# Índice:



**Sistemas de Tempo Real**
Escola Superior de Tecnologia de Viseu

## Introdução

Na sociedade actual, há uma crescente necessidade de se realizar tarefas com eficiência e precisão. A repetitividade de certas tarefas, bem como tarefas que tem de ser realizadas em lugares onde a presença humana se torna difícil, arriscada e até mesmo impossível, como o fundo do mar ou a imensidão do espaço, têm, impulsionado a tentativa de se criar dispositivos, comumente conhecidos por robôs, que de uma forma praticamente autónoma e inteligente realizem essas tarefas sem risco para a vida humana. A robótica tem-se desenvolvido a um ritmo acelerado na procura do robot mais elaborado, com capacidades que á alguns anos atrás seriam de todo inimagináveis aos olhos do ser humano.

A robótica desperta tanta curiosidade e entusiasmo por parte da comunidade cientifica que torna-se uma área multidisciplinar, altamente activa, envolvendo áreas como a engenharia electrotécnica, engenharia mecânica, inteligência artificial, entre muitas outras numa perfeita simbiose, com uma perfeita harmonia, possibilitando assim criar essas maravilhosas "criaturas tecnológicas" , se assim é permitido chamar.

Os robots estão presentes actualmente em várias áreas de nossa sociedade: robots que prestam serviços, como os desactivadores de bombas, robots com a nobre finalidade da pesquisa científica e educacional e até mesmo os robots operários, que se instalaram nas mais

variadas industrias e foram responsáveis pela "segunda Revolução Industrial", revolucionando a produção em série, substituindo a carne e o osso pelo aço, fornecendo maior rapidez na criação e qualidade aos produtos.

Com a rápida aceleração dos computadores e com o grande desenvolvimento da inteligência artificial, as acções dos robots cada vez são mais precisas e fiáveis, e bastante autónomas, a ponto de eles perante uma determinada situação poderem decidir o que fazer, mas mais do que isso, adquirem informação funcionando como conhecimento que vão armazenando e que poderá ser útil quando estiverem perante uma nova situação. São os robots inteligentes.

Até agora o homem tem perfeito controlo sobre a máquina manipulando-a a seu favor, tornando-a cada vez mais poderosa e inteligente, mas será que vai ser sempre assim? Alguma vez o processo se inverterá e a máquina passará a dominar o homem? Estas perguntas não têm uma resposta, pelo menos por agora, só o futuro permitirá responder. Até lá não há que ter receios é preciso criar, a tecnologia comanda a vida.





## Objectivos

No âmbito da disciplina de Sistemas de Tempo Real, foi colocada em cima da “mesa” a proposta de um criar um robot (sistema em tempo real), não inteligente, que recriasse um caminho previamente fornecido através de um ficheiro.

Para programar o sistema foi escolhida a linguagem de programação AdA95, trata-se de uma linguagem de alto nível, imperativa, tipicamente compilável, tendo por objectivo a utilização em sistemas em tempo real, sendo a primeira linguagem de programação orientada ao objecto padronizada internacionalmente.

## Componentes do Robot

- *Computado com processador 486DX2 da Intel*
- *Disco rígido de 50MB*
- *Memória de 8MB*
- *3 Baterias de 12V/7,2Ah*
- *1 Bateria de 2,4V*
- *Um motor de cc*
- *Um motor de passo*
- *1 Driver para o motor de cc*
- *1 Driver para o motor de passo*
- *Um ventilador*
- *Chassi*
- *4 fusíveis de 2,5A*
- *4 rodas de impressora*





## Aspecto do Robot

As seguintes imagens documentam o aspecto que o robot adquiriu ao fim de muito "suor" e "lágrimas".

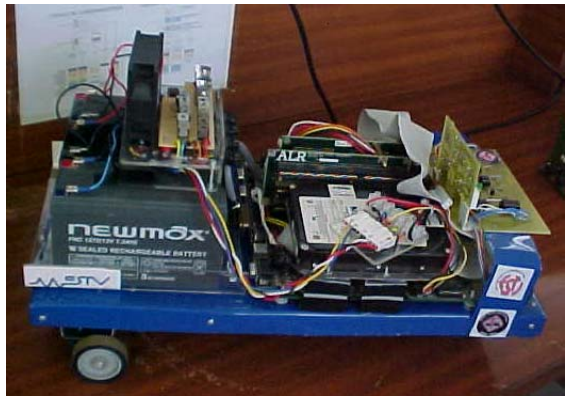



**Figura 1 – Aspecto geral do robot**

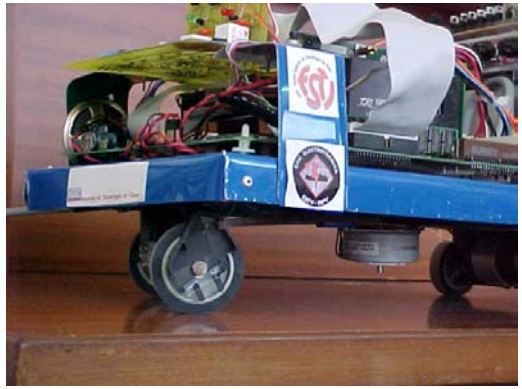


**Figura 2 – Pormenor do motor de passo que faz virar as rodas dianteiras**

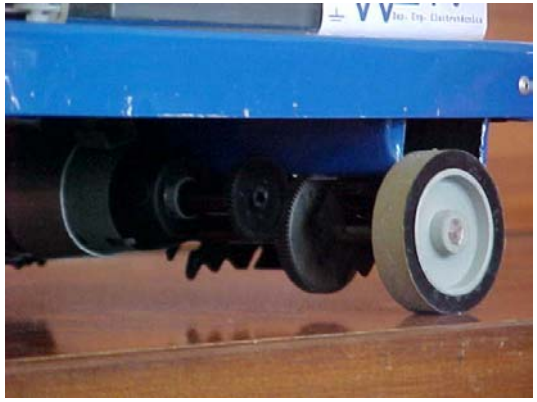


**Figura 3- Pormenor da tracção pelo motor de cc**

**Figura 4 – Pormenor da motherboard incluindo o processador**





## História do Robot

O projecto "Robot" surgiu no âmbito da disciplina de Sistemas de Tempo Real, na qual o docente propôs a elaboração de um robot que se movimentasse de forma autónoma. Devido ao problema relacionado com o prazo previsto para a entrega não foi possível adicionar-lhe sensores de modo a este poder detectar objectos e ter a capacidade de contorná-los.

Iniciamos o projecto com um chassi feito em chapa leve, a qual se moldou e quinou para à posteriori colocarem alguns reforços já que teria de suportar um peso significativo. Durante esta fase não tínhamos ainda a noção da dimensão final do robot (11,5 Kg).

Para que este se movimentasse teríamos de lhe colocar um motor, mas que desenvolve-se um binário elevado e gira-se a baixa rotação. Como teria de ser autónomo, este necessitaria de baterias, assim optamos por um motor de corrente contínua alimentado a 2,4V, que foi adquirido através da compra de uma aparafusadora, que já contém um conjunto de engrenagens redutoras de forma a fornecer o binário desejado. Após algumas dificuldades na fixação adaptamo-la cortando o cabo e acoplando uma roda dentada, fixa ao chassi juntamente com um par de rodas provenientes de uma impressora *HP690* velha que desmontamos.

A direcção fica a cargo de um motor de passo alimentado por baterias de 12V, este foi ligado mecanicamente por meio de uma correia dentada às rodas, fixadas através de um suporte de uma cadeira e peças

provenientes da impressora.

O ângulo de viragem viria a ser limitado pela dificuldade em vencer o atrito das rodas motoras, pelo que a tracção é feita simultaneamente às duas rodas traseiras. Quanto maior for o ângulo de viragem mais as rodas derrapam, esta dificuldade foi detectada após alguns ensaios e não sendo possível a sua correcção devido a uma nova reformulação de todo o sistema de tracção.

O controlo dos dois motores é conseguido por intermédio de um programa em linguagem Ada 95 ‘por detrás da porta paralela de um processador *486*.

Dado que esta porta paralela injecta correntes na ordem dos 2,5mA, que é em muito inferior ao desejado, tivemos de construir duas placas de circuito impresso, uma para cada um dos motores (drivers).

Os circuitos dessas placas foram elaborados mediante a consulta de vários livros, datasheets e sites na internet que com muita imaginação e alterações acabamos no fim de algum tempo por consegui-los de modo a desempenharem as funções pretendidas (comando dos motores).

Posteriormente efectuamos o seu teste em breadboard. Passámos à construção das placas, para tal utilizamos placas foto-sensibilizadas, que devido à sua má qualidade vimos algumas delas destruídas, com a corrosão das pistas.

Após vencermos mais esta dificuldade conseguimos finalmente obter duas placas em estado razoável nas quais foram soldados os componentes.

O robot teria de carregar com elementos provenientes do computador (motherboard, disco, etc), tivemos então de fixa-lo ao chassi.

A alimentação autónoma do computador é conseguida recorrendo





a três baterias de 12V.

Devido a este funcionar com potenciais de +5V, -5V, 0V, 12v e –12V obrigamo-nos a construir uma outra placa onde fossem garantias essas tensões.

Recorremos a uma placa pré-furada na qual foram soldados os componentes necessários para que a alimentação fosse o mais correcto.

Mediante o não funcionamento da motherboard devido a uma das baterias não estar nas condições desejadas, acabamos por efectuar algumas alterações o que conduziu à destruição da mesma, levando à sua substituição.

Com a aplicação de outra mother-board foi possível prosseguir-mos com o nosso projecto.

Devido ao sobreaquecimento de componentes desta ultima placa optamos por lhe adicionar uma ventoinha para refrigeração.

Durante o projecto, o grupo e o docente acabaram por perder algumas noites sono que passaram a ser de trabalho para superar as dificuldades encontradas com a implementação de todos os componentes necessários ao seu funcionamento.

A meta seguinte foi a elaboração do programa que controlasse todo o software implementado anteriormente.

Aqui as dificuldades surpreenderam tudo e todos graças ao compilador para DOS adquirido, que continha muitos bug’s e criava conflitos com instruções que outros compiladores para Windows ultrapassavam.

Deste modo o programa elaborado não foi o desejado, mas o possível de implementar recorrendo aos meios disponíveis.

No fim de todo pronto procedeu-se à apresentação aos colegas e

docentes de Escola Superior de Tecnologia, o robot finalmente após alguns meses de trabalho desempenhava as funções pretendias pelo grupo e docente de disciplina.





## Porta Paralela

A porta paralela é uma interface de comunicação entre o computador e um periférico. Quando a IBM criou o seu primeiro Computador Pessoal, a ideia era interligar a essa Porta uma impressora, mas actualmente, são vários os periféricos que utilizam esta Porta para enviar e receber dados do computador, exemplos disso são os Scanners, Câmaras de vídeo, Unidades de disco removível bem como muitos outros.

No entanto a porta paralela pode ter muitas outras aplicações, especialmente quando se pretende criar um robot controlado por um computador, como é o caso.

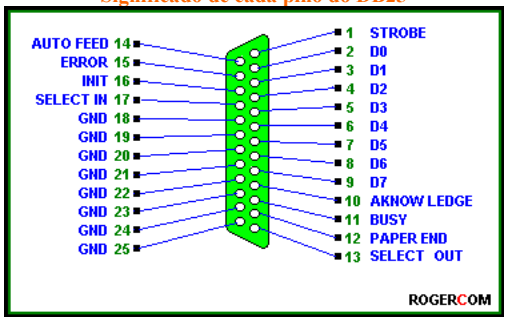
A porta paralela originária da IBM é constituída por 3 portos de 8 bit's cada, o porto de dados, o porto de estados e o porto de controlo. A todo a porta paralela apresenta 12 bits de saída e 5 de entrada.

As entradas e saídas de dados são feitas através do DB25 que é um ligador que fica ligado à motherboard do computador, e é através deste, que o cabo paralelo se conecta ao computador. No DB25, um pino está em nível lógico 0 quando a tensão no mesmo está entre 0 à 0,4v. Um pino encontra-se no nível lógico 1 quando a tensão no mesmo está acima de 3.1 e até 5v. A porta paralela fornece no máximo cerca de 2,5 mA.

A figura abaixo mostra o ligador padrão DB25, com 25 pinos, onde cada pino tem um nome que o identifica:

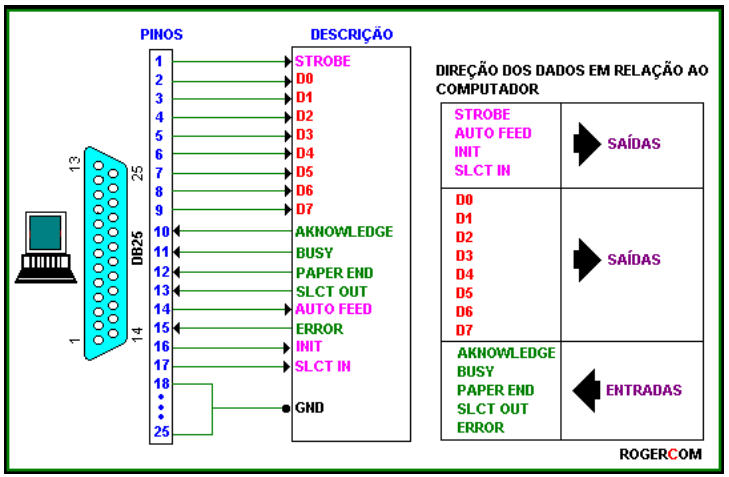
Foto do conector DB25 macho

Significado de cada pino do DB25








A porta paralela possui dois modos de transmissão:

- **Unidireccional**, neste tipo, a comunicação só se dá num sentido. Foi a primeira porta a ser criada. A velocidade de transmissão através da porta paralela SPP (Standard Parallel Port) pode chegar a uma taxa de transmissão de dados a 150KB/s. Comunica-se com a CPU utilizando um BUS de dados de 8 bits. Para a transmissão de dados entre periféricos são usado 4 bits por vez.

- **Bidirecional.** Trata-se de um tipo de porta mais avançada EPP ( Enhanced Parallel Port ), chegando a atingir uma taxa de transferência de 2 MB/s. Para atingir essa velocidade, será necessário um cabo especial.
Comunica-se com a CPU utilizando um BUS de dados de 32 bits.

Para a transmissão de dados entre periféricos são usado 8 bits por vez.

A porta avançada ECP (Enhanced Capabilities Port) tem as mesmas características que a EPP, porém, utiliza DMA (acesso directo à memória), sem a necessidade do uso do processador, para a transferência de dados. Utiliza também um buffer FIFO de 16 bytes.

O computador nomeia as Portas Paralelas, chamando-as de LPT1, LPT2, LPT3 etc, mas, a Porta física padrão do computador é a LPT1, e os seus endereços são: 378h ( para enviar um byte de dados pela Porta), 378+1h (para receber um valor através da Porta) e, 378+2h (para enviar dados). Às vezes pode estar disponível a LPT2, e seus endereços são: 278h, 278+1h e 278+2h, com as mesmas funções dos endereços da porta LPT1 respectivamente.

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

No caso da construção do robot, a porta paralela tem uma função crucial, sem ela não seria possível controlá-lo. A porta paralela é o meio de exteriorizar para a aplicação física (robot) todo o programa criado. Nesta aplicação são controlados pela porta apenas dois dispositivos, um motor de passo e um motor de corrente continua. O motor de passo roda um passo a cada sequência de quatro impulsos, pelo que para este serão necessários 4 bits da porta paralela, cada um deles fornecendo um





impulso. Na rotação inversa do motor de passo apenas se inverte a sequência dos impulsos, pelo que os bits a utilizar são os mesmos. No caso do motor de corrente contínua, para que ele rode num sentido, basta que seja alimentado por 12 Volt. Portanto basta utilizar um bit que funciona como interruptor *on/off*. Para que o motor rode em sentido contrário basta inverter a polaridade da alimentação, assim será necessário mais um bit a funcionar também em *on/off*. Uma configuração especial conseguida à custa de relés permite que a combinação destes dois bits permita os funcionamentos descritos na seguinte tabela.

| Bit 1 | Bit 2 | Movimento |
|---|---|---|
| 1 | 0 | Frente |
| 0 | 1 | Trás |
| 1 | 1 | Parado |
| 0 | 0 | Parado |

Assim são necessários 6 bits para colocar em funcionamento os dois motores utilizados. Perante este requisito utilizou-se os oito bits do porto de dados, mantendo-se dois deles (dois primeiros) sempre a zero uma vez que não têm ligação. O endereço deste porto é 3BC. Este endereço é diferente do habitual (378), uma vez que o computador utilizado é bastante antigo e portanto endereço utilizado é outro.

A tabela seguinte apresenta as combinações completas dos 6 bits utilizados, verificando-se os respectivos movimentos. Foram estas combinações que forma utilizadas na criação do algoritmo do programa, como mais à frente se verá.

| Bit1 | Bit2 | Bit3 | Bit4 | Bit5 | Bit6 | Bit7 | Bit8 | Movimento |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | Frente |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | Trás |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Parar |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | Frente esquerda 1 Impulso |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | Frente esquerda 2 Impulso |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | Frente esquerda 3 Impulso |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | Frente esquerda 4 Impulso |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | Frente direita 1 Impulso |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | Frente direita 2 Impulso |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | Frente direita 3 Impulso |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | Frente direita 4 Impulso |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | Esquerda 1 Impulso |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | Esquerda 2 Impulso |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | Esquerda 3 Impulso |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | Esquerda 4 Impulso |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | Direita 1 Impulso |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | Direita 2 Impulso |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | Direita 3 Impulso |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | Direita 4 Impulso |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | Trás esquerda 1 Impulso |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | Trás esquerda 2 Impulso |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | Trás esquerda 3 Impulso |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | Trás esquerda 4 Impulso |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | Trás direita 1 Impulso |





| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | Trás direita 2 Impulso |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | Trás direita 3 Impulso |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | Trás direita 4 Impulso |

É bom salientar que quando o robot se encontra no movimento frente ou Esquerda, existe um bit (bit 5) que se encontra a 1, isto acontece para que quando o robot andar para a frente ou para trás, o motor de passo esteja bloqueado de forma a manter o ângulo em que se encontra constante ao longo destes movimentos. Também nos movimentos Frente direita, Frente Esquerda, Trás Direita e Trás Esquerda, o Bit inicial de cada primeiro impulso é o bit 5 e não o bit 3 como se poderia pensar. Isto acontece porque quando o computador é ligado ele dá um impulso em todos os bits da porta, desligando-os logo de seguida, no entanto um dos bits permanece com um impulso, impulso esse dado no bit 5, pelo que o motor de passo já recebeu um impulso no bit 5, de forma a que não perca o passo, a restante sequencia de bits deve partir deste.

## Drivers dos Motores

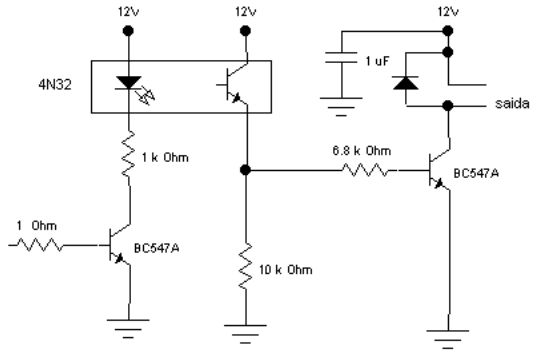
A porta paralela não é um dispositivo de potência, como tal não pode alimentar directamente dispositivos, cujo consumo seja acima de um determinado nível bastante baixo. Os níveis de tensão já foram mencionados atrás que variam entre 0 e 5 V (níveis TTL). A corrente que a porta paralela fornece anda na ordem dos 2,5mA. Portanto para se interligar dispositivos tais como motores, cujo consumo em potência é bastante maior do que aquele que a porta paralela fornece, é necessário construir-se um driver de potência, que permita criar esta compatibilidade de potencias. Assim a porta funcionará não como uma fonte, mas apenas como um dispositivo de controlo.

Um outro aspecto que se inclui neste driver além da elevação de potência, é a protecção que deve ser feita da motherboard. Uma má ligação, uma inversão de polaridade, poderá significar a destruição desta. Portanto é essencial utilizar um componente que possibilite essa protecção. Esse componente foi um acoplador óptico 2N32 que funciona através da emissão e recepção de sinais luminosos, que garante um isolamento galvânico entre a motherboard e as restantes ligações.

Foi necessário criar **dois drivers**, um para o motor de tracção e outro para o motor de passo, mas o seu princípio é bastante idêntico. Ambas as placas se baseiam no circuito base da figura seguinte. De uma placa para a outra apenas variam a aplicação e o número de saídas, cada uma constituída por uma unidade deste circuito base. Cada saída corresponde a um bit da porta paralela.








**Circuito 1 – Circuito base**

| Material Utilizado | Quantidade |
|---|---|
| Transístor BC547A | 2 |
| Resistência 1K | 1 |
| Resistência 10K | 1 |
| Resistência 6.8K | 1 |
| Acopladores ópticos 4N32 | 1 |
| Condensadores 10µF | 1 |
| Diodo | 1 |
| Led | 1 |

O funcionamento deste circuito é bastante simples. É constituído por duas etapas amplificadoras, por transístores, excitando um componente de saída (relé ou amplificador de corrente) que irá fornecer

a corrente necessária para o funcionamento dos motores. Um impulso proveniente da porta paralela equivalente ao accionamento de um bit, é aplicado na entrada do circuito, este impulso muito pequeno na ordem dos 2,5 mA, é depois inserido na base do transístor activando-o. Quando activo deixa passar corrente do colector para o emissor, o que faz com que o 2N32 se active emitindo um feixe luminoso por um foto diodo (interno), que é recebido por um fototransistor. Este por sua vez cria uma corrente de base para o segundo transístor, que irá activar o dispositivo de saída. A resistência colocada na entrada tem por objectivo "sugar" mais corrente da porta paralela, até aos 2,5 mA. Isto é necessário porque sem a resistência a corrente de entrada era muito pequena, pelo que qualquer corrente residual, activava o circuito para um modo oscilatório. O diodo serve para evitar correntes inversas no dispositivo de saída. O condensador permite uma aplicação suave da tensão ao dispositivo de saída.

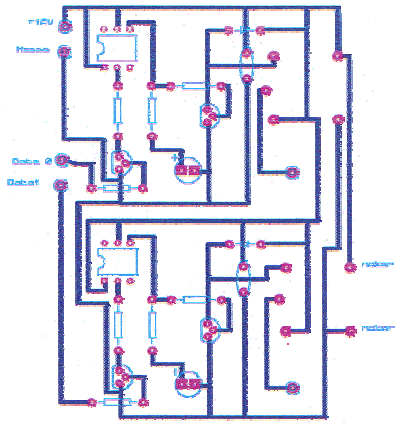
No circuito não está desenhado, mas cada uma das unidades de circuito base contém ainda um led, cuja função é servir de aviso de que a unidade se encontra em funcionamento. No caso do motor cc permitirá identificar o sentido de rotação do mesmo, pelo que quando o led verde está aceso o motor roda para a frente enquanto que quando é o vermelho roda para trás. No caso do motor de passo os quatro led's criam um efeito engraçado, mostrando a sequencia de bits a serem aplicados.





***Driver do motor cc***

O circuito da figura seguinte é referente ao **driver do motor de corrente contínua.** (motor de uma aparafusadora). É um driver alimentado por 12V, que faz a utilização de dois pinos da porta paralela.



**Circuito 2 – Driver do motor de cc**

Conforme se pode constatar este circuito é composto por duas unidades do circuito base. Nas quais o dispositivo de saída são relés. A configuração dos relés é tal que aplicando um bit o motor roda num sentido, aplicando outro o motor roda no sentido inverso. O motor é alimentado apartir de uma bateria individual de 2,4V. Portanto este circuito apenas funciona como interruptor inversor electrónico.

De salientar ainda que como foi mencionado anteriormente, no

instante de arranque do computador, todos os bit's à da porta paralela são colocados a "1", o que poderia fazer com que o motor roda-se num dos sentidos. Tal situação foi contornada pelo circuito de modo a que quando esta situação acontecer é aplicado o mesmo potencial aos dois terminais do motor, assim não havendo diferença de potencial não se proporciona nenhum dos movimentos.

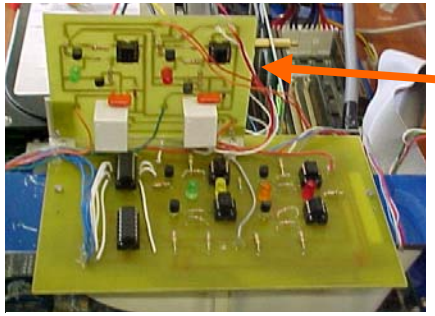


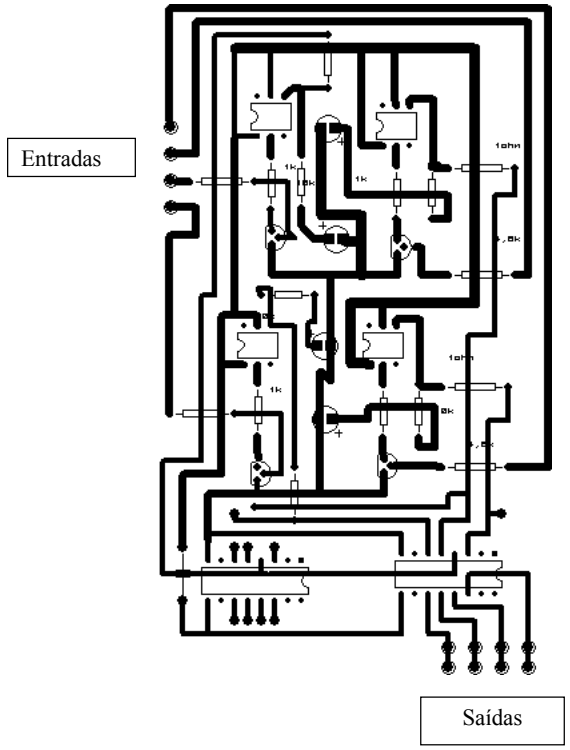

**Figura 5 – Aspecto das placas dos drivers**





***Driver do motor de passo***

O circuito da figura seguinte é referente ao **driver do motor de passo a passo,** (NMB PM42S-048-xxxx). É um driver alimentado também por 12V, que faz a utilização de quatro pinos da porta paralela.




**Circuito 3 – Driver do motor de passo**

O funcionamento deste circuito que controla o motor de passo é

muito semelhante ao anterior, uma vez que é constituído por quatro unidades de circuito básico, onde são aplicados os quatro bits da porta paralela. Apresenta quatro saídas, cada uma ligada a cada um dos quatro terminais do motor de passo. No entanto os dispositivos de saída já não são relés como acontecia anteriormente. Neste circuito são dadas sequências de impulsos com uma frequência elevada, os relés possuem uma inércia própria, pelo que quando se dão comutações a uma frequência elevada, o seu tempo de resposta é elevado, pelo que existe perda de passo. De forma a se contornar esta situação substituiu-se os relés por dois circuitos integrados *TDG2003AP*. O facto de serem dois deve-se ao facto de o circuito poder fornecer uma corrente mais elevada já que cada um deles a limita em *500mA* (muito inferior aos *800mA* consumidos por este motor).

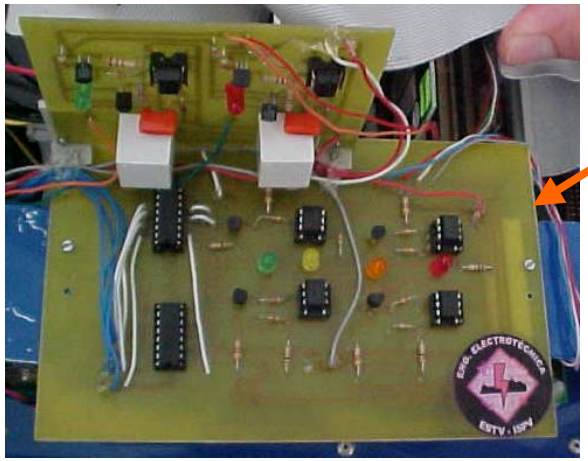



**Figura 6 – Aspecto da placa do driver do motor de passo**





## Alimentação

A criação de um robot pressupõe autonomia, pelo que a alimentação quer do computador quer dos motores foi consumada por baterias.

O computador utilizado faz uso de quatro níveis diferentes de alimentação, +12V, -12V, +5V, -5V e 0V. Esta tensões são normalmente fornecidas a partir de uma fonte de alimentação comutada ligada á rede eléctrica. Como o objectivo é a autonomia, recorre-se a três bateria de lítio de 12V cada. Para se obterem dois dos quatro níveis de tensão desejados recorre-se a dois tipos de reguladores de tensão o *LM7805 e o LM7905* que garantem as tensões +5v e –5V. O circuito que alimenta os +5V consome mais do que os restantes pelo que foram aplicadas duas baterias em paralelo, ficando em contrapolarização com a restante de forma a fornecer os +12V e -12V. O consumo do computador é apreciável, pelo que a utilização de um só regulador era inaceitável, uma vez que com a elevada dissipação de calor, conduziam a quedas internas e ao consequente corte de funcionamento do regulador. Assim colocaram-se vários reguladores em paralelo 10 para o *LM7805* e 6 *para o LM7905*. Este circuito produz uma grande quantidade de calor pelo que foi necessário colocar um ventilador de 12V a dissipar calor. A fonte de alimentação apresenta também fusíveis de protecção de calibre 2,5A.

Na saída desta fonte temos as várias tensões que em situação alguma devem ser trocadas sob pena de danificar irremediavelmente o robot, assim temos o condutor de cor amarelo transportando *+12V*, azul para *–12V*, branco para *–5V*, vermelho para *+5V* e preto para *0V*.

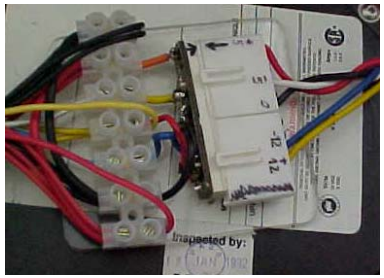



**Figura 7 – Aspecto da ligação da alimentação**

A seguinte figura mostra esta fonte criada para o robot.

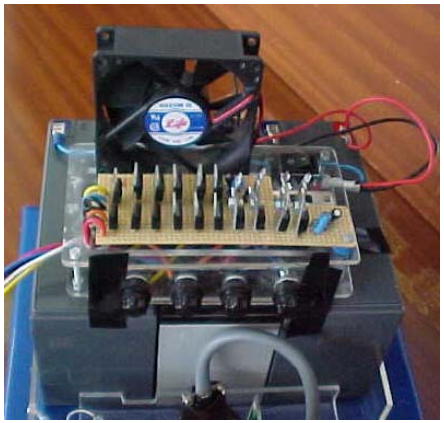


**Figura 8 – Aspecto da fonte de alimentação**





A alimentação do motor de passo é também feita a partir desta fonte, necessitando apenas dos 12V. Já o motor de corrente continua faz uso de uma bateria independente própria para o motor da aparafusadora.

É de salientar que todas estas baterias são recarregáveis.

## O Software

O software para este robot foi desenvolvido numa linguagem de programação de alto nível, utilizada na programação de sistemas de tempo real, o Ada95. É um compilador de Ada criada para funcionar sob o sistema operativo MS-DOS da Microsoft, uma vez que é este que serve de suporte ao microprocessador de controlo do robot. Constatou-se que este compilador apresenta bastantes limitações funcionais, impedindo que os programas criados não pudessem ser mais funcionais, com menor escrita de código. Uma das principais limitações foi a impossibilidade de se puder utilizar "tasks", ou seja realizar a pseudo-concorrencia entre tarefas, uma das ferramentas mais importantes na programação de um sistema em tempo real como é o robot. Devido a esta limitação toda a programação teve de ser feita com base em procedimentos, ou seja programação sequencial.

Foram criados dois programas distintos com funcionalidade bem definidas, que são no entanto dependentes um do outro.

O primeiro programa, chamado "caminho" tem a função de reconhecimento do percurso a ser percorrido pelo veículo. Isso é conseguido pela simulação de um joystick através das teclas direccionais do teclado. Cada uma das teclas marcada por uma seta representa uma direcção bem definida, a título de exemplo a tecla com a seta vertical com a orientação para cima, significa que ao se pressionar esta tecla, o robot irá se deslocar para a frente. A tecla do meio (5 no teclado numérico) recebe a indicação de paragem, à qual o robot responde por imobilização total. Toda a sequência de deslocamento criada pelo





"condutor", através do teclado é guardada num ficheiro que o próprio programa cria, chamando-o de "Data.txt".

Cada um dos movimentos possíveis é reconhecido internamente pelo programa através de mnemónicas. A tabela seguinte apresenta todas as mnemónicas utilizadas, bem como as teclas correspondentes.

| Mnemónica | Movimento | Tecla (teclado numérico) |
|---|---|---|
| ff | Andar para frente | 8 |
| tt | Andar para trás | 2 |
| pp | Parar | 5 |
| te | Trás e esquerda | 1 |
| td | Trás e direita | 3 |
| ve | Virar Esquerda | 4 |
| vd | Virar Direita | 6 |
| fe | Frente e esquerda | 7 |
| fd | Frente e direita | 9 |

De cada vez que uma tecla é pressionada a respectiva mnemónica é registrada no ficheiro, de seguida um relógio interno começa a contar, sendo registrado o seu valor no ficheiro logo após outra tecla ter sido pressionada. Este processo dá-se apenas para as teclas cujo movimento corresponde a frente, trás, frente e esquerda, frente e direita, trás e esquerda e trás direita. Isto porque nestes movimentos, o motor de tracção será sempre activado. Só será desligado quando o bit afecto a ele for colocado a zero, ou seja quando uma nova tecla for pressionada. Por isso a necessidade de um relógio, de forma a contar qual o tempo que decorre enquanto o motor executa um dos seus dois movimentos

possíveis (frente e trás). Estes seis movimentos mencionados anteriormente com excepção da frente e trás activarão também o motor de passo, que fará uma rotação de um passos para o lado correspondente.

Os movimentos de esquerda e direita, não requerem temporizador, uma vez que, o motor de tracção não é activado, funcionando apenas o motor de passo. Este executará uma rotação de cinco passos para o lado correspondente (esquerda ou direita). Apesar de não ser necessário o temporizador, ele irá se activar, sendo também registrado o seu valor no ficheiro, mas isto acontece apenas por uma questão de compatibilidade interna do programa, mais adiante será explicado em pormenor esta situação.

A tecla 0, é a tecla de saída do programa, quando pressionada ela fecha o ficheiro que estava a ser criado, “data.txt”, cria um novo ficheiro chamado “Relatório.txt”, onde regista a data em que foi efectuada a movimentação do robot, bem como o número de movimentos que foram efectuados, de seguida fecha o programa e passa para o sistema operativo.

De forma a se poder perceber melhor o mecanismo de funcionamento do programa, apresenta-se de seguida o código do mesmo, onde serão feitos comentários acerca de aspectos específicos da programação. Por uma questão de não existir repetição desnecessária, sempre que existam partes de código onde a explicação já foi feita anteriormente numa parte de código idêntica, não serão realizados novamente os comentários.

```
with Ada.Integer_Text_Io, Ada.Text_Io, Io_Ports,
Interfaces, Ada.Calendar;
use Ada.Integer_Text_Io, Ada.Text_Io, Io_Ports, Interfaces,
```





```
Ada.Calendar;


procedure Caminho is --nome do programa

package Fix_Io is new Ada.Text_Io.Fixed_Io(Day_Duration);--
pacote do cronometro
use Fix_Io;

   Register               :Unsigned_16;--variável que irá
conter o endereço da porta paralela
   Data                   :Unsigned_8;--valor dos bits a
enviar para a porta
   caminho,relatorio      :file_type;--variáveis de ficheiro
   year,month,day,tecla   :Integer;
   tempo                  :Time;
   Start,Seconds          :Day_Duration;
   validade               :integer;

   procedure Frente is--procedimento de andar para a frente
   begin
      Data:=Unsigned_8(2#00001001#);--valor dos bits a
enviar para a porta
      Register:=Unsigned_16(16#3BC#);--endereço da porta
      Disable_Interrupts;--desactivar os interrupts do
microprocessador
      Write_Io_Port(Register,Data);--escrita na porta
      Enable_Interrupts;--activar os interrupts
   end Frente;

   procedure Tras is--procedimento para andar para trás
   begin
      Data:=Unsigned_8(2#00001010#);
      Register:=Unsigned_16(16#3BC#);
      Disable_Interrupts;
      Write_Io_Port(Register,Data);
      Enable_Interrupts;
   end Tras;

   procedure Parar is--procedimento para parar
   begin
     Data:=Unsigned_8(2#00000000#);
      Register:=Unsigned_16(16#3BC#);
      Disable_Interrupts;
      Write_Io_Port(Register,Data);
      Enable_Interrupts;
```

```
   end Parar;

   procedure Frente_Esquerda  is--procediemnto para andar
para a frente e esquerda
   Register:=Unsigned_16(16#3BC#);
   begin
         Data:=Unsigned_8(2#00000101#);
         Disable_Interrupts;
         Write_Io_Port(Register,Data);
         Enable_Interrupts;
         delay 0.1;--espera do procedimento de 0.1 segundos
         Data:=Unsigned_8(2#00100001#);
         Disable_Interrupts;
         Write_Io_Port(Register,Data);
         Enable_Interrupts;
         delay 0.1;
         Data:=Unsigned_8(2#00010001#);
         Disable_Interrupts;
         Write_Io_Port(Register,Data);
         Enable_Interrupts;
         delay 0.1;
         Data:=Unsigned_8(2#00001001#);
         Disable_Interrupts;
         Write_Io_Port(Register,Data);
         Enable_Interrupts;
         delay 0.1;
   end Frente_Esquerda;

   procedure Esquerda is--procediemnto para andar para
esquerda
         Register:=Unsigned_16(16#3BC#);
   begin
      for i in 1..5 loop--rotação de cinco passos
        Data:=Unsigned_8(2#00000100#);
         Disable_Interrupts;
         Write_Io_Port(Register,Data);
         Enable_Interrupts;
         delay 0.1;
         Data:=Unsigned_8(2#00100000#);
         Disable_Interrupts;
         Write_Io_Port(Register,Data);
         Enable_Interrupts;
         delay 0.1;
         Data:=Unsigned_8(2#00010000#);
         Disable_Interrupts;
         Write_Io_Port(Register,Data);
```





```
         Enable_Interrupts;
         delay 0.1;
         Data:=Unsigned_8(2#00001000#);
         Disable_Interrupts;
         Write_Io_Port(Register,Data);
         Enable_Interrupts;
         delay 0.1;
       end loop;
     end Esquerda;

   procedure Tras_Esquerda is--procediemnto para andar para
tras e esquerda
   begin
        Data:=Unsigned_8(2#00000110#);
         Register:=Unsigned_16(16#3BC#);
         Disable_Interrupts;
         Write_Io_Port(Register,Data);
         Enable_Interrupts;
         delay 0.1;
         Data:=Unsigned_8(2#00100010#);
         Register:=Unsigned_16(16#3BC#);
         Disable_Interrupts;
         Write_Io_Port(Register,Data);
         Enable_Interrupts;
         delay 0.1;
         Data:=Unsigned_8(2#00010010#);
         Register:=Unsigned_16(16#3BC#);
         Disable_Interrupts;
         Write_Io_Port(Register,Data);
         Enable_Interrupts;
         delay 0.1;
         Data:=Unsigned_8(2#00001010#);
         Register:=Unsigned_16(16#3BC#);
         Disable_Interrupts;
         Write_Io_Port(Register,Data);
         Enable_Interrupts;
         delay 0.1;
       end Tras_Esquerda;

   procedure Frente_Direita is--procediemnto para andar
para a frente e direita
   begin
         Data:=Unsigned_8(2#00010001#);
         Register:=Unsigned_16(16#3BC#);
         Disable_Interrupts;
         Write_Io_Port(Register,Data);
```

```
         Enable_Interrupts;
         delay 0.1;
         Data:=Unsigned_8(2#00100001#);
         Register:=Unsigned_16(16#3BC#);
         Disable_Interrupts;
         Write_Io_Port(Register,Data);
         Enable_Interrupts;
         delay 0.1;
         Data:=Unsigned_8(2#00000101#);
         Register:=Unsigned_16(16#3BC#);
         Disable_Interrupts;
         Write_Io_Port(Register,Data);
         Enable_Interrupts;
         delay 0.1;
         Data:=Unsigned_8(2#00001001#);
         Register:=Unsigned_16(16#3BC#);
         Disable_Interrupts;
         Write_Io_Port(Register,Data);
         Enable_Interrupts;
         delay 0.1;
   end Frente_Direita;

   procedure Direita is--procediemnto para andar para a
direita
   begin
      for i in 1..5 loop
         Data:=Unsigned_8(2#00010000#);
         Register:=Unsigned_16(16#3BC#);
         Disable_Interrupts;
         Write_Io_Port(Register,Data);
         Enable_Interrupts;
         delay 0.1;
         Data:=Unsigned_8(2#00100000#);
         Disable_Interrupts;
         Write_Io_Port(Register,Data);
         Enable_Interrupts;
         delay 0.1;
         Data:=Unsigned_8(2#00000100#);
         Disable_Interrupts;
         Write_Io_Port(Register,Data);
         Enable_Interrupts;
         delay 0.1;
         Data:=Unsigned_8(2#00001000#);
         Disable_Interrupts;
         Write_Io_Port(Register,Data);
         Enable_Interrupts;
```





```
         delay 0.1;
         exit when Tecla=5;
      end loop;
      end Direita;

   procedure Tras_Direita is--procediemnto para andar para
tras e esquerda
   begin
         Data:=Unsigned_8(2#00010010#);
         Register:=Unsigned_16(16#3BC#);
         Disable_Interrupts;
         Write_Io_Port(Register,Data);
         Enable_Interrupts;
         delay 0.1;
         Data:=Unsigned_8(2#00100010#);
         Disable_Interrupts;
         Write_Io_Port(Register,Data);
         Enable_Interrupts;
         delay 0.1;
         Data:=Unsigned_8(2#00000110#);
         Disable_Interrupts;
         Write_Io_Port(Register,Data);
         Enable_Interrupts;
         delay 0.1;
         Data:=Unsigned_8(2#00001010#);
         Disable_Interrupts;
         Write_Io_Port(Register,Data);
         Enable_Interrupts;
         delay 0.1;
   end Tras_Direita;


begin
   validade:=0;--variável que contém o numero de
movimentos, inicializada a zero
   create(caminho, out_file, "Data.txt");--cria o ficheiro
"Data.txt"
   tempo:=clock;--lê no relogio do computador o tempo neste
instante
   Split(tempo, year, month, day, Start);--separa o valor
de tempo lido nas suas 4 componentes
   loop
      Put_Line("Escreva Tecla");
      get(tecla);--leitura de uma tecla
      if validade=0 then
         Null;--se validade for igual a zero não faz nada,
```

```
pois é o primeiro movimento
      else
         tempo:=clock;--se não é zero recolhe o tempo que o
movimento anterior durou
         Split(tempo, year, month, day, Seconds);
         Put(caminho,Seconds - Start,1,1);--(coloca o valor
no ficheiro, subtraindo ao tempo lido
         ;--no inicio do movimento ao do final do mesmo, o
que dá a duração do movimento)
         new_line
      end if;

      if Tecla = 8 then--se a tecla for 8 (seta para cima)
         Frente;--chama procedimento de andar para a frente
         Put (caminho,"ff");--coloca no ficheiro a
respectiva mnemónica
      end if;

      if Tecla=2 then
         Tras;
         Put (caminho, "tt");
      end if;
      if Tecla =5 then
         Parar;
         Put (caminho, "pp");
      end if;

      if Tecla= 9 then
         Frente_Direita;
         Put (caminho, "fd");
      end if;

      if Tecla= 6 then
         Direita;
         Put (caminho, "vd");
      end if;

      if Tecla= 4 then
         Esquerda;
         Put (caminho, "ve");
      end if;

      if Tecla=7 then
         Frente_Esquerda;
         Put (caminho, "fe");
      end if;
```





```
      if Tecla=3 then
         Tras_Direita;
         Put (caminho, "td");
      end if;

      if Tecla=1 then
         Tras_Esquerda;
         Put (caminho, "te");
      end if;

      if tecla=0 then
         close(caminho);--feicha o ficheiro Data.txt
         create(relatorio, out_file, "Relatorio.txt");--
cria o ficheiro Relatório.txt
         Put(relatorio,"O robot rodou em ");
         tempo:= Clock;
         Split(tempo, Year, Month, Day, Seconds);
         Put(relatorio,Day, 3);--coloca no ficheiro o dia
         Put(relatorio,Month, 3 );--coloca no ficheiro o
mes
         Put(relatorio,Year, 5);--coloca no ficheiro o ano
         Put(relatorio,validade);----coloca no ficheiro o
numero de movimentos
         Put(relatorio," movimentos");
         close(relatorio);--fecha o ficheiro Relatório.txt
         exit;--abandona o programa
      end if;
      tempo:=clock;--recolhe o tempo
      Split(tempo, year, month, day, Start);
      validade:=validade+1;--incrementa os movimentos
   end loop;

end Caminho;
```

O segundo programa chamado “final” é complementar a este anterior.

Faz uma leitura sequencial do ficheiro criado pelo programa anterior (Data.txt). Ou seja, lê cada uma das mnemónicas (iguais ás do programa anterior de forma a haver compatibilidade) e respectivos

tempos aí escritos. As mnemónicas e tempos correspondem a um dado movimento que o robot irá traduzir na prática. A leitura da sequência tem de ser feita na ordem correcta, primeiro a mnemónica e depois o tempo, pois caso contrário irá dar-se um erro. Por este motivo é que no programa anterior mesmo não sendo necessário, cada movimento tinha associado o cronómetro, sendo o seu valor gravado no ficheiro, para que este programa o possa ler.

Se por algum motivo o ficheiro o ficheiro data.txt não foi criado, o programa final quando o tenta ler e verifica que ele não existe, levanta uma excepção (Name_error), cujo tratamento corresponde à criação de um ficheiro chamado "temp.txt", onde são inseridos dois movimentos predefinidos (um segundo para a frente e um segundo para trás), de forma a dar a indicação de que o ficheiro data.txt não existe, não abortando abruptamente o programa.

Apresenta-se de seguida o código deste segundo programa com os respectivos comentários.

```
with Ada.Integer_text_IO,Ada.Float_text_IO, Ada.Text_Io,
Io_Ports, Interfaces, Ada.Calendar;
use Ada.Integer_Text_IO,Ada.Float_text_IO, Ada.Text_Io,
Io_Ports, Interfaces, Ada.Calendar;

procedure Final is--nome do programa

          F_trajecto     :File_Type;
            movimento      :string(1..2);
          distancia      :float;
          tempo             :Day_Duration;
          Register     :Unsigned_16;
          Data         :Unsigned_8;
            nome           :string(1..8);
            bol            :Boolean:=false;

          procedure frente is
```





```
begin
Data:=Unsigned_8(2#00001001#);
Register:=Unsigned_16(16#3BC#);
  Disable_Interrupts;
Write_Io_Port(Register,Data);
Enable_Interrupts;
end frente;

procedure tras is
begin
Data:=Unsigned_8(2#00001010#);
Register:=Unsigned_16(16#3BC#);
  Disable_Interrupts;
Write_Io_Port(Register,Data);
Enable_Interrupts;
end tras;

procedure parar is
begin
  Data:=Unsigned_8(2#00000000#);
Register:=Unsigned_16(16#3BC#);
Disable_Interrupts;
Write_Io_Port(Register,Data);
Enable_Interrupts;
end parar;

procedure frente_esquerda  is
begin
     Data:=Unsigned_8(2#00000101#);
     Register:=Unsigned_16(16#3BC#);
     Disable_Interrupts;
     Write_Io_Port(Register,Data);
     Enable_Interrupts;
     delay 0.1;
     Data:=Unsigned_8(2#00100001#);
     Register:=Unsigned_16(16#3BC#);
     Disable_Interrupts;
     Write_Io_Port(Register,Data);
     Enable_Interrupts;
        delay 0.1;
     Data:=Unsigned_8(2#00010001#);
     Register:=Unsigned_16(16#3BC#);
     Disable_Interrupts;
     Write_Io_Port(Register,Data);
     Enable_Interrupts;
        delay 0.1;
```

```
      Data:=Unsigned_8(2#00001001#);
      Register:=Unsigned_16(16#3BC#);
      Disable_Interrupts;
      Write_Io_Port(Register,Data);
      Enable_Interrupts;
      delay 0.1;
      end frente_esquerda;

procedure esquerda is
begin
        for i in 1..5 loop
        Data:=Unsigned_8(2#00000100#);
      Register:=Unsigned_16(16#3BC#);
      Disable_Interrupts;
      Write_Io_Port(Register,Data);
      Enable_Interrupts;
      delay 0.1;
      Data:=Unsigned_8(2#00100000#);
      Register:=Unsigned_16(16#3BC#);
      Disable_Interrupts;
      Write_Io_Port(Register,Data);
      Enable_Interrupts;
         delay 0.1;
      Data:=Unsigned_8(2#00010000#);
      Register:=Unsigned_16(16#3BC#);
      Disable_Interrupts;
      Write_Io_Port(Register,Data);
      Enable_Interrupts;
         delay 0.1;
      Data:=Unsigned_8(2#00001000#);
      Register:=Unsigned_16(16#3BC#);
      Disable_Interrupts;
      Write_Io_Port(Register,Data);
      Enable_Interrupts;
      delay 0.1;
         end loop;
      end esquerda;

procedure tras_esquerda is
begin
      Data:=Unsigned_8(2#00000110#);
      Register:=Unsigned_16(16#3BC#);
      Disable_Interrupts;
      Write_Io_Port(Register,Data);
      Enable_Interrupts;
      delay 0.1;
```





```
      Data:=Unsigned_8(2#00100010#);
      Register:=Unsigned_16(16#3BC#);
      Disable_Interrupts;
      Write_Io_Port(Register,Data);
      Enable_Interrupts;
         delay 0.1;
      Data:=Unsigned_8(2#00010010#);
      Register:=Unsigned_16(16#3BC#);
      Disable_Interrupts;
      Write_Io_Port(Register,Data);
      Enable_Interrupts;
         delay 0.1;
      Data:=Unsigned_8(2#00001010#);
      Register:=Unsigned_16(16#3BC#);
      Disable_Interrupts;
      Write_Io_Port(Register,Data);
      Enable_Interrupts;
      delay 0.1;
      end tras_esquerda;

procedure frente_direita is
begin
      Data:=Unsigned_8(2#00010001#);
      Register:=Unsigned_16(16#3BC#);
      Disable_Interrupts;
      Write_Io_Port(Register,Data);
      Enable_Interrupts;
      delay 0.1;
      Data:=Unsigned_8(2#00100001#);
      Register:=Unsigned_16(16#3BC#);
      Disable_Interrupts;
      Write_Io_Port(Register,Data);
      Enable_Interrupts;
         delay 0.1;
      Data:=Unsigned_8(2#00000101#);
      Register:=Unsigned_16(16#3BC#);
      Disable_Interrupts;
      Write_Io_Port(Register,Data);
      Enable_Interrupts;
         delay 0.1;
      Data:=Unsigned_8(2#00001001#);
      Register:=Unsigned_16(16#3BC#);
      Disable_Interrupts;
      Write_Io_Port(Register,Data);
      Enable_Interrupts;
      delay 0.1;
```

```
end frente_direita;

procedure direita is
begin

     Data:=Unsigned_8(2#00010000#);
     Register:=Unsigned_16(16#3BC#);
     Disable_Interrupts;
     Write_Io_Port(Register,Data);
     Enable_Interrupts;
     delay 0.1;
     Data:=Unsigned_8(2#00100000#);
     Register:=Unsigned_16(16#3BC#);
     Disable_Interrupts;
     Write_Io_Port(Register,Data);
     Enable_Interrupts;
        delay 0.1;
     Data:=Unsigned_8(2#00000100#);
     Register:=Unsigned_16(16#3BC#);
     Disable_Interrupts;
     Write_Io_Port(Register,Data);
     Enable_Interrupts;
        delay 0.1;
     Data:=Unsigned_8(2#00001000#);
     Register:=Unsigned_16(16#3BC#);
     Disable_Interrupts;
     Write_Io_Port(Register,Data);
     Enable_Interrupts;
     delay 0.1;

end direita;

procedure tras_direita is
begin

     Data:=Unsigned_8(2#00010010#);
     Register:=Unsigned_16(16#3BC#);
     Disable_Interrupts;
     Write_Io_Port(Register,Data);
     Enable_Interrupts;
     delay 0.1;
     Data:=Unsigned_8(2#00100010#);
     Register:=Unsigned_16(16#3BC#);
     Disable_Interrupts;
     Write_Io_Port(Register,Data);
```





```
     Enable_Interrupts;
        delay 0.1;
     Data:=Unsigned_8(2#00000110#);
     Register:=Unsigned_16(16#3BC#);
     Disable_Interrupts;
     Write_Io_Port(Register,Data);
     Enable_Interrupts;
        delay 0.1;
     Data:=Unsigned_8(2#00001010#);
     Register:=Unsigned_16(16#3BC#);
     Disable_Interrupts;
     Write_Io_Port(Register,Data);
     Enable_Interrupts;
     delay 0.1;
  end tras_direita;
begin
    Declare
    begin
       Put_Line("Qual o nome do ficheiro a correr
(----.txt) ?");
       get(nome);--nome do ficheiro a correr
     open(F_Trajecto,In_File,nome);
       exception--levanta uma excepção se o
ficheiro não existir
       when Name_Error =>--tratamento da excepção
       begin
          Create(F_Trajecto, out_File, "temp.txt");
          Put(F_Trajecto, "pp00ff01pp00tt01pp00");
          Close(F_Trajecto);
          bol:=true;
       end;
    end;

    if bol=true then
    open(F_trajecto, In_File, "Temp.txt");
    end if;

   loop
    exit when End_Of_File(F_Trajecto);
    get(F_Trajecto,movimento);
    get(F_Trajecto,distancia);
  tempo:=Day_Duration(distancia);--conversão
explicita
    If movimento = "ff" then
       frente;
       delay tempo;
```

```
            End if;

            If movimento="tt" then
               tras;
               delay tempo;
            End if;

            If movimento ="pp" then
               parar;
            End if;

            If movimento= "fd" then
               frente_direita;
               delay tempo;
             End if;

            If movimento= "vd" then
               direita;
            End if;

            If movimento= "ve" then
               esquerda;
            End if;

            If movimento="fe" then
               rente_esquerda;
               delay tempo;
            End if;

            If movimento="td" then
               tras_direita;
               delay tempo;
            End if;

            If movimento="te" then
               tras_esquerda;
               delay tempo;
            End if;
          end loop;
close(F_Trajecto);--fecha o ficheiro
end escrever;
```



Sistemas de Tempo Real
Escola Superior de Tecnologia de Viseu

## Problemática

Este robot apresenta um problema. O problema é que o robot em determinadas situações não consegue repetir o trajecto para o qual foi antecipadamente programado. Esse problema deve-se ao facto de a tracção ser traseira fazendo rodar em simultâneo as duas rodas. Numa situação de viragem dá-se um escorregamento das rodas dianteiras. Como este escorregamento é incalculável, o trajecto torna-se também imprevisível. Existe uma forma de contornar o problema que passa por não se definirem trajectórias com curvas muito apertadas. No futuro seria aceitável criar-se outra solução para a tracção do robot.





## Conclusão

O somatório do esforço dispendido com as vantagens adquiridas com a realização deste trabalho é de todo positivo. Sem dúvida que foi enriquecedor criar este robot, não só por ser um projecto inovador que pôs à prova a nossa capacidade de investigação e imaginação, mas também porque os conhecimentos que derivaram da sua construção foram mais do que muitos. Apesar dos receios iniciais em relação ao término deste projecto, a motivação para dar forma a um conjunto de impressoras avariadas foi crescendo á medida que as várias etapas se iam alcançando. É verdade que a determinada altura a vontade de desistir surgiu, pois por vezes a inexperiência leva a que se comentam erros cujos custos principalmente em tempo são elevados, no entanto com o apoio do docente, o encorajamento mutuo do grupo levou a que não se parasse e a obra surgiu.

O robot não está acabado, longe disso, neste momento não passa de um simples “robozinho”, se assim se pode chamar, sem inteligência e com muitas melhorias a serem feitas. Quer essas melhoria sejam feitas por nós ou por outros colegas, o importante é que se continue a desenvolver e a criar, pois só assim, deparando-nos com problemas práticos é que podemos por em evidencia a teoria aprendida e desenvolver as nossas capacidade inventiva e de perseverança.

È claramente evidente no seio deste grupo de quatro elementos a satisfação e o agrado por termos levado a “bom porto” a criação de um robot.

## Realizado por:

*Projecto realizado no âmbito da disciplina de Sistemas de Tempo Real, do 4º ano da licenciatura em Engenharia Electrotécnica por:*


- Tiago Pais de Almeida 3159
- António Lopes 3004
- Paulo Branco 3364
- Nuno Carvalho