Centro Universitário de Brasília – UniCEUB
Faculdade de Ciências Exatas e de Tecnologia – FAET
Curso de Engenharia de Computação
Disciplina: Projeto Final
Professor Orientador: Profº. José Julimá Bezerra Júnior

Movimento de um Móvel de Acordo com a Orientação Recebida nos Parâmetros de um Ângulo Geométrico em Sistema de Malha Aberta

ITALO BRUNO FREITAS DE JESUS
REGISTRO ACADÊMICO: 2011492/2

Brasília-DF, Dezembro de 2006

CENTRO UNIVERSITÁRIO DE BRASÍLIA – UniCEUB
Faculdade de Ciências Exatas – FAET
Engenharia de Computação

Movimento de um Móvel de Acordo com a Orientação Recebida nos Parâmetros de um Ângulo Geométrico em Sistema de Malha Aberta

ITALO BRUNO FREITAS DE JESUS
REGISTRO ACADÊMICO: 2011492/2

Monografia apresentada à Banca Examinadora da Faculdade de Ciências Exatas e Tecnologia - UniCEUB, para conclusão do curso de Engenharia de Computação.

Brasília-DF, Dezembro de 2006

"Se pude enxergar além de outros, é somente porque estou apoiado no ombro de gigantes"

Isaac Newton

Dedico este projeto à minha preciosa família, meus pais Paulo e Solange e meu irmão Leandro que durante todo este período tiveram muita compreensão e me apoiaram em tudo que passei, especialmente por me proporcionarem chegar até aqui, pessoas com quem contarei por toda minha vida. Dedico especialmente à minha querida mãe, que abriu mão de certa medida de conforto pessoal para me ajudar na conquista desta etapa.

Dedico ainda à minha querida Eveline. Agradeço a todo o suporte, motivação e por ter aceitado se privar da minha companhia pelos estudos e trabalho, permitindo a mim a oportunidade de concluir mais esta árdua fase da minha vida.

Agradecimentos

Agradeço primeiramente a Jeová, o Deus onipotente, que me permitiu chegar até este momento. Agradeço ainda aos professores que me apoiaram no decorrer deste curso, dedicando seu tempo e conhecimento na arte do ensino, dentre eles um agradecimento especial ao Professor Julimá, como orientador deste projeto e Professor Waldir na correção gramatical.

Agradeço ainda, aos meus caros amigos da faculdade, Fernando do Prado, Eduardo Braga, Saulo Cirineu, Gustavo Gomes.

Expresso meu agradecimento também aos meus colegas de trabalho que foram compreensíveis durante este último semestre da minha formação universitária. Especialmente ao Carlos Magno, companheiro que me sugeriu soluções durante a produção técnica e também ao Tomé, por me substituir em algumas atividades importantes.

E a todos que contribuíram direta ou indiretamente para este projeto, meu mais sincero voto de gratidão.

Resumo

Ambientes controlados e automatizados estão a cada dia em evolução. Todo sistema controlado faz este controle por meio de uma variável chave do ambiente, chamada variável controlada. Este projeto objetiva efetuar este controle a partir do ângulo geométrico de um movimento. Esta monografia foi produzida como projeto de conclusão do curso de Engenharia de Computação. É sobre o estudo do controle de um pequeno móvel de acordo com a orientação recebida nos parâmetros de um ângulo geométrico em um sistema de controle em malha aberta utilizando microcontrolador 8051. Objetiva apresentar resultados satisfatórios para a pesquisa e o desenvolvimento de um ambiente real para validação da proposta.

Palavras chave: Controle, microcontrolador, programação C, ângulo geométrico.

Abstract

Automated and controlled ambient is in evolution nowadays. Every controlled system controls by using a key variable from the ambient, called controlled variable. This project aims to do this control, from the geometric angle of a movement. This monograph was produced as a Computer Engineering conclusion project. It regards the study of controlling a small vehicle according to the orientation received in the shape of a geometric angle in a system of a open-loop controller or a non-feedback controller, using 8051 microcontroller. The aims are to present satisfactory results according to the research and the development of a real ambient pretending to validate the proposal.

Key Words: Control, microcontroller, C programming, geometric angle.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Fluxogramas

Lista de Abreviaturas

ASCII – American Standard Code for Information Interchange

ASIC – Application Specific Integrated Circuit

dB – Conector Serial de 9 pinos

ESC – Escape

E/S – Entrada e Saída

FIFO – First in First out

GND – Aterramento

LED – Light Emitting Diode

kg – Kilograma

m – Metro

m^2 – Metro Quadrado

mA – miliAmpère, medição de corrente

P41 – Porta lógica do microcontrolador

P42 – Porta lógica do microcontrolador

P43 – Porta lógica do microcontrolador

Rx – receptor

Tx – transmissor

UART – Transmissor/Receptor Assíncrono Universal

USB – Universal Serial Bus

V – Volts

Ω – ohms, medida de resistência

Capítulo 1 – Introdução

Atualmente temos uma crescente procura do homem pela automação, processo que objetiva ter um sistema automático de controle em que os mecanismos verificam seu próprio funcionamento, efetuando medições e introduzindo correções, sem a necessidade da interferência do homem.

Um exemplo disso foi a espaçonave Mars Pathfinder (Desbravador de Marte) que em dezembro de 1996, foi enviada ao espaço a bordo de um veículo lançador descartável Delta II para iniciar uma jornada de sete meses no planeta vermelho. A missão Pathfinder foi a primeira missão a pousar em Marte depois do sucesso da espaçonave Viking há mais de duas décadas. A Parhfinder desembarcou primeiro veículo explorador autônomo, conhecido como Sojouner, possui uma massa de 10,5 kg e foi projetado para percorrer uma área de 300 m^2 durante cerca de 30 dias. O controle de manobra deste veículo tem de ser preciso e deve limitar o consumo de energia. [OGATA, 2003]

Como citado no exemplo acima, sistemas que envolvam controle estão a cada dia avançando, proporcionando chegar aonde o homem, com suas próprias mãos, não chegou e talvez nem chegue, devido a limitações físicas e perigos à saúde. Tal exemplo retrata bem a intenção deste projeto, iniciar na pesquisa, nos estudos e, sucessivamente, já apresentar resultados positivos no campo de controle.

Objetivando alcançar soluções práticas a Engenharia de Computação é uma área especializada que combina a engenharia elétrica e a ciência da computação. Um engenheiro da computação é um engenheiro eletrônico com foco em lógica de sistemas digitais sendo, portanto, o arquiteto do software com que faz a interação

com um hardware, esta produção tem este foco, visando integrar um hardware ao controle de um software.

Desse modo, o Engenheiro de Computação tem como bom campo de atuação a área de controle integrando uma solução eletrônica gerenciada por um software, visando resolver um problema real de qualquer natureza.

1.1 – MOTIVAÇÃO

Desse modo, é possível concluir que qualquer projeto de Engenharia surge a partir da observância de um problema ou de uma possível melhora em determinado ambiente.

Este projeto que apresento é proveniente da intenção em proporcionar uma maneira automática para a movimentação de um móvel[1] a partir de um parâmetro numérico representando um ângulo geométrico real em um determinado ambiente.

Esta proposta visa dar subsídio a projetos mais extensos, como o movimento de cadeira de rodas automática, realizando acertos em seu movimento, a partir de um ângulo geométrico extraído do ambiente em que a cadeira está inserida. Movimento de braço mecânico ou outros possíveis. Desse modo este projeto, tendo cunho acadêmico, intenciona proporcionar uma solução para controle de dispositivos móveis na sua variável, ângulo geométrico.

1.2 – OBJETIVOS

Este projeto visa, em sua essência, proporcionar subsídios, com visão acadêmica, para solução de problemas relativos a movimentos automáticos,

[1] carro, cadeira de rodas, etc.

provendo uma possível opção para tais movimentos que possam ser controlados a partir de um parâmetro em formato de ângulo geométrico.

Visa ainda simular um ambiente real que possa ser controlado, a partir de parâmetro em formato de ângulo geométrico e sem realimentação, objetivando validar a solução deste projeto.

A Figura 1, retrata a idéia proposta.

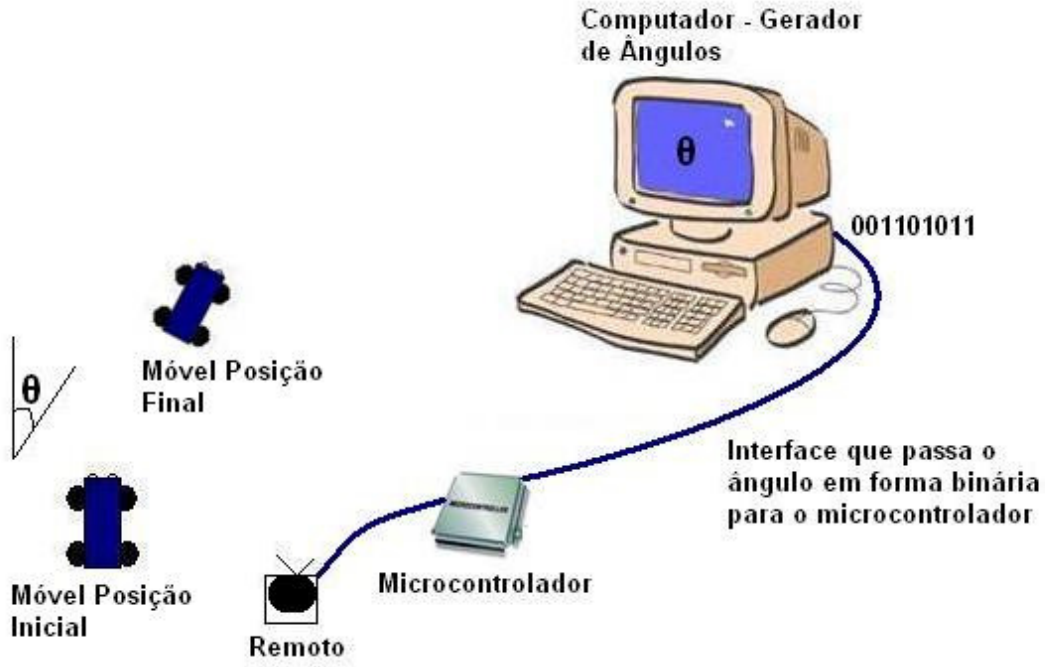


Figura 1.1 – Objetivo do Projeto

1.2.1 – Objetivos Específicos

Este projeto tem por objetivos específicos a produção de:

- Uma Interface Computacional que tem por principal objetivo interagir com o usuário do ambiente, essa interface deverá ser desenvolvida em linguagem de programação computacional C e deverá coletar o

ângulo geométrico bem como o sentido do movimento, tratando alguns possíveis erros do ambiente;

- A programação no microcontrolador 8051, definido na proposta, para efetuar o controle e interagir com o projeto físico, circuito eletrônico e móvel;

- A programação da porta serial, a qual será utilizada para comunicação entre o computador (Interface Computacional) e o microcontrolador 8051;

- A produção de circuito eletrônico necessário para a adequação do controle do carrinho (móvel) a ser controlado;

- Adequação do móvel adquirido ao sistema desenvolvido;

- Desenvolvimento de sistema manual de medição do movimento do carrinho, a fim de verificar o resultado final.

1.3 – ESTRUTURA DA MONOGRAFIA

Esta monografia de conclusão de curso está organizada da seguinte forma:

- Capítulo 1 – Introdução: Apresenta o projeto quanto à motivação, objetivos e metodologia, onde os métodos técnicos e científicos adotados para o desenvolvimento deste projeto. Linha de trabalho para tornar o projeto exeqüível.

- Capítulo 2 – Fundamentação Teórica: Embasamento referente aos temas abordados neste projeto, tais como teoria de

microcontrolador 8051 relativo ao projeto, linguagem de programação C e circuitos eletrônicos.

- Capítulo 3 – Produção Técnica: Este capítulo apresenta o desenvolvimento do projeto em sua inteireza, destacando todos os passos seguidos, utilizando o embasamento teórico do capítulo dois para sua apresentação.

- Capítulo 4 – Resultados: Destaca os resultados obtidos da implementação do projeto, sucessos e insucessos de acordo com a metodologia proposta.

- Capítulo 5 – Considerações Finais: Aborda as conclusões obtidas deste projeto, os problemas encontrados em seu desenvolvimento e sugestões para estudos futuros.

Capítulo 2 – Fundamentação Teórica

2.1 – COMPUTAÇÃO

2.1.1 – Interface Computacional

Entende-se por Interface, no uso comum da palavra, como o ponto, área ou superfície ao longo da qual duas substâncias ou outras coisas qualitativamente diferentes se encontram, ou seja, é um camada situada entre eles, que trabalha como um tradutor, fazendo com que ambos se entendam, a fim de se comunicarem. Segundo Pierre Lévy, é conjunto de programas e aparelhos, materiais que possibilitem a comunicação entre o homem e a máquina, Pierre Lévy refere-se à grande influência que as tecnologias exercem sobre nosso modo de agir e pensar ao afirmar que "diversos trabalhos desenvolvidos em psicologia cognitiva a partir dos anos sessenta mostraram que a dedução ou a indução formais estão longe de serem praticadas espontaneamente e corretamente por sujeitos reduzidos apenas aos recursos de seus sistemas nervosos (sem papel, nem lápis, nem possibilidade de discussão coletiva)" Lévy (LEVY, 1998).

No âmbito da computação uma Interface é um 'contrato' que determina a forma de comunicação entre componentes de software e/ou hardware. Para os usuários comuns de computadores, uma Interface é a forma de apresentação de programas ou sistemas. Com o avanço na capacidade de processamento das máquinas, aconteceu uma mudança profunda na interfaces dos programas e sistemas. A tela (geralmente) preta e baseada em texto, foi trocada por janelas, botões, abas, caixas de texto ou de checagem, ícones, etc. Esta nova apresentação gráfica, também conhecida como "ambiente gráfico", tornou o uso do computador mais amigável.

Entretanto, a forma da inteface (seja gráfica ou baseada em texto) não pode ser responsabilizada pela facilidade ou dificuldade de interação com um sistema ou programa. Uma interface está diretamente ligada a funcionalidade ou comportamento do sistema. Uma mesma tela, com os mesmos componentes e imagens, pode ter um comportamento diferente sob os mesmos "estímulos" do usuário, ou seja um aperto de botão, um arrastar e soltar.

Deste modo uma interface é o mecanismo, automático, que integra dois ambientes distintos, porém que necessitam se comunicar. Como exemplo, temos o usuário e o computador, esta comunicação se dá por meio da tela do computador, onde é possível visualizar as informações inseridas pelo usuário e as informações enviadas pelo computador.

2.1.2 – Porta serial em detalhes (RS-232C)

O padrão RS-232C define diversos parâmetros para a comunicação serial, entre eles os níveis de tensão que representam o 0 e o 1 lógicos, os conectores a serem utilizados (formato e numero de pinos), e o formato dos dados a serem utilizados.

Com relação a portabilidade, a existência do padrão RS-232C simplifica a vida do desenvolvedor de aplicações que utilizam a porta serial. A porta paralela é bem mais simples de se utilizar do ponto de vista do desenvolvedor, porém a diversidade de padrões pode fazer com que uma aplicação não funcione em qualquer máquina.

Uma vez identificada a porta paralela em uso, e o padrão utilizado, o desenvolvedor não precisa se preocupar em implementar nenhum protocolo especial para acionar ou receber informações do dispositivo externo. Todos os dados

trafegados pela porta paralela ficarão armazenados (“bufferizados”) no registrador de dados da porta.

Já em uma porta serial existe um módulo responsável pela gerência da comunicação, e o dispositivo externo deverá “falar” a mesma língua desse módulo de comunicação serial.

No caso dos computadores padrão IBM-PC, o módulo responsável pela gerência da porta paralela é implementado com um circuito integrado baseado no Intel 8250.

O 8250 implementa uma UART (Universal Asynchronous Receiver/Transmitter) que, basicamente, se encarrega de serializar os bytes recebidos (conversão paralelo para serial), adicionar os bits de controle definidos no protocolo RS-232C (start bit, stop bit e paridade), e enviar os bits um a um para o pino de transmissão do conector.

O 8250 realiza também a função inversa, ou seja, receber os bits serialmente, remover os bits de controle, remontar o byte recebido e disponibilizar esse byte no registrador de dados do 8250.

A UART 8250 passou a fazer parte dos computadores padrão IBM-PC em 1981. Porém, para solucionar um problema relativo à velocidade de conversão de paralelo para serial e vice-versa, a partir de 1984 os computadores pessoais passaram a utilizar a UART 16450 que e uma versão melhorada da 8250. Essa nova UART também possui apenas um byte para armazenamento dos dados recebidos, mas o circuito interno e bem mais rápido do que o da 8250, resolvendo o problema de perda de dados para os modems da época.

Em 1987 os PCs passaram a utilizar a UART 16550A com capacidade de armazenamento para mais do que um byte recebido, e posteriormente a UART 166550 que possui uma FIFO[2] com capacidade para armazenar ate 16 bytes recebidos. Essa fila e uma boa solução para armazenar os dados recebidos de modems de alta velocidade em ambientes multitarefa.

As UARTs utilizadas nos PCs atuais são compatíveis com a 166550, porém são encapsuladas em dispositivos do tipo ASIC (Application Specific Integrated Circuit) ficando difícil sua identificação visual na placa do computador.

A tabela 2.1 apresenta a pinagem de conectores de 9 (DB-9) pinos de acordo com o padrão RS-232C.

9-pinos (DB-9)	Símbolo	Função
3	Tx	Transmite dados
2	Rx	Recebe dados
7	RTS	Permissão para envio
8	CTS	Permissão fornecida
6	DSR	Dado pronto
5	GND	Terra
1	CD	Detecção de portadora
4	DTR	Terminal pronto
9	RI	Indicador de Ring

Tabela 2.1 – Pinagem da Porta Serial db-9

No projeto da interface para conexão a um dispositivo a ser controlado via porta serial, bastam 3 pinos (3 fios): pino 5 – Terra, pino 3 – transmissão de dados, pino 2 – recepção de dados.

[2] FIFO – First in First out, primeiro a chegar é o primeiro a sair. É representação de uma fila em ordem de chegada, significa que o primeiro bit a chegar é o primeiro a sair.

Abaixo segue um diagrama de blocos mostrando os registradores da UART 8250.

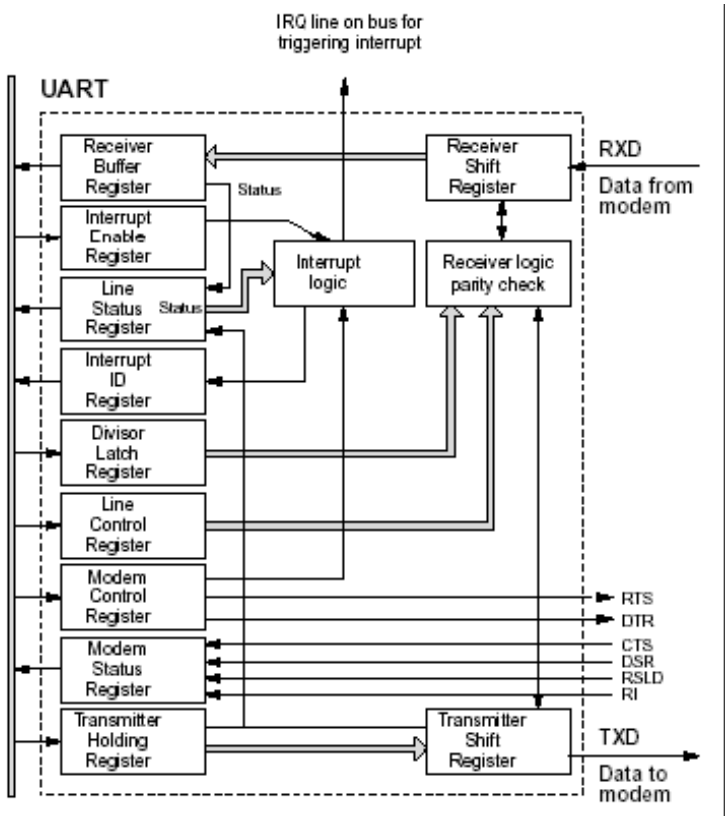


Figura 2.1 – Representação da UART 8250

A seguir são descritos funcionalmente os registradores da UART 8250:

Registrador de dados:

Registrador de Recepção: Ao se realizar uma leitura na 8250, obtêm-se o byte contido nesse registrador proveniente da conversão de serial para paralelo dos bits de entrada (pino Rx).

Registrador de transmissão: Uma operação de escrita carrega um byte nesse registrador para ser convertido de paralelo para serial pela 8250, e posterior transmissão pelo pino Tx).

Status:

Registrador de Status de Linha: utilizado para indicar condições de operação da linha.

7	6	5	4	3	2	1	0
0	TxE	TBE	Break	Erro de framming	Erro de paridade	Erro de overrun	RxRdy

Bit 0: 1 = existe byte pronto para ser lido no registrador de recepção.

Bit 1: 1 = um byte no registrador de recepção foi sobre escrito por um novo byte. 0 primeiro byte foi perdido.

Bit 2: 1 = erro de paridade.

Bit 3: 1 = stop bit invalido

Bit 4: 1 = interface detecta a linha em zero durante um tempo maior que a duração de um byte assíncrono.

Bit 5: Buffer de transmissão vazio. 1 = um byte e movido do buffer de transmissão para o registro de deslocamento, onde o byte e transmitido serialmente.

Bit 6: Transmissor vazio. 1 = registro de deslocamento vazio.

Registrador de Identificação de Interrupção: Após uma interrupção, o bit 0 recebe 0, e os bits 1 e 2 determinam a fonte da interrupção.

Bit 2	Bit 1	Bit 0	Identificador
0	0	1	Sem interrupção
1	1	0	Condição de erro

1	0	0	Byte recebido
0	1	0	Buffer de transmissão vazio
0	0	0	Mudança na entrada da porta serial

Tabela 2.2 – Registrador de Identificação de Interrupção

Controle:

Registrador de Controle de Linha: utilizado para formatação dos dados.

7	6	5	4	3	2	1	0
DLAB	Break	Paridade			Stop bit	Bits de dados	

Bits 0 - 1 : quantidade de bits por caractere:

00 = 5 bits

01 = 6 bits

10 = 7 bits

11 = 8 bits

Bit 2 : quantidade de bits por caractere

0 = 1 stop bit

1 = 2 stop bits

Obs. Se a quantidade de bits por caractere for igual a 5, o numero de stop bits será automaticamente 1 ½ stop bits.

Bits 3 - 5 : determinam a paridade.

000 = sem paridade

001 = paridade impar

011 = paridade par

101 = marca (mark)

111 = espaço (space)

Bit 6 : 1 = saída Tx vai para o nível lógico 0.

Bit 7 : 1 = registro de transmissão recebe o byte de menor órdem (LSB) da taxa de transmissão e o registro de controle de interrupção recebe o byte de maior ordem (MSB).

Registrador de Controle de Interrupção: utilizado para habilitar os quatro tipos de interrupção do 8250.

7	6	5	4	3	2	1	0
0	0	0	0	Pinos de entrada	Erro na recepção	TBE	RxRdy

Bit 0: 1 = uma interrupção é gerada quando um byte estiver disponível no registrador de recepção.

Bit 1: 1 = uma interrupção e gerada quando a 8250 puder receber um novo byte para transmissão.

Bit 2: 1 = uma interrupção e gerada quando ocorrer um erro de paridade, overrun (sobre escrita) ou stop bit.

Bit 3: 1 = uma interrupção e gerada quando qualquer entrada da porta serial mudar de estado.

Por fim, para programar a taxa de transferência (velocidade da comunicação), e preciso utilizar o bit 7 (DLAB) do Registrador de Controle de Linha, em conjunto com o Registradores de Controle de Interrupção e com o Registrador de Dados, da seguinte forma:

1. Colocar o bit DLAB em 1 para indicar que a parte baixa (LSB) da programação velocidade será colocada no Registrador de Dados, e a parte alta (MSB) no Registrador de Controle de Interrupção.

2. Escrever no Registrador de Controle de Interrupção o valor desejado de acordo com a tabela 2.3 (00H[3] para 9600bps, por exemplo).

3. Escrever no Registrador de Dados o valor desejado de acordo com a tabela a seguir (0CH para 9600bps, por exemplo).

Velocidade (bps)	Divisor (Dec)	Divisor MSB Reg. Contr.Interrupção	Divisor (LSB) Reg. Dados
50	2304	09h	00h
300	384	01h	80h
600	192	00h	C0h
2400	48	00h	30h
4800	24	00h	18h
9600	12	00h	0Ch
19200	6	00h	06h
38400	3	00h	03h
57600	2	00h	02h
115200	1	00h	01h

Tabela 2.3 – Velocidade da Comunicação Serial

[3] H ou h, significa que o valor mencionado está em base hexadecimal, o que, em muitos casos, é a representação de um espaço de memória.

Endereços da UART 8250 (porta COM1):

Dados (escrita/leitura)	03F8H
Registrador de Controle de Interrupção	03F9H
Registrador de Identificação de Interrupção	03FAH
Registrador de Controle de linha	03FBH
Registrador de Controle do Modem	03FCH
Registrador de Status da Linha	03FDH
Registrador de Status do Modem	03FEH

Tabela 2.4 – Endereços da UART

O funcionamento da troca de dados assíncrona entre duas UARTs e relativamente simples. No lado do transmissor o pino de envio de dados Tx permanece em nível lógico 1 ate existir um caractere pronto para transmissão. Nesse instante o pino e colocado em zero, representando o start bit. Os bits de dados são enviados logo apos o start bit, um apos o outro. Um bit de paridade, opcionalmente, segue os bits de dados. Após isso um ou mais stop bits são enviados. O bit de paridade é a soma dos bits de dados e indica se o dado contém um número ímpar ou par de bits 1. Para paridade par esse bit será 0. Para paridade impar o bit será 1.

Abaixo segue um exemplo de transmissão do caractere A via RS-232.

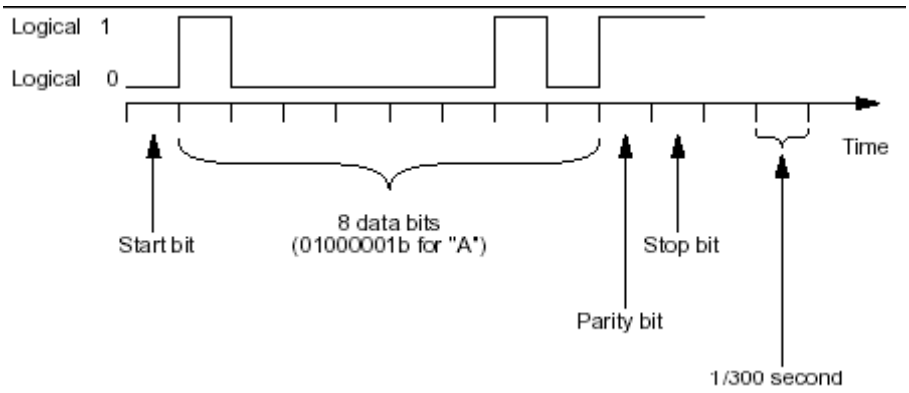
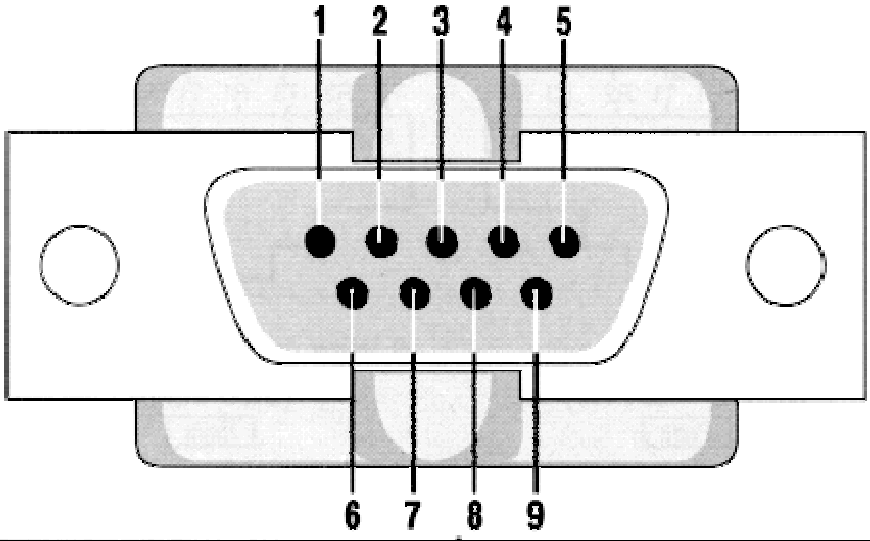


Figura 2.2 – Envio de Caractere Via r2-232



Pin	Signal	Pin	Signal
1	Data Carrier Detect	6	Data Set Ready
2	Received Data	7	Request to Send
3	Transmitted Data	8	Clear to Send
4	Data Terminal Ready	9	Ring Indicator
5	Signal Ground		

Figura 2.3 – Representação Física da rs-232
(Retirado de http://www.arcelect.com/rs232.htm)

2.2 – MICROCONTROLADORES

Um microcontrolador é um computador programável, um elemento eletrônico, em um chip otimizado para controlar dispositivos eletrônicos. É uma espécie de microprocessador, com memória e interfaces de E/S – Entrada e Saída (I/O) integrados, enfatizando a auto-suficiência, em constraste com um microprocessador de propósito geral, o mesmo tipo usado nos PCs, que requer chips adicionais para prover as funções necessárias. Esta integração reduz drasticamente o número de chips e conexões entre outros dispositivos necessários, já que o microcontrolador tem dentro de si uma série de recursos. É desenvolvido para executar tarefas específicas, com linguagem de comando específica.

O microcontrolador faz uso de uma memória própria para a armazenagem do programa que será lido durante a execução das instruções e de uma memória temporária que armazena as informações de uso próprio das instruçõe, enquanto essas informações devem ser armazenadas.

Programa-se a memória com códigos compatíveis com a linguagem do microcontrolador para executar uma tarefa ou grupo de tarefas criadas pelo programador.

Os microcontroladores são componentes utilizados em muitos tipos de equipamentos eletrônicos, sendo a grande maioria entre os chips vendidos. Cerca de 50% são controladores "simples", outros 20% são processadores de sinais digitais (DSPs) mais especializados. Os microcontroladores podem ser encontrados em máquinas de lavar, forno de microondas, telefones, etc.

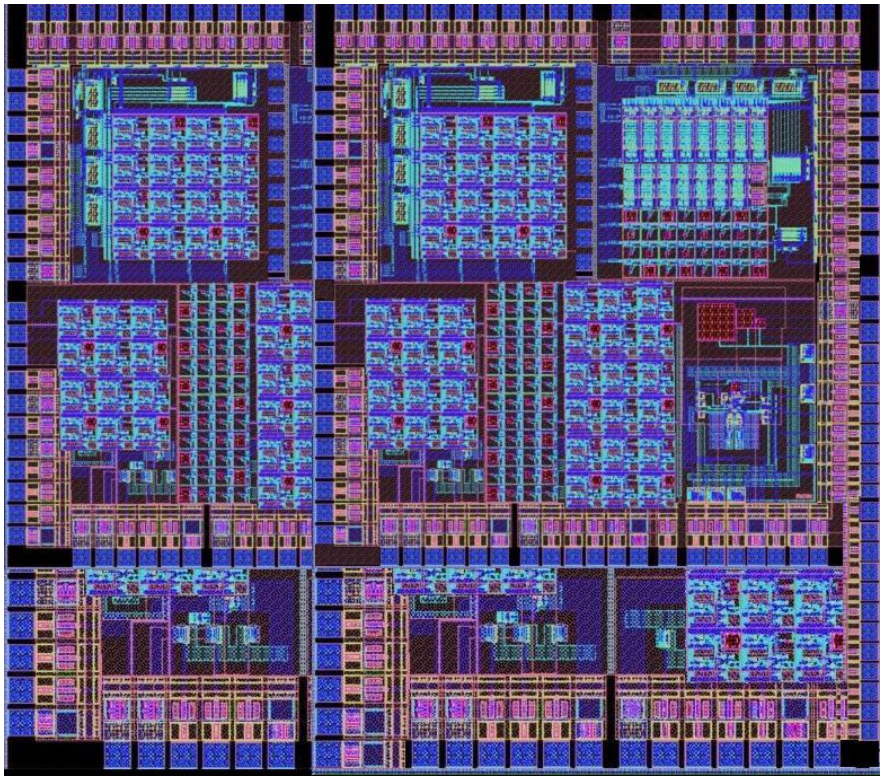


Figura 2.4 – Arquitetura Interna de um Microcotrolador

2.2.1 – Família 8051

O Intel 8051 faz parte de uma popular família de microcontroladores de 8 bits lançada pela Intel[4] em 1977. É conhecido por sua facilidade de programação, em linguagem assembly[5] graças ao seu poderoso conjunto de instruções. É tido como o microcontrolador mais popular do mundo, pois existem milhares de aplicações para o mesmo, e existem pelo menos dois mil fabricantes produzindo variantes e clones do modelo. Atualmente possui diversos modelos clones sendo produzidos por empresas diversas à Intel. Por ser um microcontrolador CISC, oferece um conjunto de instruções muito vasto que permite executar desde um

[4] Intel Corporation é a contração de Integrated Electronics Corporation, empresa multinacional de origem americana fabricante de circuitos integrados, especialmente microprocessadores.
[5] Linguagem Assembly é uma linguagem constituída de mnemônicos, que são mais fáceis para o ser humano ler, decorar e operar, porém mais próxima da linguagem de máquina em que cada dispositivo programável tem uma variação própria.

simples programa que faz piscar um LED até um programa de controle de acesso controlado por rede.

O 8051 possui uma memória ROM[6] que faz parte da arquitetura interna do chip, na qual será armazenado exclusivamente o programa que a CPU executará, não os dados, pois esses serão gravados em outra memória (RAM[7]), que pode ser interna ou externa. A memória ROM tem a característica de poder ser gravada apenas uma vez, em geral, na fábrica.

2.3 – LINGUAGEM DE PROGRAMAÇÃO

Uma linguagem de programação é um método padronizado para expressar instruções para um computador. É um conjunto de regras sintáticas e semânticas usadas para definir um programa de computador. Uma linguagem permite que um programador especifique precisamente sobre quais dados um computador vai atuar, como estes dados serão armazenados ou transmitidos e quais ações devem ser tomadas sob várias circunstâncias.

O conjunto de palavras (tokens), compostos de acordo com essas regras, constituem o código fonte de um software. Esse código fonte é depois traduzido para código de máquina, que é executado pelo processador.

Uma das principais metas das linguagens de programação é permitir que programadores tenham uma maior produtividade, permitindo expressar suas intenções mais facilmente do que quando comparado com a linguagem que um computador entende nativamente (código de máquina). Assim, linguagens de

[6] A sigla ROM é, pelo inglês, originária das iniciais de Read Only Memory - Memória Apenas de Leitura.
[7] RAM (Random Access Memory), ou memória de acesso aleatório (randômico), é um tipo de memória que permite a leitura e a escrita, utilizada como memória primária em sistemas eletrônicos digitais.

programação são projetadas para adotar uma sintaxe de nível mais alto, que pode ser mais facilmente entendida por programadores humanos. Linguagens de programação são ferramentas importantes para que programadores e engenheiros de software possam escrever programas mais organizados e com maior rapidez.

Linguagens de programação também tornam os programas menos dependentes de computadores ou ambientes computacionais específicos (propriedade chamada de portabilidade). Isto acontece porque programas escritos em linguagens de programação são traduzidos para o código de máquina do computador no qual será executado em vez de ser diretamente executado.

Como já mencionado a programação é o conjunto sequencial de passos, que a partir de agora podemos chamar de conjunto de instruções, que o computador deverá seguir para atingir um fim desejado pelo programador, seja uma operação matemática ou a automatização de um processo.

A maneira de se comunicar com um computador chama-se programa e a única linguagem que o computador entende chama-se linguagem de máquina, esta está diretamente relacionada à eletrônica, pois pelos componentes do computador passam eletricidade. Portanto todos os programas que se comunicam com a máquina devem estar em linguagem de máquina.

A forma como os programas são traduzidos para a linguagem de máquina classifica-os em duas categorias:

- Interpretados;
- Compilados.

2.3.1 – Interpretadores

Um interpretador lê a primeira instrução do programa, faz uma consistência de sua síntaxe e se não houver erro converte-a para linguagem de máquina para finalmente executá-la. Segue, então, para a próxima instução, repetindo o processo até que a última instrução seja executada ou a consistência aponte algum erro.

O interpretador precisa estar presente todas as vezes que vamos executar o nosso programa e o trabalho de checagem da sintaxe e tradução deverá ser repetido. Se uma parte do programa necessitar ser executada muitas vezes, o processo é feito o mesmo número de vezes.

2.3.2 – Compiladores

Um compilador lê a primeira instrução do programa, faz uma consistência de sua sintaxe e se não houver erro converte-a para linguagem de máquina e, em vez de executá-la, segue para a próxima instrução repetindo o processo até que a última instrução seja atingida ou a consistência aponte algum erro.

Se não houver erros, o compilador gera um programa em disco com o sufixo **.OBJ** com as instruções já traduzidas. Este programa não pode ser executado até que sejam agregadas a ele rotinas em linguagens de máquina que lhe permitirão a sua execução. Este trabalho é feito por um programa chamado “linkeditor” que, além de juntar as rotinas necessárias ao programa **.OBJ**, cria um produto final em disco com o sufixo **.EXE** que pode ser executado diretamente do sistema operacional.

Com isto a elocidade de execução do programa chega a ser 15 a 20 vezes mais rápida do que quando o programa é interpretado.

Outras vantagens, além da velocidade, podem ser mencionadas a falta da necessidade da presença do interpretador ou do compilador para executar o programa já compilado e inkeditado e que programas **.EXE** não podem ser alterados, o que protege o código-fonte[8].

Apesar das vantagens algumas considerações sobre os compiladores devem ser feitas.

Os interpretadores permitem que se produzam programas com maior facilidade na fase de aprendizado, pois podem ser criados e executados imediatamente, enquanto um compilador requer muito mais tempo para que um programa possa ser executado.

Um compilador não criará um programa em linguagem de máquina antes que este esteja absolutamente livre de erros, e durante o processo de aprendizado de uma linguagem a quantidade de erros gerados é relativamente grande, o que faz com que um interpretador seja muito mais conveniente.

Outro detalhe é que programas gerados em compiladores exigem um tipo de máquina específico para que funcionem. O tipo de máquina em que foi gerado o seu programa executável. Para que máquinas com outra arquitetura interna possam utilizar o mesmo programa, é necessário que o programador compile novamente o programa para este novo tipo de máquina, diferentemente de programas gerados sob o conceito de interpretadores, onde o programador só se preocupa com o funcionamento lógico do programa e não em que tipo de arquitetura de computadores este irá funcionar.

[8] Código Fonte: Seqüência de instruções de um programa, nele constam todas as operações realizadas pelo programa e as informações do mesmo.

2.3.3 – Linguagem C

A linguagem C foi primeiramente criada por Dennis M. Ritchie e Ken Thompson no laboratório Bell em 1972, baseada na linguagem B de Thompson que era uma evolução da antiga linguagem BCPL. B foi nomeada com a primeira letra de BCPL e C como a segunda. A próxima linguagem progressiva da idéia de C provavelmente se chamaria P.

C é uma linguagem vitoriosa como ferramenta na programação de qualquer tipo de sistema (sistemas operacionais, planilhas eletrônicas, procesadores de texto, gerenciadores de banco de dados, para soluções de problemas de engenharia ou física, etc.). Como exemplo, o sistema operacional UNIX foi desenvolvido em C.

A linguagem de programação C tornou-se rapidamente uma das mais importantes e populares, principalmente por ser muito poderosa, portátil, pela padronização dos compiladores existentes e flexível.

A linguagem de programação C é compilada, deste modo oferece agilidade na execução.

C foi desenhada para que o usuário possa planejar programas estruturados e modulares. O resultado é um programa mais legível e documentado. Os programas em C tendem a ser bastante compactos e de execução rápida.

2.4 – CONTROLE E AUTOMAÇÃO

Os sistemas de controle e automação utilizam, na maioria dos casos, o conjunto dos conceitos acima citados, desde a eletrônica, à computação e programação, visando produzir meios para otimizar o desempenho dos sistemas dinâmicos, melhorar a produtiidade, diminuir o trabalho árduo de várias rotinas de

operações manuais repetitivas, entre outros, automatizar um processo e/ou um ambiente. Estes têm desempenhado um papel fundamental no avanço da engenharia e da ciência.

Um Sistema de Controle pode ser definido em duas partes interdepententes [GEORGINI, 2003]:

- Sistema Controlado – sistema que executa a operação física e desejada;
- Equipamento de Controle – equipamento que recebe informações provenientes do operador, do processo a ser controlado, etc, e emite órdens ao Sistema Controlado.

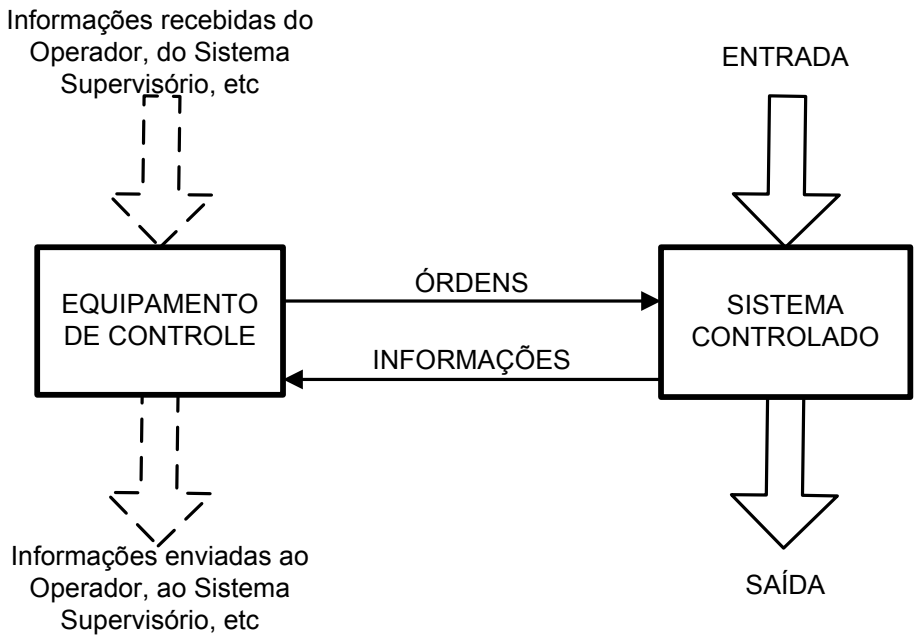


Figura 2.5 – Sistema de Controle

O objetivo em um sistema de controle é fazer com que o objeto a ser controlado atue por conta própria, sem interferência humana na maioria das suas

tarefas, ou seja, deixar que este tome decisões em direção ao objetivo como em um sistema inteligente e 'aparentemente', independente. A estes ambientes dá-se o nome de Sistemas com Realimentação, ou em Malha Fechada, ou seja, nestes casos o ambiente gera um erro de acordo com o objetivo a ser alcançado e este erro gera uma reentrada no sistema possibilitando uma nova ação do sistema que intenciona minimizar o erro ou alcançar o objetivo com maior precisão.

Porém o segmento da Engenharia de Controle também abrange sistemas não realimentados, ou seja, ambientes que, embora passíveis de gerar erros de acordo com o objetivo, não o repassam para o sistema. A este se dá o nome de Sistemas de Controle Automático a Malha Aberta, daí o título deste projeto.

Num sistema controlado tem-se a variável controlada que é a grandeza ou a condição que é medida e controlada. Também, a variável manipulada, a grandeza ou a condição modificada pelo controlador, de modo que afete o valor da variável controlada. Normalmente, a variável controlada é a saída do sistema. Controlar significa medir o valor da variável controlada do sistema e utilizar a variável manipulada ao sistema para corrigir ou limitar os desvios do valor medido a partir de um valor desejado.

2.4.1 – Sistemas em Malha Aberta

O controle em malha aberta consiste em aplicar um sinal de controle pré-determinado, esperando-se que, ao final de um determinado tempo, a variável controlada atinja um determinado valor ou apresente um determinado comportamento. Neste tipo de sistema de controle não são utilizadas informações sobre evolução do processo para determinar o sinal de controle a ser aplicado em um determinado instante. Mais especificamente, o sinal de controle não é calculado

a partir de uma medição do sinal de saída, deste modo, não há uma dependência do sinal de saída para realizar o controle do ambiente, o sinal de saída não exerce nenhuma ação de controle no sistema.

O sistema que se deseja controlar normalmente é chamado de planta e este pode ser parte de um equipamento ou apenas um conjunto de componentes de um equipamento que funcione de maneira integrada, com o objetivo de realizar determinada operação.

Figura 2.6 – Sistema em Malha Aberta

2.4.2 – Sistemas em Malha Fechada

Neste caso o sistema de controle utiliza das informações sobre como a saída de controle está evoluindo relativo ao objetivo do controle. São utilizadas as saídas para determinar o sinal de controle que deve ser aplicado ao processo em um instante específico. Isto é feito a partir de uma realimentação da saída para a entrada. Em geral, a fim de tornar o sistema mais preciso e de fazer com que ele reaja a perturbações externas, o sinal de saída é comparado com um sinal de referência, normalmente relacionado ao objetivo do controle já previamente determinado, e o desvio (erro) entre estes dois sinais é utilizado para determinar o sinal de controle que deve efetivamente ser aplicado ao processo. Assim, o sinal de controle é determinado de forma a corrigir este desvio entre a saída e o sinal de referência.

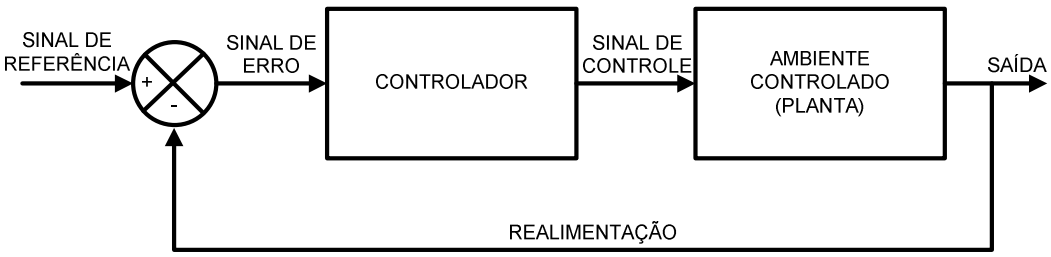


Figura 2.7 – Sistema em Malha Fechada

2.4.3 – Malha Aberta x Malha Fechada

Uma vantagem do sistema de controle de malha fechada é o fato de que o uso de realimentação faz com que a resposta do sistema seja relativamente insensível a distúrbios e variações internas nos parâmetros do sistema. Dessa maneira, é possível a utilização de componentes relativamente imprecisos e baratos para obter o controle preciso de determinado sistema, ao passo que isso não é possível nos sistemas de malha aberta.

Do ponto de vista da estabilidade, o sistema de controle de malha aberta é mais fácil de ser construído devido ao fato de a estabilidade ser um problema menos significativo. Por outro lado, a estabilidade constitui um problema importante nos sistemas de controle de malha fechada, que podem apresentar uma tendência de correção de erros além do necessário, causando oscilações de amplitude constante ou variável. (OGATA, 2003)

Capítulo 3 – Produção Técnica

O projeto - movimento de um móvel de acordo com a orientação recebida nos parâmetros de um ângulo geométrico em sistema de malha aberta - tem como objetivo realizar o controle de um móvel, neste caso um carrinho de controle remoto, a partir da passagem de uma variável nos parâmetros de ângulo geométrico.

Para a inserção do ângulo geométrico no ambiente proposto foi desenvolvida uma interface programada em linguagem computacional. Essa interface tem por objetivo captar o ângulo geométrico digitado pelo controlador[9], fazer as devidas transformações matemáticas padronizadas na programação e enviar o ângulo via porta serial (DB-9)[10] que será recebido pela interface serial do kit CW552, tendo por responsabilidade efetuar o controle do móvel, enviando sinais de controle ao circuito eletrônico desenvolvido que está conectado ao controle remoto do móvel.

3.1 – SISTEMA ADOTADO

Todo sistema de controle possui um objetivo a ser alcançado, neste caso tem-se como variável controlada o movimento do carrinho, e, para alcançá-lo, controla-se a variável manipulada, que neste projeto é o ângulo do movimento que é alcançado pela quantidade de tempo em que o móvel se movimenta com as rodas

[9] Entende-se por controlador nesse ponto um ser humano interagindo com o ambiente.

[10] Um conector DB contém duas ou mais fileiras paralelas de pinos ou soquetes, usualmente circundados por um envoltório metálico em formato de **D** que provê um anteparo contra interferência eletromagnética e garante um encaixe correto. A parte que contém os pinos de contacto é chamada de "conector macho", enquanto a parte que contém soquetes é denominada "conector fêmea", ou simplesmente "soquete". O envoltório do soquete se encaixa com firmeza no envoltório que contém os pinos. Ambos os envoltórios são conectados à blindagem dos cabos (quando cabos blindados são usados), criando uma proteção eletricamente contínua que engloba todo o sistema do cabo e do conector. DB-9 significa que o conector possui conexão com 9 pinos.

giradas para a esquerda, perfazendo o sentido anti-horário, ou para a direita, no sentido horário.

Porém, para realizar o movimento em todos os 360º poderia haver inúmeros possíveis movimentos, caso não fossem especificados padrões limitantes para o objetivo.

Deste modo, os possíveis movimentos realizáveis foram fixados entre 0º e 360º em variações de 10º, ou seja, o móvel deve mover-se em 0º, que resultaria em movimento frontal, 10º, 20º, 30º, ..., 340º, 350º e 360º, em movimento anti-horário e horário.

Para alcança tal feito, foi realizado um estudo no móvel adquirido e medido quanto tempo o móvel necessita, para realizar o movimento em cada angulação.

3.1.1 – Móvel Adquirido

Conforme já previsto, não é escopo deste projeto a confecção de um móvel para os testes. Este projeto visa o estudo do movimento controlado a partir da variável ângulo do movimento.

Porém, intencionando ir mais a fundo nesta pesquisa, proporcionando um ambiente que valide a proposta, um carrinho de controle remoto foi adquirido para ser utilizado no ambiente de teste. Vale ressaltar que o móvel, por não ser produzido para este fim, não possui nível de precisão elevado, podendo contribuir para possíveis erros no movimento.

A figura 3.1 mostrará uma ilustração do móvel adquirido.

Figura 3.1 – Móvel Adquirido para o Projeto

Para ter um maior controle sobre o movimento do carrinho, foi inserido em série ao motor um resistor de 5,6 Ω, a fim de reduzir a velocidade do carrinho.

3.2 – AMBIENTE DE PROGRAMAÇÃO

O ambiente de desenvolvimento e programação utilizado para o desenvolvimento deste projeto é baseado em linguagem C, e o compilador utilizado foi Turbo C++ 3.0 desenvolvido pela Borland International, Inc.

A programação feita para o microcontrolador 8051 foi também baseada em linguagem C e o compilador utilizado foi SDCC – Small Device C Compiler versão 2.6.

3.3 – INTERFACE COMPUTACIONAL

Uma interface computacional é o mecanismo, automático, que integra dois ambientes distintos, porém que necessitam se comunicar. Um programa que

interage com o usuário e com outro ambiente, fazendo como que uma tradução de idioma entre os dois.

No caso específico deste projeto a Interface Computacional foi gerada em linguagem computacional utilizando compilador C.

A interface tem por objetivo receber o ângulo e sentido do movimento desejado pelo usuário, converte-os e envia para o microcontrolador, dentro dos parâmetros estabelecidos na comunicação serial.

3.3.1 – Método ASCII

ASCII é um acrônimo para American Standard Code for Information Interchange. É um conjunto de códigos para o computador representar números, letras, pontuação e outros caracteres. Surgido em 1961, um dos seus inventores foi Robert W. Bemer.

ASCII é uma padronização da indústria de computadores, onde cada carácter é manipulado na memória discos etc, sob forma de código binário. O código ASCII é formado por todas as combinações possíveis de 7 bits, sendo que existem várias extensões que abrangem 8 ou mais bits.

Abaixo segue uma tabela representativa de todos os caracteres possíveis na padronização ASCII.

				B_6	0	0	0	0	1	1	1	1
				B_5	0	0	1	1	0	0	1	1
				B_4	0	1	0	1	0	1	0	1
B_3	B_2	B_1	B_0	HEXA	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	0	0	0	**0**	NUL	DLE	*ESPAÇO*	0	@	P	`	p

0	0	0	1	**1**	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	**2**	STX	DC2	“	2	B	R	b	r
0	0	1	1	**3**	ETX	DC3	#	3	C	S	c	s
0	1	0	0	**4**	EOT	DC4	$	4	D	T	d	t
0	1	0	1	**5**	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	**6**	ACK	SYN	&	6	F	V	f	v
0	1	1	1	**7**	BEL	ETB	‘	7	G	W	g	w
1	0	0	0	**8**	BS	CAN	(	8	H	X	h	x
1	0	0	1	**9**	HT	EM	)	9	I	Y	i	y
1	0	1	0	**A**	LF	SUB	*	:	J	Z	j	z
1	0	1	1	**B**	VT	ESC	+	;	K	[	k	{
1	1	0	0	**C**	FF	FS	,	<	L	\	l	|
1	1	0	1	**D**	CR	GS	-	=	M	]	m	}
1	1	1	0	**E**	S0	RS	.	>	N	^	n	~
1	1	1	1	**F**	SI	US	/	?	O	_	o	DEL

Tabela 3.1 – Tabela ASCII

A motivação pela escolha da padronização ASCII neste projeto deu-se pelo fato do padrão utilizar 7 bits para representação dos caracteres e, como previsto, a comunicação entre o computador e o microcontrolador dar-se-ía pela interface serial, RS-232, trabalhando também com 7 bits com 1 bit de controle. Deste modo, fica perfeitamente adaptável a integração entre os sistemas.

Foi convencionado que os sentidos coletados representariam caracteres pré-aguardados. Caso escolhido movimento anti-horário, o caractere ‘e’, com sua

correspondente em decimal igual a 101 e em binário 1100101, é enviado para ao espaço de memória do computador, alocado para envio ao microcontrolador via porta serial RS-232. Porém, caso escolhido movimento horário, o caractere ‘d’, com sua correspondente em decimal igual a 100 e em binário 1100100, é enviado ao espaço de memória específico. Caso contrário, se escolhido movimento frontal o caractere ‘0’ é selecionado, com sua correspondente em decimal igual a 48 e em binário 0110000.

Para o ângulo selecionado no ambiente convencionou-se que, conforme o movimento dá-se de com intervalos de 10º, entre 0º e 360º, é feito uma divisão simples por 10 a cada ângulo escolhido, desta forma fica facilmente possível representar todos os valores dentro do intervalo.

Por exemplo:

Caso escolhido movimento Anti-horário é enviado o caractere ‘e’ para o microcontrolador, que o tratará como informação binária 1100101. Após esta escolha, decide-se pelo ângulo do movimento, supondo 70º, é feito uma divisão simples por 10, o que resulta em 7 e é enviado este caractere ao microcontrolador que o tratará como informação binária 0000111. Deste modo foi possível representar todos os 37 níveis, podendo ainda ser representados em escala mais precisa como a cada 5 graus.

No próximo item seguem algumas telas com explicações do programa responsável pela interface computacional.

3.3.2 – Free Serial Port Monitor

Para o desenvolvimento da Interface Computacional e sua transmissão pela porta serial, foi feito uso do software livre para uso Free Serial Port Monitor. Esse software tornou possível analisar os dados enviados para a porta serial. Na fase de desenvolvimento do software do microcontrolador, o Free Serial Port Monitor possibilitou a análise dos dados recebidos na porta serial. Este software lê os registradores da porta serial e, quando recebido um dado, imprime o dado recebido na tela.

O Free Serial Port Monitor monitora as portas seriais utilizadas por qualquer aplicação que rode nas versões NT 4.0, 2000, XP e 2003 Server do Microsoft Windows. Esse software intercepta, analisa e mostra todos os dados transferidos entre a aplicação e o dispositivo conectado na porta serial. É um mecanismo eficiente que ajuda a encontrar e eliminar erros de software ou hardware.

No projeto foi utilizada a versão 3.31 do Free Serial Port Monitor, disponibilizada em maio de 2005. Outras informações e download gratuito no endereço http://www.serial-port-monitor.com

Uma representação do uso do Free Serial Port Monitor em conjunto com a Interface Computacional produzida é ilustrado a seguir. Neste exemplo é feito o envio pela porta serial de dois dados, o sentido, representado por 100 que significa movimento horário ou à direita e o ângulo de 250º representado por 25, sendo 250 dividido por 10.

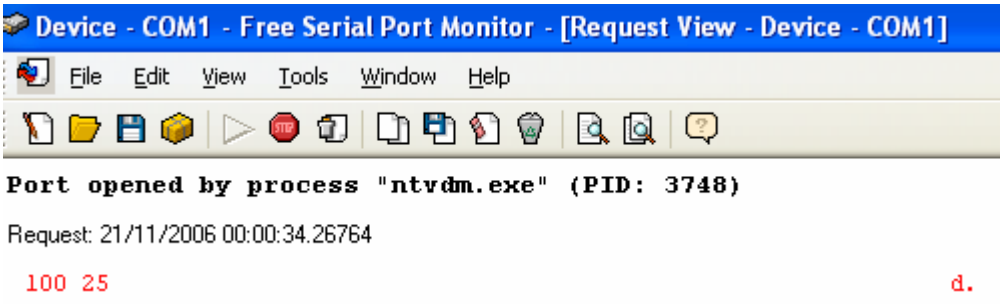


Figura 3.2 – Free Serial Port Monitor

3.3.3 – Ambiente visual da Interface Computacional

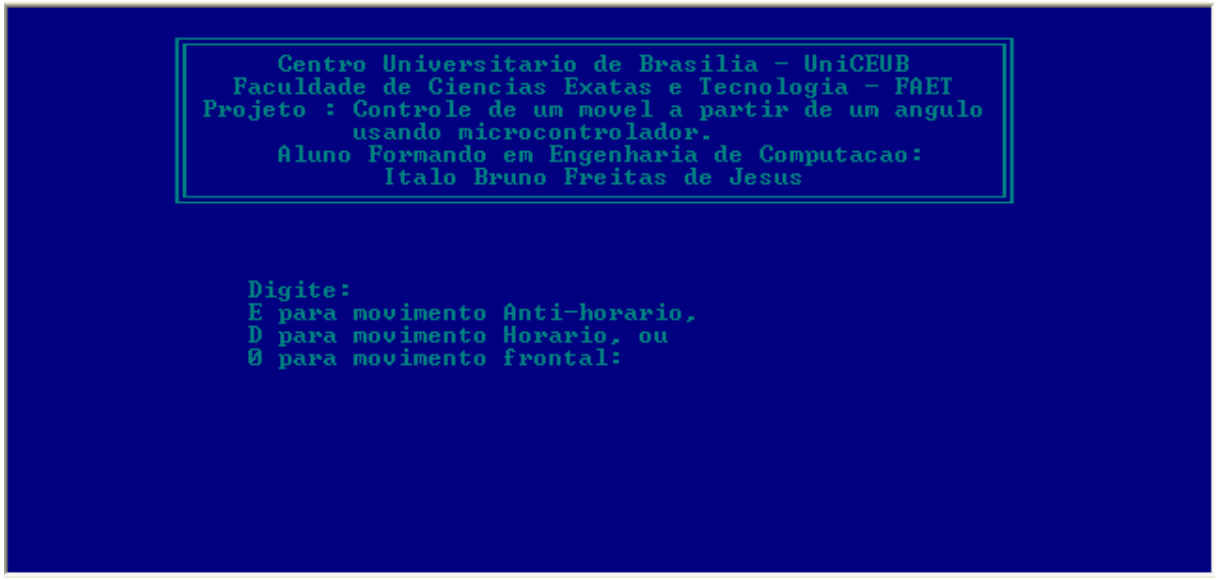


Figura 3.3 – Tela de Entrada do Programa Interface Computacional

Na tela acima se observa as opções para o movimento do móvel que podem perfazer o sentido anti-horário, horário ou frontal digitando-se as opções ‘E’ para anti-horário, ‘D’ para horário ou ‘0’ para frontal.

O programa foi feito para que não seja Case Sensitive[11].

[11] Case sensitive (sensível ao tamanho da letra) em computação significa que um programa ou um compilador faz a diferença entre letras maiusculas e minusculas.

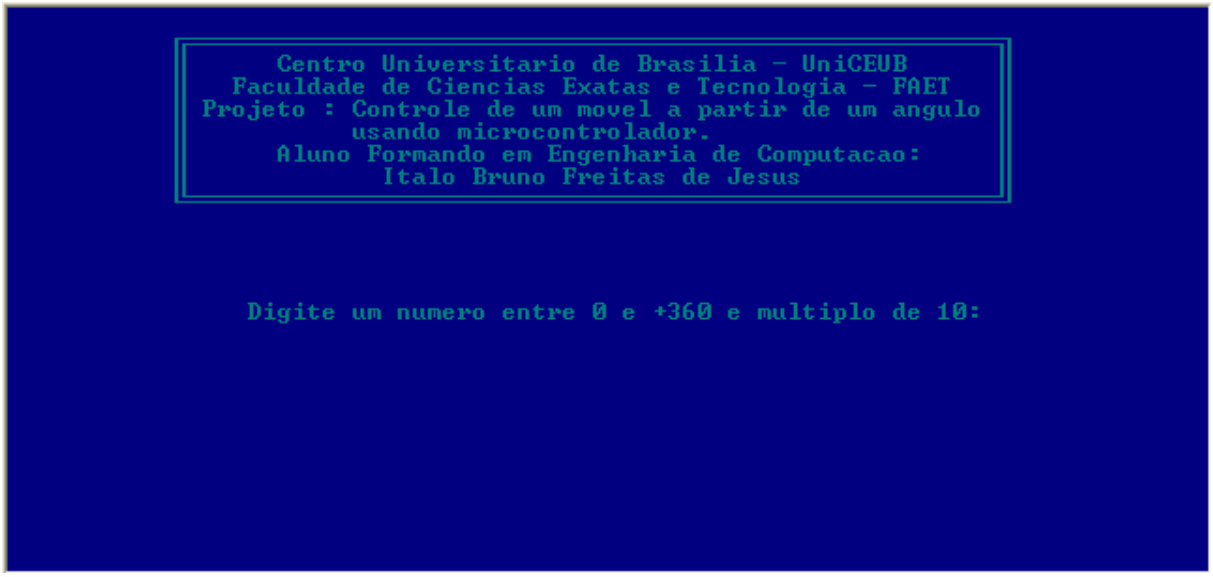


Figura 3.4 – Tela para Inserção do Ângulo

A figura 3.4 ilustra a coleta do ângulo geométrico inserido pelo usuário, sendo que o mesmo deve ser múltiplo de 10, maior ou igual a 0 e menor ou igual a 360. O tratamento referente a múltiplo foi conseguido coletando as informações inseridas e realizando uma operação matemática de divisão do ângulo inserido por 10, caso este resultado seja diferente de 0 o ângulo inserido não é múltiplo de 10 e deve-se inserir outro valor compatível.

Deste modo, caso alguma das informações acima mencionadas não seja seguida, o programa retorna para a tela da Figura 3.5 para que o usuário efetue a entrada no programa corretamente. Deste modo a informação errada não é enviada ao microcontrolador. Abaixo segue a tela apresentada para tratamento de erros.

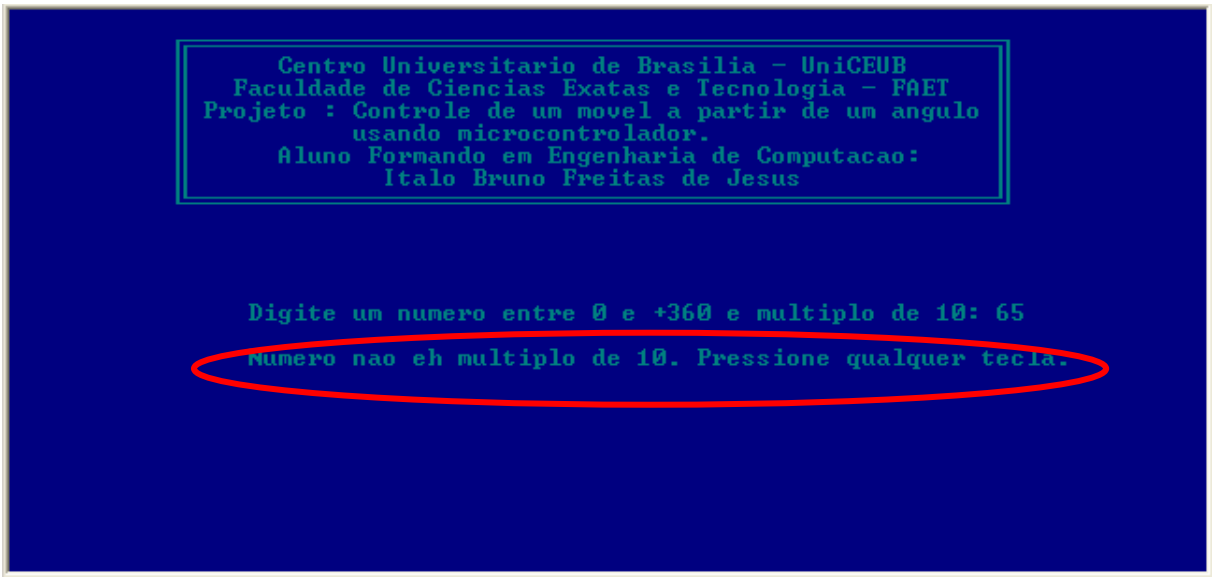


Figura 3.5 – Tela para o Tratamento de Erros

Uma vez que o ângulo inserido estiver dentro dos parâmetros pré-estabelecidos o programa envia o ângulo e apresenta a tela de retorno abaixo. Neste exemplo foi efetuado um movimento sentido anti-horário a 90º.

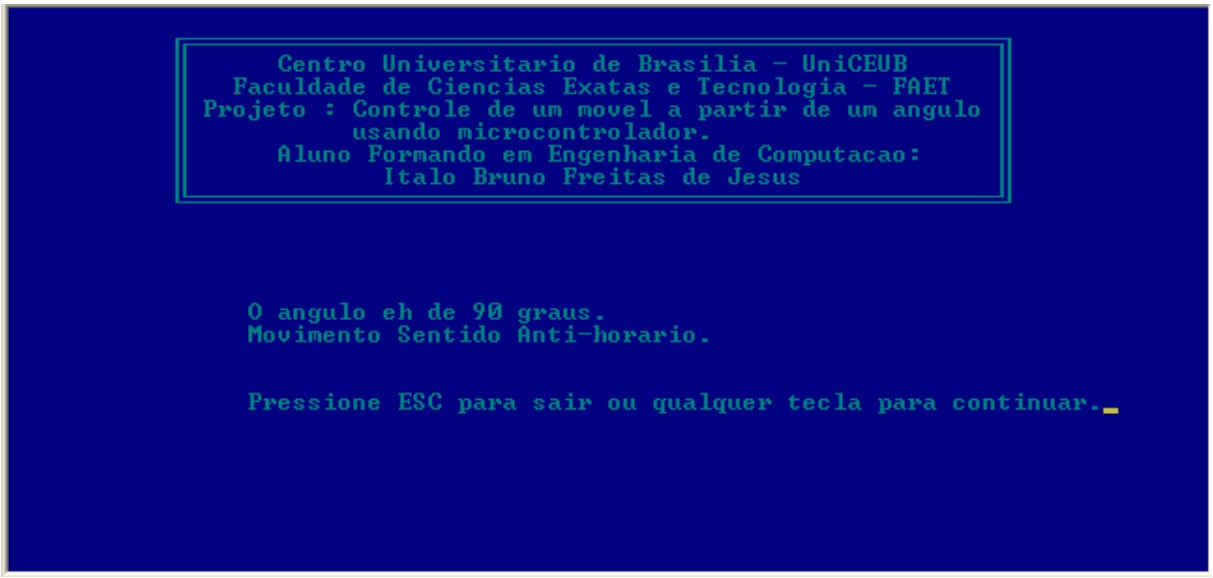


Figura 3.6 – Tela de Confirmação dos Dados Inseridos

Para que o usuário possa sair do programa é necessário, no momento da tela representada na figura 3.6, ser pressionada a tecla ESC. Caso qualquer outra

tecla seja pressionada o programa retorna para a origem a fim de receber um novo conjunto de dados para efetuar um outro movimento, sendo necessário inserir o sentido e o ângulo.

3.3.4 – Código Fonte – Interface Computacional

O código fonte da Interface Computacional encontra-se em sua totalidade no Apêndice I, nesta sessão serão explanadas algumas das funcionalidades do programa.

```
#include <stdio.h>
#include <conio.h>
#include <dos.h>

#define end_ctrl_50 0x03FD /* Enderecos da UART 8250 */
#define end_dado_50 0x03F8
#define LSB 0x03F8
#define MSB 0x03F9
#define LRC 0x03FB
#define MCR 0x03FC
#define LSR 0x03FD
#define MSR 0x03FE
#define IOR 0x03F8
#define IIR 0x03FA
#define IER 0x03F9
```

O código acima tem duas partes, a primeira é onde são feitos a inclusão de bibliotecas já prontas para serem utilizadas no programa. Isso é feito com a inserção de #include <*nome_da_biblioteca*.h>.

Neste programa foram utilizados as bibliotecas:

- stdio.h: significa "standard input-output header" é um cabeçalho da biblioteca padrão da linguagem de programação C que contêm definições macro e declaração de funções e tipos usados em

várias operações de entrada e saída padrões dos dispositivos básicos (teclado, ficheiros: discos).

- conio.h: Significa Console I/O. As funções presentes nessa biblioteca são úteis para manipular caracteres na tela, especificar cor de caractere e de fundo.

- dos.h: Biblioteca pré-definida para trabalhar com portas físicas do computador. Já tem em si uma série de funções que auxiliam neste objetivo.

A segunda parte desta introdução do programa é onde são feito definições sobre algumas variáveis utilizadas no programa. Isso é feito por meio do comando #define *nome_variável valor*.

Neste programa as variáveis são endereços de memória, e estão representadas em hexadecimal.

O código abaixo descreve a inicialização da porta serial segundo os padrões RS-232C especificado no item 2.1.2 – desta monografia.

```
void inic8250() {
outportb(LRC, 0x80);
outportb(MSB, 0x00);
outportb(LSB, 0x0C);
outportb(LRC, 0x07); }
```

O código seguinte é utilizado para limpar e resetar a comunicação serial. Isto é feito pela utilização da função inportb que lê um byte de uma porta física. Neste caso, é feita a leitura do bit menos significativo, do mais significativo.

```
// reseta a 8051
void reseta8051() {
inportb(LSR);
inportb(MSR);
inportb(IOR);
inportb(IIR);
}
```

O código abaixo desabilita o uso de interrupções utilizando a função pré-definida outportb, que insere um valor em uma porta física. Neste caso insere-se no registrador de interrupção o valor 0.

```
/* desabilita interrupcoes da 8250 */
void disable8250() {
outportb(IER, 0x00);
}
```

A função abaixo envia um caractere presente no registrador da porta serial pela interface serial. Isto se dá com a leitura do último bit do caractere, onde enviado a informação, bit a bit, enquanto estiver bit na porta de saída definida por end_dado_50.

```
/* transmite um caracter */
void transm_dado(unsigned char caracter) {
while (!(32 & inportb(end_ctrl_50)) == 32);
outportb(end_dado_50, caracter);
}
```

Função fApresentacao

Usada apenas para criar um cabeçalho de identificação no programa, conforme figura 3.7.

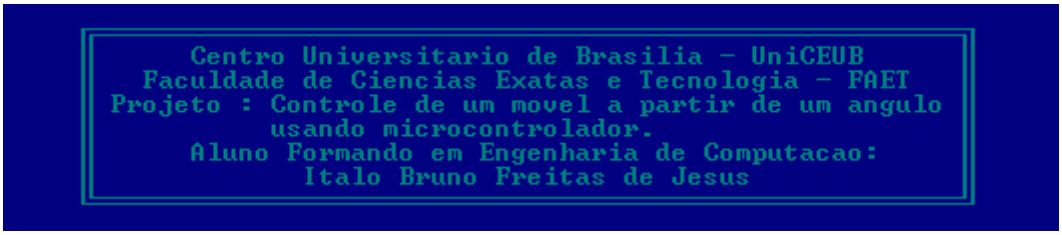


Figura 3.7 – Cabeçalho do Programa

Função fSentido

Função utilizada para coletar o sentido que o móvel deverá perfazer e retornar o sentido em forma de caractere para a função principal.

Para tal escolha deve-se digitar ‘E’ para esquerda ou anti-horário, ‘D’ para direita ou horário, ou ‘0’ para sentido frontal. O programa já trabalha não case sensitive, deste modo não há diferença na entrada de dados ser ‘E’ (maiúsculo) ou ‘e’ minúsculo, o mesmo para ‘D’ (maiúsculo) ou ‘d’ (minúsculo).

A parte do programa abaixo diz que enquanto os dados que estão sendo digitados pelo teclado forem diferente das aceitáveis pelo programa este ficará em looping, até que sejam digitadas as informações corretas.

```
do
while((sentido1 != '0') && (sentido1 != 'E') && (sentido1 != 'D') &&
(sentido1 != 'e') && (sentido1 != 'd'));
```

Ainda nesta função convencionou-se que se o campo sentido escolhido for 0 é enviado um caractere de valor 48 em decimal que é correspondente ao número ‘0’ no padrão ASCII, caso escolha-se ‘e’ ou ‘E’ o valor enviado é 101, correspondente à letra ‘e’ ou se escolhido ‘d’ ou ‘D’ é enviado o valor 100 correspondente à letra ‘d’. Essa convenção não influi diretamente no programa, poderiam ser utilizados

qualquer caractere, foram estes os escolhidos por não terem relação com nenhuma outra função do programa, ou conjunto de dados utilizados.

Função fVerifica

A função verifica tem por objetivo coletar o ângulo a ser enviado ao microcontrolador e retornar um número inteiro para a função principal.

O campo do programa *while((angulo2 < 0) || (angulo2 > 360))* coloca o programa em looping até que a condição do ângulo ser maior do que 0 e menor do que 360 seja cumprida.

Na parte *if ((angulo2%10)!=0)* exige-se que a informação inserida seja múltiplo de 10. Esta é uma operação matemática que informa logicamente se o resultado da divisão entre angulo2 e 10 é diferente de 0, caso diferente de 0, o número na variável angulo2 não é múltiplo de 10.

Função fComparanp

A função comparanp tem como objetivo apenas gerar uma informação a ser impressa na tela e não retorna nenhum valor válido para a função principal.

Como parâmetro ela recebe duas variáveis, passadas pelo programa principal, o sentido e o ângulo escolhidos. As considerações dão da seguinte forma:

- Se o sentido for igual a ‘e’ ou ‘E’ e o ângulo diferente de 0 então imprime-se na tela *Movimento Sentido Anti-horario.*

- Se o sentido for igual a ‘0’ ângulo igual a 0 então imprime-se na tela *Movimento para frente.*

- Se o sentido for igual a ‘d’ ou ‘D’ e o ângulo diferente de 0 então imprime-se na tela *Movimento Sentido Horario.*

Esta função, quando requisitada, apenas verifica o sentido do movimento e no final do programa imprime na tela se o sentido escolhido é Anti-horário, Horário ou Frontal.

Função fEnviacaracter

Esta função, quando requisitada, envia um caractere para a função transm_dado que fará a transferência física do dado pela porta serial.

Função main

Esta é a função principal do programa, por ela que há uma inicialização do programa.

Esta função começa inicializando as cores de fundo e da fonte da interface com o comando abaixo descrito.

```
textbackground(1);
textcolor(3);
```

Em seguida inicializa a porta serial a ser utilizada.

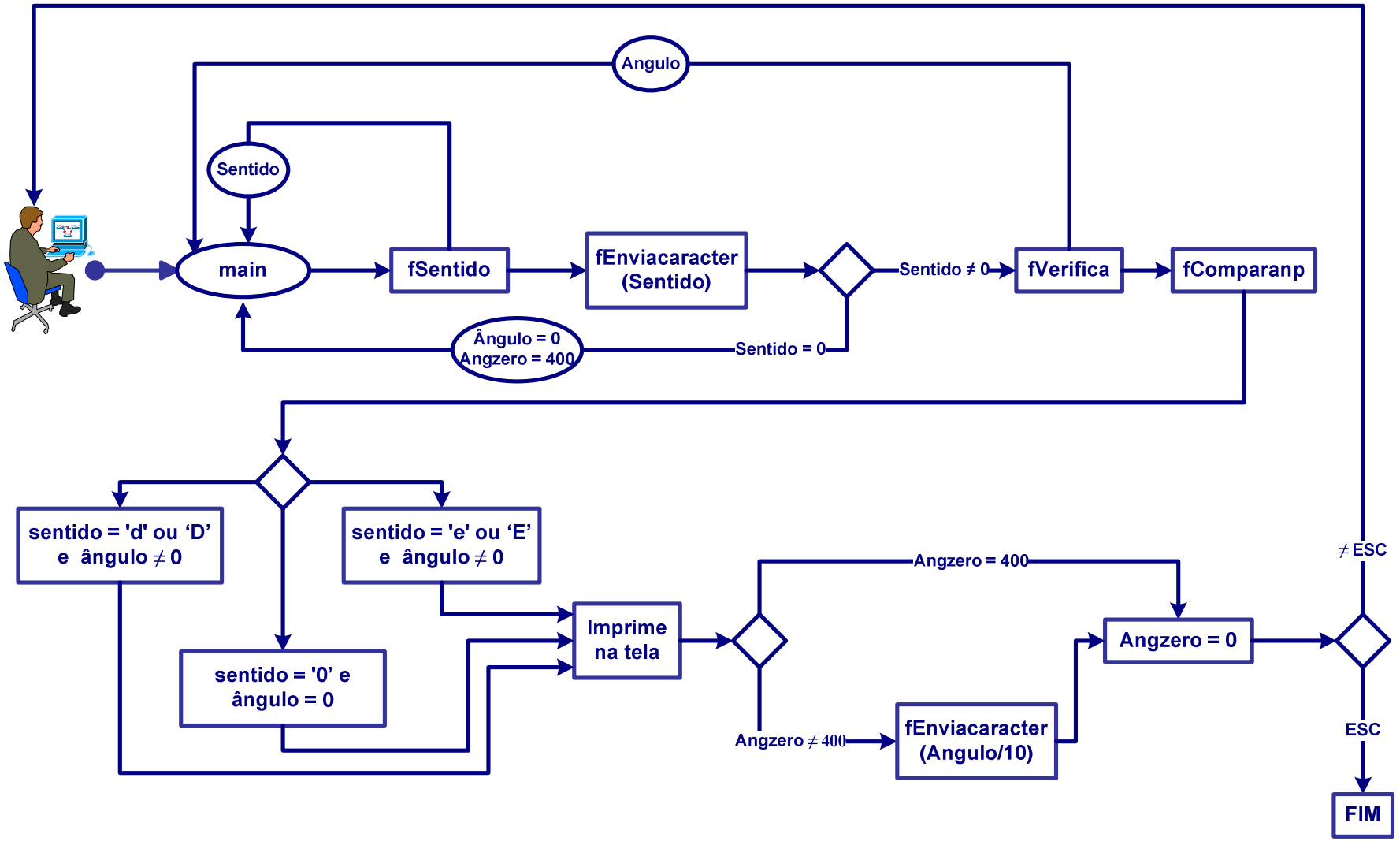
```
inic8250();
reseta8250();
disable8250();
```

Sequencialmente o programa entra em repetição até que a variável fim seja preenchida com a informação da tecla ESC.

```
while(!(fim==27))
```

A partir deste ponto dá-se a inicialização do programa para o usuário do ambiente, é deste ponto em diante que as funções com interação com o controlador iniciarão, necessitando a inserção de dados via teclado.

3.3.5 – Fluxograma da Interface Computacional



Fluxograma 3.1 – Fluxograma Interface Computacional

3.4 – MICROCONTROLADOR 8051

3.4.1 – Kit CW552

Para desenvolvimento deste projeto foi utilizado o Kit CW552 que é uma ferramenta de desenvolvimento simples e completa para sistemas baseados em microcontroladores. O kit é composto por base de apoio, placa de controle com o processador 80C552, display LCD de duas linhas de 40 caracteres (2x40), fonte de alimentação e interface serial RS-232C a ser ligada a um micro PC. O microcontrolador 80C552 tem a arquitetura básica do bem difundido 80C51 da INTEL, acrescido de 8 canais conversores A/D de 10 bits, duas saídas PWM com resolução de 8 bits, WatchDog, interfaces seriais UART e i2C. Tem amplo uso em aplicações de tempo real, tipicamente instrumentação, controle industrial e controle de processos [Manual Referência]

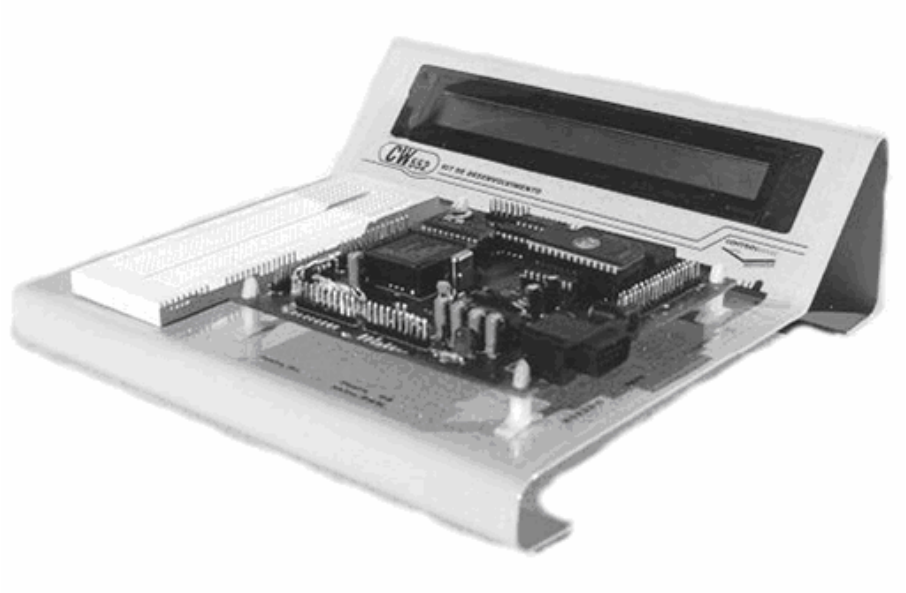


Figura 3.8 – Kit cw552

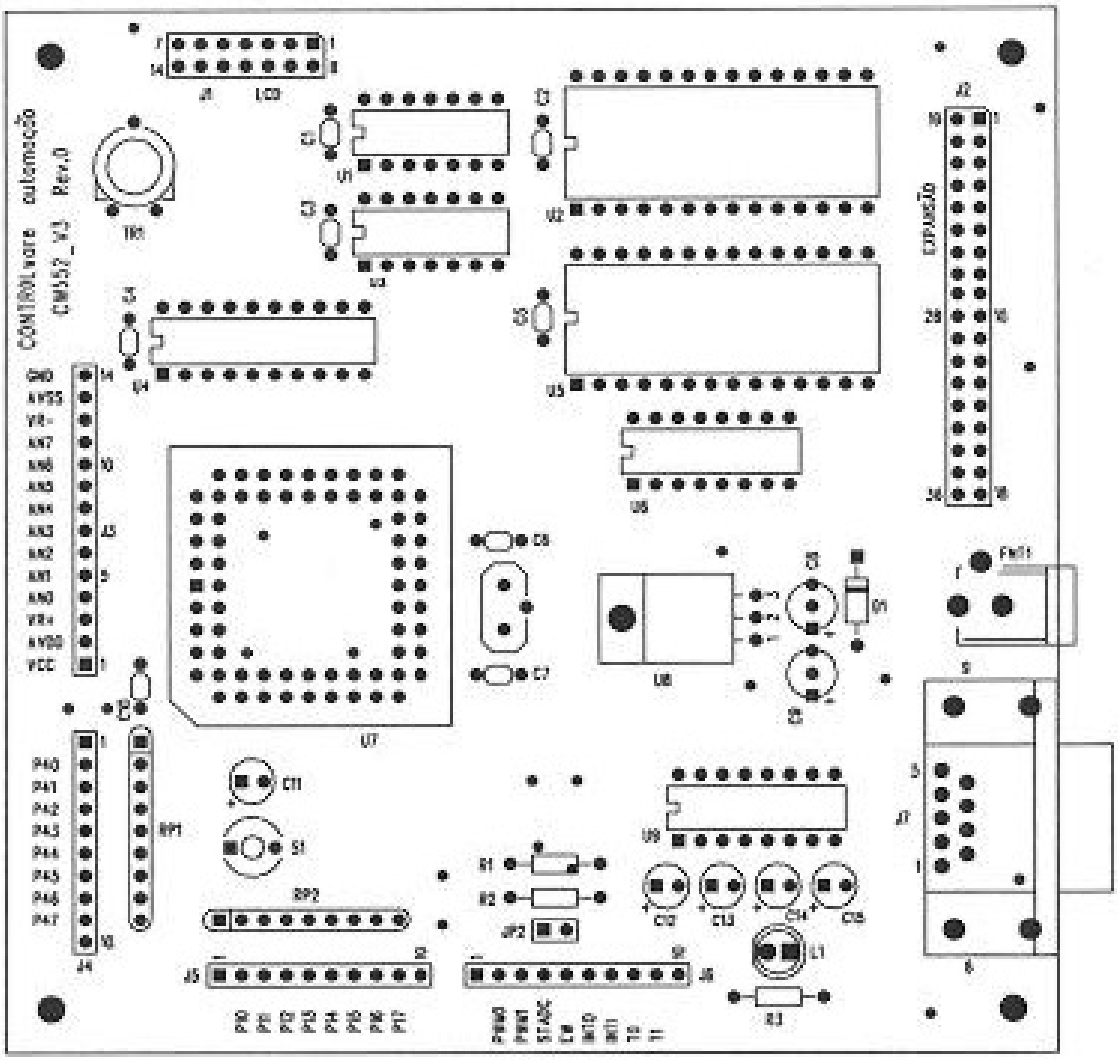


Figura 3.9 – Layout da Placa

No layout da placa de circuito impresso os conectores utilizados foram P41, P42 e P43, sendo as saídas do circuito lógico. Tiveram como função o acionamento do circuito de controle do móvel, conforme a tabela 3.2.

Conector	Função
P41	Acionamento do circuito de ativação para Esquerda
P42	Acionamento do circuito de ativação para Frente
P43	Acionamento do circuito de ativação para Direita

Tabela 3.2 – Tabela de Acionamento

Também foram utilizados os pinos de fornecimento de energia como fontes de tensão, sendo estes de 5 Volts. Também pinos de aterramento (GND), para suprimento de circuito impresso desenvolvido para este projeto.

Ainda neste kit, foi utilizada a porta serial, interface com o computador, que tem como função o recebimento das informações do sistema, sentido e ângulo. Sendo este um sistema de malha aberta, não há retorno de informação pela porta serial, não havendo coleta de dados nem passagem de informações no sentido microcontrolador para computador.

A comunicação entre o computador e o microcontrolador deu-se utilizando um cabo conversor da porta USB[12], saindo do computador, para a porta serial do kit. O cabo adquirido para este fim, deu-se pelo fato do computador utilizado, um laptop - HP, não possuir interface serial fisicamente instalada, sendo necessário utilizar uma interface física USB. A ilustração 3.10 mostra o cabo e a interface USB.

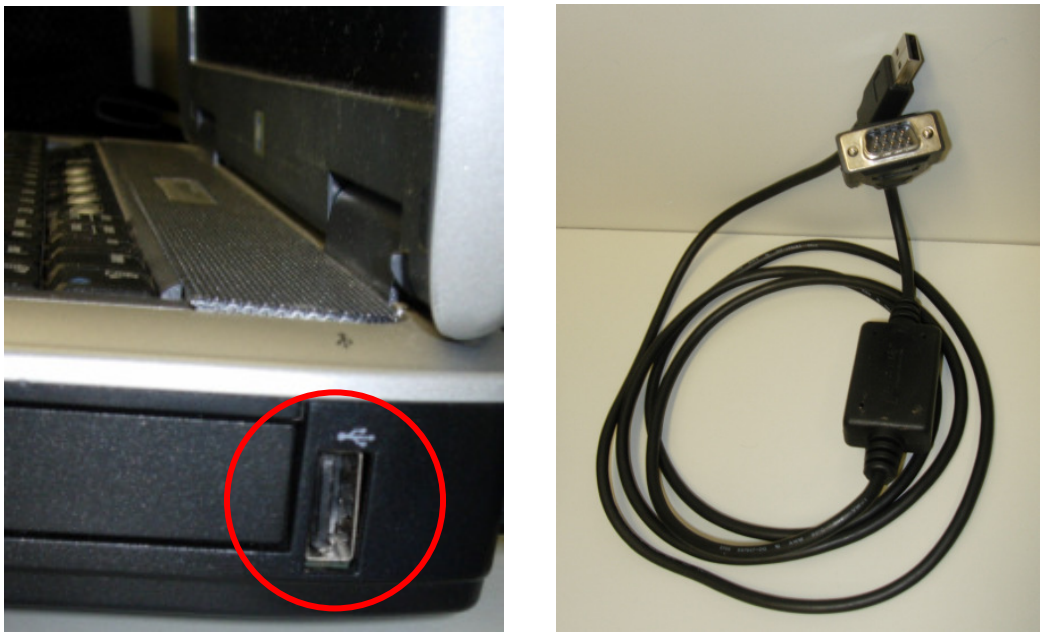


Figura 3.10 – Interface USB e Cabo Leadership USB – DB-9

[12] USB: Universal Serial Bus. É uma interface também serial porém que tem características elétricas próprias.

O display que o kit oferece também foi utilizado objetivando apresentar na tela as informações recebidas pela porta serial, imprimindo na mesma as expressões de acordo com o dado recebido, conforme tabela 3.3. Angulo:

Dado (número inteiro)	Expressão Impressa
48 ou 0	Movimento Sentido Frontal
101	Movimento Sentido Anti-Horário
100	Movimento Sentido Horário
Ângulo	Angulo: *Ângulo*

Tabela 3.3 – Tabela das Expressões Impressas no Display do Kit cw552

3.4.2 – Programação do Microcontrolador

O microcontrolador permite uma programação, delegando tarefas a serem desempenhadas, como por exemplo, a impressão dos dados no display. A programação do microcontrolador tem como função receber os dados enviados pela porta serial, tratá-las, identificando a informação recebida e tomando algumas decisões e realizando algumas tarefas.

Macrofuncionamento da programação no Microcontrolador

Na inicialização do microcontrolador aparecerá no display do microcontrolador a informação Porta Serial Aguardando .... e Produzido por Italo Bruno até que um dado seja recebido no registro de memória da porta serial do microcontrolador, conforme figura 3.11.

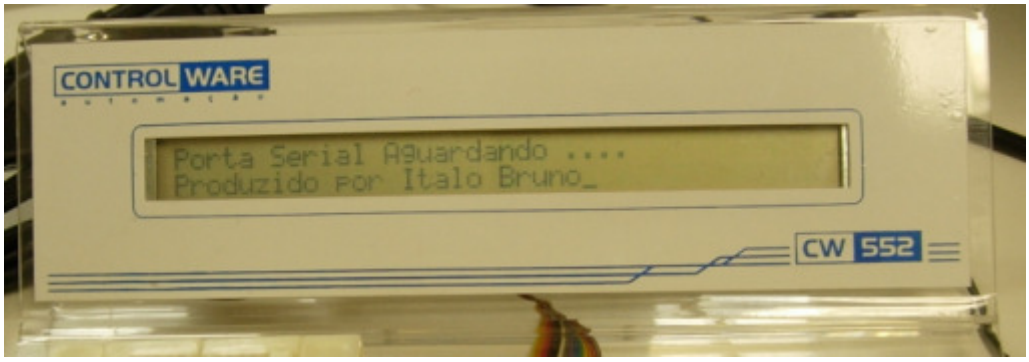


Figura 3.11 – Inicialização do Microcontrolador

Quando um dado é coletado na Interface Computacional, sentido e ângulo, é enviado para o kit CW552 via porta serial e será tratado pelo microcontrolador 80C552. Sendo este dado o valor decimal 48 ou 0 imprime-se na tela a expressão: Movimento Sentido Frontal. Posteriormente modifica-se a porta P42 em seu nível lógico de 0 para 1 o que proporciona nesta porta mudança de 0Volts para 5Volts, isto aciona o circuito de controle fazendo o móvel mover-se no sentido frontal pela quantidade de tempo devida para atingir o movimento frontal. A figura 3.12 ilustra este passo.

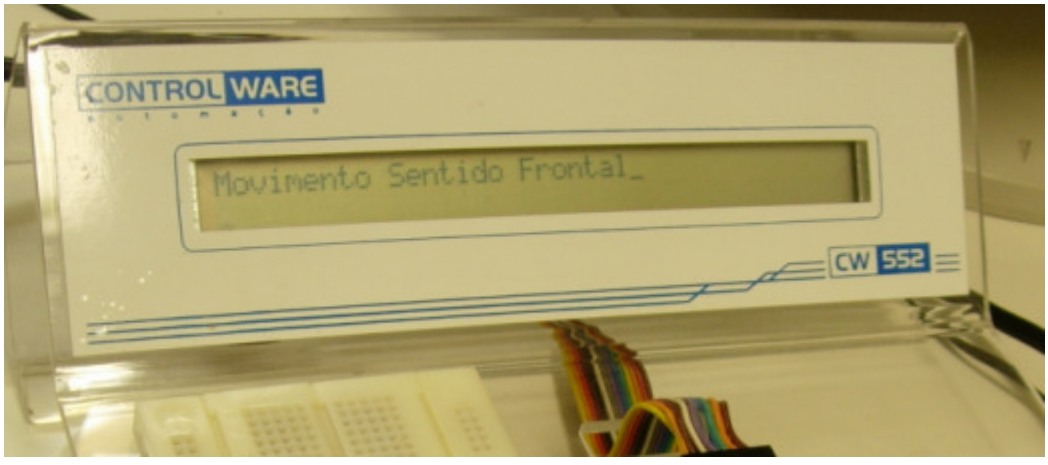


Figura 3.12 – Movimento Sentido Frontal

Caso a informação recebida tenha o valor decimal 101 imprime-se na tela a expressão: Movimento Sentido Anti-Horário. Posteriormente modifica-se a porta P41 em seu nível lógico de 0 para 1 o que proporciona nesta porta mudança de 0Volts para 5 Volts, isto aciona o circuito de controle fazendo o móvel girar a roda dianteira para a esquerda e aguardar nesta posição até que um ângulo seja coletado para completar o movimento. A figura 3.13 ilustra este passo.

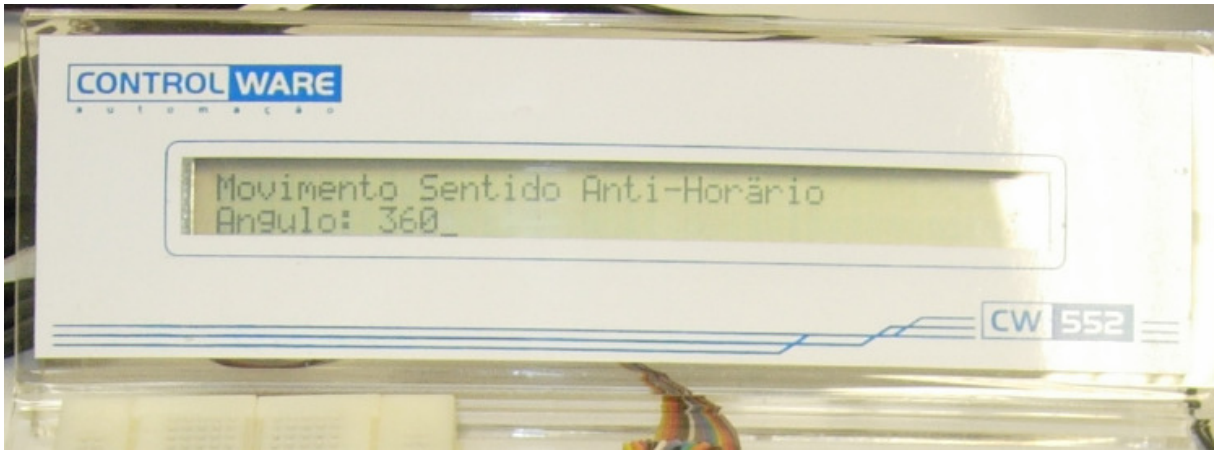


Figura 3.13 – Movimento Sentido Anti-Horário

Em outra opção, a informação recebida tendo o valor decimal 100 imprime-se na tela a expressão: Movimento Sentido Horário. Posteriormente modifica-se a porta P43 em seu nível lógico de 0 para 1 o que proporciona nesta porta mudança de 0 Volts para 5 Volts, isto aciona o circuito de controle fazendo o móvel girar a roda dianteira para a direita e aguardar nesta posição até que um ângulo seja coletado para completar o movimento. A figura 3.14 retrata esta situação.

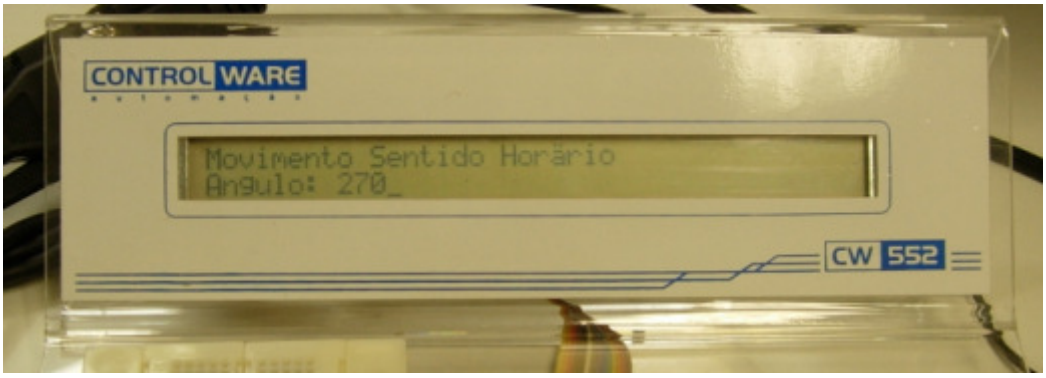


Figura 3.14 – Movimento Sentido Horário

Neste ponto um ângulo poderá ser enviado para o microcontrolador, que, ao recebê-lo, imprime na tela o ângulo recebido e aciona a porta P42, modificando seu nível lógico de 0 para 1 o que proporciona neta porta mudança de 0 Volts para 5 Volts, acionando o circuito de controle que ligará o motor do móvel a fim de realizar o movimento, seja frontal, esquerda ou direita, pela quantidade de tempo necessário para perfazer o movimento solicitado que é baseado no ângulo recebido.

Depois de atingido o objetivo, retorna-se o nível lógico das portas P41, P42 e P43 para 0, desativando qualquer movimento e aguarda-se um novo movimento.

3.4.3 – Código Fonte – Microcontrolador

Parte da programação realizada para o microcontrolador foi retirada do material do próprio Kit CW552, em que explica a utilização do display e das interfaces, como a serial utilizada.

O programa é inicializado com a definição de variáveis e a inserção de inserção de bibliotecas necessárias.

```
#include<8051.h>
#include<at89s8252.h>

#define linha1 0x80
```

```
#define linha2 0xc0

sbit at 0xC1 P41; //Esquerdo
sbit at 0xC2 P42; //Frontal
sbit at 0xC3 P43; //Direito

xdata at 0x3801 unsigned char Lcd_dado;
xdata at 0x3800 unsigned char Lcd_cont;
```

As bibliotecas 8051 e at89s8252 trazem consigo parâmetros necessários para algumas funções utilizadas na programação.

A definição de linha1 é a referência ao campo da memória que associa a primeira linha do display. O mesmo se dá com a linha 2.

A utilização de sbit significa tipo de dado e class de armazenamento, e é usado para designar os bits especiais nos registradores especiais do 8051. os endereços referenciados são os referentes à porta P41, P42 e P43. Deste modo, durante o programa não é necessário inserir o endereço de registro de cada porta, apenas o nome definido e que logicamente facilita a identificação.

A declaração das variáveis utilizando xdata são alocadas na memória RAM externa e é o campo que será utilizado durante a execução do programa para a impressão de dados no display.

Das funções disponíveis para utilização uma teve de sofrer modificação para atender às necessidades do projeto. Nas instruções do kit havia a impressão na tela de números inteiros apenas entre 0 e 99, ou seja, com unidade e dezena. Porém este projeto visa tratar dados entre 0 e 360, sendo os ângulos recebidos. Para tal uma versão da função disponível foi modificada e abaixo segue uma explicação do seu conteúdo.

```
void lcd_int(unsigned int a)
{
  a=a%361;
  wr_lcd(48+a/100);
  wr_lcd(48+((a%100)/10));
  wr_lcd(48+((a%100)%10));
}
```

Esta função não por objetivo retornar nenhuma informação, apenas imprimir na tela do display dados. A impressão no display é feita caractere a caractere, seja de uma frase ou de um número inteiro. No caso de um número inteiro, matematicamente extrai-se, unidade, dezena e centena e envia para a função wr_lcd que tem por objetivo imprimir na tela o caractere recebido.

Para extrair a centena é feito uma operação matemática conforme segue na expressão 3.1:

$$48+\left(\frac{dadorecebido}{100}\right), \tag{3.1}$$

Neste caso, recebido, por exemplo, 340, divide-se por 100, têm-se como resultado 3 e resto da divisão igual a 40. O resto da divisão é desconsiderada, resta o número 3 que é exatamente o valor da centena. Como a função wr_lcd imprime apenas caracteres na tela, soma-se 48 a este valor, onde têm-se o correspondente ASCII para o número 3, que é 51 em decimal. Assim é possível imprimir qualquer valor entre 0 e 9.

Para a dezena executa-se a operação 3.2, onde % é a operação que coleta o resto da divisão entre dado recebido e 100:

$$48+\left(\frac{dadorecebido\%100}{10}\right), \tag{3.2}$$

Neste caso, extrai-se a dezena do dado recebido, semelhante ao exemplo anterior, caso seja 340 o resto da divisão entre 340 e 100 é igual a 40 a divisão de 40 por 10 é igual a 4, que é exatamente o algarismo da dezena. Somado a 48 têm-se o representante ASCII para 4, que é 52 decimal.

Para a unidade o processo é semelhante e segue a representação matemática abaixo:

$$48 + ((dadorecebido \% 100) \% 10), \tag{3.3}$$

Nesta operação, retira-se apenas a unidade do número recebido. Caso seja 340, o resto da divisão entre 340 e 100 é 40 e o resto da divisão entre 40 e 10 é 0, que é exatamente o algarismo da unidade. Somado a 48 têm-se o representante ASCII para 0 que é o próprio 48 decimal.

Função Tempo

A função abaixo tem por objetivo a geração de um tempo de espera compatível com um movimento, ou seja, quando chamada esta função não executa uma nenhuma operação, apenas contadores que aguardam um tempo relativo à um movimento, como por exemplo o movimento de 10º. O exemplo representa apenas uma parte da função, que está em sua totalidade no Apêndice II.

```
void tempo (int i)
{
  unsigned int j, k, l;
  j = i*3;
  if (i == 10) {
  for (k=0; k<=j; k++)
  {
    for (l=0; l<=2000; l++);  }  }
  if (i == 20) {
  for (k=0; k<=j; k++)
  {
    for (l=0; l<=1400; l++); } }
```

Função recebeDado

Nesta função é efetuada a consulta constante no buffer da porta serial para verificar se um novo dado chegou na porta. Quando há um dado na porta é feita uma cópia do dado na variável dadoRecebido que será tratada no programa principal. A função abaixo é a utilizada nesta etapa.

```
void recebeDado()
{
// Quando RI = 1 a recepção chegou ao fim
while(!RI);
// Salva o que foi recebido na variável dadoRecebido
dadoRecebido=SBUF;
RI = 0;
```

Função main

A função principal inicializa as configurações da porta serial e, após impressão de expressões pré-escritas no display, entra em um looping contínuo que sempre ficará lendo o buffer da porta serial.

Havendo dado na porta recebe a informação na variável dadoRecebido e tratará a informação para efetuar o controle do móvel.

Quando recebido o valor 0 significa a intenção em movimento frontal, deste modo, imprime-se no display *Movimento Sentido Frontal* e modifica-se o sinal lógico da porta P42 de 0 para 1, em seguida chama-se a função tempo que contará o tempo necessário para efetuar um movimento frontal. Após o movimento, muda-se o sinal lógico da porta P42 de 1 para 0, retornando o móvel para a posição inerte.

Caso o dado recebido seja o valor 101 decimal ou a letra ‘e’ no padrão ASCII, significa a intenção no movimento anti-horário. Para este caso, imprime-se no display *Movimento Sentido Anti-Horário* e modifica-se o sinal lógico da porta P41 de

0 para 1 o que acionará o circuito de controle para girar as rodas frontais para a esquerda. Caso o dado recebido seja o valor 100 decimal ou a letra ‘d’ no padrão ASCII, significa a intenção no movimento horário e é modificado a porta lógica P43 de 0 para 1 acionando o circuito de controle para girar as rodas frontais para a direita.

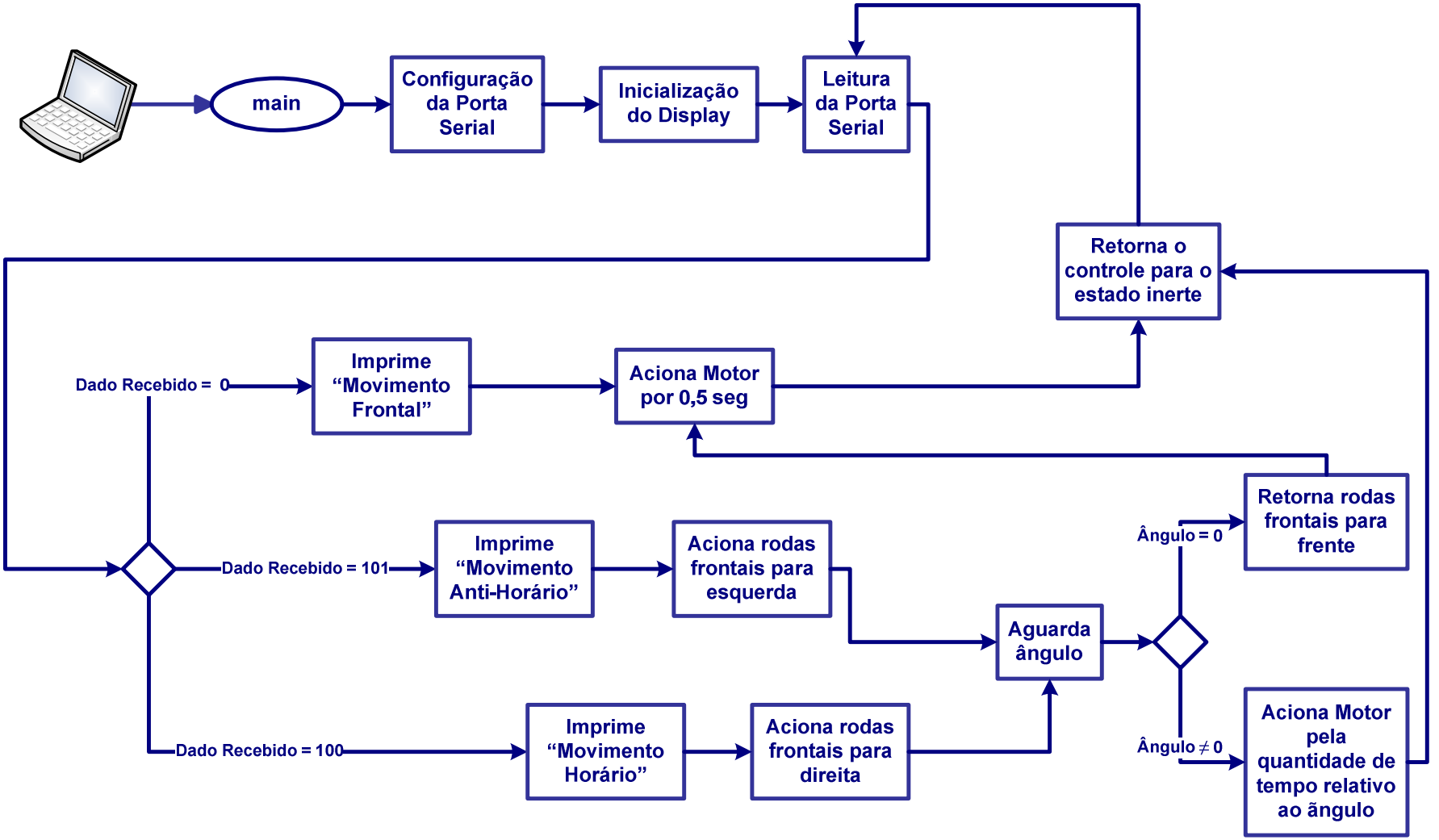
Após esta etapa aguarda-se a informação do ângulo que deverá ser a variável de controle do movimento. Uma vez recebido o ângulo, se este for o valor equivalente a 0, novamente é direcionado para a etapa da programação que fará o movimento frontal, deste modo muda-se o sinal lógico da porta P41 ou P43 de 1 para 0, uma vez que não haverá movimento frontal e modifica-se a porta P42 de 0 para 1, acionando o circuito de controle que efetuará o movimento frontal pelo tempo de um movimento.

Porém se o ângulo recebido for diferente de 0, então o ângulo será impresso no display e será modificado a porta lógica P42 de 0 para 1 e o movimento será acionado o circuito de controle para ligar o motor do carrinho a fim de que o movimento seja feito. O motor permanece ligado até que a condição da função tempo seja preenchida. Neste caso, a condição está relacionada com a variável enviada para a função tempo, que é a própria variável dadoRecebido. Segue um exemplo abaixo para explicação mais detalhada.

Digita-se na Interface Computacional o movimento anti-horário, neste momento o microcontrolador recebe a informação 101 e aciona o controle para girar as rodas dianteiras para a esquerda. A próxima tela coletará o ângulo, uma vez digitado 100, representando 100º, o microcontrolador receberá esta informação e acionará o motor do móvel o que fará com que o mesmo se mova. A informação de

100º é enviada para a função tempo que contará o tempo necessário, realizando o movimento aproximado a 100º no sentido anti-horário.

3.4.4 – Fluxograma da Programação do Microcontrolador



Fluxograma 3.2 – Fluxograma Programação Microcontrolador

A conexão entre o computador e microcontrolador ocorre conforme a ilustração 3.15.

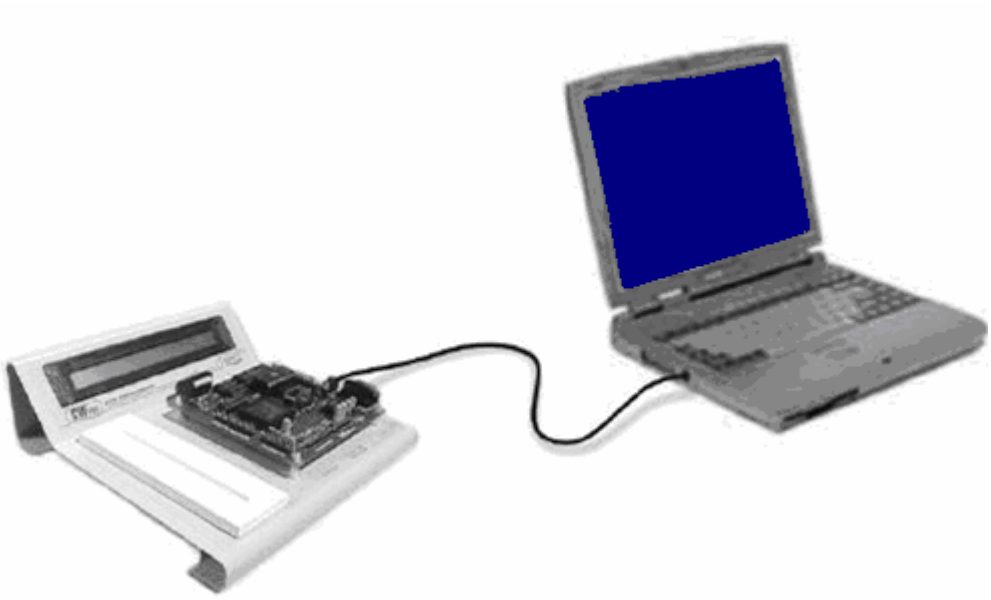


Figura 3.15 – Comunicação Computdor – Kit

3.5 – CIRCUITO ELÉTRICO – CONTROLE

Conforme já abordado, um dos objetivos deste projeto é a produção de um ambiente de teste para o tema abordado. Para tal feito, adquiriu-se um carrinho de controle remoto, não sendo escopo do projeto a produção de um.

Para efetuar o controle do móvel à partir das informações inseridas no computador, pela Interface Computacional, foi aberto o controle remoto do carrinho e desenvolvido um circuito com a função de acionar os contatos responsáveis movimentação do carrinho, movimentos esquerda, direita e frontal.

Abaixo segue a ilustração 3.16 representando o controle remoto do carrinho, com a localização dos respectivos contatos referentes a cada movimento.

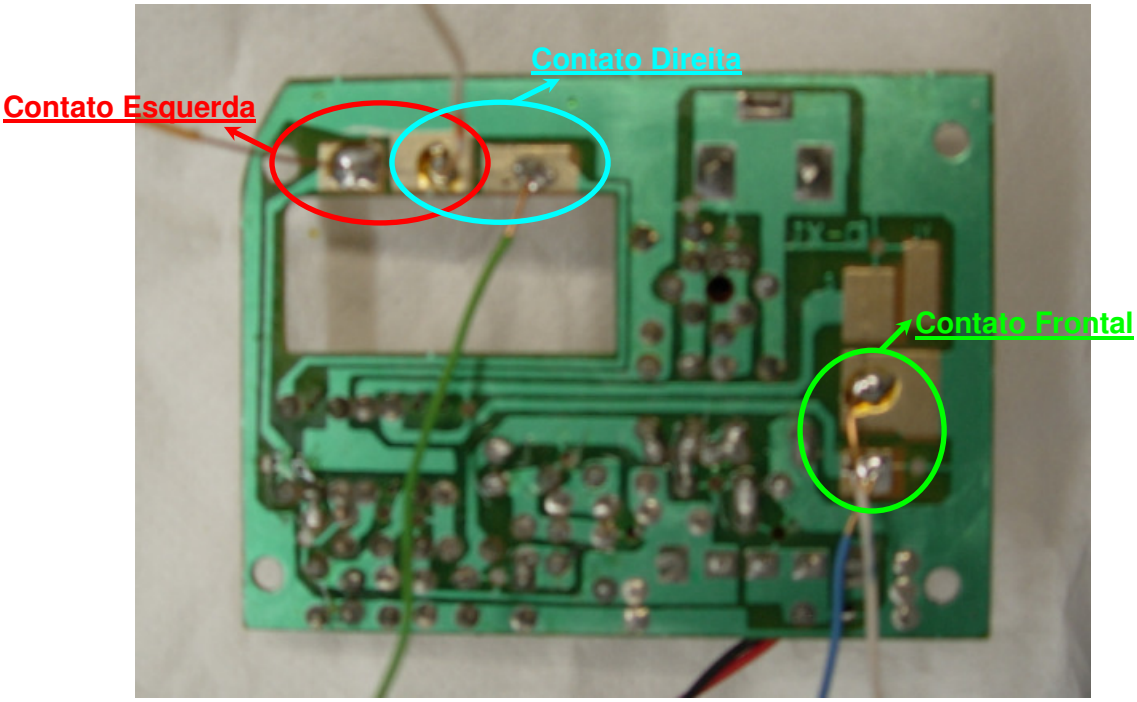


Figura 3.16 – Controle Remoto do Carrinho – Visualização dos Contatos

O estudo no controle remoto do carrinho comprovou que a movimentação do móvel adquirido dá-se pelo contato físico entre cada uma das pontas destacadas na figura 3.16, ou seja, para efetuar o movimento frontal do carrinho basta realizar um curto nos dois contatos destacados em verde. O mesmo se dá com o movimento das rodas frontais, para esquerda os contatos destacado em vermelho, para direita os contatos destacado em azul.

Para conseguir este acionamento de forma automática, pensou-se numa solução baseada em relés, que são nada mais nada menos do que chaves mecânicas com acionamento eletrônico, porém o uso de relés poderia causar atrasos na resposta ou mau funcionamento juntamente com o controle remoto do carrinho.

Por isso, com estudos mais profundos no assunto chegou-se a uma solução simples e com resposta mais rápida, por chave eletrônica. Adquiriu-se optoacopladores modelo 4N25, transistores BC548B e resistores.

3.5.1 – Funcionamento dos Dispositivos

Um optoacoplador é um circuito integrado em uma cápsula com dois dispositivos em seu conteúdo, um diodo emissor de luz (fotodiodo) e um transistor sensível à luz (foto-transistor). São componentes que possibilitam a transferência de um sinal de controle ou mesmo de um sinal que carrega uma informação, de um circuito para outro, sem a necessidade de acoplamento elétrico. O sinal é transferido por um feixe de luz produzido por um emissor LED e recebido por um sensor, que neste caso é um transistor.

Abaixo têm-se a estrutura interna e o símbolo usado pelo optoacoplador 4N25, o qual faz uso de um LED emissor de infravermelho e um foto-transistor bipolar como sensor.

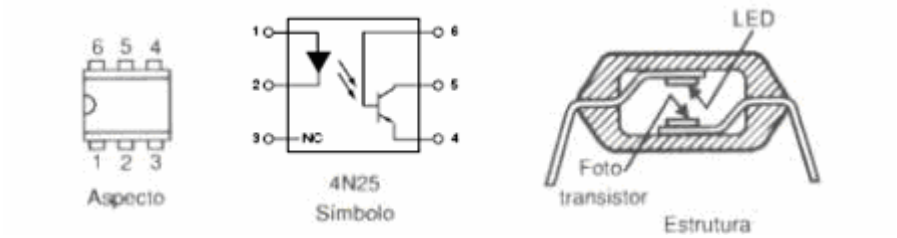


Figura 3.17 – Optoacoplador 4n25

Esta solução é particularmente interessante, porque neste projeto eletrônico está trabalhando-se com 2 fontes de tensão, a fonte de tensão do microcontrolador, como parte do circuito de controle e ativação e a fonte de tensão do controle remoto que serve para enviar os sinais de radiofreqüência para o móvel.

Estas duas fontes de tensão têm valores diferentes, sendo a do microcontrolador 5 V e a do controle remoto 9 V. Isto poderia, eventualmente, causar o que chamamos de retorno de carga, onde circuitos com fontes de tensão diferentes diretamente interligados, poderia gerar uma corrente de retorno no circuito de menor tensão e provocar algum dano, como a perda de componentes e, tratando-se de componentes eletrônicos, estamos falando de dispositivos sensíveis.

Deste modo, o uso de um dispositivo optoacoplador é uma idéia apropriada para o caso pois, além do isolamento elétrico entre os dois circuitos, este servirá como chave eletrônica para o acionamento dos contatos que realizam o movimento do carrinho.

O transistor BC548B é um transistor amplificador NPN, onde a saturação da base implica em passagem de corrente do coletor para o emissor. Este dispositivo, em conjunto com resistores, foi utilizado para acionar o fotodiodo, a partir do sinal de controle emitido pelas portas lógicas do microcontrolador.

Abaixo segue uma representação ilustrativa deste transistor.

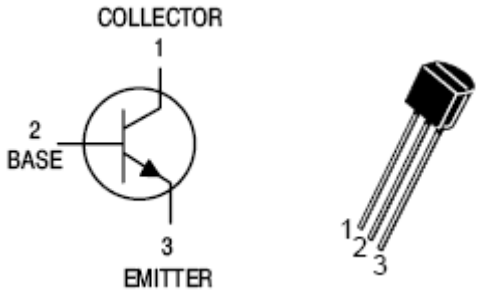


Figura 3.18 – Transistor bc548b

3.5.2 – Circuito Eletrônico Montado

O circuito composto pelos dispositivos citados na sessão anterior, em seu macrofuncionamento, caracteriza-se pela acoplagem de cada porta de controle do microcontrolador à base de um transistor BC548B. No coletor conectou-se uma fonte de alimentação de 5 V, proveniente do próprio Kit CW552, conectado serialmente a uma resistência de 212 ohms. No emissor foi conectado uma resistência de 470 ohms. Deste modo, quando houver o envio de um sinal de controle a partir do microcontrolador, saturando a base, há uma passagem de corrente gerada pela fonte conectada no coletor passando pelas resistências de 212 ohms e 470 ohms. Esta corrente é suficiente para ativar o fotodiodo localizado no optoacoplador, excitando o foto-transistor que tem em seu coletor e emissor conectados os contatos do controle remoto, seja para esquerda, direita ou frontal.

Assim foi necessário a montagem de 3 sistemas como o descrito acima, um para esquerda, outro para direita e um terceiro para o frontal.

3.5.3 – Esquema Elétrico – Controle

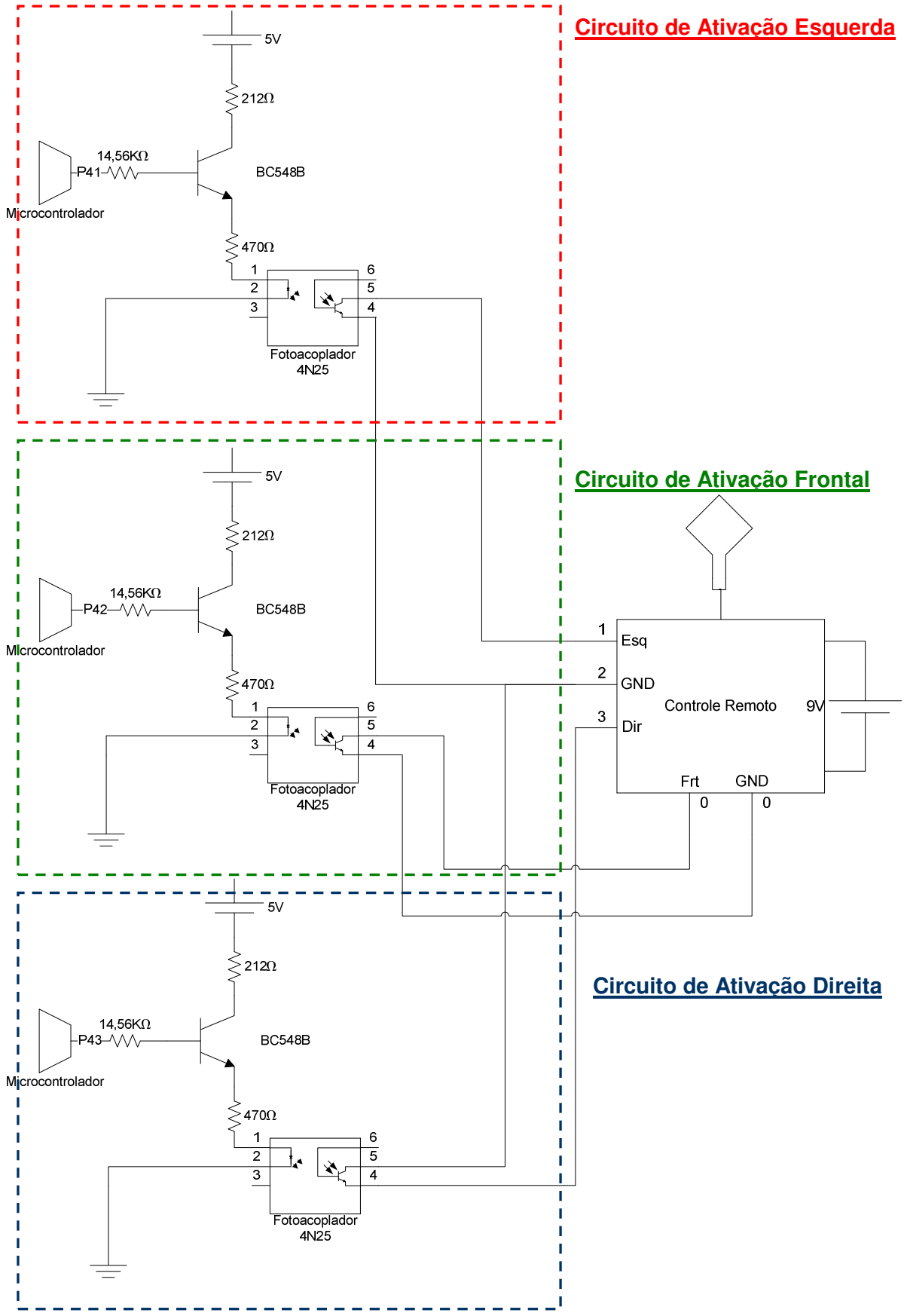


Figura 3.19 – Esquema Elétrico – Circuito de Controle

3.5.4 – Circuito Elétrico em Detalhes

Inicialmente pensou-se em utilizar a porta do microcontrolador disponível no kit CW552, porém em estudo no kit observou-se que a corrente proveniente desta porta é apenas de 0,5 mA a 5 V, não sendo suficiente para ativar o fotodiodo. Em outras palavras, a conexão desta porta diretamente em um dispositivo como um diodo, poderia resultar na perda da porta ou de algum outro dispositivo do kit, devido à alta corrente necessária para o funcionamento do diodo.

Sendo assim, foi necessário usar esta porta apenas como ativação do transistor BC548B. De acordo com a especificação técnica deste componente, a saturação da base ocorre na escala de 0,5 mA que é exatamente a corrente máxima provida na saída do microcontrolador.

Objetivando menor queda possível na tensão de saída da porta e geração de corrente necessária para a saturação da base do transistor foi utilizado um resistor de 14,56 kΩ. Esta associação proporciona uma corrente de 0,495 mA na base do transistor BC548B, suficiente para que houvesse saturação do mesmo.

Deste modo foi possível manter um nível de tensão razoavelmente próximo dos 5V. Em testes a medição mostrou 4,94 V. Não havendo uma superutilização do kit.

A alimentação do fotodiodo localizado no optoacoplador 4N25 é proveniente de uma fonte de tensão de 5 V ligada em série com uma resistência de 212Ω e, consequentemente, ao coletor do transistor BC548B. O emissor está conectado a uma resistência de 470 Ω e a resistência em série ao fotodiodo. Deste modo, uma vez estando a base saturada uma corrente de 4,95 mA ativará o diodo.

Esta corrente é suficiente para ativação da base que, segundo especificação, é de no máximo 10 mA a 6 V.

Na figura 3.20 segue um modelo do circuito conforme gerado no software Circuit Maker Student Version. Essa é uma versão freeware que possibilita a montagem e a simulação de circuitos elétricos e eletrônicos. É possível encontrar a versão de estudante para download no endereço abaixo.

http://superdownloads.uol.com.br/download/i9609.htm.

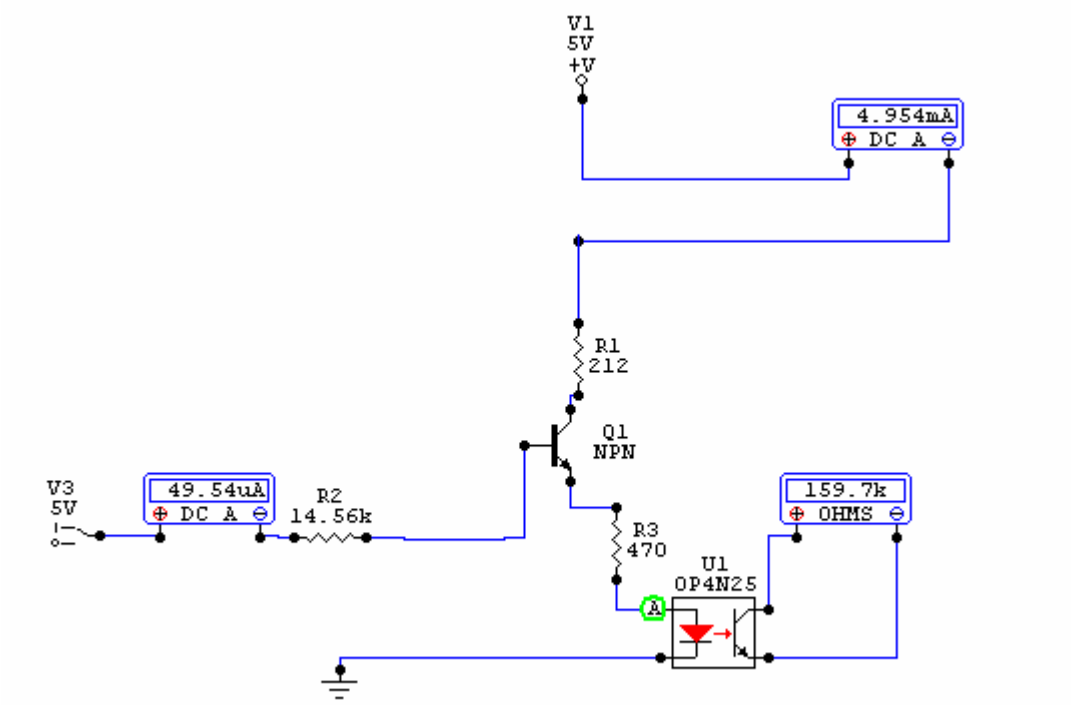


Figura 3.20 – Esquema Elétrico – Circuit Maker

Uma vez o fotodiodo emitindo luz suficiente, é saturada a base do fototransistor, localizado no chip 4N25.

Funcionamento do circuito de ativação

Os transistores funcionam em corte e em saturação (aberto e fechado, respectivamente). A corrente na base do transistor controla o fechamento do contato emissor-coletor. Se a corrente na base for zero, a corrente no coletor será próxima de zero e o transistor estará em corte. Se a corrente na base for maior ou igual à corrente de saturação, a corrente no coletor será máxima e o transistor estará em saturação. [Bertoli, 2000].

Existem dois tipos de saturação. Saturação fraca significa que o transistor está levemente saturado, ou seja, a corrente na base é suficiente apenas para operar o transistor, não levando em conta as variações de ganho de corrente βcc . Saturação forte significa que existe uma corrente na base suficiente para saturar o transistor para todas as variações de ganho de corrente βcc. Para garantir uma saturação forte, é necessário que a corrente na base seja um décimo da corrente no coletor ($\beta cc = 10$). [Bertoli, 2000].

No projeto, a corrente na base do transistor é aproximadamente 0,49 mA e a corrente no coletor é aproximadamente 4,9 mA, ou seja, o transistor está funcionando com saturação forte ($\beta cc = 10$). Isso garantirá saturação para todas as variações de ganho de corrente.

Do coletor para o emissor do transistor NPN localizado internamente ao 4N25 foi conectado o contato responsável por cada movimento no controle remoto do carrinho. Assim, quando o fototransistor for saturado sua base, haverá um curto entre o coletor e o emissor, caracterizando-se no fechamento do contato que, em termos práticos, representa um comando no controle remoto.

O circuito foi montado com todos os dispositivos soldados em uma placa de fenolite e fixado em uma placa de madeira. A ilustração 3.21 mostrará a montagem em detalhes.

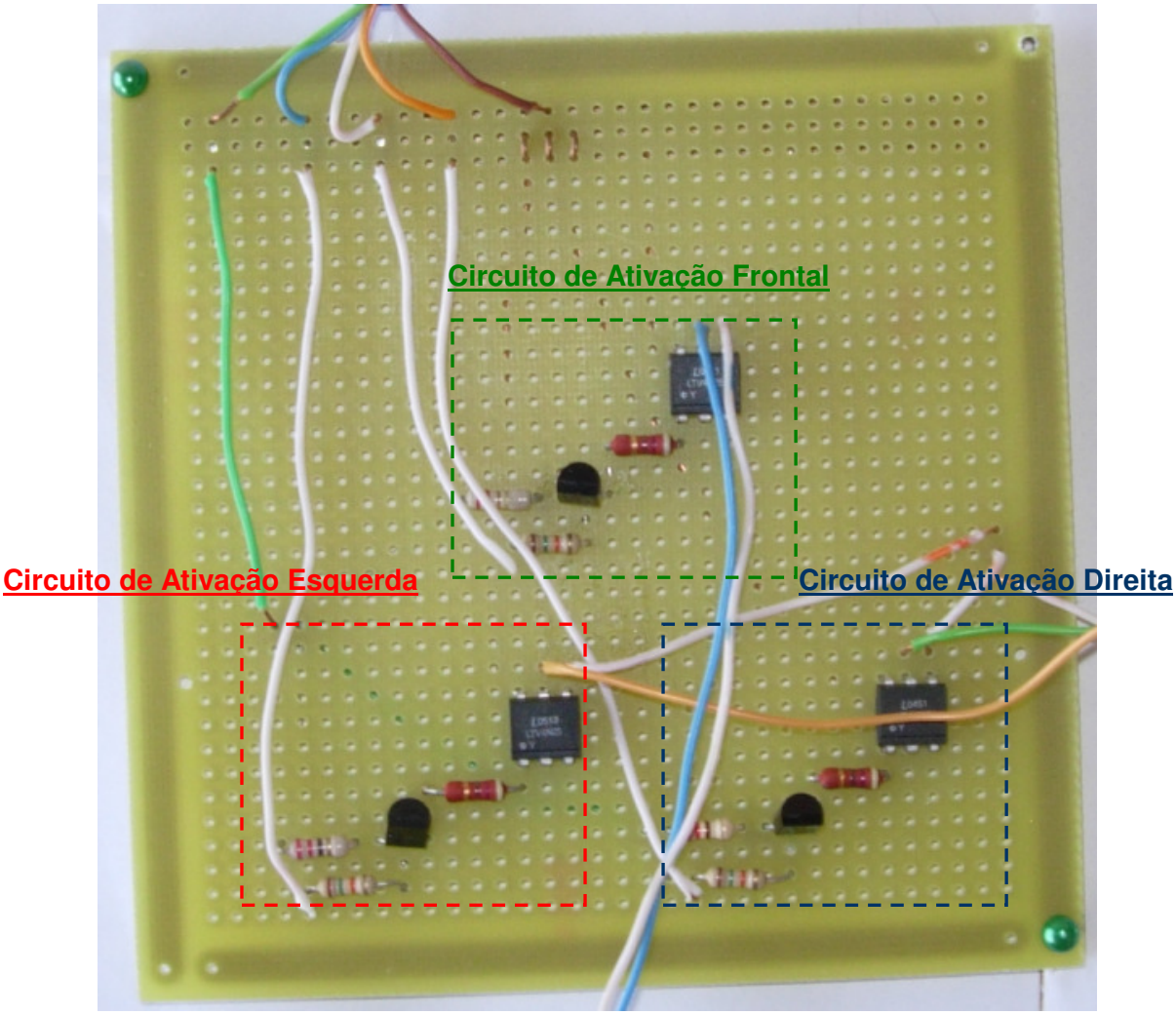


Figura 3.21 – Ilustração do Circuito de Controle

As conexões com o controle está representado pelos fios maiores saindo da placa de fenolite, a ilustração 3.22 mostrará toda a montagem.

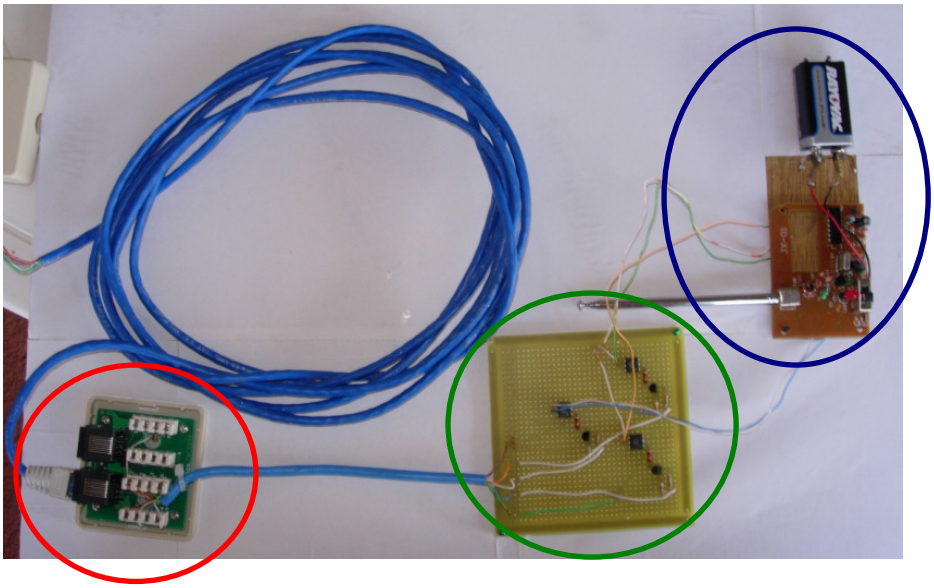


Figura 3.22 – Circuito Completo

A ilustração 3.22 mostra o controle remoto do carrinho ligado à bateria de 9 volts representado no círculo em azul. No círculo verde têm-se o circuito de controle montado na placa de fenolite, que aciona os contatos do controle remoto do carrinho. Circulado em vermelho localiza-se um conector RJ-45 fêmea que foi utilizado para fazer a conexão com o kit CW552, facilitando a montagem na hora dos testes, não havendo uma quantidade desnecessária de fios à mostra no ambiente.

A ilustração 3.23 mostra a montagem do conector RJ-45 fêmea.

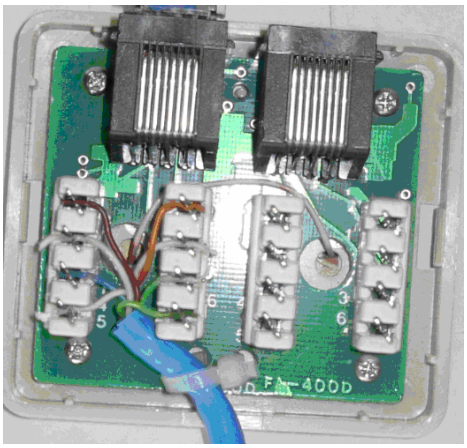
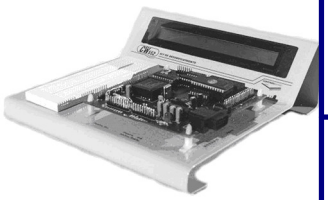
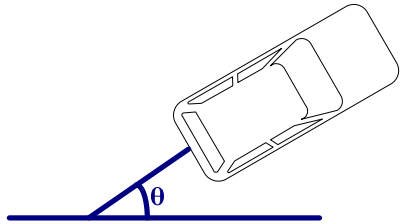


Figura 3.23 – Montagem do Conector RJ-45 Fêmea Conexão com Kit cw552

3.5.5 – Fluxograma Circuito Eletrônico





Fluxograma 3.3 – Fluxograma Circuito Eletrônico

3.6 – RESULTADOS

3.6.1 – Ambiente de teste

Para a realização de testes deste ambiente foi produzido em AutoCad 2004, software com precisão para desenhos estruturais, um transferidor para que fosse possível realizar a medição dos resultados do ambiente. Em AutoCad foi possível criar com precisão as linhas dispersas em sua devida angulação Abaixo segue o transferidor produzido, com as possíveis escalas.

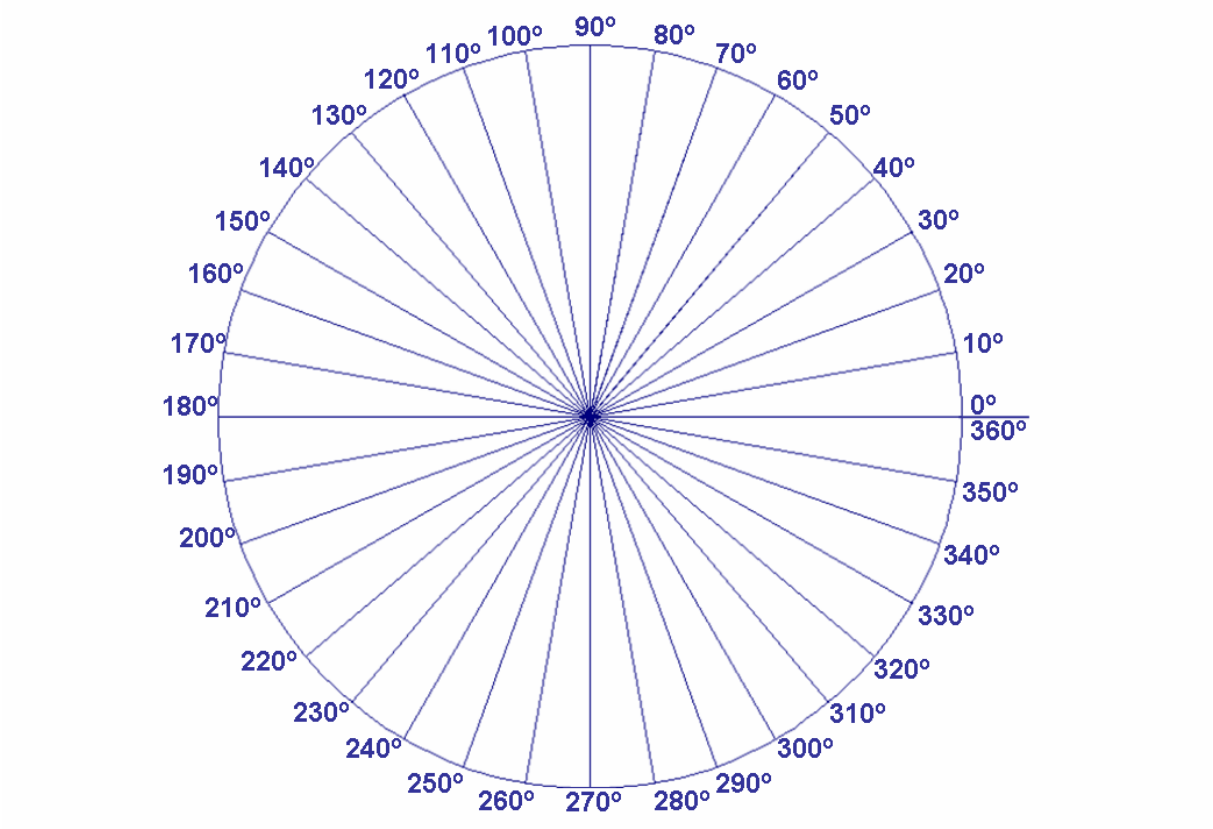


Figura 3.24 – Transferidor para Testes

A fim de comparar o resultado final, posição final do móvel, foi afixado no carrinho uma haste metálica rígida na parte traseira, deste modo, após o movimento é possível comparar sua posição inicial com sua posição final, apenas aproximando o transferidor produzido. Conforme ilustração 3.25

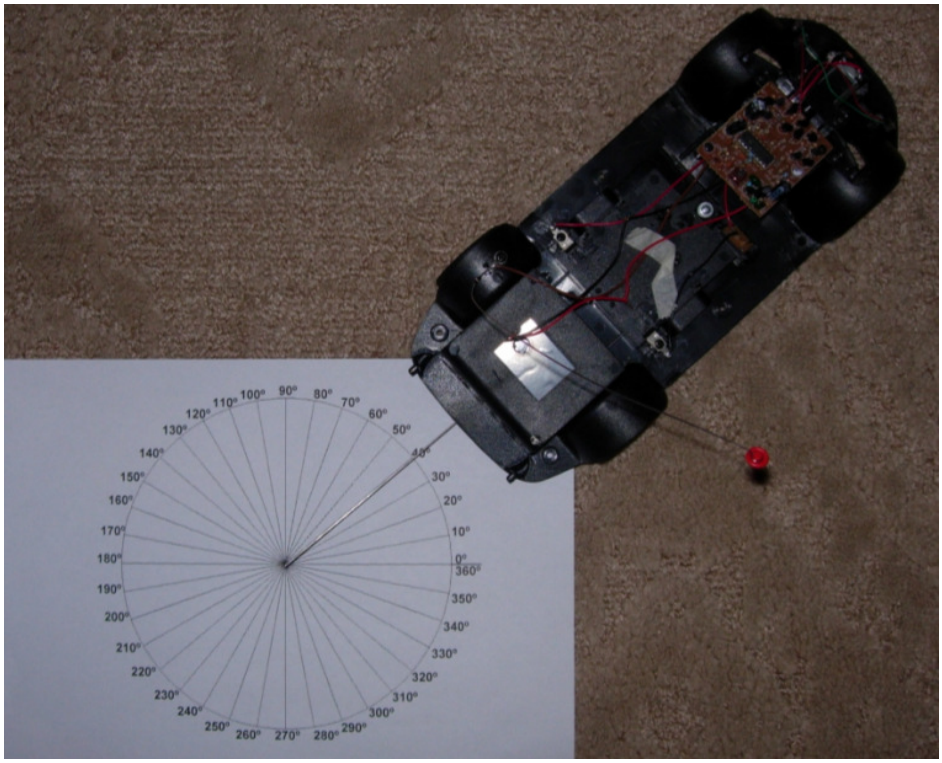


Figura 3.25 – Sistema de Medição

3.6.2 – Resultados do Ambiente

Como resultados foi possível observar, após uma série de amostras, uma faixa de erro entre 5% e 20%. Observou-se ainda que para graus menores o erro é maior devido a necessidade de maior precisão no movimento. A tabela 3.4 apresenta os dados coletados do ambiente.

Sentido	Ângulo Desejado	Ângulo Coletado
Horário	10º	9º - 12º
Horário	20º	18º - 21º
Horário	50º	47º - 52º
Horário	90º	88º - 93º
Horário	120º	117º - 124 º
Horário	180º	175º - 187º
Horário	270º	264º - 280º
Horário	300º	290º - 315º
Horário	360º	346º - 374º

Anti-Horário	10º	8º - 13º
Anti-Horário	20º	18º - 22º
Anti-Horário	50º	45º - 55º
Anti-Horário	90º	83º - 96º
Anti-Horário	120º	114º - 129 º
Anti-Horário	180º	172º - 190º
Anti-Horário	270º	258º - 290º
Anti-Horário	300º	290º - 312º
Anti-Horário	360º	343º - 380º

Tabela 3.4 – Tabela de Resultados Coletados

Conforme a tabela acima é possível observar uma variação no resultado, apresentando uma maior faixa de erro nos movimentos Anti-Horários. Este resultado foi observado em piso de ardósia.

O ambiente apresentou certa instabilidade na freqüência dos resultados devido ao móvel não ser preparado para este tipo de produção técnica.

Conforme já especificado, este projeto baseou-se em sistema de controle em malha aberta, de acordo com o tempo previsto para apresentação dos resultados. Além do mais, o móvel utilizado não foi produzido com este fim, sendo apenas adaptado. Sendo assim, erros apresentados nesta faixa são aceitáveis.

No entanto, sugere-se a continuação deste projeto com sistema em malha fechada, tratando o erro como uma variável de entrada, acertando com maior precisão o objetivo. Como vantagem este projeto, já lançou base para esta nova produção técnica.

O ambiente de testes foi feito em uma superfície razoavelmente lisa e com coeficiente de atrito entre médio e baixo. O material do piso era ardósia.

Observou-se ainda que a carga existente na bateria do móvel influencia diretamente no resultado final, podendo aumentar o erro ou diminuí-lo.

Os resultados obtidos coletados em testes realizados em solo do tipo ardósia e com a bateria, inicialmente, totalmente carregada.

Não foram coletados dados em outros ambientes ou outros tipos de solo.

Capítulo 4 – Considerações Finais

4.1 – CONCLUSÃO

Foi possível aproximar-se do resultado proposto, sendo o controle do móvel a partir de uma variável ângulo geométrico do ambiente, mostrando a possibilidade do controle de dispositivos móveis de velocidade constante a partir desta variável.

Os resultados, porém, foram limitados no sucesso por problemas listados abaixo:

- Não houve modelagem do móvel, por não ser escopo do projeto; Sendo assim, observou-se uma variação na angulação da roda frontal de forma aleatória;
- Ainda quanto ao móvel, observou-se uma não linearidade no movimento angular, isto dificultou o resultado final, pois não havendo linearidade o mapeamento do movimento fica com relativa deficiência, baseado no método empregado;
- Não houve um tratamento estatístico dos dados coletados.

Mesmo com os resultados acima destacados, a conclusão final é bastante satisfatória para possíveis continuações do projeto.

4.2 – PROBLEMAS ENCONTRADOS

A produção técnica de um projeto exige muita dedicação e empenho, como qualquer atividade profissional de pesquisa, é natural encontrar problemas, alguns

são vencidos com sucesso, outros parcialmente e ainda alguns não são possíveis de ser solucionados, por serem problemas que necessitem de uma outra abordagem.

Outra dificuldade encontrada no desenvolvimento do projeto, foi a adaptação do móvel ao ambiente, pois, conforme já mencionado, não era escopo deste projeto a modelagem e produção de um móvel. Isto tornou-se particularmente um desafio, pois adaptar o móvel adquirido constituiu-se uma tarefa árdua e não totalmente bem sucedida, daí a sugestão em um projeto que execute a produção de um móvel próprio para ambientes como o dimensionado neste projeto. Isto poderia ainda ser reduzido com a inserção do conceito de malha fechada, o que sugere-se para posterior projeto.

Pôde-se ainda extrair do projeto a dificuldade em adquirir componentes eletrônicos precisos, dado seu alto custo e este projeto não ter nenhuma espécie de patrocínio, já que seu teor é puramente educacional.

4.3 – PROJETOS FUTUROS

Como projetos futuros, sugere-se a produção deste projeto com a implementação em malha fechada, inserindo-se sensores no móvel para medição do resultado com o objetivo. A partir desta medição é possível reajustar o movimento, aproximando-se do objetivo com maior precisão.

O controle, mesmo que em malha aberta, poderia ser incrementado com tratamento estatístico dos dados, ou a utilização de técnicas mais apuradas de controle, como lógica fuzzy, redes neurais.

Outra sugestão é a produção de um móvel com as características para o movimento controlando a variável ângulo. Este móvel pode ser produzido reduzindo

a possibilidade de erros pela utilização de componentes precisos. Poderia ser um móvel já controlado por microcontrolador em sua arquitetura e com interface serial embutida.

A partir destes projetos é possível a produção de soluções com aplicações reais e práticas para o dia-a-dia.

Referência Bibliográfica

1. [BENCHIMOL] Benchimol, Augusto, Uma breve História da Eletrônica, Interciência, 1995.

2. [BERTOLI] Bertoli, Roberto Ângelo, Eletrônica. São Paulo: Colégio Técnico de Campinas, 2000.

3. [CARVALHO] Carvalho, J. L. Martins de, Sistemas de Controle Automático, LTC, 2000

4. ControlWare Automação. Manual do Usuário do Kit de Desenvolvimento CW552. Out. 2001.

5. Free Serial Port Monitor [Home Page]. 2006. Disponível em: <http://www.serial-port-monitor.com>. Acesso em: 05 set. 2006.

6. [GEORGINI], Georgini, Marcelo, Automação Aplicada: Descrição e Implementação de Sistemas Seqüenciais com PLCs, 5ª Edição, Érica, 2003.

7. [GIL] Gil, Antonio Carlos. *Métodos e Técnicas de Pesquisa Social*. 5. ed, São Paulo: Atlas, 1999.

8. [LEVY] Levy, Pierre, A Máquina Universo: Criação, Cognição e Cultura Informática, Porto Alegre: Artmed, 1998

9. [MALVINO] Malvino, Albert Paul, Eletrônica, Makron Books, 1997

10. [MIZRAHI] Mizrahi, Viviane Victorine, Treinamento em linguagem C curso completo, módulo 1, Makron Books, 1990

11. [MIZRAHI] Mizrahi, Viviane Victorine, Treinamento em linguagem C curso completo, módulo 2, Makron Books, 1995

12. [MIZRAHI] Mizrahi, Viviane Victorine, Treinamento em linguagem C, Módulo Professional, Ed. Makron Books, 1993.

13. [OGATA] Ogata, Katsuhiko, Engenharia de Controle Moderno, 4. ed. Pearson, 2003

14. Rogercom [Home Page]. 2006. Disponível em: <http://www.rogercom.com>. Acesso em: 27 set. 2006.

15. [SANTAELLA] Santaella, Lucia. Comunicação e Pesquisa: Projetos para Mestrado e Doutorado. São Paulo: Hacker Editores, 2001.

16. SDCC – Small Device C Compiler [Home Page]. 2006. Disponível em: <http://sdcc.sourceforge.net>. Acesso em: 25 set. 2006.

17. Superdownloads [Home Page]. – Circuit Maker Student Version [Home Page]. 2006. Disponível em:
<http://superdownloads.uol.com.br/download/i9609.htm>. Acesso em: 27 set. 2006.

18. [bc548b] [Home Page]. Datasheet bc548b disponível em, http://www.chipcatalog.com/ONSemi/BC548B.htm, Acesso em 30 nov. 2006.

19. [4n25] [Home Page]. Datasheet 4n25 disponível em http://info.hobbyengineering.com/specs/Fairchild-4N28-M.pdf#search=%22data%20sheet%204n25%22, Acesso em 30 nov 2006.

Apêndice I – Código Fonte Interface Computacional

```
#include <stdio.h>
#include <conio.h>
#include <dos.h>

#define end_ctrl_50 0x03FD /* Enderecos da UART 8051 */
#define end_dado_50 0x03F8
#define LRC 0x03FB
#define MSB 0x03F9
#define LSB 0x03F8
#define MCR 0x03FC
#define LSR 0x03FD
#define MSR 0x03FE
#define IOR 0x03F8
#define IIR 0x03FA
#define IER 0x03F9

// inicializacao da 8051

void inic8051() {
/* DLAB = 1, programa velocidade em MSB e LSB */
outportb(LRC, 0x80);
/* 1 stop bit, sem paridade */
outportb(MSB, 0x00);
/* 00H 0CH == 9600 bps */
outportb(LSB, 0x0C);
/* 8 bits de dados, 1 stop bit, sem paridade */
outportb(LRC, 0x07);
}

// reseta a 8051
void reseta8051() {
inportb(LSR);
inportb(MSR);
inportb(IOR);
inportb(IIR);
}

// desabilita interrupcoes da 8051
void disable8051() {
outportb(IER, 0x00);
}

// transmite um caracter
void transm_dado(unsigned char caracter) {
while (!(32 & inportb(end_ctrl_50)) == 32);
outportb(end_dado_50, caracter);
}

// Função para criar cabeçalho com identificação do programa
void fApresentacao (void) {
     clrscr();
```

```
	printf("\n\t
ÉÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ»\n");
	printf("\t Centro Universitario de Brasilia - UniCEUB\n");
	printf("\t Faculdade de Ciencias Exatas e Tecnologia - FAET\n");
	printf("\t Projeto : Controle de um movel a partir de um ângulo \n");
	printf("\t usando microcontrolador. \n");
	printf("\t Aluno Formando em Engenharia de Computacao: \n");
	printf("\t Italo Bruno Freitas de Jesus \n");
	printf("\t
ÈÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ¼\n\n\n");
}

//Função para seleção do sentido que o móvel deverá percorrer
char fSentido() {
	char sentido1;
	do {
	   clrscr();
	   fApresentacao();
	   printf("\n\t\tDigite: \n\t\tE para movimento Anti-horario, \n\t\tD para movimento Horario, ou \n\t\t0 para movimento frontal: ");
	   scanf ("%c", &sentido1);
	} while((sentido1 != '0') && (sentido1 != 'E') && (sentido1 != 'D') && (sentido1 != 'e') && (sentido1 != 'd'));
	if((sentido1 == 'E') || (sentido1 =='e'))
	     sentido1 = 101;
	else if((sentido1 == '0'))
	     sentido1 = 0;
	else if((sentido1 == 'D') || (sentido1 =='d'))
	     sentido1 = 100;
	return(sentido1);
}

//Função para coleta do ângulo que o móvel deverá percorrer
//Sempre em múltiplo de 10
int fVerifica () {
	int angulo2;
	inicio:
	angulo2 = -2;
	while((angulo2 < 0) || (angulo2 > 360)) {
	clrscr();
	fApresentacao();
	printf("\n\n\t\tDigite um numero entre 0 e +360 e multiplo de 10: ");
	scanf ("%d", &angulo2);
	if ((angulo2%10)!=0) {
	  printf("\n\t\tNumero nao eh multiplo de 10. Pressione qualquer tecla.");
	  getch();
	  goto inicio; }
	 }

	return(angulo2);
}
```

```
//Função sem retorno, objetiva imprimir na tela o movimento
percorrido pelo móvel
char fComparanp (char sentido2, int angulo2) {
     if((sentido2 == 'e' || sentido2 == 'E') && angulo2 != 0) {
          printf("\t\tMovimento Sentido Anti-horario.\n"); }
     else if (sentido2 == '0' || angulo2 == 0) {
          printf("\t\tMovimento para frente.\n"); }
     else if ((sentido2 == 'd' || sentido2 == 'D') && angulo2 != 0)
{
          printf("\t\tMovimento Sentido Horario.\n"); };
return(0);
}

//Função que envia a informação para o microcontrolador 8051
fEnviacaracter(char senvia) {
    disable8051();
     transm_dado(senvia); // envia um caracter
          return(0); }

//Função principal
void main() {
       int angulo, angzero, fim = 0;
       char sentido, angenvia;
       textbackground(1);
       textcolor(3);

       inic8051();
       reseta8051();
       disable8051(); /* sem interrupcoes */
       clrscr(); //limpa a tela

while(!(fim==27)) {
       sentido = fSentido(); //atribui à variável sentido a
informação coletada em fSentido
       fEnviacaracter(sentido);   //envia   o   sentido   para   o
microcontrolador
       if(sentido != 0)
            angulo = fVerifica(); //se o sentido é diferente de 0,
coleta o ângulo
        else            {
            angzero = 400; //atribui 400 a angzero, como medida
apenas comparativa a ser utilizado no programa
           angulo = 0; } //se o sentido é igual a 0, o ângulo é
igual a 0
       clrscr();
       fApresentacao();
       printf ("\n\n\t\tO angulo eh de %d graus. \n", angulo);
       fComparanp(sentido, angulo);
       if(angzero != 400) {
          angenvia = angulo/10;
          fEnviacaracter(angenvia); }
       printf("\n\n\t\tPressione ESC para sair ou qualquer tecla
para continuar.");
       angzero = 0;
       fim=getch();       }
}
```

Apêndice II – Código Fonte Microcontrolador

```
#include<8051.h>
#include<at89s8252.h>

#define linha1 0x80
#define linha2 0xc0

sbit at 0xC1 P41; //Esquerda
sbit at 0xC2 P42; //Frontal
sbit at 0xC3 P43; //Direita

xdata at 0x3801 unsigned char Lcd_dado; //Endereco de memoria do
display
xdata at 0x3800 unsigned char Lcd_cont; //Endereco de memoria do
display

// início das rotinas do LCD
void wr_ctr_lcd(unsigned char a)
{
  int i;
  Lcd_cont = a;
  for (i=1;i!=1000;i++);
}

void wr_lcd(unsigned char a)
{
  int i;
  Lcd_dado = a;

  for (i=1;i!=100;i++);
}

void ini_lcd(void)
{
  wr_ctr_lcd(0x38);
  wr_ctr_lcd(0x06);
  wr_ctr_lcd(0x0E);
  wr_ctr_lcd(0x01);
}

void lcd_str(char *s)
{
  do wr_lcd(*s);
  while (*++s);
}

void goto_lcd(unsigned char l, unsigned char c)
{
  unsigned char a;
  if (l==1)
  {
    a = linha1;
  }
```

```
  if (l==2)
  {
    a = linha2;
  }
   wr_ctr_lcd(a+c-1);
}

//imprime números com até 3 dígitos na tela do display
void lcd_int(unsigned int a)
{
  a=a%361;
  wr_lcd(48+a/100);
  wr_lcd(48+((a%100)/10));
  wr_lcd(48+((a%100)%10));
}

// fim das rotinas LCD


//Rotina tempo - Controla tempo que será feito um movimento.

void tempo (int i)
{
  unsigned int j, k, l;
  j = i*3;
  if (i == 10) {
  for (k=0; k<=j; k++)
  {
    for (l=0; l<=2000; l++);
   }
  }

  if (i == 20) {
  for (k=0; k<=j; k++)
  {
    for (l=0; l<=1400; l++);
   }
  }

  else if (i == 30) {
  for (k=0; k<=j; k++)
  {
    for (l=0; l<=1200; l++);
   }
  }
  else if (i == 40) {
  for (k=0; k<=j; k++)
  {
    for (l=0; l<=1180; l++);
   }
  }
  else if (i == 50) {
  for (k=0; k<=j; k++)
  {
    for (l=0; l<=1100; l++);
   }
```

```
}
else if (i == 60) {
for (k=0; k<=j; k++)
{
  for (l=0; l<=990; l++);
 }
}
else if (i == 70) {
for (k=0; k<=j; k++)
{
  for (l=0; l<=940; l++);
 }
}
else if (i == 80) {
for (k=0; k<=j; k++)
{
  for (l=0; l<=925; l++);
 }
}
else if (i == 90) {
for (k=0; k<=j; k++)
{
  for (l=0; l<=915; l++);
 }
}
else if (i == 100 || i == 110) {
for (k=0; k<=j; k++)
{
  for (l=0; l<=875; l++);
 }
}
else if (i == 120) {
for (k=0; k<=j; k++)
{
  for (l=0; l<=865; l++);
 }
}
else if (i == 130) {
for (k=0; k<=j; k++)
{
  for (l=0; l<=855; l++);
 }
}
else if (i >= 140 && i <= 230) {
for (k=0; k<=j; k++)
{
  for (l=0; l<=795; l++);
 }
}
else if (i >= 240 && i <= 270) {
for (k=0; k<=j; k++)
{
  for (l=0; l<=780; l++);
 }
}
else if (i >= 280 && i <= 300) {
```

```
  for (k=0; k<=j; k++)
  {
    for (l=0; l<=760; l++);
   }
  }
  else if (i >= 310 && i <= 360) {
  for (k=0; k<=j; k++)
  {
    for (l=0; l<=750; l++);
   }
  }
}

//Função para configurar a comunicação serial
int dadoRecebido;
unsigned int angulo;

void confSerial()
{
SCON=0x50;
TCLK=1;
RCLK=1;
C_T2=0;
RCAP2H=0xFF;
RCAP2L=0xFD;
TR2=1;
}


//Função que recebe os dados da Interface Computacional
void recebeDado()
{
// Quando RI = 1 a recepção chegou ao fim
while(!RI);
// Salva o que foi recebido na variável dadoRecebido
dadoRecebido=SBUF;
RI = 0;
}

void main()
{

// Configura comunicação serial
confSerial();

P41 = 0;
P42 = 0;
P43 = 0;

// inicialização do LCD
ini_lcd();
lcd_str("Porta Serial Aguardando ....");
goto_lcd(2, 1);
lcd_str("Produzido por Italo Bruno");

//Verifica porta serial indeterminadamente
```

```
while (1) {

  recebeDado();

//se o dado recebido for igual a zero efetua movimento frontal
  if(dadoRecebido==48 || dadoRecebido==0) {
    ini_lcd();
    goto_lcd(1, 1);
    lcd_str("Movimento Sentido Frontal");
    P41 = 0;
    P43 = 0;
    P42 = 1;
    tempo(10);
    P42 = 0;
    }

//se o dado recebido for igual a 101, gira rodas frontais para
esquerda
  else if(dadoRecebido==101) {
    ini_lcd();
    goto_lcd(1, 1);
    lcd_str("Movimento Sentido Anti-Horário");
    P41 = 1;
    }

//se o dado recebido for igual a 100, gira rodas frontais para
direita
  else if(dadoRecebido==100) {
    ini_lcd();
    goto_lcd(1, 1);
    lcd_str("Movimento Sentido Horário");
    P43 = 1;
    }

//recebe o valo do ângulo a realizar o movimento
  else {
    goto_lcd(2, 1);
    lcd_str("Angulo:");
    angulo = (dadoRecebido*10);
    goto_lcd(2, 9);
    lcd_int(angulo);
    P42 = 1;
    tempo(dadoRecebido*10);
    P42 = 0;
    P41 = 0;
    P43 = 0;

    }

 }

}
```