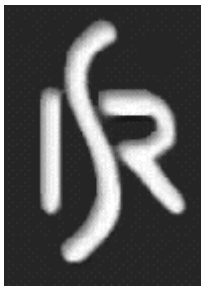


Instituto de Sistemas e Robótica

Pólo de Lisboa

Kit para Controlo de Robots[1]

Jorge Paiva

Luís Tavares

Michael Salgueiro

Miguel Lombo

João Sequeira

Junho 2006

RT-602-06

ISR Torre Norte

Av. Rovisco Pais

1049-001 Lisboa

Portugal

[1] Trabalho suportado pelo projecto FCT POSI/SRI/40999/2001 - SACOR e pelo Programa Operacional Sociedade da Informação (POSI) no quadro do QCA III.

Índice

1. Introdução

1.1 Descrição

O desenvolvimento de plataformas robóticas é uma actividade dispendiosa, principalmente devido ao custo das estruturas físicas, o que inviabiliza esta actividade em muitas instituições. A construção de protótipos usando kits Mindstorm, da LEGO, tem sido a opção adoptada em diversos meios académicos, devido à sua versatilidade, possibilidade de reutilização e baixo custo. Esta alternativa possui no entanto a desvantagem de o número de actuadores e sensores suportados pelos RCX da LEGO ser bastante reduzido. O Kit de Interface para LEGO surge como resposta a esta limitação. O hardware desenvolvido permite a utilização de uma vasta gama de periféricos, podendo usar-se até 31 em simultâneo. As comunicações são efectuadas via rádio, estando incluído no kit uma placa rádio para ligação ao PC. O software disponibilizado para o kit viabiliza a sua utilização numa grande variedade de aplicações, garantindo ao utilizador uma gestão eficiente e uma interface simples com as potencialidades do sistema.

Este manual está dividido em três secções principais, respeitantes à descrição do hardware, à interface para a placa de controlo e ao software de comunicação rádio para o PC.

1.2 Principais Características

- Capacidade máxima para 31 periféricos
- Utilização flexível de uma vasta gama de periféricos
- Frequências de operação dos periféricos independentes e configuráveis
- Autonomia mínima de 1h
- Opção de alimentação por transformador
- Alimentação independente para motores
- Comunicação via rádio
- Potência e frequência de comunicação configuráveis
- Programação da placa de controlo via MPLab ICD2 ou compatível
- Software de interface com o hardware simples (linguagem C)
- Software de comunicação rádio para PC (ambiente Matlab)

1.3 Periféricos suportados

- 6 PWM (*Pulse Width Modulation*) para actuação de motores
- 31 Sensores de Contacto
- 10 Sensores Activos
    - Sensores de Rotação (LEGO)
    - Sensores de Proximidade
- 6 Sensores de Rotação de alta Resolução
- 7 Sensores Temporais
    - Acelerómetro de 2 eixos
    - Sonares

1.4 Material do kit

- 1 Placa de Controlo
- 1 Placa Rádio para PC
- 1 Placa de Interface para Módulos de Potência
- 4 Módulos de potência
- 1 Cabo de programação compatível com MPLab ICD2
- 1 Cabo flat de ligação entre placas
- 1 Bateria de LiPo, 3 células, 11,1 V, 2000 mAh
- 1 Bateria de LiPo, 2 células, 7,4 V, 1000 mAh
- Software de interface com o hardware
- Software de comunicação rádio para PC

1.5 Material extra

- Placa com acelerómetro de 2 eixos
- Cabos de interface com motores e sensores LEGO
- Cabos de interface para outros periféricos

2. Software da Placa de Controlo

2.1 Noções Básicas

O software para a Placa de Controlo é uma interface, em linguagem de programação C, para trabalhar com periféricos e comunicação rádio. De forma a optimizar o desempenho do sistema fornecido, existe um ficheiro (*USER_CONFIG.h*) com informação relativa a todos os sensores e actuadores conectados às placas, que permite compilar e executar apenas o código necessário. Neste ficheiro é ainda possível alterar a frequência de funcionamento dos diferentes tipos de periféricos, tornando-os compatíveis com um maior conjunto de aplicações. Embora existam indicações de preenchimento dentro do ficheiro com as definições, deve ser consultada a documentação referente a cada periférico, presente na secção de Hardware deste manual.

2.2 Ambiente de desenvolvimento de projectos

O ambiente de desenvolvimento de projectos aconselhado é o MPLAB IDE. Este programa oferece diversas funcionalidades úteis ao aperfeiçoamento de software, como simulação do PIC, "in circuit debug" e programação de PICs (através do MPLAB ICD2). Este programa não inclui, no entanto, um compilador para C. O utilizador deve por isso instalar o MPLAB IDE, adicionando posteriormente o compilador MCC18.

2.3 Inclusão do software em projectos

O software para a placa de controlo é disponibilizado na forma de bibliotecas (ficheiros de código). Cada biblioteca possui funções para um conjunto específico de periféricos. O utilizador deve assim incluir nos seus projectos todos os ficheiros fornecidos, cuja listagem se encontra na Secção 2.5. Caso esteja a ser utilizado o MPLAB IDE, como ambiente de desenvolvimento, é possível usar ficheiros de projecto (*work.mcp* e *work.mcw*) onde já estão incluídos todos os ficheiros e efectuadas as configurações necessárias.

2.4 Programação da Placa de Controlo

A programação do PIC, incorporado na Placa de Controlo, pode ser realizada sem que seja necessário proceder à remoção deste. Existe para este efeito, na Placa de Controlo, um terminal onde deve ser conectado o cabo que efectua a ligação ao dispositivo da Microchip, MPLAB ICD 2 ou compatível. Caso o ambiente de desenvolvimento de projectos seja o aconselhado, é possível proceder à programação do PIC directamente, sem que seja necessário outro programa.

2.5 Estrutura de ficheiros

O software fornecido, para interface com a Placa de Controlo, foi desenvolvido de forma modular, encontrando-se organizado em bibliotecas. Estas relacionam-se de acordo com o seguinte diagrama:

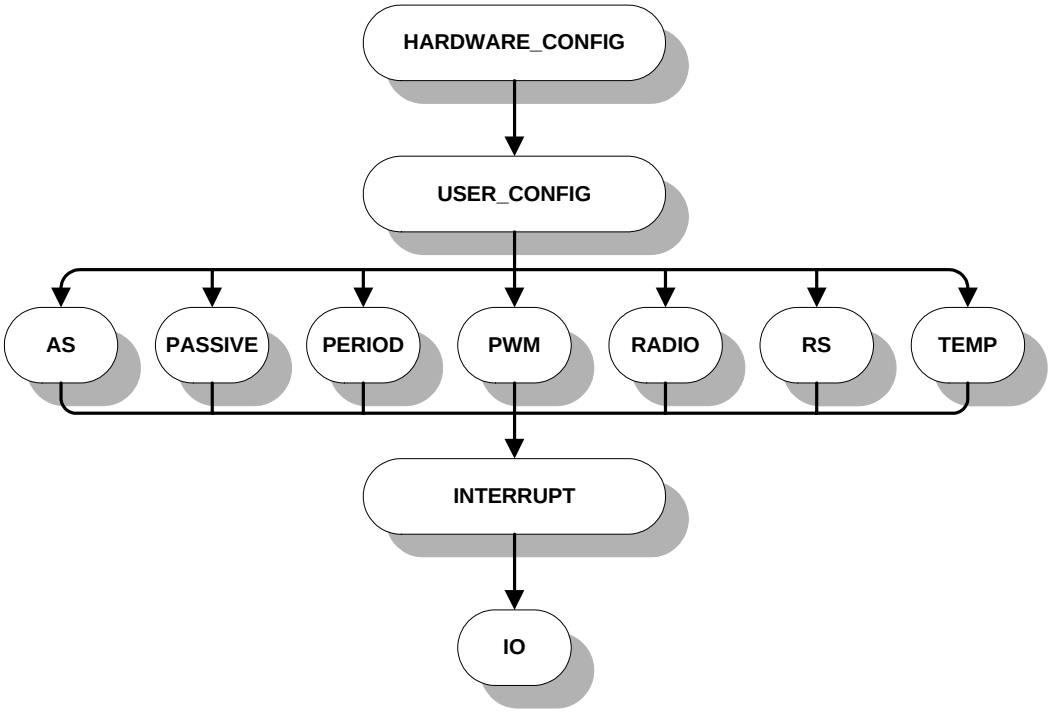


Figura 1 – Estrutura do software da Placa de Controlo.

Os ficheiros que corporizam a estrutura acima são:

AS.h	*AS.c*
HARDWARE_CONFIG.h	
INTERRUPT.h	*INTERRUPT.c*
IO.h	*IO.c*
PASSIVE.h	*PASSIVE.c*
PERIOD.h	*PERIOD.c*
PWM.h	*PWM.c*
RADIO.h	*RADIO.c*
RS.h	*RS.c*
TEMP.h	*TEMP.c*
USER_CONFIG.h	

Os ficheiros *HARDWARE_CONFIG.h* e *USER_CONFIG.h* possuem informação relativa aos periféricos que estão ligados às placas. Apenas o *USER_CONFIG.h* deve ser preenchido pelo utilizador. Estas informações são disponibilizadas a todas as bibliotecas, que adaptam o sistema de gestão de periféricos à configuração actual, de forma a obter um desempenho optimizado.

As bibliotecas AS (Sensores Activos), PASSIVE (Sensores Passivos), PERIOD (Temporizador), PWM (geração de sinais de comando PWM), RADIO (Comunicação Rádio), RS (Sensores de Rotação) e TEMP (Sensores Temporais) possuem funções que permitem o uso de recursos e periféricos específicos da Placa de Controlo. Estas estão incluídas em duas "super bibliotecas", uma que realiza a interface com o utilizador (IO), devendo ser a única a ser incluída nos seus ficheiros de código (**Exemplo 1**). A outra biblioteca é a INTERRUPT, que possui o código que irá executar concorrentemente com o do utilizador.

```
#include "IO.h"

void main(void)
{
      // Inicia o sistema de gestão dos recursos da Placa de Controlo
      init_IO();

      //
      // Código do utilizador
      //
}
```

Exemplo 1 – Inclusão das bibliotecas da Placa de Controlo em ficheiros de código.

2.6 Custo Computacional

O custo computacional do software da Placa de Controlo é uma medida que avalia o uso que o sistema de gestão dos recursos faz do PIC. Este custo varia inevitavelmente com o número de recursos a gerir, sendo garantido um valor inferir a 5,21% do tempo do processador. As fórmulas seguintes permitem calcular o custo computacional associada a cada tipo de recursos:

Custo computacional dos sinais PWM:

$$CustoPWM\,(\%) = \frac{26 + 2 \cdot N\_PWM}{1024} \cdot 100$$

Onde:

N_PWM - Número de sinais PWM a gerar

Custo computacional dos Sensores Activos (AS):

$$CustoAS(\%) = \left(\frac{2}{1024} + \frac{7 + 48 \cdot N\_AS + N\_AS^2}{1024 \cdot AS\_CNT\_MAX} \right) \cdot 100$$

Onde:

N_AS - Número de Sensores Activos a ler

AS_CNT_MAX - Frequência de leitura (configurável em *USER_CONFIG.h*)

Custo computacional dos Sensores de Rotação (RS):

$$CustoRS(\%) = \left(\frac{3}{1024} + \frac{2 + 12 \cdot N\_RS}{1024 \cdot RS\_CNT\_MAX} \right) \cdot 100$$

Onde:

N_RS - Número de Sensores de Rotação a ler

RS_CNT_MAX - Frequência de leitura (configurável em *USER_CONFIG.h*)

Custo computacional dos Sensores Temporais (TEMP):

$$CustoTEMP(\%) = \left(\frac{3}{1024} + \frac{2 + 93 \cdot N\_TEMP + 5 \cdot N\_TEMP^2}{1024 \cdot 127 \cdot TEMP\_CNT\_MAX} \right) \cdot 100$$

Onde:

N_TEMP - Número de Sensores Temporais a ler

$TEMP\_CNT\_MAX$ - Frequência de leitura (configurável em `USER_CONFIG.h`)

Custo computacional da Comunicação Rádio (RADIO):

$$CustoEnvioRADIO(\%) = \left(\frac{3}{1024 \cdot 127} + \frac{45 + 39 \cdot N\_SEND\_BYTES}{1024 \cdot 127 \cdot RADIO\_CNT\_MAX} \right) \cdot 100$$

$$Custo\,\mathrm{Re}cepçãoRADIO(\%) = \left(\frac{8 + 33 \cdot N\_RECV\_BYTES}{1024 \cdot 127 \cdot RADIO\_CNT\_MAX} \right) \cdot 100$$

Onde:

N_SEND_BYTES - Número de bytes a transmitir

N_RECV_BYTES - Número de bytes a receber

$RADIO\_CNT\_MAX$ - Frequência de comunicação (configurável em `USER_CONFIG.h`)

Custo computacional do Temporizador (PERIOD):

$$CustoPERIOD(\%) = \left(\frac{3}{1024 \cdot 127} + \frac{3}{1024 \cdot 127 \cdot PERIOD\_CNT\_MAX} \right) \cdot 100$$

Onde:

$PERIOD\_CNT\_MAX$ - Frequência do Temporizador (configurável em `USER_CONFIG.h`)

O custo computacional total resulta da soma de todas a componentes, sendo dado por:

$$\begin{aligned} CustoTotal(\%) = {} & CustoPWM + CustoAS + CustoRS + CustoTEMP + \\ & CustoEnvioRADIO + Custo\,\mathrm{Re}cepçãoRADIO + \\ & CustoPERIOD \end{aligned}$$

2.7 Recursos do PIC ocupados

O software desenvolvido para gerir a placa de Controlo ocupa recursos do PIC, que passam a estar indisponíveis para o utilizador. A alteração de registos associados a estes recursos leva ao funcionamento errático do sistema de gestão. Para informação mais detalhada deve ser consultado o manual do PIC18F4520. A listagem seguinte identifica os recursos ocupados pelo software:

- TIMER0, TIMER1 e TIMER3
- Módulos CCP1 e CCP2
- Módulo de Conversão A/D
- Módulo de Comunicação EUSART
- Interrupções de alta prioridade
- Interrupções externas
- Interrupções de alteração do valor de pinos
- Memória do banco de acesso rápido

2.8 Ficheiros de Configuração de Hardware

Os ficheiros de configuração de Hardware, *HARDWARE_CONFIG.h* e *USER_CONFIG.h*, permitem ao sistema saber quais os recursos que deve gerir e onde é que estes estão conectados. O ficheiro *HARDWARE_CONFIG.h* possui definições relativas às placas e ao sistema de gestão, pelo que não deve ser alterado. O ficheiro *USER_CONFIG.h* possibilita ao utilizador especificar quais os periféricos que estão ligados, onde estão ligados e com que frequências devem ser lidos/actuados. Este ficheiro encontra-se dividido em diversas secções, sendo na primeira possível definir as frequências de operação e nas restantes indicar quais os periféricos de cada tipo que estão activos. Para cada periférico é necessário indicar se este está activo e qual o porto e pino a que está ligado. A especificação do porto obriga ao preenchimento de dois campos, TRIS e LAT/PORT, que são usados para configurar os pinos da placa como entrada/saída e o seu valor lógico (para mais informação relativa a estes campos, consulte o manual do PIC). É possível visualizar a distribuição dos pinos e portos nas placas na **Secção 4.2.3.** O exemplo seguinte configura a geração do sinal PWM 0 nos pinos 2 e 3 do porto A.

```
...
//
//     Definição dos PWM
//

// Define o estado dos PWM (ON/OFF)
#define PWM0_STATE  ON
#define PWM1_STATE  OFF
#define PWM2_STATE  OFF
#define PWM3_STATE  OFF
#define PWM4_STATE  OFF
#define PWM5_STATE  OFF

// Configura o PWM0
#if PWM0_STATE
#define PWM0_TRIS   TRISA
#define PWM0_LAT    LATA
#define PWM0_PIN0   2
#define PWM0_PIN1   3
#endif
...
```

Exemplo 2 – Configuração da geração do sinal de PWM 0.

2.9 Comunicação Rádio

A Placa de Controlo incorpora um transmissor/receptor rádio [2], usado para comunicar com a Placa Rádio (que liga ao PC via porta série). O software para PC, usado para interagir com esta placa, encontra-se descrito na **Secção 3**. Para o PIC é disponibilizada uma camada de comunicação de baixo nível, que garante acesso exclusivo ao meio, verificação de consistência das mensagens, controlo de perdas e detecção de falhas na comunicação. São ainda fornecidas funções que permitem configurar o canal (0 - 9) e a potência de transmissão (1 - 10mW), possibilitando uma gestão eficiente da bateria e a utilização em simultâneo de diversos kits.

2.9.1 Protocolo de Comunicação

O Protocolo de Comunicação desenvolvido para esta aplicação garante comunicação bidireccional eficiente e robusta, com perdas inferiores a 0,05%. Para isso foram implementados mecanismos que garantem o acesso concorrente ao meio de comunicação, verificação da consistência das mensagens, detecção de perdas e de falhas na comunicação. Não existindo a possibilidade de escutar o meio antes de transmitir (“Carry Sense”) ou de detectar colisões (“Collision Detect”), optou-se por uma estratégia de comunicação síncrona. A Placa de Controlo transmite periodicamente uma mensagem, tipicamente com o estado dos periféricos, que é usada para sincronizar a Placa Rádio. Esta apenas transmite imediatamente após a recepção com sucesso de uma mensagem. A transmissão síncrona inicia-se com o envio da primeira mensagem de dados da Placa de Controlo. As mensagens periódicas enviadas a partir desse momento contêm os últimos dados disponibilizados pela aplicação do utilizador.

A verificação de consistência das mensagens e controlo de erros é efectuado através do cabeçalho que é acrescentado, explicado em detalhe na **Secção 2.9.3**. A detecção de falha nas comunicações é efectuada por tempo. O utilizador define o período de comunicação e o número de períodos consecutivos sem receber uma mensagem, para que se assinale que se perderam as comunicações. Estas configurações são efectuadas no ficheiro *USER_CONFIG.h* e são abordadas na **Secção 2.9.4**. O tamanho das mensagens trocadas entre as placas é fixo e também definível neste ficheiro.

[2] Easy Radio – ER400TRS

O diagrama temporal ilustra o mecanismo de sincronização entre as duas placas:

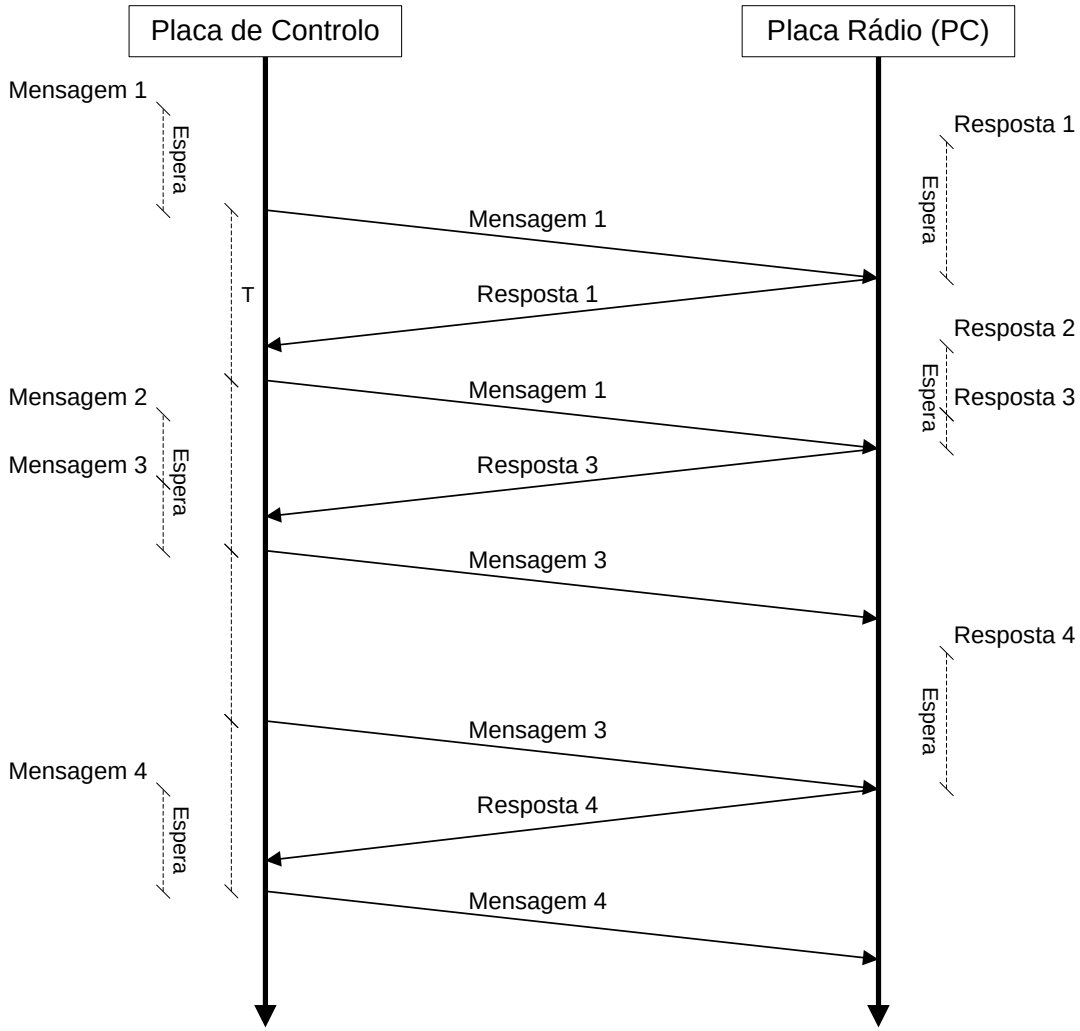


Figura 2 – Diagrama temporal de troca de mensagens síncrona entre placas.

2.9.2 Velocidade de Comunicação

A frequência de troca de mensagens entre as placas é configurável, sendo no entanto limitada por diversos factores, como o tamanho das mensagens a transmitir, o tempo de transmissão no ar e a velocidade de comunicação com os transmissores/receptores rádio das placas.

A figura seguinte representa os tempos relacionados com a comunicação entre a Placa de Controlo e o PC.

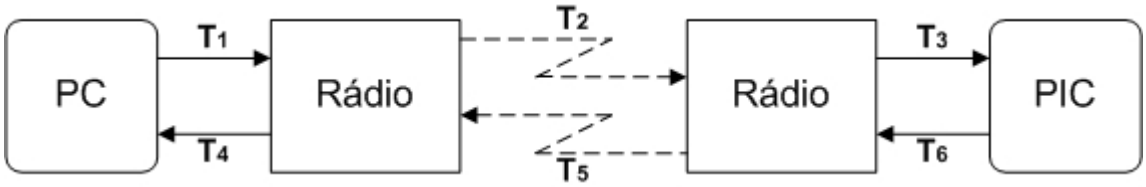


Figura 3 – Tempos associados à comunicação rádio.

A frequência máxima de troca de mensagens entre placas pode então ser calculada pela seguinte expressão:

$$f_{máx} = \frac{1}{1{,}15} \cdot \frac{1}{T_1 + T_2 + T_3 + T_4 + T_5 + T_6}$$

$$f_{máx} = \frac{1}{1{,}15} \cdot \frac{1}{\frac{(Bytes\_in + Bytes\_out + 10) \cdot 10 \cdot (BaudRate + 19200)}{BaudRate \cdot 19200} + \frac{264 + (Bytes\_in + Bytes\_out + 6) \cdot 8}{10000}}$$

Onde:

$Bytes\_in$ - Tamanho das mensagens a receber na Placa de Controlo (em bytes)

$Bytes\_out$ - Tamanho das mensagens a transmitir na Placa de Controlo (em bytes)

$BaudRate$ - Velocidade da ligação série entre o PC e a Placa Rádio

A utilização de uma frequência superior à resultante da expressão acima conduz ao funcionamento incorrecto do protocolo de comunicação.

2.9.3 Estrutura das mensagens

As mensagens rádio que são transmitidas podem ser separadas em duas componentes: cabeçalho e corpo. O corpo das mensagens é constituído pelos dados a transmitir e o seu tamanho é fixo, tendo que ser definido no ficheiro *USER_CONFIG.h*. O cabeçalho das mensagens possui três campos: identificador, número de mensagem e código de Redundância. O identificador é utilizado para localizar o início de cada mensagem. O número de mensagem é usado para controlar as perdas da comunicação. Cada mensagem enviada da Placa Rádio é marcada com um número diferente, que é esperado na próxima mensagem a receber da Placa de Controlo. Desta forma é possível determinar quais das mensagens enviadas são efectivamente recebidas na Placa de Controlo. O código de redundância garante a consistência das mensagens, sendo calculado pela soma de todos os elementos do corpo da mensagem.

A figura a baixo mostra a estrutura de uma mensagem:

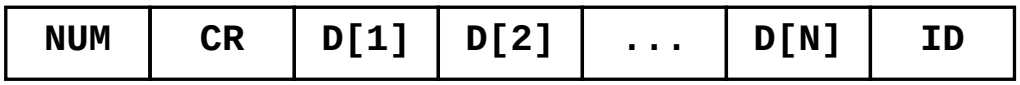


Figura 4 – Estrutura das mensagens rádio.

Onde:

`NUM` – Número da mensagem

`CR` – Código de Redundância

`D[]` – Corpo da mensagem (tamanho `N`)

`ID` – Identificador da mensagem

2.9.4 Configuração

A utilização da comunicação rádio pressupõe a configuração da velocidade de comunicação, do tamanho das mensagens a receber e a enviar e do número de períodos até se assinalar a falha de comunicações. A definição da velocidade de comunicação é feita através da variável *RADIO_CNT_MAX*, que pode variar entre 0 e 255 e se relaciona através da expressão:

$$f_{Comunicação} = \frac{5 \cdot 10^6}{1024 \cdot 127 \cdot RADIO\_CNT\_MAX} \quad [Hz]$$

Onde:

$f_{Comunicação}$ - Frequência de comunicação (em Hertz).

Estas definições são feitas no ficheiro *USER_CONFIG.h*. O exemplo seguinte configura o rádio para comunicar a 12 Hz, com mensagens a receber/enviar com 4 e 5 bytes e perda de comunicação ao fim de 16 períodos.

```
...

// Frequência de envio de mensagens
// Fsend = 5MHz/(1024*127*RADIO_CNT_MAX)
#define RADIO_CNT_MAX     3

...

//
//     Definições da comunicação rádio
//
//
// Formato das mensagens:
// [NUM CR DATA(1) DATA(2) ... DATA(N) ID]
//

// Define o estado do rádio (ON/OFF)
#define RADIO_STATE       ON


// Define o tamanho das mensagens a transmitir...
#define RADIO_MSG_OUT_SIZE 5

// Define o tamanho das mensagens a receber (apenas dados, mín. 1,...
#define RADIO_MSG_IN_SIZE  4

// Define o número de comunicações sem recepção de mensagem até que...
// Ver frequência de comunicação definida para obter o respectivo...
#define RADIO_LOST_MAX    16
```

Exemplo 3 – Configuração da comunicação rádio.

2.9.5 Funções C

As funções disponibilizadas para a comunicação rádio encontram-se nos ficheiros *RADIO.c* e *RADIO.h* e são as seguintes:

```
void init_RADIO(void);
```

Função usada pelo sistema para iniciar a comunicação rádio. Não deve ser chamada pelo utilizador.

```
void radio_send(unsigned char * msg);
```

Função que envia mensagens de dados via rádio. A chamada a esta função inicia o envio periódico de mensagens. O tamanho da mensagem a enviar deve ser concordante com o definido no ficheiro *USER_CONFIG.h*.

```
void radio_send_cmd(unsigned char type, unsigned char value);
```

Função que envia mensagens de comando para o transmissor/receptor rádio e que é usada para ao configurar.

```
void radio_send_ack(void);
```

Função que envia mensagens de ACK ao transmissor/receptor rádio, para confirmar comandos enviados previamente.

```
void radio_send_stop(void);
```

Função que interrompe o envio periódico de mensagens, iniciado por uma chamada à função `radio_send`.

```
unsigned char * radio_recv(void);
```

Função que retorna a última mensagem nova recebida. Caso não tenha sido recebida uma nova mensagem é retornado 0.

```
unsigned char radio_config(unsigned char channel, unsigned char power);
```

Função que configura o transmissor/receptor rádio. Os argumentos desta função são o canal e a potência de transmissão, ambos numerados de 0 a 9. A potência de transmissão varia proporcionalmente com argumento, correspondendo 0 a 1mW e 9 a 10mW. Aconselha-se a não utilização dos canais 1 e 9, devido a um funcionamento incorrecto do dispositivo. A função retorna 1 em caso de sucesso e 0 caso contrário.

```
void radio_wait_send(void);
```

Função que espera até que a mensagem carregada na última chamada à função `radio_send` seja enviada.

```
void radio_wait_recv(void);
```

Função que espera até que seja recebida uma nova mensagem rádio, ou até que seja detectada uma falha de comunicação. É possível verificar se a função retornou devido a uma falha de comunicação chamando a função `radio_status`.

```
unsigned char radio_status(void);
```

Função que retorna o estado da comunicação rádio. Caso a comunicação esteja activa retorna 0, caso tenha falhado retorna 1.

```
void radio_status_reset(void);
```

Função que reinicia o estado da ligação, ou seja, põe a zero o número de períodos a que não é recebida uma mensagem nova e assinala a ligação como activa.

As funções a negrito acedem a recursos do sistema de gestão de periféricos, inibindo para isso a sua execução, sendo designadas por funções não concorrentes. Estas funções não devem ser executadas de forma exaustiva, devendo ser usadas como se encontra descrito no **Exemplo 4**.

```
// Utilização incorrecta, execução exaustiva de funções não concorrentes
while(1)
{
      radio_send(a);      // Envia uma nova mensagem rádio
}


// Utilização correcta de funções não concorrentes
while(1)
{
      wait_PERIOD();      // Espera um determinado período
      radio_send(a);      // Envia uma nova mensagem rádio
}
```

Exemplo 4 – Utilização de funções rádio não concorrentes.

2.9.6 Exemplos

O seguinte exemplo mostra uma aplicação típica da comunicação rádio:

```
#include “IO.h”

void main(void)
{
	// Declaração de variáveis locais
	unsigned char a[2];
	unsigned char * b;

	// Inicia o sistema de gestão de periféricos
	init_IO();

	// Configura a comunicação rádio para o canal 0, potência 1mW
	if( radio_config(0,0) == 0 )
		// Ciclo infinito em caso de erro na configuração
		while(1);

	// Ciclo principal
	while(1)
	{
		// Espera que passe um período
		wait_PERIOD();

		// Actualiza as informações de 2 sonares
		a[0] = get_TEMP0();
		a[1] = get_TEMP1();

		// Envia mensagem com os novos dados
		radio_send(a);

		// Tenta obter uma nova mensagem
		b = radio_recv();

		// Verifica se foi recebida uma nova mensagem
		if( b != 0)
		{
			// Actualiza o valor de 2 PWM
			set_PWM0(b[0]);
			set_PWM1(b[1]);
		}
	}
}
```

Exemplo 5 – Aplicação típica com funções de comunicação rádio.

2.10 Sinais PWM

A Placa de Controlo é capaz gerar até 6 sinais PWM. Estes sinais são destinados à actuação de motores, possuindo 127 níveis distintos de intensidade em cada sentido. Cada sinal PWM é gerado a uma frequência de 38,4 Hz, valor muito superior ao pólo dominante de uma vasta gama de motores eléctricos (ex. motores da LEGO), o que permite uma actuação suave destes. Os sinais gerados são enviados para a Placa de Interface para Módulos de Potência, onde são amplificados e aplicados aos motores. As características eléctricas e físicas dos motores suportados encontram-se descritas na **Secção 5.1**.

2.10.1 Geração dos PWM

A cada sinal PWM correspondem duas linhas, que são actuadas de acordo com a intensidade e sentido desejadas. Cada período do sinal PWM é discretizado em 127 instantes, sendo assim possível actuar os motores com 127 intensidades distintas. A utilização de duas linhas permite rodar o motor em ambos os sentidos, dependendo da linha em que o sinal PWM é aplicado. As figuras seguintes ilustram o processo de geração dos PWM descrito:

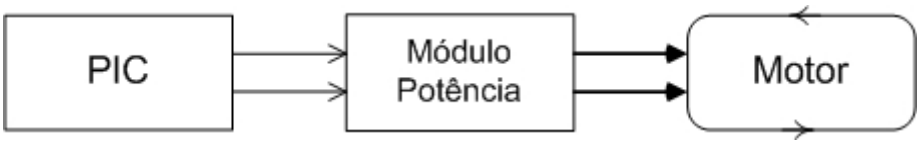


Figura 5 – Esquema conceptual de ligação a motores.

Figura 6 – Geração de sinal PWM, com *Duty-Cycle* a 50% no sentido positivo.

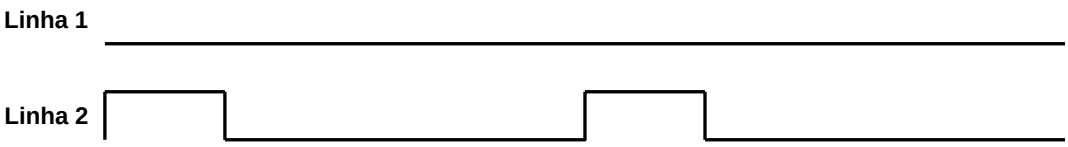
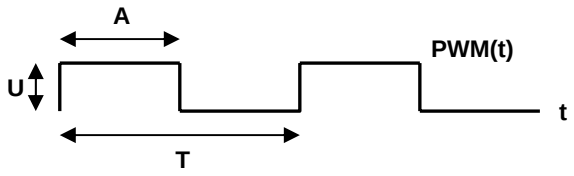


Figura 7 – Geração de sinal PWM, com *Duty-Cycle* a 25% no sentido negativo.

Uma vez que a frequência do sinal PWM é muito superior ao pólo dominante da maioria dos motores eléctricos, estes são sensíveis ao valor eficaz da tensão que lhes é aplicada. A relação entre o *Duty-Cycle* de um sinal PWM e a sua tensão eficaz é dada por:



$$DC = \frac{A}{T} \cdot 100 \text{ , onde } DC - \textit{Duty-Cycle}$$

$$V(DC) = \sqrt{\frac{1}{T}\int_{T} (PWM(t))^2 \, dt} = U\sqrt{\frac{DC}{100}}$$

Figura 8 – Sinal PWM. (U – tensão, T – período)

2.10.2 Configuração

Os PWM são gerados a uma frequência fixa de 38,4 Hz, que não é configurável pelo utilizador. Apenas é preciso definir os pinos e o porto onde cada sinal deve ser criado. Embora diferentes PWM possam estar em portos diferentes, cada PWM só pode estar localizado num porto. De seguida exemplifica-se como especificar, no ficheiro *USER_CONFIG.h*, a utilização do PWM 0 e 1, um no porto A, pinos 0 e 1, outro no porto C, pinos 4 e 7. Como se pode visualizar, a especificação do porto traduz-se pela identificação do LAT e TRIS desejado (consulte **Secção 4.2.3** para ver o mapeamento dos pinos).

```
//
//     Definição dos PWM
//

// Define o estado dos PWM (ON/OFF)
#define PWM0_STATE  ON
#define PWM1_STATE  ON
#define PWM2_STATE  OFF
#define PWM3_STATE  OFF
#define PWM4_STATE  OFF
#define PWM5_STATE  OFF

// Configura o PWM0
#if PWM0_STATE
#define PWM0_TRIS   TRISA
#define PWM0_LAT    LATA
#define PWM0_PIN0   0
#define PWM0_PIN1   1
#endif

// Configura o PWM1
#if PWM1_STATE
#define PWM1_TRIS   TRISC
#define PWM1_LAT    LATC
#define PWM1_PIN0   4
#define PWM1_PIN1   7
#endif
```

Exemplo 6 – Especificação do porto e pinos para os PWM 0 e 1.

2.10.3 Funções C

As funções desenvolvidas para trabalhar com os PWM encontram-se nos ficheiros *PWM.c* e *PWM.h*. De seguida efectua-se uma breve descrição destas funções:

```
void init_PWM(void);
```

Função utilizada pelo sistema de gestão de periféricos, para proceder às inicializações necessárias à geração dos sinais PWM especificados no ficheiro *USER_CONFIG.h*. Esta função não deve ser chamada pelo utilizador.

```
void set_PWMX(char dc);
```

Função que actualiza a intensidade e sentido do sinal PWM X. X é um número de 0 a 5 que identifica um sinal PWM. O sentido do PWM é definido pelo sinal do argumento dc, enquanto que a intensidade é dada pelo seu módulo. Uma vez que dc é do tipo char (1 byte), o seu módulo varia entre 0 e 127.

```
char get_PWMX(void);
```

Função que retorna o sentido e intensidade do PWM X. X é um número de 0 a 5 que identifica o sinal PWM.

As funções a negrito acedem a recursos do sistema de gestão de periféricos, inibindo para isso a sua execução, sendo designadas por funções não concorrentes. Estas funções não devem ser executadas de forma exaustiva, devendo ser usadas como se encontra descrito no **Exemplo 7**.

```
// Utilização incorrecta, execução exaustiva de funções não concorrentes
while(1)
{
      set_PWM0(-127);     // Actualiza a intensidade e sentido do PWM0
}


// Utilização correcta de funções não concorrentes
while(1)
{
      wait_PERIOD();      // Espera um determinado período
      set_PWM0(-127);     // Actualiza a intensidade e sentido do PWM0
}
```

Exemplo 7 – Utilização de funções PWM não concorrentes.

2.10.4 Exemplos

O seguinte exemplo mostra uma utilização típica de funções relativas a sinais PWM:

```
#include “IO.h”

void main(void)
{
	// Declaração de variáveis locais
	char a, b;

	// Inicia o sistema de gestão de periféricos
	init_IO();


	// Inicia o PWM0 com um offset
	set_PWM0(128);


	// Ciclo principal
	while(1)
	{
		// Espera que passe um período
		wait_PERIOD();

		// Obtém o valor dos PWM
		a = get_PWM0();
		b = get_PWM1();

		// Incrementa o valor dos PWM
		a++;
		b++;

		// Incrementa os valores dos PWM
		set_PWM0(a);
		set_PWM1(b);
	}
}
```

Exemplo 8 – Utilização típica de funções relativas a sinais PWM.

2.11 Sensores de Rotação de Alta Resolução

Os Sensores de Rotação de Alta Resolução permitem saber com grande precisão a posição angular de uma junta. Estes sensores efectuam 2048 contagens por revolução, permitindo uma resolução de 360/2048 graus. Esta elevada resolução é no entanto contrabalançada por uma limitação na velocidade máxima de rotação permitida. Este aspecto é discutido mais profundamente na **Secção 2.11.2**. O funcionamento e ligação a esta classe de sensores é abordado em detalhe na **Secção 5.4** deste manual. A utilização de estes sensores a juntas actuadas por motores, possibilita o seu controlo com uma elevada precisão, proporcional à resolução dos sensores.

2.11.1 Modo de leitura

A leitura dos Sensores de Rotação de Alta Resolução é feita através de duas linhas. Como descrito na **Secção 5.4**, o disco de codificação possui 512 tiras. Este é analisado por dois sensores de luz paralelos, que retornam níveis lógicos 0 e 1, de acordo com a presença ou não de uma risca do disco. Este mecanismo gera quatro combinações de níveis lógicos por tira, obtendo-se 2048 comutações de níveis lógicos por revolução. A figura seguinte ilustra o mecanismo descrito:

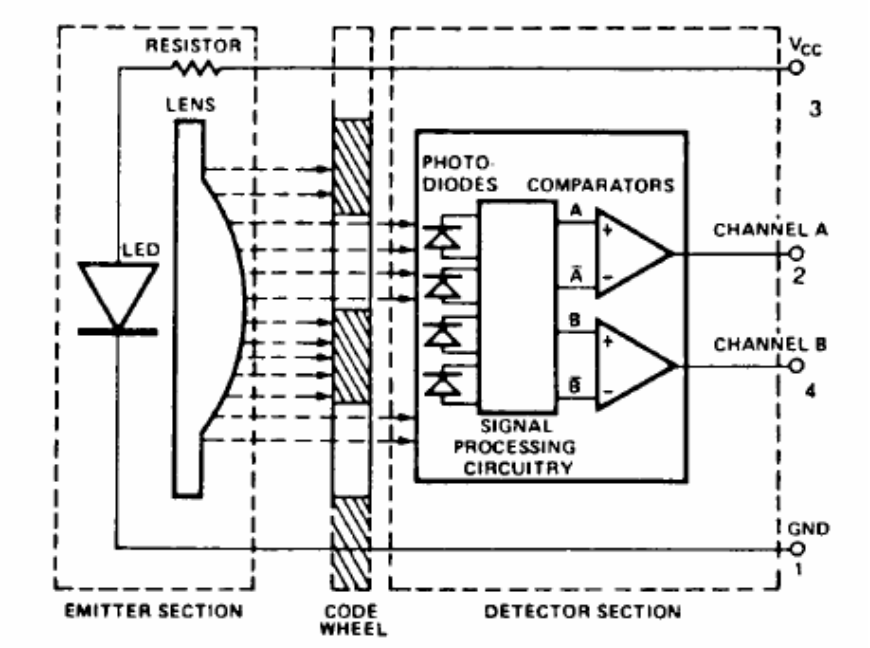


Figura 9 – Esquema do Sensor de Rotação de Alta Resolução (imagem do manual do equipamento).

A leitura destes sensores é baseada na leitura do canal A e B. Estes alternam ao longo do tempo, de acordo com a figura seguinte:

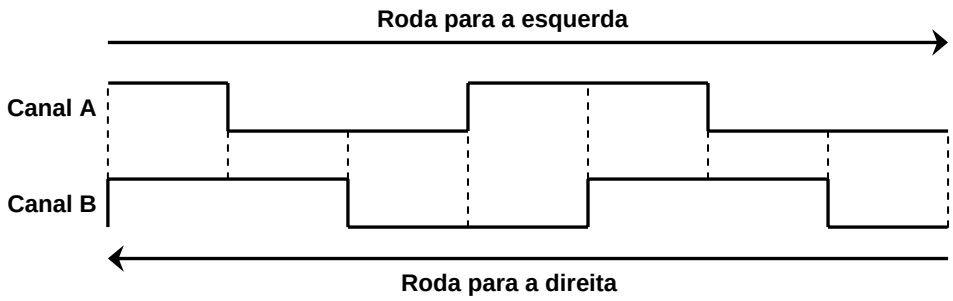


Figura 10 – Forma de onda dos Sensores de Rotação de Alta Resolução.

Como é possível observar na figura acima, sabendo o estado anterior e actual de um sensor, é possível determinar para que lado é que a junta rodou. Caso seja perdido um estado intermédio, já não é possível saber o sentido da rotação, sendo no entanto possível detectar a ocorrência. O respeito da velocidade máxima de rotação, permitida pela frequência de leitura do sensor, é assim essencial para o seu correcto funcionamento.

2.11.2 Frequência de leitura

A frequência de leitura especifica o número de vezes por segundo que é obtido o estado do sensor. Como já foi visto, é necessário detectar todos os estados por que o sensor passa para que se consiga determinar a sua posição angular com precisão. A frequência de leitura é igual para todos os Sensores de Rotação de Alta Resolução, pelo que deve ser projectada com base na velocidade de rotação mais elevada. A sua configuração é feita no ficheiro `USER_CONFIG.h`. A relação entre a frequência de leitura e a velocidade de rotação máxima permitida é dada pela expressão:

$$\omega_{máx} = \frac{f_{leitura}}{2048} \cdot 60 \qquad [rpm]$$

Onde:

$\omega_{máx}$ - Velocidade de rotação máxima (rotações por minuto)

$f_{leitura}$ - Frequência de leitura do sensor (Hz)

2.11.3 Configuração

A utilização dos Sensores de Rotação de Alta Resolução obriga à especificação, no ficheiro *USER_CONFIG.h*, da frequência de leitura e dos pinos a que os sensores estão ligados. O dimensionamento da frequência de leitura é abordado na **Secção 2.11.2**. A sua definição é feita através da variável *RS_CNT_MAX*, que pode variar entre 0 e 255 e se relaciona com a frequência de leitura segundo a expressão:

$$f_{leitura} = \frac{5 \cdot 10^6}{1024 \cdot RS\_CNT\_MAX} \quad [Hz]$$

Os sensores podem ser ligados a portos diferentes, embora os pinos de cada sensor tenham que estar no mesmo porto. Consulte a **Secção 4.2.3** para ver o mapeamento dos portos e pinos na Placa de Controlo. O porto é especificado através do PORT e TRIS. O exemplo seguinte configura a leitura de dois sensores, o primeiro no porto A, pinos 0 e 1 e o segundo no porto E, pinos 2 e 4. A frequência de leitura é de 542,5 Hz, possibilitando uma velocidade de rotação máxima de 15,9 rpm.

```
...

// Frequência de amostragem dos Sensores de Rotação...
// Velocidade máxima de rotação, Vmáx = F_RS/2048*60 [rpm]
#define RS_CNT_MAX        9
...

//
//     Definição de Sensores de Rotação de Alta Resolução
//

// Define o estado dos Sensores de Rotação (ON/OFF)
#define RS0_STATE   ON
#define RS1_STATE   ON
#define RS2_STATE   OFF
#define RS3_STATE   OFF
#define RS4_STATE   OFF
#define RS5_STATE   OFF

// Configura o Sensor de Rotação 0
#if RS0_STATE
#define RS0_PORT    PORTA
#define RS0_TRIS    TRISA
#define RS0_PINA    0
#define RS0_PINB    1
#endif

// Configura o Sensor de Rotação 1
#if RS1_STATE
#define RS1_PORT    PORTE
#define RS1_TRIS    TRISE
#define RS1_PINA    2
#define RS1_PINB    4
#endif
```

Exemplo 9 – Configuração dos Sensores de Rotação de Alta Resolução 0 e 1.

2.11.4 Funções C

As funções de interface com os Sensores de Rotação de Alta Resolução estão nos ficheiros *RS.h* e *RS.c*. As suas características são descritas de seguida:

```
void init_RS(void);
```

Função chamada pelo sistema de gestão de periféricos que procede às inicializações necessárias para ler os sensores desta classe, declarados no ficheiro *USER_CONFIG.h*. Esta função não deve ser utilizada.

```
int get_RSX(void);
```

Função que obtém o número de contagens relativas ao sensor X. X é um número de 0 a 5 que identifica o sensor desta classe a ler. Cada sensor possui um contador de 16 bits, permitindo valores entre -32768 e 32767. Caso a contagem ultrapasse 32767 salta para -32768 e vice-versa. A conversão de contagens para ângulo é feita através da expressão:

$$\theta = \frac{Contagem}{2048} \cdot 360 \quad [^\circ]$$

```
int reset_RSX(void);
```

Função que obtém e limpa o número de contagens relativas ao sensor X. X é um número de 0 a 5 que identifica o sensor desta classe a ler. Cada sensor possui um contador de 16 bits, permitindo valores entre -32768 e 32767. Caso a contagem ultrapasse 32767 salta para -32768 e vice-versa. A conversão de contagens para ângulo é feita através da expressão:

$$\theta = \frac{Contagem}{2048} \cdot 360 \quad [^\circ]$$

As funções a negrito acedem a recursos do sistema de gestão de periféricos, inibindo para isso a sua execução, sendo designadas por funções não concorrentes. Estas funções não devem ser executadas de forma exaustiva, devendo ser usadas como se encontra descrito no **Exemplo 10**.

```
// Utilização incorrecta, execução exaustiva de funções não concorrentes
while(1)
{
	a = get_RS0();	// Obtém o valor do sensor 0
}


// Utilização correcta de funções não concorrentes
while(1)
{
	wait_PERIOD();	// Espera um determinado período
	a = get_RS0();	// Obtém o valor do sensor 0

}
```

Exemplo 10 – Utilização de funções dos Sensores de Rotação de Alta Resolução não concorrentes.

2.11.5 Exemplos

O seguinte exemplo consiste numa aplicação típica de Sensores de Rotação de Alta Resolução.

```
#include “IO.h”

void main(void)
{
	// Declaração de variáveis locais
	int a, b;

	// Inicia o sistema de gestão de periféricos
	init_IO();


	// Ciclo principal
	while(1)
	{
		// Espera que passe um período
		wait_PERIOD();

		// Obtém o valor dos RS 0 e 1
		a = get_RS0();
		b = get_RS1();

		// Actua o motor 0 de acordo com o sinal do RS 0
		if( a > 0 )
			set_PWM0(-127);
		else
			set_PWM0(127);

		// Actua o motor 1 de acordo com o sinal do RS 1
		if( b > 0 )
			set_PWM1(-127);
		else
			set_PWM1(127);
	}
}
```

Exemplo 11 – Aplicação típica dos Sensores de Rotação de Alta Resolução.

2.12 Sensores Activos

A classe dos Sensores Activos engloba Sensores de Rotação da LEGO e Sensores de Proximidade. Estes possuem duas características em comum, que tornam o procedimento de leitura semelhante. Em ambos os casos o valor medido é fornecido após uma actuação sobre o sensor, que retorna uma tensão contínua. Esta é posteriormente convertida por um A/D, durante um período de tempo comum a ambos os sensores. Uma vez que o PIC da Placa de Controlo apenas possui um módulo A/D, este é multiplexado por todos os Sensores Activos. No máximo é possível usar até 10 Sensores Activos de ambos os tipos, estando o número de Sensores de Proximidade limitado a 4. A leitura destes Sensores na Placa de Controlo é efectuada por apenas 10 pinos, que devem ser ocupados por uma ordem específica. Este procedimento é descrito em detalhe na **Secção 2.12.4** deste manual. Embora esta classe possua dois tipos distintos de sensores, as funções de interacção são as mesmas, funcionando de acordo com o sensor a que são aplicadas. Estas características são abordadas mais profundamente na Secção **2.12.5**.

2.12.1 Tipos de Sensores Activos

A classe dos Sensores Activos contém dois tipos de sensores: Proximidade e Rotação da LEGO. Embora o procedimento de leitura seja muito semelhante para ambos os casos, as suas aplicações são distintas.

Os Sensores de Rotação da LEGO medem a posição angular de juntas de revolução. Estes sensores possuem uma resolução de 16 contagens por volta, muito inferior à dos Sensores de Rotação de Alta Resolução, descritos na **Secção 2.11.** Esta característica permite no entanto que operem a velocidades de rotação mais elevadas, com um custo computacional inferior (ver **Secção 2.6**). É de notar que os sensores de rotação da LEGO são alimentados e lidos através dos mesmos terminais, possuindo por isso apenas duas ligações.

Os Sensores de Proximidade retornam uma tensão proporcional à distância até ao objecto mais próximo. Embora seja possível operar dos 2 aos 400 mm, a conversão A/D apenas obtém 102 níveis de quantização. Este facto não é crítico, uma vez que estes sensores devem ser usados para detectar se existe um obstáculo próximo e não para determinar a distância até ele. O componente principal da placa destes sensores é o Agilent HSDL-9100 Miniature Surface-Mount Proximity Sensor (ver **Secção 4.6**).

2.12.2 Modos de leitura

A operação com os Sensores de Rotação LEGO é constituída por duas fases, uma de alimentação e outra de leitura. Na fase de alimentação o sensor armazena energia, não sendo possível ler o seu valor. Na fase de leitura é retirada a alimentação. A tensão na linha passa a ser mantida pelo sensor, possibilitando a leitura do seu estado. Esta fase tem uma duração máxima de 100 µs, sendo a conversão A/D iniciada após um período de estabilização do sinal. Os Sensores de Rotação da LEGO impõem quatro níveis de tensão, que correspondem a quatro estados distintos.

As figuras seguintes ilustram os tempos envolvidos na aquisição dos estados do sensor e a sua evolução com o movimento de rotação.

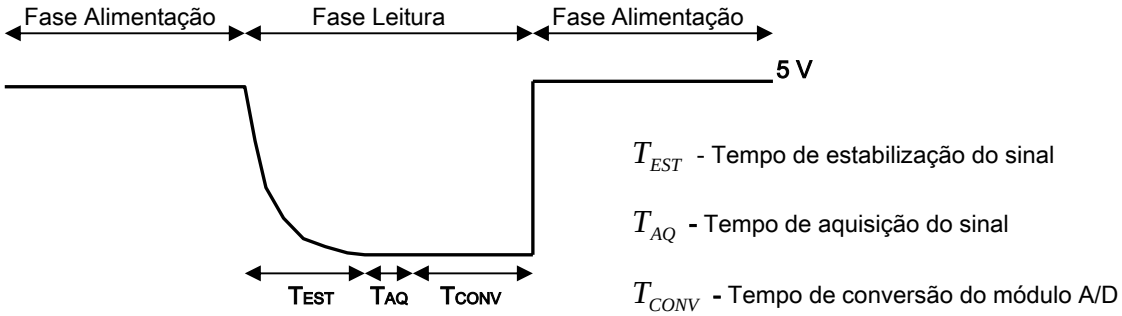


Figura 11 – Gráfico temporal da leitura dos Sensores de Rotação da LEGO.

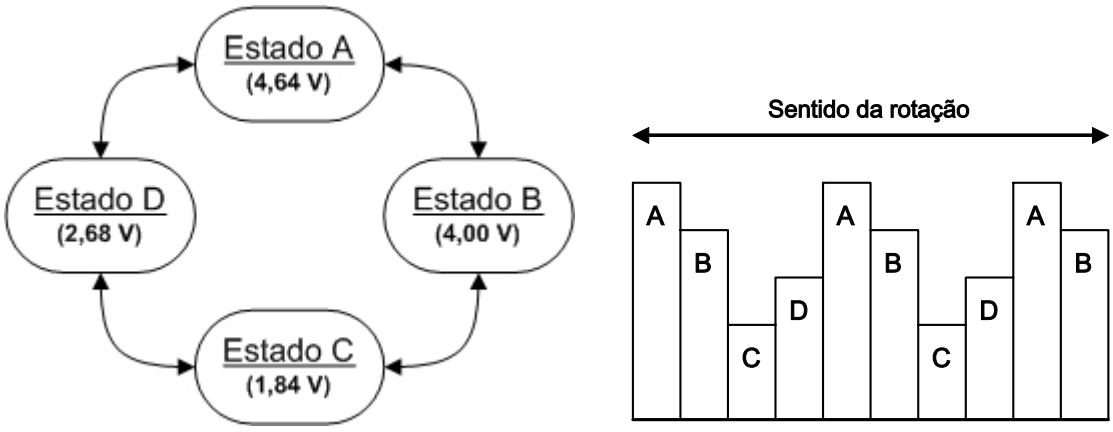


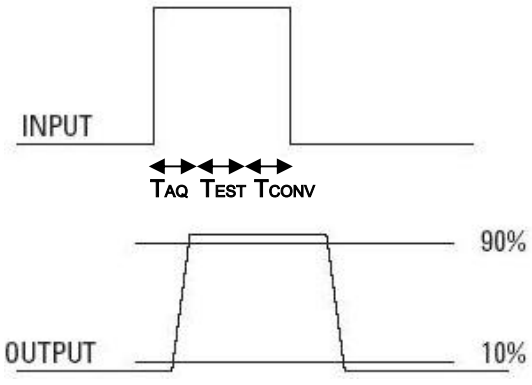
Figura 12 – Máquina de estados dos Sensores de Rotação da LEGO e respectiva evolução temporal.

Experimentalmente verificou-se que após a remoção da alimentação, a tensão imposta pelo sensor demora aproximadamente 12 µs a estabilizar. De forma a garantir uma boa margem de segurança, escolheu-se um T_{EST} de 25,6 µs. O tempo de aquisição de sinal é de 3,2 µs e o de conversão A/D de 9,6

µs. Estes tempos são constantes, resultando um tempo total por sensor de 38,4 µs, valor usado para calcular a frequência máxima de leitura dos Sensores Activos **(Secção 2.12.3)**.

Como é possível observar na figura acima, sabendo o estado anterior e actual de um sensor, é possível determinar para que lado é que a junta rodou. Caso seja perdido um estado intermédio, já não é possível saber o sentido da rotação, sendo no entanto possível detectar a ocorrência. O respeito da velocidade máxima de rotação, permitida pela frequência de leitura do sensor, é assim essencial para o seu correcto funcionamento.

A leitura dos Sensores de Proximidade baseia-se na conversão de uma tensão contínua, proporcional à distância ao objecto mais próximo. Estes sensores são constituídos por um par emissor/receptor de luz. Inicialmente é enviado um pulso que acende o LED (emissor de luz). A luz emitida é reflectida pelo objecto mais próximo, sendo depois captada pelo fotodíodo (receptor de luz). A intensidade da luz recebida é proporcional à distância ao objecto mais próximo, que por sua vez vai ser proporcional à tensão imposta pelo fotodíodo. A conversão desta tensão é iniciada após um período de espera que permite estabilizar a tensão aos terminais do sensor. As figuras seguintes ilustram os tempos de leitura do sensor e sua a característica:



T_{EST} - Tempo de estabilização do sinal

T_{AQ} - Tempo de aquisição do sinal

T_{CONV} - Tempo de conversão do módulo A/D

Figura 13 – Processo de leitura dos sensores de proximidade (imagem do manual do equipamento).

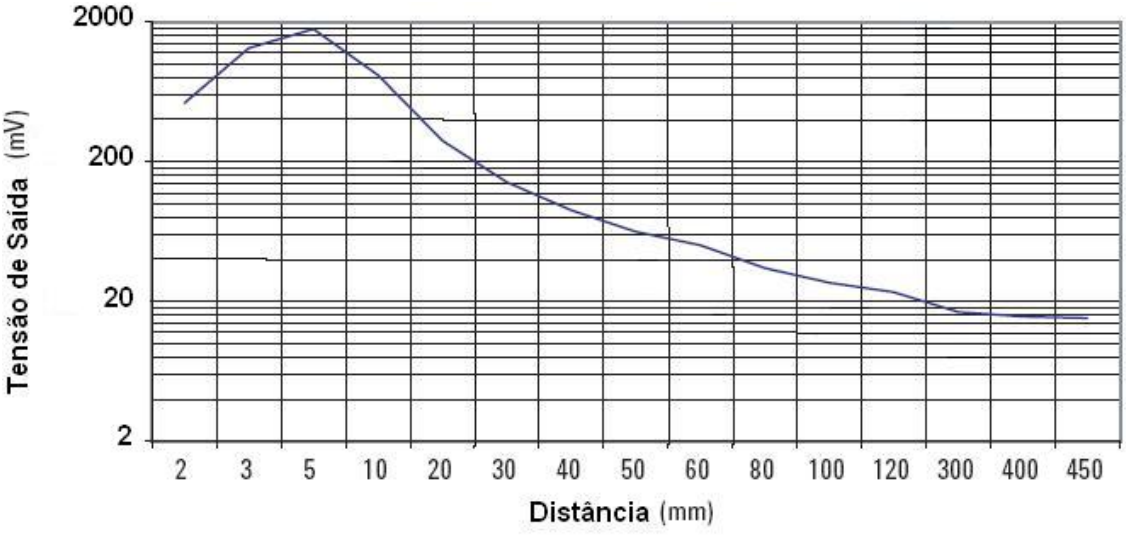


Figura 14 – Relação entre a tensão de saída e a distância (imagem do manual do equipamento).

Os tempos envolvidos na leitura dos Sensores de Proximidade são, como já referido, semelhantes aos dos Sensores de Rotação LEGO. O tempo de estabilização é de 51,2 μs, o de aquisição de 3,2 μs e o de conversão do A/D de 9,6 μs, resultando um tempo total de 64,0 μs. Neste caso o tempo de estabilização é inferior a 50us, sendo ajustável através do valor de uma resistência da placa do sensor. A margem de segurança é assim reduzida, o que não é crítico visto que a informação a obter destes sensores é qualitativa e não quantitativa.

Como se pode verificar a tensão máxima de saída do sensor é 2V. Este facto é responsável pela reduzida resolução de leitura destes sensores, uma vez que a tensão de referência do A/D são os 5V. A gama de valores de saída fica assim limitada entre 0 e 102 (numa escala de 0 a 255). A detecção de objectos próximos deve ser feita com base num valor de diferenciação, escolhido de acordo com a aplicação. Valores retornados pelo sensor acima deste limite indicam a existência de um obstáculo.

2.12.3 Frequência de leitura

A frequência de leitura especifica o número de vezes por segundo que é obtido o valor de um sensor. Como já foi visto, é necessário detectar todos os estados dos Sensores de Rotação da LEGO para que se consiga determinar a sua posição angular com precisão. A frequência de leitura é igual para todos os Sensores Activos, pelo que deve ser projectada com base na velocidade de rotação máxima prevista para os Sensores de Rotação da LEGO. A sua configuração é feita no ficheiro `USER_CONFIG.h`. A relação entre a frequência de leitura e a velocidade de rotação máxima permitida é dada pela expressão:

$$\omega_{máx} = \frac{f_{leitura}}{16} \cdot 60 \qquad [rpm]$$

Onde:

$\omega_{máx}$ - Velocidade de rotação máxima (rotações por minuto)

$f_{leitura}$ - Frequência de leitura do sensor (Hz)

É de notar que devido ao tempo de leitura de cada Sensor Activo, a frequência de leitura é limitada, sendo o seu máximo dado por:

$$f_{MáxLeitura} = \frac{1}{38{,}4 \cdot 10^{-6} \cdot N\_ROT + 64 \cdot 10^{-6} \cdot N\_PROX} \qquad [Hz]$$

Onde:

N_ROT - Número de Sensores de Rotação da LEGO.

N_PROX - Número de Sensores de Proximidade.

$f_{MáxLeitura}$ - Frequência máxima de leitura dos Sensores Activos.

2.12.4 Configuração

A utilização dos Sensores Activos obriga à especificação, no ficheiro *USER_CONFIG.h*, da frequência de leitura e do tipo de sensor em utilização. No caso dos Sensores de proximidade é também necessário indicar o porto e pino por onde é enviado o pulso de activação. O dimensionamento da frequência de leitura é abordado na **Secção 2.12.3**. A sua definição é feita através da variável *AS_CNT_MAX*, que pode variar entre 0 a 255 e se relaciona com a frequência de leitura segundo a expressão:

$$f_{leitura} = \frac{5 \cdot 10^6}{1024 \cdot AS\_CNT\_MAX} \quad [Hz]$$

Os Sensores Activos apenas podem ser ligados a 10 pinos, segundo uma ordem específica, razão pela qual não é necessário indicar a localização dos pinos de leitura. A tabela seguinte indica a ordem por que os pinos de leitura dos Sensores Activos têm que ser ligados:

AS 0	AS 1	AS 2	AS 3	AS 4	AS 5	AS 6	AS 7	AS 8	AS 9
RA0	RA1	RA2	RA3	RA5	RE0	RE1	RE2	RB2	RB3

Tabela 1 – Ordem de ligação Sensores Activos à Placa de Controlo.

Na tabela acima, AS X significa Sensor Activo número X, enquanto que RXY identifica o porto X (A ou E) e pino Y (de 0 a 7). Para localizar correctamente estes pinos e os de envio de pulsos para os Sensores de Proximidade, consulte a **Secção 4.2.3**, onde se encontra o mapeamento da Placa de Controlo. No ficheiro *USER_CONFIG.h*, a selecção do porto de envio de pulsos é feita através da configuração do LAT e TRIS. A distinção entre os dois tipos de Sensores Activos é feita na sua activação, sendo os de Rotação da LEGO activados com *ON_ROTATION* , enquanto que os de Proximidade são por *ON_LIGHT*.

O exemplo seguinte configura dois Sensores Activos, o primeiro de Rotação da LEGO e o segundo de Proximidade. A frequência de leitura é de 97,7 Hz, permitindo uma velocidade máxima de rotação de 366 rpm. O pulso do Sensor de Proximidade é enviado através do pino 2 do porto E.

```
...

// Frequência de amostragem dos Sensores Activos, F_AS = ...
// Velocidade máxima de rotação, Vmax = F_AS/16*60 [rpm]
#define AS_CNT_MAX        50

//
//     Definição dos Sensores Activos (Rotação da LEGO ou Proximidade)
//

// Os Sensores Activos têm que ser activados por ordem indicada abaixo.
// Para os Sensores Activos tipo do Luz é necessário definir o pino ...
// Máximo 4 Sensores Activos de Proximidade.
//
// Tabela de atribuição de pinos:
//     AS0    AS1    AS2    AS3    AS4    AS5    AS6    AS7    AS8    AS9
//     RA0    RA1    RA2    RA3    RA5    RE0    RE1    RE2    RB2    RB3

// Define o estado dos Sensores Activos (OFF/ON_ROTATION/ON_LIGHT)
#define AS0_STATE   ON_ROTATION
#define AS1_STATE   ON_LIGHT
#define AS2_STATE   OFF
#define AS3_STATE   OFF
#define AS4_STATE   OFF
#define AS5_STATE   OFF
#define AS6_STATE   OFF
#define AS7_STATE   OFF
#define AS8_STATE   OFF
#define AS9_STATE   OFF


// Define o pino de pulso do Sensor Activo 0 do tipo Proximidade
#if AS0_STATE == ON_LIGHT
#define AS0_TRISOUT 0
#define AS0_LATOUT  0
#define AS0_PINOUT  0
#endif

// Define o pino de pulso do Sensor Activo 1 do tipo Proximidade
#if AS1_STATE == ON_LIGHT
#define AS1_TRISOUT TRISE
#define AS1_LATOUT  LATE
#define AS1_PINOUT  2
#endif
```

Exemplo 12 – Configuração dos Sensores Activos 0 e 1.

2.12.5 Funções C

As funções de interface com os Sensores Activos estão nos ficheiros *AS.h* e *AS.c*. As suas características dependem do tipo de Sensor Activo. De seguida é descrito o seu funcionamento para ambos os sensores:

```
void init_AS(void);
```

Função chamada pelo sistema de gestão de periféricos que procede às inicializações necessárias para ler os sensores desta classe, declarados no ficheiro *USER_CONFIG.h*. Esta função não deve ser utilizada.

```
int get_ASX(void);
```
(Sensor de Rotação da LEGO)

Função que obtém o número de contagens relativas ao sensor X. X é um número de 0 a 9 que identifica o Sensor de Rotação da LEGO a ler. Cada sensor possui um contador de 16 bits, permitindo valores entre -32768 e 32767. Caso a contagem ultrapasse 32767 salta para -32768 e vice-versa. A conversão de contagens para ângulo é feita através da expressão:

$$\theta = \frac{Contagem}{16} \cdot 360 \quad [^\circ]$$

```
unsigned char get_ASX(void);
```
(Sensor de Proximidade)

Função que obtém o valor do sensor X. X é um número de 0 a 9 que identifica o Sensor de Proximidade a ler. Cada sensor possui um registo de 8 bits, permitindo valores entre 0 e 255. Neste tipo de sensores, os valores retornados apenas variam entre 0 e 51 (ver **Secção 2.12.2**). A conversão para tensão deve ser feita através da equação abaixo. A relação entre a tensão e a distância não é linear, devendo ser consultada a **Figura 14**.

$$U = \frac{Valor}{255} \cdot 5 \quad [V]$$

`int` **`reset_ASX`**`(void);` (Sensor de Rotação da LEGO)

Função que obtém e limpa o número de contagens relativas ao sensor X. X é um número de 0 a 9 que identifica o Sensor de Rotação da LEGO a ler. Cada sensor possui um contador de 16 bits, permitindo valores entre -32768 e 32767. Caso a contagem ultrapasse 32767 salta para -32768 e vice-versa. A conversão de contagens para ângulo é feita através da expressão:

$$\theta = \frac{Contagem}{16} \cdot 360 \quad [^\circ]$$

As funções a negrito acedem a recursos do sistema de gestão de periféricos, inibindo para isso a sua execução, sendo designadas por funções não concorrentes. Estas funções não devem ser executadas de forma exaustiva, devendo ser usadas como se encontra descrito no **Exemplo 13**.

```
// Utilização incorrecta, execução exaustiva de funções não concorrentes
while(1)
{
      a = get_AS0();      // Obtém o valor do sensor 0
}


// Utilização correcta de funções não concorrentes
while(1)
{
      wait_PERIOD();      // Espera um determinado período
      a = get_AS0();      // Obtém o valor do sensor 0

}
```

Exemplo 13 – Utilização de funções dos Sensores Activos não concorrentes.

2.12.6 Exemplos

O seguinte exemplo consiste numa aplicação típica de Sensores Activos do tipo Rotação da LEGO.

```
#include “IO.h”

void main(void)
{
	// Declaração de variáveis locais
	int a, b;

	// Inicia o sistema de gestão de periféricos
	init_IO();


	// Ciclo principal
	while(1)
	{
		// Espera que passe um período
		wait_PERIOD();

		// Obtém o valor dos AS 0 e 1 (ambos de rotação)
		a = get_AS0();
		b = get_AS1();

		// Actua o motor 0 de acordo com o sinal do AS 0
		if( a > 0 )
			set_PWM0(-127);
		else
			set_PWM0(127);

		// Actua o motor 1 de acordo com o sinal do AS 1
		if( b > 0 )
			set_PWM1(-127);
		else
			set_PWM1(127);
	}
}
```

Exemplo 14 – Aplicação típica dos Sensores Activos.

2.13 Sensores Temporais

A classe de Sensores Temporais engloba Sonares e Acelerómetros. Em ambos os casos, os valores mesurados são proporcionais à duração temporal de pulsos gerados por estes sensores, o que torna a sua interface semelhante. Os Sonares suportados são os SRF04 e SRF05, sendo no entanto possível usar equipamento com características semelhantes. Para informações mais detalhadas sobre o modo de operação consulte a Secção 2.13.2. Relativamente ao acelerómetro, o software foi projectado para o ADXL202 da Analog Devices, que deve ser utilizado numa montagem típica, como a da placa descrita na Secção 4.5. É possível usar até 7 Sensores Temporais, estando o número de eixos de acelerómetros limitado a 3. É de referir que cada acelerómetro, do tipo referido acima, mede a aceleração segundo 2 eixos. A ligação destes Sensores à Placa de Controlo é efectuada em 7 pinos específicos, que não necessitam de ser ocupados por uma ordem específica. A configuração de Sensores Temporais é descrita em detalhe na Secção 2.13.4. Embora esta classe possua dois tipos distintos de sensores, as funções de interacção são as mesmas, funcionando de acordo com o sensor a que são aplicadas. Estas características são abordadas mais profundamente na Secção 2.13.5.

2.13.1 Tipos de Sensores Temporais

A classe dos Sensores Temporais contém dois tipos de sensores: Sonares e Acelerómetros. Embora o procedimento de leitura seja muito semelhante para ambos os casos, as suas aplicações são distintas.

Os Sonares permitem determinar a distância a obstáculos que se encontrem dos 2 cm aos 3m, com uma resolução de 1,766 cm. Estes sensores emitem ultra sons para o meio e geram uma onda quadrada com duração igual ao tempo que os ecos demoram a ser recebidos. Este tempo é proporcional à distância até ao objecto mais próximo, permitindo a sua determinação.

Os acelerómetros de 2 eixos, como o ADXL202, medem a aceleração segundo dois eixos ortogonais. Em diversas aplicações é possível medir inclinações usando estes dispositivos. Esta funcionalidade é descrita em pormenor na Secção 2.13.2. A aceleração sentida em cada um dos eixos é adquirida periodicamente, não sendo necessário qualquer tipo de activação. Como já foi referido, o valor medido corresponde à duração da onda quadrada retornada.

2.13.2 Modos de leitura

A operação com Sonares é realizada através de dois canais, um de pulso e outro de eco. O primeiro é utilizado para enviar um sinal de activação, com duração 12,8 µs. O segundo é monitorizado de forma a contabilizar a duração da resposta do dispositivo. Esta é inferior a 18 ms, sendo por isso usado um contador de 8 bits com período de 102,4 µs, obtendo-se uma capacidade máxima de 26,1 ms. A relação entre a distância e a duração da resposta é dada por:

$$d = \frac{t}{58}$$

Onde *d* é a distância em centímetros e *t* a duração da resposta em micro segundos. É assim possível registar distâncias até aos 4,5 metros, valor superior ao alcance dos Sonares (aproximadamente 3,1 m). Como já foi referido, a leitura de cada sensor é guardada num registo de 8 bits, obtendo-se uma gama de valores compreendida entre 0 e 255. A relação entre o valor do registo e a distância até ao objecto mais próximo é dada por:

$$d = Valor \cdot 1{,}766 \quad [cm]$$

A figura seguinte ilustra o funcionamento dos Sonares:

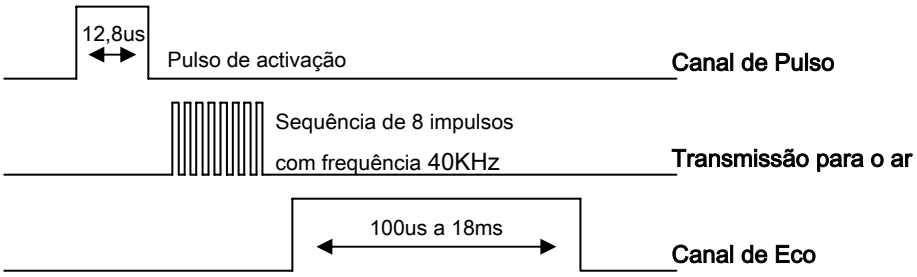


Figura 15 – Diagrama temporal de funcionamento dos Sonares.

É de notar que os Sensores Temporais são lidos sequencialmente, pelo que Sonares que disparem na mesma direcção não devem ser ligados juntos, uma vez que interferem entre si. O exemplo da **Secção 2.13.4** resolve este problema.

A leitura da aceleração registada em cada um dos eixos do acelerómetro não requer o envio de qualquer sinal de activação. Existe uma ligação à Placa de Controlo para cada um dos eixos, que retorna leituras com uma frequência de 135Hz. O Duty-Cycle das ondas geradas em cada linha é proporcional à aceleração sentida. A figura seguinte explicita essa relação:

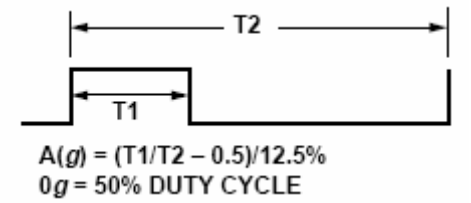


Figura 16 – Relação entre a aceleração e o Duty-Cycle (imagem do manual do equipamento).

A determinação do Duty-Cycle das ondas é feita através do T_1, uma vez que T_2 é fixo. Este é contabilizado usando um contador de 8 bits com período 25,6 µs, obtendo-se uma contagem máxima de 6,5536 ms. Embora este valor seja inferior ao período das ondas gerada pelo dispositivo, permite a leitura das acelerações a que o acelerómetro é sensível, que variam entre -2g e 2g, ou seja, entre 1,85 ms e 5,55 ms. A resolução de leitura dos acelerómetros é de 27,6757 mg, sendo a relação entre o número de contagens e a aceleração dada por:

$$A = 27{,}6757 \cdot 10^{-3} \cdot (CNT - CALIB) \quad [g]$$

Onde *CNT* é o número de contagens (de 0 a 255) e *CALIB* é o valor de calibração do sensor (ver na página seguinte). Estes dispositivos podem também ser usados para medir inclinações. A figura seguinte ilustra como é possível relacionar a aceleração num eixo com a inclinação segundo essa direcção.

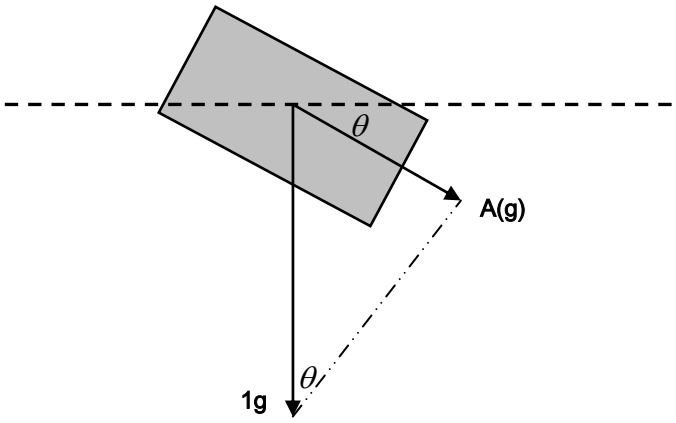


Figura 17 – Relação entre a aceleração e a inclinação.

Nesta situação as acelerações registadas variam entre -1g e 1g, sendo a relação com o ângulo de inclinação dada por:

$$\theta = \sin^{-1}\left(27{,}6757 \cdot 10^{-3} \cdot \left[CNT - CALIB\right]\right) \quad \left[rad\right]$$

Onde *CNT* é o número de contagens (de 0 a 255) e *CALIB* é o valor de calibração do sensor.

A utilização do acelerómetro obriga a uma calibração prévia. Embora não seja disponibilizado nenhum mecanismo de calibração automática, esta pode ser realizada de forma simples pelo utilizador. Para isso o dispositivo deve ser posicionado na horizontal, registando-se o valor retornado em cada eixo. Estes valores são designados por *CALIB* e são tipicamente 144. Cada eixo possui um valor distinto de *CALIB*, que deve ser utilizado na determinação da aceleração e da inclinação.

2.13.3 Frequência de leitura

A frequência de leitura especifica o número de vezes por segundo que é obtido o valor de um sensor. A frequência de leitura é igual para todos os Sensores Temporais, encontrando-se limitada pelo tempo máximo de leitura destes. A sua configuração é feita no ficheiro *USER_CONFIG.h*. A frequência máxima de leitura é assim dada por:

$$f_{MáxLeitura} = \frac{1}{36 \cdot 10^{-3} \cdot N\_TEMP} \quad [Hz]$$

Onde:

N_TEMP - Número de Sensores Temporais em utilização.

$f_{MáxLeitura}$ - Frequência máxima de leitura dos Sensores Temporais.

2.13.4 Configuração

A utilização dos Sensores Temporais obriga à especificação, no ficheiro `USER_CONFIG.h`, da frequência de leitura e do tipo de sensor em utilização. No caso dos Sonares é também necessário indicar o porto e pino por onde é enviado o pulso de activação. O dimensionamento da frequência de leitura é abordado na **Secção 2.13.3**. A sua definição é feita através da variável *TEMP_CNT_MAX*, que pode variar entre 0 a 255 e se relaciona com a frequência de leitura segundo a expressão:

$$f_{leitura} = \frac{5 \cdot 10^6}{1024 \cdot 127 \cdot TEMP\_CNT\_MAX} \quad [Hz]$$

Os Sensores Temporais apenas podem ser ligados a 7 pinos, podendo os eixos do acelerómetro ser ligados apenas a 3. A tabela seguinte indica qual o porto e pino a que cada Sensor Temporal está associado, estando a negrito aqueles a que é possível ligar os eixos do acelerómetro.

TEMP 0	TEMP 1	TEMP 2	TEMP 3	TEMP 4	TEMP 5	TEMP 6
RB0	**RB1**	**RB2**	RB4	RB5	RB6	RB7

Tabela 2 – Pinos onde é possível conectar Sensores Temporais.

Na tabela acima, TEMP X significa Sensor Temporal número X, enquanto que RBY identifica o porto B e pino Y (de 0 a 7). Para localizar correctamente estes pinos e os de envio de pulsos para os Sonares, consulte a **Secção 3.2.3**, onde se encontra o mapeamento da Placa de Controlo. No ficheiro `USER_CONFIG.h`, a selecção do porto de envio de pulsos é feita através da configuração do LAT e TRIS. A distinção entre os dois tipos de Sensores Temporais é feita na sua activação, sendo os Sonares activados como *ON_SONAR* e os eixos do Acelerómetro por *ON_ACCEL*.

O exemplo da página seguinte configura 1 eixo de Acelerómetro e 3 Sonares, onde o 0 e 3 disparam sobre a mesma direcção. De forma a evitar interferência entre estes Sonares, a sua leitura é realizada separadamente. A frequência de leitura seleccionada é de 3,8 Hz. Os pinos de pulso dos Sonares são os 0, 1 e 2 do porto D.

```
...

// Frequência de amostragem dos Sensores Temporais F_TEMP = ...
// Período de amostragem mínimo dos Sensores Temporais = 36mS*N_TEMP
#define TEMP_CNT_MAX      10
...
//
//    Definição dos Sensores Temporais (Sonares ou Acelerómetros)
//
//
// As saídas do acelerómetro só podem ser ligadas aos pinos RB0,
// RB1 e RB2.
//
// Tabela de atribuição dos pinos de leitura Sensores Temporais:
//    TEMP0  TEMP1  TEMP2  TEMP3  TEMP4  TEMP5  TEMP6
//    RB0    RB1    RB2    RB4    RB5    RB6    RB7

// Define o estado dos Sensores Temporais (OFF/ON_SONAR/ON_ACCEL)
#define TEMP0_STATE ON_SONAR
#define TEMP1_STATE ON_SONAR
#define TEMP2_STATE ON_ACCEL
#define TEMP3_STATE ON_SONAR
#define TEMP4_STATE OFF
#define TEMP5_STATE OFF
#define TEMP6_STATE OFF


// Configura o Sonar 0
#if TEMP0_STATE == ON_SONAR
#define TEMP0_TRISOUT     TRISD
#define TEMP0_LATOUT      LATD
#define TEMP0_PINOUT      0
#endif

// Configura o Sonar 1
#if TEMP1_STATE == ON_SONAR
#define TEMP1_TRISOUT     TRISD
#define TEMP1_LATOUT      LATD
#define TEMP1_PINOUT      1
#endif

// Configura o Sonar 2
#if TEMP2_STATE == ON_SONAR
#define TEMP2_TRISOUT     0
#define TEMP2_LATOUT      0
#define TEMP2_PINOUT      0
#endif

// Configura o Sonar 3
#if TEMP3_STATE
#define TEMP3_TRISOUT     TRISD
#define TEMP3_LATOUT      LATD
#define TEMP3_PINOUT      2
#endif
```

Exemplo 15 – Configuração de 1 eixo de Acelerómetro e 3 Sonares.

2.13.5 Funções C

As funções de interface com os Sensores Temporais estão nos ficheiros *TEMP.h* e *TEMP.c*. As suas características dependem do tipo de Sensor Temporal. De seguida é descrito o seu funcionamento:

```
void init_TEMP(void);
```

Função chamada pelo sistema de gestão de periféricos que procede às inicializações necessárias para ler os sensores desta classe, declarados no ficheiro *USER_CONFIG.h*. Esta função não deve ser utilizada.

```
unsigned char get_TEMPX(void);        (Sonar)
```

Função que obtém o valor do sensor X. X é um número de 0 a 6 que identifica o eixo do Acelerómetro a ler. Cada sensor possui um registo de 8 bits, permitindo valores entre 0 e 255. A conversão do valor retornado para distância é feito através da equação abaixo.

$$d = CNT \cdot 1{,}766 \quad [cm]$$

Onde:

CNT – Valor retornado pela função.

d – Distância até ao objecto mais próximo (em cm).

```
unsigned char get_TEMPX(void);        (Acelerómetro)
```

Função que obtém o valor do sensor X. X é um número de 0 a 6 que identifica o eixo do Acelerómetro a ler. Cada sensor possui um registo de 8 bits, permitindo valores entre 0 e 255. A conversão do valor retornado para aceleração ou inclinação é feito através das equações abaixo.

$$A = 27{,}6757 \cdot 10^{-3} \cdot (CNT - CALIB) \quad [g]$$

$$\theta = \sin^{-1}\left(27{,}6757 \cdot 10^{-3} \cdot [CNT - CALIB]\right) \quad [rad]$$

Onde:

A – Aceleração segundo um dado eixo (em g).

CNT – Valor retornado pela função.

CALIB – Valor obtido na calibração do acelerómetro.

θ - Ângulo de inclinação (em radianos).

2.13.6 Exemplos

O seguinte exemplo consiste numa aplicação típica de Sensores Temporais:

```
#include “IO.h”

void main(void)
{
	// Declaração de variáveis locais
	Unsigend char a[4];

	// Inicia o sistema de gestão de periféricos
	init_IO();


	// Ciclo principal
	while(1)
	{
		// Espera que passe um período
		wait_PERIOD();

		// Obtém o valor dos Sensores Temporais
		a[0] = get_TEMP0();
		a[1] = get_TEMP1();
		a[2] = get_TEMP2();
		a[3] = get_TEMP3();

		//
		// Código que processa os dados recolhidos
		//
	}
}
```

Exemplo 16 – Aplicação típica dos Sensores Temporais.

2.14 Sensores Passivos

A classe de Sensores Passivos possibilita a utilização de sensores binários, ou seja, dispositivos que apenas retornam dois valores. Tipicamente esta classe funciona com Sensores de Contacto, cujas aplicações vão desde detectar colisões a limitar a amplitude de movimentos de juntas. Independentemente do dispositivo, este tem que ser alimentado por dois canais, um de massa e outro com 5V, ambos obtidos da Placa de Controlo. As leituras são realizadas através dum terceiro canal, que possui a tensão de um dos canais de alimentação. Uma vez que apenas é necessário um pino para ler o estado destes sensores, é possível usar até 31 em simultâneo. Os sensores desta classe não necessitam de ser lidos periodicamente, pelo que não ocupam recursos do sistema de gestão de periféricos. O seu estado é actualizado apenas quando o utilizador necessita.

2.14.1 Modo de leitura

Esta classe foi projectada para funcionar com Sensores de Contacto. Estes são tipicamente alimentados por duas linhas, que devem ser obtidas a partir da Placa de Controlo. A tensão lida destes sensores corresponde à imposta por uma das linhas de alimentação. Por convenção assume-se que quando os sensores estão pressionados são retornados 5V (V_{CC}), caso contrário 0V (GND). O estado dos sensores só é lido quando requisitado pelo utilizador, não existindo por isso Custo Computacional associado a esta classe. A figura seguinte ilustra como varia a tensão nas linhas do sensor:

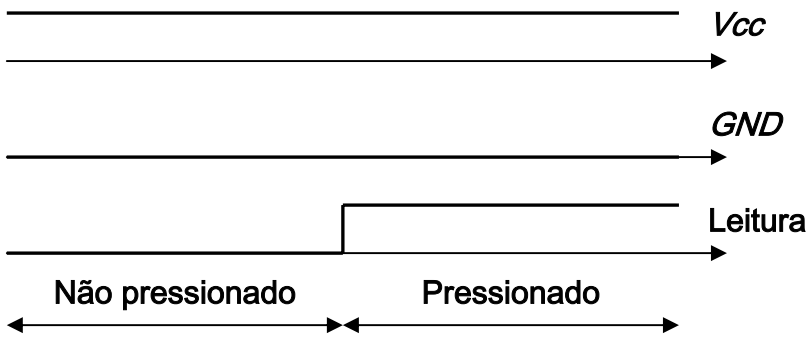


Figura 18 – Funcionamento dos Sensores de Contacto.

2.14.2 Configuração

A utilização dos Sensores Passivos obriga à especificação, no ficheiro *USER_CONFIG.h*, do pino e porto onde cada um é lido. A selecção do porto de leitura é efectuada através da indicação do PORT e TRIS respectivos. Os Sensores Passivos podem ser ligados aos 31 pinos disponíveis na Placa de Controlo, cujo mapeamento se encontra na **Secção 4.2.3**.

O exemplo seguinte configura 2 Sensores passivos, o primeiro no porto A, pino 2 e o segundo no porto C, pino 3.

```
//
//     Definição dos Sensores Passivos
//

// Define o estado dos Sensores Passivos
#define      PS0_STATE    ON
#define      PS1_STATE    ON
#define      PS2_STATE    OFF
#define      PS3_STATE    OFF
#define      PS4_STATE    OFF
...

// Configura o Sensor Passivo 0
#if PS0_STATE
#define PS0_TRIS    TRISA
#define PS0_PORT    PORTA
#define PS0_PIN     2
#endif

// Configura o Sensor Passivo 1
#if PS1_STATE
#define PS1_TRIS    TRISC
#define PS1_PORT    PORTC
#define PS1_PIN     3
#endif
```

Exemplo 17 – Configuração de dois Sensores Passivos.

2.14.3 Funções C

As funções de interface com os Sensores Passivos estão nos ficheiros *PASSIVE.h* e *PASSIVE.c*. As suas funcionalidades são descritas de seguida:

```
void init_PS(void);
```

Função chamada pelo sistema de gestão de periféricos que procede às inicializações necessárias para ler os sensores desta classe, declarados no ficheiro *USER_CONFIG.h*. Esta função não deve ser utilizada.

```
unsigned char get_PSX(void);
```

Função que obtém o estado do sensor X. X é um número de 0 a 30 que identifica o Sensor Passivo a ler. Os valores retornados são 0 ou 1, que correspondem a não premido e premido respectivamente.

2.14.4 Exemplos

O seguinte exemplo mostra uma aplicação típica de Sensores Passivos:

```
#include “IO.h”

void main(void)
{
	// Declaração de variáveis locais
	unsigend char a[2];

	// Inicia o sistema de gestão de periféricos
	init_IO();


	// Ciclo principal
	while(1)
	{
		// Espera que passe um período
		wait_PERIOD();

		// Obtém o valor dos Sensores Passivos
		a[0] = get_PS0();
		a[1] = get_PS1();

		//
		// Código que processa os dados recolhidos
		//
	}
}
```

Exemplo 18 – Aplicação típica de Sensores Passivos.

2.15 Temporizador

O temporizador possibilita ao utilizador esperar intervalos de tempo específicos. Este mecanismo permite uma discretização de sistemas muito fiável, com uma resolução de 0,2 μs. A interface é semelhante à dos semáforos em C, sendo chamada uma função que suspende a execução do código durante um determinado período de tempo. Este período é definido pelo utilizador e a sua contagem é realizada consecutivamente, sem que seja necessário assinalar o seu início. É usual ser necessário esperar intervalos de tempo diferentes ao longo do código. Isto é possível indicando qual o período desejado, antes de efectuar a espera. Note-se que ao fim de um período, o sistema volta a ser discretizado com base no valor definido inicialmente. Estas duas funcionalidades são abordadas no exemplo da **Secção 2.15.4**.

2.15.1 Frequência de Discretização

A Frequência de Discretização define quantas vezes por segundo é que o código condicionado pelo temporizador é executado. A definição da frequência é feita no ficheiro *USER_CONFIG.h*, sendo o valor máximo permitido de 38,44 Hz.

2.15.2 Configuração

A utilização do Temporizador obriga à sua activação e à definição da frequência de discretização, no ficheiro *USER_CONFIG.h*. Esta é realizada através da variável *PERIOD_CNT_MAX,* que pode variar entre 0 e 255 e se relaciona com a frequência de discretização segundo a equação abaixo:

$$f_{PERIOD} = \frac{5 \cdot 10^6}{1024 \cdot 127 \cdot PERIOD\_CNT\_MAX} \quad [Hz]$$

Onde:

f_{PERIOD} - Frequência de Discretização (em Hertz).

O exemplo seguinte ilustra como activar o Temporizador e configurar a sua frequência de funcionamento para 9,6 Hz.

```
...
// Frequência do temporizador
// Ftemp = 5MHz/(1024*127*PERIOD_CNT_MAX)
#define PERIOD_CNT_MAX    41
...

//
//     Definições do Temporizador
//

// Define o estado do Temporizador (ON/OFF)
#define PERIOD_STATE      ON
```

Exemplo 19 – Configuração do Temporizador.

2.15.3 Funções C

As funções de interface com o Temporizador estão nos ficheiros *PERIOD.h* e *PERIOD.c*. As suas funcionalidades são descritas de seguida:

```
void init_PERIOD(void);
```

Função chamada pelo sistema de gestão de periféricos que procede às inicializações necessárias para usar o Temporizador. Esta função não deve ser chamada pelo utilizador.

```
void wait_PERIOD(void);
```

Função que espera até que passe um determinado intervalo de tempo. Este período é definido no ficheiro *USER_CONFIG.h*, podendo ser alterado durante uma iteração pela função `set_PERIOD`. Os períodos são contados consecutivamente, não sendo necessário activar a contagem após uma chamada a esta função.

```
void set_PERIOD(unsigned char wait_value);
```

Função que altera o intervalo de tempo a esperar durante uma iteração. A relação entre o tempo a esperar e `wait_value` é dada por:

$$T_{PERIOD} = \frac{1024 \cdot 127 \cdot wait\_value}{5 \cdot 10^6} \quad [s]$$

Onde:

T_{PERIOD} - Período de espera (em segundos).

2.15.4 Exemplos

O seguinte exemplo mostra uma aplicação típica do temporizador:

```
#include “IO.h”

void main(void)
{
	// Declaração de variáveis locais
	unsigend char a[2];

	// Inicia o sistema de gestão de periféricos
	init_IO();


	// Ciclo principal
	while(1)
	{
		// Espera que passe um período normal
		wait_PERIOD();

		// Pára um motor
		set_PWM0(0);

		// Obtém o valor dos Sensores Passivos
		a[0] = get_PS0();
		a[1] = get_PS1();

		//
		// Código que processa os dados recolhidos
		//

		// Especifica um tempo de espera diferente
		set_PERIOD(4);

		// Espera que passe o período definido acima
		wait_PERIOD();

		// Actua um motor
		set_PWM0(127);
	}
}
```

Exemplo 20 – Aplicação típica do Temporizador.

3. Software da Placa Rádio

3.1 Noções Básicas

O software para a Placa Rádio permite transmitir/receber dados através deste dispositivo. A interface fornecida é constituída por duas camadas: uma de baixo nível que consiste num Servidor Rádio, programado em C++, que garante a comunicação em tempo real com a placa e outra de alto nível, composta por um conjunto de funções em *Matlab* que permitem interagir com o servidor. O utilizador apenas necessita trabalhar em *Matlab*, sendo o baixo nível gerido automaticamente. As funcionalidades disponibilizadas são: envio e recepção de dados, controlo do número de mensagens enviadas, transmitidas e perdidas e verificação do estado da ligação. O Servidor Rádio disponibiliza ao utilizador a última mensagem recebida e transmite de acordo com o protocolo descrito na **Secção 2.9.1**. As funções *Matlab* são descritas em pormenor na **Secção 3.5**.

3.2 Inclusão do software em projectos

As funções disponibilizadas para interagir com a Placa Rádio foram desenvolvidas para *Matlab*, versão 7.1 ou posteriores. A directoria com os ficheiros descritos na **Secção 3.3** não deve ser incluída nas conhecidas pelo *Matlab.* A utilização das funções fornecidas obriga a que estas estejam na directoria de projecto do utilizador, juntamente com os executáveis relacionados com o Servidor Rádio.

A listagem seguinte identifica os 18 ficheiros que o utilizador deve copiar para as directorias com os seus projectos:

```
radio.mexw32
radio_server.exe
radio_end.m
radio_lost_number.m
radio_lost_reset.m
radio_off.m
radio_on.m
radio_recv.m
radio_recv_number.m
radio_recv_reset.m
radio_send.m
radio_send_cmd.m
radio_send_number.m
radio_send_reset.m
radio_status.m
radio_status_reset.m
udp_off.m
udp_on.m
```

3.3 Estrutura de ficheiros

O software para interface com a Placa Rádio é formado por dois pacotes: um com as funções *Matlab* e os executáveis e outro com os ficheiros de código. O primeiro pacote inclui 18 ficheiros, cuja listagem se encontra na **Secção 3.2**. Os executáveis são o `radio_server.exe` e o `radio.mexw32`, que implementam respectivamente o Servidor Rádio e aplicação *Matlab* que o lança. O Servidor Rádio foi criado em linguagem de programação C++, estando os seus ficheiros de código e de projecto (*Visual Studio*) na directoria `radio_server`. Os ficheiros que geram a aplicação *Matlab* estão na directoria `radio` e são o `radio.c` e `radio.h`. Estes foram desenvolvidos em *Matlab* e são compilados escrevendo na linha de comandos:

```
mex –g radio.c
```

O diagrama seguinte ilustra como as diferentes directorias e pacotes se relacionam:

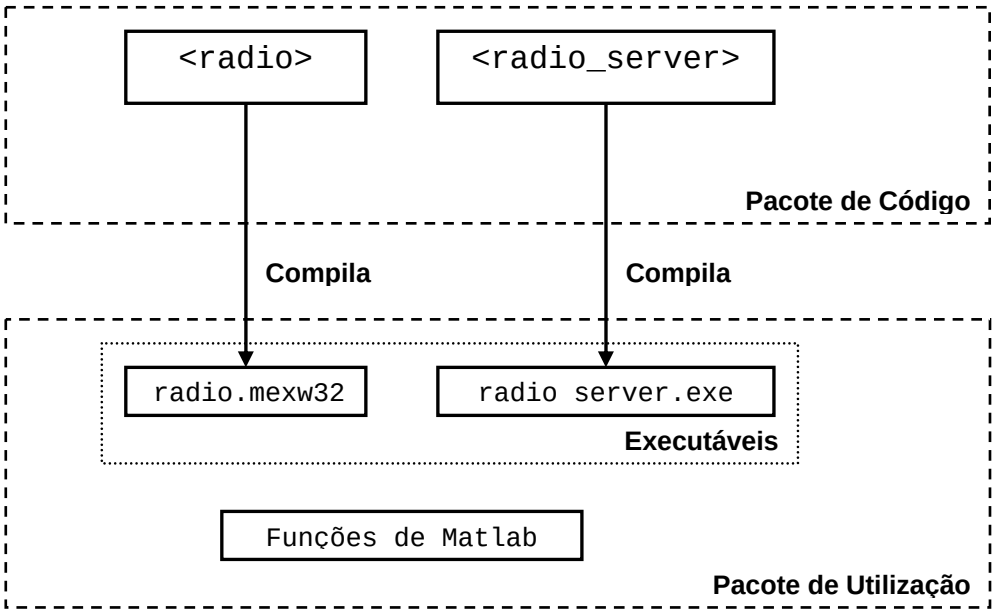


Figura 19 – Diagrama da estrutura do software para a Placa Rádio.

3.4 Servidor Rádio

O Servidor Rádio é uma aplicação de baixo nível, programada em C++, que gere a comunicação com a Placa Rádio em tempo real. Os principais objectivos desta camada são implementar o Protocolo de Comunicação, descrito na **Secção 2.9.1** e efectuar a interface com o *Matlab*. A ligação à placa é efectuada via porta série, sendo as suas propriedades definíveis pelo utilizador. Para informações relativas aos parâmetros que é possível definir consulte a **Secção 3.5**. A comunicação com o *Matlab* é realizada via UDP, sendo disponibilizadas diversas funcionalidades: envio e recepção de dados, contabilização das mensagens transmitidas, recebidas e perdidas e controlo do estado a ligação. É de notar que o sistema garante que o utilizador recebe sempre a mensagem mais recente e que a transmissão dos seus dados é sincronizada pelo Placa de Controlo. A figura seguinte explicita a interacção entre o *Matlab*, o Servidor Rádio e a Placa Rádio.

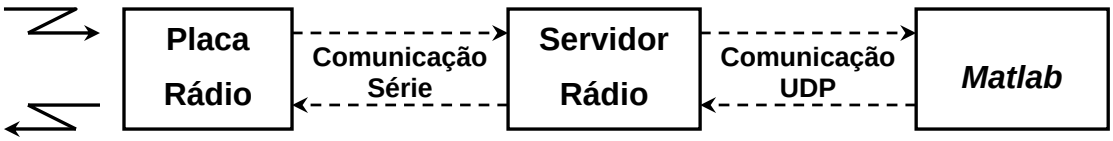


Figura 20 – Esquema de ligação entre a Placa Rádio, o Servidor Rádio e o *Matlab*.

3.5 Funções de *Matlab*

O software para a Placa Rádio possui 16 funções em *Matlab* que disponibilizam diversas funcionalidades ao utilizador. As suas características são abordadas seguidamente em detalhe. É possível obter ajuda para cada função escrevendo na linha de comandos do *Matlab*:

```
help [nome da função]
```

`radio_end`

Função que envia uma mensagem de controlo ao Servidor Rádio para que este termine. Não existem argumentos de entrada. É retornado 1 em caso de sucesso e 0 caso contrário. Esta função é evocada em `radio_off`, não devendo ser utilizada directamente pelo utilizador.

`radio_lost_number`

Função que obtém o número de mensagens perdidas, ou seja, que foram enviadas pelo utilizador e não foram recebidas na Placa de Controlo. Não existem argumentos de entrada. É retornado o número de mensagens perdidas em caso de sucesso e -1 caso contrário. É possível reiniciar a contagem executando a função `radio_lost_reset`.

`radio_lost_reset`

Função que põe a zero o número de mensagens perdidas, ou seja, que foram enviadas pelo utilizador e não foram recebidas na Placa de Controlo. Não existem argumentos de entrada. É retornado 1 em caso de sucesso e 0 caso contrário.

`radio_off`

Função que desliga o Servidor Rádio e termina a comunicação. Esta função deve ser sempre executada após o término da comunicação rádio, caso contrário o Servidor fica a correr, mesmo que o *Matlab* seja encerrado. Não existem argumentos de entrada. É retornado 1 em caso de sucesso e 0 caso contrário.

`radio_on`

Função que inicia a comunicação rádio. Esta função inicia a comunicação UDP, lança o Servidor Rádio, configura a Placa Rádio e a ligação série a esta. Os argumentos de entrada desta função são:

`port` – nome da porta série (COM1, COM2, COM3, etc)

baudrate – velocidade de comunicação via porta série (4800, 9600, 19200 e 38400 bps)

`buffer_in_size` – número de bytes das mensagens a receber da Placa de Controlo (1-140)

`buffer_out_size` – número de bytes das mensagens a transmitir para a Placa de Controlo (1-140)

`power` – potência de transmissão da Placa Rádio (0-9)

`channel` – canal de comunicação da Placa Rádio (0-9)

`timeout` – tempo sem receber mensagens da Placa de Controlo para que se considere que falharam as comunicações (em milisegundos)

Devido a funcionamento incorrecto de hardware, a comunicação nos canais 1 e 9 não se efectua correctamente. A potência de transmissão 0 corresponde a 1mW e 9 a 10mW. É retornado 1 em caso de sucesso e 0 caso contrário. É possível terminar a comunicação rádio chamando a função `radio_off`.

`radio_recv`

Função que obtém a última mensagem rádio recebida. Não existem argumentos de entrada. É retornada a última mensagem recebida em caso de sucesso e `[]` caso não exista uma nova mensagem ou em caso de erro.

`radio_recv_number`

Função que obtém o número de mensagens recebidas da Placa de Controlo. Não existem argumentos de entrada. É retornado o número de mensagens recebidas em caso de sucesso e -1 caso contrário. É possível reiniciar a contagem executando a função `radio_recv_reset`.

`radio_recv_reset`

Função que põe a zero o número de mensagens recebidas da Placa de Controlo. Não existem argumentos de entrada. É retornado 1 em caso de sucesso e 0 caso contrário.

`radio_send`

Função que envia mensagens via rádio, para a Placa de Controlo. O tamanho da mensagem a enviar deve corresponder ao declarado em `radio_on`, caso contrário a mensagem não é enviada. Esta situação não origina no entanto uma mensagem de erro. O único argumento de entrada é a mensagem a transmitir. É retornado 1 em caso de sucesso e 0 caso contrário.

`radio_send_cmd`

Função que envia comandos para o Servidor Rádio. O único argumento de entrada é o comando a enviar. Esta função é usada por outras, não devendo ser chamada directamente pelo utilizador. É retornado 1 em caso de sucesso e 0 caso contrário.

`radio_send_number`

Função que obtém o número de mensagens enviadas para a Placa de Controlo. Não existem argumentos de entrada. É retornado o número de mensagens enviadas em caso de sucesso e -1 caso contrário. É possível reiniciar a contagem executando a função `radio_send_reset`.

`radio_send_reset`

Função que põe a zero o número de mensagens enviadas para a Placa de Controlo. Não existem argumentos de entrada. É retornado 1 em caso de sucesso e 0 caso contrário.

`radio_status`

Função que obtém o último erro/aviso relativo à ligação rádio. É ainda gerada uma mensagem para cada erro, que é apresentada num *handle* especificado pelo utilizador ou na linha de comandos. O único argumento de entrada da função é opcional e especifica o *handle* para imprimir a mensagem. A função retorna 0 caso não exista nenhum erro e o código do erro caso contrário. A listagem seguinte identifica todos os erros que podem ocorrer:

```
255    - Erro: falhou a comunicação com a porta série
1      - Erro: não foi encontrada a porta série
2      - Erro: porta série em utilização
3      - Erro: argumentos inválidos
4      - Erro: não foi possível configurar a porta série
5      - Erro: não foi possível configurar os timeout da porta série
6      - Erro: BaudRate não suportada pelo transceiver rádio
7      - Erro: não foi possível configurar a BaudRate da porta série
8      - Erro: não foi encontrada a Placa Rádio
9      - Erro: não foi possível configurar o handshake com a Placa Rádio
```

```
10    - Erro: não foi possível configurar a potência da Placa Rádio
11    - Erro: não foi possível configurar o canal da Placa Rádio
12    - Erro: falhou o envio de dados para a Placa Rádio
14    - Erro: falhou o lançamento da thread do Servidor Rádio
200   - Aviso: perdeu-se a comunicação com o robot
```

`radio_status_reset`

Função que põe apaga o último erro/aviso que ocorreu na ligação rádio. Não existem argumentos de entrada. É retornado 1 em caso de sucesso e 0 caso contrário.

`udp_off`

Função que termina a comunicação UDP com o Servidor Rádio. Esta função é evocada em `radio_off` e não deve ser chamada pelo utilizador. Não existem argumentos de entrada. É retornado 1 em caso de sucesso e 0 caso contrário.

`udp_on`

Função que inicia a comunicação UDP com o Servidor Rádio. Esta função é evocada em `radio_on` e não deve ser chamada pelo utilizador. Não existem argumentos de entrada. É retornado 1 em caso de sucesso e 0 caso contrário.

3.6 Exemplos

O exemplo seguinte é um *script* em *Matlab* que inicia a comunicação rádio, recebe mensagens com dados e envia comandos para a Placa de Controlo com um período de 0,2 segundos. As propriedades da comunicação rádio são: porta série 'COM1', velocidade de comunicação série 38400 bps, tamanho das mensagens a receber 10 bytes, tamanho das mensagens a transmitir 5 bytes, potência de transmissão 10mW, canal de comunicação 0 e perda de comunicações em 1000 ms.

```
% Inicia a comunicação rádio
if ~radio_on('COM1',38400,10,5,9,8,1000)

      % Termina
      return;
end


% Ciclo de espera de mensagens
while(1)

      % Recebe uma nova mensagem
      new_msg = radio_recv();

      % Verifica se recebe uma nova mensagem
      if ~isempty(new_msg)

            %
            % Calcula a actuação a enviar para a Placa de Controlo
            %

            % Envia a actuação para Placa de Controlo
            radio_send(new_msg);

      end

      % Verifica o estado da ligação
      if radio_status()

            % Termina
            return;
      end

end
```

Exemplo 21 – *Script Matlab* com uma aplicação de controlo genérica.

4 Hardware

4.1 Placa Rádio para PC

4.1.1 Listagem de Componentes

Referência	Componente	Fabricante	Quantidade
-	Resistência 150Ω	-	3
-	Condensador Cerâmico 100nF	-	4
-	Condensador Electrolítico 100nF	-	6
-	Díodo 1N4148	-	1
-	Regulador LM7805CT	-	1
19058720	Conjunto Led's Vermelho/Amarelo/Verde	MENTOR	1
ST232ABN	MAX232	STMICROELECTRONICS	1
ER400TRS	Transceiver ER400RTS	EASY RADIO	1
5504F1-09S-02-03	Conector DB9 PCB	MULTICOMP	1
SCRTM4RA	Conector PCB para antena	RF SOLUTIONS	1
FLEXI-M4-433	Antena 50Ω	RF SOLUTIONS	1
2AS3T2A1M2RE	Interruptor 3 posições	MULTICOMP	1
FK 242 SA 220 O	Dissipador de calor para regulador	FISCHER ELEKTRONIK	1
816-AG11D-ESL-LF	Socket IC 16 Pinos	TYCO ELECTRONICS/AUGAT	1

Tabela 3 – Listagem de material para a Placa Rádio para PC.

4.1.2 Descrição

A placa rádio para ligação a um PC via porta série (*RS-232*), permite a comunicação sem fios entre dois terminais. O software para esta placa é descrito na **Secção 2**.

O circuito da placa permite ligar o *Transceiver*[3] (modelo *ER400RTS*) à porta série. A interface entre estes dois componentes é realizada por um *MAX232*, que efectua a conversão dos níveis de tensão da porta série (0V a 9V) para níveis de tensão suportados pelo *Transceiver* (0V a 5V).

Os condensadores acoplados às linhas de alimentação permitem eliminar componentes de ruído que interferem com o funcionamento do *ER400TRS*, enquanto que os conectados ao *MAX232* fazem parte da sua montagem típica.

Os componentes existentes no circuito funcionam a 5V, pelo que existe um regulador de tensão (*LM7805CT*) que garante este valor de alimentação, atenuando ruído e os efeitos de picos. A existência deste componente permite ainda a possibilidade de ligação de diversas fontes de alimentação, cuja tensão seja superior a 7.4V e inferior a 9V. Caso seja trocado o sentido da alimentação da placa, existe um díodo de protecção (*1N4148*) que evita que os componentes sejam danificados.

Os emissores de luz (*LED – Light Emitting Diode*) são indicadores do estado da placa. O emissor vermelho assinala que esta se encontra ligada, o verde que está a transmitir para o meio e o amarelo que se encontra a receber dados. Os *leds* verde e amarelo apagam-se quando se enviam/recebem dados, pelo que piscam quando estas acções decorrem.

O circuito pode ser alimentado por uma pilha de 9V, uma bateria ou uma fonte de alimentação, desde que estas possuam um adaptador para ligação ao terminal de alimentação da placa (compatível com pilha de 9V). Como o regulador de tensão necessita dissipar a energia em excesso, para manter a tensão de 5V à sua saída, é normal que se verifique um ligeiro aquecimento. Este dispositivo possui por isso *thermal shutdown,* que corta a corrente ao circuito caso seja atingida uma temperatura máxima de segurança.

[3] Dispositivo responsável pelas comunicações rádio.

4.1.3 Esquemas

O esquema eléctrico do circuito é o que se segue:

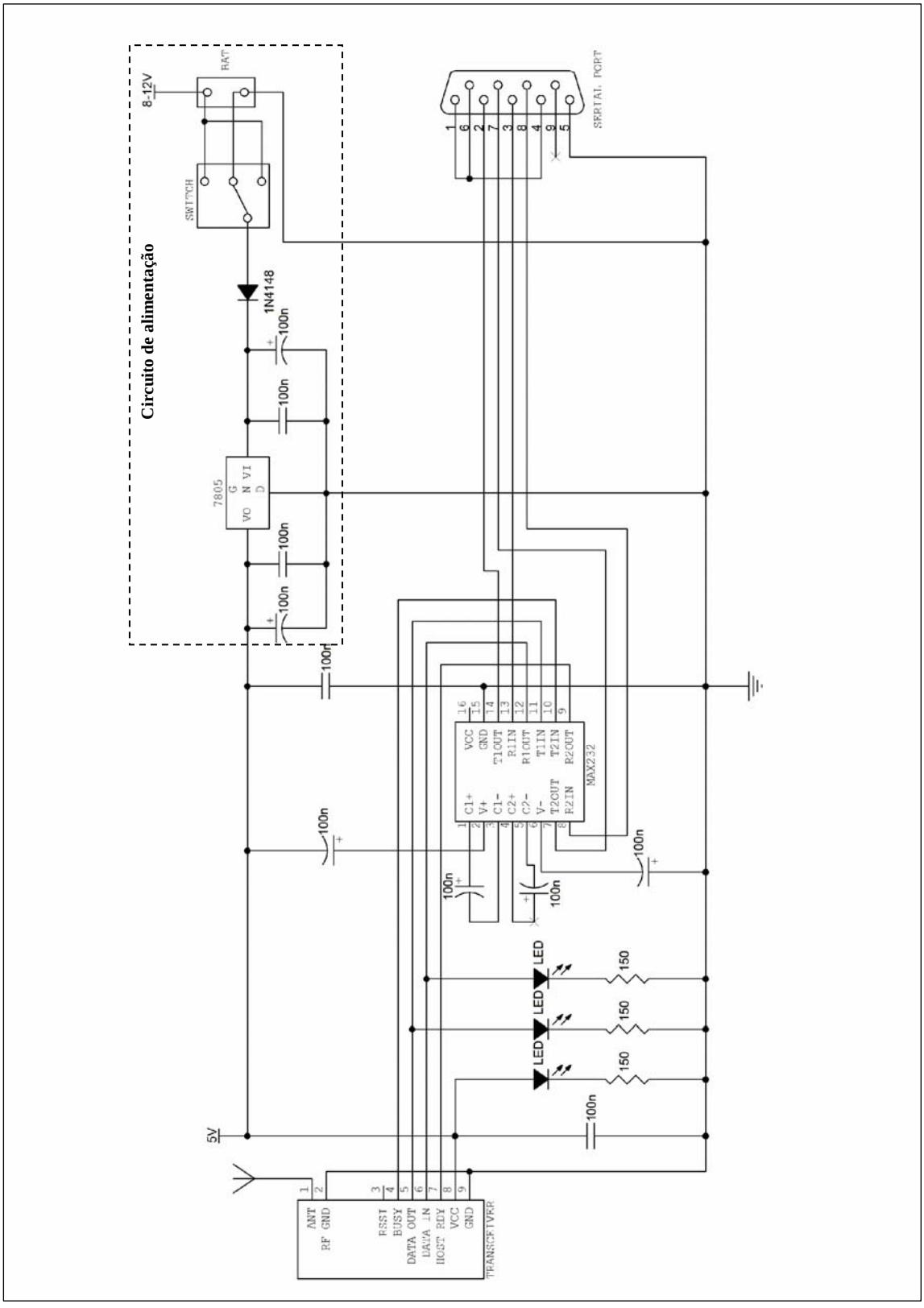


Figura 21 – Esquema eléctrico do circuito da Placa Rádio para PC.

As imagens que se seguem são *layouts* de cada uma das faces, necessários para o fabrico desta placa.

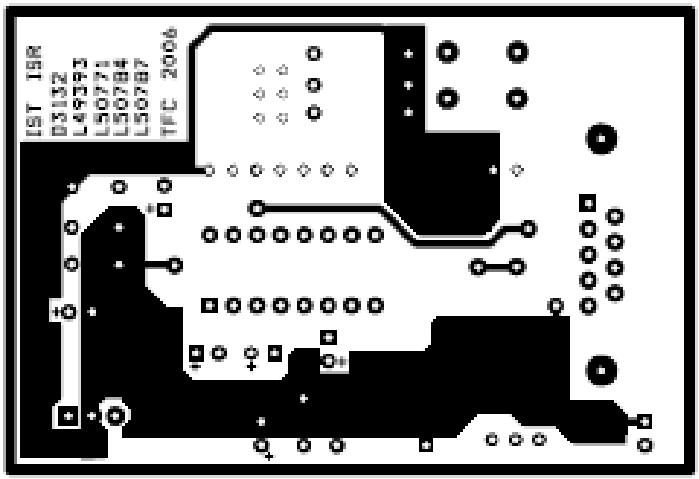


Figura 22 – Face superior da Placa Rádio para PC.

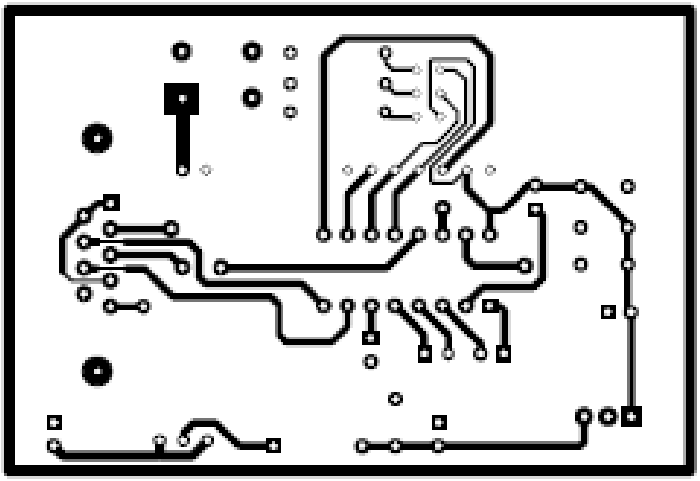


Figura 23 – Face inferior da Placa Rádio para PC.

Note-se que estas imagens se encontram espelhadas e não estão à escala. No final deste manual encontram-se as cópias, em tamanho real, que podem ser usados para o fabrico de novas placas.

4.1.4 Modo de Funcionamento

Na figura seguinte encontram-se assinalados os componentes mais importantes presentes na placa.

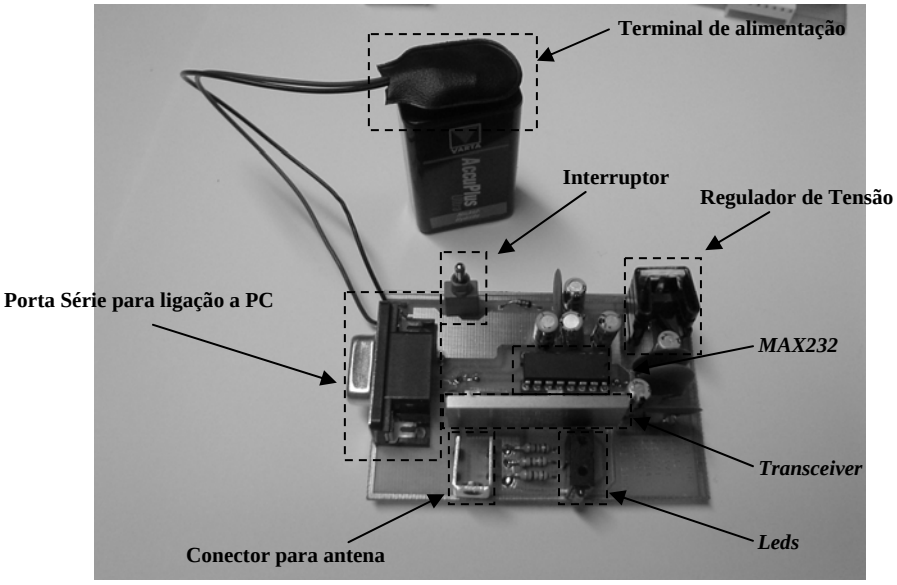


Figura 24 – Placa Rádio para PC.

Note-se que existe um terminal para ligação de uma antena. É conveniente a sua utilização para aumentar o alcance de transmissão rádio.

Para utilizar a Placa Rádio para PC, é conveniente seguir o seguinte procedimento de ligação:

i. Ligar pilha 9V, fonte de alimentação ou bateria (Tensão máxima 9V);
ii. Conectar o Cabo Série ao PC;
iii. Ligar o Cabo Série à Placa Rádio para PC;
iv. Ligar o interruptor para um dos lados (é indiferente qual o sentido para onde se liga o interruptor pois apenas existe um tipo de alimentação). Note-se que ao ligar todos os *leds* devem ficar acesos.

4.2 Placa de Controlo

4.2.1 Listagem de Componentes

Referência	Componente	Fabricante	Quantidade
-	Resistência 270Ω	-	1
-	Resistência 390Ω	-	3
-	Resistência 10kΩ	-	7
-	Condensador Cerâmico 0.1µF	-	3
-	Condensador Cerâmico 15pF	-	2
-	Condensador Cerâmico 1µF	-	1
-	Condensador Electrolítico 1µF	-	2
	Díodo Led Verde	-	1
	Díodo Led Vermelho	-	1
	Díodo Led Amarelo	-	1
STPS2L60	Díodo Schottky 2A	STMICROELECTRONICS	1
-	Regulador LM7805CT	-	1
LF A189A	Cristal 20MHz	C-MAC FREQUENCY PRODUCTS	1
ER400TRS	Transceiver ER400RTS	EASY RADIO	1
PIC18F4520-I/P	Micro controlador PIC18F4520	MICROCHIP	1
DC10A	Conector para transformador	CLIFF ELECTRONIC	1
MCDTSHW6-9N	Botão de pressão	MULTICOMP	1
FLEXI-M4-433	Antena 50Ω	RF SOLUTIONS	1
1MS3T1B5M1QE	Interruptor 3 posições	MULTICOMP	1
FK 242 SA 220 O	Dissipador de calor para regulador	FISCHER ELEKTRONIK	1
390262-5	Socket IC 40 Pinos	TYCO ELECTRONICS/AMP	1
22-27-2061	Header 6 pinos	MOLEX	1
22-01-2065	Crimp Housing 6 pinos	MOLEX	1
MC9A12-2034	Header 20 pinos para *flat cable*	MULTICOMP	1
77311-401-36	Régua Simples	FCI	2
77313-418-72	Régua Dupla	FCI	2

Tabela 4 – Listagem de material para a placa de controlo.

4.2.2 Descrição

A Placa de Controlo tem a possibilidade de ser acoplada a um robot ou a qualquer outra aplicação onde sejam necessárias comunicações sem fios e ligação de diversos sensores/actuadores. Esta efectua a interface entre um micro controlador (*PIC18F4520*) e os diversos dispositivos que lhe sejam ligados. As comunicações rádio são realizadas por um *Transceiver* (modelo *ER400RTS*) e permitem a comunicação com a Placa Rádio para PC (consultar **Secção 3**).

A Placa de Controlo pode ser dividida em vários "sub circuitos" (representados na **Figura 24**), que serão descritos detalhadamente em seguida.

Os componentes existentes no circuito funcionam a 5V, pelo que existe um regulador de tensão (*LM7805CT*) que garante este valor de alimentação, atenuando ruído e os efeitos de picos. A existência deste componente permite ainda a possibilidade de ligação de diversas fontes de alimentação, cuja tensão seja superior a 7.4V e inferior a 9V. Caso seja trocado o sentido da alimentação da placa, existe um díodo de protecção (*1N4148*) que evita que os componentes sejam danificados.

Existe um circuito oscilador externo para que se gera a frequência de funcionamento para o micro controlador. Este é constituído por dois condensadores, uma resistência e um cristal de 20MHz. Sendo necessários quatro ciclos de relógio para executar uma operação no PIC, a frequência de funcionamento deste é de 5 MHz.

Para efectuar a programação do micro controlador, existe um terminal ao qual é ligado o cabo que conecta ao programador *ICD2*, permitindo assim que se possa actualizar o *software* da PIC sem que seja necessário proceder à sua remoção.

É possível fazer *reset* ao micro controlador sem ser por *software*. Para isso foi adicionado um pequeno circuito constituído por um botão de pressão, um condensador e uma resistência. Este botão encontra-se numa das extremidades da placa e ao ser premido o PIC pára e faz *reset*. Durante esta operação o *led* verde permanece apagado, indicando que a comunicação rádio está inactiva.

Para a comunicação rádio foi utilizado um *Transceiver ER400RTS*. Este componente necessita de uma alimentação sem ruído de alta-frequência, pelo que foi necessário incluir no circuito alguns condensadores. O envio/recepção de dados é feito através do *Data Out* e *Data In* (pinos 5 e 6), que estão ligados aos portos *TX* e *RX* do micro controlador (pinos 25 e 26).

A Placa de Controlo permite a ligação de diversos periféricos, existindo cabos específicos para cada um deles. Os portos existentes em redor do micro controlador são usados para conectar estes cabos. No caso dos odómetros da Lego é ainda necessário realizar a ligação a um conjunto de resistências de *pull up,* que se encontram assinaladas na **Figura 29**.

A esta placa podem ser ligados diversos tipos de sensores e actuadores que se encontram agrupados em classes, de acordo com o seu funcionamento. Estas são:

- Motores

- Sensores Passivos

  Sensores de Contacto

- Sensores Activos:

  Sensores de Proximidade

  Sensores de Rotação (LEGO)

- Sensores de Rotação de Alta Resolução

  Odómetros de alta Resolução

- Sensores Temporais

  Acelerómetro de 2 eixos

  Sonares

A ligação a Motores e Sensores de Proximidade é feita através da Placa de Interface para Módulos de Potência. Esta comunica com a Placa de Controlo através de um *flat cable*.

Os emissores de luz (*LED – Light Emitting Diode*) são indicadores do estado da placa. O emissor vermelho assinala que esta se encontra ligada, o verde que está a transmitir para o meio e o amarelo que se encontra a receber dados. Os *leds* verde e amarelo apagam-se quando se enviam/recebem dados, pelo que piscam quando estas acções decorrem.

O circuito pode ser alimentado por uma pilha de 9V, uma bateria ou uma fonte de alimentação, desde que estas possuam um adaptador para ligação ao terminal de alimentação da placa (compatível com pilha de 9V). Como o regulador de tensão necessita dissipar a energia em excesso, para manter a tensão de 5V à sua saída, é normal que se verifique algum aquecimento. Este dispositivo possui por isso *thermal shutdown,* que corta a corrente ao circuito caso seja atingida uma temperatura máxima de segurança.

4.2.3 Esquemas

O esquema eléctrico do circuito é o que se segue:

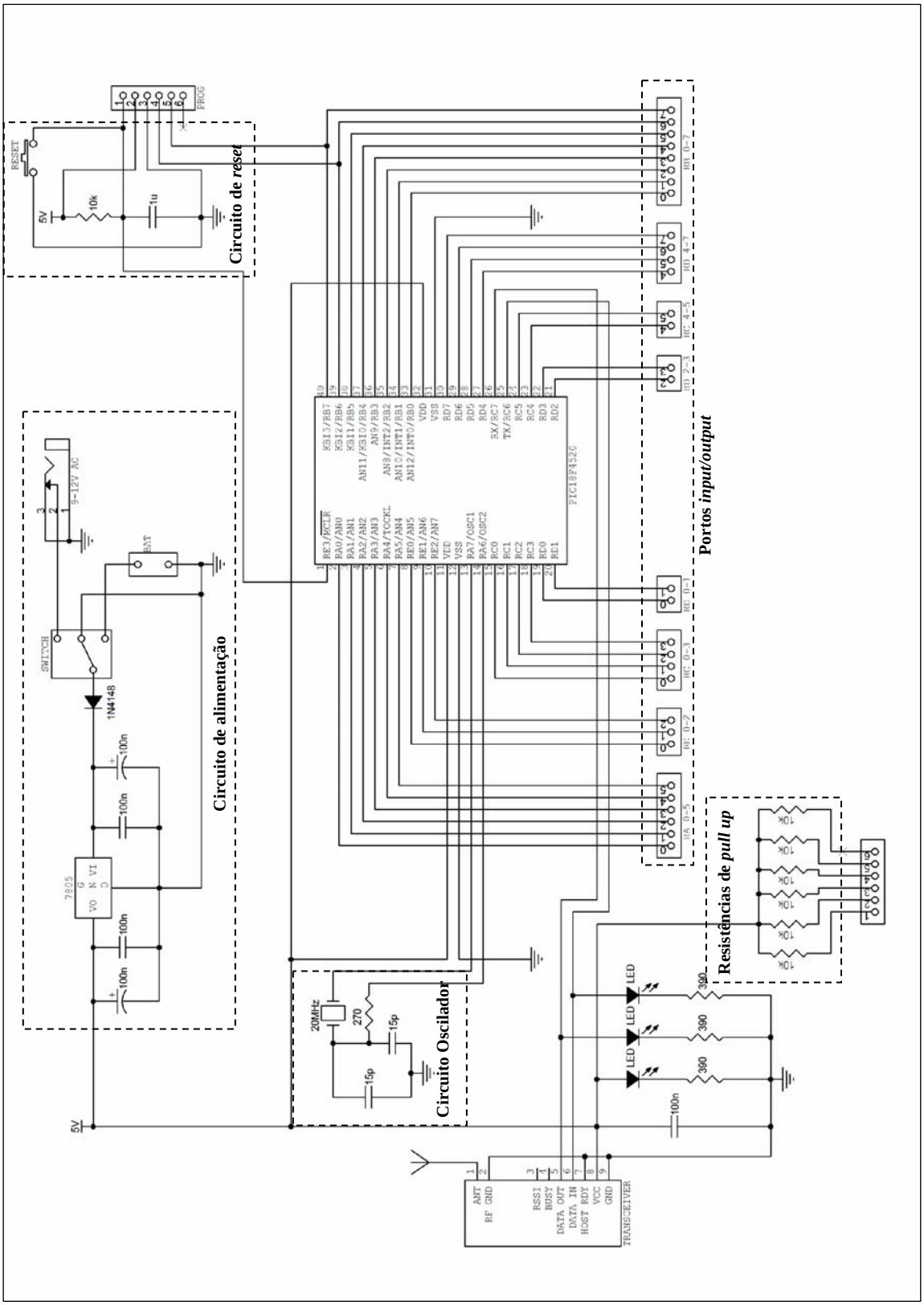


Figura 25 – Esquema eléctrico do circuito da Placa de Controlo.

A figura que se segue mostra, esquematicamente, a localização dos diversos componentes e portos na placa.

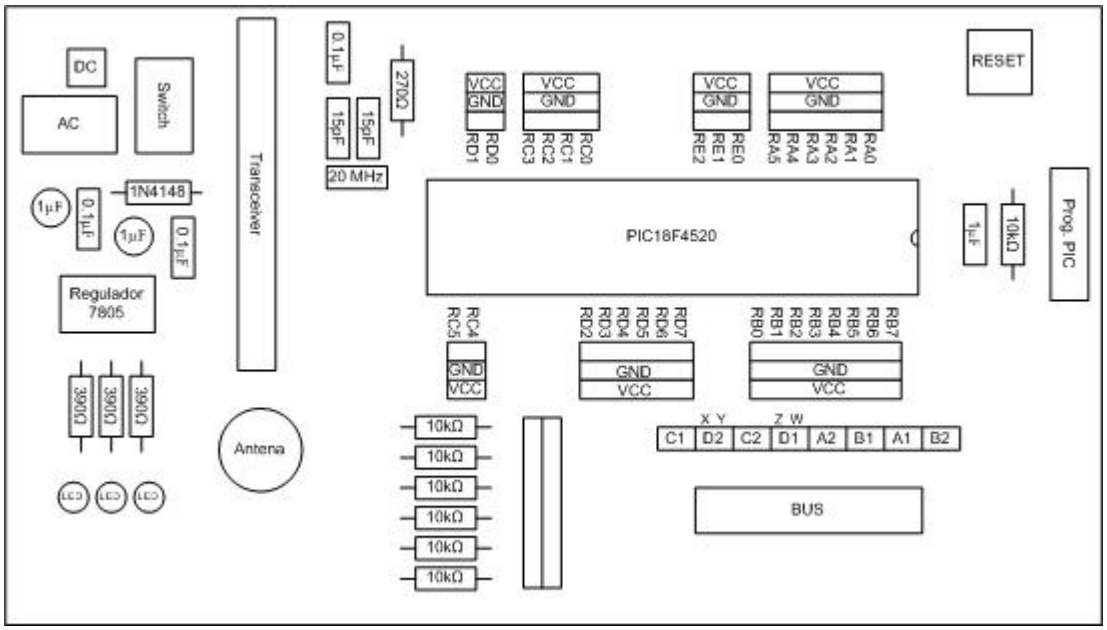


Figura 26 – Mapeamento da Placa de Controlo.

As imagens que se seguem são *layouts* de cada uma das faces, necessários para o fabrico desta placa.

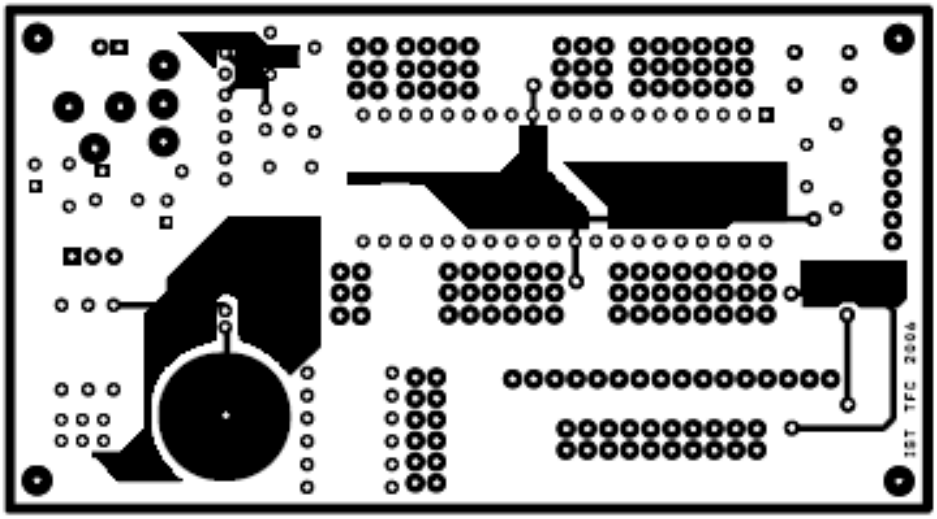


Figura 27 – Face superior da Placa de Controlo.

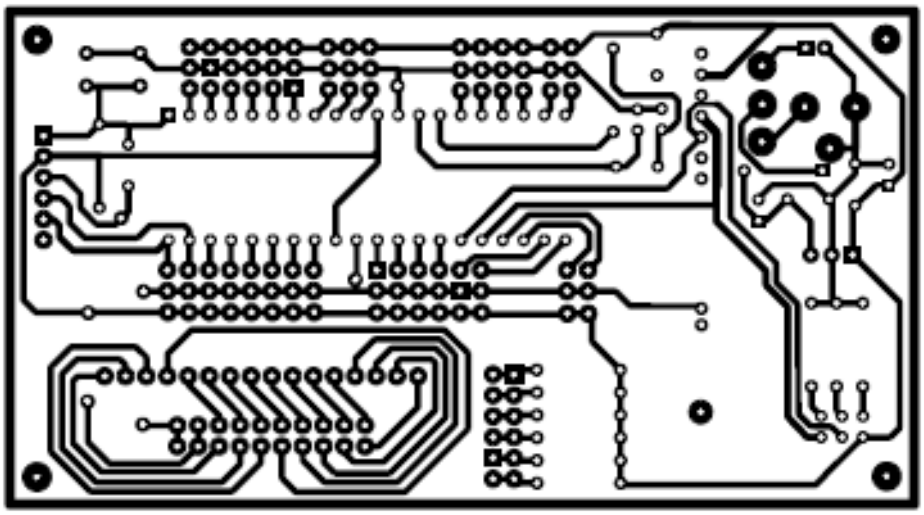


Figura 28 – Face inferior da Placa de Controlo.

4.2.4 Modo de Funcionamento

Na figura seguinte encontram-se assinalados os componentes mais importantes presentes na placa.

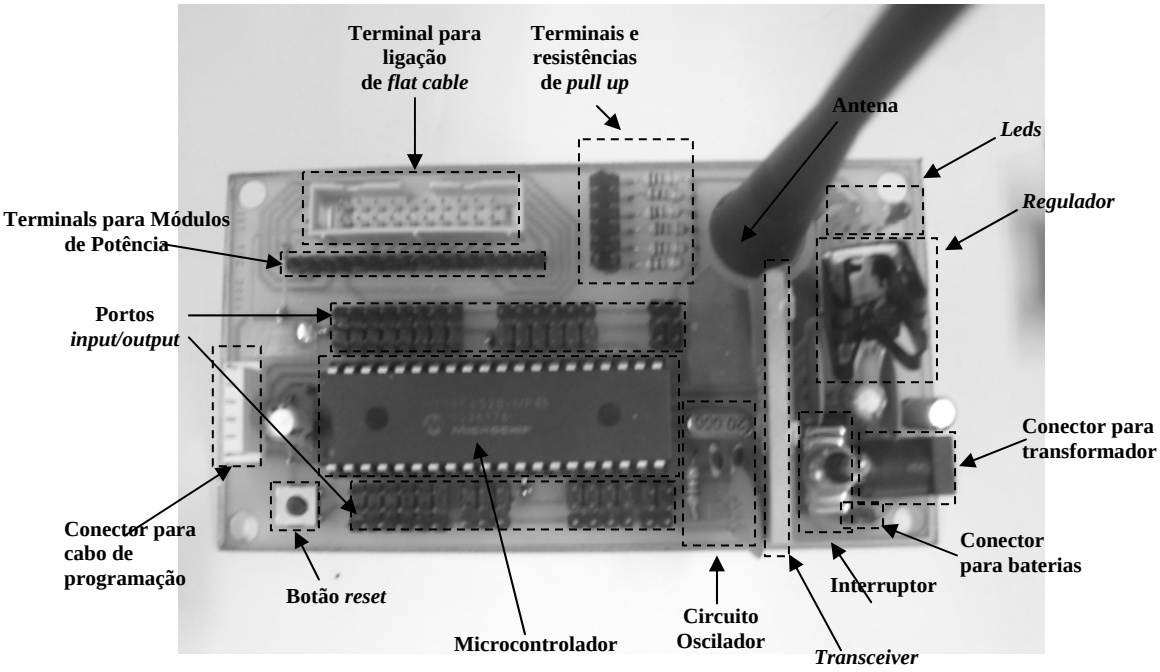


Figura 29 – Placa de Controlo.

4.2.4.1 Alimentação

Existem dois terminais diferentes para ligar a alimentação à placa, um para baterias, fontes de alimentação ou pilha de 9V e outro para transformadores. Apesar do interruptor permitir seleccionar a origem da alimentação, deve assegurar-se que apenas um tipo de alimentação está conectado à placa. Para utilizar a Placa de Controlo, é conveniente seguir o seguinte procedimento de ligação:

i. Ligar pilha 9V, fonte de alimentação, bateria ou transformador (Tensão máxima 9V) à placa;

ii. Ligar o interruptor para um dos lados consoante o tipo de alimentação se esteja a utilizar. Notar que neste instante se acendem todos os *leds* da placa, o que indica que está a funcionar correctamente. Ao colocar o interruptor para o lado do regulador é seleccionada a

alimentação ligada ao conector para baterias. Para o lado exterior da placa selecciona-se a alimentação por transformador (indicado na **Figura 29**).

Caso seja utilizada uma fonte de alimentação é conveniente ligá-la antes de conectar o cabo à placa. Este procedimento impede que os picos de tensão iniciais, gerados pela fonte, danifiquem o equipamento.

4.2.4.2 Ligação da Antena

Existe um terminal na placa para ligação da antena. É conveniente a sua utilização para aumentar o alcance de transmissão rádio. Esta deve ser conectada recorrendo à utilização de uma rosca e uma anilha. A anilha deve ser colocada entre a antena e a parte superior da placa, de modo a que seja estabelecido o contacto. Na parte inferior da placa dever-se-á colocar a rosca de modo a fixar a antena. É conveniente que a antena fique bem conectada à placa, pois pequenos deslocamentos desta sobre a superfície da placa farão *reset* ao micro controlador. Este efeito deve-se ao facto da trepidação da antena provocar variações elevadas na impedância imposta ao circuito.

4.2.4.3 Programação do micro controlador

Para programar o micro controlador deve conectar-se o cabo que liga ICD2, ao terminal da Placa de Controlo, instalado para este efeito. O programador deve por sua vez ser ligado a um PC via porta *USB*.

Figura 30 – Cabo para programar micro controlador (esquerda) e programador *ICD2* (direita).

Note-se que antes de conectar o cabo de programação à Placa de Controlo, há que assegurar que nada está conectado aos portos *RB6* e *RB7* do micro controlador (pinos 39 e 40, respectivamente). Esta

restrição resulta do facto destes pinos para além de poderem ser utilizados para ligar a sensores/actuadores, também são utilizados para efectuar a programação do dispositivo.

4.2.4.4 *Reset* ao micro controlador

É possível fazer *reset* ao micro controlador premindo o botão de pressão, existente numa das extremidades da Placa de Controlo. Note-se que caso esteja a ser utilizado um programa onde a Placa de Controlo esteja a transmitir dados, enquanto o botão estiver premido, o *led* verde vai estar apagado. Após o *reset,* o *led* verde deverá demorar alguns instantes até ficar novamente a piscar. Caso fique permanentemente aceso, significa que o *reset* não foi devidamente executado, devendo-se voltar a premir o botão.

4.2.4.5 Envio de Sinais PWM da Placa de Controlo para a Placa de Interface de Módulos de Potência

Como os motores são ligados aos Módulos de Potencia, é necessário enviar os sinais de comando da Placa de Controlo para a placa que os aloja. Para este fim existe um terminal, identificado como "Terminal para Módulos de Potência", junto ao conector para o *flat cable,* na **Figura 29**. Entre este terminal e os portos do micro controlador, onde são gerados os sinais de comando, devem ser ligados cabos idênticos ao apresentado na figura seguinte:

Figura 31 – Cabo para efectuar ligação entre portos do micro controlador e Terminal para Módulos de Potência.

4.3 Placa de Interface para Módulos Potência

4.3.1 Listagem de Componentes

Referência	Componente	Fabricante	Quantidade
-	Resistência 560Ω	-	1
-	Condensador Cerâmico 0.33µF	-	1
-	Condensador Cerâmico 0.1µF	-	1
STPS2L60	Díodo Schottky 2A	STMICROELECTRONICS	1
-	Regulador LM7805CT	-	1
-	Led Vermelho	-	1
22-27-2081	Header 8 pinos	MOLEX	4
DC10A	Conector para transformador	CLIFF ELECTRONIC	1
MC9A12-2034	Header 20 pinos para *flat cable*	MULTICOMP	1
1MS3T1B5M1QE	Interruptor 3 posições	MULTICOMP	1
FK 242 SA 220 O	Dissipador de calor para regulador	FISCHER ELEKTRONIK	1
77311-401-36	Régua Simples	FCI	1

Tabela 5 – Listagem de material para a Placa de Controlo de *PWM*.

4.3.2 Descrição

A Placa de Interface para Módulos de Potência permite a ligação dos Módulos de Potência (ver **Secção 4.4**) para controlo de periféricos que funcionem com sinais *PWM*. Esta placa encontra-se ligada à Placa de Controlo através de um *flat cable*.

A alimentação pode ser efectuada recorrendo a um transformador ou a baterias. Note-se que esta placa possui duas tensões de alimentação: 5V para dispositivos existentes nos Módulos de Potência (Opto acopladores e *Bridge*) e os 11,1V para actuação de motores. Existe assim um circuito regulador de tensão que assegura os 5V necessários. Os sinais de comando, necessários para a actuação dos motores, são obtidos da Placa de Controlo através de um *flat cable*.

Esta placa permite a ligação de quatro Módulos de Potência, sendo um deles apenas alimentado a 5V, uma vez que é usado para actuar os Sensores de Proximidade (consultar **Figura 33**).

4.3.3 Esquemas

O esquema eléctrico do circuito é o que se segue:

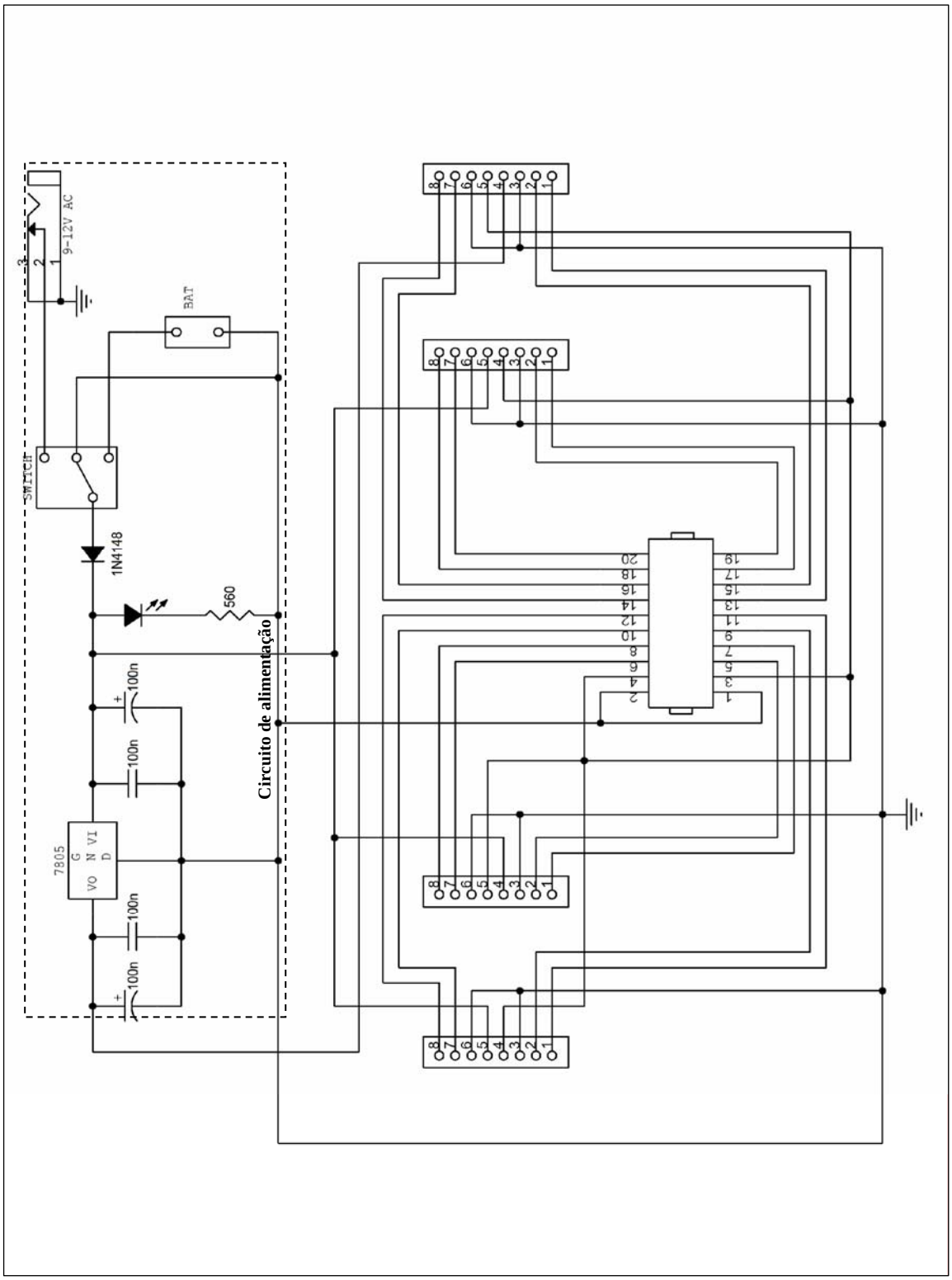


Figura 32 – Esquema eléctrico da Placa de Interface para Módulos de Potência.

A figura que se segue mostra, esquematicamente, a localização dos diversos componentes na placa, a localização dos conectores para os Módulos de Potência e a correspondência entre os portos de saída da Placa de Controlo e os Módulos de Potência.

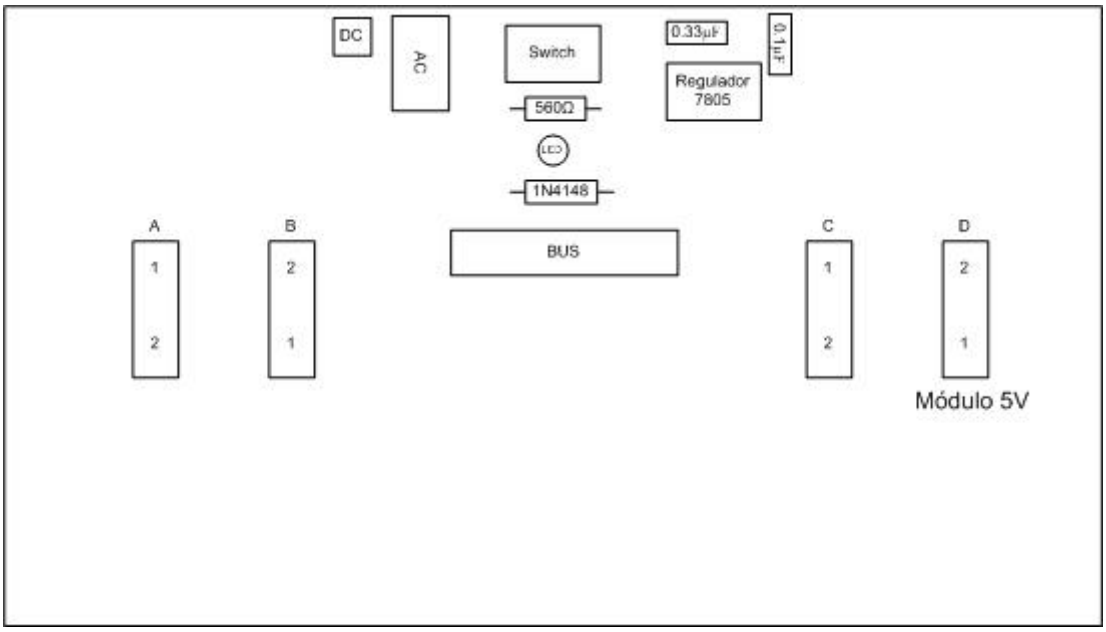


Figura 33 – Mapeamento da Placa de Interface para Módulos de Potência.

As imagens que se seguem são *layouts* de cada uma das faces, necessários para o fabrico desta placa.

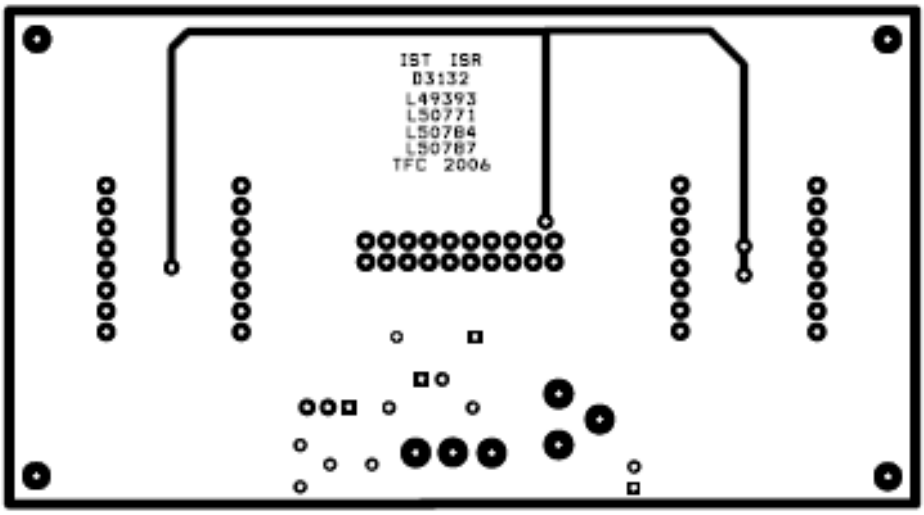


Figura 34 – Face superior da Placa de Interface para Módulos de Potência.

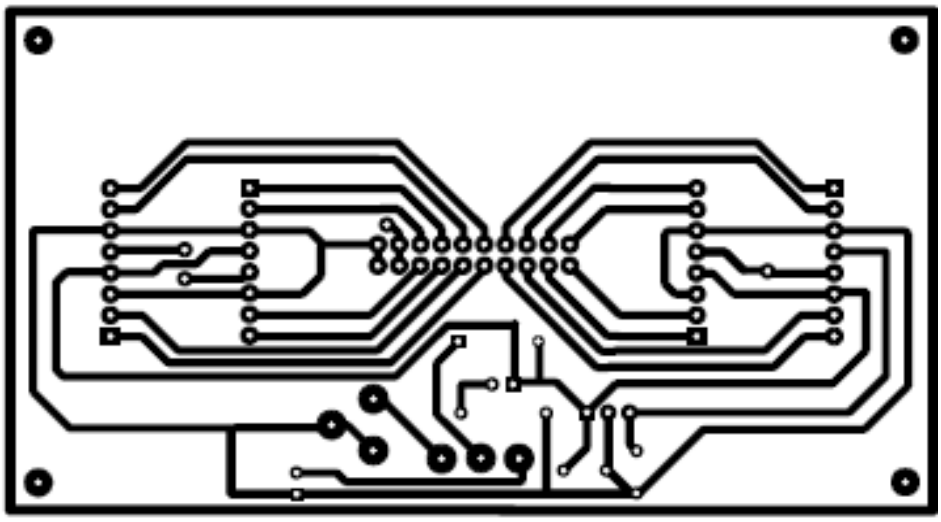


Figura 35 – Face inferior da Placa de Interface para Módulos de Potência.

Note-se que estas imagens se encontram espelhadas e não estão à escala. No final deste manual encontram-se as cópias, em tamanho real, que podem ser usados para o fabrico de novas placas.

4.3.4 Modo de Funcionamento

Na figura seguinte encontram-se assinalados os componentes mais importantes da placa, assim como a localização dos conectores para Módulos de Potência. Note-se que está assinalada na figura qual o conector a utilizar para o módulo com Sensores de Proximidade.

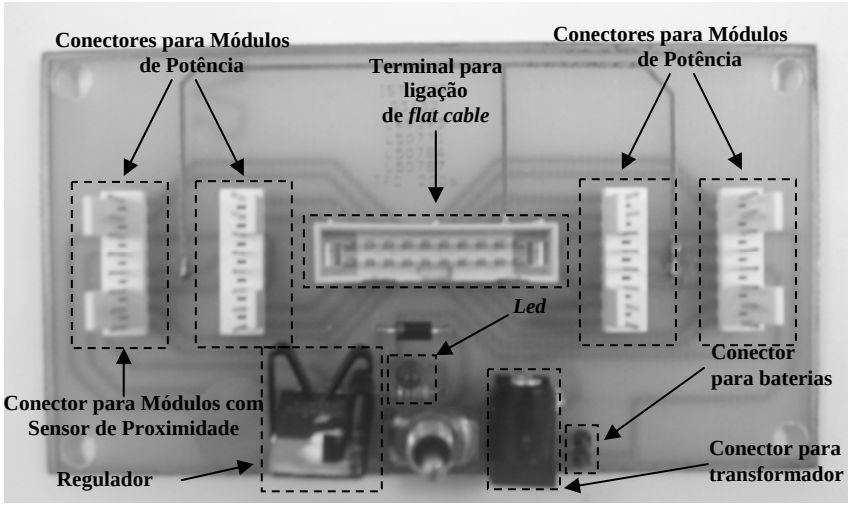


Figura 36 – Placa de Interface para Módulos de Potência.

4.3.4.1 Alimentação

Existem dois terminais diferentes para ligar a alimentação à placa, um para baterias, fontes de alimentação ou pilha de 9V e outro para transformadores. Apesar do interruptor permitir seleccionar a origem da alimentação, deve assegurar-se que apenas um tipo de alimentação está conectado à placa.

Para utilizar a Placa de Interface para Módulos de Potência, é conveniente seguir o seguinte procedimento de ligação:

i. Ligar fonte de alimentação, bateria ou transformador (Tensão máxima 11,1V) à placa;
ii. Ligar o interruptor para um dos lados consoante o tipo de alimentação se esteja a utilizar. Notar que neste instante o *led* vermelho se acende, indicando assim que a placa está correctamente alimentada. Ao colocar o interruptor para o lado do regulador é seleccionada a alimentação por transformador., caso contrário selecciona-se a entrada das baterias (ver Figura 36).

Caso seja utilizada uma fonte de alimentação é conveniente ligá-la antes de conectar o cabo à placa. Este procedimento impede que os picos de tensão iniciais, gerados pela fonte, danifiquem o equipamento.

4.3.4.2 Ligação de Módulos de Potência

Os Módulos de Potência são ligados à Placa de Interface com esta desligada. Devem ser apenas ligados os módulos necessários à aplicação em causa.

Como se pode verificar, as faces dos Módulos de Potência que contém as soldas ficam voltados para o mesmo lado, portanto, recomenda-se a utilização de um isolante entre as placas de modo a evitar curto-circuitos entre as placas.

4.4 Módulos de Potência

4.4.1 Listagem de Componentes

Referência	Componente	Fabricante	Quantidade
-	Resistência 330Ω	-	4
-	Resistência 4.7kΩ	-	4
SN754410NEG4	Bridge	TEXAS INSTRUMENTS	1
4N25	Opto acoplador	FAIRCHILD	4
38-00-1338	Socket 8 pinos para *PCB*	MOLEX	4
816-AG11D-ESL-LF	Socket IC 16 pinos	TYCO ELECTRONICS/AUGAT	1
506-AG11DESL	Socket IC 6 pinos	TYCO ELECTRONICS/AMP	4
77311-401-36	Régua Simples	FCI	1

Tabela 6 – Listagem de material para módulos de potência.

4.4.2 Descrição

Esta placa liga os sensores/actuadores que necessitam de sinais *PWM* para funcionar, nomeadamente motores e sensores de proximidade. Cada Módulo contém dois terminais para ligação de periféricos e permite a ligação de dois motores, ou de quatro sensores de proximidade.

A placa é composta por oito resistências, quatro opto acopladores e uma *bridge*. Os opto acopladores são usados para proteger os portos do PIC de picos de corrente gerados pelos motores. Este tipo de dispositivos efectua o isolamento óptico entre a entrada e a saída. A sua montagem típica exige a utilização de duas resistências, que devem ser projectadas de acordo com a tensão de saída e o tempo de resposta desejado.

A *bridge* é um dispositivo de potência que amplifica os sinais de comando do PIC, com amplitude 5V e os converte para sinais de potência com amplitude 11V (aproximadamente igual à tensão de alimentação da Placa de Interface para Módulos de Potência.

4.4.3 Esquemas

O esquema eléctrico do circuito é apresentado de seguida:

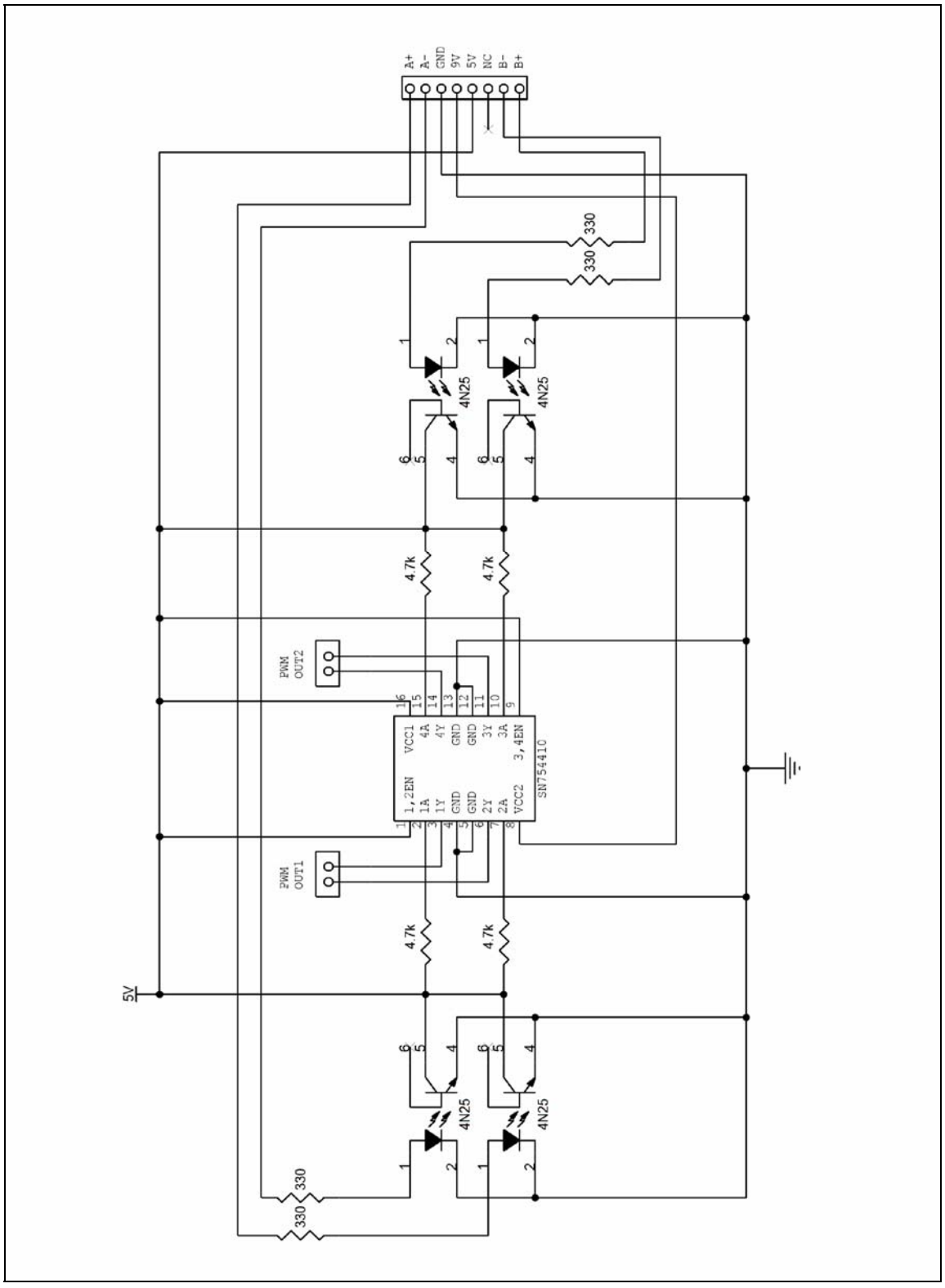


Figura 37 – Esquema eléctrico dos Módulos de Potência.

A figura que se segue mostra, esquematicamente, a localização dos diversos componentes na placa e a localização dos terminais para ligação de periféricos.

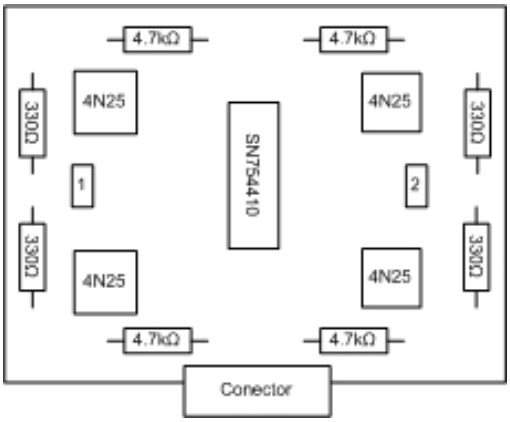


Figura 38 – Mapeamento dos Módulos de Potência.

As imagens que se seguem são *layouts* de cada uma das faces, necessários para o fabrico desta placa.

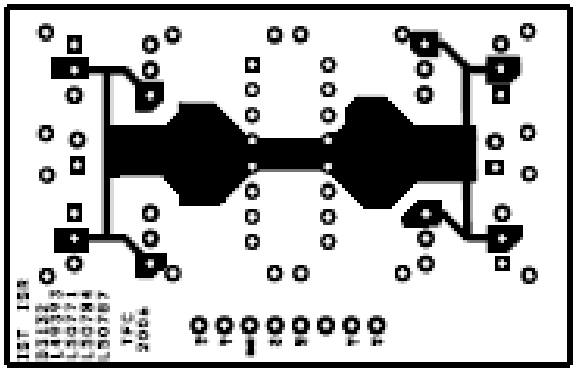


Figura 39 – Face superior do Módulo de Potência.

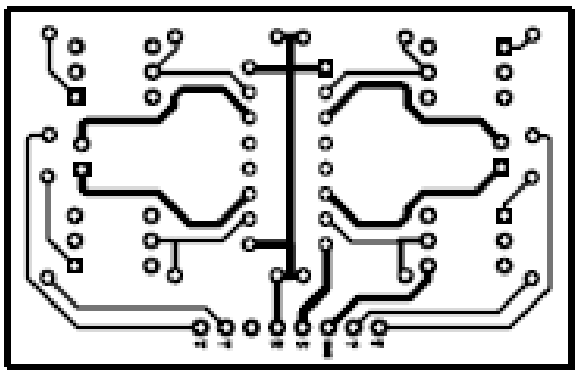


Figura 40 – Face inferior do Módulo de Potência.

Note-se que estas imagens se encontram espelhadas e não estão à escala. No final deste manual encontram-se as cópias, em tamanho real, que podem ser usados para o fabrico de novas placas.

4.4.4 Modo de Funcionamento

Na figura seguinte encontram-se assinalados os componentes mais importantes presentes na placa, assim como a localização dos terminais para ligação de Motores e Sensores de Proximidade.

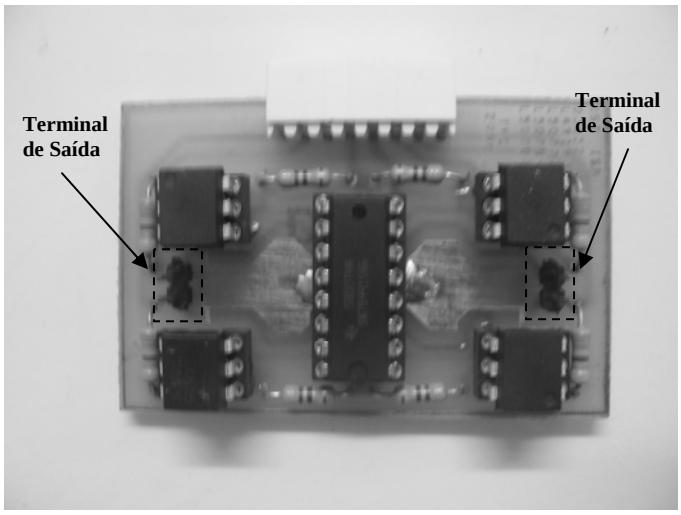


Figura 41 – Módulo de Potência.

4.4.4.1 Ligação de Módulos de Potência

Os Módulos de Potência são ligados à Placa de Interface com esta desligada. Devem ser apenas ligados os módulos necessários à aplicação em causa.

Como se pode verificar, as faces dos Módulos de Potência que contém as soldas ficam voltados para o mesmo lado, portanto, recomenda-se a utilização de um isolante entre as placas de modo a evitar curto-circuitos entre as placas.

4.4.4.2 Ligação de periféricos aos Módulos de Potência

A ligação de motores ao Módulo de Potência é feita com a Placa de Interface desligada. É de notar que a maneira como o cabo do motor é ligado à placa influência o sentido de rotação deste.

4.5 Placa de Acelerómetro

4.5.1 Listagem de Componentes

Referência	Componente	Fabricante	Quantidade
-	Resistência 1.2MΩ	-	1
-	Condensador Cerâmico 0.1µF	-	1
-	Condensador Cerâmico 1µF	-	2
ADXL202	Acelerómetro	ANALOG DEVICES	1
652-0162211E001	IC Socket para SMD	WELLS-CTI	1
77311-401-36	Régua Simples	FCI	1

Tabela 7 – Listagem de material para placa de Acelerómetro.

4.5.2 Descrição

O acelerómetro utilizado nesta placa (*ADXL202*) permite a leitura de acelerações segundo dois eixos perpendiculares entre si. A placa desenvolvida contém a montagem típica, constituída por uma resistência e condensadores, necessários para o funcionamento deste dispositivo. Este sensor é usado numa vasta de aplicações, como na detecção de inclinações.

Na Secção 2.13 podem ser consultadas informações mais detalhadas acerca do seu funcionamento.

4.5.3 Esquemas

O esquema eléctrico do circuito é apresentado de seguida:

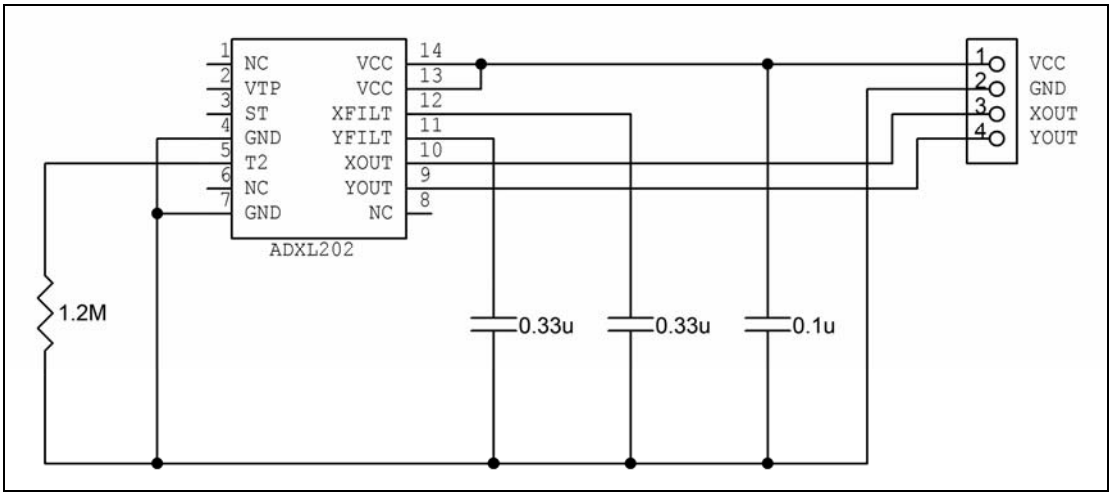


Figura 42 – Esquema eléctrico da Placa de Acelerómetro.

A figura que se segue mostra, esquematicamente, a localização dos diversos componentes na placa e a localização dos terminais de ligação.

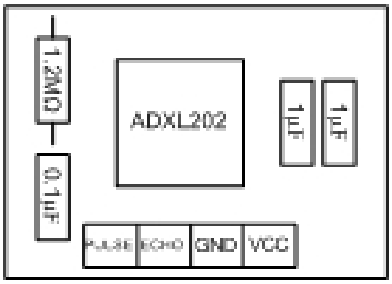


Figura 43 – Mapeamento da Placa de Acelerómetro.

As imagens que se seguem são *layouts* de cada uma das faces, necessários para o fabrico desta placa.

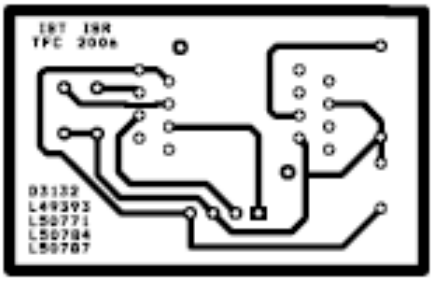


Figura 44 – *Layout* da Placa de Acelerómetro.

Note-se que estas imagens se encontram espelhadas e não estão à escala. No final deste manual encontram-se as cópias, em tamanho real, que podem ser usados para o fabrico de novas placas.

4.5.4 Modo de Funcionamento

Na figura seguinte encontram-se assinalados os componentes mais importantes da placa, assim como a localização dos conectores para ligação à Placa de Controlo.

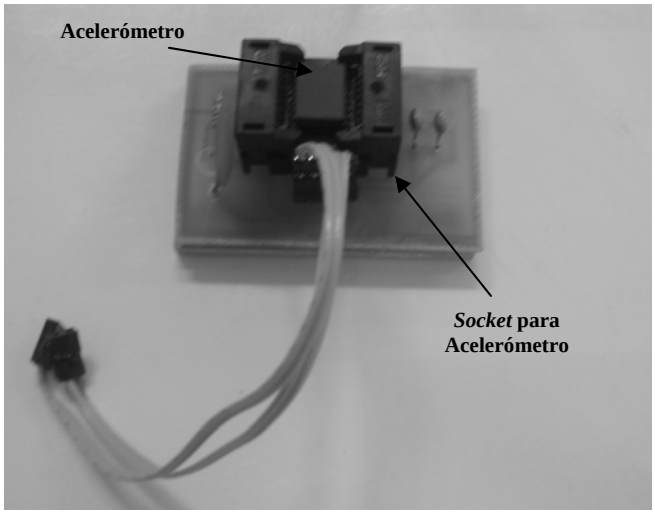


Figura 45 – Placa de Acelerómetro.

A Placa de Acelerómetro é ligada a dois dos seguintes portos da Placa de Controlo: RB0, RB1, RB2, RB4, RB5, RB6 ou RB7 (consultar o mapeamento na **Secção 4.2.3** para determinar a sua localização na placa). É necessária a sua ligação a dois portos, uma vez que o *ADXL202* retorna o valor da aceleração sentida segundo dois eixos independentes.

Para conectar a Placa do Acelerómetro à Placa de Controlo é necessária a utilização de dois cabos semelhantes ao apresentado abaixo:

Figura 46 – Cabo para ligar a Placa de Acelerómetro à Placa de Controlo.

Um dos cabos a utilizar deverá ser ligado ao *Vcc* e *Ground* da Placa de Controlo. O outro deverá ser ligado aos terminais de leitura da Placa de Acelerómetro e aos portos da Placa de Controlo.

Para saber como configurar o acelerómetro em *software* consulte a **Secção 2.13.4**.

4.6 Placa de Sensor de Proximidade

4.6.1 Listagem de Componentes

Referência	Componente	Fabricante	Quantidade
-	Resistência 150Ω	-	1
-	Resistência 100kΩ	-	1
HSDL-9100-021	Sensor de Proximidade	AVAGO TECHNOLOGIES	1

Tabela 8 – Listagem de material para Placa de Sensor de Proximidade.

4.6.2 Descrição

Este sensor permite determinar a existência de obstáculos entre os 0 e os 6 cm de distância. Para isto é usado um *led* (emissor de luz) e um foto-transístor (receptor de luz), que mede a intensidade do reflexo criado pelo *led*. Na Secção 2.12.1 encontra-se explicado o funcionamento deste sensor.

A montagem típica consiste em duas resistências, uma para o *led* e outra para o foto-transístor.

4.6.3 Esquemas

O esquema eléctrico do circuito é apresentado de seguida:

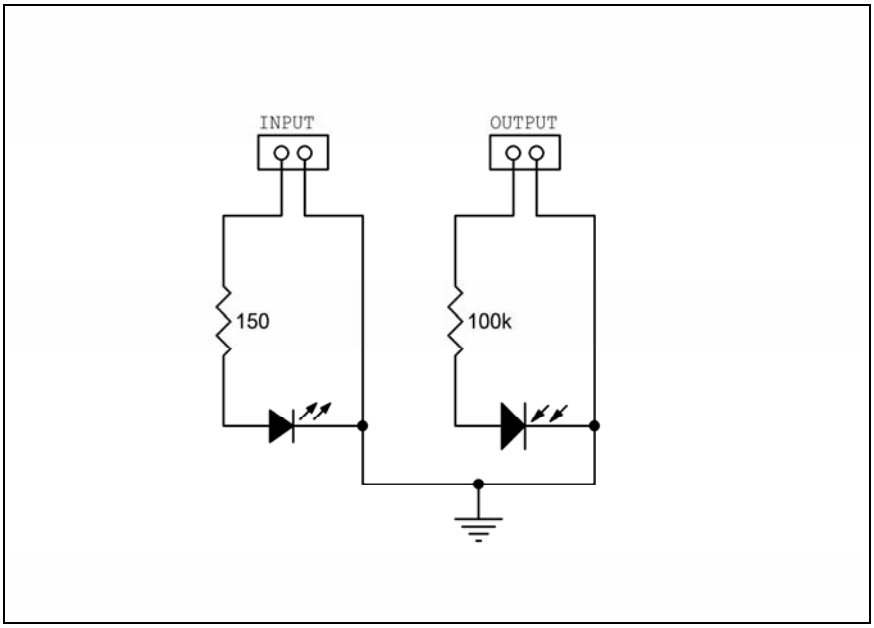


Figura 47 – Esquema eléctrico da Placa do Sensor de Proximidade.

As imagens que se seguem são *layouts* de cada uma das faces, necessários para o fabrico desta placa.

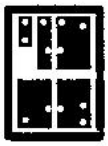


Figura 48 – Layout da Placa do Sensor de Proximidade.

Note-se que estas imagens se encontram espelhadas e não estão à escala. No final deste manual encontram-se as cópias, em tamanho real, que podem ser usados para o fabrico de novas placas.

4.6.4 Modo de Funcionamento

Na figura seguinte encontram-se assinalados os componentes mais importantes da placa.

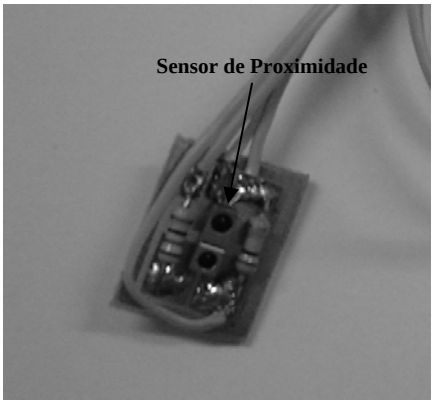


Figura 49 – Placa de Sensor de Proximidade.

Este sensor é ligado à Placa de Controlo e a um dos terminais de saída do Módulo de Potência. A ligação à Placa de Controlo permite a leitura da tensão proporcional à luz reflectida pela superfície mais próxima, enquanto que a ligação ao Módulo de Potência é usada para alimentar o *led* que emite a luz.

A ligação deste sensor ao Módulo de Potência deve ser feita já com este encaixado na Placa de Interface para os Módulos.

Esta placa pode ser ligada a um dos portos RA0, RA1, RA2, RA4, RA5, RE0, RE1, RE2, RB2 e RB3 (consultar o mapeamento na **Secção 4.2.3** para determinar a sua localização na placa). Note-se que estes sensores só podem ser ligados ao módulo que opera a 5V, assinalado na **Figura 33**. A ligação a qualquer um dos outros módulos danificaria o sensor permanentemente.

As figuras que se seguem ilustram a correspondência entre os pinos do "Terminal para Módulos de Potência" existente na Placa de Controlo e os pinos do Módulo de Potência.

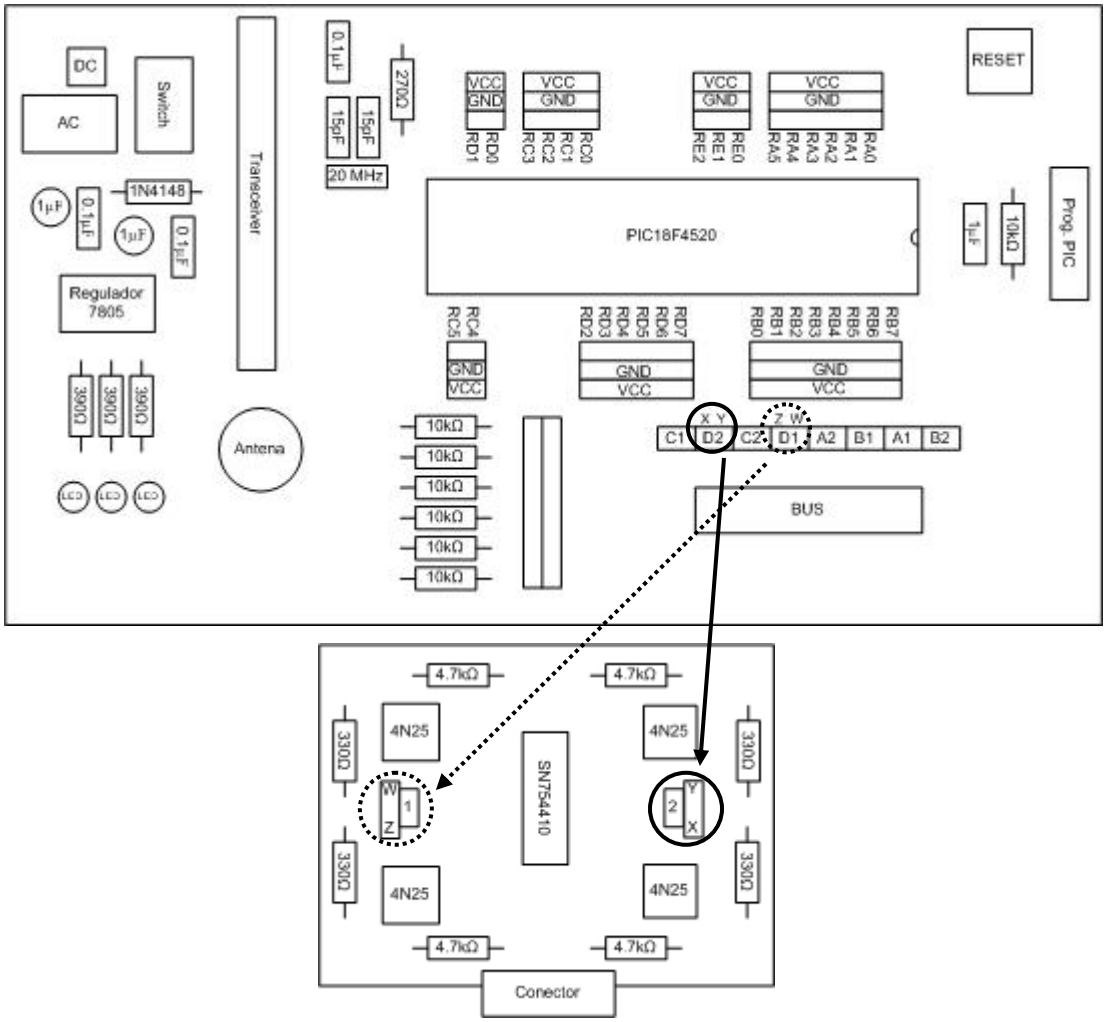


Figura 50 – Ilustração da correspondência entre pinos nas duas placas.

Para configurar os sensores de proximidade em *software* consulte a **Secção 2.12.4**.

5 Ligação de Sensores e Actuadores

Na **Figura 51** está indicada a localização, na Placa de Controlo, dos portos onde se podem ligar os diversos tipos de sensores e actuadores.

Junto à ligação de cada porto do micro controlador, existe um terminal de *Ground* (0V) e um de *Vcc* (5V), que se encontram assinalados na figura seguinte.

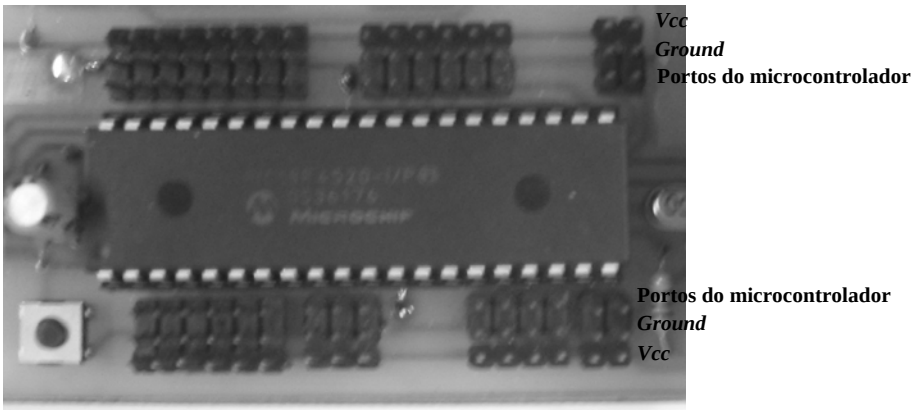


Figura 51 – Localização dos portos, *Ground* e *Vcc* na Placa de Controlo.

Como foi descrito na **Secção 4.2.2** existem diversos tipos de sensores agrupados em diferentes classes. Em seguida é explicado o modo de ligação de cada um destes à Placa de Controlo.

5.1 Motores

Os motores suportados por este kit são motores LEGO ou compatíveis, cujas características eléctricas permitam alimentação a 11,1V e corrente até 500mA. Estes ligam directamente aos terminais de saída dos Módulos de Potência, devendo ter-se em atenção que o sentido de rotação é afectado pela forma como o cabo é conectado à placa.

Note-se que é necessário fazer chegar os sinais gerados na Placa de Controlo à Placa de Interface para Módulos de Potência. Este procedimento está descrito na **Secção 4.2.4.5**.

Para configurar os motores em *software* consulte a **Secção 2.10.2**.

5.2 Sensores de Contacto

Os sensores de contacto são dispositivos binários, i.e., quando estão premidos retornam o valor de tensão *Vcc* e quando estão livres devolvem *Ground*. O seu modo de funcionamento é descrito em detalhe na **Secção 2.14**. Estes sensores pode ser ligados a qualquer um dos portos da Placa de Controlo.

De acordo com o descrito acima, estes dispositivos apenas possuem três ligações, uma para o porto de leitura do micro controlador e as outras duas para a alimentação (5V e 0V).

Para configurar os sensores de contacto em *software* consulte a **Secção 2.14.2**.

Figura 52 – Sensor de Contacto.

5.3 Sensores de Rotação LEGO

Para funcionarem correctamente, estes sensores devem ser conectados aos portos do micro controlador e às resistências de *pull up*. Estes sensores apenas podem ser ligados aos portos: RA0, RA1, RA2, RA4, RA5, RE0, RE1, RE2, RB2 e RB3 (consultar o mapeamento na **Secção 4.2.3** para determinar a sua localização na placa). O esquema de ligação do sensor é o indicado:

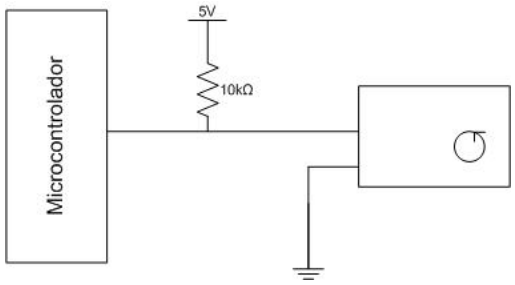


Figura 53 – Esquema de ligação do odómetro LEGO.

A figura seguinte mostra o cabo utilizado para ligar os odómetros LEGO à Placa de Controlo. Note-se que este cabo possui três terminais a conectar. Uma das extremidades do cabo encontra-se assinalada a vermelho e deve ser conectada às resistências de *pull up* e a que contém um conector do tipo macho deve ser ligada ao terminal do odómetro. O terceiro conector é constituído por dois fios, o que se encontra assinalado a preto deve ser conectado a um dos portos do micro controlador e o outro a *ground* (ver **Figura 51** para identificar a localização dos terminais na placa).

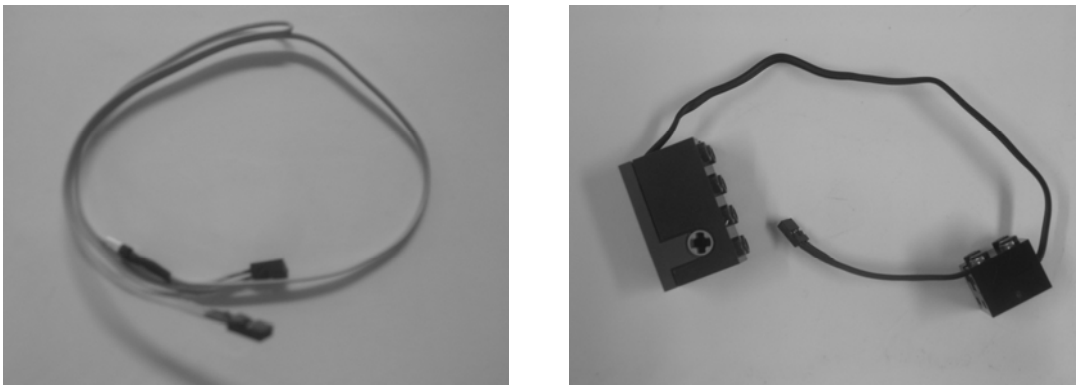


Figura 54 – Cabo para ligação à Placa de Controlo (esquerda) e odómetro Lego (direita).

Para configurar os odómetros LEGO em software consulte a **Secção 2.12.4**.

5.4 Sensores de Rotação de Alta Resolução

Os Sensores de Rotação de Alta Resolução podem ser ligados a qualquer porto do PIC. Existem 4 terminais a ligar, dois para a alimentação (*Ground* e *Vcc*) e dois de leitura (Canais A e B).

Este sensor é constituído por dois dispositivos, um disco de 512 tiras e um *encoder* (*led* e fotodíodos), que são apresentados na **Figura 55**. O *led* emite luz que atravessa o disco. De acordo com a posição das tiras do disco, a luz é ou não captada por um par de fotodíodos. Estes geram níveis lógicos 0 e 1 consoante recebem ou não luz. Na **Figura 56** encontra-se ilustrado o princípio de funcionamento do sensor.

Figura 55 – Disco de 512 tiras (esquerda) e *encoder* (direita).

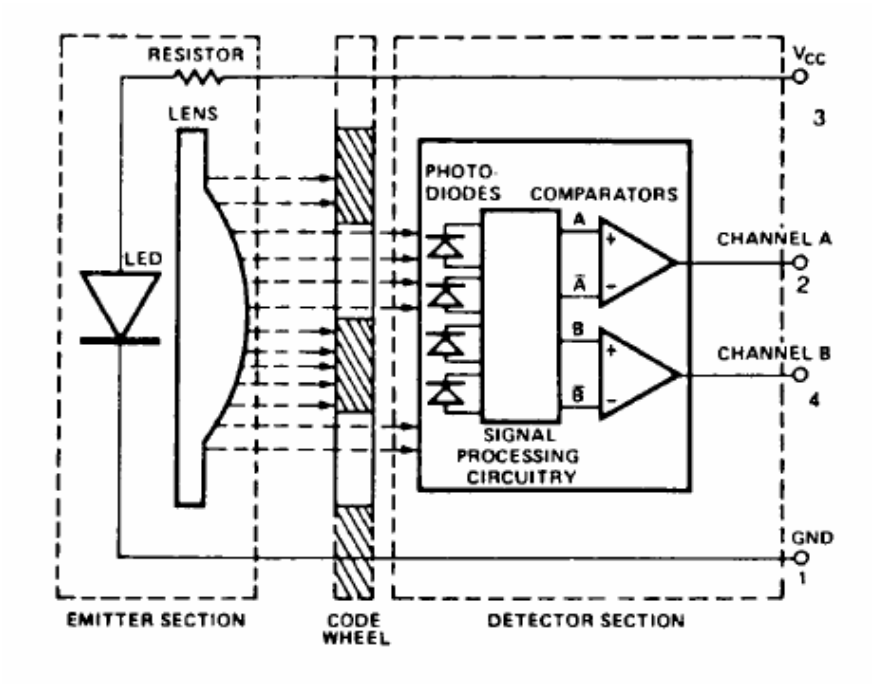


Figura 56 – Esquema do Sensor de Rotação de Alta Resolução (imagem do manual do equipamento).

A montagem do disco ao *encoder* deve ser realizada com cuidado, uma vez que este tem que ficar alinhado correctamente com o *led*. Na **Figura 57** está ilustrada a forma como o disco deve ser acoplado a este.

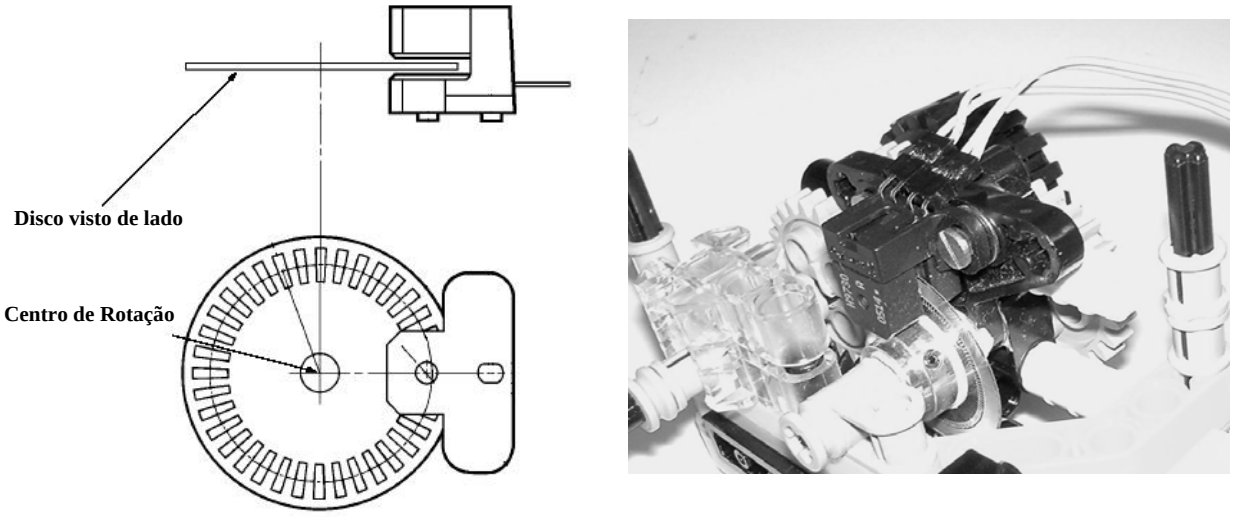


Figura 57 – Acoplagem do disco ao *encoder* (imagem do *datasheet* do dispositivo à esquerda).

Para ligar o sensor à Placa de Controlo é necessário um cabo de quatro fios. A figura seguinte mostra a ordem dos pinos no sensor.

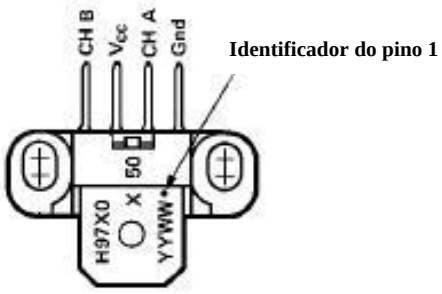


Figura 58 – Esquema de ligação dos cabos ao sensor (imagem do manual do equipamento).

Não é necessário respeitar a ordem de ligação, i.e., os canais podem ser ligados em dois portos não consecutivos.

Para configurar os Sensores de Alta Resolução em *software* consulte a **Secção 2.11.3**.

5.5 Sonares

Os sonares podem ser ligados aos portos RB0, RB1, RB2, RB4, RB5, RB6, RB7 (consultar o mapeamento na **Secção 4.2.3** para determinar a sua localização na placa).

Existem dois modelos de sonares que podem ser utilizados com este kit, o SRF04 e o SRF05. O modo de funcionamento destes sensores encontra-se descrito em detalhe na **Secção 2.13**.

Em seguida encontra-se o esquemático de cada um dos sonares (retirados dos *datasheets*):

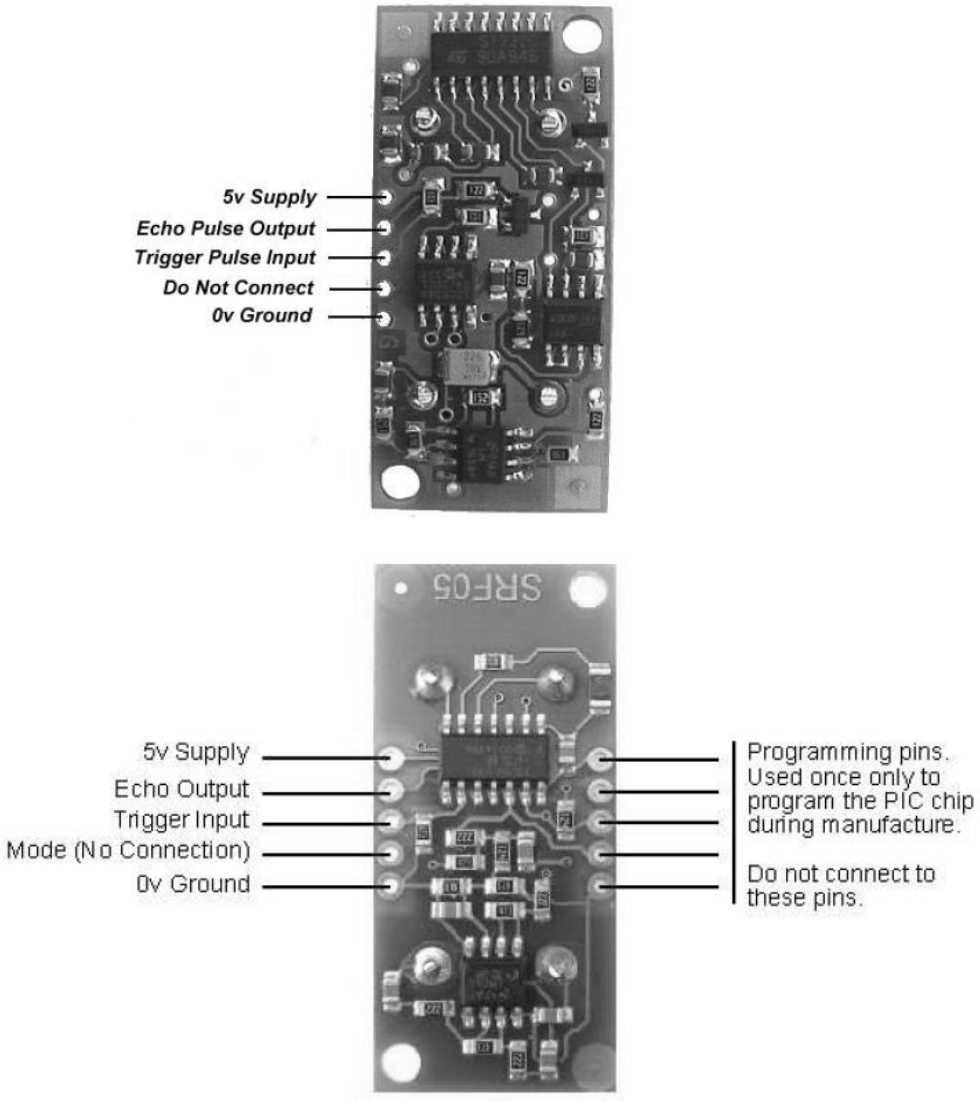


Figura 59 - Esquema de ligações dos sonares (SRF04 em cima e SRF05 em baixo).

O modelo mais recente destes sonares possui um pequeno *led* vermelho que pisca quando o sensor é activado.

Os sonares necessitam ser ligados a dois terminais do micro controlador. Num deles é enviado um sinal para activar o sonar (canal de *pulse*) e noutro é lido o valor mesurado pelo sensor (canal de *echo*). Não é necessário ligar os dois canais a portos consecutivos, não podendo no entanto trocar-se os portos de *pulse* com o de *echo*.

Figura 60 – Sonar e cabo a conectar à Placa de Controlo.

Para conectar um sonar à Placa de Controlo é necessário efectuar quatro ligações, sendo duas delas as descritas acima e as outras as referentes à alimentação. O cabo a conectar a *Vcc* (5V) encontra-se assinalado a vermelho e o cabo a ligar a *Ground* está marcado a azul. O canal de *echo* corresponde ao fio junto a *Vcc* e o de *trigger* encontra-se encostado ao de *Ground*.

Para configurar os sonares em *software* consulte a **Secção 2.13.4**.

6 Exemplo de ligação de vários sensores/actuadores

Nas figuras seguintes encontram-se exemplos de ligação de diversos sensores/actuadores à Placa de Controlo. Tratam-se de duas aplicações diferentes, a primeira consiste num robot que se desloca em cenários interiores genéricos e a outra e a segunda num veículo capaz de se mover em terreno acidentado.

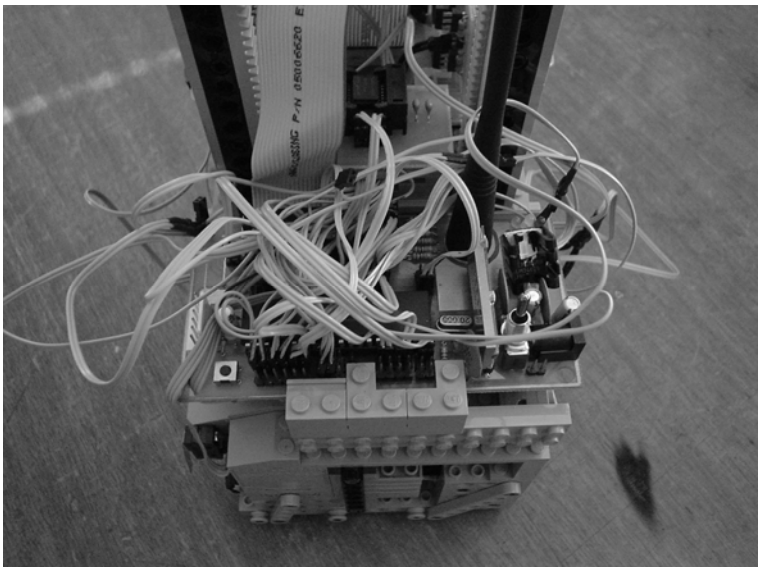


Figura 61 – Exemplo de ligação de sensores/actuadores à Placa de Controlo.

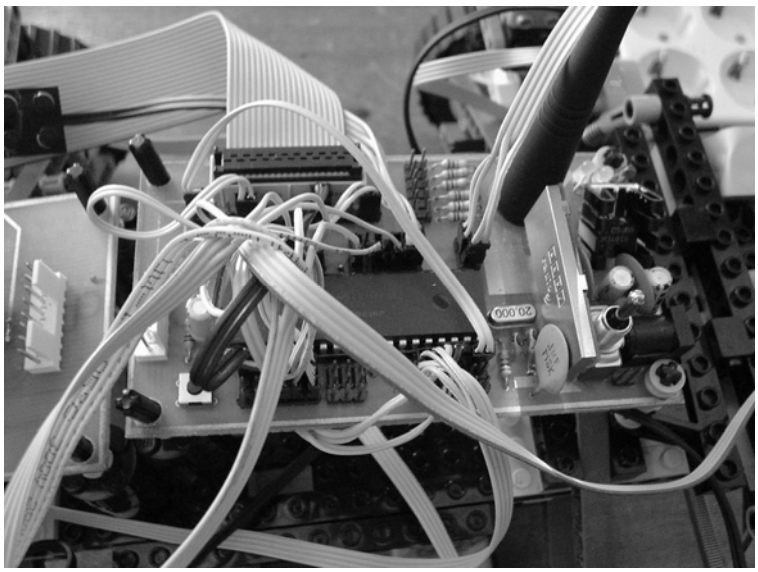


Figura 62 – Exemplo de ligação de sensores/actuadores à Placa de Controlo.

Nas imagens seguintes, mostra-se a utilização da Placa de Interface para Módulos de Potência com os devidos Módulos.

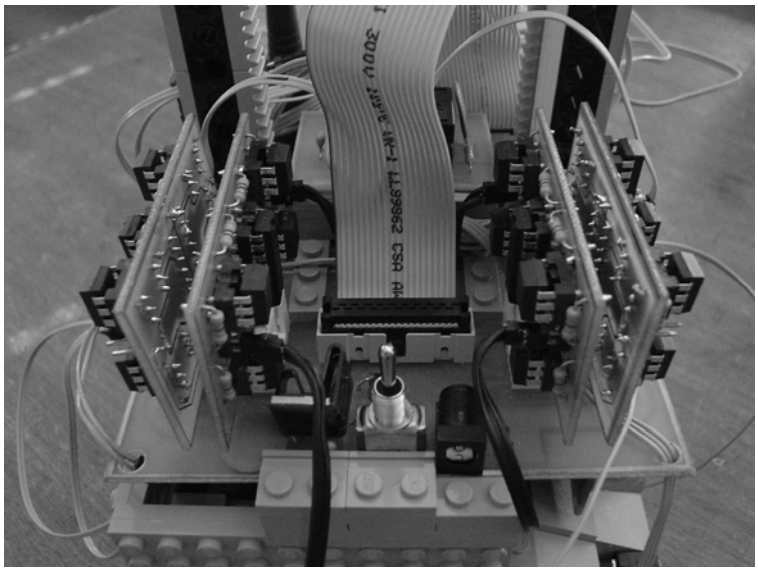


Figura 63 – Exemplo de ligação dos Módulos de Potência à Placa de Interface.

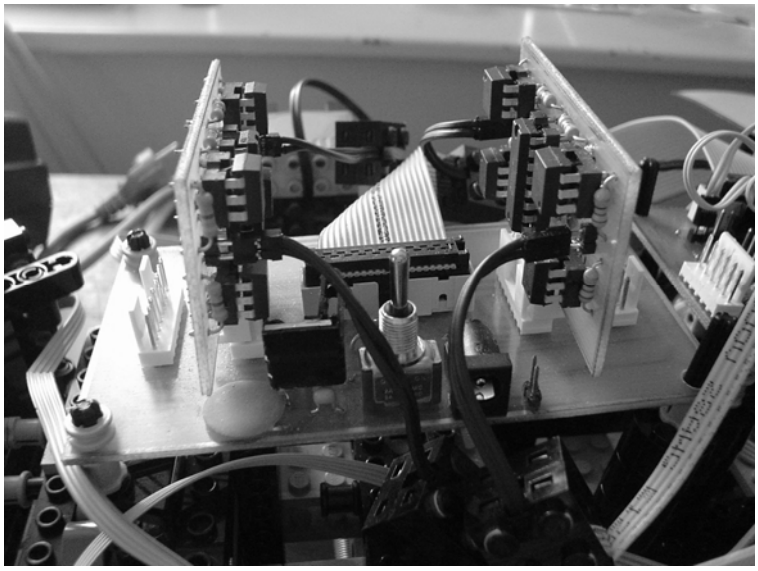


Figura 64 – Exemplo de ligação dos Módulos de Potência à Placa de Interface.

7 Resolução de Problemas

Nesta secção são descritos problemas relacionados com o kit, cujas causas e soluções são conhecidas. De seguida é apresentada uma lista com estas situações, as possíveis causas e os procedimentos a adoptar. Caso surja alguma situação não contemplada nesta lista, relate via e-mail para `jplt@isr.ist.utl.pt`.

Problema	Causa	Procedimento a adoptar
Placa de Controlo não funciona	Sobre aquecimento Falta de alimentação	Verifique as ligações Verifique alimentação Verifique a temperatura do regulador
Não é possível programar o PIC	Cabo não conectado Portos RB6 e RB7 ocupados	Verifique se o cabo de programação está correctamente conectado à Placa Desligue o cabo conectado aos pinos 39 e 40 do micro controlador
Reset ao micro controlador falhou	-	Volte a premir o botão até o *led* verde começar a piscar
Sonar/Acelerómetro não funciona	-	Verifique as ligações Verifique estado do cabo

Tabela 9 – Lista de problemas relacionados com o kit.

Anexo

Esquemáticos PCB do Hardware