. Além disso, acrescentar ao trabalho mais itens de segurança como câmeras filmadoras integradas ao sistema.

- Usar o RFID em garagens públicas como shoppings e prédios comerciais para o sistema de pagamentos de forma que o usuário não precisasse pagar imediatamente no dia.

- Usar o RFID em garagens públicas como shoppings e prédios comerciais para a localização de vagas disponíveis.

REFERÊNCIAS BIBLIOGRÁFICAS

Artigos técnicos. Servo motores.
Em: http://www.nei.com.br/artigos/artigo.aspx?i=67
Acesso em: 20/02/2010.

APOSTILANDO. Evolução da linguagem C.
Em: http://www.apostilando.com/pagina.php?cod=1
Acesso em: 20/12/2009.

BOYLESTAD, Robert; NASHELSKI, Louis. Dispositivos Eletrônicos e teoria de circuitos. 5º edição. Editora afiliada. Rio de Janeiro 1994.
Páginas:

BRAGA, Newton. Acopladores e Chaves ópticas.
Em: http://www.newtoncbraga.com.br/index.php/como-funciona/872-acopladores-e-chaves-opticas-art120.html
Acesso em: 25/04/2010.

Condos Using RFID to Keep Access in Check
Em: http://www.rfidjournal.com/article/articleview/2341/1/1/
Acesso: 15/12/2009.

Datasheet NE555.
Em: http://www.datasheetcatalog.com/datasheets_pdf/N/E/5/5/NE555.shtml
Acesso em: 15/04/2010.

DOBKIN, Daniel M.The RF in RFID: Passive UHF RFID in Practice.
Newnes. Massachusetts 2007.
Páginas 504.

ELECTROSOFTS. Interfacing the LCD module to Parallel Port
Em: http://electrosofts.com/parallel/lcd.html.
Acesso em: 15/05/2010.

FINKENZELLER, Klaus. RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification.
2º Edição. Wiley & Sons LTD. Munich 2003.
Páginas 446.

FRANCISCO, Antônio. Motores Eléctricos.
Editora ETEP - Edições Técnicas e Profissionais. Portugal 2008.
Páginas: 162.

GLOVER, Bill; BHATT, Himanshu. Fundamentos de RFID.
Alta Books. Rio de Janeiro 2007.

JAMSA, Kris; KLANDER, Lars. Programando em C/C++ A Biblia.
Editora afiliada- Sao Paulo 1999.
Páginas: 1012.

Jornal de Representação Comercial sdr
Em: http://www.sdr.com.br/Ideias/538.htm
Acesso em: 30/03/2010.

LicitaMais. Onda crescente de invasões com assaltos derruba mito de segurança de condomínios.
Em: http://licitamais.com.br/noticias/news/1907.html
Acesso em: 10/05/2010.

M5eFamily development guide

MARIOTTO, Paulo Antonio. Análise de circuitos elétricos.
Editora afiliada - Sao Paulo 2003.
Páginas: 371.

MARKS, Tecnologia e design. O que é RFID.
Em: http://www.marx.com.br/rfid/o-que-e-rfid
Acesso em: 21/11/2009.

MATIAS, Sandra Regina. Identificação por rádio frequência.
Em: http://www.wirelessbrasil.org/wirelessbr/colaboradores/sandra_santana/rfid_02.html
Acesso em: 10/11/2009.

MIGUEL, Afonso. Sensor Infravermelho.
Em: http://www.afonsomiguel.com/content/sensor-infra-vermelho
Acesso em: 12/01/2010.

MILANI, Andre. MySQL Guia do programador .
Novatec editora ltda - São Paulo 2008.
Páginas: 397.

NATALE, Ferdinando. Automação industrial.
6 º edição. Editora erica- São Paulo 2004.
Páginas: 234.

NILSSON, James; RIEDEL, Susan. Circuitos elétricos.
6 º edição. Editora LTC - São Paulo 2003.
Páginas 656.

PyroElectro. Servo Motor Control: Introduction
Em: http://www.pyroelectro.com/tutorials/servo_motor/index.html
Acesso em: 19/01/20010

ROGERCOM. Porta paralela.
Em: http://www.rogercom.com/
Acesso: 15/04/2010.

SABERELETRONICA. 10 circuitos com chaves ópticas.
Em: http://www.sabereletronica.com.br/secoes/leitura/11.
Acesso em: 16/05.2010.

SAKAMOTO, Juliana. RFID – Radio Frequency Identification.
Em: http://www.gta.ufrj.br/grad/09_1/versao-final/rfid/historico.html.
Acesso em: 10/11/2009.

SANGHERA, Paul. RFID+: Study Guide and Practice Exam.
Syngress Publishing. Massachusetts 2007
Páginas 352.

Secretaria de Segurança Pública.
Em: http://www.pm.df.gov.br/Default.asp?pag=noticia&txtCodigo=4309
Acesso em: 12/05/2010.

The university of Arizona - RFID Access for Garages.
Em: http://parking.arizona.edu/permits/rfid.php
Acesso em: 13/12/2009.

THINGMAGIC. The engine in RFID.
Em: http://www.thingmagic.com/
Acesso em: 25/05/2010.

THOMAZINI, Daniel; BRAGA DE ALBUQUERQUE, Pedro Urbano. Sensores Industriais Fundamentos e Aplicações.
1º Edição. Editora Érica. São Paulo 2005.

THORNTON, Frank.. RFID Security.
Syngress Publishing. Massachusetts 2006.
Páginas 264.

ANEXO A – COMUNICAÇÃO COM A PORTA PARALELA.

Fonte: (ROGERCOM)

```
void outB(short end, short data) {
    PtrOut outportb; //ponteiro que recebe os parâmetros
    if(hLib == NULL)//carrega a DLL, se nulo, a DLL não foi encontrada
    {
      printf("Erro. O arquivo inpout32.dll não foi encontrado.\n");
      getch();
    }
    outportb = (PtrOut) GetProcAddress(hLib, "Out32");//carregar outportb
<- DLL, Out32
    if(outportb == NULL) //Verifica se houve erro.
    {
      printf("Erro. A função Out32 não foi encontrada.\n");
      getch();
    }
    outportb(end,data);
}

int inpB(int end)// função que recebe um endereço e faz a leitura deste
endereço. 0x378
{
   PtrInp inportb;
   if(hLib == NULL)
   {
      printf("Erro. O arquivo inpout32.dll não foi encontrado.\n");
      getch();
   }
      inportb = (PtrInp) GetProcAddress(hLib, "Inp32");
   if(inportb == NULL) //Verifica se houve erro.
   {
      printf("Erro. A função Inp32 não foi encontrada.\n");
      getch();
   }
   return inportb(end);// retorna o valor do endereço lido
}
```

ANEXO B – FUNÇÕES DO *DISPLAY* LCD.

FONTE: (ELECTROSOFTS).

```
void lcd_init()//inicializa o LCD
{
      outB(CONTROL, inpB(CONTROL) & 0xDF);
      outB(CONTROL, inpB(CONTROL) | 0x08);
      lcd_write(0x0f);
      Sleep(20);
      lcd_write( 0x01);
      Sleep(20);
      lcd_write( 0x38);
      Sleep(20);
}
void lcd_write(char char2write)//envia caracter ASCII para o Display
{
      outB(DATA, char2write);
      outB(CONTROL,inpB(CONTROL) | 0x01); /* Set Strobe */
      Sleep(20);
      outB(CONTROL,inpB(CONTROL) & 0xFE); /* Reset Strobe */
      Sleep(20);
}
void lcd_puts(char *str2write)//recebe a string lê caracter por caracter e
envia pro LCD
{
      outB(CONTROL, inpB(CONTROL) & 0xF7);
      //RS=low: data
      while(*str2write)
            lcd_write(*(str2write++));
}
void lcd_goto(int row, int column)// Seleciona linha e coluna do LCD
{
      outB(CONTROL, inpB(CONTROL) | 0x08);
      if(row==2) column+=0x40;
      /* Add these if you are using LCD module with 4 columns
      if(row==2) column+=0x14;
      if(row==3) column+=0x54;
      */
      lcd_write(0x80 | column);
                                          }
void lcd_clear()//Apaga o buffer do LCD
{
      outB(CONTROL, inpB(CONTROL) | 0x08);
      lcd_write(0x01);
}
```

ANEXO C – RFID.

FONTE: (THINGMAGIC).

```
/*
-------------------------------------------------------------------------
-----
Comandos em Hexadecimal para a configuração da leitora RFID ThingMagic
-------------------------------------------------------------------------
-----
*/

char *boot[] = { "00", "04" }; //Inicializa a aplicação da leitora
char *SetaRegiao[] = { "01","97","01" }; //Seta a regiao de operação da
leitora
char *ProtocoloGEN2[] = { "02", "93", "00", "05" }; //Seta o protocolo a
ser utilizado na leitura/gravação de Tags
char *PotAntLeitura[] = { "02", "92", "08", "FC" }; //Seta potencia da
antena que lê as Tags em centi-dBm - 08FC = 2300 = 23dBm(100%)
char *Antena[] = { "02", "91", "01", "01" }; // Habilita somente uma das
duas antenas - Apenas para leitura

HANDLE hCom;

/*
-------------------------------------------------------------------------
-----
Definindo protótipos das funções
-------------------------------------------------------------------------
-----
*/
void enviaMsg(int numArgs, char *args[]);
int SendData(const char *sendBuf, int numBytesToSend);
int Ascii2Byte(char *asciiData, unsigned int *byteOut);
int Ascii2Nibble(char asciiData);
int AbrePorta();
void FechaPorta(void);
signed long int LerDados(void *buffer, int MaxBytes);
void MSG_receiveMsgObj(struct MsgObj *hMsg);
signed __int32 ReadData(void *readBuf, u32 maxNumBytesToRead);
void CRC_calcCrc8(u16 *crcReg, u16 poly, u16 u8Data);
void Configura_RFID(void);
u32 StartBuscaTag ();

/*
-------------------------------------------------------------------------
-----
Funcoes modificadas do software de demonstração 'ArbSer' do ThingMagic
-------------------------------------------------------------------------
-----
*/

int AbrePorta() {
   DCB dcb;
   BOOL fSuccess;
   char *pcCommPort = "COM4";
   u16 baudRate = 9600;
   hCom = CreateFile( pcCommPort,
                      GENERIC_READ | GENERIC_WRITE,
                      0,    // must be opened with exclusive-access
```

```
                    NULL, // no security attributes
                    OPEN_EXISTING, // must use OPEN_EXISTING
                    0,    // not overlapped I/O
                    NULL);  // hTemplate must be NULL for comm devices

   if (hCom == INVALID_HANDLE_VALUE)
   {
       // Handle the error.
       printf  ("Nao  foi  possivel  conectar  a  leitora  RFID.\nPorta:
%s\nVelocidade: %dbps",pcCommPort,baudRate);
       return(1);
   }
   //Timeouts

    COMMTIMEOUTS  timeOuts;

    timeOuts.ReadIntervalTimeout          = 0xFFFFFFFF;
    timeOuts.ReadTotalTimeoutConstant     = 0;
    timeOuts.ReadTotalTimeoutMultiplier   = 0;
    timeOuts.WriteTotalTimeoutConstant    = 5000;
    timeOuts.WriteTotalTimeoutMultiplier  = 0;

    SetCommTimeouts(hCom, &timeOuts);


   // Build on the current configuration, and skip setting the size
   // of the input and output buffers with SetupComm.

   fSuccess = GetCommState(hCom, &dcb);

   if (!fSuccess)
   {
      // Handle the error.
      printf ("GetCommState failed with error %d.\n", GetLastError());
      return(2);
   }

   // Fill in DCB: 57,600 bps, 8 data bits, no parity, and 1 stop bit.

    dcb.DCBlength = sizeof(DCB);
    GetCommState(hCom, &dcb);
    dcb.BaudRate  = baudRate;
    dcb.ByteSize  = 8;
    dcb.fRtsControl = RTS_CONTROL_DISABLE;

   fSuccess = SetCommState(hCom, &dcb);

   if (!fSuccess)
   {
      // Handle the error.
      printf ("SetCommState failed with error %d.\n", GetLastError());
      return(3);
   }

   printf ("Leitora configurada na porta %s com sucesso.\n", pcCommPort);
   return (0);
}

void FechaPorta(void) {
     CloseHandle(hCom);
}
```

```
signed long int LerDados(void *buffer, int MaxBytes) {
       BOOL    readStatus;
       DWORD   bytesRead;
       DWORD   errorFlags;
       COMSTAT comStat;

       ClearCommError(hCom, &errorFlags, &comStat);
       if(!comStat.cbInQue)
       {
        // No data to read!
        return(0);
       }
       if(comStat.cbInQue < MaxBytes)
       {
                         bytesRead = comStat.cbInQue;
       }
       else
       {
                         bytesRead = MaxBytes;
       }
       readStatus = ReadFile(hCom, buffer, bytesRead, &bytesRead, NULL);
       return(bytesRead);
}

void enviaMsg(int numArgs, char *args[]) {
  char  buf[256];
  u16 crcReg;
  int i,idx,dataToSend;
  idx = 0;
  buf[idx++] = (char)0xFF;
  // Interpret the bit stream as hex
  for(i=0; i<numArgs; i++)
  {
    if(strlen(args[i]) != 2)
    {
      printf("\nDados incorretos %s", args[i]);
//      return(false);
    }

    if(Ascii2Byte(args[i], &dataToSend) == false)
    {
      printf("\nDados incorretos: %s", args[i]);
//      return(false);
    }
    buf[idx++] = dataToSend;
  }

  //Acrescenta o CRC ao final do pacote

    crcReg = MSG_CRC_INIT;
    for(i=1; i<idx; i++)
    {
      CRC_calcCrc8(&crcReg, MSG_CCITT_CRC_POLY, buf[i]);
    }

    buf[idx++] = (crcReg >> 8) & 0xFF;
    buf[idx++] = crcReg & 0xFF;

  // Send buffer
  SendData(buf, idx);
```

```
  // Read the serial port to get data
//    struct MsgObj  rxMsg;

    MSG_receiveMsgObj(&rxMsg);

//  return(true);
}



int Ascii2Nibble(char asciiData)
{
  signed int binData = -1;
  if((asciiData >= '0') && (asciiData <= '9'))
  {
    binData = asciiData - '0';
  }
  else if((asciiData >= 'a') && (asciiData <= 'f'))
  {
    binData = asciiData - 'a' + 0xA;
  }
  else if((asciiData >= 'A') && (asciiData <= 'F'))
  {
    binData = asciiData - 'A' + 0xA;
  }
  return(binData);
}

int Ascii2Byte(char *asciiData, unsigned int *byteOut)
{
  signed int nibble;

  // Convert first nibble
  {
    nibble = Ascii2Nibble(asciiData[0]);

    // Check to make sure nibble is valid
    if(nibble == -1)
    {
       return(false);
    }

    // Put nibble at high position
    *byteOut = (unsigned int)(nibble & 0xF) << 4;
  }
  // Convert second nibble
  {
    nibble = Ascii2Nibble(asciiData[1]);
    // Check to make sure nibble is valid
    if(nibble == -1)
    {
      return(false);
    }
    // Place nibble at lower position
    *byteOut |= (unsigned int)(nibble & 0xF);
  }
  return(true);
}
```

```
int SendData(const char *sendBuf, int numBytesToSend)
{
  DWORD numBytesWritten;
//  printf("Buffer to send: %s with %dbytes",sendBuf,numBytesToSend);
  WriteFile(hCom, sendBuf, numBytesToSend, &numBytesWritten, NULL);
  return(numBytesWritten);
}

signed __int32 ReadData(void *readBuf, unsigned __int32 maxNumBytesToRead)
{
  BOOL    readStatus;
  DWORD   bytesRead;
  DWORD   errorFlags;
  COMSTAT comStat;

  ClearCommError(hCom, &errorFlags, &comStat);
  if(!comStat.cbInQue)
  {
    // No data to read!
    return(0);
  }

  if(comStat.cbInQue < maxNumBytesToRead)
  {
    bytesRead = comStat.cbInQue;
  }
  else
  {
    bytesRead = maxNumBytesToRead;
  }

  readStatus = ReadFile(hCom, readBuf, bytesRead, &bytesRead, NULL);

  return(bytesRead);
}

void MSG_receiveMsgObj(struct MsgObj *hMsg)
{
  signed __int32 bytesRead;
  unsigned __int8  soh;
  unsigned __int8  i;
  unsigned __int8  crc[2];
  int retVal;

  while(1)
  {
    bytesRead = ReadData(&soh, 1);
    soh &= 0xFF;

    if((bytesRead == 1) && (soh == 0xFF))
    {
      break;
    }
  }

  hMsg->crc = MSG_CRC_INIT;

  while(ReadData(&hMsg->dataLen, 1) == 0);
  while(ReadData(&hMsg->OpCode, 1) == 0);
  while(ReadData(&hMsg->status[0], 1) == 0);
  while(ReadData(&hMsg->status[1], 1) == 0);
```

```
  for(i=0; i<hMsg->dataLen; i++)
  {
    while(ReadData(&hMsg->data[i], 1) == 0);
  }

  while(ReadData(&crc[0], 1) == 0);
  while(ReadData(&crc[1], 1) == 0);
  hMsg->crc = ((crc[0] & 0xFF) << 8) | (crc[1] & 0xFF);

}


//Funcao para calcular o CRC do pacote.
void CRC_calcCrc8(u16 *crcReg, u16 poly, u16 u8Data)
{
      u16 i;
      u16 xorFlag;
      u16 bit;
      u16 dcdBitMask = 0x80;

      for(i=0; i<8; i++)
      {
            // Get the carry bit.  This determines if the polynomial should
be xor'd
            //    with the CRC register.
            xorFlag = *crcReg & 0x8000;

            // Shift the bits over by one.
            *crcReg <<= 1;

            // Shift in the next bit in the data byte
            bit = ((u8Data & dcdBitMask) == dcdBitMask);
            *crcReg |= bit;

            // XOR the polynomial
            if(xorFlag)
            {
                  *crcReg = *crcReg ^ poly;
            }

            // Shift over the dcd mask
            dcdBitMask >>= 1;
      }
}

void Configura_RFID(void) {
    Sleep(50);
    enviaMsg(02,boot);
    Sleep(50);
    enviaMsg(3,SetaRegiao);
    Sleep(50);
    enviaMsg(4,ProtocoloGEN2);
    Sleep(50);
    enviaMsg(4,PotAntLeitura);
    Sleep(50);
    enviaMsg(4,Antena);
    Sleep(50);
}

u32 StartBuscaTag() {
```

```
    struct MsgObj *resposta;
    resposta = &rxMsg;
    ponteiroTagLida = TagLida;
    int contador_tag;
    char *LerTag[] = { "02", "21", "01", "F4" };
    enviaMsg(4,LerTag);
    Sleep(50);
    if((resposta->status[0] == 04 && resposta->status[1] == 00)) {
        return(1);
    }
    else if ((resposta->status[0] == 00 && resposta->status[1] == 00)) {
         for(contador_tag=0;contador_tag<12; contador_tag++) {
              ponteiroTagLida
+=sprintf(ponteiroTagLida,"%02X",(u8)resposta->data[contador_tag]);
         }
         return(0);
    }
    else {
         return(2);
         }
}
```

APÊNDICE 1 – PROGRAMA PRINCIPAL

```
#include <stdio.h>
#include <conio.h>
#include <windows.h>
#include <time.h>
#include "parallel.h"
#include "lcd.h"
#include "HostTypes.h"
#include <mysql.h>
#define true 0
#define false 1
#define MSG_MAX_DATA_LENGTH        250
#define MSG_CRC_INIT                              0xFFFF
#define MSG_CCITT_CRC_POLY         0x1021
#define HOST "192.168.200.1"
#define USER "usr_rfid"
#define PASS "usr_rfid"
#define DB "rfid"


char TagLida[25];
char *ponteiroTagLida;
struct MsgObj {
       u8 dataLen;
       u8 OpCode;
       u8 status[2];
       u8 data[MSG_MAX_DATA_LENGTH];
       u16 crc;
} MsgObj;

struct MsgObj rxMsg; // Estrutura que recebe os dados enviados pelo RFID

#include "rfid_bd.h"

MYSQL conexao;
MYSQL_RES *resp;
MYSQL_ROW resultado;

void Retry(int tentativas);
void CtrlCancela(int angulo);
int consulta_bd();

int main() {
    int lcd_escrito=0;
    int RetByte; // Resultado da leitura da porta paralela - Controla os
sensores
    int state=0; //State 0 = Cancela fechada; State 1 = Cancela aberta
    char query2[1000];
    char dentro[]="1";
    if(inpB(0x379) != 88) {
                  CtrlCancela(2);
    }
    lcd_init();
    lcd_cursor(2); // Seta o cursor do lcd para _ sem piscar
    lcd_goto(1,0);
    lcd_puts("Projeto Final");
    lcd_goto(2,5);
    lcd_puts("Luciana Melo");
    printf("Pressione qualquer tecla para iniciar o sistema");
    getch();
```

```
    lcd_clear();
    lcd_goto(1,0);
    lcd_puts("Inicializando...");
    lcd_goto(2,0);
    lcd_puts("Leitora RFID             ");
    Sleep(1000);
    if(AbrePorta() == 0) {
                   Configura_RFID();
                   lcd_goto(2,15);
                   lcd_puts("OK");
    }
    else {
         lcd_goto(2,15);
         lcd_puts("Falhou");
         printf("\nPressione qualquer tecla para encerrar o programa");
         getch();
         return(1);
    }
    Sleep(1500);
    lcd_goto(2,0);
    lcd_puts("Banco de dados            ");
    mysql_init(&conexao);
    char query[1000];
    int i;
    if(mysql_real_connect(&conexao,HOST,USER,PASS,DB,0,NULL,0)) {
         lcd_goto(2,17);
         lcd_puts("OK");
    }
    else {
         printf("\nErro na conexao com o banco de dados");
         if(mysql_errno(&conexao)) {
              printf("\nErro      %d:      %s",      mysql_errno(&conexao),
mysql_error(&conexao));
              lcd_goto(2,17);
              lcd_puts("Falhou");
              printf("\nPressione   qualquer   tecla   para   encerrar   o
programa");
              getch();
              return(1);
         }
    }
    Sleep(1500);
    lcd_goto(2,0);
    lcd_puts("Teste Mecanico #1:         ");
    Sleep(1500);
    CtrlCancela(14);
    Sleep(100);
    RetByte = inpB(0x379);
    if(RetByte == 72 || RetByte == 8 || RetByte == 0 || RetByte == 64) {
               lcd_goto(2,18);
               lcd_puts("Falhou");
               printf("\nCancela nao abriu totalmente ou Sensor Optico fora
de posicao.\nVerifique antes de continuar\n\n");
               system("pause");
    }
    else {
         lcd_goto(2,18);
         lcd_puts("OK     ");
    }
    Sleep(1500);
    lcd_goto(2,0);
```

```
    lcd_puts("Teste Mecanico #2:         ");
    CtrlCancela(2);
    RetByte = inpB(0x379);
    if(RetByte == 72 || RetByte == 8 || RetByte == 0 || RetByte == 64) {
              lcd_goto(2,18);
              lcd_puts("Falhou");
              printf("\nCancela parcialmente aberta. Verifique antes de
continuar\n\n");
              system("pause");
    }
    else {
         lcd_goto(2,18);
         lcd_puts("OK      ");
    }
    Sleep(2000);
    printf("\n\n\n\n");
    printf("Sistema de RFID em seguranca de predios");
    while(1) {
         TagLida[0] = '\0';
         RetByte = inpB(0x379);
         Sleep(250);
       //  printf("Porta:\t%d\n", RetByte);
         if(state == 0 && RetByte == 24 ) {
                  Retry(1);
                  if(TagLida[0] != '\0' ) {
                                 if(consulta_bd() == 0) {
                                      mysql_init(&conexao);

if(mysql_real_connect(&conexao,HOST,USER,PASS,DB,0,NULL,0)) {
                                      sprintf(query2,"SELECT    `estado`
FROM `tb_cadastro` where `tag`='%s';",TagLida);
                                      mysql_query(&conexao,query2);

resp=mysql_store_result(&conexao);
                                      resultado=mysql_fetch_row(resp);
                                      if(strcmp(dentro,resultado[0])  ==
0) {
                                           mysql_close(&conexao);
                                           lcd_clear();
                                           lcd_goto(1,0);
                                           lcd_puts("Entrada        nao
autorizada");
                                           lcd_goto(2,0);
                                           lcd_puts("Entrada
duplicada");
                                           printf("\nAVISO:   Tentativa
de entrada duplicada\n");
                                           Sleep(1000);
                                           lcd_escrito=0;
                                      }
                                      else {
                                           sprintf(query2,"INSERT   INTO
`tb_controle`               (`tag`,`entrada`,`saida`)               VALUES
('%s',now(),saida);",TagLida);

mysql_query(&conexao,query2);
                                           lcd_clear();
                                           lcd_goto(1,0);
                                           lcd_puts("UniCEUB");
                                           lcd_goto(2,0);
```

```
                                                lcd_puts("Bem    vindo    ao
predio");
                                                state=1;
                                                CtrlCancela(14);
                                                lcd_escrito=0;
                                                sprintf(query2,"UPDATE
tb_cadastro SET estado='1' WHERE tag='%s';",TagLida);

mysql_query(&conexao,query2);
                                                mysql_close(&conexao);
                                           }
                                      }
                                 }
                                 else if(consulta_bd() == 1) {
                                      lcd_clear();
                                      lcd_goto(1,0);
                                      lcd_puts("Nao cadastrado. Entre em");
                                      lcd_goto(2,0);
                                      lcd_puts("contato com Adminstrador");
                                      state=0;
                                      Sleep(2000);
                                      lcd_escrito=0;
                                 }
                  }
                  else {
                        lcd_clear();
                        lcd_goto(1,0);
                        lcd_puts("Nao autorizado.Dirija-se");
                        lcd_goto(2,0);
                        lcd_puts("a passagem de visitantes");
                        Sleep(2000);
                        lcd_escrito=0;
                  }
         }
         else if(state == 1 && RetByte == 104 ) {
                  CtrlCancela(4);
                  state=0;
                  lcd_escrito=0;
                  Sleep(100);
         }
         else if (RetByte == 72 || RetByte == 8 || RetByte == 0 || RetByte
== 64){
                   lcd_clear();
                   lcd_goto(1,0);
                   lcd_puts("Falha Mecanica.");
                   lcd_goto(2,0);
                   lcd_puts("Aguarde a manutencao");
                   system("cls");
                   int alarme=1;
                   while(alarme) {
                                 if(kbhit()) {
                                             alarme=0;
                                 }
                                 else {
                                  printf("\a\a\nHouve uma falha mecanica no
sistema");
                                  printf("\nVerique   o   posicionamento   da
cancela");
                                  printf("\n\n\nPressione alguma tecla para
parar o alarme");
                                  Sleep(1000);
```

```
                                system("cls");

                                }
                 }
                 printf("\nSISTEMA EM MANUTENCAO\n\n");
                 system("pause");
                 lcd_escrito=0;
        }
        //lcd_escrito = 0;
        else if ((RetByte == 88) && (lcd_escrito == 0)) {
                 lcd_clear();
                 lcd_goto(1,0);
                 lcd_puts("UniCEUB - Portaria");
                 lcd_escrito=1;
        }
        else if(state == 0 && RetByte == 80 ) {
                 Retry(1);
                 if(TagLida[0] != '\0' ) {
                              if(consulta_bd() == 0) {
                                   mysql_init(&conexao);

if(mysql_real_connect(&conexao,HOST,USER,PASS,DB,0,NULL,0)) {
                                                  sprintf(query2,"INSERT   INTO
`tb_controle`                (`tag`,`entrada`,`saida`)                VALUES
('%s',entrada,now());",TagLida);

mysql_query(&conexao,query2);
                                                  lcd_clear();
                                                  lcd_goto(1,0);
                                                  lcd_puts("UniCEUB");
                                                  lcd_goto(2,0);
                                                  lcd_puts("Ate logo");
                                                  state=1;
                                                  CtrlCancela(14);
                                                  lcd_escrito=0;
                                                  sprintf(query2,"UPDATE
tb_cadastro SET estado='0' WHERE tag='%s';",TagLida);

mysql_query(&conexao,query2);
                                                  mysql_close(&conexao);
                                        }
                                   }
                                   else if(consulta_bd() == 1) {
                                        lcd_clear();
                                        lcd_goto(1,0);
                                        lcd_puts("Nao cadastrado. Entre em");
                                        lcd_goto(2,0);
                                        lcd_puts("contato com Adminstrador");
                                        state=0;
                                        Sleep(2000);
                                        lcd_escrito=0;
                                   }
                 }
                 else {
                       lcd_clear();
                       lcd_goto(1,0);
                       lcd_puts("Nao autorizado.Dirija-se");
                       lcd_goto(2,0);
                       lcd_puts("a passagem de visitantes");
                       Sleep(2000);
                       lcd_escrito=0;
```

```
                    }
          }
          }
     return 0;
}


int consulta_bd() {
     mysql_init(&conexao);
     char query[1000];
     int i;
     if(mysql_real_connect(&conexao,HOST,USER,PASS,DB,0,NULL,0)) {
          sprintf(query,"SELECT COUNT(tag) FROM `tb_cadastro`;");
          mysql_query(&conexao,query);
          resp=mysql_store_result(&conexao);
          resultado=mysql_fetch_row(resp);
          sprintf(query,"SELECT `tag` FROM `tb_cadastro`;");
          mysql_query(&conexao,query);
          resp=mysql_store_result(&conexao);
          while ((resultado=mysql_fetch_row(resp)) != NULL)
          {
                for(i=0;i<mysql_num_fields(resp);i++)
                {
                     if(strcmp(TagLida,resultado[0]) == 0) {
                          mysql_close(&conexao);
                          return(0);
                     }
                }

          }
     mysql_close(&conexao);
     return(1);
     }
     else {
          mysql_close(&conexao);
          return(-1);
     }
}


void Retry(int tentativas) {
     while ((StartBuscaTag() != 0 && tentativas != 0)) {
          tentativas--;
     }
}

void CtrlCancela(int angulo) {
     int i,j;
     for (i=0; i<5;i++) {
               outB(0x37A,0x0F);
          for (j=0;j<angulo;j++) {
              outB(0x80,0);
          }
          outB(0x37A,0x0);
          Sleep(80);
    }
}
```

APÊNDICE 2 – CADASTRO DE ETIQUETAS.

```
#include <stdio.h>
#include <conio.h>
#include <windows.h>
#include <stdlib.h>
#include <time.h>
#include "HostTypes.h"
#include <mysql.h>
/*
         Definições globais
*/
#define true 0
#define false 1
#define MSG_MAX_DATA_LENGTH        250
#define MSG_CRC_INIT                              0xFFFF
#define MSG_CCITT_CRC_POLY         0x1021
/*
        Dados de conexão com o banco de dados
*/
#define HOST "192.168.200.1"
#define USER "usr_rfid"
#define PASS "usr_rfid"
#define DB "rfid"


char TagLida[25];
char *ponteiroTagLida;
struct MsgObj {
       u8 dataLen;
       u8 OpCode;
       u8 status[2];
       u8 data[MSG_MAX_DATA_LENGTH];
       u16 crc;
} MsgObj;

struct MsgObj rxMsg; // Estrutura que recebe os dados enviados pelo RFID

#include "rfid_bd.h"

MYSQL conexao;
MYSQL_RES *resp;
MYSQL_ROW resultado;
MYSQL_FIELD *campos;

u32 cadastrar_tag(char *nome, char *placa);
u32 descadastrar_tag();
void Consulta_Registros();
void Consulta_Controle();

int main() {
             char opcao,confirmacao;
             char nome[60];
             char placa_carro[8];
             AbrePorta();
             Configura_RFID();
             int tentativas = 3;
             while(1) {
             system("cls");
             printf("################################################");
```

```
            printf("\n#\tSistema de controle de etiquetas       #");
            printf("\n################################################");
            printf("\n\n\n");
            if(TagLida[0] != '\0') {
                        printf("Tag encontrada: %s\n\n",TagLida);
            }
            printf("Digite uma opcao: \n\t(1) Ler TAG\n\t(2) Cadastrar
etiqueta.\n\t(3)    Descadastrar    etiqueta.\n\t(4)    Exibir    etiquetas
cadastradas.\n\t(5) Exibir registros de entradas e saidas.\n\t(6) Sair do
programa.\n\n\tOpcao: ");
            scanf("%c",&opcao);
            switch (opcao) {
                  case '1':
                       {
                            system("cls");
                            TagLida[0] = '\0';
                            printf("\n\tPor favor aproxime a etiqueta da
leitora e pressione enter.");
                            getch();
                            printf("\n\n\tPor favor, aguarde...");
                            while((StartBuscaTag() != 0 && tentativas !=
0)) {
                                                    tentativas--;
                            }
                            if(TagLida[0] != '\0') {
                                                printf("\nTag    encontrada:
%s",TagLida);
                            }
                            else {
                                                system("cls");
                                                printf("\n\tNenhuma      Tag
encontrada");
                                                printf("\n\nPressione
qualquer tecla para continuar");
                                                getch();
                                                TagLida[0] = '\0';
                            }
                  } break;
                  case '2' :
                       {
                           system("cls");
                           if(TagLida[0] == '\0') {
                                                printf("\n\tNenhuma    Tag    foi
lida. Escolha a opcao (1) no menu principal");
                                                printf("\n\n\tPressione  qualquer
tecla para continuar.");
                                                getch();
                                                break;
                           }
                           else {
                                fflush(stdin);
                                printf("\n\tNome do utilizador: ");
                                fgets(nome,60,stdin);
                                printf("\n\tPlaca do carro: ");
                                fgets(placa_carro,8,stdin);
                                if(cadastrar_tag(nome, placa_carro) == 0)
{
                                                        printf("\n\tDados
gravados com sucesso.");

printf("\n\n\tPressione qualquer tecla para continuar.");
```

```
                                                  getch();
                                        }
                                        else {
                                             printf("\n\tErro      durante      o
cadastro");
                                             if(mysql_errno(&conexao)) {

printf("\n\tErro %d: %s", mysql_errno(&conexao), mysql_error(&conexao));
                                             }
                                        }
                                   }
                         } break;
                         case '3' :
                              {
                                   system("cls");
                                   if(TagLida[0] == '\0') {
                                             printf("\n\tNenhuma    Tag    foi
lida. Escolha a opcao (1) no menu principal");
                                             printf("\n\n\tPressione  qualquer
tecla para continuar.");
                                             getch();
                                             break;
                                   }
                                   else {
                                        printf("\n\tOs dados associados a Tag %s
serao apagados. Deseja continuar ? (y/n) ",TagLida);
                                        fflush(stdin);
                                        scanf("%c",&confirmacao);
                                        if(confirmacao = 'y') {
                                                  descadastrar_tag();
                                        }
                                   }
                         } break;
                         case '4' :
                              {
                                   system("cls");
                                   Consulta_Registros();
                                   printf("\n\n\tPressione  qualquer  tecla  para
continuar.");
                                   getch();

                         } break;
                         case '5' :
                              {
                                   system("cls");
                                   Consulta_Controle();
                                   printf("\n\n\tPressione  qualquer  tecla  para
continuar.");
                                   getch();
                         } break;
                         case '6' :
                              {
                              FechaPorta();
                              exit(0);
                         } break;
                }
    }
       getch();
    return(0);
}
```

```
u32 cadastrar_tag(char *nome, char *placa) {
     mysql_init(&conexao);
     char query[1000];
     if(mysql_real_connect(&conexao,HOST,USER,PASS,DB,0,NULL,0)) {
          sprintf(query,"INSERT    INTO    `tb_cadastro`    (`tag`,`nome`,
`placa_carro`) VALUES ('%s', '%s', '%s');",TagLida,nome,placa);
          if(mysql_query(&conexao,query)) {
               printf("\n\tErro: %s",mysql_error(&conexao));
          }
          mysql_close(&conexao);
          return(0);
     }
     else {
          mysql_close(&conexao);
          return(2);
     }
     mysql_close(&conexao);
}


u32 descadastrar_tag() {
     mysql_init(&conexao);
     char query[1000];
     if(mysql_real_connect(&conexao,HOST,USER,PASS,DB,0,NULL,0)) {
          sprintf(query,"DELETE       FROM       `tb_cadastro`       WHERE
`tag`='%s';",TagLida);
          if(mysql_query(&conexao,query)) {
               printf("\n\tErro: %s",mysql_error(&conexao));
          }
          mysql_close(&conexao);
          return(0);
     }
     else {
          mysql_close(&conexao);
          return(2);
     }
     mysql_close(&conexao);
}

void Consulta_Registros() {
    mysql_init(&conexao);
    int i,j,conta;

    char query[1000];
    if(mysql_real_connect(&conexao,HOST,USER,PASS,DB,0,NULL,0)) {
          sprintf(query,"SELECT tag,placa_carro,nome FROM `tb_cadastro`;");
          mysql_query(&conexao,query);
          resp=mysql_store_result(&conexao);
          if (resp)
          {
           campos = mysql_fetch_fields(resp);
           for (conta=0;conta<mysql_num_fields(resp);conta++) {
              printf("%s",(campos[conta]).name);
              if (mysql_num_fields(resp)>1)
                  printf("\t\t\t");
              }
           }
           printf("\n\n");
           while ((resultado=mysql_fetch_row(resp)) != NULL)
              {
```

```
                for (conta=0;conta<mysql_num_fields(resp);conta++)
                printf("%s     ",resultado[conta]);
                printf("\n");
             }
          mysql_free_result(resp);
    }
    else {
             printf("\nFalha na conexão com o banco de dados");
             if(mysql_errno(&conexao)) {
                  printf("\nErro       %d.       %s",mysql_errno(&conexao),
mysql_error(&conexao));
             }
    }
}

void Consulta_Controle() {
    mysql_init(&conexao);
    int i,j,conta;

    char query[1000];
    if(mysql_real_connect(&conexao,HOST,USER,PASS,DB,0,NULL,0)) {
          sprintf(query,"SELECT           nome,entrada,saida           from
tb_cadastro,tb_controle where tb_cadastro.tag=tb_controle.tag;");
          mysql_query(&conexao,query);
          resp=mysql_store_result(&conexao);
          if (resp)
          {
           campos = mysql_fetch_fields(resp);
           for (conta=0;conta<mysql_num_fields(resp);conta++) {
              printf("%s",(campos[conta]).name);
              if (mysql_num_fields(resp)>1)
                  printf("\t\t\t");
              }
           }
           printf("\n\n");
           while ((resultado=mysql_fetch_row(resp)) != NULL)
              {
                 for (conta=0;conta<mysql_num_fields(resp);conta++)
                 printf("%s     ",resultado[conta]);
                 printf("\n");
              }
          mysql_free_result(resp);
    }
    else {
             printf("\nFalha na conexão com o banco de dados");
             if(mysql_errno(&conexao)) {
                  printf("\nErro       %d.       %s",mysql_errno(&conexao),
mysql_error(&conexao));
             }
    }
}
```

APÊNDICE 3 – CÓDIGO-FONTE DA CRIAÇÃO DO BANCO DE DADOS.

```
CREATE DATABASE  IF NOT EXISTS `rfid` DEFAULT CHARACTER SET latin1;
USE `rfid`;
--
-- Estrutura da tabela `tb_cadastro`
--

CREATE TABLE `tb_cadastro` (
  `tag` varchar(25) NOT NULL,
  `nome` varchar(60) DEFAULT NULL,
  `placa_carro` varchar(8) NOT NULL,
  `estado` tinyint(1) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Estrutura da tabela `tb_controle`
--
CREATE TABLE `tb_controle` (
  `tag` varchar(24) DEFAULT NULL,
  `entrada`  timestamp  NOT  NULL  DEFAULT  CURRENT_TIMESTAMP  ON  UPDATE
CURRENT_TIMESTAMP,
  `saida` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00'
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```