

INSTITUTO TECNOLÓGICO DE AERONÁUTICA



GABRIELA WERNER GABRIEL

DESENVOLVIMENTO DE SOFTWARE PARA
ACIONAMENTO DE UMA PLATAFORMA MÓVEL
AUTÔNOMA COM MICROCONTROLADOR

Trabalho de Graduação
2003

Eletrônica

GABRIELA WERNER GABRIEL

**DESENVOLVIMENTO DE SOFTWARE PARA ACIONAMENTO DE UMA
PLATAFORMA MÓVEL AUTÔNOMA COM MICROCONTROLADOR**

Orientadores

Prof. Dr. Cairo Lúcio Nascimento Júnior – Instituto Tecnológico de Aeronáutica

Prof. Eduardo Hisasi Yagyu – Instituto Tecnológico de Aeronáutica

Divisão de Engenharia Eletrônica

SÃO JOSÉ DOS CAMPOS

CENTRO TÉCNICO AEROESPACIAL

INSTITUTO TECNOLÓGICO DE AERONÁUTICA

2003

Dados Internacionais de Catalogação-na-Publicação (CIP)

Divisão Biblioteca Central do ITA/CTA

Gabriel, Gabriela Werner

Desenvolvimento de software para acionamento de uma plataforma móvel autônoma com microcontrolador / Gabriela Werner Gabriel.

São José dos Campos, 2003.

104f.

Trabalho de Graduação – Divisão Engenharia Eletrônica – Instituto Tecnológico de Aeronáutica, 2003. Orientadores: Prof. Dr. Cairo Lúcio Nascimento Jr., Prof. Eduardo Hisasi Yagyu.

1. Controladores. 2. Sistemas de computadores embarcados. 3. Dinâmica de robôs. I. Centro Técnico Aeroespacial. Instituto Tecnológico de Aeronáutica. Divisão de Engenharia Eletrônica. II. Título.

REFERÊNCIA BIBLIOGRÁFICA

GABRIEL, Gabriela Werner. **Desenvolvimento de software para acionamento de uma plataforma móvel autônoma com microcontrolador**. 2003. 104f. Trabalho de Conclusão de Curso. (Graduação) – Instituto Tecnológico de Aeronáutica, São José dos Campos.

CESSÃO DE DIREITOS

NOME DO AUTOR: Gabriela Werner Gabriel

TÍTULO DO TRABALHO: Desenvolvimento de software para acionamento de uma plataforma móvel autônoma com microcontrolador

TIPO DO TRABALHO/ANO: Graduação / 2003

É concedida ao Instituto Tecnológico de Aeronáutica permissão para reproduzir cópias deste trabalho de graduação e para emprestar ou vender cópias somente para propósitos acadêmicos e científicos. O autor reserva outros direitos de publicação e nenhuma parte desta monografia de graduação pode ser reproduzida sem a autorização do autor.



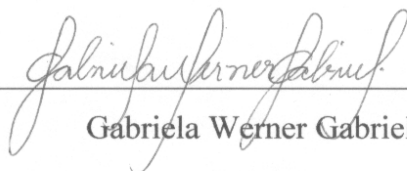
Gabriela Werner Gabriel

Av. Nascimento de Castro, 1506

CEP: 59056-450 – Natal – RN

DESENVOLVIMENTO DE SOFTWARE PARA ACIONAMENTO DE UMA PLATAFORMA MÓVEL AUTÔNOMA COM MICROCONTROLADOR

Essa publicação foi aceita como Relatório Final de Trabalho de Graduação



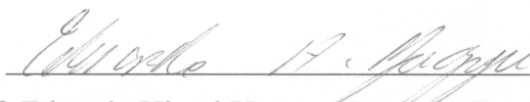
Gabriela Werner Gabriel

Autora



Prof. Dr. Cairo Lúcio Nascimento Júnior – Instituto Tecnológico de Aeronáutica

Orientador



Prof. Eduardo Hisasi Yagyu – Instituto Tecnológico de Aeronáutica

Orientador



Prof. Dr. David Fernandes

Coordenador do Curso de Engenharia Eletrônica

São José dos Campos, 21 de novembro de 2003.

Dedicatória

Dedico este trabalho, e todos os louros, que vieram, vêm ou virão, provenientes especialmente da minha carreira profissional, a meu querido avô Horst Werner, de quem mais ouvi falar do que convivi, mas que mesmo ausente me serve de inspiração na conquista de cada sonho.

Dedico este trabalho ainda a meus queridos pais, Oscar e Marliz, que foram os que primeiro acreditaram em mim. Mais do que isto, eles foram aqueles que quando as forças me faltaram, me fizeram enxergar o caminho certo a seguir.

Agradecimentos

Agradeço aos meus pais pelo carinho, amor e compreensão, as minhas irmãs pela compreensão e por terem contribuído por eu hoje ser quem sou. Agradeço, em especial, aos meus orientadores, pelo apoio, compreensão e paciência e à Fapesp (proc. FAPESP 02/01583-5) pelo apoio concedido a este projeto que nasceu de uma bolsa de iniciação científica.

Agradeço aos meus amigos que constituíram o meu lar nestes dias longe de casa e me fizeram sentir querida, sendo muitas vezes pacientes e compreensivos. Alguns foram verdadeiros anjos enviados por Deus.

E, agradeço ao meu pai do céu, Deus, por tantas vezes ter me segurado em suas mãos.

Resumo

Neste trabalho de graduação foi implementado um microcontrolador embarcado em uma plataforma móvel com o objetivo de transferir parte da inteligência do microcomputador para a plataforma de forma a aumentar a autonomia de movimento da mesma. A plataforma foi construída utilizando motores de passo acoplados a duas rodas dianteiras e mais duas rodas livres traseiras, um kit, o AES-51, contendo o microcontrolador Intel 80C32 (da família 8051) e sensores de contato capazes de detectar a colisão, além dos circuitos necessários para integração destes conjuntos.

A plataforma construída é capaz de realizar os movimentos dentro de um labirinto previamente determinado, sem a necessidade da comunicação física com o computador. Para realizar o movimento dentro do labirinto é necessário inicialmente gerar a trajetória a partir de um programa, desenvolvido num trabalho de graduação anterior pelo aluno Fabio Eiji Yoshitome [1], localizado no microcomputador e então transferi-lo, utilizando interfaces apropriadas, para o microcontrolador. A transmissão é realizada utilizando a interface serial via cabo e após a transmissão o cabo de conexão pode ser retirado e o programa que determinará a trajetória passa a ser executado pelo microcontrolador.

Além disso, foram instalados sensores que detectam a colisão, a qual faz com que o programa no microcontrolador seja interrompido, sendo necessário que o operador sinalize para a plataforma que a mesma pode continuar o percurso inicial.

A trajetória realizada pela plataforma é satisfatória apesar de não ser ainda a ideal, já que durante a execução desta trajetória incidem erros sobre a plataforma, principalmente sobre as rodas da plataforma, que fazem com que a distância percorrida pelas rodas a cada pulso seja aleatória.

Possíveis aplicações vão desde o uso deste tipo de eletrônica embarcada em automóveis guiados por motoristas automáticos, até em aspiradores automáticos capazes de aspirar uma casa sem o auxílio da dona-de-casa.

Abstract

This graduation work is concerned with the implementation of a embedded microcontroller in a mobile platform with the purpose of transferring part of the intelligence from the microcomputer to the platform in order to increase its movement autonomy. The platform is built with two tyres in front of it, drifted by two step motors, two free tyres, a AES-51 kit, with a 8051 family microcontroller (the 80C32), and contact sensors acting as a collision detect system. Moreover, the platform has some circuits used to integrate all this systems.

The constructed platform is able to implement all the necessary movements in a determined labyrinth, without the need of cables to connect the microcomputer to the platform. In order to implement the movement in the labyrinth, it is necessary, first of all, to create the path in which the platform will move. This is done using a program that has been developed by the student Fabio Eiji Yoshitome [1]. Then, we need to transfer the generated information to the microcontroller, placed on the platform. The transmission is done using the serial interface with a flat cable, and after the transmission, the cable can be disconnected from the platform and all the subsequent actions will be implemented by the microcontroller only.

In addition to this, some contact sensors were installed to detect when a collision occurs. At this moment, the program running in the microcontroller is interrupted, remaining in this state until the operator press the enter key on the AES-51 board.

The trajectory implemented by the platform is satisfactory, but, as expected, the planned path was not exactly reached, mainly because of the tyres skid problem. This fact implies in a aleatory event, that is, the different offsets run by the tyres at each motor pulse.

Possibilities of future implementations goes from its use in an automatic driver for automobiles to its use in houseworks, helping with cleaning activities without human interference.

Lista de Figuras

Figura 1 – Foto da parte superior da plataforma móvel autônoma acionada por um microcontrolador presente no kit AES-51.....	3
Figura 2 – Esquema representativo do acionamento dos motores a partir do sinal proveniente do microcontrolador.	13
Figura 3 – Esquema simplificado de um motor de passo.....	13
Figura 4 – Esquema representativo do circuito de potência com os sinais de controle ($Q1$, $Q2 = \overline{Q1}$, $Q3$ e $Q4 = \overline{Q3}$) necessários para o acionamento do motor (representado pelas bobinas B1, B2, B3 e B4).	15
Figura 5 – Representação esquemática das entradas e saídas do circuito lógico para o acionamento do motor de passo.....	16
Figura 6 – Especificação do circuito lógico da figura 5.	17
Figura 7 – Circuito lógico responsável por gerar os sinais para o acionamento do motor de passo.	18
Figura 8 – Circuito de acionamento do motor de passo a partir dos sinais de controle Pulso e S.R.	20
Figura 9: Esboço representativo da conexão física entre o microcontrolador e o microcomputador sendo utilizados dois conectores do tipo DB9 e cabo null modem.	22
Figura 10 – Padrão utilizado na comunicação entre o microcontrolador e o microcomputador.	23
Figura 11 – Byte de configuração para a transmissão de dados via porta serial.	23
Figura 12 – Configuração do byte de status.	24
Figura 13 – Foto dos sensores de contato utilizados para detectar a colisão.....	27
Figura 14 – Configuração do bit de habilitação de interrupção IE.....	29
Figura 15 – Configuração do bit de habilitação de interrupção IE.....	29
Figura 16 – Esboço do posicionamento dos sensores na parte dianteira da plataforma.....	30
Figura 17 – Circuito associado a cada uma das chaves de contato.	31
Figura 18 – Circuito intermediário entre as saídas dos circuito dos contatos NA, figura 17, e a entrada da interrupção Int1.	31
Figura 19 – Circuito responsável por gerar o sinal de interrupção para a porta Int1 do kit AES-51 quando uma colisão é detectada.	32
Figura 20 – Esquema representativo do fluxo de informação através dos programas desenvolvidos para implementar o movimento da plataforma através do labirinto.	33
Figura 21 – Esquema representativo do labirinto no qual a plataforma executa seu movimento. O labirinto representado corresponde ao utilizado para na fase de testes da plataforma móvel.	34
Figura 22 – Arquivo LAB.TXT de entrada para o programa ROMEO.CPP correspondente ao labirinto da figura 21	36
Figura 23 – Arquivo TABM_NEW.TXT contendo os movimentos a serem realizados pela plataforma com base no labirinto da figura 21 e que é gerado pelo programa ROMEO.CPP para posterior leitura pelo programa SEND.CPP.	36
Figura 24 – Figura representativa das dimensões da plataforma.....	45
Figura 25 – Resultado do monitoramento da plataforma relativo ao teste 1. Condições: plataforma operando sem carga e sem pedido de interrupção.....	50
Figura 26 – Resultado do monitoramento da plataforma relativo ao teste 2. Condições: plataforma operando sem carga e sem pedido de interrupção.....	51

Figura 27 – Resultado do monitoramento da plataforma relativo ao teste 3. Condições: plataforma operando sem carga e sem pedido de interrupção.	52
Figura 28 – Resultado do monitoramento da plataforma relativo a um zoom do sinal obtido no teste 3. Condições: plataforma operando sem carga e sem pedido de interrupção.	53
Figura 29 – Resultado do monitoramento da plataforma relativo ao teste 4. Condições: plataforma operando sem carga e com pedido de interrupção.	54
Figura 30 – Resultado do monitoramento da plataforma relativo ao teste 5. Condições: plataforma operando sem carga e com pedido de interrupção.	55
Figura 31 – Resultado do monitoramento da plataforma relativo ao teste 6. Condições: plataforma operando com carga e sem pedido de interrupção.	56
Figura 32 – Resultado do monitoramento da plataforma relativo a um zoom do sinal obtido no teste 6. Condições: plataforma operando com carga e sem sinal de interrupção.	57
Figura 33 – Resultado do monitoramento da plataforma relativo ao teste 7. Condições: plataforma operando com carga e sem sinal de interrupção.	58
Figura 34 – Resultado do monitoramento da plataforma relativo ao teste 8, do monitoramento dos bits MOTORL e Int1. Condições: plataforma operando com carga e com pedido de interrupção.	60
Figura 35 – Resultado do monitoramento da plataforma relativo ao teste 8, do monitoramento dos bits MOTORL e Int1. Condições: plataforma operando com carga e com pedido de interrupção.	60
Figura 36 – Circuito esquemático do kit AES-51.	66
Figura 37 – Circuito esquemático do kit AES-51.	67
Figura 38 – Diagrama esquemático completo do circuito de driver para acionamento dos motores de passo. Os sinais de entrada são obtidos do 80C32 através do jumper J6 (figura 37 – Anexo A).	68
Figura 39 – Esquema representativo da sequência de ações necessárias para que a plataforma execute os movimentos previamente determinados.	91

Lista de Tabelas

Tabela 1 – Tabela verdade para os sinais de saída do circuito lógico da figura 5.....	16
Tabela 2 – Mapa de Karnaugh para obtenção do sinal $Q1(n+1)$	17
Tabela 3 – Mapa de Karnaugh para obtenção do sinal $Q3(n+1)$	18
Tabela 4 – Registradores utilizados na comunicação serial.	24
Tabela 5 – Valores de frequência possíveis para transmissão e recepção de dados com seus respectivos divisores.....	25
Tabela 6 – Tabela referente às posições de memória dos vetores de interrupção para cada um dos tipos de interrupções disponíveis para o microcontrolador 80C32.....	28
Tabela 7 – Correspondência entre os valores de tipos e subtipos e o movimento a ser realizado pela plataforma.	35
Tabela 8 – Tabela indicativa da estrutura do arquivo TABM_NEW.TXT gerado pelo programa ROMEO.CPP.	35
Tabela 9 – Relação dos níveis lógicos dos bits que indicam o sentido de rotação de cada um dos motores e o movimento realizado pela plataforma (saída do programa).	39
Tabela 10 – Resultados obtidos das medições realizadas na plataforma representada pela figura 24.....	44
Tabela 11 – Resultados calculados para as dimensões da plataforma móvel.....	46
Tabela 12 - Tabela referente aos valores medidos para os deslocamentos da plataforma.	46
Tabela 13 – Valores calculados da média e desvio padrão para as grandezas que determinam os deslocamentos reais da plataforma para cada pulso.....	47
Tabela 14 – Tabela dos movimentos realizados pela plataforma nos testes que se seguem de 1 a 8.	49

Sumário

1. Introdução.....	1
2. Documentação da Plataforma.....	3
2.1. Documentação do Kit AES-51	4
2.1.1. Descrição do Hardware do Kit AES-51	4
2.1.1.1. O Microcontrolador e a Memória	5
2.1.1.2. Controle de Entrada do Teclado	6
2.1.1.3. Visor de Cristal Líquido (LCD)	7
2.1.1.4. Entradas e Saídas Digitais	7
2.1.1.5. Entrada e Saída Analógica.....	8
2.1.1.6. Porta Serial	9
2.1.1.7. Lógica de Controle do Sistema.....	9
2.1.2. Desenvolvimento de Programas Utilizando o AES-51	10
2.2. Driver Utilizado para o Acionamento dos Motores	12
2.2.1. Motor Utilizado	13
2.2.2. Descrição do Funcionamento da Placa de Driver.....	14
2.2.3. Projeto da Placa de Acionamento dos Motores e do Circuito de Potência	15
2.2.4. Diagrama do Circuito Elétrico.....	20
2.3. Chave para Acionamento da Plataforma	21
3. Comunicação Serial	22
3.1. Configuração da Comunicação Serial	22
4. Motivação para o Uso de Sensores	26
4.1. Implementação dos Sensores de Colisão.....	26
4.2. As Interrupções na Implementação dos Sensores de Colisão.....	27
4.2.1. Configuração da Interrupção.	28
4.3. Hardware Implementado para Detectar a Colisão	30
5. Programas que Implementam o Movimento da Plataforma no Labirinto.....	33
5.1. Alterações Realizadas no Programa ROMEO.CPP.....	37
5.2. Programa do Microcontrolador MOVEA.ASM	39
5.2.1. Funcionamento do Programa de Acionamento da Plataforma	39
5.2.2. Programa MOVEA.ASM	41
5.3. Programa de Transmissão de Dados.....	42
6. Testes de Dimensões.....	44
7. Resultados.....	48
8. Discussão dos Resultados.....	61
9. Conclusão e Comentários Finais.....	62
10. Referências Bibliográficas	64
10.1. Literatura	64

10.2. Softwares Utilizados.....	65
<i>ANEXO A – Desenho Esquemático do Circuito do kit AES-51.</i>	<i>66</i>
<i>ANEXO B – Circuito Completo do Driver de Acionamento dos Motores de Passo</i>	<i>68</i>
<i>ANEXO C – Código Referente às Alterações Realizadas no Programa ROMEO.CPP</i>	<i>69</i>
<i>ANEXO D – Código do Programa MOVEA.ASM</i>	<i>73</i>
<i>ANEXO E – Código do Programa SEND.CPP.....</i>	<i>86</i>
<i>ANEXO F – Execução do Movimento da Plataforma Passo a Passo.....</i>	<i>91</i>

1. Introdução

Este trabalho é a continuidade de um projeto inicial que calcula a melhor trajetória entre dois pontos, o inicial e o final fornecidos, a ser desenvolvida por uma plataforma móvel, dentro de um labirinto, também previamente fornecido.

Como, de acordo com este trabalho anterior, iniciado com o aluno Fábio Eiji Yoshitome [1], a plataforma movimentava-se utilizando os pulsos gerados pelo próprio programa, que rodava no microcomputador, tínhamos uma conexão física entre a plataforma e o microcomputador, o que limitava a amplitude do movimento da mesma.

Assim, a idéia principal deste trabalho de graduação é deslocar parte da inteligência para a plataforma de forma a aumentar tal amplitude de movimento. Para isto, poder-se-ia escolher uma entre duas possibilidades: utilizar ou a comunicação via sinais de rádio ou implementar o programa de movimento na própria plataforma, através do uso de um microcontrolador.

O método escolhido foi a implementação do programa de acionamento da plataforma na própria plataforma mediante o uso de um microcontrolador da família 8051, especificamente o 80C32, presente no kit de desenvolvimento de softwares AES-51.

Decorrente disto, este trabalho tem por objetivo desenvolver um software para ser executado no microcontrolador presente na plataforma móvel de forma que possamos realizar o movimento da mesma sem a necessidade da comunicação física com o computador, além da transferência prévia de dados para a plataforma. Dados estes provenientes do programa ROMEO.EXE, desenvolvido pelo aluno do Instituto Tecnológico de Aeronáutica, Fábio Eiji Yoshitome, Turma 97 [1], que escolhe a melhor trajetória a ser desenvolvida num labirinto previamente fornecido, utilizando o algoritmo A*.

Este trabalho realizou algumas alterações no programa ROMEO de forma a fazer com que os dados gerados estivessem prontos para serem transmitidos para a plataforma de forma simples e de fácil decodificação pela mesma.

A plataforma é constituída por um conjunto de dois motores de passo acionados mediante pulsos provenientes de sinais digitais do microcontrolador, presente na placa do kit AES-51, intermediado por um circuito de potência / acionamento capaz de converter o sinal digital em sinal de potência para acionar os motores de passo. Além disso, faz-se necessária uma bateria, tanto para fornecer a tensão necessária para acionar o kit quanto para fornecer a

tensão necessária para ativar o circuito de potência e a energia necessária para acionar os motores de passo.

Neste trabalho foram ainda implementados na plataforma sensores de contato de forma a detectar a colisão da plataforma com qualquer outro objeto presente na trajetória da mesma, evitando assim, sua destruição.

2. Documentação da Plataforma

A plataforma construída é composta pelas seguintes partes:

- KIT AES-51;
- Motores de passo;
- Driver para acionamento dos motores;
- Rodas;
- Bateria;
- Plataforma.
- Sensores de Contato
- Circuito dos Sensores

As quais partes podem ser vistas na figura 1.

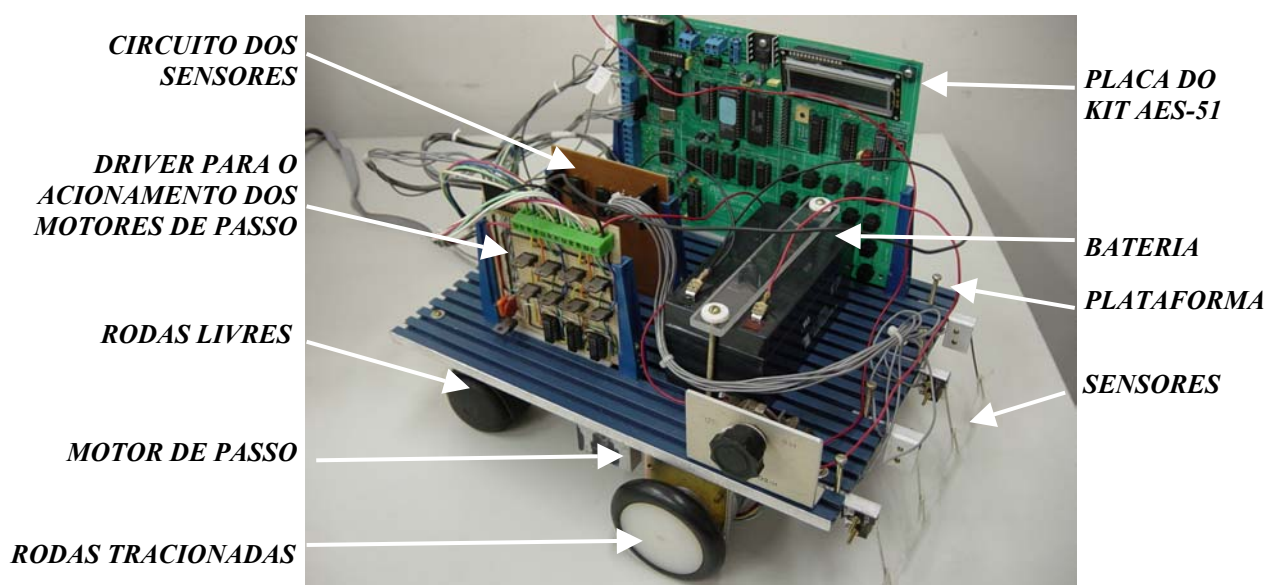


Figura 1 – Foto da parte superior da plataforma móvel autônoma acionada por um microcontrolador presente no kit AES-51.

Tais partes estão descritas a seguir, exceto as rodas, a plataforma e a bateria escolhidas. Sendo a plataforma escolhida em tamanho razoável desde que coubesse e suportasse os demais componentes da mesma (bateria, driver de acionamento, circuito de detecção de colisão, sensores de colisão, motores de passo e kit AES-51); as rodas, escolhidas de forma que estivessem em concordância com as dimensões da plataforma escolhida e de forma que suportasse o peso a que estariam submetidas, sendo utilizadas duas rodas grandes

na parte dianteira da plataforma, acionadas pelos motores de passo e duas rodas livres na parte traseira para equilíbrio da mesma; e a bateria escolhida de acordo com as exigências do kit e dos motores de passo utilizados, tendo a mesma as seguintes especificações técnicas:

Bateria Recarregável

Modelo: CP1232 – 12V; 3,2AH

Center Power Tech Co., LTD

2.1. Documentação do Kit AES-51

O AES-51, Advanced Educational Systems, utilizando o microcontrolador 80C32, é um kit didático utilizado para auxiliar no estudo dos microcontroladores. É interessante notarmos que os microcontroladores diferenciam-se dos microprocessadores pelas suas interfaces de hardware: UARTs, timers, portas de I/O e uma ROM interna. Mas, independentemente de ser um microcontrolador ou um microprocessador o princípio de funcionamento é o mesmo, pois controlam, ou contém, os mesmos blocos funcionais.

2.1.1. Descrição do Hardware do Kit AES-51

O kit contendo o microcontrolador utilizado pode ser dividido nos seguintes blocos funcionais principais:

- Microcontrolador e memória, sendo uma memória RAM estática de 32kB e uma EPROM de 32kB;
- Lógica de controle da placa;
- Controle da entrada do teclado;
- Um visor de cristal líquido de 2 linhas de 16 caracteres;
- Entradas e saídas digitais;
- Portas de I/O analógicas;
- Portas seriais de I/O.

O esquema elétrico do kit está apresentado nas figuras 36 e 37 (Anexo A).

Além disso, o kit acompanha um conjunto de arquivos utilizados para fazer a comunicação entre a placa e o microcomputador, para escrever e compilar os arquivos, além de alguns arquivos de exemplos de programação para facilitar o aprendizado do uso do kit.

Os principais programas incluídos no kit são:

- EEDIT – editor de textos;
- ASM51 – utilizado para compilar programas em ASSEMBLY, criando o arquivo .HEX.
- TE – programa que realiza a comunicação entre a placa e o microcomputador transferindo os códigos necessários – do computador para o microcontrolador – para a execução dos programas pelo microcontrolador.

Além disso, para o funcionamento da placa é necessária uma fonte de tensão externa DC, que pode ser aplicada ou mediante uma bateria de 6V ou mediante um adaptador da rede elétrica de 115VAC para 12VDC (ou bateria de 12VDC).

2.1.1.1. O Microcontrolador e a Memória

Este bloco é composto pelos chips:

80C32 (U1) – Microcontrolador da família 8051

74HC373 (U2) – flip flop D

27C256 (U3) – EPROM 32 kB

62256 (U4) – RAM 32 kB

A memória do 80C32 é de 64kB endereçada por 16 bits no barramento de endereços. No entanto, no kit, são utilizados somente 32 kB de memória e conseqüentemente, o bit mais significativo (A15) do barramento não é utilizado. Além disso, os bits menos significativos do barramento de endereço são multiplexados a bits de dados do barramento de dados.

Como os bits de endereço A0~A7 são multiplexados aos bits de dados D0~D7, é necessário um circuito de latch e um sinal de ALE (Address Latch Enable), de forma que possamos habilitar ou não o barramento de endereços ou de dados.

Para a memória utilizada pelo sistema temos a RAM e a ROM ambas de 32 kB.

É interessante observar que devido ao sistema do AES-51 possuir um interpretador BASIC no seu interior parte da memória é utilizada por estas sub-rotinas e funções próprias da linguagem BASIC.

Existem dois modos de operação da memória disponível na placa:

- Se o jumper J5 (figura 37 – Anexo A) estiver conectado (caso utilizado), a primeira metade dos endereços de memória (de 0 a 3FFFH) são endereços de memória ROM utilizados para código de programa, no chip U3, sendo a segunda parte dos endereços deste chip não utilizados. E a segunda metade da memória utilizada, do endereço 4000H ao 7FFFH, de memória RAM, também destinada a código de programa, localizada no chip U4. Sendo a segunda metade dos endereços deste chip, de 4000H a 7FFFH, destinados tanto para memória de código de programa, quanto para alocação de dados.
- E, no caso do jumper J5 (figura 37 – Anexo A) desconectado, toda a memória de código de programa é armazenada no chip U3, do endereço 0 ao 7FFFH, e toda memória destinada a alocação de dados localiza-se no chip U4, do endereço 0 ao 7FFFH.

O modo básico de operação é aquele com o jumper J5 conectado. Este modo foi o utilizado, devido ao modo de operação com jumper J5 desconectado exigir a programação de uma ROM própria para operação do microcontrolador.

2.1.1.2. Controle de Entrada do Teclado

Composto pelos chips:

74C923 (U5) – Codificador de 20 linhas

74HC244 (U6) – Buffer tri-state de 8 vias

Quando uma tecla é pressionada no teclado o chip U5 codifica aquela tecla gerando um código binário de cinco dígitos, representativo daquela tecla. Ao mesmo tempo, este chip gera um pulso positivo que aciona o clock de um Flip Flop D, no chip 74HC74 (U10), isso faz com que tenhamos uma saída em nível lógico 0, na saída –Q do Flip Flop, este pulso aciona a interrupção 0 do microcontrolador 80C32, avisando que uma tecla foi pressionada no teclado. O microcontrolador verifica a variável INT0, alocada na ROM, para saber qual dispositivo

está fazendo o pedido de interrupção. O teclado é então selecionado e o microcontrolador dá um reset no Flip Flop D. Neste instante o código da tecla lida é lançado no barramento de dados sendo este lido pelo microcontrolador, finalmente o pedido de interrupção 0 recebe um reset.

2.1.1.3. Visor de Cristal Líquido (LCD)

O único chip que compõe este bloco é o 74HC374 (U7), que é um flip flop tipo D.

Para acessar o LCD, deve-se selecionar este dispositivo através do chip 74HC138 (U11). Este chip envia, então, um pulso de clock para o chip U7 (o Flip Flop D), o qual dispõe o código do caractere a ser escrito no LCD na porta de entrada do mesmo.

O flip flop é necessário devido ao tempo de processamento do LCD ser muito maior que o tempo de processamento do microcontrolador.

Para habilitar o LCD e o código do dado a ser escrito são utilizados os bits P10 e P11 do chip do microcontrolador, ligados aos bits RS e E, presentes no LCD.

O LCD é indicado no circuito esquemático da figura 36 (Anexo A) pelo jumper J4.

2.1.1.4. Entradas e Saídas Digitais

As entradas e saídas digitais do chip estão localizadas nos conectores de entrada e saída digital de dados ou no conector de entrada e saída analógica. Estes conectores são responsáveis por levar dados para o exterior ou receber dados do exterior da placa.

- I0, I1, I2 e I3 são pinos que podem ser utilizados tanto como entrada quanto como saída de dados, sendo I0 habilitado para uso somente quando o teclado não estiver habilitado e o jumper J7 (figura 36 – Anexo A) estiver removido;
- P1.0 e P1.1 podem ser utilizados como entrada ou saída de dados somente quando o LCD não estiver sendo usado;
- P1.2 está disponível no conector J3 (figura 36 – Anexo A) se as funções PWM ou conversor D/A não estiverem sendo utilizadas;

- P1.3 está disponível como entrada ou saída de dados digital somente se o conversor A/D não estiver sendo utilizado;
- P1.4, P1.5, P1.6 e P1.7 estão disponíveis como entradas de dados através do jumper J6 (figura 37 – Anexo A), ou como saída (denominadas O1, O2, O3 ou O4) mediadas por um buffer (chip U15). Estes pinos são conectados ainda a quatro LEDs distintos de forma que é possível verificar o sinal de saída nestas portas. Olhando-se com atenção ao esquemático das figuras 36 e 37 (Anexo A), podemos verificar que estas saídas estão disponíveis tanto no conector de saída quanto no jumper J6, no entanto, devido à presença de um chip inversor, utilizado para proteger os LEDs, o sinal enviado para cada uma destas portas, disponível em J6, apresenta-se invertido no conector de saída. Para o caso em questão, foi utilizado o jumper J6 como saída de dados.

2.1.1.5. Entrada e Saída Analógica

Este bloco é composto pelos chips:

ADC0804 (U8) – Conversor A/D de 8 bits

LM385 (U16) – Amplificador Operacional

A conversão é iniciada quando o conversor A/D é selecionado através do pino DS7 do chip U11, o 74C923. A conversão é feita e quando o chip U8 a termina, avisa ao microcontrolador, atribuindo nível lógico 0 ao pino P1.3. Neste instante, o microcontrolador lê o sinal convertido através do bit DS7. Enquanto a leitura é feita um reset é automaticamente realizado no chip U8. O tempo de conversão está entre 100 e 150 μ s, e a excursão do sinal de entrada pode ser calibrada através de um resistor destinado a este fim. Além disso, o pino da entrada está protegido por um resistor de 1k e um diodo zener de 5,6 V.

Já o conversor D/A é selecionado pelo pino P1.2. Se o jumper J3 estiver na posição PWM, o chip U16 atua como um simples seguidor de tensão para o pino P1.2. Caso J3 esteja na posição de filtro D/A, a saída analógica é proporcional ao pulso em P1.2, e o chip U16 atua como um conversor D/A, sendo a amplitude máxima do sinal de saída no chip U16 de 3.5 V.

2.1.1.6. Porta Serial

O chip principal deste bloco é MAX233 (U9), responsável pela interface da transmissão e recepção de dados do tipo RS-232 através de um conector do tipo DB-9 (T1/R1) referente à primeira porta serial, ou através dos pinos Tx e Rx disponíveis no conector da segunda porta serial (T2/R2). A primeira ou a segunda porta serial é selecionada através do chip U11, nos pinos DS5 ou DS6, e o chaveamento entre as portas é realizado através de duas portas lógicas NANDs, duas portas lógicas ORs e de um flip flop D, de forma que a saída do flip flop em nível lógico alto seleciona a segunda porta serial e em nível lógico baixo seleciona a primeira porta serial. Além disso, é importante notar que a primeira porta serial está sempre habilitada, sendo o flip flop responsável por habilitar ou não a segunda porta serial.

O chip U9 é responsável por fazer a conversão do sinal de entrada de ± 12 V, padrão em comunicação do tipo RS-232, para níveis de 0 a 5 V, excursão dos níveis de tensão dos dados na placa do kit AES-51, bem como realizar a conversão inversa.

Para o desenvolvimento deste projeto utilizou-se a primeira porta serial para a comunicação com o microcomputador e envio de dados deste para o microcontrolador.

2.1.1.7. Lógica de Controle do Sistema

Esta parte do circuito é composta pelos chips:

- 74HC74 (U10) – Flip Flop tipo D
- 74HC138 (U11) – Decodificador de 3-8 linhas
- 74HC08 (U12) – Porta Lógica AND
- 74HC32 (U13) – Porta Lógica OR
- 74HC00 (U14) – Porta Lógica NAND

Ela implementa o controle de acesso às diferentes memórias interna ou externa ao microcontrolador, nos chips U3 e U4, selecionando a parte da memória destinada a dado ou a código de programa dependendo do modo de operação da memória utilizado (selecionado pelo jumper J5) e o tipo de acesso a esta memória (escrita ou leitura).

Além disso, esta parte do circuito implementa o acesso a outras diferentes partes da placa, através dos terminais DS0 ao DS7 do chip U11: DS3 acessa o LCD, DS4 acessa o teclado, DS5 e DS6 acessam as duas portas seriais, DS7 acessa o conversor A/D-D/A, e DS0, DS1 e DS2 são disponíveis ao usuário, através do jumper J1 (figura 37 – Anexo A), para acesso a equipamentos externos que possam vir a ser implementados pelo mesmo.

Além destes blocos, para a alimentação da maioria dos chips é necessário um nível de tensão de 5 V. Isto é conseguido através do chip regulador de tensão LM2940.

2.1.2. Desenvolvimento de Programas Utilizando o AES-51

Para desenvolver algum tipo de programa utilizando o kit AES-51, podemos fazê-lo através de duas possibilidades distintas:

- Escrever o programa utilizando o código hexadecimal de cada comando desejado, fazendo a entrada de dados com o próprio teclado do kit;
- Escrever o programa em ASSEMBLY ou em BASIC utilizando um PC, e realizando a comunicação via porta serial.

Assim, devido à forma como o kit foi desenvolvido, a programação do microcontrolador utilizando este kit é bastante simples, podendo ser escrito utilizando a linguagem ASSEMBLY ou BASIC-32, já que existe uma biblioteca para interpretar tais programas no firmware referente ao kit, que ocupa uma boa parte da memória disponível.

Para criar os programas na linguagem ASSEMBLY, bem como na linguagem BASIC-32, o kit acompanha um editor de textos, o EEDIT.EXE, executado em ambiente DOS no microcomputador. Para a linguagem BASIC, existe ainda a possibilidade de o programa ser escrito na própria interface de comunicação com a placa (programa TE.EXE).

O kit acompanha também um manual de programação com suas principais instruções.

A linguagem utilizada no programa de controle da plataforma foi a ASSEMBLY, devido a anterior familiaridade com a mesma.

Para a programação em ASSEMBLY o arquivo texto criado, no caso utilizando o EEDIT.EXE, contendo o código do programa, deve ser salvo com a extensão .ASM. E, tendo-se feito isto, a compilação é realizada com a linha de comando no prompt do DOS:

\>ASM51 <nome do programa>.ASM

Este comando traduz o programa criado para a linguagem de máquina, em dois passos distintos: no primeiro passo, o Cross Assembler (compilador utilizado) cria uma tabela de símbolos a partir dos símbolos e labels utilizados no programa escrito (o arquivo fonte); e no segundo passo, a tradução do arquivo fonte para a linguagem de máquina é realmente realizada. O arquivo em linguagem de máquina tem a extensão .HEX.

A transferência do programa para a placa do kit para a posterior execução do mesmo é realizada pelo programa TE.EXE, Terminal Emulator, que também acompanha o kit.

A respeito do programa TE, existe um conjunto de comandos que são importantes para a execução do programa de acionamento da plataforma:

- Barra de espaços – Toda vez que o botão de reset é pressionado, limpando assim a memória de programa no microcontrolador, é necessário realizar a comunicação entre a plataforma e o microcomputador. Esta comunicação é realizada utilizando-se a barra de espaços do teclado do PC.
- Alt+c – Este comando carrega as configurações relativas à utilização da interface do TE.EXE com o usuário e aquelas relativas à comunicação do PC com a placa do kit. A configuração utilizada foi a padrão, sendo destacados os seguintes itens relativos a comunicação:

Porta serial utilizada: COM 1.

Velocidade de transferência de dados: 9600 kbps.

Não são utilizados bits de paridade.

Tamanho dos dados transmitidos: 8 bits

Número de stop bits utilizados: 1

- RX – Quando este comando é utilizado informamos ao TE que o programa a ser lido encontra-se em linguagem de máquina. Isto é necessário devido a quando trabalhamos com programas escritos em BASIC, a interpretação do programa é realizada pelo próprio TE.
- Alt+s – No caso de o programa ser escrito em ASSEMBLY, este comando faz a transferência do programa para o microcontrolador. A versão do programa aqui utilizado é aquela que está traduzida em linguagem de máquina, no caso, com a extensão .HEX.

- Call <posição de memória> - Comando utilizado para executar o programa, que está na posição de memória indicada, no microcontrolador,
- Alt+a – Comando utilizado para sair do programa.
- Alt+x – Comando utilizado para limpar a tela do programa TE no PC.

Para executar um programa basta, então, seguir a sequência de comandos acima descrita.

2.2. Driver Utilizado para o Acionamento dos Motores

Para o acionamento dos dois motores de passo é necessário utilizarmos um circuito de potência intermediado por um circuito de acionamento.

O circuito de potência é necessário na medida em que o sinal fornecido pela placa do kit é digital e a corrente que o kit fornece é muito baixa, e o motor de passo precisa de uma corrente muito acima da máxima corrente que o kit pode fornecer. Assim, se ligássemos o motor de passo ao circuito do kit, o motor de passo não seria acionado.

Além disso, para o acionamento dos motores é necessário um circuito intermediador, entre a placa do kit e o circuito de potência. Isto é necessário devido a quando a placa do kit está desligada, todos os seus barramentos estão em nível baixo. Ao ser ligada, todos os seus barramentos vão a nível lógico alto. Assim, se o circuito de potência está diretamente ligado aos barramentos do microcontrolador, os transistores de potência entram em condução. Conseqüentemente, os motores entram num estado indefinido e a fonte deverá fornecer o dobro da corrente usualmente consumida.

Outra vantagem de se utilizar um circuito de acionamento é que, qualquer problema que vier a ocorrer nos motores de passo ou no circuito de potência não danificará o microcontrolador.

2.2.1. Motor Utilizado

Para facilitar a programação do kit do microcontrolador foi decidido que a plataforma seria acionada por dois motores de passo. O motor utilizado para imprimir torque nas rodas da plataforma é um motor de passo da Astrosyn, tipo 23LM – 005, Miniangle Stepper, cujas especificações técnicas são:

- Tensão nominal = 12V/fase
- Corrente nominal = 0,6A/fase
- Rotação de 1,8°/pulso

O acionamento do motor é feito segundo o esquema representado na figura 2, abaixo.

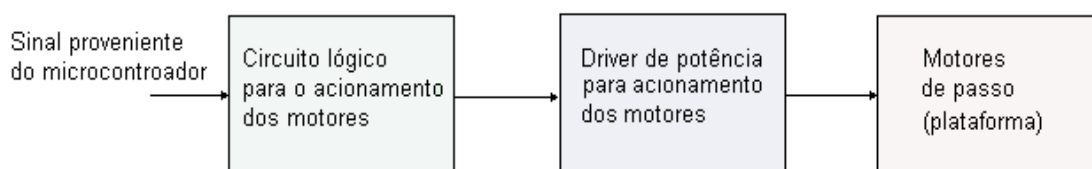


Figura 2 – Esquema representativo do acionamento dos motores a partir do sinal proveniente do microcontrolador.

Os motores de passo são uma espécie de motor síncrono construídos de forma a rotacionar um número fixo de graus para cada pulso elétrico recebido. Por isso, geralmente são utilizados para realizar controle de posicionamento em ambientes bidimensionais.

Uma figura simplificada de um motor de passo pode ser vista na figura 3, em que temos um campo devido às correntes que circulam no estator e um campo devido ao rotor.

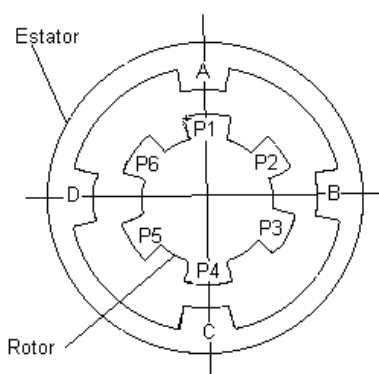


Figura 3 – Esquema simplificado de um motor de passo.

O estator tem, então, suas bobinas alimentadas de forma que o seu fluxo resultante, no gap da máquina, gire no sentido horário ou no sentido anti-horário. Isto pode ser conseguido alimentando-se suas bobinas com corrente num sentido ou no outro.

Assim, se fizermos o campo do estator girar, o rotor entrará em movimento na tentativa de alinhar seu campo com o resultante do estator.

É interessante notar que os campos do estator giram assumindo posições intermediárias entre A e B, ou entre B e C, e assim sucessivamente, de forma que quando o alinhamento é conseguido tenhamos um torque resultante, chamado de torque de retenção, que faz com que o motor pare naquela posição.

Como, para fazer o campo do estator girar, devemos aplicar correntes em suas bobinas ora num sentido, ora no outro, é usual utilizarmos duas bobinas por cada um dos eixos de forma que o fluxo possa ser invertido simplesmente aplicando-se corrente numa bobina ou na outra.

Além disso, é importante sabermos, para o controle da plataforma, que o motor de passo utilizado tem uma curva de torque fornecida pelo fabricante, tal que a taxa máxima de pulsos no tempo que conseguimos imprimir ao motor de forma que ele consiga entrar em movimento sem precisar utilizar uma rampa de alimentação para acelerar e desacelerar o motor, é de 400 pulsos por segundo.

A taxa de pulsos fornecida ao motor em uso é calculada da seguinte forma:

$$taxa = \frac{1}{440ms / pulso} = 2,27 \text{ pulsos} / s, \quad (1)$$

em que o período do pulso aplicado é de 440 ms.

2.2.2. Descrição do Funcionamento da Placa de Driver

A placa de driver do sistema recebe como entrada pulsos que correspondem ao sentido de rotação de cada um dos motores e pulsos responsáveis por fazer o motor rotacionar em um passo.

O circuito de driver toma tais sinais dois a dois, um par para cada motor (um relativo ao pulso e um relativo ao sentido de rotação), e através de uma lógica digital envia os

comandos necessários aos motores para que eles possam realizar a rotação num sentido ou no outro.

O sentido de rotação é determinado pela ordem com que as bobinas de cada motor são alimentadas de forma a produzir um campo girante no sentido desejado. Sendo esta alimentação realizada através de transistores de potência, de forma que a corrente que sai do circuito de acionamento, com baixa potência, seja amplificada de forma a fornecer potência suficiente para o acionamento dos motores.

2.2.3. Projeto da Placa de Acionamento dos Motores e do Circuito de Potência

Uma forma simplificada de representar o circuito de acionamento de um motor de passo e os sinais necessários para o seu acionamento está representada na figura 4, abaixo.

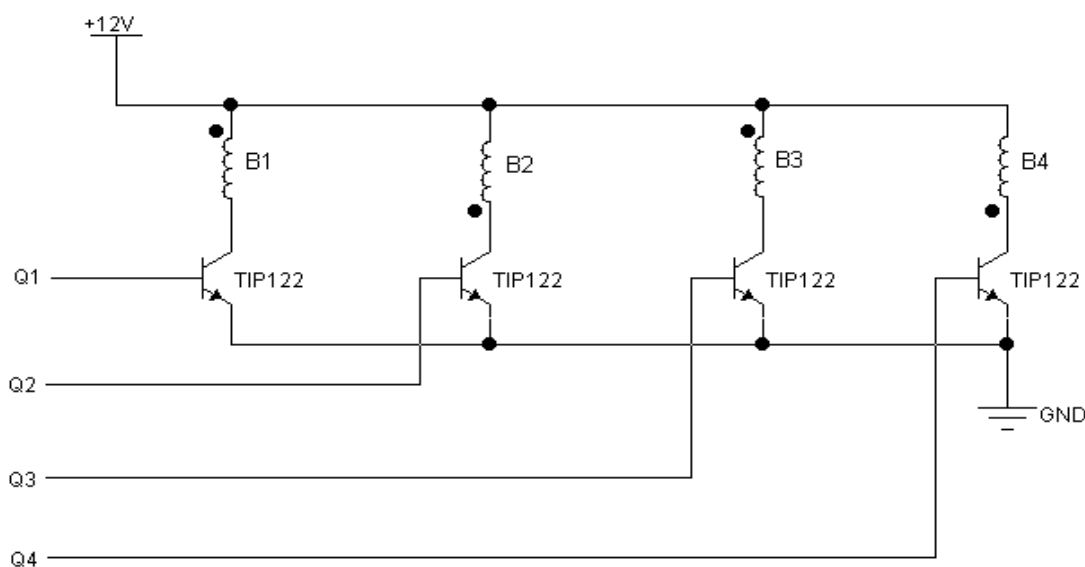


Figura 4 – Esquema representativo do circuito de potência com os sinais de controle ($Q1$, $Q2 = \overline{Q1}$, $Q3$ e $Q4 = \overline{Q3}$) necessários para o acionamento do motor (representado pelas bobinas B1, B2, B3 e B4).

Assim, temos como entrada para o acionamento do motor os sinais Q1, Q2, Q3 e Q4. No entanto, o programa executado pelo microcontrolador fornece como saída apenas sinais de pulso (doravante, Pulso) e sinais de controle de sentido de rotação (doravante, S.R.).

Portanto, faz-se necessária a introdução de um circuito lógico entre as entradas Pulso e S.R. e as saídas Q1, Q2, Q3 e Q4, que acionarão o motor através do circuito de potência da figura 4 (Figura 5).

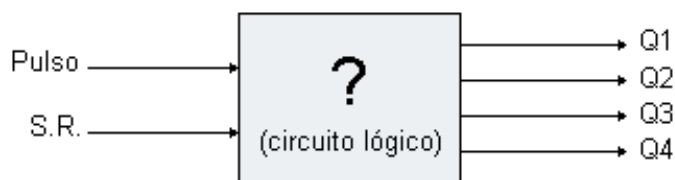


Figura 5 – Representação esquemática das entradas e saídas do circuito lógico para o acionamento do motor de passo.

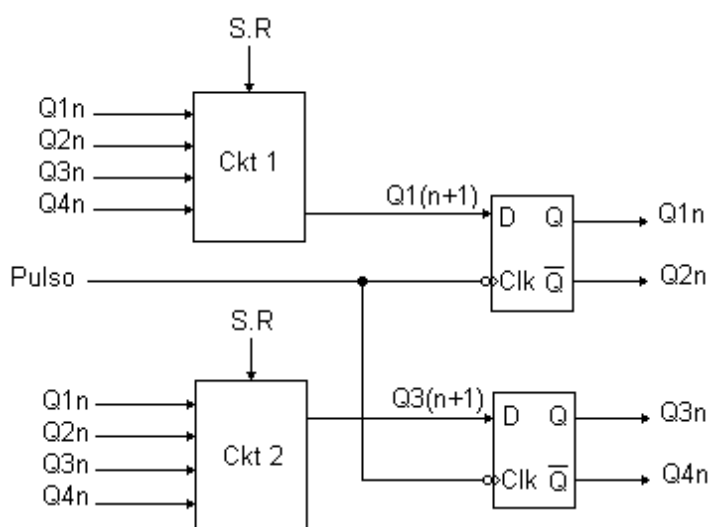
Fazendo a equivalência: nível lógico 1 correspondente a corrente de base do transistor suficiente para polarizá-lo e nível lógico 0, correspondente a corrente de base nula, ou seja, o transistor não está polarizado, obtemos a tabela verdade apresentada na tabela 1 para a seqüência de polarização das bobinas B1, B2, B3 e B4 no acionamento do motor.

Tabela 1 – Tabela verdade para os sinais de saída do circuito lógico da figura 5.

S.R.	Q1	Q2	Q3	Q4
0	1	0	1	0
0	1	0	0	1
0	0	1	0	1
0	0	1	1	0
1	0	1	1	0
1	0	1	0	1
1	1	0	0	1
1	1	0	1	0

Esta seqüência de comandos é obtida tendo-se em vista o funcionamento do motor de passo, que para fazer o rotor girar num sentido ou no outro, deve seguir uma determinada seqüência de pulsos. Como a descrita na tabela 1.

A tabela 1 é referente a uma máquina seqüencial já que a ordem com que as bobinas são alimentadas num sentido e no outro, influenciará diretamente no sentido em que o rotor vai girar. Portanto, podemos utilizar no projeto da sua lógica flip flops. No caso, basta utilizarmos dois flip flops, um referente ao sinal Q1, e outro referente ao sinal Q3, nos próximos estados. Isto devido aos sinais Q2 e Q4 serem, respectivamente, $\overline{Q1}$ e $\overline{Q3}$. Assim, o circuito da figura 5 pode ser representado pelo circuito da figura 6, abaixo.



Legenda:

Q_{in} : cada um dos estados atuais, com $i=1,2,3$ e 4.

$Q1(n+1)$: cada um dos próximos estados, com $i=1$ e 3.

Figura 6 – Especificação do circuito lógico da figura 5.

Os circuitos lógicos 1 e 2, referentes à figura 6, podem ser obtidos utilizando-se os mapas de Karnaugh, que estão apresentados nas tabelas 2 e 3 referentes a $Q1(n+1)$ e $Q3(n+1)$, respectivamente.

Tabela 2 – Mapa de Karnaugh para obtenção do sinal $Q1(n+1)$.

		$Q1n$	$\overline{Q1n}$	
$\overline{S.R.}=0$		0	0	$\overline{Q3n}$
		1	1	$Q3n$
$S.R.=1$		0	0	$\overline{Q3n}$
		1	1	$Q3n$

$$Q1(n+1) = \overline{S} R Q3n + S \overline{R} \overline{Q3n}$$

Tabela 3 – Mapa de Karnaugh para obtenção do sinal $Q3(n+1)$.

	$Q1n$	$\overline{Q1n}$	
$\overline{S.R.}=0$	0	1	$\overline{Q3n}$
	0	1	$Q3n$
$S.R.=1$	1	0	$Q3n$
	1	0	$\overline{Q3n}$

$$Q3(n+1) = \overline{S.R.} \overline{Q1n} + S.R. Q1n$$

Portanto, o circuito da figura 6 torna-se o circuito apresentado na figura 7.

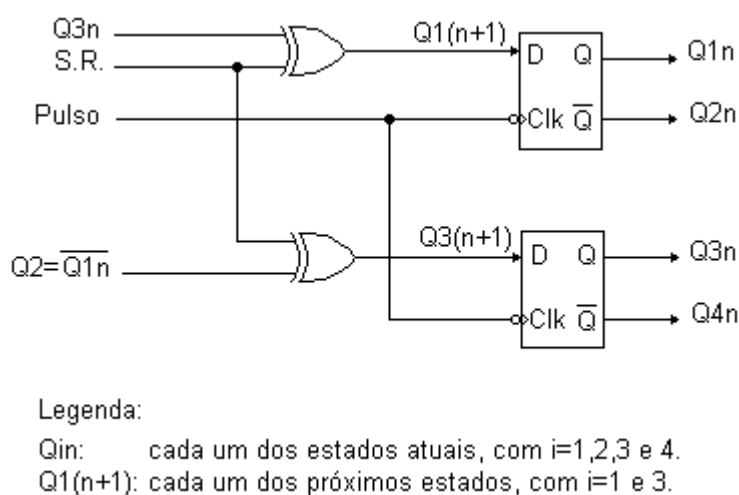


Figura 7 – Circuito lógico responsável por gerar os sinais para o acionamento do motor de passo.

Conectando-se os circuitos das figuras 4 e 7 temos o circuito completo para o acionamento do motor de passo a partir de sinais de controle fornecidos pelo microcontrolador, Pulso e S.R.

No entanto, é válido serem ressaltados alguns pontos importantes:

1. Os sinais de controle são obtidos a partir de portas do tipo CMOS, e portanto, podem alimentar no máximo até duas portas do tipo TTL (no caso, as portas ou exclusivas) de forma que o nível do sinal nas portas do tipo TTL seja suficiente para ser percebido por ela.

Caso tivéssemos mais portas do tipo TTL “penduradas” em uma porta do tipo CMOS, deveríamos utilizar um amplificador de corrente, o que não é o nosso caso.

2. A junção dos circuitos das figuras 4 e 7 deve ser feita mediante a introdução de uma resistência R, de valor a ser determinado, de forma que consigamos limitar a corrente que flui do circuito lógico para o circuito de potência, alimentando-o, sem danificar os componentes utilizados.

Cálculo da resistência R:

Os transistores utilizados no circuito da figura 4 são do tipo TIP122, para os quais temos:

- $\beta \approx 1000$, para corrente de coletor aproximadamente 4 A.
- Tensão de trabalho típica na base do transistor: $V_{be,Típ} =$ entre 1,4V e 2,5 V. (neste caso, temos liberdade de escolher qual o valor de V_{be} .)
- Tensão de trabalho mínima na base para polarizar o transistor: $V_{be,Min} = 1,4$ V.
- Corrente de base mínima para polarizar o transistor: $I_{b,Min} = 0,1$ mA.
- Corrente de base máxima: $I_{b,Max} = 120$ mA.

Para as bobinas do motor de passo a corrente nominal é de 0,6A, portanto a fonte deve ser capaz de fornecer 1,2A.

E, para os flip flops utilizados, que é uma porta do tipo TTL, 74LS74, temos:

- Corrente máxima na saída Qn: $I_{Max} = 0,4$ A.
- Nível alto de tensão mínimo na saída Qn: $V_{OH,Min} = 2,4$ V.
- Nível alto de tensão máximo na saída Qn: $V_{OH,Max} = 5$ V(max).
- Corrente na saída Qn: $I_{Qn} = 0 \sim 400$ uA.

Assim, podemos calcular os valores máximos e mínimos da resistência R.

O valor máximo da resistência R ocorre para a corrente mínima que deve circular entre os circuitos de potência e de lógica digital, de forma que os componentes continuem polarizados, garantindo assim o funcionamento do circuito. Portanto,

$$R < \frac{V_{OH,Max} - V_{be,Min}}{I} < 9k\Omega \quad (2)$$

Em que I corresponde ao máximo entre a corrente mínima na saída Qn e a corrente de base mínima para polarizar o transistor.

Além disso, deve-se observar que a resistência R não pode ser menor que um determinado valor, de forma que a corrente que flui neste ramo do circuito não ultrapasse o valor máximo de forma que os componentes não sejam danificados. Assim,

$$R > \frac{V_{OH,Min} - V_{b,Tip}}{I} > 3.3k\Omega \quad (3)$$

Em que I, neste caso, corresponde ao mínimo das correntes máximas suportadas pelo flip flop e pela base do transistor.

Tomou-se, então, o valor de resistência de 3.9 kΩ.

2.2.4. Diagrama do Circuito Elétrico

O circuito completo para o acionamento de um motor de passo a partir de sinais de controle de rotação do motor e de pulsos para fazer o motor girar o equivalente a um passo é apresentado na figura 8.

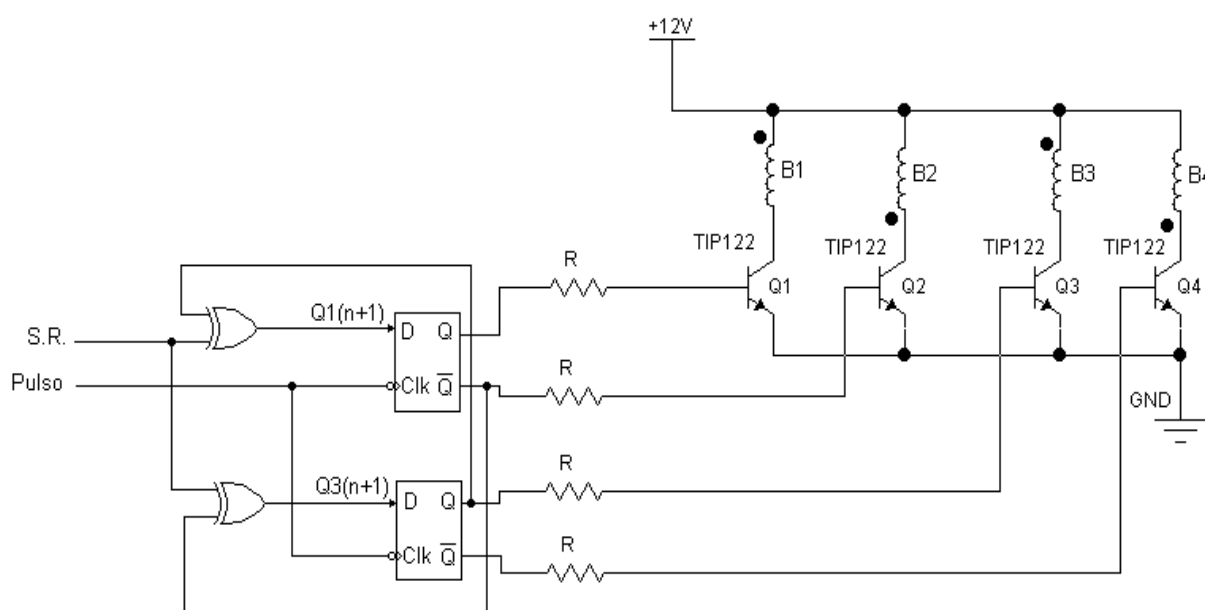


Figura 8– Circuito de acionamento do motor de passo a partir dos sinais de controle Pulso e S.R.

O programa executado pelo microcontrolador fornece como saída, nas portas de saída relativas ao 80C32, apenas sinais Pulso, em P1.4 e P1.6, e sinais S.R, em P1.5 e P1.7, para cada um dos motores, sendo P1.4 e P1.5 relativos ao motor do lado esquerdo e P1.6 e P1.7 relativos ao motor do lado direito.

O circuito esquemático completo pode ser visto na figura 38, no Anexo B.

2.3. Chave para Acionamento da Plataforma

Para acionar a plataforma é utilizada uma chave que ao ser modificada da posição ON para a posição OFF, energiza primeiro o kit AES-51 e em seguida os motores. Para o caso dos motores de passo a ordem com que os equipamentos são acionados não é tão crítica, como seria caso fossem utilizados motores de corrente contínua. No entanto, preferiu-se realizar a energização dos circuitos desta forma criteriosa para que caso, em trabalhos posteriores, opte-se por utilizar motores de corrente contínua não haja danos nos circuitos utilizados.

Para o caso do motor de corrente contínua, caso a energização ocorra primeiro nos motores e num instante posterior no circuito de potência, corre-se o risco de danificar o hardware da plataforma, pois quando a placa de acionamento está desligada, os níveis lógicos estão todos baixos (correspondente ao nível de tensão 0V), e, portanto, os motores assumiriam que este nível estaria na sua entrada, o que poderia ser fatal para os mesmos.

Então, liga-se, primeiramente, o circuito de acionamento, de forma que os níveis lógicos fiquem estabelecidos em níveis de tensão próprios, para, posteriormente, ligar-se os motores.

3. Comunicação Serial

A comunicação entre o microcomputador e o microcontrolador é feita através da porta serial, devido ao tipo de comunicação já existente entre o computador e o microcontrolador, utilizando o protocolo de comunicação RS-232.

A comunicação serial, no microcomputador, pode ser realizada por uma das portas COM1, COM2, COM3 ou COM4. Aqui foi utilizada a porta COM1, cujo endereço em um microcomputador padrão IBM PC-AT é 03F8h.

Além disso, a comunicação realizada é conhecida como “without handshake”, em que o dado é enviado e não é aguardada a confirmação do receptor. A configuração física das linhas de comunicação entre as portas seriais do microcomputador e do microcontrolador é apresentada na figura 9 abaixo.

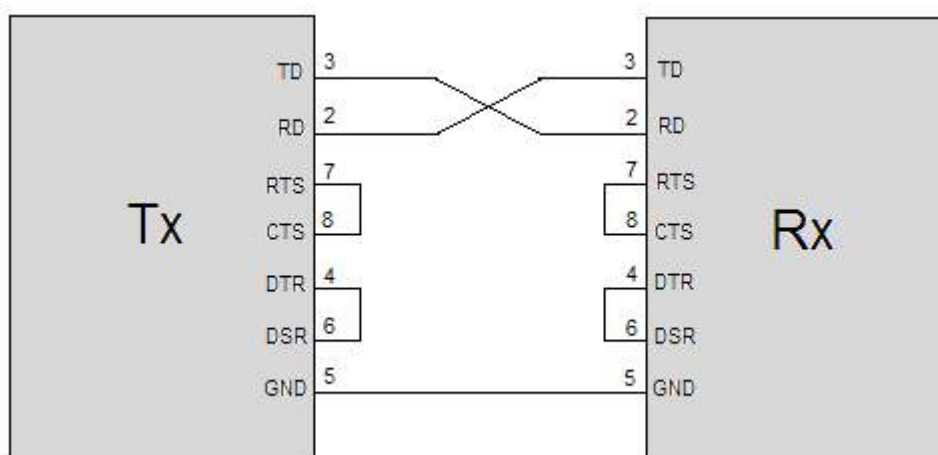


Figura 9: Esboço representativo da conexão física entre o microcontrolador e o microcomputador sendo utilizados dois conectores do tipo DB9 e cabo null modem.

3.1. Configuração da Comunicação Serial

Para que a comunicação seja possível é necessário que a configuração de todos os dados utilizados para a transmissão seja a mesma, tanto no microcontrolador quanto no computador utilizado. Assim, o padrão utilizado foi o apresentado na figura 10, abaixo.

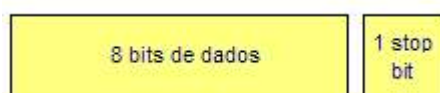


Figura 10 – Padrão utilizado na comunicação entre o microcontrolador e o microcomputador.

É interessante observar que este padrão não é fixo nem para o microcomputador nem para o microcontrolador, podendo ser alterado na tela de configurações do programa TE (no caso do microcontrolador), o qual foi anteriormente apresentado, e no próprio código do programa de transmissão de dados (no caso do microcomputador).

A configuração da porta serial RS-232 do microcomputador é realizada enviando-se um byte de configuração, antes da transmissão, para o endereço do LCR (Line Control Register) que corresponde ao endereço base da porta serial + 3 (ver tabela 4), onde o endereço base para a porta serial COM1 é 03F8h.

Este byte de configuração é do tipo apresentado na figura 11.

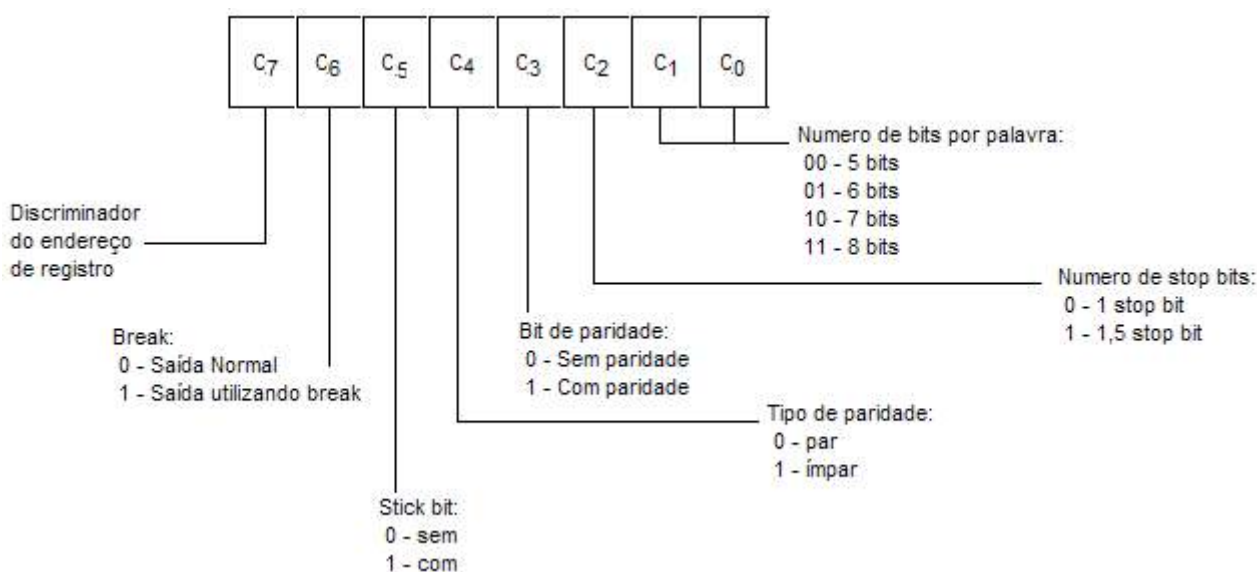


Figura 11 – Byte de configuração para a transmissão de dados via porta serial.

O byte de configuração utilizado neste trabalho é 00000011, pois desejamos C1C0: 11 → 8 bits; C2: 0 → 1 stop bit; C3: 0 → sem paridade; C4: indiferente; C5: 0 → sem stick bit; C6: 0 → saída normal; C7: 0.

Para uma melhor visualização, os registradores utilizados na transmissão serial são os apresentados na tabela 4.

Tabela 4 – Registradores utilizados na comunicação serial.

Registrador	Endereço	Comentário
TD/RD Buffer	03F8h	Endereço Base da COM1
Interrupt Enable	03F9h	Endereço Base da COM1 + 1
Interrupt Identity	03FAh	Endereço Base da COM1 + 2
Line Control	03FBh	Endereço Base da COM1 + 3
Modem Control	03FCh	Endereço Base da COM1 + 4
Line Status (LSR)	03FDh	Endereço Base da COM1 + 5
Modem Status	03FEh	Endereço Base da COM1 + 6
Scratch Pad	03FFh	Endereço Base da COM1 + 7

Além disso, é interessante observar que temos disponível o LSR (Line Status Register), que é um byte que nos fornece o status da transmissão (endereço base + 5).

Com este byte podemos saber qual tipo de erro ocorreu durante a transmissão. Isto pode ser verificado de acordo com a figura 12, que indica a configuração do byte de status.

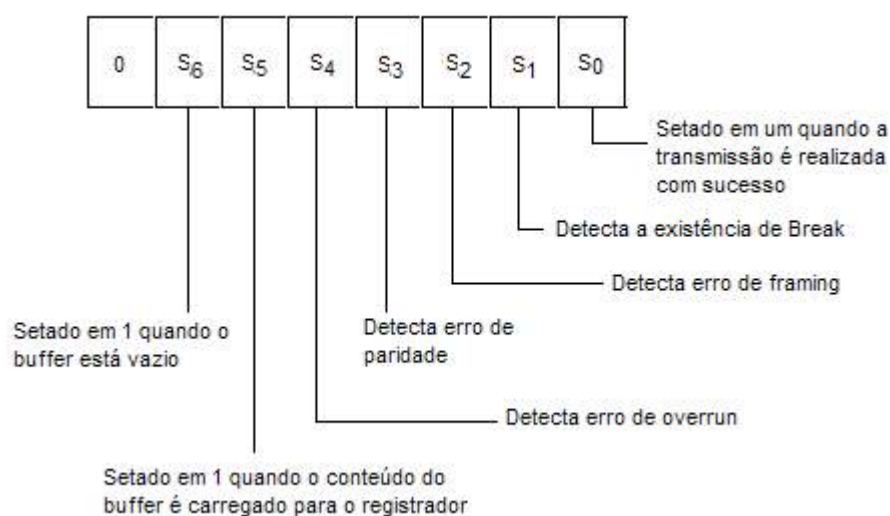


Figura 12 – Configuração do byte de status.

Para que a comunicação seja possível é necessário ainda que as velocidades de transmissão e de recepção sejam as mesmas, sob pena da comunicação se tornar inviável. Tais velocidades devem ser divisores da frequência do clock do computador utilizado e do clock presente no microcontrolador.

A operação aritmética utilizada para a determinação do fator multiplicativo que irá configurar a transmissão na frequência desejada é:

$$Baud_Rate = \frac{frequencia_do_clock}{N.16} \quad (4)$$

$$N = \frac{frequencia_do_clock}{Baud_Rate.16} \quad (5)$$

Assim, sendo a frequência do clock do microcomputador de 18,432 MHz e uma baud rate de 9600 pré-estabelecida no programa do microcontrolador para a transmissão serial e que corresponde a maior velocidade de transmissão por parte do microcontrolador, temos N=12 (000CH).

A velocidade de transmissão é setada enviando-se o divisor de frequência para o endereço do TD/RD Buffer e a posição de memória seguinte, já que o divisor é um número de 16 bits, sendo enviado primeiro a parte baixa do divisor (0Ch) e posteriormente a parte alta do mesmo (00h).

As frequências possíveis para a transmissão ou recepção de dados são as apresentadas na tabela 5 a seguir.

Não se pode esquecer que qualquer alteração que seja feita na configuração da transmissão a ser realizada (no microcomputador) deve ser feita também na configuração da recepção. Como neste caso, a recepção é feita no microcontrolador, a mudança da configuração é feita através do programa Terminal Emulator - TE, utilizando-se o comando ALT+C.

É interessante observar ainda que do lado do microcontrolador as taxas de transmissão disponíveis são 1.2k, 2.4k, 4.8k e 9.6k para a baud rate, o que limita ainda mais as possibilidades para a taxa de transmissão.

Tabela 5 – Valores de frequência possíveis para transmissão e recepção de dados com seus respectivos divisores.

Baud Rate	Divisor
110	0417h
300	0180h
600	00C0h
1200	0060h
1800	0040h
2400	0030h
4800	0018h
9600	000Ch
19200	0006h

4. Motivação para o Uso de Sensores

Para que exista de fato o controle da plataforma ao longo do labirinto descrito pela mesma, é necessário que esta sinta o espaço e o ambiente no qual está inserida pois, por mais que o software e o hardware implementados sejam perfeitos e teoricamente sejam suficientes para fazer com que a plataforma execute a trajetória programada, existem diversos fatores que provocam erros reais que fazem com que a plataforma se desvie deste caminho pré-determinado.

Assim, faz-se necessário que a plataforma seja dotada de um conjunto de sensores que lhe confirmem sentidos em relação a este ambiente, possibilitando desta forma o “auto controle” da mesma.

Este conjunto de sensores é deixado para trabalhos posteriores. Aqui, iniciaremos com o mais simples e imprescindível dentre todos os sensores, que é um sensor de contato capaz de detectar a colisão.

Este sensor é indispensável no projeto de qualquer robô móvel na medida em que, uma colisão poderia inutilizar todo o trabalho realizado. Portanto, é necessário que, se uma colisão vier a ocorrer, a plataforma seja capaz de sentir o perigo eminente e abortar o programa que vinha sendo realizado.

4.1. Implementação dos Sensores de Colisão.

Na tarefa de detectar a colisão foram utilizados sensores de contato NA (normalmente abertos) associados a interrupção do microcontrolador.

Para implementação do hardware dos sensores de contato foram utilizadas chaves de contato com hastes metálicas compridas de forma que quando pressionadas, fecham o contato.

As especificações das chaves, apresentadas na figura 13, utilizadas são:

MM2G6

0.1A 12.5Vca

8918



Figura 13 – Foto dos sensores de contato utilizados para detectar a colisão.

A implementação do software dos sensores foi realizada utilizando rotinas de interrupção. Isto devido à colisão ser um evento de máxima importância para plataforma de forma que todo programa que estiver sendo executado deve ser interrompido por ocasião deste evento.

A interrupção aqui utilizada foi a interrupção externa Int1 devido a Int0 estar vinculada ao teclado do kit AES-51.

O funcionamento dos sensores se dá de forma que quando um objeto encosta na haste das chaves de contato, posicionadas na parte dianteira da plataforma, uma interrupção é gerada no software da plataforma e o programa que vinha sendo implementado é abortado, entrando em funcionamento a rotina de interrupção, a qual será abortada e retornará ao programa original quando o teclado for acionado.

4.2. As Interrupções na Implementação dos Sensores de Colisão

O uso de interrupções é particularmente interessante para esta aplicação pela rapidez de resposta à eventos externos e pelo tamanho do código utilizado para este fim, quando comparadas com as rotinas de polling.

A interrupção aqui utilizada é a mascarada que precisa ser habilitada em algum instante do programa para que possa entrar em vigor. No caso, sua habilitação ocorre no instante em que a execução da trajetória se inicia, já que anteriormente a este instante, período

de transmissão de dados do PC para o microcontrolador e tratamento dos dados recebidos, não há sentido em se falar em colisão pois a plataforma encontra-se parada.

O 80C32, sendo um microcontrolador de 8 bits atende às interrupções buscando os endereços das mesmas nos vetores de interrupção, daí existir a necessidade de uma configuração prévia da interrupção.

Além disso, existe a possibilidade de definirmos a prioridade nas interrupções, o que precisa ser também previamente configurado.

4.2.1. Configuração da Interrupção.

Para o microcontrolador em questão os endereços dos vetores de interrupção para cada uma das interrupções existentes são as que seguem (tabela 6):

Tabela 6 – Tabela referente às posições de memória dos vetores de interrupção para cada um dos tipos de interrupções disponíveis para o microcontrolador 80C32.

<i>Posição da Memória</i>	<i>Interrupção</i>
4003H	Interrupção Externa 0 (Int0)
400BH	Timer 0 Overflow (T0)
4013H	Interrupção Externa 1 (Int1)
401BH	Timer 1 Overflow (T1)
4023H	Porta Serial
402BH	Timer 2 Overflow / Interrupção Externa 2 (T2 / Int2)

Assim, o endereço da interrupção de colisão deve ser gravado na posição 4013H, de forma que, quando a colisão ocorrer e a interrupção for acionada, o microcontrolador possa buscar pela rotina de interrupção.

Vale salientar que a interrupção é acionada quando o conector Int1 (figura 37 – Anexo A) for carregado com nível baixo (0V), pois tal bit encontra-se normalmente em nível alto (5V).

Após ter carregado o vetor de interrupção com o endereço da rotina de interrupção, é necessário configurar a interrupção propriamente dita. No entanto, como tanto na recepção de

dados, quando na rotina de detecção de colisão foram utilizado o Timer 1 para definir os parâmetros de interrupção, tais configurações prévias tornam-se desnecessárias.

Basta então, que sejam habilitadas a Int1 e esta seja definida como de prioridade máxima.

O byte que habilita as interrupções é o IE (endereço 0A8H), endereçável bit a bit, e cuja composição é apresentada na figura 14 a seguir.

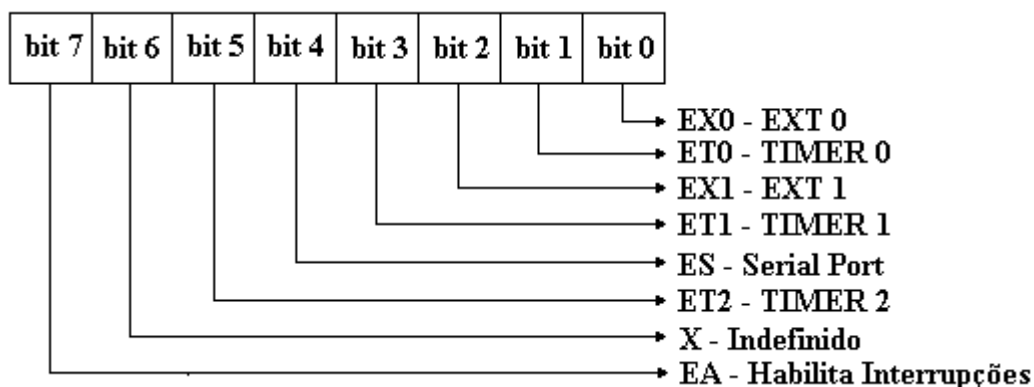


Figura 14 – Configuração do bit de habilitação de interrupção IE.

Assim, para que a rotina de interrupção esteja habilitada é necessário carregarmos IE com o valor 10000100B.

E, para definição de prioridade, é utilizado o byte IP (endereço 0B8H), também bit a bit endereçável, cujos significados são apresentados na figura 15 a seguir.

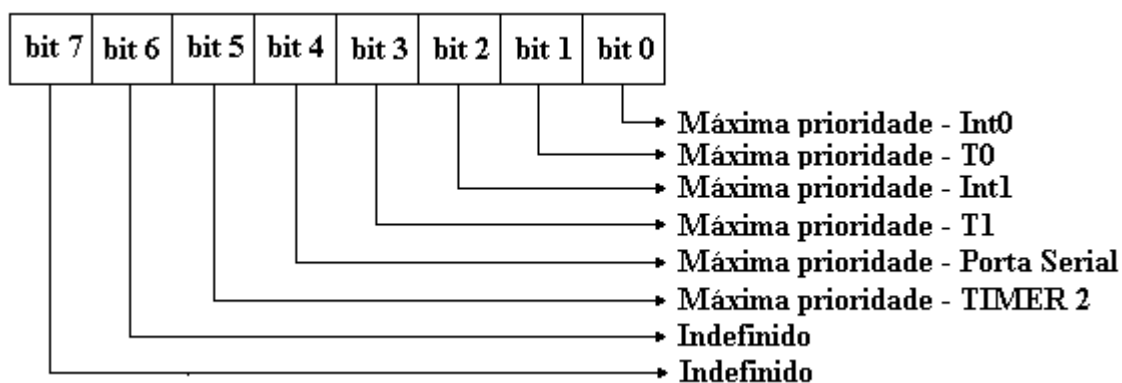


Figura 15 – Configuração do bit de habilitação de interrupção IP.

Logo, o valor que deve ser carregado em IP, para definir a rotina de colisão como a de máxima prioridade é 00000100B.

Então, para que a rotina seja efetivamente executada quando uma colisão ocorrer é necessário que a primeira linha de código relativa a esta rotina esteja no endereço gravado no vetor de interrupção relativo à Int1.

4.3. Hardware Implementado para Detectar a Colisão

Os sensores de contato, anteriormente comentados, foram dispostos ao longo da parte dianteira da plataforma de forma que cobrissem todo este lado. Assim, foram utilizados 4 sensores, aos quais foram coladas pequenas placas de plástico, como mostra a figura 16 ilustrativa, a seguir, bem como pode ser visto na figura 13.

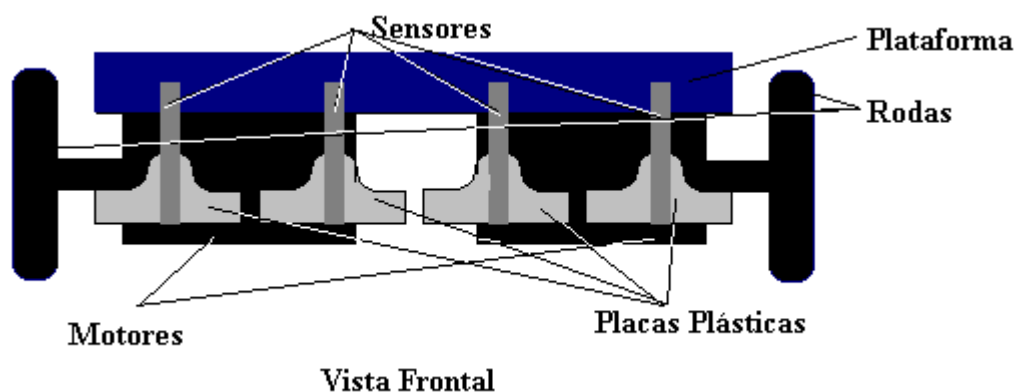


Figura 16 – Esboço do posicionamento dos sensores na parte dianteira da plataforma.

E, o circuito construído foi elaborado de forma a aplicar nível alto à interrupção Int1 constantemente e baixa-lo no instante em que a colisão for detectada. Tal circuito, construído para cada um dos sensores, encontra-se na figura 17, na qual a saída S é conectada à interrupção Int1 do kit AES-51, intermediado por um circuito de lógica.

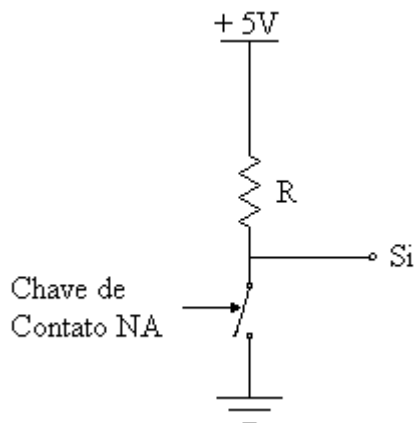


Figura 17 – Circuito associado a cada uma das chaves de contato.

Para a resistência R foi utilizado o valor de $R = 560 \text{ ohms}$. Este valor foi determinado devido a impedância de entrada da porta Int1 não ser afetada por este valor. Por este critério um valor razoável para R seria:

$$500 \Omega < R < 1k\Omega \quad (6)$$

pois a impedância de entrada, Z_{in} , da Int1 corresponde a, aproximadamente, 360Ω .

Além disso, como foram utilizados quatro sensores independentes, e temos apenas uma entrada Int1, utilizou-se para intermediar a entrada Int1 e as quatro saídas Si uma porta NAND de quatro entradas com Schmitt-Trigger para estabilizar o sinal seguido de um inversor também com Schmitt-Trigger, por motivo análogo.

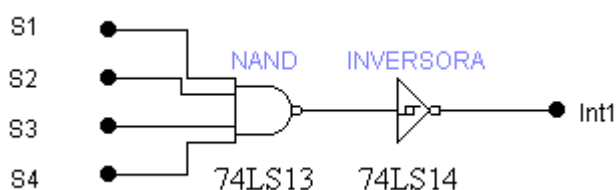


Figura 18 – Circuito intermediário entre as saídas dos circuito dos contatos NA, figura 17, e a entrada da interrupção Int1.

A porta lógica NAND foi preferida aqui em detrimento de um circuito paralelo simples devido a deixar aqui uma possibilidade de desenvolvimento para um trabalho posterior a implementação de um decodificador nas saídas Si, com a finalidade de sabermos qual dos sensores foi acionado e assim podermos verificar o tamanho do objeto (com auxílio de outros sensores) e, conseqüentemente podermos determinar um desvio ou um recálculo da trajetória.

Finalmente, o circuito relacionado às colisões corresponde ao apresentado na figura 19 a seguir.

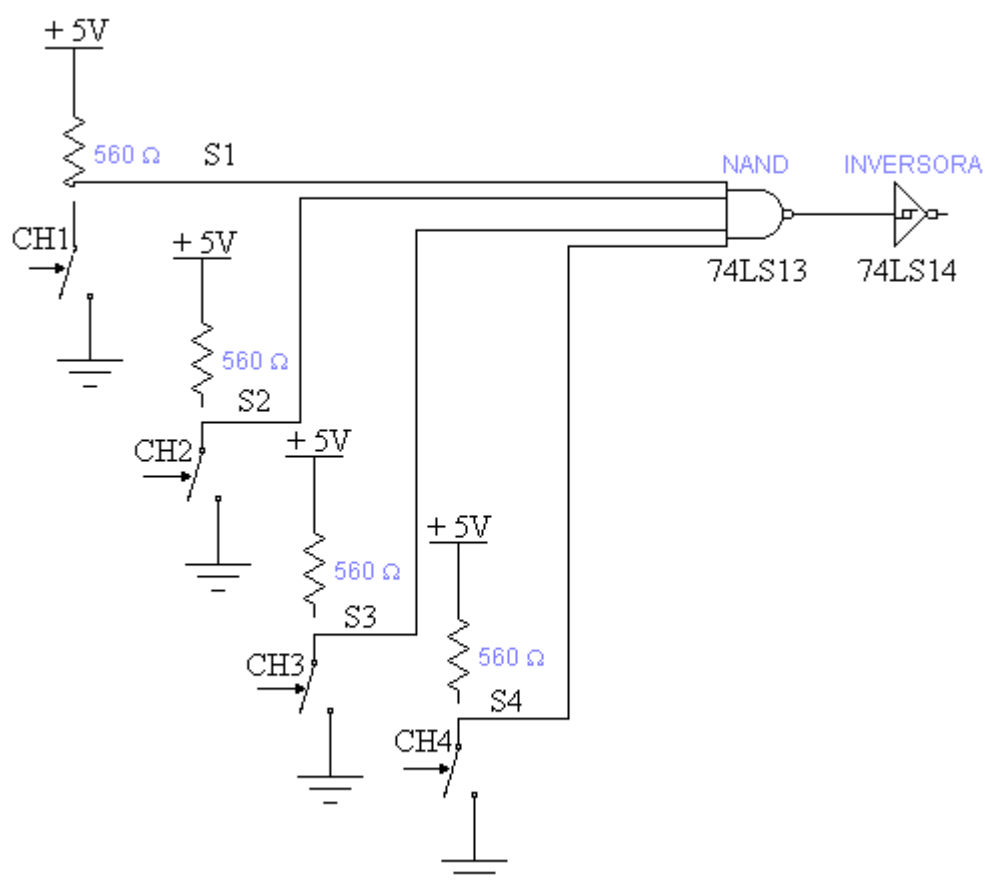


Figura 19 – Circuito responsável por gerar o sinal de interrupção para a porta Int1 do kit AES-51 quando uma colisão é detectada.

5. Programas que Implementam o Movimento da Plataforma no Labirinto

Inicialmente, para compreensão dos programas desenvolvidos é necessário entender como é realizada a comunicação entre a plataforma e o computador, o que é feito a seguir.

Para que a comunicação computador-plataforma possa funcionar de acordo com o especificado e a plataforma possa realizar o movimento programado no labirinto fornecido, como o apresentado na figura 21, é necessária a execução de três programas:

- ROMEO.CPP: programa executado no microcomputador que calcula a trajetória a ser seguida pela plataforma, a partir do arquivo <nome do labirinto>.TXT, e gera o arquivo TABM_NEW.TXT a ser enviado para a plataforma contendo os movimentos desejados;
- SEND.CPP: programa executado no microcomputador que envia os dados da trajetória calculada, presentes no arquivo TABM_NEW.TXT, via porta serial RS-232, para o microcontrolador do kit AES-51 montado na plataforma;
- MOVEA.ASM: programa executado no microcontrolador do kit AES-51 que: a) recebe os dados enviados pelo programa SEND.CPP, b) grava-os em posições de memória pré-estabelecidas, c) realiza os movimentos da plataforma através do labirinto, e d) implementa a rotina de detecção de colisão.

O fluxo de informação ao longo da execução do movimento da plataforma desde a escolha da melhor trajetória até o fim da execução dos movimentos propriamente ditos pode ser visto na figura 20, a seguir.

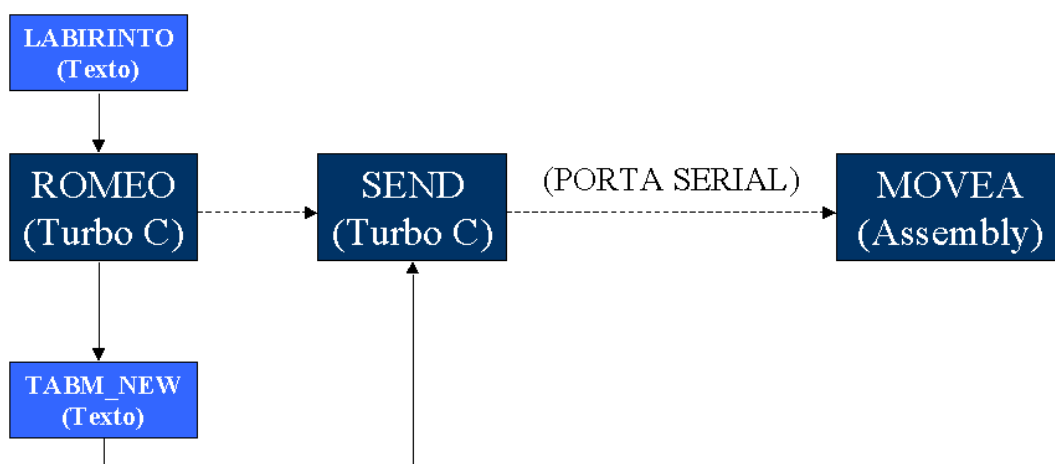


Figura 20 – Esquema representativo do fluxo de informação através dos programas desenvolvidos para implementar o movimento da plataforma através do labirinto.

O código fonte, impresso, e compilado (extensão .EXE e .HEX) destes programas estão disponíveis como anexo deste trabalho de graduação.

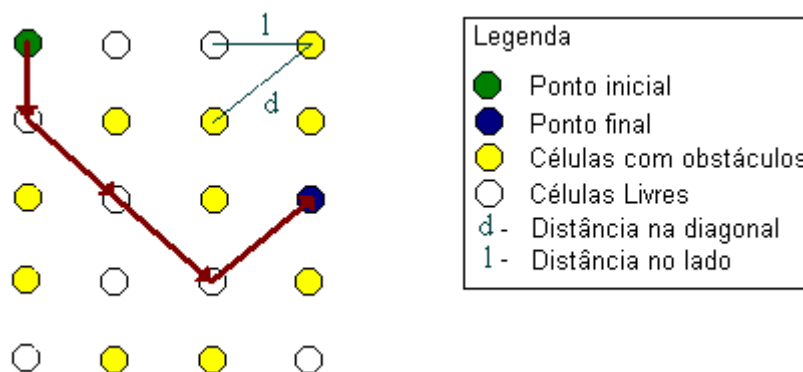


Figura 21 – Esquema representativo do labirinto no qual a plataforma executa seu movimento. O labirinto representado corresponde ao utilizado para na fase de testes da plataforma móvel.

Primeiramente o programa ROMEO.CPP é executado e um arquivo TABM_NEW.TXT, em formato texto, é gerado e armazenado no disco rígido do microcomputador. Este arquivo contém os dados da trajetória que deve ser seguida pela plataforma, e é estruturado de forma a apresentar todos os valores das variáveis que armazenam o número de pulsos para que cada um dos movimentos de translação e de rotação seja executado, além de conter os movimentos de rotação e de translação a serem realizados pela mesma num determinado labirinto.

Mais especificamente, os movimentos de translação e rotação a serem executados pela plataforma são gravados no arquivo TABM_NEW.TXT em várias linhas (tantas quantas necessárias de acordo com o movimento apresentado), sendo cada linha constituída por um número que indica o tipo de movimento a ser realizado ou o subtipo do mesmo. (De acordo com esta codificação dos movimentos estes são codificados em números pré-determinados correspondentes a cada tipo e sub-tipo de movimento como apresentado na tabela 7 apresentada a seguir)

Este arquivo é esperado pelo programa SEND.CPP.

Tabela 7 – Correspondência entre os valores de tipos e subtipos e o movimento a ser realizado pela plataforma.

Tipo	Movimento Correspondente	Subtipo	Movimento Correspondente
1	Frente	1	Translação-lado
		2	Translação-diagonal
2	Trás	1	Translação-lado
		2	Translação-diagonal
3	Rotação Horária	1	Rotação de 45°
		2	Rotação de 90°
		3	Rotação de 135°
4	Rotação Anti-horária	1	Rotação de 45°
		2	Rotação de 90°
		3	Rotação de 135°

A tabela 8 indica a estrutura especificada para o arquivo TABM_NEW.TXT.

Tabela 8 – Tabela indicativa da estrutura do arquivo TABM_NEW.TXT gerado pelo programa ROMEO.CPP.

Número da linha	Significado do número referente à linha
1, 2 e 3	Número de pulsos necessários para realizar uma rotação de 45°.
4, 5 e 6	Número de pulsos necessários para realizar uma rotação de 90°.
7, 8 e 9	Número de pulsos necessários para realizar uma rotação de 135°.
10, 11 e 12	Número de pulsos necessários para realizar uma translação entre dois pontos do labirinto alinhados na horizontal ou na vertical (translação-lado). Indicado por l na figura 2.
13, 14 e 15	Número de pulsos necessários para realizar uma translação entre dois pontos do labirinto alinhados na diagonal (translação-diagonal). Indicado por d na figura 2.
16	Tipo do primeiro movimento.
17	Subtipo do primeiro movimento.
18	Tipo do segundo movimento.
19	Subtipo do segundo movimento.
...	...
Último	A (utilizada para terminar o programa rodado no microcontrolador).

Este arquivo texto deve ser gerado no mesmo diretório no qual o programa de transmissão de dados será executado.

Na figura 22 é apresentado o arquivo de entrada do programa ROMEO, correspondendo ao labirinto da figura 21. O arquivo TABM_NEW.TXT que é gerado pelo programa ROMEO é apresentado na figura 23.

5	4		
2	0	0	1
0	1	1	1
1	0	1	3
1	0	0	1
0	1	1	0

Figura 22 – Arquivo LAB.TXT de entrada para o programa ROMEO.CPP correspondente ao labirinto da figura 21

(labirinto 4 x 4, 0 = célula livre, 1 = obstáculo, 2 = partida, 3 = chegada).

003
006
009
004
006
1 ; 1
4 ; 1
1 ; 2
1 ; 2
4 ; 2
1 ; 2
A

Figura 23 – Arquivo TABM_NEW.TXT contendo os movimentos a serem realizados pela plataforma com base no labirinto da figura 21 e que é gerado pelo programa ROMEO.CPP para posterior leitura pelo programa SEND.CPP.

Uma vez que o programa ROMEO.CPP produziu o arquivo TABM_NEW.TXT com a sequência de movimentos, o passo seguinte é executar o programa MOVEA.ASM no microcontrolador (tarefa realizada através do uso do programa Terminal Emulator, TE.EXE,

executado no microcomputador). Este arquivo, que em sua parte inicial é responsável por receber os dados da porta serial do kit, que está conectada ao microcomputador, fica esperando até que algum caractere seja enviado.

No microcomputador o programa SEND.CPP é então executado. Este programa abre o arquivo TABM_NEW.TXT e envia seus caracteres um a um através da porta serial do microcomputador (que deve estar configurada em concordância com o programa de recepção dos dados no microcontrolador).

É interessante observar que o dado enviado corresponde ao código ASCII do caractere que se deseja transmitir.

À medida que a recepção dos dados é implementada pelo microcontrolador, estes dados recebidos são gravados automaticamente em posições de memória específicas, para serem posteriormente tratados e usados pelas rotinas de execução da trajetória.

Terminada a transmissão dos dados, a execução do programa SEND.EXE no microcomputador é encerrado automaticamente e o programa MOVEA.ASM no microcontrolador trata os 15 primeiros dados recebidos de forma que cada número seja transformado em centena, dezena e unidade da variável correspondente. Feito isso, o programa espera até que a tecla enter do kit seja pressionada para que possa ser iniciada a execução dos movimentos dos motores através dos circuitos de lógica e de potência, conforme apresentado anteriormente.

5.1. Alterações Realizadas no Programa ROMEO.CPP

O programa ROMEO.EXE é um programa que implementa o algoritmo A* para a busca da melhor trajetória a ser realizada pela plataforma num ambiente do tipo labirinto. Este programa foi desenvolvido pelo aluno de graduação do ITA Fabio Eiji Yoshitome [1].

Este trabalho de graduação realizou, no entanto, a inserção de alguns trechos de programa no trabalho acima mencionado, de forma a fazer com que os dados, inicialmente gerados, referentes ao movimento da plataforma, e aqueles referentes às variáveis que indicam à plataforma a quantidade de pulsos necessários, fossem gravados num programa próprio, o TABM_NEW.TXT, para que pudessem, posteriormente, ser transferidos para plataforma.

Para decisão da forma na qual os dados seriam transmitidos foi levada em consideração a simplicidade e versatilidade de transmissão dos dados, de forma a haver o mínimo de erros de aproximação na quantidade de pulsos a serem realizados pela plataforma e de forma a fazermos todo o tratamento dos números em ponto flutuante no programa ROMEO, escrito em TurboC e assim não precisaríamos fazer tal tipo de tratamento na linguagem ASSEMBLY, o que tornaria o desenvolvimento do programa de transmissão bastante complexo e trabalhoso.

O trecho inserido no programa ROMEO.CPP, o qual tornou-se versão 2.0, encontra-se no anexo C deste trabalho e o programa ROMEO.EXE versão 2.0 pode ser encontrado no formato eletrônico também em anexo.

A configuração do código inserido segue a seguinte estrutura:

1. Definição das variáveis que definem o mínimo de pulsos para cada trecho do labirinto pelo usuário do programa.
2. Na subrotina GERAR_MOV_TRAJETÓRIA() o arquivo TABM_NEW.TXT, no qual deverão ser gravados os dados do movimento da plataforma é aberto para leitura e caso já exista não é aberto novamente. Caso não exista, é aberto novamente, mas agora como escrita.
3. São gravadas as variáveis ROT_45, ROT_90, ROT_135, TR_SIDE e TR_DIAG, que correspondem, respectivamente, ao número de pulsos necessários para executar um giro de 45 graus em torno do ponto médio entre os motores, uma rotação de 90 graus e uma de 135 graus em torno do mesmo eixo, uma translação num lado de uma célula do labirinto e uma translação em uma diagonal de uma célula do labirinto.
4. Dependendo do valor da rotação, ou da translação, a ser executada pela plataforma (determinados pelo Algoritmo A*) são determinados os tipos e subtipos do movimento segundo a tabela 7, e os quais são gravados nas variáveis TIPOR e SUBTIPOR, para rotação, e TIPOT e SUBTIPOT, para translação.
5. Tais valores são gravados no arquivo TABM_NEW.TXT.
6. Finalizada a sequência de movimentos é gravada a letra A, indicando o final da tabela de movimentos.
7. O arquivo TABM_NEW.TXT é então, encerrado.

5.2. Programa do Microcontrolador MOVEA.ASM

5.2.1. Funcionamento do Programa de Acionamento da Plataforma

O programa MOVEA, desenvolvido neste trabalho de graduação, responsável por receber os dados referentes aos movimentos da plataforma do microcomputador e a partir destes realizar o acionamento da plataforma, deve ter como saídas quatro variáveis, sendo duas para o acionamento de cada um dos motores, em que uma delas corresponde ao bit SR que identifica o sentido de rotação do motor de passo e a outra corresponde a um Pulso que aciona o motor no sentido especificado. O movimento da plataforma é, então, realizado seguindo a tabela 9, apresentada.

Tabela 9 – Relação dos níveis lógicos dos bits que indicam o sentido de rotação de cada um dos motores e o movimento realizado pela plataforma (saída do programa).

Movimento	Bit de sentido de rotação do motor esquerdo	Bit de sentido de rotação do motor direito
Translação para Frente	1	1
Translação para Trás	0	0
Rotação Horária	1	0
Rotação Anti-horária	0	1

É importante observar que os valores apresentados na tabela 9 correspondem a valores lógicos. Em níveis de tensão, 5V corresponde ao nível lógico 1 (alto) e 0V corresponde ao nível lógico 0 (baixo).

Além disso, para fazer com que o movimento da plataforma seja uma linha reta faz-se necessário que os bits de pulso que acionam os motores sejam sincronizados, no caso, o bit Pulso do motor direito e o bit Pulso do motor esquerdo.

Como entrada do programa MOVEA, rodado no microcontrolador, temos a tabela de movimentos gerada pelo programa ROMEO, TABM_NEW.TXT, contendo cada um dos movimentos a ser realizado pela plataforma, bem como os valores das variáveis que determinam o número de pulsos a ser realizado pela plataforma em cada um dos movimentos, determinados pela tabela 7.

A trajetória a ser realizada pela plataforma é descrita de acordo com um labirinto, como o apresentado na figura 21, em que as dimensões do mesmo são fornecidas pelo usuário no início da execução do programa A* (ROMEO).

Por simplicidade, foi utilizado um labirinto de dimensões quadradas, em que as células equidistam uma das outras na direção horizontal e na direção vertical. A idéia para mapear os possíveis movimentos a serem realizados pela plataforma foi, então, traduzir os movimentos em códigos, no caso, números inteiros (pela facilidade de tratamento deste tipo de número pela linguagem ASSEMBLY).

A idéia foi fazer com que dado o movimento, o microcontrolador já soubesse quantos pulsos seriam necessários para acionar os motores para aquele movimento, bem como qual seria a combinação dos bits de sentido de movimento, já que tais valores já estariam previamente definidos na memória do microcontrolador.

Assim, inicialmente para gerar esta seqüência de movimentos na plataforma, uma tabela de movimentos e os valores dos tamanhos dos pulsos para cada tipo de movimento são copiados na memória do microcontrolador pelo próprio programa de acionamento.

Este programa consiste numa seqüência de iterações em que os números da tabela de movimentos são lidos de dois em dois (tipo e subtipo de cada movimento), como especificado pela tabela 7, e o movimento referente a estes dois números é realizado. O fim do movimento é indicado pelo número 41H, que equivale ao código ASCII da letra A (maiúsculo).

5.2.2. Programa MOVEA.ASM

Este programa, executado na plataforma, foi desenvolvido em linguagem ASSEMBLY, e é responsável por receber os dados via comunicação serial do computador, ao qual a plataforma está conectada, e fazer com que a mesma realize os movimentos dentro do labirinto. O uso do mesmo programa para estas duas funções é necessário devido ao fato do Terminal Emulator (TE.COM), programa que faz a interface de comunicação entre a plataforma e o microcomputador, não ser capaz de transferir dois programas distintos para a plataforma sem que o kit que opera na plataforma seja resetado. Por isso, o programa de recepção de dados na plataforma é parte integrante do programa de execução de trajetória.

O problema proveniente do fato de o kit ser resetado antes que o outro programa seja transferido é devido às posições de memória do kit que estão sendo utilizadas para gravar os dados referentes aos movimentos da plataforma serem também resetadas, por ocasião do reset, fazendo com que tais dados sejam perdidos.

O programa MOVEA.ASM, em linhas gerais, tem como entrada a tabela de movimentos gerada pelo programa ROMEO, cujo nome é TABM_NEW.TXT, e apresentado na figura 23, e inicia com as declarações de todas as variáveis e textos a serem utilizados durante a execução do mesmo. Em seguida sua estrutura pode ser dividida em três partes. A primeira parte constitui uma rotina para leitura dos dados provenientes da porta serial do kit (PARTE 1), em que primeiramente é feita a configuração da porta serial para recepção de dados, que é realizada utilizando uma sub-rotina de interrupção, cujo endereço é indicado pelo vetor de interrupção definido pela biblioteca BASIC-52 disponível no kit AES-51. Esta sub-rotina (INTR), lê os dados da porta serial e os grava numa posição de memória específica e conhecida do kit. E, posteriormente, o programa é deixado em modo de espera para receber outra interrupção, indicando a existência de outro dado a ser recebido pela porta serial. O endereço a partir do qual é gravada a tabela de movimento é o 4500H, sendo os 15 primeiros caracteres correspondentes as variáveis ROT_45, ROT_90, ROT_135, TR_SIDE e TR_DIAG, cujo significado já foi anteriormente discutido, e sendo os movimentos gravados a partir da posição 450FH.

Quando o caractere 'A', indicativo do final da tabela de movimentos, é recebido o programa salta para segunda parte (PARTE 2), que constitui o tratamento dos dados recebidos.

Esta segunda parte trata os 15 primeiros dados recebidos, que correspondem ao número de pulsos para que a plataforma percorra uma das dimensões do labirinto. Ela é

responsável por tomar os dados três a três e transformá-los em dezena, centena ou unidade. A decomposição, por parte do programa ROMEO.CPP e posterior composição destes dados por parte do programa MOVEA.ASM se faz necessário devido à transmissão ser feita byte a byte.

Por fim, a terceira parte do programa (PARTE 3) é aquela que realiza os movimentos propriamente ditos, resgatando valores inteiros previamente gravados na memória do microcontrolador em sequência, cujo significado já foi bastante discutido.

A partir de tais valores, que são utilizados como entrada, o programa envia pulsos para os bits referentes às portas P1.4, P1.5, P1.6 e P1.7, chamados MOTORL, DRIVEL, MOTORR e DRIVER, respectivamente, e que correspondem aos bytes 94H, 95H, 96H e 97H mapeados na memória do microcontrolador, de forma a controlar os motores de passo que acionarão as rodas da plataforma.

A estrutura desta terceira parte é bastante simples: os valores de tipo e subtipo dos movimentos, que estão em sequência na tabela de movimentos, seguindo a estrutura da tabela 8, são comparados aos referentes a cada movimento como mostrado na tabela 7, e, caso uma das condições se verifique, o programa implementa o movimento. Caso contrário, o programa salta para uma mensagem de erro de tipo ou subtipo inválido, apresentada no LCD do kit AES-51, presente na plataforma.

O código do programa MOVEA.ASM impresso e comentado encontra-se no Anexo D, e o arquivo compilado e executado deste programa pode ser encontrado no formato eletrônico, também em anexo a este trabalho de graduação.

5.3. Programa de Transmissão de Dados

O programa responsável por enviar dados para a plataforma móvel através da porta de comunicação serial do microcomputador é o SEND.CPP.

Este programa foi desenvolvido e compilado utilizando a linguagem de programação TURBO C versão 3.0 executado em sistema operacional DOS.

A linguagem de programação TURBO C versão 3.0 foi escolhida levando-se em consideração principalmente o fato de o programa inicial, que deu origem a esta bolsa, ROMEO.CPP, ter sido escrito nesta linguagem, o que facilitaria a junção dos dois programas em um único, caso posteriormente fosse desejado.

Além disso, o ambiente DOS é conveniente, neste caso, devido ao Terminal Emulator (TE.COM), programa fornecido pelo fabricante do kit AES-51 que realiza a interface de comunicação entre o microcomputador e o microcontrolador do kit, rodar somente em ambiente DOS.

Como entrada para o programa SEND.CPP é necessário o arquivo contendo a tabela de movimentos especificada anteriormente, o TABM_NEW.TXT, e de uma forma geral, este programa transfere os dados presentes no arquivo para a porta serial.

O programa SEND.CPP possui três blocos principais em sua estrutura geral, sendo constituído por três sub-rotinas e uma rotina principal. O programa principal apenas chama as sub-rotinas, nas quais todo o código do programa está inserido.

A primeira sub-rotina, SETUP_SERIAL, é responsável por configurar a transmissão de dados, ajustando o baud rate para 9600, e a palavra de transmissão para 8 bits, sem bits de paridade e um stop bit, como apresentado no capítulo referente à comunicação serial.

Estas configurações são realizadas utilizando-se o comando `outportb(<endereço>, <dado>)` que envia o <dado> para o <endereço> especificado.

E a segunda sub-rotina é estruturada da seguinte forma: primeiro, o arquivo de entrada TABM_NEW.TXT é aberto como somente leitura. Caso não haja erro na abertura, o arquivo é lido e sendo o byte então lido um byte válido (todos os números de 0 a 9 e o caractere A), este é escrito na tela do PC de forma a ser facilmente compreendido pelo operador, e posteriormente é transferido para porta serial (através da terceira sub-rotina), caso contrário, o mesmo é descartado.

A terceira sub-rotina, que envia o dado para o kit AES-51, presente na plataforma, possui a seguinte estrutura: verifica se o buffer do teclado está vazio, o que indica que uma nova transmissão pode ser realizada. Caso contrário o programa aguarda através de um laço “while”, até que seja possível a transmissão. Caso afirmativo, a transmissão é feita utilizando o comando `outportb` para o endereço da porta serial COM1, como definido no capítulo referente à comunicação serial.

O código fonte do programa SEND.CPP encontra-se em anexo, na forma impressa (Anexo D), e o arquivo compilado, bem como o executável resultante na forma eletrônica, também em anexo.

6. Testes de Dimensões

Tendo sido implementados todos os programas que realizam o movimento da plataforma, bem como o hardware necessário para o movimento da mesma, foram realizados os testes para determinar o número de pulsos necessários para fazer com que a plataforma realize cada um dos movimentos: rotação de 45 graus, rotação de 90 graus, rotação de 135 graus, translação em um lado de uma célula do labirinto, suposto com dimensões quadradas, e translação numa diagonal de uma célula do mesmo.

Inicialmente, foram especificadas as dimensões cabíveis da plataforma, sendo para tanto realizado um conjunto de 40 medidas para cada uma das dimensões. As medidas obtidas são aquelas apresentadas na tabela 10, sendo estas realizadas utilizando uma régua Trident, Indústria Brasileira, 40cm e, portanto, com uma precisão de 0,05cm.

Tabela 10 – Resultados obtidos das medições realizadas na plataforma representada pela figura 24.

Medida	C (cm)	L (cm)	LT (cm)
1	31,75	23,40	26,60
2	31,80	23,40	26,60
3	31,75	23,40	26,75
4	31,80	23,40	26,70
5	31,75	23,40	26,80
6	31,80	23,40	26,75
7	31,75	23,40	26,80
8	31,80	23,40	26,70
9	31,80	23,40	26,80
10	31,80	23,40	26,70
11	31,80	23,35	26,70
12	31,80	23,30	26,70
13	31,80	23,30	26,70
14	31,80	23,30	26,65
15	31,80	23,30	26,70
16	31,80	23,35	26,70
17	31,80	23,30	26,75
18	31,80	23,30	26,70
19	31,80	23,30	26,75
20	31,80	23,30	26,70
21	31,80	23,30	26,70
22	31,80	23,30	26,75
23	31,80	23,30	26,70
24	31,80	23,30	26,70
25	31,80	23,30	26,70
26	31,80	23,30	26,70
27	31,80	23,30	26,70

28	31,80	23,30	26,70
29	31,80	23,35	26,70
30	31,80	23,35	26,70
31	31,80	23,35	26,75
32	31,85	23,35	26,70
33	31,85	23,35	26,70
34	31,80	23,35	26,70
35	31,80	23,35	26,70
36	31,75	23,40	26,70
37	31,80	23,40	26,70
38	31,80	23,40	26,70
39	31,80	23,40	26,70
40	31,75	23,40	26,65

As dimensões da plataforma referentes à tabela 10 estão representadas na figura 24.

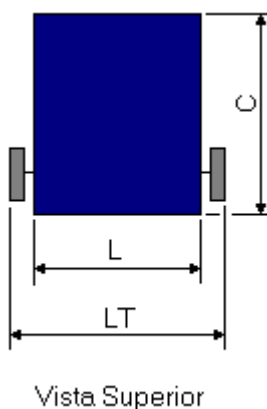


Figura 24 – Figura representativa das dimensões da plataforma.

A partir destes dados, foram calculados as médias e os desvios padrão para cada dimensão, segundo as equações (7) e (8), respectivamente:

$$Media = \frac{\sum_{i=1}^N (Medida_i)}{N} \quad (7)$$

Em que , N corresponde ao número de medidas realizadas.

$$\sigma = \sqrt{\frac{\sum_{i=1}^N (\delta x_i)^2}{N-1}} \quad (8)$$

Onde, σ é o símbolo atribuído para o desvio padrão;

δx_i é a diferença entre o cada medição i e a média das medições;

N corresponde ao número de medições totais realizadas.

Os valores obtidos calculados estão apresentados na tabela 11.

Tabela 11 – Resultados calculados para as dimensões da plataforma móvel.

Cálculo	C (cm)	L (cm)	LT (cm)
Média	31,80	23,35	26,71
Desvio Padrão	0,02	0,04	0,04

Além disso, foram realizadas medidas do deslocamento da plataforma quando a mesma é submetida a uma quantidade fixa de pulsos. Foram realizadas quarenta medidas para cada uma das distâncias. Os valores obtidos encontram-se na tabela 12, na qual as 20 primeiras medidas correspondem a medidas realizadas na roda direita, para deslocamento longitudinal e as 20 últimas, a medidas realizadas na roda esquerda. Em relação às medidas de rotação, as 20 primeiras correspondem a medidas realizadas no sentido horário e as 20 últimas a medidas realizadas no sentido anti-horário.

Tabela 12 - Tabela referente aos valores medidos para os deslocamentos da plataforma.

Medida	Deslocamento Longitudinal para 10 pulsos (cm)	Rotação para 15 pulsos (°)
1	0,20	7,5
2	0,40	7,5
3	0,50	8,0
4	0,50	8,0
5	0,50	7,5
6	0,50	8,0
7	0,35	8,0
8	0,30	8,0
9	0,30	7,5
10	0,50	8,0
11	0,30	8,0
12	0,50	8,0
13	0,40	8,0
14	0,35	7,5
15	0,20	8,0
16	0,55	8,0
17	0,40	8,0
18	0,25	8,0
19	0,25	8,0
20	0,30	8,0
21	0,50	8,0
22	0,45	8,0
23	0,45	8,0
24	0,30	8,0

25	0,40	8,0
26	0,20	8,0
27	0,40	8,0
28	0,25	8,0
29	0,40	8,0
30	0,20	8,0
31	0,50	8,0
32	0,50	8,0
33	0,10	8,0
34	0,45	8,0
35	0,50	8,0
36	0,30	8,0
37	0,40	8,0
38	0,50	8,0
39	0,25	8,0
40	0,30	8,0

As médias e desvios padrão das medidas realizadas foram calculadas utilizando as equações 7 e 8, apresentadas anteriormente. Os valores calculados encontram-se na tabela 13.

Tabela 13 – Valores calculados da média e desvio padrão para as grandezas que determinam os deslocamentos reais da plataforma para cada pulso.

Cálculo	Deslocamento Longitudinal para 10 pulso (cm)	Rotação para 15 pulso (°)
Média	0,37	8,0
Desvio Padrão	0,12	0,2

Para as dimensões de um labirinto quadrado, temos os seguintes valores medidos, utilizando a mesma régua de medida, para cada uma das células do mesmo: 30cm x 30cm.

Assim, utilizando regra de três, para cada uma das variáveis do labirinto temos as seguintes quantidades de pulso:

ROT_45: 84,38 pulsos → 85 pulsos

ROT_90: 168,75 pulsos → 169 pulsos

ROT_135: 253,13 pulsos → 253 pulsos

TR_SIDE: 810,81 pulsos → 811 pulsos

TR_DIAG: 1146,66 pulsos → 1147 pulsos

Observa-se aqui que para gerar os valores destas variáveis seriam necessários dois bytes. O que fica como sugestão para trabalhos posteriores.

7. Resultados

Com a plataforma pronta: em funcionamento, foram realizados alguns testes para verificar seu funcionamento e desempenho. Foram adquiridos os sinais gerados para a mesma em diferentes situações, nas quais foram testados os bits de Pulso e SR que são enviados para os motores, com e sem carga, com e sem a presença de interrupção que detecta a colisão.

Nestes testes, podemos perceber que os níveis de tensão enviados para os motores de passo, são níveis 5V ou 0V, havendo um pequeno ruído em torno das medidas, mas que em nada afetam o bom funcionamento da plataforma, devido em parte pelo sinal lido ter sido gerado dentro do chip, e portanto tal sinal carrega as flutuações provenientes deste processo, e parte devido ao erro de conversão ocorrido no conversor A/D.

O erro de conversão pode ainda ser estimado considerando-se que os níveis de tensão variam entre 0 e 20V (de acordo com a escala adotada no osciloscópio utilizado, especificado a seguir) e, considerando-se que temos 256 níveis diferentes para armazenar os valores provenientes da conversão (devido aos 8 bits de resolução, que é a palavra de conversão utilizada pelo osciloscópio). Assim,

$$Erro = \frac{20V}{256niveis} \cong 0,078 \quad (9)$$

Ou seja, temos quase 0,1V de erro devido à conversão.

Pode-se notar ainda, nos gráficos que se seguem, 4 e 5, e 7 e 8, que a interrupção atua de forma interromper de fato a trajetória programada, deixando os níveis de tensão travados no nível em que se encontravam imediatamente antes da colisão, e, ao retornar da mesma, tais níveis fazem com que a trajetória simplesmente continue o movimento do ponto em que parou.

Além disso, o fato de os motores estarem sendo adicionados, o que pode ser observado nos testes 6 a 8, em contraposição àqueles nos quais os motores não estavam inseridos, testes 1 a 5, não fez com que os níveis de tensão fossem, de alguma forma, perceptivelmente alterados.

O material utilizado para teste foi um osciloscópio capaz de adquirir os dados através do computador e, portanto, haver, neste caso, a possibilidade de gravarmos a tela gerada no osciloscópio e então plotarmos os gráficos gerados utilizando, por exemplo, o programa MATLAB.

- Osciloscópio utilizado foi o HM407 (Analog / Digital Scope) da HAMEG.
- PC 386 166MHz – rodando o software SP107E (versão FC 3.22 DG 1.63) da HAMEG.
- Computador PC AT rodando os demais programas, já mencionados ao longo deste relatório, e em comunicação com a plataforma.
- Plataforma propriamente dita, com todos os seus componentes.

Tais testes foram realizados com base no labirinto da figura 21, o qual foi utilizado em todos os exemplos ao longo deste relatório.

Relembrando: com as condições iniciais de orientação inicial da plataforma de -90° e com os custos de translação e de rotação iguais a 1, requisitos para execução do programa A*, de procura da trajetória, os passos a serem realizados pela plataforma, de acordo com o arquivo TABM_NEW.TXT gerado, representado na figura 23, são os apresentados na tabela 14 a seguir.

Tabela 14 – Tabela dos movimentos realizados pela plataforma nos testes que se seguem de 1 a 8.

Movimento	Número de Pulsos	MOTORL (Nível Lógico)	MOTORR (Nível Lógico)	Tipo do Movimento Realizado
1	4	1	1	Translação para frente - lado
2	3	1	0	Rotação Anti-Horária 45°
3	6	1	1	Translação para frente – diagonal
4	6	1	1	Translação para frente – diagonal
5	6	1	0	Rotação Anti-Horária 90°
6	6	1	1	Translação para frente – diagonal

Os testes realizados, seguidos dos seus respectivos resultados, estão apresentados a seguir. Nestes testes os gráficos foram obtidos com o auxílio do EXCEL, do pacote OFFICE 2000 da Microsoft, e do MATLAB versão 6.1.

Teste 1:

Este teste foi realizado monitorando-se os bits MOTORL e DRIVEL, com a plataforma operando sem carga, ou seja, sem os motores (apenas com o kit AES-51 acionado), e sem pedido de interrupção de colisão. Os parâmetros, do osciloscópio, utilizados são os que seguem:

- Eixo horizontal: 2s / divisão.
- Eixo vertical do canal 1: 2V / divisão – monitorando o bit MOTORL.
- Eixo vertical do canal 2: 2V / divisão – monitorando o bit DRIVEL.

O que se pretendeu com a realização deste teste foi atestar que os níveis de tensão, sem carga, estavam sendo gerados satisfatoriamente pelo circuito do kit AES-51, mais que isso se pretendeu verificar que os pulsos provenientes do software são exatamente aqueles que deveriam estar sendo gerados para que a plataforma seguisse a trajetória especificada pelo labirinto teste, o mesmo representado na figura 21.

O resultado encontra-se na figura 25 a seguir, na qual se observa que todos os níveis e inversões estão de acordo com o esperado.

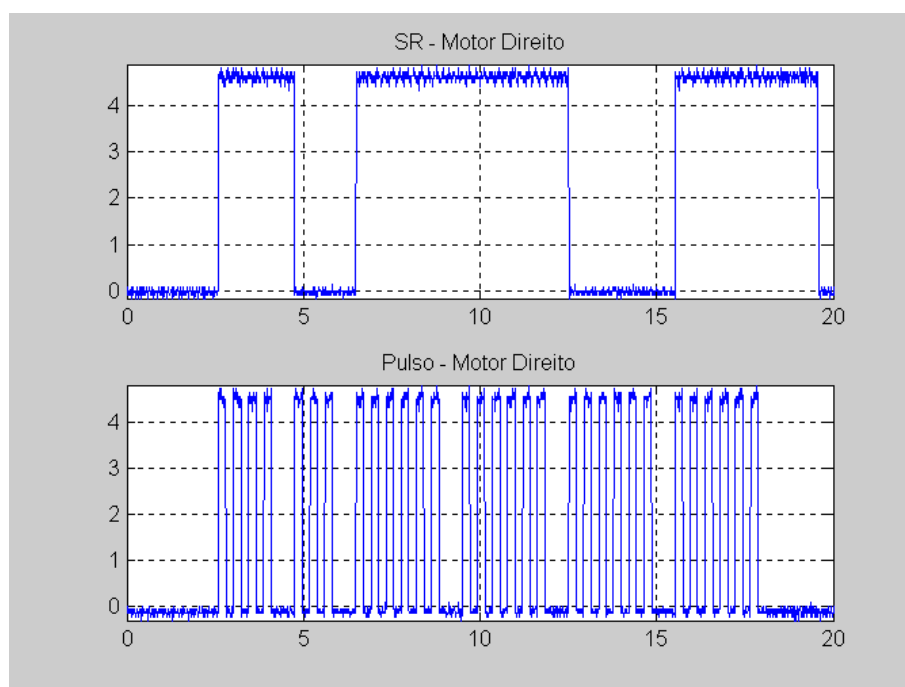


Figura 25 – Resultado do monitoramento da plataforma relativo ao teste 1. Condições: plataforma operando sem carga e sem pedido de interrupção.

Teste 2:

Este teste foi realizado monitorando-se os bits MOTORR e DRIVER, com a plataforma operando sem carga, ou seja, sem os motores (apenas com o kit AES-51 acionado), e sem pedido de interrupção de colisão. Os parâmetros, do osciloscópio, utilizados são os que seguem:

- Eixo horizontal: 2s / divisão.
- Eixo vertical do canal 1: 2V / divisão – monitorando o bit MOTORR.
- Eixo vertical do canal 2: 2V / divisão – monitorando o bit DRIVER.

Analogamente ao que foi realizado para o motor direito, teste 1, realizou-se para o motor esquerdo neste teste, pretendendo-se atestar que os níveis de tensão, sem carga, estavam sendo gerados satisfatoriamente pelo circuito do kit AES-51 e pretendendo-se verificar que os pulsos provenientes do software são exatamente aqueles que deveriam estar sendo gerados para que a plataforma seguisse a trajetória especificada pelo labirinto teste.

O resultado encontra-se na figura 26 a seguir, na qual se observa que todos os níveis e inversões de sinal estão de acordo com o esperado.

O resultado encontra-se na figura 26 a seguir.

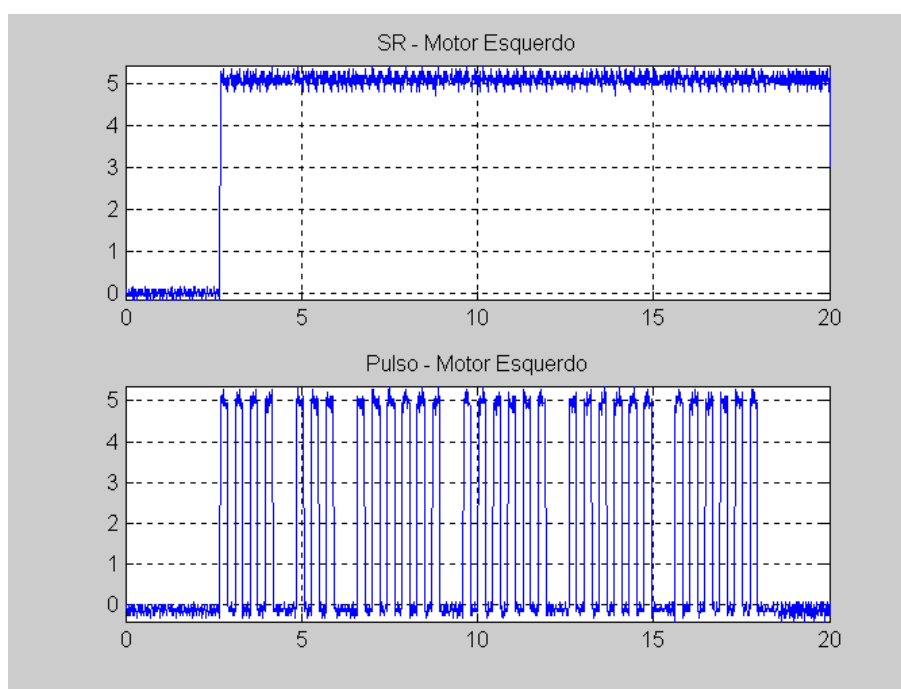


Figura 26 – Resultado do monitoramento da plataforma relativo ao teste 2. Condições: plataforma operando sem carga e sem pedido de interrupção.

Teste 3:

Este teste foi realizado monitorando-se os bits DRIVEL e DRIVER, com a plataforma operando sem carga, ou seja, sem os motores (apenas com o kit AES-51 acionado), e sem pedido de interrupção de colisão. Os parâmetros, do osciloscópio, utilizados são os que seguem:

- Eixo horizontal: 2s / divisão.
- Eixo vertical do canal 1: 2V / divisão – monitorando o bit DRIVEL.
- Eixo vertical do canal 2: 2V / divisão – monitorando o bit DRIVER.

Observou-se neste teste que os níveis do sinal Pulso enviados tanto para o motor esquerdo quanto para o motor direito estão sincronizados, de forma a não comprometer no resultado final, ou seja, não provoca o desvio da trajetória realizada em relação à trajetória que deveria ser realizada de acordo com o planejado via software.

O resultado obtido encontra-se na figura 27 a seguir.

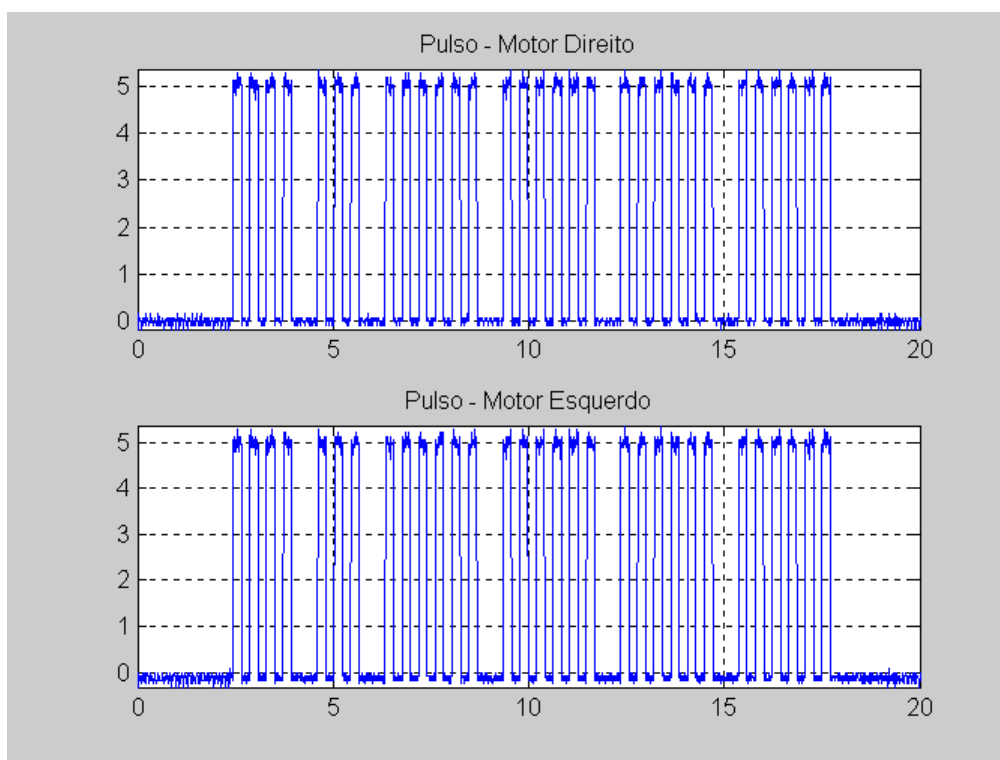
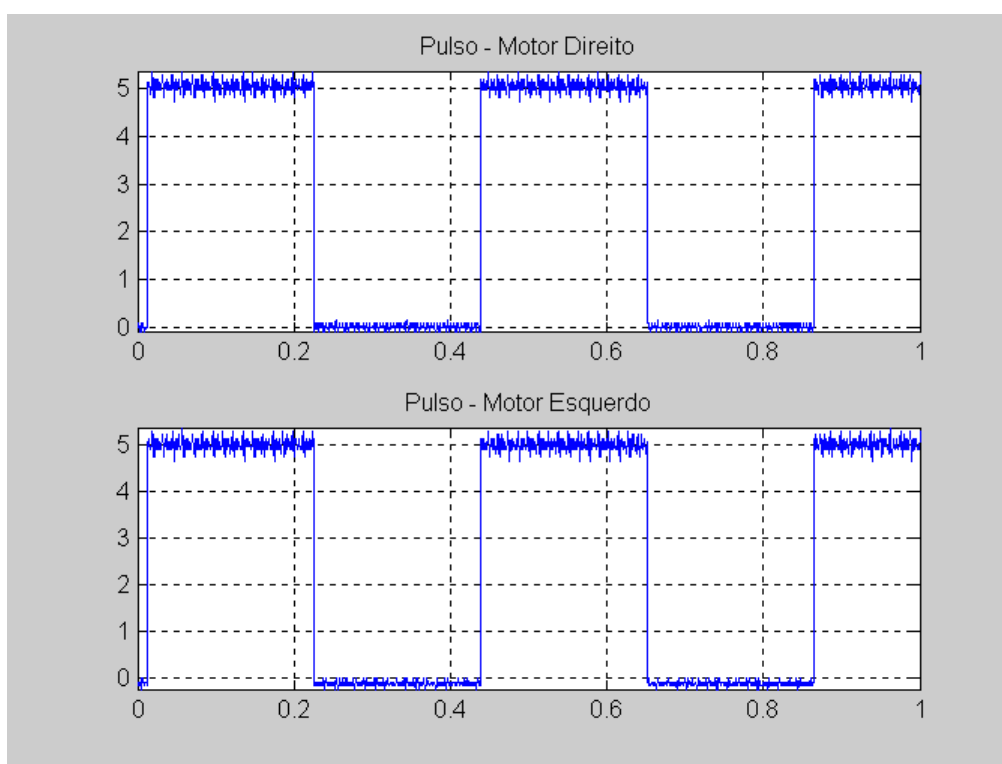


Figura 27 – Resultado do monitoramento da plataforma relativo ao teste 3. Condições: plataforma operando sem carga e sem pedido de interrupção.

Neste teste ainda, realizou-se uma segunda aquisição de sinais, utilizando diferentes escalas do osciloscópio, com o objetivo de verificar com um maior detalhamento os sinais de pulso enviados para os motores esquerdo e direito. Assim, aqui foi realizado o monitoramento dos bits DRIVEL e DRIVER, com a plataforma operando sem carga, ou seja, sem os motores (apenas com o kit AES-51 acionado), e sem pedido de interrupção de colisão, utilizando os parâmetros que se seguem:

- Eixo horizontal: 100ms / divisão.
- Eixo vertical do canal 1: 2V / divisão – monitorando o bit DRIVEL.
- Eixo vertical do canal 2: 2V / divisão – monitorando o bit DRIVER.

O resultado obtido encontra-se na figura 28 a seguir.



**Figura 28 – Resultado do monitoramento da plataforma relativo a um zoom do sinal obtido no teste 3.
Condições: plataforma operando sem carga e sem pedido de interrupção.**

Teste 4:

Este teste foi realizado monitorando-se os bits MOTORL e Int1, com a plataforma operando sem carga, ou seja, sem os motores (apenas com o kit AES-51 acionado), e com pedido de interrupção de colisão. Os parâmetros, do osciloscópio, utilizados são os que seguem:

- Eixo horizontal: 2s / divisão.
- Eixo vertical do canal 1: 2V / divisão – monitorando o bit MOTORL.
- Eixo vertical do canal 2: 2V / divisão – monitorando o bit Int1.

O resultado encontra-se na figura 29, a seguir, em que a interrupção é solicitada através do sensor de contato e posteriormente cancelada. Observando-se o resultado obtido pode-se observar que o pedido de interrupção é realizado num instante posterior ao da subida de nível do bit MOTORL, ficando tal bit em nível alto durante um intervalo de tempo maior que aquele durante o qual ficaria caso a interrupção não tivesse sido acionada (comparar com o teste 1). Este intervalo maior corresponde ao tempo de acionamento da interrupção e posterior cancelamento da mesma (o que é realizado pressionando-se enter por duas vezes consecutivas no teclado do kit, como apresentado no capítulo referente à colisão).

A mesma interrupção monitorada em paralelo com o bit DRIVEL encontra-se no teste 5.

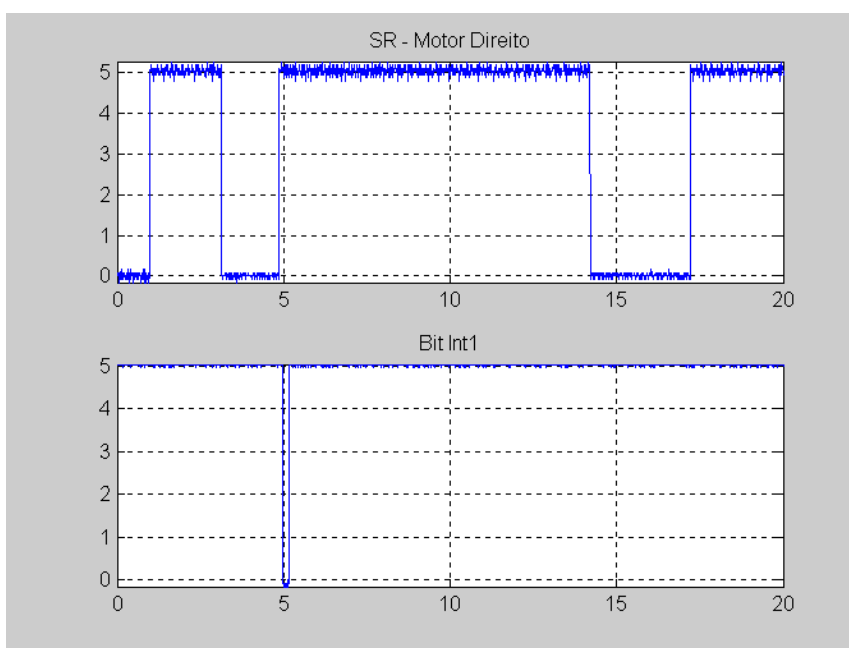


Figura 29 – Resultado do monitoramento da plataforma relativo ao teste 4. Condições: plataforma operando sem carga e com pedido de interrupção.

Teste 5:

Este teste foi realizado monitorando-se os bits DRIVEL e Int1, com a plataforma operando sem carga, ou seja, sem os motores (apenas com o kit AES-51 acionado), e com pedido de interrupção de colisão. Os parâmetros, do osciloscópio, utilizados são os que seguem:

- Eixo horizontal: 2s / divisão.
- Eixo vertical do canal 1: 2V / divisão – monitorando o bit DRIVEL.
- Eixo vertical do canal 2: 2V / divisão – monitorando o bit Int1.

O resultado encontra-se na figura 30, a seguir, em que a interrupção é solicitada através do sensor de contato e posteriormente abortada através do teclado do kit AES-51. Este resultado concorda com o anterior, teste 4, no tempo.

Aqui pode ser visto ainda que o número de pulsos continua concordando com o esperado, apresentado na tabela 14. O que se pode observar aqui é o retorno da rotina de interrupção, no qual observa-se que a sequência de pulsos não é prejudicada com este pedido de interrupção (observar ainda que o pedido de interrupção é feito independentemente do estado do bit DRIVEL ou MOTORL, teste 4).

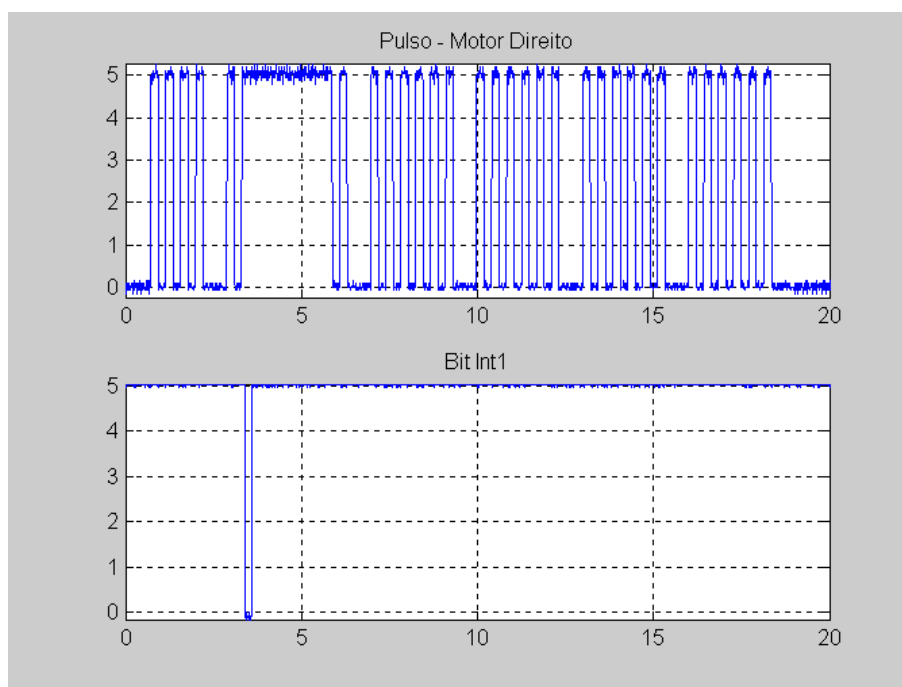


Figura 30 – Resultado do monitoramento da plataforma relativo ao teste 5. Condições: plataforma operando sem carga e com pedido de interrupção.

Teste 6:

Avaliando-se agora o desempenho da plataforma como um todo, ou seja, incluindo-se os motores de passo na aquisição dos sinais, realizou-se este teste fazendo-se o monitoramento dos bits DRIVEL e DRIVER, com a plataforma operando agora com carga, e sem pedido de interrupção de colisão. Os parâmetros, do osciloscópio, utilizados são os que seguem:

- Eixo horizontal: 2s / divisão.
- Eixo vertical do canal 1: 2V / divisão – monitorando o bit MOTORL.
- Eixo vertical do canal 2: 2V / divisão – monitorando o bit DRIVEL.

Como já sabemos que os bits são gerados de acordo com a trajetória especificada, basta avaliarmos a redução do sinal, se é que a mesma ocorre, decorrente da inserção dos circuitos de acionamento dos motores e de potência para os motores.

E, observa-se que os níveis não foram prejudicados com esta inserção, indicando o bom projeto dos circuitos de acionamento e de potência dos motores de passo.

O resultado pode ser visto na figura 31 a seguir.

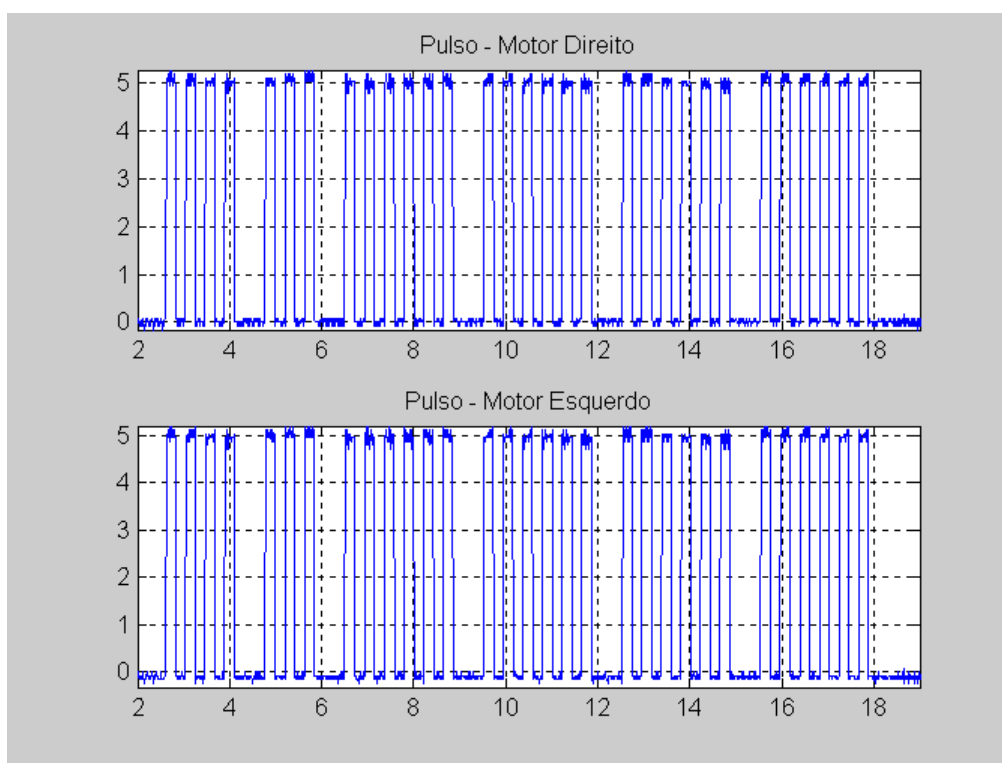


Figura 31 – Resultado do monitoramento da plataforma relativo ao teste 6. Condições: plataforma operando com carga e sem pedido de interrupção.

Ampliando-se a escala de tempo do osciloscópio, pode-se observar com maior detalhe o sincronismo entre os sinais de pulso enviados aos motores esquerdo e direito.

As condições correspondem as mesmas daquelas realizadas anteriormente, sendo realizado o monitoramento dos bits DRIVEL e DRIVER, com a plataforma operando com carga, ou seja, com os motores de passo, e sem pedido de interrupção de colisão. Os parâmetros, do osciloscópio, utilizados são os que seguem:

- Eixo horizontal: 100ms / divisão.
- Eixo vertical do canal 1: 2V / divisão – monitorando o bit DRIVEL.
- Eixo vertical do canal 2: 2V / divisão – monitorando o bit DRIVER.

O resultado pode ser visto na figura 32 a seguir.

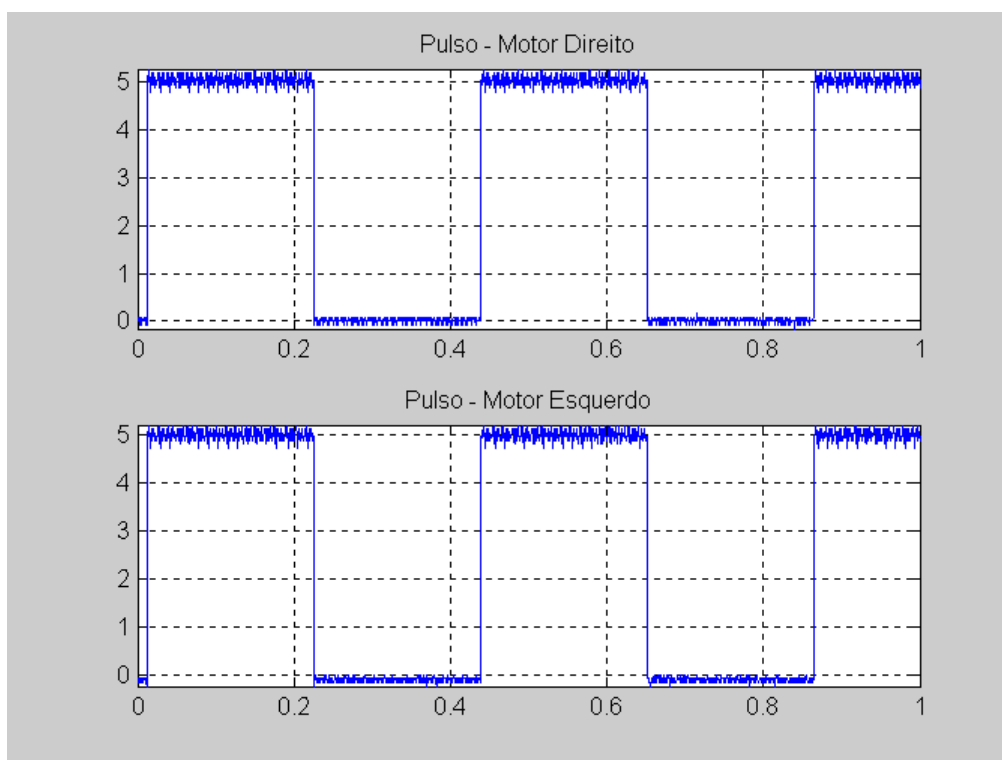


Figura 32 – Resultado do monitoramento da plataforma relativo a um zoom do sinal obtido no teste 6.
Condições: plataforma operando com carga e sem sinal de interrupção.

Teste 7:

Analogamente, com o objetivo de verificar que os bits MOTORL e MOTORR continuam fornecendo sinais lógicos com nível 5V e 0V, além de não ter havido alterações quanto a sequência de níveis lógicos para que a plataforma siga a sequência de passos para completar o labirinto de forma satisfatória, como apresentado na tabela 14.

Assim, este teste foi realizado monitorando-se os bits MOTORL e MOTORR, com a plataforma operando com carga, ou seja, com os motores de passo, e sem pedido de interrupção de colisão. Os parâmetros, do osciloscópio, utilizados são os que seguem:

- Eixo horizontal: 2s / divisão.
- Eixo vertical do canal 1: 2V / divisão – monitorando o bit MOTORL.
- Eixo vertical do canal 2: 2V / divisão – monitorando o bit MOTORR.

O resultado encontra-se na figura 33 a seguir.

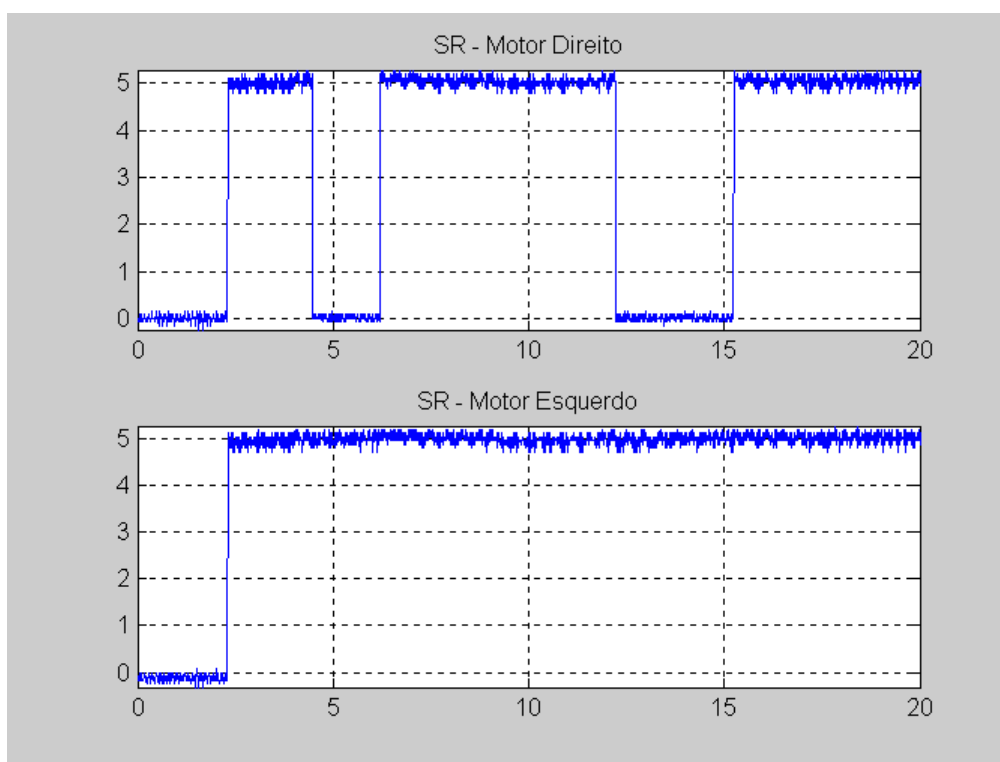


Figura 33 – Resultado do monitoramento da plataforma relativo ao teste 7. Condições: plataforma operando com carga e sem sinal de interrupção.

Teste 8:

Este teste é composto por dois monitoramentos distintos, relacionados ao pedido de colisão com a plataforma operando com carga, um monitorando o pedido de interrupção em paralelo com o bit MOTORL e o outro monitorando o pedido de interrupção em paralelo com o bit DRIVEL. Assim, na figura 34, foi realizado o monitoramento dos bits MOTORL e Int1, com a plataforma operando com carga, ou seja, com os motores de passo, e com pedido de interrupção de colisão. Os parâmetros, do osciloscópio, utilizados são os que seguem:

- Eixo horizontal: 2s / divisão.
- Eixo vertical do canal 1: 2V / divisão – monitorando o bit MOTORL.
- Eixo vertical do canal 2: 2V / divisão – monitorando o bit Int1.

No resultado obtido observa-se que quando a colisão é acionada pelos sensores de contato e abortada através do teclado do kit AES-51, como descrito na seção 4.2.2 deste trabalho de graduação, a sequência anterior dos bits permanece inalterada, ou seja, concordando ainda com o especificado pela tabela 14, a menos de um delay no tempo referente ao tempo necessário para a interrupção ser acionada através dos sensores e posteriormente cancelada através do teclado do kit.

E, na figura 35, temos o monitoramento dos bits DRIVEL e Int1, com a plataforma operando com carga, ou seja, com os motores de passo, e com pedido de interrupção de colisão. Os parâmetros, do osciloscópio, utilizados neste caso são os que seguem:

- Eixo horizontal: 2s / divisão.
- Eixo vertical do canal 1: 2V / divisão – monitorando o bit DRIVEL.
- Eixo vertical do canal 2: 2V / divisão – monitorando o bit Int1.

A partir deste resultado, assim como em relação ao do monitoramento do bit MOTORL em paralelo com o pedido de interrupção, verifica-se que a inclusão dos motores de passo e seus circuitos de acionamento e de potência em nada interferem no bom funcionamento da plataforma ao longo do labirinto.

Analogamente ao observado no teste 3, observa-se aqui que os bits MOTORL e DRIVEL, em nada influenciam no pedido de interrupção. Isto pode ser observado nos gráficos, na medida em que durante o acionamento da Int1, não temos variações nestes outros bits.

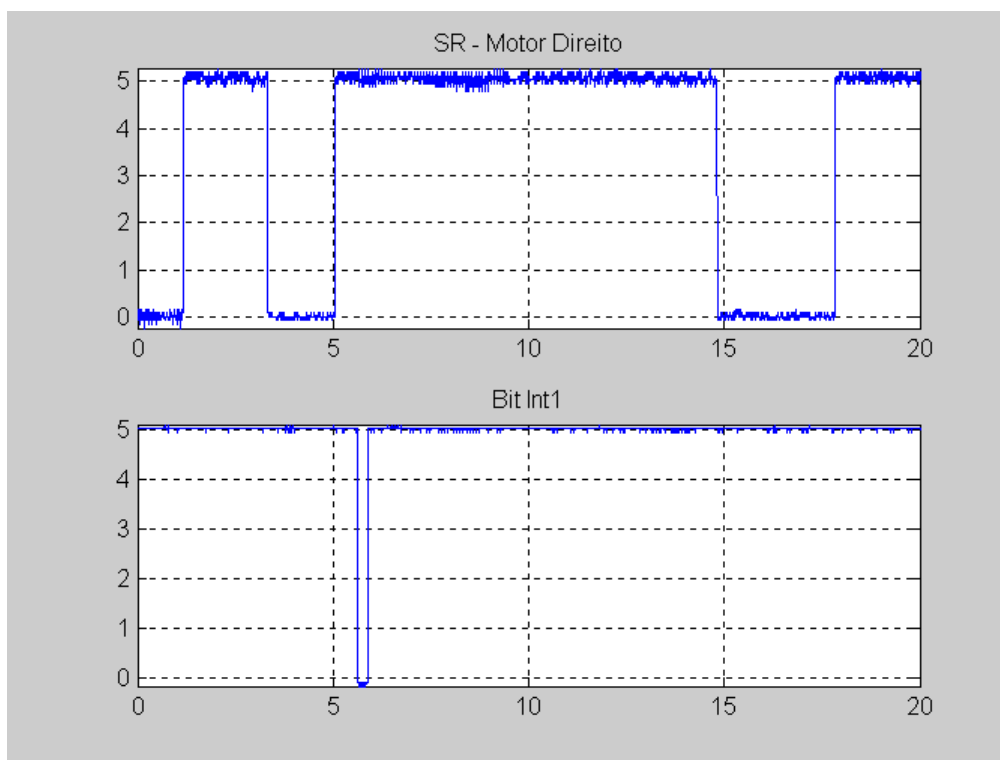


Figura 34 – Resultado do monitoramento da plataforma relativo ao teste 8, do monitoramento dos bits MOTORL e Int1. Condições: plataforma operando com carga e com pedido de interrupção.

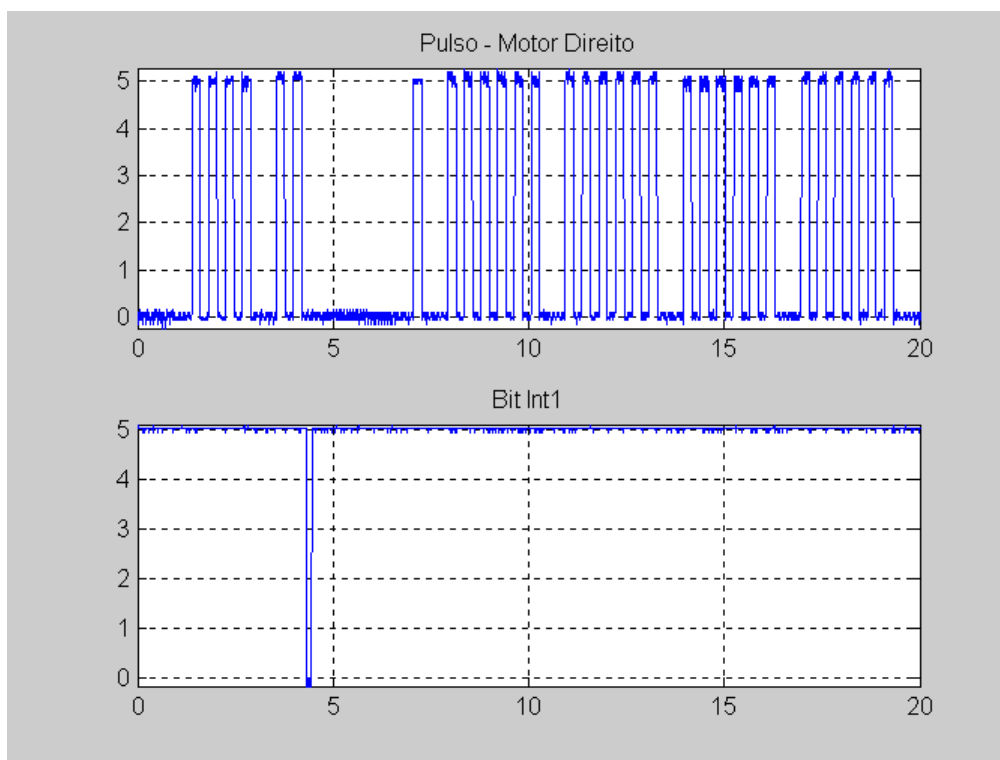


Figura 35 – Resultado do monitoramento da plataforma relativo ao teste 8, do monitoramento dos bits MOTORL e Int1. Condições: plataforma operando com carga e com pedido de interrupção.

8. Discussão dos Resultados

Nota-se que, nos gráficos obtidos, os sinais gerados são bastante satisfatórios para garantir o bom funcionamento da plataforma e um bom desempenho da mesma.

É observado nestas medidas um ruído que faz com que os níveis de tensão flutuem em torno dos valores +5V e 0V. Este ruído é devido, em parte, a flutuações na própria placa do kit. E, como é utilizada uma escala que amplifica os sinais para que os mesmos possam ser visualizados satisfatoriamente pelo operador, os ruídos aparentam ser bastante grandes. Além disso, outra parte do erro é devido à conversão A/D utilizada para adquirir os dados do osciloscópio para o computador.

Nota-se ainda que quando adicionamos a carga, ou seja, os motores de passo e a eletrônica de potência, ao circuito, o sinal gerado permanece praticamente inalterado, indicando que o circuito de potência foi bem projetado e por isso, não exerce influências visíveis ao funcionamento da plataforma.

Quanto a existência das interrupções, verifica-se que no instante em que ocorrem, o programa é interrompido, sendo interrompidos todos os bits de sinais: MOTORL, DRIVEL, MOTORR e DRIVER (testes 4 e 5, sem carga, e 8, com carga). E, por ocasião da saída da rotina de interrupção, os valores são restaurados também satisfatoriamente, como esperado, fazendo a plataforma seguir sua trajetória original, bem como programado.

9. Conclusão e Comentários Finais

Os testes realizados comprovam e validam o trabalho realizado, e portanto, o mesmo, desta forma, atingiu satisfatoriamente o objetivo proposto: desenvolvimento de um software capaz de tornar uma plataforma móvel autônoma utilizando para isso um microcontrolador da família 8051, o 80C32, além de implementar sensores que a tornem inteligente no sentido de detectar uma colisão, caso esta venha a existir.

Vale salientar aqui ainda que o desenvolvimento do hardware da plataforma contou com o apoio dos professores orientadores deste trabalho de graduação.

Salienta-se aqui que, a medida que o trabalho evoluiu, novas possibilidades para aperfeiçoamento da plataforma surgiram, as quais ficam como sugestão para trabalhos posteriores:

- Desenvolvimento do programa MOVEA.ASM utilizando 2 bytes para cada uma das variáveis que armazenam os números de pulsos ROT_45, ROT_90, ROT_135, TR_SIDE e TR_DIAG, para que trajetórias maiores possam ser desenvolvidas.
- Realizar comando e monitoramento da plataforma via Internet.
- Implementação de um rádio transmissor-receptor para que a plataforma possa trocar dados com uma estação-base em tempo real, ou aumento da memória e capacidade de processamento do kit AES-51, para que o desenvolvimento da melhor trajetória possa ser feito em tempo real na própria plataforma.
- Alteração do programa que implementa o movimento da plataforma de forma que o sinal resultante enviado aos motores seja mais suave, reduzindo assim o escorregamento das rodas tracionadas.
- Implementação de sensores que detectem os obstáculos em volta da plataforma e os identifiquem (quanto ao tamanho principalmente) para que se possa realizar um mapeamento do ambiente no qual a plataforma está inserida, e assim, um controle em tempo real da mesma.
- Implementação de sensores externos a plataforma que permitam a mesma corrigir sua trajetória ao longo de um determinado labirinto previamente programado.

Desta forma, este é um trabalho ainda em andamento, que constitui uma continuação do trabalho de graduação do aluno Fabio Eiji Yoshitome [1], com possibilidades de serem implementados diversos outros tópicos de melhoramento na plataforma móvel autônoma.

É importante lembrar aqui que este trabalho acadêmico possui bastantes aplicações que podem ser aplicadas ao meio industrial, indo desde projetos complexos como:

- Implementação de dispositivos que tornem os veículos inteligentes, de forma que tais estes possam ser dirigidos por motoristas automáticos dispensando o ser humano para tal papel.

até projetos menos complexos e igualmente úteis para melhorar o conforto do homem:

- Desenvolvimento de aspiradores eletrônicos capazes de, por exemplo, aspirar uma casa sem o auxílio da dona-de-casa, tornando-lhe a vida mais cômoda.

Por fim, gostaria de agradecer mais uma vez à FAPESP que concedeu os incentivos iniciais para o desenvolvimento deste trabalho.

10. Referências Bibliográficas

10.1. Literatura

- [1] YOSHITOME, F.E., *Planejamento de Trajetórias para Robôs Móveis*, Trabalho de Graduação, Divisão de Engenharia Eletrônica, ITA, São José dos Campos, 1997.
- [2] *AES-51 Computer Emdedded Controller, User's and Language Manual*, Advanced Educational Systems, Califórnia, 1996.
- [3] SCHILDT, H., *Borland C++: Completo e Total*, Makron Books do Barsil Editora Ltda, São Paulo, 1998.
- [4] NICOLOSI, D. E. C., *Microcontrolador 8051 Detalhado*, Ed. Érica, São Paulo, 2000.
- [5] FITZGERALD, A.E., KINGSLEY, C. Jr., KUSKO, A., *Electric Machinery*, International Student Edition, McGraw-Hill, Japão, 1971.
- [6] PIERI, E., SCHOLZE, R., *Assembler - Aprenda como Programar o seu PC*, Editora Érica Ltda., São Paulo, 1993.
- [7] *NMB Minebea Co. Ltda., Stepping Motor*, Catálogo de Motores da Astrosyn.
- [8] Catálogos de dados referentes aos componentes eletrônicos utilizados, obtidos na página, <http://focus.ti.com>, da Texas Instrument.
- [9] LAFORE, R., *The Waite Group's – C Programming Using Turbo C++*, Ed. SAMS Publishing, 2ª Edição, Indianápolis, 1993.
- [10] BUCHANAN, W., *Applied PC Interfacing, graphics and Interrupts*, Ed. Addison-Wesley, Harlon, Inglaterra, 1996.
- [11] NASCIMENTO JR., C e YONEYAMA, T., *Inteligência Artificial em Controle e Automação*, Ed. Edgard Blücher, São Paulo, 2000 (ver http://www.ele.ita.br/ia_contaut/).
- [12] GABRIEL, GABRIELA W., *Desenvolvimento de Software para o Acionamento de uma Plataforma Móvel Autônoma com Microcontrolador*, 1º e 2º Relatório de Bolsa de IC, FAPESP Proc. 02/01583-5, Nov. 2002.
- [13] MIZRAHI, V.V., *Treinamento em Linguagem C – Módulo Profissional*, MAKRON Books do Brasil Editora Ltda., São Paulo, 1993.

- [14] MCCOMB, G., *The Robot Builder's Bonanza – 99 Inexpensive Robotics Projects*, Ed. TAB Books, USA, 1987.
- [15] *Oscilloscope HM 407, Manual, HAMEG Instruments*, 1998.
- [16] Tabela do Código ASCII retirado da página <http://www.asciitable.com/>.
- [17] *Embedded Processor and Microcontroller primer and FAQ*, retirado da página <http://www.faqs.org/faqs/microcontroller-faq/primer/>, última modificação em 1997.
- [18] *MCS BASIC-52 REFERENCE MANUAL*, Intel Corporation, Santa Clara, 1986.

10.2. Softwares Utilizados

Eletronic Workbench, versão 5.12

Eedit, editor de textos que acompanha o kit AES-51.

TE, programa que estabelece a comunicação entre o microcomputador e a placa do kit AES-51

ASM51, compilador Cross Assembler.

Matlab, versão 6.1 Release 12 (para o traçado dos gráficos).

Microsoft Excel 2000 – Microsoft Office.

Turbo C, versão 3.0.

SP107E, versão FC 3.22 DG 1.63 da HAMEG.

ANEXO A – Desenho Esquemático do Circuito do kit AES-51.

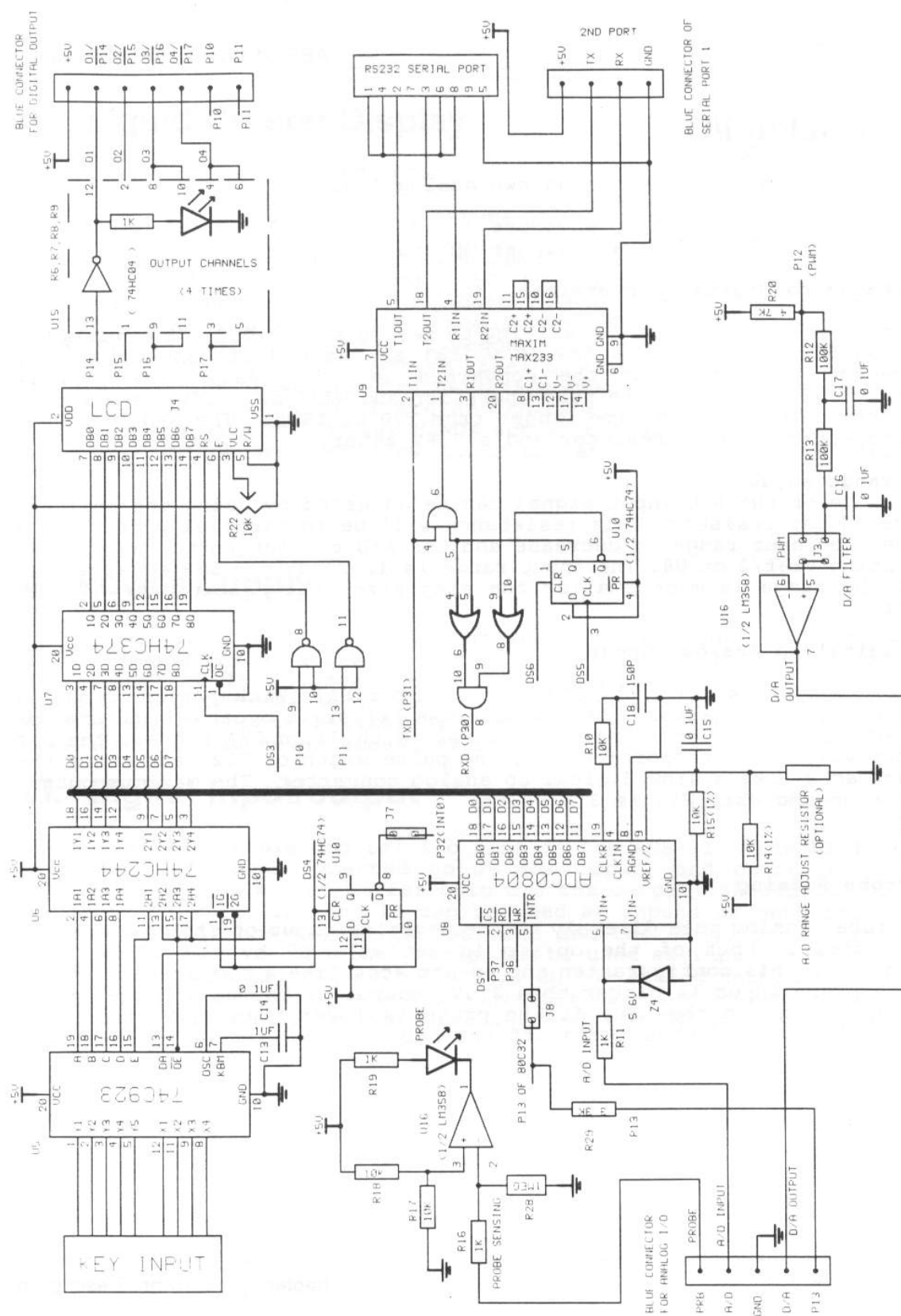


Figura 36 – Circuito esquemático do kit AES-51.

ANEXO B – Circuito Completo do Driver de Acionamento dos Motores de Passo

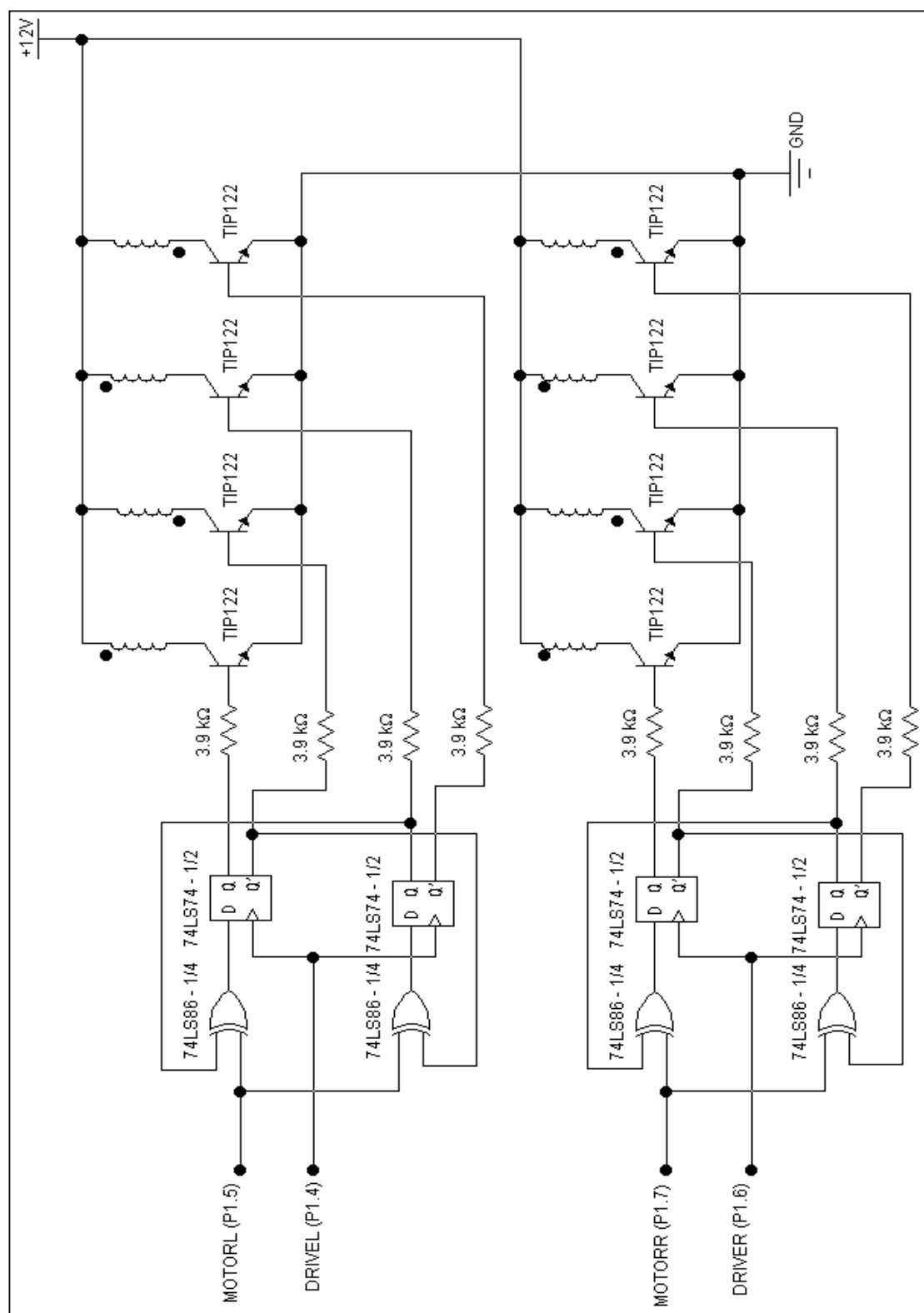


Figura 38 – Diagrama esquemático completo do circuito de driver para acionamento dos motores de passo. Os sinais de entrada são obtidos do 80C32 através do jumper J6 (figura 37 – Anexo A).

ANEXO C – Código Referente às Alterações Realizadas no Programa ROMEO.CPP

ROMEO.CPP versão 2.0

...

```
#define ROT_45 3          /*numero de pulsos para rotacao de 45 graus*/
#define ROT_90 6          /*numero de pulsos para rotacao de 90 graus*/
#define ROT_135 9        /*numero de pulsos para rotacao de 135 graus*/
#define TR_SIDE 4         /*numero de pulsos para translacao em um lado*/
#define TR_DIAG 6         /*numero de pulsos para translacao em uma
diagonal*/
```

...

```
/*-----*/
void gerar_mov_trajetoria()
{
    int ind,CodEscrita=1,tipor,tipot,subtipor,subtipot;
    double xp,yp,xc,yc,teta;
    FILE *foutptr;

    /* Msg avisando que vai comear a enviar os comandos para a plataforma*/
    preparar_topo_tg(FONT_COLOR);
    outtext("Incio da rotina de envio de comandos para a plataforma");
    inicializar_hard_plat();

    /* Usando apenas os vetores rotac e desloc refazer a trajetria */
    setlinestyle(SOLID_LINE,0,THICK_WIDTH); /* usar NORM_WIDTH ou
THICK_WIDTH*/
    xp=dx+js*espacx;
    yp=dy+is*espay;
    teta=teta0;
    desenhar_set(xp,yp,teta0,cor[2]);
    testar_tecla_ESC(); /* PARADA */

    /* Criar arquivo de saida para gravar a tabela de movimentos */
    /* Se ja' existe o arquivo de saida, entao a tabela de movimentos */
    /* nao sera' criada e uma msg de erro e' emitida para o usuario */
    if ((foutptr = fopen(FileOutName, "r")) != NULL)
    {
        CodEscrita=0;
        beep(); /* BEEP */
        preparar_topo_tg(FONT_COLOR_WARNING);
        outtext("Arquivo de saida ja' existe. Nova tabela de movimentos nao
sera' criada.");
        fclose(foutptr);
        getch();
    }
    else if ((foutptr = fopen(FileOutName, "wt")) == NULL)
    /* Erro se arquivo de saida nao existe e nao foi possivel abri-lo. */
    /* Este teste nao aponta erro se o arquivo de saida ja' existe, */
    /* mesmo se o disco esta' cheio ou protegido contra escrita. */
    {
        CodEscrita=0;
        beep(); /* BEEP */
    }
}
```

```

        preparar_topo_tg(FONT_COLOR_WARNING);
        outtext("Nao foi possível criar arquivo de saida com a tabela de
movimentos.");
        getch();
        preparar_topo_tg(FONT_COLOR_WARNING);
        outtext("Possíveis causas: disco cheio ou protegido contra
escrita.");
        getch();
    }

/*Tratamento das variaveis que determinam o numero de pulsos a serem
gravados no arquivo de saida. Este tratamento e necessario devido a
enviarmos byte a byte cada numero, e devido a termos dividido o numero
em parte alta e parte baixa, pela facilidade de leitura pelo programa
em assembly MOVEA.ASM*/

```

```

        if (ROT_45<10) fprintf (foutptr,"00%d\n",ROT_45);
        else if (ROT_45<100) fprintf (foutptr,"0%d\n",ROT_45);
        else fprintf (foutptr,"%d\n",ROT_45);
        if (ROT_90<10) fprintf (foutptr,"00%d\n",ROT_90);
        else if (ROT_90<100) fprintf (foutptr,"0%d\n",ROT_90);
        else fprintf (foutptr,"%d\n",ROT_90);
        if (ROT_135<10) fprintf (foutptr,"00%d\n",ROT_135);
        else if (ROT_135<100) fprintf (foutptr,"0%d\n",ROT_135);
        else fprintf (foutptr,"%d\n",ROT_135);
        if (TR_SIDE<10) fprintf (foutptr,"00%d\n",TR_SIDE);
        else if (TR_SIDE<100) fprintf (foutptr,"0%d\n",TR_SIDE);
        else fprintf (foutptr,"%d\n",TR_SIDE);
        if (TR_DIAG<10) fprintf (foutptr,"00%d\n",TR_DIAG);
        else if (TR_DIAG<100) fprintf (foutptr,"0%d\n",TR_DIAG);
        else fprintf (foutptr,"%d\n",TR_DIAG);

        /* Loop para executar cada a sequênci a de movimentos */
        for (ind=0;ind<npt-1;ind++)
        {
                                /* Mostrar msg para usu rio */
            preparar_topo_tg(FONT_COLOR); /* texto na cor branca */
            sprintf(buffer,"Mov: %d --> teta = %.2f, desl =
%.3f",ind+1,rotac[ind],translac[ind]);
            outtext(buffer);

            /* escrever no arquivo de saida se ele estiver aberto para escrita */
            if (CodEscrita == 1)
            /* Os códigos a serem escritos no arquivo de saída, de acordo com a
            estrutura a seguir, são aqueles apresenados na tabela abaixo:

```

Estrutura do arquivo de saída:

- 1° número - Byte alto do número de pulsos necessários para realizar uma rotação de 45°
- 2° número - Byte baixo do número de pulsos necessários para realizar uma rotação de 45° - centena
- 3° número - Byte baixo do número de pulsos necessários para realizar uma rotação de 45° - dezena
- 4° número - Byte baixo do número de pulsos necessários para realizar uma rotação de 45° - unidade
- 5° número - Byte alto do número de pulsos necessários para realizar uma rotação de 90°
- 6° número - Byte baixo do número de pulsos necessários para realizar uma rotação de 90° - centena
- 7° número - Byte baixo do número de pulsos necessários para realizar

uma rotação de 90° - dezena
 8° número - Byte baixo do número de pulsos necessários para realizar uma rotação de 90° - unidade
 9° número - Byte alto do número de pulsos necessários para realizar uma rotação de 135°
 10° número - Byte baixo do número de pulsos necessários para realizar uma rotação de 135° - centena
 11° número - Byte baixo do número de pulsos necessários para realizar uma rotação de 135° - dezena
 12° número - Byte baixo do número de pulsos necessários para realizar uma rotação de 135° - unidade
 13° número - Byte alto do número de pulsos necessários para realizar uma translação entre dois pontos do labirinto alinhados horizontal ou verticalmente (translação-lado).
 14° número - Byte baixo do número de pulsos necessários para realizar uma translação entre dois pontos do labirinto alinhados horizontal ou verticalmente (translação-lado). - centena
 15o número - Byte baixo do número de pulsos necessários para realizar uma translação entre dois pontos do labirinto alinhados horizontal ou verticalmente (translação-lado). - dezena
 16° número - Byte baixo do número de pulsos necessários para realizar uma translação entre dois pontos do labirinto alinhados horizontal ou verticalmente (translação-lado). - unidade
 17° número - Byte alto do número de pulsos necessários para realizar uma translação entre dois pontos do labirinto alinhados na diagonal (translação-diagonal).
 18° número - Byte baixo do número de pulsos necessários para realizar uma translação entre dois pontos do labirinto alinhados na diagonal (translação-diagonal). - centena
 19° número - Byte baixo do número de pulsos necessários para realizar uma translação entre dois pontos do labirinto alinhados na diagonal (translação-diagonal). - dezena
 20° número - Byte baixo do número de pulsos necessários para realizar uma translação entre dois pontos do labirinto alinhados na diagonal (translação-diagonal). - unidade
 21° número - Tipo do primeiro movimento
 22° número - Subtipo do primeiro movimento
 23° número - Tipo do segundo movimento
 24° número - Subtipo do segundo movimento
 ...
 Último número da tabela - 9 (numero utilizado para finalizar a tabela).

Tabela da relação do movinto a ser realizado e do código a ser transmitido a plataforma:

Tabela 1: Tabela de TIPOS e SUB_TIPOS:

* TIPO *	MOVIMENTO		* SUBTIPO *	MOVIMENTO	*

* 1 *	Frente		* 1 *	Translação-lado	*
* *			* 2 *	Translação-diagonal	*
* 2 *	Trás		* 1 *	Translação-lado	*
* *			* 2 *	Translação-diagonal	*
* 3 *	Rotação-horária		* 1 *	45 graus	*
* *			* 2 *	90 graus	*
* *			* 3 *	135 graus	*
* 4 *	Rotação-anti-horária		* 1 *	45 graus	*
* *			* 2 *	90 graus	*
* *			* 3 *	135 graus	*

```

*/

    if (rotac[ind]>=0) tipor=3;
    else tipor=4;
    if (fabs(rotac[ind])==45.0) subtipor=1;
    else if (fabs(rotac[ind])==90.0) subtipor=2;
    else subtipor=3;
    if (translac[ind]>=0) tipot=1;
    else tipot=2;
    if (fabs(translac[ind])==1.0) subtipot=1;
    else subtipot=2;

fprintf(foutptr,"%d ; %d\n%d ; %d\n",tipor,subtipor,tipot,subtipot);

testar_tecla_ESC();
/* Calcular novo fngulo */
teta=teta+rotac[ind];
/* Desenhar seta com o novo fngulo no ponto de partida */
desenhar_seta(xp,yp,teta,cor[2]);
colocar_texto(1,YELLOW); /* escrever "rota+Eo" */
    /* acionar hardware da plataforma para fazer movimento de rota+Eo */
    acionar_hard_plat(rotac[ind],0);
testar_tecla_ESC(); /* PARADA */

/* Calcular a posi+Eo de chegada */
xc=xp+translac[ind]*cos(teta*((double) (PI/180.0)))*espacx;
yc=yp-translac[ind]*sin(teta*((double) (PI/180.0)))*espacx;
/* Mostrar o trajeto entre os pontos de partida e chegada */
setcolor(WHITE); /* cor branca */
line(xp,yp,xc,yc);
/* Desenhar a seta mostrando o fngulo no ponto de chegada */
desenhar_seta(xp,yp,teta,cor[2]);
desenhar_seta(xc,yc,teta,cor[2]);
colocar_texto(2,YELLOW); /* escrever "transla+Eo" */
/* acionar hardware da plataforma para fazer movimento de rota+Eo */
acionar_hard_plat(0,translac[ind]);
testar_tecla_ESC();

/* Preparar para próximo ponto da trajetória */
xp=xc; /* Fazer ponto de partida = ponto de chegada */
yp=yc;
colocar_texto(1,BLACK); /* apagar texto "rota+Eo" */
colocar_texto(2,BLACK); /* apagar texto "transla+Eo" */
}

/* fechar arquivo aberto para escrita da tabela de movimentos*/
/*Caractere que indica o fim da tabela de movimentos*/
fprintf(foutptr,"\nA");
if (CodEscrita == 1) fclose(foutptr);

preparar_topo_tg(FONT_COLOR);
outtext("FIM ... Pressione ESC para finalizar ...");
setcolor(WHITE);
settextjustify(RIGHT_TEXT, TOP_TEXT);
moveto(mx,dy+(nl-1)*espacy);
outtext("ESC");
while (getch() != ESC);

} /* FIM DA ROTINA */
/*-----
*/

```

ANEXO D – Código do Programa MOVEA.ASM

Programa MOVEA.ASM versão 2.0TG

```
;*****
;*****
;* Instituto Tecnológico de Aeronautica *
;* Divisao de Engenharia Eletronica *
;* Aluna: Gabriela Werner Gabriel *
;* Programa: MOVEA.ASM - Versao: 2.0TG *
;* Data: 25/Outubro/2003 *
;*****

;*****
;* DESCRICAO DO PROGRAMA: ESTE PROGRAMA RECEBE OS DADOS ENVIADOS, *
;* GRAVA-OS NA POSICAO DE MEMORIA RESERVADA PARA TABELA DE MOVI- *
;* MENTOS, INICIADA EM 4500H, E REALIZA O MOVIMENTO DA PLATAFORMA *
;* PROPRIAMENTE DITO. *
;* NO DECORRER DO PROGRAMA SAO REALIZADOS VARIOS PUSH E POP PARA *
;* PILHA, DEVIDO AS ROTINAS DE BASIC-52 QUE ESTAO INSERIDAS NO *
;* PROGRAMA. *
;*****

;*****
;* ALOCACAO DA MEMORIA: *
;* 4500H - 51FFH : MEMORIA PARA A TABELA DE MOVIMENTOS *
;* 5200H - 54FFH : MEMORIA DESTINADA PARA OS TEXTOS DO PROGRAMA *
;* 5500H - 7FFFH : MEMORIA PARA O CODIGO *
;*****

;*****
;* DEFINICAO DAS VARIAVEIS GERAIS DO PROGRAMA *
;*****

        ACC    DATA    0E0H
        DPH    DATA    083H
        DPL    DATA    082H
        SP     DATA    081H

;*****
;* ALOCACAO DA PILHA *
;* A PILHA TEM INICIO NA RAM PARA FINS GERAIS DO MICROCONTROLADOR *
;*****

        MOV     SP,30H

;*****
;* DECLARACAO DAS VARIAVEIS UTILIZADAS NA RECEPCAO DE DADOS *
;* TH1 - BYTE ALTO, TIMER 1 *
;* TMOD - REGISTRADOR PARRA MODO DE OPERACAO DO TIMER *
;* SBUF - BUFFER DA PORTA SERIAL *
;* SCON - REGISTRADOR PARA CONTROLE DA PORTA SERIAL *
;* IE - HABILITA INTERRUPCAO *
;* PSW - PALAVRA QUE DEFINE O STATUS DE PROGRAMA *
;* RCLK - SELECIONA O MODO DE RECEPCAO DE DADOS *
;* TR1 - BIT ON/OFF DO TIMER 1 *
```

```

;*      RI - BIT DE INTERRUPTAO PARA RECEPCAO PELA PORTA SERIAL      *
;*      IP - BIT QUE DEFINE A PRIORIDADE DAS INTERRUPTOES          *
;*****
                TH1      DATA      08DH
                TMOD      DATA      089H
                SBUF      DATA      099H
                SCON      DATA      098H
                IE        DATA      0A8H
                PSW       DATA      0D0H
                RCLK      BIT        0CDH
                TR1       BIT        08EH
                RI        BIT        098H
                IP        DATA      0B8H

;*****
;*      DECLARACAO DAS VARIAVEIS PARA MOVIMENTO DA PLATAFORMA      *
;*      MOTORL, MOTORR - BITS DE DIRECAO PARA O MOTOR ESQUERDO E DIREI- *
;*      TO, RESPECTIVAMENTE                                          *
;*      DRIVEL, DRIVER - BITS DE PULSO PARA OS MOTORES ESQUERDO E DI- *
;*      REITO, RESPECTIVAMENTE                                       *
;*      NPULSE - NUMERO DE PULSOS RELATIVO AO MOVIMENTO A SER EXECUTADO *
;*      TIPO - TIPO DO MOVIMENTO                                     *
;*      S_TIPO - SUB-TIPO DO MOVIMENTO                               *
;*      ROT_45 - GUARDA O NUMERO DE PULSOS A SER REALIZADO NUMA ROTACAO *
;*      DE 45 GRAUS                                                  *
;*      ROT_90 - GUARDA O NUMERO DE PULSOS A SER REALIZADO NUMA ROTACAO *
;*      DE 90 GRAUS                                                  *
;*      ROT_135 - GUARDA O NUMERO DE PULSOS A SER REALIZADO NUMA ROTA- *
;*      CAO DE 135 GRAUS                                             *
;*      TR_SIDE - GUARDA O NUMERO DE PULSOS A SER REALIZADO NUMA TRANS- *
;*      LACAO NUM LADO DO LABIRINTO                                  *
;*      TR_DIAG - GUARDA O NUMERO DE PULSOS A SER REALIZADO NUMA TRANS- *
;*      LACAO NUMA DIAGONAL DO LABIRINTO                             *
;*****

                MOTORL    BIT        094H
                DRIVEL     BIT        095H
                MOTORR     BIT        096H
                DRIVER     BIT        097H
                NPULSE     EQU        R7
                ROT_45     DATA      08H
                ROT_90     DATA      0AH
                ROT_135    DATA      0CH
                TR_SIDE    DATA      0EH
                TR_DIAG    DATA      10H
                TIPO       DATA      11H
                S_TIPO     DATA      12H

;*****
;*      CONFIGURACAO DO ENDEREÇO DE ATENDIMENTO DA INTERRUPTAO DE DE- *
;*      TECCAO DE COLISAO                                           *
;*****

                ORG        4013H      ;INTERRUPTAO EXTERNA 1 INT1 DO
                                      ;BASIC-52
                LJMP      COLISAO

;*****
;*      CONFIGURACAO DO ENDEREÇO DA INTERRUPTAO DE RECEPCAO DE DADOS *
;*      UTILIZANDO ROTINA BASIC-52 DISPONIVEL NO KIT AES-51        *

```



```

;*****
;
;          ORG      4023H          ;VETOR DE ATENDIMENTO DA INTER-
;                                ;RUPCAO SERIAL DO BASIC-52
;          LJMP     INTR          ;JUMP PARA INTERRUPCAO
;*****
;*      DEFINICAO DOS TEXTOS APRESENTADOS NO DECORRER DO PROGRAMA      *
;*      TAIS TEXTOS DEVEM SER TERMINADOS PELO PONTO FINAL, INDICATIVO  *
;*      DO FIM DA STRING                                                *
;*****

TEXT01:      ORG      5200H
TEXT01:      DB      'FIM TRANSMISSAO - ENTER P/ CONT.'
TEXT02:      DB      'ROTACAO ANTI-HORARIA.'
TEXT03:      DB      'ROTACAO HORARIA.'
TEXT04:      DB      'TRANSLACAO TRAS.'
TEXT05:      DB      'TRANSLACAO FRENTE.'
TEXT06:      DB      'ERRO: TIPO INVALIDO.'
TEXT07:      DB      'ERRO: SUB-TIPO INVALIDO.'
TEXT08:      DB      'FIM LABIRINTO - RESET P/ TERMIN.'
TEXT09:      DB      'AGUARDANDO DADOS.'
TEXT010:     DB      'PROGRAMA INTERROMPIDO - COLISAO.'

;*****
;*      PARTE 1 - RECEPCAO DOS DADOS VIA PORTA SERIAL                  *
;*****

PARTE_1:     ORG      5500H
PARTE_1:     PUSH    DPL
PARTE_1:     PUSH    DPH
PARTE_1:     CALL    4100H
PARTE_1:     MOV     DPTR, #TEXT09
PARTE_1:     CALL    LCD          ;ESCREVE O TEXTO 9 NO LCD
PARTE_1:     POP     DPH
PARTE_1:     POP     DPL
PARTE_1:     MOV     TMOD, #20H    ;MODO AUTO-RELOAD
PARTE_1:     MOV     SCON, #50H    ;PALAVRA 8-BIT BAUD RATE
PARTE_1:     MOV     TH1, #0FDH    ;BAUD RATE 9600
PARTE_1:     CLR     RCLK         ;SELECIONA TIMER 1
PARTE_1:     SETB    TR1          ;INICIA TIMER 1
PARTE_1:     ORL     IE, #10010000B ;HABILITA INT SERIAL
PARTE_1:     MOV     DPTR, #4500H  ;CARREGA O INICIO DO ENDE-
;                                ;RECO DA MEMORIA NO QUAL
;                                ;DEVERA SER GRAVADA A TA-
;                                ;BELA DE MOVIMENTOS

ESPERA:      JMP     ESPERA

;*****
;*      PARTE 2 - TRATAMENTO DOS DADOS RECEBIDOS                      *
;*      ESTA PARTE DO PROGRAMA TRATA OS DADOS DE QUANTIDADE DE PULSOS  *
;*      RECEBIDOS. NO CASO, SAO RECEBIDOS BYTE POR BYTE, PORTANTO, PARA *
;*      COMPOR NUMEROS GRANDES E NECESSARIO COMPO-LOS NA RECEPCAO. EN- *
;*      TAO, PARA CADA VARIAVEL E RECEBIDO PRIMEIRO A CENTENA, DEPOIS A *
;*      DEZENA E POR UTIMO A UNIDADE                                    *
;*****

PARTE_2:     MOV     DPTR, #4500H  ;ENDERECO INICIO TABELA
;DADO ROT_45, SEQUENCIA DE TRATAMENTO DE DADOS
PARTE_2:     MOVX    A, @DPTR      ;TOMA O 1o DADO DA TABELA
PARTE_2:     CLR     C            ;LIMPA CARRY

```

```

SUBB A, #30H           ;TRNSFORMA ASCII EM ALG
CALL MUL100            ;TRANSFORMA EM CENTENA
MOV ROT_45, A          ;GUARDA NUMERO
INC DPTR
MOVX A, @DPTR          ;TOMA O 2o DADO DA TABELA
CLR C                  ;LIMPA CARRY
SUBB A, #30H           ;TRANSOMA ASCII EM ALG
CALL MUL10             ;TRANSFOEMA EM DEZENA
ADD A, ROT_45          ;SOMA NUMERO COM ANTERIOR
MOV ROT_45, A          ;GUARDA NUMERO
INC DPTR
MOVX A, @DPTR          ;TOMA O 3o DADO DA TABELA
CLR C                  ;LIMPA CARRY
SUBB A, #30H           ;TRANSFORMA ASCII EM ALG
ADD A, ROT_45          ;SOMA COM ANTERIORES
MOV ROT_45, A          ;GUARDA RESULTADO
;DADO ROT_90, MESMA SEQUENCIA DE TRATAMENTO FEITA PARA ROT_45
INC DPTR
MOVX A, @DPTR
CLR C
SUBB A, #30H
CALL MUL100
MOV ROT_90, A
INC DPTR
MOVX A, @DPTR
CLR C
SUBB A, #30H
CALL MUL10
ADD A, ROT_90
MOV ROT_90, A
INC DPTR
MOVX A, @DPTR
CLR C
SUBB A, #30H
ADD A, ROT_90
MOV ROT_90, A
;DADO ROT_135, MESMA SEQUENCIA DE TRATAMENTO FEITA PARA ROT_45
INC DPTR
MOVX A, @DPTR
CLR C
SUBB A, #30H
CALL MUL100
MOV ROT_135, A
INC DPTR
MOVX A, @DPTR
CLR C
SUBB A, #30H
CALL MUL10
ADD A, ROT_135
MOV ROT_135, A
INC DPTR
MOVX A, @DPTR
CLR C
SUBB A, #30H
ADD A, ROT_135
MOV ROT_135, A
;DADO TR_SIDE, MESMA SEUENCIA DE TRATAMENTO EITA PARA ROT_45
INC DPTR
MOVX A, @DPTR
SUBB A, #30H
CALL MUL100

```

```

MOV    TR_SIDE, A
INC    DPTR
MOVX   A, @DPTR
CLR    C
SUBB   A, #30H
CALL   MUL10
ADD    A, TR_SIDE
MOV    TR_SIDE, A
INC    DPTR
MOVX   A, @DPTR
CLR    C
SUBB   A, #30H
ADD    A, TR_SIDE
MOV    TR_SIDE, A
;DADO TR_DIAG, MESMA SEUENCIA DE TRATAMENTO FEITA PARA ROT_45
INC    DPTR
MOVX   A, @DPTR
CLR    C
SUBB   A, #30H
CALL   MUL100
MOV    TR_DIAG, A
INC    DPTR
MOVX   A, @DPTR
CLR    C
SUBB   A, #30H
CALL   MUL10
ADD    A, TR_DIAG
MOV    TR_DIAG, A
INC    DPTR
MOVX   A, @DPTR
CLR    C
SUBB   A, #30H
ADD    A, TR_DIAG
MOV    TR_DIAG, A

;ESPERA TECLA DO KIT PARA DAR INICIO AO MOVIMENTO DOS MOTORES
CALL   4108H                ;ROTINA BASIC-52

;*****
;*      PARTE 3 - MOVIMENTO DOS MOTORES DE PASSO      *
;*****

PARTE_3:  ORL    IE, #10000100B    ;HABILIDADE A INTERRUPCAO
                                                ;INT1 EXTERNA
ORL       IP, #00000100B          ;DEFINE INT1 COMO DE MA-
                                                ;XIMA PRIORIDADE. A INT1
                                                ;E UTILIZADA PARA DETEC-
                                                ;CAO DE COLISAO
MOV       DPTR, #450FH            ;POSICAO DE MEMORIA NO
                                                ;QUAL ENCONTRA-SE O 1o
                                                ;TIPO DA TABELA DE MOVIM
BEGIN:    MOVX   A, @DPTR          ;TOMA O TIPO
CLR       C                      ;LIMPA CARRY
SUBB      A, #30H                ;TRANSFORMA ASCII EM ALG
MOV       TIPO, A                ;CARREGA TIPO DO MOVIMENTO
INC       DPTR
MOVX      A, @DPTR              ;TOMA O SUBTIPO DA TA-
                                                ;BELA DE MOVIMENTO
CLR       C                      ;LIMPA CARRY
SUBB      A, #30H                ;TRANSFORMA ASCII EM ALG
MOV       S_TIPO, A              ;CARREGA SUB-TIPO DO MOV

```

```

;TESTA SE O MOVIMENTO E O DE TRANSLACAO PARA FRENTE
FORW:    MOV    A, TIPO                ;CARREGA O TIPO
        CJNE   A, #1, BACK            ;TIPO NAO PARA FRENTE,
        ;SALTA PARA O TESTE TRAS
        MOV    A, S_TIPO              ;CARREGA O SUBTIPO
        CJNE   A, #1, HOR1            ;TESTA SUB-TIPO DE TRANS-
        ;LACAO NO LADO DO LABIRINT
        MOV    A, TR_SIDE              ;SE SUBTIPO E 1 --> MO-
        ;VIMENTO NUM LADO DO LABIR
        MOV    NPULSE, A              ;CARREGA O NUMERO DE PULSOS
        ;PARA UMA TRANSLACAO NUM
        ;LADO
        JMP    FOR2                  ;IMPLEMENTA TRANSLACAO PARA
        ;FRENTE
HOR1:    MOV    A, S_TIPO              ;CARREGA O SUBTIPO
        CJNE   A, #2, ERRO1          ;TESTA SUB-TIPO DE TRANS-
        ;LACAO NA DIAGONAL DO
        ;LABIRINTO
        MOV    A, TR_DIAG              ;SE SUBTIPO E 2 --> MO-
        ;VIMENTO NUMA DIAGONAL DO
        ;LABIRINTO
        MOV    NPULSE, A              ;CARREGA O NUMERO DE PULSOS
        ;PARA UMA TRANSLACAO NUMA
        ;DIAGONAL
        JMP    FOR2                  ;IMPLEMENTA TRANSLACAO PARA
        ;FRENTE
ERRO1:   PUSH   DPL
        PUSH   DPH
        CALL   4100H                  ;LIMPA LCD - BASIC-52
        MOV    DPTR, #TEXT06
        CALL   LCD                    ;ESCREVE TEXTO LCD
        CALL   4108H                  ;ESPERA QUE UMA TECLA DO KIT
        ;SEJA PRESSIONADA
        POP    DPH
        POP    DPL
        RET
FOR2:    MOV    A, NPULSE
        PUSH   ACC
        PUSH   DPL
        PUSH   DPH
        CALL   4100H                  ;LIMPA LCD - BASIC-52
        MOV    DPTR, #TEXT05
        CALL   LCD                    ;ESCREVE TEXTO LCD
        POP    DPH
        POP    DPL
        POP    ACC
        MOV    NPULSE, A
        CALL   FORWARD                ;TRANSLACAO PARA FRENTE
        JMP    O_LINE                 ;LE PROXIMO TIPO
;TESTA SE O MOVIMENTO E O DE TRANSLACAO PARA TRAS. A SEQUENCIA DE CODIGO
;PARA OS DEMAIS MOVIMENTOS: TRAS, ROTACAO HORARIA E ROTACAO ANTI-HORARIA
;E SEMELHANTE A DE TRANSLACAO PARA FRENTE.
BACK:    MOV    A, TIPO
        CJNE   A, #2, RIGHT          ;TIPO NAO PARA TRAS, SALTA
        ;PARA ROTACAO SENT HORARIO
        MOV    A, S_TIPO
        CJNE   A, #1, HOR2            ;TESTA SUB-TIPO DE TRANS-
        ;LACAO NO LADO DO LABIRINT
        MOV    A, TR_SIDE
        MOV    NPULSE, A
        JMP    BAC2

```

```

HOR2:      MOV    A, S_TIPO
           CJNE   A, #2, ERRO2      ;TESTA SUB-TIPO DE TRANS-
                                     ;LACAO NA DIAGONAL NO LA-
                                     ;BIRINTO

           MOV    A, TR_DIAG
           MOV    NPULSE, A
           JMP    BAC2

ERRO2:     PUSH   DPL
           PUSH   DPH
           CALL   4100H              ;LIMPA LCD - BASIC-52
           MOV    DPTR, #TEXT06
           CALL   LCD                ;ESCREVE TEXTO LCD
           CALL   4108H
           POP    DPH
           POP    DPL
           RET

BAC2:      MOV    A, NPULSE
           PUSH   ACC
           PUSH   DPL
           PUSH   DPH
           CALL   4100H              ;LIMPA LCD - BASIC-52
           MOV    DPTR, #TEXT04
           CALL   LCD                ;ESCREVE TEXTO LCD
           POP    DPH
           POP    DPL
           POP    ACC
           MOV    NPULSE, A
           CALL   BACKWARD           ;TRANSLACAO PARA TRAS
           JMP    O_LINE             ;LE PROXIMO TIPO

;TESTA SE O MOVIMENTO E O DE ROTACAO HORARIA
RIGHT:     MOV    A, TIPO
           CJNE   A, #3, LEFT        ;TIPO NAO ROTACAO HORARIA,
                                     ;SALTA PARA ROT ANTI-HORAR

           MOV    A, S_TIPO
           CJNE   A, #1, HOR3        ;TESTA SUB-TIPO DE ROTACAO
                                     ;DE 45 GRAUS EM TORNO DO
                                     ;EIXO DA PLATAFORMA

           MOV    A, ROT_45
           MOV    NPULSE, A
           JMP    RIG2

HOR3:      MOV    A, S_TIPO
           CJNE   A, #2, HOR4        ;TESTA SUB-TIPO DE ROTACAO
                                     ;DE 90 GRAUS EM TORNO DO
                                     ;EIXO DA PLATAFORMA

           MOV    A, ROT_90
           MOV    NPULSE, A
           JMP    RIG2

HOR4:      MOV    A, S_TIPO
           CJNE   A, #3, ERRO3        ;TESTA SUB-TIPO DE ROTACAO
                                     ;DE 135 GRAUS EM TORNO DO
                                     ;EIXO DA PLATAFORMA

           MOV    A, ROT_135
           MOV    NPULSE, A
           JMP    RIG2

ERRO3:     PUSH   DPL
           PUSH   DPH
           CALL   4100H              ;LIMPA LCD - BASIC-52
           MOV    DPTR, #TEXT06
           CALL   LCD                ;ESCREVE TEXTO LCD
           CALL   4108H
           POP    DPH

```

```

      POP    DPL
      RET
RIG2:  MOV    A, NPULSE
      PUSH  ACC
      PUSH  DPL
      PUSH  DPH
      CALL  4100H          ;LIMPA LCD - BASIC-52
      MOV   DPTR, #TEXT03
      CALL  LCD            ;ESCREVE TEXTO LCD
      POP   DPH
      POP   DPL
      POP   ACC
      MOV   NPULSE, A
      CALL  CWISE          ;ROTACAO HORARIA
      JMP   O_LINE        ;LE PROXIMO TIPO
;TESTA SE O MOVIMENTO E O DE ROTACAO ANTI-HORARIA
LEFT:  MOV    A, TIPO
      CJNE  A, #4, ENDED   ;TIPO NA ROT ANTI-HORARIA,
                          ;SALTA PARA FIM DO MOVIMEN

      MOV   A, S_TIPO
      CJNE  A, #1, HOR5    ;TESTA SUB-TIPO DE ROTACAO
                          ;DE 45 GRAUS EM TORNO DO
                          ;EIXO DA PLATAFORMA

      MOV   A, ROT_45
      MOV   NPULSE, A
      JMP   LEF2
HOR5:  MOV    A, S_TIPO
      CJNE  A, #2, HOR6    ;TESTA SUB-TIPO DE ROTACAO
                          ;DE 90 GRAUS EM TORNO DO
                          ;EIXO DA PLATAFORMA

      MOV   A, ROT_90
      MOV   NPULSE, A
      JMP   LEF2
HOR6:  MOV    A, S_TIPO
      CJNE  A, #3, ERRO4   ;TESTA SUB-TIPO DE ROTACAO
                          ;DE 135 GRAUS EM TORNO DO
                          ;EIXO DA PLATAFORMA

      MOV   A, ROT_135
      MOV   NPULSE, A
      JMP   LEF2
ERRO4:  PUSH  DPL
      PUSH  DPH
      CALL  4100H          ;LIMPA LCD - BASIC-52
      MOV   DPTR, #TEXT07
      CALL  LCD            ;ESCREVE TEXTO LCD
      CALL  4108H
      POP   DPH
      POP   DPL
      RET
LEF2:  MOV    A, NPULSE
      PUSH  ACC
      PUSH  DPL
      PUSH  DPH
      CALL  4100H          ;LIMPA LCD - BASIC-52
      MOV   DPTR, #TEXT02
      CALL  LCD            ;ESCREVE TEXTO LCD
      POP   DPH
      POP   DPL
      POP   ACC
      MOV   NPULSE, A
      CALL  CCWISE        ;ROTACAO ANTI-HORARIA

```

```

        JMP     O_LINE                ;LE PROXIMO TIPO
;TESTA O FIM DA TABELA DE MOVIMENTOS
ENDED:   MOV     A, TIPO
        CJNE    A, #11H, ERRO5       ;TESTA FIM DA TABELA DE
        LJMP    TERM                 ;MOVIMENTOS, QUE DEVE SER
                                      ;TERMINADA POR 'A', 41H-30H

        RET

ERRO5:   PUSH    DPL
        PUSH    DPH
        CALL    4100H                ;LIMPA LCD - BASIC-52
        MOV     DPTR, #TEXT06
        CALL    LCD                  ;ESCREVE TEXTO LCD
        CALL    4108H
        POP     DPH
        POP     DPL
        RET

O_LINE:  INC     DPTR
        LJMP    BEGIN

;*****
;*      ROTINA QUE IMPLEMENTA O MOVIMENTO PARA FRENTE, SEGUNDO A QUAN-   *
;*      TIDADE DE PULSOS DEFINIDA POR NPULSE                           *
;*****

FORWARD: SETB    MOTORL              ;ROTACAO DO MOTOR ES-
                                      ;QUERDO PARA FRENTE
        SETB    MOTORR              ;ROTACAO DO MOTOR DIREI-
                                      ;TO PARA FRENTE
LOOPF:   CALL    PULSE               ;ENVIA NPULSE PULSOS AOS
        DJNZ    NPULSE, LOOPF       ;MOTOR
        SETB    DRIVEL
        SETB    DRIVER
        CALL    AMOVE               ;ESPERA ENTRE MOVIMENTOS
        RET

;*****
;*      ROTINA QUE IMPLEMENTA O MOVINETO PARA TRAS, SEGUNDO A QUANTI-   *
;*      DADE DE PULSOS DEFINIDA POR NPULSE                             *
;*****

BACKWARD: CLR     MOTORL             ;ROTACAO DO MOTOR ES-
                                      ;QUERDO PARA TRAS
        CLR     MOTORR              ;ROTACAO DO MOTOR DIREI-
                                      ;TO PARA TRAS
LOOPB:   CALL    PULSE               ;ENVIA NPULSE PULSOS AOS
        DJNZ    NPULSE, LOOPB       ;MOTOR
        SETB    DRIVER
        SETB    DRIVEL
        CALL    AMOVE               ;ESPERA ENTRE MOVIMENTOS
        RET

;*****
;*      ROTINA QUE IMPLEMENTA O MOVIMENTO DE ROTACAO HORARIA, SEGUNDO A  *
;*      QUANTIDADE DE PULSOS DEFINIDA POR NPULSE                       *
;*****

CWISE:   SETB    MOTORL             ;ROTACAO DO MOTOR ES-
                                      ;QUERDO PARA FRENTE
        CLR     MOTORR              ;ROTACAO DO MOTOR DIREI-
                                      ;TO PARA TRAS

```

```

LOOPCW:    CALL    PULSE                ;ENVIA NPULSE PULSOS AOS
           DJNZ    NPULSE, LOOPCW        ;MOTOR
           SETB    DRIVER
           SETB    DRIVEL
           CALL    AMOVE                ;ESPERA ENTRE MOVIMENTOS
           RET

;*****
;*  ROTINA QUE IMPLEMENTA O MOVIMENTO DE ROTACAO ANTI-HORARIA, SE-  *
;*  GUNDO A QUANTIDADE DE PULSOS DEFINIDA POR NPULSE              *
;*****

CCWISE:    CLR     MOTORL                ;ROTACAO DO MOTOR ES-
           SETB    MOTORR                ;QUERDO PARA TRAS
           SETB    MOTORR                ;ROTACAO DO MOTOR DIREI-
           SETB    MOTORR                ;TO PARA FRENTE

LOOPCCW:    CALL    PULSE                ;ENVIA NPULSE PULSOS AOS
           DJNZ    NPULSE, LOOPCCW        ;MOTOR
           SETB    DRIVER
           SETB    DRIVEL
           CALL    AMOVE                ;ESPERA ENTRE MOVIMENTOS
           RET

;*****
;*  ROTINA QUE IMPLEMENTA O ENVIO DOS PULSOS AOS MOTORES DE PASSO.  *
;*  ENTRE CADA PULSO E IMPLEMENTADA UM ESPERA DE FORMA QUE OS MO-  *
;*  TORES CONSIGAM RESPONDER SATISFATORIAMENTE A ESTES PULSOS. O  *
;*  FORMATO DA ONDA ENVIADA AOS MOTORES E ENTAO QUADRADA          *
;*****

PULSE:     CLR     DRIVEL                ;SETA BIT DE PULSO DO
           CLR     DRIVER                ;MOTOR ESQUERDO
           CLR     DRIVER                ;SETA BIT DE PULSO DO
           CALL    WAIT                  ;MOTOR DIREITO
           SETB    DRIVEL                ;TEMPO
           SETB    DRIVER                ;ZERA BIT DE PULSO DO
           SETB    DRIVER                ;MOTOR ESQUERDO
           SETB    DRIVER                ;ZERA BIT DE PULSO DO
           CALL    WAIT                  ;MOTOR DIREITO
           RET

;*****
;*  ROTINA QUE IMPLEMENTA TEMPO DE APROXIMADAMENTE 1 SEGUNDO      *
;*****

WAIT:      MOV     R2, #100
           MOV     R3, #255
           MOV     R4, #255
           DJNZ    R2, $
           DJNZ    R3, $-1
           DJNZ    R4, $-2
           RET

;*****
;*  ROTINA QUE IMPLEMENTA ESPERA ENTRE MOVIMENTOS. ESTA ESPERA E  *
;*  QUE A IMPLEMENTADA POR WAIT, POIS AQUI PODE OCORRER INVERSAO  *
;*  NO SENTIDO DOS MOTORES                                         *
;*****

AMOVE:     MOV     R2, #255

```



```

        MOV    R3, #255
        MOV    R4, #255
        MOV    R1, #2
        DJNZ   R2, $
        DJNZ   R3, $-1
        DJNZ   R4, $-2
        DJNZ   R1, $-3
        RET

;*****
;*    ROTINA QUE TRANSFORMA ALGARISMO EM DEZENA - LOOP DE MULTIPLICA- *
;*    CAO                                                                *
;*****

MUL10:    MOV    R3, #9
          MOV    R4, A
MULT1:    ADD    A, R4
          DJNZ   R3, MULT1
          RET

;*****
;*    ROTINA QUE TRANSFORMA ALGARISMO EM CENTENA - LOOP DE MULTIPLI- *
;*    CACAO                                                            *
;*****

MUL100:   MOV    R3, #99
          MOV    R4, A
MULT2:    ADD    A, R4
          DJNZ   R3, MULT2
          RET

;*****
;*    ROTINA QUE ESCRIVE STRING NO LCD DO KIT AES-51 - CORRESPONDE A *
;*    UM LOOP QUE ESCRIVE BYTES QUE TERMINA QUANDO UM BYTE ESPECIFICO *
;*    E ENCONTRADO, NO CASO O PONTO FINAL                            *
;*****

LCD:      MOVX   A, @DPTR          ;CARREGA O BYTE DA SE-
                                   ;QUENCIA APONTADO POR
                                   ;DPTR
          MOV    21H, A
          INC    DPTR              ;TOMA O PROXIMO BYTE
          CALL   4104H             ;ROTINA QUE ENVIA DADO
                                   ;PRESENTE NO ENDEREÇO 21H
                                   ;PARA O LCD - BASIC-52
          CJNE   A, #2EH, LCD      ;COMPARA SE O BYTE LIDO
                                   ;CORRESPONDE AO PONTO

          RET

;*****
;*    SUB-ROTINA DE INTERRUPCAO DE DETECCAO DE COLISAO - AQUI E UTI- *
;*    LIZADA A INTERRUPCAO INT1 PARA DETECTAR A COLISAO.            *
;*    INICIALMENTE OS VALORES DE DPTR E DOS REGISTRADORES DE QUANTI- *
;*    DADE DE PULSOS SAO SALVOS NA PILHA, E AO FINAL DA INTERRUPCAO *
;*    RESGATADOS                                                       *
;*****

COLISAO:  MOV    A, NPULSE
          PUSH   ACC
          PUSH   DPH
          PUSH   DPL

```



```

        RETI
;CASO TERMINADA A INTERRUPCAO O PROGRAMA AVISA NO LCD E SALTA PARA ROTI-
;NA DE EXECUCAO DO MOVIMENTO DA TRAJETORIA
FIMR:    CALL  4100H                ;LIMPA LCD - BASIC-52
        MOV   DPTR, #TEXT01
        CALL  LCD                   ;ESCREVE TEXTO 1 NO LCD
        LJMP  PARTE_2
        RET

;*****
;*      FINALIZACAO DO PROGRAMA E DO LABIRINTO      *
;*****

TERM:    PUSH  DPL
        PUSH  DPH
        CALL  4100H                ;LIMPA LCD - BASIC-52
        MOV   DPTR, #TEXT08
        CALL  LCD                   ;ESCREVE TEXTO 8 LCD
        CALL  4108H                ;AGUARDA
        POP   DPH
        POP   DPL
        END                        ;DIRETIVA FIM DE PROGRAMA

```



```

/*Definicoes das subrotinas utilizadas no decorrer do programa      */
/*                                                                    */
/*setup_serial - Subrotina que configura os parametros da porta serial*/
/* para transmissao de dados                                       */
/*send_number - Subrotina que envia um dado para a porta de comunica-*/
/* cao serial                                                       */
/*le_arquivo - Subrotina que le dados do arquivo TABM_NEW.TXT gerado */
/* pelo programa ROMEO e chama a subrotina send_number para enviar os */
/* dados lidos para o microcontrolador                             */
/*.....*/

void setup_serial(void);
void send_number(int ch);
int le_arquivo (void);

/*.....*/
/*Aqui comeca a rotina principal                                   */
/*.....*/

int main(void)
{
    setup_serial(); //Configuracao da porta de saida de dados
    le_arquivo (); //Subrotina que le o arquivo gerado por ROMEO
    return (0);
}

/*.....*/
/*Rotina para configuracao da porta serial. A configuracao utilizada */
/* e 8 bits de dados,1 bit de stop bit, 1 bit de paridade sendo a pa- */
/* ridade.                                                             */
/*.....*/

void setup_serial(void)
{
    outportb(LCR, 0x80); //Seta todos os bits do endereco asso-
                        //ciado ao LCR, no caso o 03FBh
    outportb(TXDATA, 0x0C); //Envia o divisor de frequencia para o
                        //endereço do LCR e seu subsequente,
                        //já que este e um dado de 16 bits.
                        //Nao esquecer que o numero enviado e
                        //em hexadecimal, mesmo nao estando
                        //indicado
    outportb(TXDATA+1,0x00); //Este numero e enviado para a posicao
                        //subsequente do LCR, para completar o
                        //dado anterior que seria de 16 bits
    outportb(LCR, 0x03); //Aqui sao enviadas as demais configu-
                        //racoes para a transmissao, conforme
                        //a descricao abaixo
}
/* O bit carregado no LCR corresponde a 00000011b, que corresponde a (do
bit menos significativo para o mais):
0 - acesso ao buffer TD/RD, 0 - saída normal
0 - sem stick bit, 0 - paridade par
0 - sem bit de paridade, 0 - 1 stop bit
11 - 8 bits de dados */

/*.....*/
/*Rotina de leitura de arquivo. O arquivo lido e o TABM_NEW.TXT que e */
/*gerado pelo programa ROMEO.CPP, responsavel por calcular a melhor */
/*trajetoria entre dois pontos dentro de um labirinto. Este programa */
/*encontra-se em anexo.                                             */

```

```

/*A estrutura do arquivo TABM_NEW.TXT segue o seguinte padrao: */
/*(Em que cada variavel e composta por tres bytes, em que um corres- */
/*ponde a centena, outro a dezena e outro a unidade, nesta ordem) */
/*1o, 2o e 3o bytes - ROT_45: numero de pulsos para ser realizada uma */
/* rotacao de 45 graus. */
/*4o, 5o e 6o bytes - ROT_90: numero de pulsos para ser realizada uma */
/* rotacao de 90 graus. */
/*7o, 8o e 9o bytes - ROT_135: numero de pulsos para ser realizada uma */
/* rotacao de 135 graus. */
/*10o, 11o e 12o bytes - TR_LADO: numero de pulsos para ser realizada */
/* uma translacao em um lado referente as dimensoes do labirinto. */
/*13o, 14o e 15o bytes - TR_DIAG: numero de pulsos para ser realizada */
/* uma translacao em uma diagonal referente as dimensoes do labirinto. */
/*16o byte em diante: tipos e subtipos de movimento, na sequencia do */
/* movimento sendo alternados os tipos e os subtipos, ate o final do */
/* movimento desejado. */
/*A tabela de tipos e subtipos corresponde a tabela apresentada a seguir:

```

```

*****
* TIPO * MOVIMENTO * SUBTIPO * MOVIMENTO *
*****
* 1 * Frente * 1 * Translacao-lado *
* * * * 2 * Translacao-diagonal *
* 2 * Tras * 1 * Translacao-lado *
* * * * 2 * Translacao-diagonal *
* 3 * Rotacao horaria * 1 * 45 graus *
* * * * 2 * 90 graus *
* * * * 3 * 135 graus *
* 4 * Rotacao anti-horaria * 1 * 45 graus *
* * * * 2 * 90 graus *
* * * * 3 * 135 graus *
***** */

```

```

int le_arquivo (void)
{
    /*Definicao das variaveis locais da sub_rotina le_arquivo*/
    FILE *pa;
    int ch, cont;
    float resto;

    /*Abertura do arquivo para leitura da tabela de movimentos*/
    cont=1;
    if((pa=fopen("TABM_NEW.txt","r"))==NULL)
    {
        printf("Nao foi possivel abrir o arquivo.");
        return 1 ;
    }
    ch=getc(pa);          //Le um caractere do arquivo

    /*Envia os dados imprimiveis para a porta serial*/

    while (ch!=EOF)
    {
        if (ch>=48 && ch<=57 || ch==65 || ch==97)
        {
            /*Laco que escreve na tela do microcomputador os
            dados enviados*/
            if (ch==97) ch=65;
            if (1<=cont<=15)
            {

```

```

        if (cont==1)
        {
            printf ("\nRotacao 45 graus = %c",ch);
        }
        if (cont==4)
        {
            printf ("\nRotacao 90 graus = %c",ch);
        }
        if (cont==7)
        {
            printf ("\nRotacao 135 graus = %c",ch);
        }
        if (cont==10)
        {
            printf ("\nTranslacao num lado = %c",ch);
        }
        if (cont==13)
        {
            printf ("\nTranslacao numa diagonal = %c",ch);
        }
        if (cont==2||cont==5||cont==8||cont==11||cont==14)
        {
            printf("%c",ch);
        }
        if (cont==3||cont==6||cont==9||cont==12||cont==15)
        {
            printf("%c pulsos",ch);
        }
    }
    if (cont==15) printf("\n\n Tabela de Movimentos:");
    if (cont>=16 && ch!=65)
    {
        resto=cont%2;
        if (resto==0.0) printf ("\n Tipo: %c ",ch);
        else printf("- Sub-tipo: %c",ch);
    }
    send_number(ch); //Envia o dado pela porta serial
    cont=cont+1;
    }
    ch=getc(pa);
}
printf("\n");
fclose (pa); //Fecha o arquivo TABM_NEW.TXT
return (0);
} //Fim da subrotina le_arquivo

/*.....*/
/*Esta subrotina e responsavel por enviar caracteres para a porta se- */
/*rial do microcomputador com as configuracoes pre-definidas em subro-*/
/*tina anterior */
/*.....*/

void send_number(int ch)
{
    /*Definicao das variaveis locais pertencentes a esta subrotina*/
    char status;

    /*Laco que verifica se uma nova transmissao ja pode ser realizada,
    testando se o buffer da porta serial esta vazio ou nao. Caso nao
    esteja, a verificacao e repetida ate que a condicao desejada seja
    atingida*/

```

```
do
{
    status=inportb(LSR) & 0x40;
}while (status!=0x40);

/*Comando que envia o dado para porta serial*/
outportb(TXDATA, (char) ch);
}//Chave de fim de subrotina
```


ANEXO F – Execução do Movimento da Plataforma Passo a Passo

Como forma de auxiliar o operador, este capítulo descreve detalhadamente todos os passos para realizar o movimento da plataforma num labirinto previamente determinado pelo operador.

Apresentando-se uma visão geral do funcionamento da plataforma, para que a mesma execute uma trajetória previamente determinada ao longo de um labirinto inicialmente fornecido, é necessária a sequência, de ações, apresentada na figura 39.

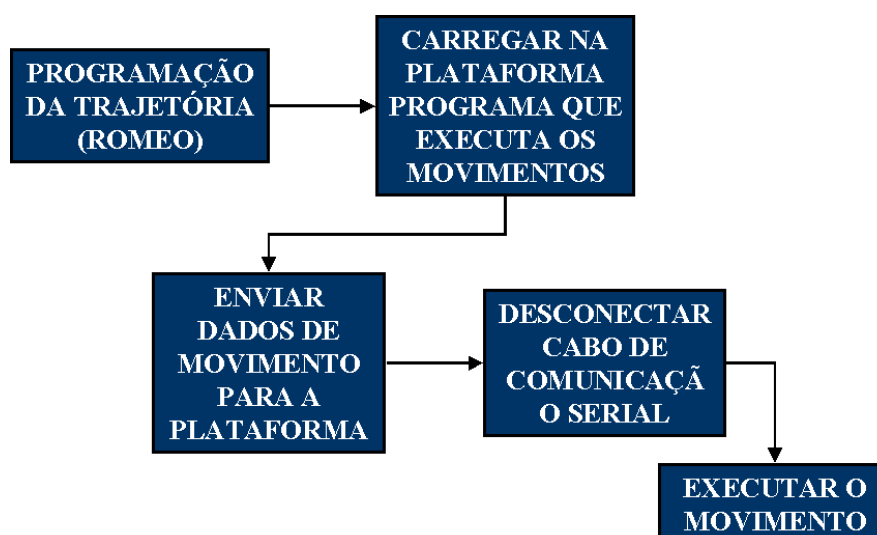


Figura 39 – Esquema representativo da sequência de ações necessárias para que a plataforma execute os movimentos previamente determinados.

Seguindo esta sequência apresentada, os passos detalhados para pô-la em prática são apresentados a seguir:

1. Conectar o cabo serial no PC (na porta COM1) e no conector DB-9 do kit AES-51 presente na plataforma.
2. Ligar o PC.
3. Criar arquivo contendo o labirinto inicial (o nome deste arquivo deve conter no máximo 8 caracteres e sua extensão deverá ser .TXT), utilizando o NOTEPAD. A estrutura para este arquivo é a que segue:

L	C
X	... X
:	:
:	:
X	... X

No qual, L é o número de linhas, C é o número de colunas do labirinto e X deve ser 0 para célula vazia, 1 para célula com obstáculo, 2 para a célula de partida e 3 para a célula de chegada.

4. Verificar se existe um arquivo TABM_NEW.TXT no diretório corrente, caso positivo deletá-lo.
5. Confirmar a existência do arquivo EGAVGA.bgi no diretório corrente.
6. Executar o programa ROMEO.EXE, chamando no prompt do DOS:
`\>ROMEO.↵`
7. Copiar o arquivo TABM_NEW.TXT gerado para o diretório no qual se encontram os arquivos que acompanham o kit AES-51.
8. Verificar se a bateria está propriamente carregada.
9. Verificar se os arquivos MOVEA.HEX e SEND.EXE estão disponíveis. Caso contrário gerar tais arquivos através dos programas ASM51 e TC, respectivamente.
10. Ligar a plataforma através da chave seletora ON-OFF.
11. Executar o programa TE, através do comando, no prompt do DOS:
`\>TE.↵`
12. Resetar o kit AES-51 presente na plataforma, através do botão de reset (localizado logo abaixo do LCD).
13. Pressionar a barra de espaços do teclado do PC para executar a comunicação entre o kit AES-51 e o PC.
14. No prompt do programa TE digitar RX para indicar que o arquivo a ser carregado posteriormente encontra-se no formato .HEX, utilizando o comando, digitado com o teclado do PC:
`#>RX.↵`
15. Pressionar ALT+S, simultaneamente, e digitar o nome do arquivo MOVEA com extensão .HEX, da seguinte:
`MOVEA.HEX.↵`
16. Digitar, ainda no prompt do programa TE:

#>call 5500H.↵

17. Pressionar ALT+X, simultaneamente, e logo em seguida enter, para sair do programa TE.
18. De volta ao DOS, executar o programa SEND.EXE, utilizando a seguinte linha de comando:
↵>SEND.↵
19. Desconectar o cabo serial, tomando-se o cuidado para não encostar na parte metálica do cabo.
20. Executar o movimento da plataforma seguindo as instruções apresentadas no LCD do kit AES-51.
21. Caso ocorra colisão deve-se pressionar a tecla enter duas vezes para que a plataforma prossiga o movimento segundo a trajetória original.
22. Para finalizar o movimento, quando o labirinto tiver terminado, deve-se resetar o kit e de preferência, desconectar a bateria.

FOLHA DE REGISTRO DO DOCUMENTO			
1. CLASSIFICAÇÃO/TIPO TC	2. DATA 11 de novembro de 2003	3. DOCUMENTO N° CTA/ITA-IEE/TC-007/2003	4. N° DE PÁGINAS 104
5. TÍTULO E SUBTÍTULO: Desenvolvimento de Software para o Acionamento de uma plataforma Móvel Autônoma com Microcontrolador			
6. AUTOR(ES): Gabriela Werner Gabriel			
7. INSTITUIÇÃO(ÕES)/ÓRGÃO(S) INTERNO(S)/DIVISÃO(ÕES): Instituto Tecnológico de Aeronáutica. Divisão de Engenharia Eletrônica – ITA/IEE			
8. PALAVRAS-CHAVE SUGERIDAS PELO AUTOR: Robótica Móvel; eletrônica embarcada; microcontrolador; planejamento de trajetória; sensores; eletrônica de potência			
9. PALAVRAS-CHAVE RESULTANTES DE INDEXAÇÃO: Controladores; Sistemas de computadores embarcados; Dinâmica de robôs; Planejamento de tarefas (robótica); Sensores; Microeletrônica; Inteligência artificial; Engenharia eletrônica			
10. APRESENTAÇÃO: X Nacional Internacional Trabalho de Graduação, ITA, São José dos Campos, 2003. 104 páginas.			
11. RESUMO: <p>Neste trabalho de graduação foi implementado um microcontrolador embarcado em uma plataforma móvel com o objetivo de transferir parte da inteligência do microcomputador para a plataforma de forma a aumentar a autonomia de movimento da mesma. A plataforma foi construída utilizando motores de passo acoplados a duas rodas dianteiras e mais duas rodas livres traseiras, um kit, o AES-51, contendo o microcontrolador Intel 80C32 (da família 8051) e sensores de contato capazes de detectar a colisão, além dos circuitos necessários para integração destes conjuntos.</p> <p>A plataforma construída é capaz de realizar os movimentos dentro de um labirinto previamente determinado, sem a necessidade da comunicação física com o computador. Para realizar o movimento dentro do labirinto é necessário inicialmente gerar a trajetória a partir de um programa, desenvolvido num trabalho de graduação anterior pelo aluno Fabio Eiji Yoshitome [1], localizado no microcomputador e então transferi-lo, utilizando interfaces apropriadas, para o microcontrolador. A transmissão é realizada utilizando a interface serial via cabo e após a transmissão o cabo de conexão pode ser retirado e o programa que determinará a trajetória passa a ser executado pelo microcontrolador.</p> <p>Além disso, foram instalados sensores que detectam a colisão, a qual faz com que o programa no microcontrolador seja interrompido, sendo necessário que o operador sinalize para a plataforma que a mesma pode continuar o percurso inicial.</p> <p>A trajetória realizada pela plataforma é satisfatória apesar de não ser ainda a ideal, já que durante a execução desta trajetória incidem erros sobre a plataforma, principalmente sobre as rodas da plataforma, que fazem com que a distância percorrida pelas rodas a cada pulso seja aleatória.</p> <p>Possíveis aplicações vão desde o uso deste tipo de eletrônica embarcada em automóveis guiados por motoristas automáticos, até em aspiradores automáticos capazes de aspirar uma casa sem o auxílio da dona-de-casa.</p>			
12. GRAU DE SIGILO: (X) OSTENSIVO () RESERVADO () CONFIDENCIAL () SECRETO			