Concepção e Desenvolvimento de Unidades de Percepção e Controlo para um Robot Humanóide

Relatório Final

Luís Gomes Nº19654
Mauro Silva Nº23034

Orientação:

Prof. Dr. Vítor Santos

Universidade de Aveiro

Departamento de Engenharia Mecanica

Julho de 2005

Nesta versão falta o código dos microcontroladores

Resumo

Neste projecto desenvolveu-se módulos dedicados de controlo e percepção para uma plataforma robótica humanóide dando continuidade aos trabalhos já desenvolvidos no ano anterior. Em termos de software completou-se o código de controlo dos microcontroladores utilizados e desenvolveram-se variadas aplicações de alto nível para controlar o robô. As placas de circuito impresso de controlo e de aquisição de valores sensoriais foram desenvolvidas e colocadas em funcionamento.

1. Enquadramento e Objectivos

Este trabalho faz parte de um plano mais alargado para o desenvolvimento de uma plataforma robótica humanóide que permita fazer investigação na área, bem como o de fornecer oportunidade de trabalhos aliciantes a estudantes prégraduados em Automação e Robótica.
Em rigor, a ideia original do projecto surgiu há mais de dois anos, mas foi desde o ano lectivo de 2003-2004 que se desenvolveram trabalhos significativos no âmbito de dois projectos de alunos de fim de curso com resultados que encorajam e muito contribuem para o desenvolvimento a curto prazo de uma plataforma de trabalho. Nesta componente agora proposta pretende-se, além de ultimar e interligar os sistemas já desenvolvidos, implementar sistemas adicionais na área da percepção como um pé sensível à força e a percepção por visão. Os objectivos propostos para este ano lectivo foram os seguintes:

1. **Desenvolvimento de um pé sensível à força**
2. **Implementação de controlo de velocidade em servomotores de posição**
3. **Integração com sistema de comunicações CAN**
4. **Sistema de visão monocular**
5. **Unidade central de controlo**

No ano anterior desenvolveu-se a unidade de comunicações entre módulos. O sistema já era expansível a qualquer número de módulos. Diversas experiências e circuitos encontravam-se desenvolvidos para a implementação e medição de forças no sistema (no pé, em particular).

2. Arquitectura distribuída

A diversidade de sub-componentes envolvidos num projecto desta natureza requer uma organização que permita o desenvolvimento modular e contínuo do sistema. Há essencialmente dois tipos de arquitecturas possíveis pelas quais se pode optar.
A primeira é a arquitectura centralizada, em que um módulo central se encarrega do processamento de todos os dados adquiridos pelos diversos sensores instalados a bordo e da tomada de decisão em função dos resultados obtidos.
Por outro lado, numa arquitectura distribuída existem diversos módulos que processam localmente os dados adquiridos pelos seus sensores e que transmitem a informação resultante a uma unidade central para a tomada de decisão; esta última envia os comandos necessários às diversas unidades para que alterem a configuração dos actuadores de forma a que seja atingindo o estado desejado. Nesta abordagem tem se por assim dizer vários coprocessadores que possuem tarefas específicas sendo estas executadas em paralelo sem que a unidade principal de processamento tenha que dispensar tempo de processamento para efectuar o controlo a baixo nível.

A plataforma tem vários graus de liberdade (22 na versão actual) e diferentes grupos de sensores: um sensor de visão (CCD), um potenciómetro de posição por cada actuador, unidades de monitorização de corrente nos motores, giroscópios para medir a velocidade angular em certos pontos do corpo, inclinómetros, e vários extensómetros para as medições de força nos pés. A arquitectura distribuída apresenta-se a mais

indicada dado que permite distribuir as tarefas por diversos módulos para que possuam algoritmos específicos para as acções que lhe estão atribuídas, rumo a uma simplificação do sistema de controlo.
As tarefas podem ser ligeiramente diferentes de módulo para módulo, no entanto pretende-se que o *hardware* que constitui estes módulos seja idêntico. Implementando esta estratégia consegue-se um grau de fiabilidade superior, uma vez que os módulos são independentes permitindo que as anomalias sejam mais facilmente detectadas e corrigidas. Os módulos podem ser trocados facilmente e adaptados à tarefa necessária, pois bastará programar o algoritmo específico à tarefa.
Em resumo, apresentam-se as vantagens da arquitectura distribuída:

- Sistemas fiáveis (operação independente)
- Sistemas de controlo mais simples
- Mais fácil detecção de anomalias
- Actualização fácil de *firmware*

O sistema de controlo implementado é constituído por três unidades diferentes ligadas em rede:

- A **unidade central** de controlo é responsável pelo cálculo das configurações que as juntas tem de adoptar em função dos valores dos sensores e gestão global dos procedimentos;
- A **unidade mestre** tem como principal tarefa servir de interface entre os restantes controladores da rede e a unidade principal de controlo;
- As **unidades escravo,** cujas principais funções são a geração da onda de pulso modulado (PWM) de controlo dos servomotores e a aquisição dos sinais dos diversos sensores da plataforma. Nos escravos espera-se, numa fase mais adiantada do projecto, desenvolver algum poder de controlo local para determinadas situações.

A unidade central de controlo ainda não está completamente definida, permanecendo em aberto soluções baseadas num PDA, placas de controlo genéricas (como as baseadas no padrão PC104) ou placas de controlo dedicadas.

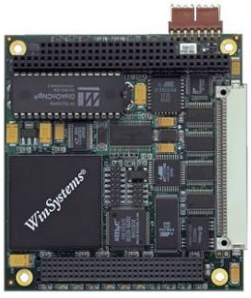


Figura 1 - Exemplo de uma board PC-104

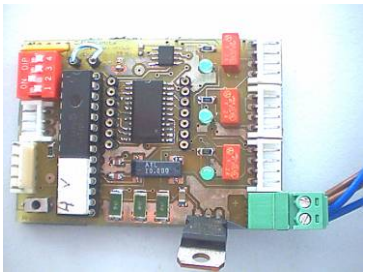


Figura 2 – Placa de controlo desenvolvida

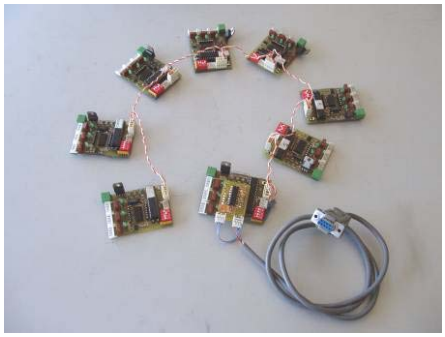


Figura 3 – Rede completa de Microcontroladores

Por enquanto é utilizado um PC externo para enviar e receber dados por uma linha série para o controlador mestre.

Como unidade de controlo local (mestre e escravos), a escolha recaiu sobre os microcontroladores PIC da série 18 da MICROCHIP: possuem variados periféricos e interfaces para redes de comunicações, incluindo o CAN (*Controller Area Network*).
Até ao momento, a rede implementada é constituída por nove placas controladoras, uma delas é a placa mestre as restantes são as placas escravo que efectuam o controlo local. As placas desenvolvidas permitem o controlo máximo de três actuadores através da geração de uma onda de pulso modulado em largura (PWM), e a aquisição de 16 sinais analógicos usando um *multiplexer*.
Esta organização tem como objectivo agrupar as juntas que estão directamente relacionadas, como é o caso das juntas das ancas ou as juntas do tornozelo e do joelho. No caso dos controladores MCU4 e MCU5 é importante ter este agrupamento dado que facilita a implementação do controlo em malha fechada que vai permitir o equilíbrio do robô. Estes são os controladores que adquirem os sinais analógicos dos sensores de força instalados nos pés e que controlam as duas juntas do tornozelo e a do joelho. Numa evolução futura direccionada para a adaptação à irregularidade do solo, estas juntas poderão corrigir a sua posição de forma reactiva para que a projecção do centro de massa do robô não caía fora da área de apoio dos pés. Obter-se-á assim um controlo localizado independente do resto do sistema sem que haja necessidade permanente de interagir com a unidade central de controlo.
Os sinais analógicos adquiridos actualmente, para além dos sensores de força, são os sinais provenientes dos potenciómetros de posição de cada actuador. Estes valores são convertidos e registados localmente e depois enviados via CAN para o controlador mestre. Os escravos estão preparados também para receber mensagens via CAN. Estas mensagens consistem basicamente nas posições que os actuadores têm que tomar e a velocidade a que eles têm que se mover. O controlador mestre tem a tarefa de receber a informação enviada pelos escravos via CAN e registá-la de forma a que esteja disponível para ser enviada para a unidade central de controlo quando solicitada. O controlador mestre mantém uma representação do estado actual das juntas (actuadores e sensores associados) que disponibilizará ao controlo central quando este o pedir.
O processo é bidireccional e o controlador mestre também recebe as ordens da unidade central e despacha para o controlador escravo respectivo.
Como já foi referido anteriormente, recorreu-se ao microcontrolador 18F258 da MICROCHIP como unidade de processamento. Este microcontrolador permite atribuir diferentes prioridades aos *interrupts*; esta característica é fundamental para a geração do PWM que controla os servomotores. Como ambiente de desenvolvimento utilizou-se o MPLab IDE 6.62 com o compilador da linguagem C MCC18 da MICROCHIP.

2.1 Comunicações

2.1.1 O CAN – Controller Area Network

O CAN é um sistema de comunicações série que foi desenvolvido pela indústria automóvel para possibilitar as comunicações de diversos componentes em ambientes extremamente ruidosos. O sinal que serve de suporte à comunicação é definido em corrente e não em tensão. É um protocolo baseado na mensagem e não no endereço, o que significa que as mensagens não são transmitidas entre nós com base no seu endereço; na própria mensagem está incluída a prioridade e os dados a serem transmitidos, a uma velocidade máxima de 1Mbit/s. A mensagem é escutada por todos os nós na rede e cabe a cada nó realizar a análise da mensagem aceitando-a ou não para

posterior processamento. A mensagem pode ser enviada para um ou mais nós dependendo da forma como a rede foi desenvolvida. Outro benefício do protocolo baseado na mensagem é o facto de se poderem adicionar outros nós à rede sem haver necessidade de reprogramar todos os nós existentes. O novo nó adicionado com base na mensagem recebida decide se deve ou não processar a sua informação. O comprimento da mensagem é fixo, sendo possível enviar 8 *bytes* de informação no campo Dados da mensagem.

A escolha do sistema de comunicações recaiu sobre o CAN principalmente devido ao facto de este ser um protocolo baseado na mensagem. Desta forma, futuramente, a construção da rede pode ser modificada para que os controladores locais possam "ouvir" a informação uns dos outros tomando decisões (*i.e.*, dotado de poder local).

2.1.2 Comunicações internas

O controlador Mestre recebe mensagens enviadas da Unidade Principal de Controlo através de uma comunicação Rs-232 com um comprimento fixo de 3 *bytes*:

Definições actuais para a comunicação Série Rs232:

Bits por segundo:	19200
Bits de dados:	8
Paridade:	*None*
Bits de Paragem:	1
Controlo do fluxo:	*None*

O primeiro indica o tipo de mensagem que, se for um comando de movimento para o sistema, leva a que o 2º e 3º *byte* sejam, respectivamente, a velocidade e a posição desejada da junta. O primeiro *byte* pode ter outros significados, como o pedido de leituras dos sensores.

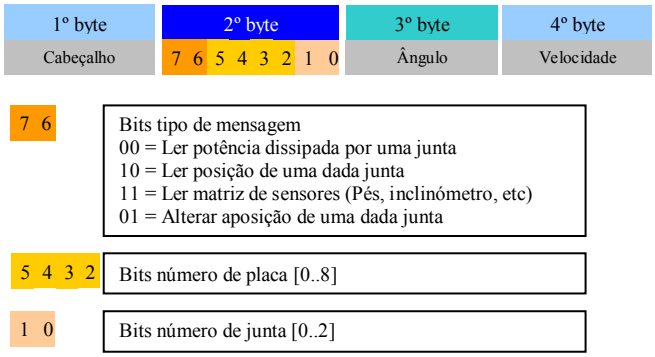


Figura 4 – Tipos de mensagem PC-Master0

Exemplo:

-Para movimentar a junta número 1 do primeiro Slave da rede controlo para a posição 90 a uma velocidade média temos que enviar a seguinte mensagem:

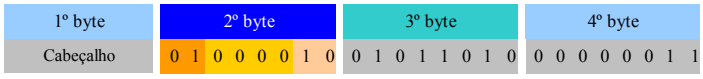


Figura 5 – Exemplo de uma mensagem

Para os diferentes tipos de comandos o Master responde com diferentes tipos de mensagens. Todas elas têm um comprimento fixo de 6 bytes. O primeiro é um byte de sincronismo necessário para fiabilizar as comunicações a taxas elevadas. O segundo byte identifica o tipo de mensagem. A construção deste bit é efectuada de forma idêntica à apresentada anteriormente.
Apresenta-se a seguir os tipos de mensagens que o Master devolve.

	1º byte	2º byte	3º byte	4º byte	5º byte	6º byte
1	Cabeçalho	Identificador	0	0	0	Valor
2	Cabeçalho	Identificador	0	0	0	Ângulo
3	Cabecalho	Identificador	Sensor 1	Sensor 2	Sensor 3	Sensor 4

Tipos de mensagens:

1. Ler potência dissipada por uma junta
2. Ler posição de uma dada junta
3. Ler matriz de sensores

Figura 6 - Tipos de mensagem Master-PC

As mensagens enviadas do mestre para os escravos têm um comprimento de seis *bytes* embebidos no campo de Dados da trama do CAN. Esses seis *bytes* enviam a velocidade e a posição para cada um os três servomotores ligados ao escravo.

Mensagem enviada pelo CAN do Master para o Slave							
1ºByte	2ºByte	3ºByte	4ºByte	5ºByte	6ºByte	7ºByte	8ºByte
Posição Servo 1	Posição Servo 2	Posição Servo 3	Velocidade Servo 1	Velocidade Servo 2	Velocidade Servo 3	X	X

Por outro lado, as mensagens enviadas dos escravos para o mestre têm um comprimento 8 *bytes* e transmitem a identificação do emissor, as posições das três juntas e as restantes leituras sensoriais (*e.g.*, força de reacção nos pés ou outras).

1.ª Mensagem enviada pelo CAN do Slave para o Master							
1ºByte	2ºByte	3ºByte	4ºByte	5ºByte	6ºByte	7ºByte	8ºByte
HostMcu	Identificador de Mensagem	ValSens[0]	ValSens[1]	ValSens[2]	ValSens[3]	ValSens[4]	ValSens[5]

2.ª Mensagem enviada pelo CAN do Slave para o Master							
1ºByte	2ºByte	3ºByte	4ºByte	5ºByte	6ºByte	7ºByte	8ºByte
HostMcu	Identificador de Mensagem	ValSens[6]	ValSens[7]	ValSens[8]	ValSens[9]	X	X

O Master envia mensagens aos Slaves a uma frequência de 10kHz. As matrizes NovPosServ e NovVelServ são varridas ciclicamente. Sempre que tenha havido uma alteração nessas matrizes esses novos valores são enviados ao respectivo Slave.
Sempre que um Slave recebe uma mensagem pelo CAN que seja destinada para ele, este responde com estas duas mensagens. A primeira mensagem que ele envia está relacionada com valores sensoriais dos motores. Após ter sido enviada a primeira mensagem com sucesso é enviada a segunda que contém os valores sensoriais dos sensores adicionais de cada placa.

3. Controlo baixo nível

3.1 Documentação do Código dos microcontroladores Escravos

3.1.1 Enumeração e explicação de todas as funções

	Funções contidas neste ficheiro:	Descrição da função:
		
	void InitCan (byte)	Configura o módulo de comunicações por CAN
		
	void InitPic (unsigned long, unsigned long, int)	Inicializa e configura todos o periféricos necessários para o funcionamento desejado do Microcontrolador.
		
	void CANInitialize (byte SJW, byte BRP, byte PHSEG1, byte PHSEG2, byte PROPSEG, enum CAN_CONFIG_FLAGS flags)	Inicializa o módulo de comunicações por CAN
	void CANSetOperationMode (CAN_OP_MODE mode)	Configura o Microcontrolador no modo de operações desejado.
	void CANSetBaudRate (byte SJW, byte BRP, byte PHSEG1, byte PHSEG2, byte PROPSEG, enum CAN_CONFIG_FLAGS flags)	Configura o baud rate do CAN

	void CANSetMask (enum CAN_MASK code, unsigned long val, enum CAN_CONFIG_FLAGS type)	Configura as mascaras de recepção de mensagens por CAN.
	void CANSetFilter (enum CAN_FILTER code, unsigned long val, enum CAN_CONFIG type)	Configura os filtros dos buffers de recepção do CAN.
	BOOL CANSendMessage (unsigned long id, byte *Data, byte DataLen, enum CAN_TX_MSG_FLAGS MsgFlags)	Trata do envio de mensagens por CAN.
	BOOL CANReceiveMessage (unsigned long *id, byte *Data, byte *DataLen, enum CAN_RX_MSG_FLAGS *MsgFlags)	Trata da recepção de mensagens via CAN.
		
	void ServoIsr(void)	Rotina de serviço dos interrupts de alta prioridade.
	void ComIsr(void)	Rotina de serviço dos interrupts de baixa prioridade.
	void EnviaValSens (byte, byte, byte *)	Função para montagem e envio de mensagens por CAN.
	void Get_dis (byte *)	Função para determinar a distribução da força nos pés.
	void Contr_Velo(void)	Função que implementa o controlo de força dos servomotores.
		
	void EnvCh (byte *, byte);	Função para montagem e envio de mensagens por comunicação série.

Antes de se passar à explicação da rotina principal (void main(void)) deste programa são discutidas as funções InitCan e InitPic. Esta funções são evocadas uma única vês na rotina principal e são fundamentais para configurar funcionamento do microcontrolador.

void InitCan(HostMcu)

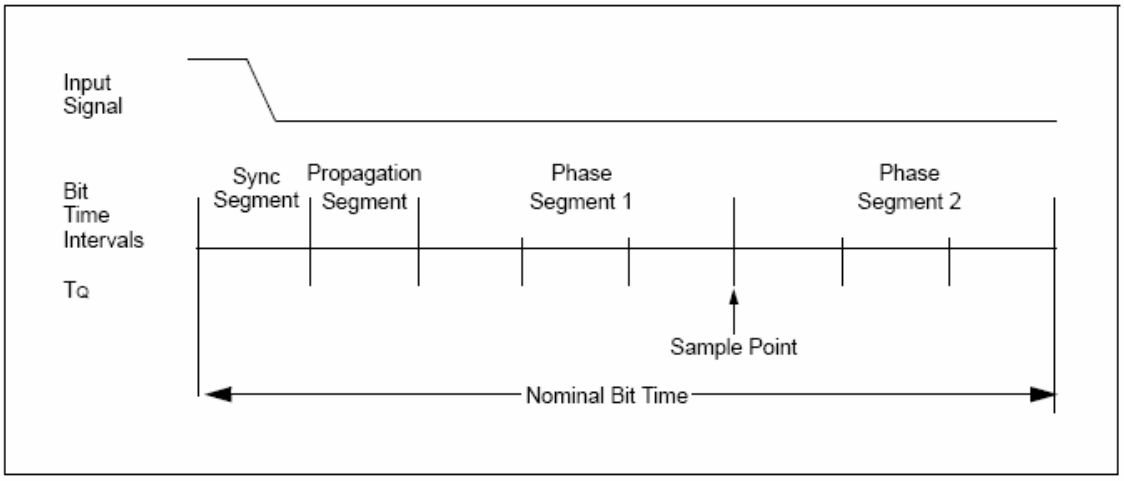
Como o próprio nome indica esta função inicializa o protocolo de comunicações CAN.

Todos os nós numa rede de comunicações CAN têm de ter uma taxa de transmissão de bits idêntica(*nominal-bit-rate*). O protocolo CAN utiliza uma codificação que não retorna ao valor zero (*Non-Returnto-Zero*) o que não possibilita a codificação de um relógio de sincronismo embebido nos dados a transmitir. Assim sendo um nó que esteja a receber uma mensagem pelo CAN tem ter a capacidade de sincronizar o seu relógio de recepção com o relógio do nó que enviou a mensagem.
O que se utiliza para poder garantir este sincronismo de transmissão de dados é aquilo a que se chama um *Phase-Lock-Loop* (PLL), existe a possibilidade de se ter microcontroladores numa rede CAN a comunicar entre si com osciladores de frequências diferentes. Basta garantir-se que se tenha o chamado *nominal-bit-rate* idêntico. São apresentados a seguir os cálculos necessários para definir o *nominal-bit-rate* em função da frequência do oscilador que estamos a utilizar.
Neste trabalho está se a utilizar microcontroladores a funcionar todos à mesma frequência o que facilita a configuração do CAN dado que será a mesma para todos os nós. As propriedades que temos de configurar será o *prescaler* do *baud rate* e numero de segmentos infinitesimais de cada fase de sincronização (Tq)
O Nominal Bit Rate é o numero de bits transmitidos por segundo. A taxa máxima está definida como sendo de 1 Mb/s. O Nominal *Bit Time* é definido como sendo T_{BIT} =1/*Nominal-Bit-Rate*. O *bit time* é por assim dizer a divisão do bit em diferentes segmentos de tempo:

- Segmento de sincronização (*Sync_Seg*)
- Segmento de propagação (*Prop_Seg*)
- Segmento da Fase de Buffer 1 (*Phase_Seg1*)
- Segmento da Fase de Buffer 2 (*Phase_Seg2*)

Estes segmentos são ainda constituídos por unidades de tempo mais pequenas designadas por *Time Quanta* (T_Q).

FIGURE 19-7: BIT TIME PARTITIONING



Por imposição o *bit time* é programável 8 T_Q até 25 T_Q. Também por imposição o nominal *bit time* tem um valor mínimo de 1µ (corresponde à taxa máxima do CAN de um 1 Mb/s. O *Nominal Bit Time* é calculado da seguinte forma:

Nominal Bit Time = T_Q*(Sync_Seg+ Prop_Seg+Phase_Seg1+ Phase_Seg2)

A *Time Quanta* (T_Q) é calculada da seguinte forma:

TQ (µs) = (2 * (BRP + 1))/FOSC (MHz)
ou
TQ (µs) = (2 * (BRP + 1)) * TOSC (µs)

Onde FOSC é a frequência do oscilador a utilizar, TOSC é o respectivo período e BRP é um inteiro (a até 63) que representa o valor binário de BRGCON<5:0>.
Como é possível ver o TQ depende da frequência do oscilador, este ano efectou-se a alteração da frequência dos oscilação dos relógios dos CPUs de 20 MHz para 40 MHz. Em resultado disto o CAN deixou de funcionar uma vez que com a configuração que vinha do ano anterior *Nominal Bit Time* ficou com um valor inferior a 1 Mb/s.
São apresentados a seguir os cálculos efectuados para reconfigurar o módulo de comunicações CAN.

Tem-se uma frequência de oscilador de 40 MHz e quer-se um Nominal Bit Rate de 1 Mb/s)

TQ (µs) = (2 * (BRP + 1))/FOSC (MHz)
TBIT (µs) = TQ (µs) * numero de T_Q por intervalo
Nominal Bit Rate (bits/s) = 1/TBIT

TBIT (µs) = TQ (µs) * numero de T_Q por intervalo ⇔
TQ (µs) = TBIT (µs) / numero de T_Q por intervalo⇔
TQ (µs) = 1 (µs) / 8 ⇔
TQ (µs) = 0.125 ⇔

TQ (µs) = (2 * (BRP + 1))/FOSC (MHz)
BRP =((TQ (µs) * FOSC (MHz))/ 2)-1
BRP =((0.125 (µs) * 40 (MHz))/ 2)-1
BRP =1.5≈ 2

Resumindo definiu-se que cada segmento de sincronização tem dois segmentos de Time Quanta e dai resulta que se necessite de um *baud rate* de 2. Efectivamente optou-se por utilizar um BRP igual a 3 no que resulta numa taxa de comunicação do CAN ligeiramente inferior a 1 Mb/s.

```
#define FASE_SALTO 2
#define FASE_PROP 2
#define FASE_SEG1 2
#define FASE_SEG2 2
#define BRP 3
```

void InitPic(FREQ_OSC, BAUD)

Esta função configura todas as funcionalidades e periféricos do *pic* necessários para o realizar as tarefas pretendidas. Após esta função ter corrido o microcontrolador entra no funcionamento que se pretende para este trabalho.

Configuração dos Ports de Entradas/Saidas do Microcontrolador

```
TRISA = 0b00000001;        //PortoA <1:7> sa¡das exepto RA0

TRISB = 0b00001011;        //PortoB <0:2> entradas(INT0, INT1, CANRX), <3:7> saidas

TRISC = 0b10001111;        //PortoC <0:6> saidas, RC7 entrada(RX USART)
```

Nos registos TRIS (…) são registos de configuração que definem quais os canais dos *Ports* são entrada ou saídas. O *Port* A tem um único canal de entrada, mais à frente este canal será configurado para efectuar as aquisições de sinais analógicos. Os restantes canais deste *Port* estão definidos como saídas digitais. Quatro destas saídas (Ra1, Ra2, Ra3 e Ra5) são utilizadas para efectuar a comutação dos canais do multiplexer.

Configuração do Timer0 e do Timer1

Estes dois *timers* são utilizados para gerar o sinal de controlo dos servomotores. Os cálculos necessários para definir os *prescalers* a utilizar assim como o valor de arranque dos *timers* não são aqui discutidos nesta secção mas sim mais à frente quando se apresentar a rotina de serviço dos *interrupts* destes mesmos *timers*.

```
T0CON = 0b11000000;        //Activa Timer0
                           //Timer a 8 bits
                           //Clock interno

T1CON= 0b00110001;         //Leitura e Escrita em duas operações de 8 bits do registo do Timer
                           //Prescaler de 1:8
                           //Clock interno
                           //Activa Timer1
```

Timer0

O registo T0CON é o registo de configuração do timer0. É apresentado de seguida o significado dos seus bits.

T0CON REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0
bit 7							bit 0

TMR0ON - coloca em funcionamento o *timer*
T08BIT - 1= 8 bits
0 = 16 bits
T0CS - 0 = escolhe osc. Interno como relógio
T0SE - escolha *high-to-low* ou *low-to-high* (osc. Externo)
PSA - activa *prescaler*
T0PS2 - escolha *prescaler* (vêr tabela)

T0PS1 - escolha *prescaler* (vêr tabela)
T0PS0 - escolha *prescaler* (vêr tabela)

111	1:256
110	1:128
101	1:64
100	1:32
011	1:16
010	1:8
001	1:4
000	1:2

Configuração adoptada neste trabalho:

T0CON = 0b11001000;

- *Timer0* activado
- *Timer* a 8 bits
- *Clock* interno
- *Prescaler* activado
- Relação do *prescaler* 1:2

Timer1

T1CON REGISTER

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RD16	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON
bit 7							bit 0

RD16 - 1= 8 bits
0 = 16 bits
T1CKPS1 - escolha *prescaler*
T1CKPS0 - escolha *prescaler*
TMR1CS - escolha do relógio
TMR1ON - activa funcionamento do temporizador

Configuração adoptada neste trabalho:

T1CON= 0b00110001;

- Leitura e Escrita em duas operações de 8 bits do registo do *Timer*
  Relação do *prescaler* 1:8
- *Clock* interno
- Activa *Timer1*

Configuração da Usart

O módulo de comunicações série deste microcontrolador permite ser configurado como um sistema assíncrono *full-duplex* para comunicar com sistemas externos assim como por exemplo com um computador pessoal. Será este tipo de funcionamento que nos interessa para este trabalho pelo qual será somente apresentada a configuração necessária para este efeito. A configuração dos restantes modos de funcionamento pode ser consultados no *datasheet* deste microcontrolador.

Para configuramos o *baud rate* de comunicação é necessário efectuar um cálculo que tem como variáveis de entrada a frequência do oscilador do CPU assim como o valor pretendido para o *baud rate*. O resultado desse cálculo será registado no registo do *Baud Rate Generator*. Dado que se pretende que o sistema seja assíncrono e funcione a alta velocidade (*high speed*) a formula indicada é a seguinte:

Baud: 19200

FreqOsc: 40

SPBRG = (byte)((FreqOsc*1E6/Baud)/16.0)-1;

O resultado do cálculo para este caso é 129.

TXSTA: TRANSMIT STATUS AND CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D
bit 7							bit 0

Depois de se efectuar a escrita deste valor no registo do *Baud Rate Generator* é necessário configurar no registo TXSTA os seguintes bits:

BRGH - colocar este bit a *true* para funcionamento a alta velocidade.

TXEN - colocar este bit a *true* para activar a transmissão

RCSTA: RECEIVE STATUS AND CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
bit 7							bit 0

No registo RCSTA são necessárias as seguintes configurações:

SPEN - colocar este bit a *true* para efectuar a activação a comunicação série.

CREN - colocar este bit para permitir modo continuo de recepção.

Código que configura o modulo de comunicação série:

```
SPBRG = (byte)((FreqOsc*1E6/Baud)/16.0)-1;

TXSTAbits.BRGH = true;          //Usar a USART em High Speed,(BRG=0 Low Speed)
RCSTAbits.SPEN = true;          //Activa os pinos da porta serie

TXSTAbits.TXEN = true;
RCSTAbits.CREN = true;          //Modo continuo na recepcao fica activo
```

Configuração da Modulo de Aquisição e Conversão de Sinais Analógicos (ADC)

```
ADCON0 = 0b1000001;             //Fosc/32
                                //CHANNEL 0 (AN0)
                                //ADC ON
ADCON1 = 0b00000000;            //Left justified
                                //Vref+ --> Vdd
                                //Vref- --> Vss
```

Com o código apresentado configura-se o modulo ADC do microcontrolador para utilizar como relógio de conversão o oscilador externo com um valor de escala de 32. Define-se ainda que o canal de aquisição é o pino 0 do *Port* A (AN0).

A seguir indica-se a informação consultada para se efectuar a conversão do modulo ADC. Estes dados encontram-se disponíveis no *datasheet* do microcontrolador.

ADCON0 REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0
ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	—	ADON
bit 7							bit 0

ADCS0:ADCS1 -escolha do relógio de conversão (ver tabela)

ADCON1 <ADCS2>	ADCON0 <ADCS1:ADCS0>	Clock Conversion
0	00	Fosc/2
0	01	Fosc/8
0	10	Fosc/32
0	11	FRC (clock derived from the internal A/D RC oscillator)
1	00	Fosc/4
1	01	Fosc/16
1	10	Fosc/64
1	11	FRC (clock derived from the internal A/D RC oscillator)

CHS0: CHS2 -escolha dos canais para efectuar as conversões
000 = Canal 0 (AN0)
001 = Canal 1 (AN1)
010 = Canal 2 (AN2)
011 = Canal 3 (AN3)
100 = Canal 4 (AN4)

GO/DONE - (1) inicia a conversão , é colocado automaticamente a (0) quando conversão estiver completa.

ADON - 1 = modulo de conversão A/D ON
0 = modulo de conversão A/D OFF

ADCON1 REGISTER

R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

ADFM - justificação à direita ou esquerda do resultado da conversão.
ADCS2 - escolha do relógio de conversão.
PCFG3:PCFG0 - configuração do porta que vai efectuar a conversão

PCFG	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0	VREF+	VREF-	C / R
0000	A	A	A	A	A	A	A	A	VDD	VSS	8 / 0
0001	A	A	A	A	VREF+	A	A	A	AN3	VSS	7 / 1
0010	D	D	D	A	A	A	A	A	VDD	VSS	5 / 0
0011	D	D	D	A	VREF+	A	A	A	AN3	VSS	4 / 1
0100	D	D	D	D	A	D	A	A	VDD	VSS	3 / 0
0101	D	D	D	D	VREF+	D	A	A	AN3	VSS	2 / 1
011x	D	D	D	D	D	D	D	D	—	—	0 / 0
1000	A	A	A	A	VREF+	VREF-	A	A	AN3	AN2	6 / 2
1001	D	D	A	A	A	A	A	A	VDD	VSS	6 / 0
1010	D	D	A	A	VREF+	A	A	A	AN3	VSS	5 / 1
1011	D	D	A	A	VREF+	VREF-	A	A	AN3	AN2	4 / 2
1100	D	D	D	A	VREF+	VREF-	A	A	AN3	AN2	3 / 2
1101	D	D	D	D	VREF+	VREF-	A	A	AN3	AN2	2 / 2
1110	D	D	D	D	D	D	D	A	VDD	VSS	1 / 0
1111	D	D	D	D	VREF+	VREF-	D	A	AN3	AN2	1 / 2

A = Analog input D = Digital I/O
C / R = # of analog input channels / # of A/D voltage references

Configuração dos Interrupts e das suas prioridades

As linhas de código que se seguem configuram as prioridades dos interrupts necessários para o funcionamento de desejado do microcontrolador. Será mais à frente descrito todo o funcionamento deste *interrupts* e das suas rotinas de serviço.

```
RCON = 0b10000000;          //Activa os Interrupts de alta prioridade

IPR1 = 0b00000001;          //Interrupts relacionados com a USART
IPR3 = 0b00000000;          //estão declarados como de baixa prioridade.
                            //Interrupt Timer1 alta prioridade

INTCON2 = 0b00000100;       //Interrupt de "overflow" do Timer 0
                            // defenido como sendo de alta prioridade.
```

O próximo passo será a activação os *interrupts* propriamente ditos. É necessário ter alguma atenção em colocar esta activação sempre no final da função de configuração do Pic. Isto porque o microcontrolador começa logo a atender aos *interrupts* podendo não completar eventuais configurações que fossem efectuadas após a activação dos *interrupts.*

```
PIE1 = 0b01100001;          //Activa Interrupt caso cheguem dados
                            //pela USART.
                            //Activa Interrupt caso overflow Timer1

INTCON = 0b11100000;        //Activação global dos Interrupts
                            //Activação global dos Interrupts periféricos
                            //Activação do Interrupt do Timer 0
```

A seguir são apresentados os registos de configuração utilizados em detalhe.

INTCON REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
bit 7							bit 0

No registo INTCON configura-se, entre outros, o funcionamento do *interrupt* do *timer0*, os seguintes bits são utilizados para esse efeito:

TMR0IE - activa o funcionamento do *interrupt* do *timer0*
TMR0IF - *overflow interrupt flag bit*, sempre que ocorre um *interrupt* passa ao estado 1

Configuração adoptada neste trabalho:

```
INTCON = 0b11100000;
```

Relativamente ao timer0 efectuou-se a activação do seu *interrupt*, são efectuadas ainda as seguintes configurações neste registo:

- Activação global dos *Interrupts*
- Activação global dos *Interrupts* periféricos

Configuração adoptada neste trabalho:

INTCON2 = 0b00000100;

- *Interrupt* de "*overflow*" do *Timer 0* definido como sendo de alta prioridade.

PIE1: PERIPHERAL INTERRUPT ENABLE REGISTER 1

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
bit 7							bit 0

Neste registo é activado *interrupt* associado ao *overflow timer1* (TMR1IE).

Configuração adoptada neste trabalho:

PIE1 = 0b01100001;

Relativamente ao timer1 é activado o *interrupt* de *overflow*, são efectuadas ainda as restantes configurações neste registo.

- Activa *Interrupt* caso cheguem dados pela USART.
- Activa *Interrupt* caso seja efectuada uma conversão no modulo ADC.

PIR1: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 1

R/W-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
bit 7							bit 0

Neste registo é encontra-se *interrupt overflow flag* bit do timer1, este bit passa a um sempre que ocorre um *interrupt*.

Configuração adoptada neste trabalho:

PIR1= 0b00000001;

- Activa *Interrupt Timer1*

IPR1: PERIPHERAL INTERRUPT PRIORITY REGISTER 1

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
PSPIP(1)	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP
bit 7							bit 0

Neste registo atribui-se o tipo de prioridade do *interrupt* do timer1.

Configuração adoptada neste trabalho:

IPR1 = 0b00000001;

- *Interrupt Timer1* alta prioridade

void main(void)

Após se ter apresentado as duas funções anteriores passa se à descrição do funcionamento da rotina principal.
Os módulos desenvolvidos permitem o endereçamento de cada placa por meio de um *dip switch* de quatro posições. Este *switch* está ligado ao *Port* C do microcontrolador. Para que a respectiva placa seja bem endereçada é necessário ler o *Port* C para depois inicializar o CAN com o endereço do Slave em questão. É por esta razão que se efectua a respectiva leitura e registo do estado do *Port* C numa variável logo no início da rotina principal e antes de se fazer outra coisa qualquer. Também é necessário definir o endereço do Master antes de inicializar o CAN.

```
temporario = PORTC;
HostMcu = temporario & 0x0F  ;
MasterID = 0x01;
```

Em função do endereço da placa foi também necessário conferir *home positions* distintas por razões construtivas e de simetria na montagem dos servomotores nas pernas do robô.

```
if ((HostMcu == 2))
{       dutycicle3 = 225;
        pwm3 = 225;
        dutycicle2 = 225;
        pwm2 = 225;
}
else
{
        if((HostMcu == 3))
        {       dutycicle3 = 105;
          pwm3 = 105;
                dutycicle2 = 105;
          pwm2 = 105;
        }
        else
        {
                if((HostMcu == 4))
                {       dutycicle3 = 105;
                pwm3 = 105;
```

```
                    }
                    else
                    {
                        if((HostMcu == 5))
                        {       dutycicle3 = 225;
                        pwm3 = 225;

                        }
                        else
                        {
                            dutycicle3 = 168;
                            pwm3 = 168;
                        }
                    }
                }
            }
```

Nesta fase já se está em condições de se inicializar o CAN e efectuar as restantes configurações do microcontrolador.

```
InitCan(HostMcu);

InitPic(FREQ_OSC, BAUD);
```

Estando o microcontrolador a funcionar correctamente o programa fica preso num ciclo *while* infinito que é somente interrompido para atender aos diversos *interrupts* que gerem todo o funcionamento do modulo de controlo.

```
while(1)
{


};
```

void ServoIsr(void) – Rotina de Serviço aos interrupts de alta prioridade

Esta rotina é responsável pela geração dos sinais de controlo dos três servomotores controlados por cada microcontrolador. Esta função já tinha sido desenvolvida no ano anterior no entanto foi necessário efectuar modificações consideráveis dado que o controlo desenvolvido não se verificava satisfatório uma vez que os servomotores apresentavam um comportamento algo instável. Como se consegue verificar no vídeo os motores quando deviam permanecer numa determinada posição apresentavam constantemente ligeiros desvios que eram prontamente corrigidos. Embora fossem flutuações muito ligeiras era importante que se conseguisse corrigir este comportamento uma vez que o carácter do trabalho exige um controlo muito rigoroso dos actuadores.
Começou-se por verificar a qualidade do sinal de controlo gerado pelo ***Pic***. Rapidamente verificou-se que o PWM (*Pulse With Modelation*) de 50 Hz apresentava esporadicamente uma anomalia. Por vezes a onda alterava a sua frequência para 100 Hz . O facto de não se verificar sistematicamente o erro tornou algo complicada a procura da fonte de tal comportamento, no entanto verificou-se que a causa não estaria no oscilador externo uma vez que a onda que este gerava era de 50Hz sem nenhuma anomalia.

Tomou-se então a decisão de trocar o oscilador externo por um *timer* interno que se encarregaria de gerar a frequência de 50Hz. Optou-se também por trocar o *timer* que gerava a frequência de resolução que permite a variação do *duty cicle*. É necessário referir que este Microcontrolador possui um modulo de geração PWM (*CCP module*), o *timer* associado por defeito à geração do PWM é o *timer2*. Este *timer* verificou-se impróprio para geração de um PWM com 50 Hz de frequência, uma vez que se trata de um *timer* de 8 bits com um *prescaler* máximo de 1:16 e um *postscaler* máximo de 1:16. No máximo permite gerar um PWM com frequências superiores a 76.3 Hz. Verificou-se também que era impossível trocar este *timer* por um de 16 bits. Foi por esta razão que se optou por não utilizar o módulo CCP para gerar o PWM.
São apresentados a seguir os cálculos necessários para garantir a cadência correcta dos interrupts dos timers para gerar o sinal de controlo pretendido.

Cálculos efectuados para a geração da onda de 50 Hz

Os cálculos efectuados são apresentados de seguida:

50 Hz equivale a um período P = 0.02s

$F_{TIMER} = F_{OSC} / 4 = 5 \times 10^{6}$Hz

5×10^{6} Hz equivale a um período de 2.0×10^{-7}s --> $P_T = 2.0 \times 10^{-7}$s

Um *timer* de 8 bits sem *prescaler* permite contagem de um período máximo de :

$P_{MAX} = P_T * 256 = 5.12 \times 10^{-5}$s $< 2.0 \times 10^{-2}$s ; não serve.

Um *timer* de 8 bits com um *prescaler* 1:8 permite contagem de um período máximo de :

$P_{MAX} = P_T * 256 * 8 = 4.1$ x10-4s < 2.0 x10-2s ; não serve.

Um *timer* de 16 bits com um *prescaler* 1:8 permite contagem de um período máximo de :

$P_{MAX} = P_T * (256)^2 * 8 = 0.104$s $> 2.0 \times 10^{-2}$s ; já serve.

P_{MAX} - P = 0.085s --> temos esta diferença.

O *timer* não pode arrancar do zero mas sim com um incremento correspondente a 0.085s para que se consiga ter um período de 0.02 *s* até que ocorra o *overflow*.

0.085s * (F_{OSC} / 4) = 53125 incrementos

O numero 53125 equivale em binário a 1100111110000101, trata-se de um valor de 16 bits que tem de ser dividido por dois, como se pode ver no extracto de código apresentado a seguir os bits mais significativos são introduzidos no registo do timer1 TMR1H e os bits menos significativos são introduzidos no registo TMR1L.

A rotina aqui apresentada está associada ao *interrupt* de alta prioridade do timer1, uma vez que se configurou este *timer* como sendo de alta prioridade. A função desta rotina é iniciar um novo ciclo de controlo dos servomotores. Como é possível verificar nas linhas iniciais atribui-se ao timer1 o seu valor de arranque para que se tenha sempre a temporização correcta. A seguir é iniciado o timer0 responsável pela geração da frequência de resolução do *dutycicle*. As três linhas de código seguintes colocam os pinos Rb5, Rb6 e Rb7 com 5 Volts. A última linha limpa a *flag* de *interrupt* do timer1.

```
if (PIR1bits.TMR1IF)
 {                  //o interrupt do Timer1 esta configurado como sendo de alta prioridade
        TMR1H= 0b11001111;
        TMR1L= 0b10000101;

        INTCONbits.TMR0IE = true;


        PINO_SERVO1 = true;
        PINO_SERVO2 = true;
        PINO_SERVO3 = true;

        contador = 0;

  PIR1bits.TMR1IF = false;

 };
```

Cálculos efectuados para a geração da frequência de resolução

Da documentação dos servomotores retirou-se a informação de que o *dutycicle* mínimo corresponde a 8 % e o *dutycicle* máximo corresponde a 12% do período de 0.02s.

8 % ---> 8.0×10^{-4}s
12% --> 2.4×10^{-3}s

Teremos então uma variação do *dutycicle* de :

$\Delta(dutycicle) = 8.0 \times 10^{-4} - 2.4 \times 10^{-3} = 1.2 \times 10^{-3}$ s

Para uma resolução de 3 graus no posicionamento dos motores verifica-se:

1.2×10^{-3}s / 60 = 2.0×10^{-5}s

- este será o período que define a frequência de resolução de 3 graus.

2.0×10^{-5} * (F_{OSC} / 4) = 100 incrementos

São necessário 100 incrementos (ciclos relógio) para se ter um período de 2.0×10^{-5}s

O *timer* não pode arrancar do zero mas sim com um incremento correspondente a 3.12 x10^{-5} s (156 incrementos) para que se consiga ter um período de 2.0 x10-5 s até que ocorra o *overflow*. O numero 156 equivale em binário a 10011100.

Como na rotina anterior o valor de arranque do *timer* é escrito no registo do *timer0* logo no início da rotina para que seja garantida a temporização exacta.
Esta rotina tem como objectivo por os pinos Rb5, Rb6 e Rb7 no momento exacto a 0 Volts. Este *timing* corresponde a uma determinada posição do servomotor.
À que realçar o facto de se desligar o *timer0* após estar concluída uma fase de controlo. Isto evita que os *interrupts* prioritários de alta frequência estejam a ocorrer sem necessidade sendo assim libertado o ***Pic*** para efectuar a leitura dos sensores e ler os *buffers* de chegada do CAN.

```
if(INTCONbits.TMR0IF)
{       TMR0L= 0b10011100;

        contador++;
        if ( contador >= pwm1)// PWM do primeiro motor
                PINO_SERVO1 = false;

        if (contador >= pwm2)// PWM do segundo motor
                PINO_SERVO2 = false;

        if (contador >= pwm3)// PWM do terceiro motor
                PINO_SERVO3 = false;

   INTCONbits.TMR0IF = false;

        if (contador>MaxCont)
        {       INTCONbits.TMR0IE = false;
                ADCON0bits.GO = true;
        }
};
```

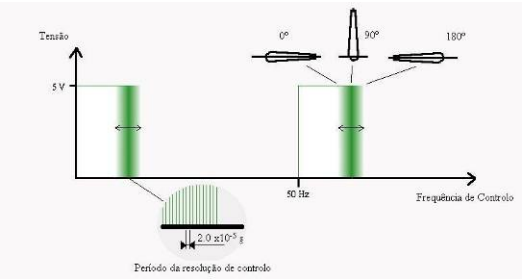


Figura 7 – Sinal de controlo dos Servomotores

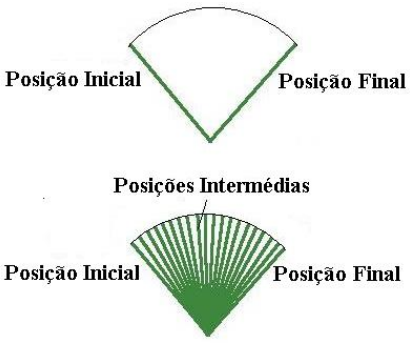


Figura 8 – Ilustração do controlo de velocidade

Verificou-se que esta alteração correspondeu da melhor maneira uma vez que se conseguiu que os servos apresentassem um óptimo comportamento para todas as solicitações necessárias.

void Contr_Velo(void)

Esta função é responsável pelo controlo de velocidade dos servomotores
Os servomotores de posição utilizados possuem um controlador integrado que avalia o sinal de um potenciómetro interno e permite assim garantir a posição correcta em malha fechada. O movimento é efectuado a velocidades distintas dependendo da diferença entre a posição desejada e a posição actual. Se o desvio de posicionamento for elevado, o servomotor desloca-se à sua velocidade máxima para depois reduzir rapidamente evitando que ocorra *overshoot* da posição requerida.
No âmbito do actual projecto é de interesse poder controlar a velocidade dos servomotores sem que se perca o seu controlo de posição. São necessárias velocidades distintas para permitir uma movimentação suave e em simultâneo de várias juntas, ou velocidades mais altas para o caso de correcções de emergência.
Para efectuar o controlo de velocidade neste tipo de motores substitui-se o potenciómetro por uma resistência fixa que equivale à posição de 90º. A aplicação deste método implica a perda do controlo de posição do servomotor, o que não é relevante caso se pretenda utilizá-los unicamente como motores de actuação contínua. A velocidade a que o motor gira é controlada através dos valores de referência de posição (PWM). Dado que não existe controlo de malha fechada o motor roda sem parar a uma velocidade que depende da diferença entre o valor fornecido e o valor de 90º.
Para os valores próximos dos 90º as velocidades de rotação são baixas, à medida que a diferença entre estes aumenta a velocidade aumenta rapidamente até atingir o seu valor máximo. Se o valor de referência for inferior a 90º o motor roda no sentido inverso.
Outra técnica será deixar os servomotores com a sua configuração original, efectuando unicamente uma ligação ao terminal do potenciómetro para que se conheça a posição corrente do servo.
O controlo de posição é garantido pelo próprio servomotor, o controlo de velocidade é garantido por *software*. Sabendo que existe uma gama de velocidades que depende do erro de posição do servo, implementou-se uma rotina de controlo que, para uma dada variação de posição obriga que o motor atinja a posição efectuando pequenas etapas intermédias.
Dado que estamos constantemente a ordenar que o motor efectue pequenos incrementos ele roda a uma mais baixa velocidade. O *timing* destes incrementos é sempre suficientemente elevado para que os motores nunca cheguem a parar. O controlo de velocidade é feito em malha fechada utilizando o sinal do potenciómetro para definir se é necessário incrementar mais um avanço ou se deve esperar que "passe" mais tempo antes de impor novo ponto de destino.
A principal vantagem desta técnica está no facto de se aproveitar o controlador de posição integrando no servomotor não sendo necessário sobrecarregar o PIC com um algoritmo adicional. Dada a filosofia de agrupamento de juntas por microcontrolador o PIC teria de apresentar um esforço computacional considerável para o controlo dos três motores. Outra vantagem reside no facto de exigir alterações mínimas no servomotor. Uma vez que se verificaram resultados satisfatórios com esta abordagem não se passou à implementação da primeira técnica.
Dada a importância da função que se desenvolveu para efectuar esta tarefa apresenta-se a seguir uma descrição detalhada da mesma.

```
void Contr_Velo(void)
{
        control_vel1++;
        control_vel2++;
        control_vel3++;

        if (ValSens[2] < MAX_SENS)          //estas linhas de código
                new_value_1 = ValSens[2];   //limitam o valor máximo
                                            //dos potenciómetros
                                            //(filtro passa baixo)
        if (ValSens[1] < MAX_SENS)
                new_value_2 = ValSens[1];


        if (ValSens[0] < MAX_SENS)
                new_value_3 = ValSens[0];

        if (velocidade1 > 5)
                velocidade1 = 1;

        if (control_vel1 > velocidade1)
        {
        Erro1 = pwm1 - (byte)(100 + 130.0 * (new_value_1 - 35)/49.0 );

                if (Erro1 != 0)
                {
                        if ( Erro1 > 5)
                                dutycicle1++;

                        if ( Erro1 < -5)
                                dutycicle1--;
                }
                control_vel1 = 0;
        }

}
```

Dado que se queria incrementar a posição dos servomotores por posições intermédias até à posição final (pwm1..2) começou-se por calcular o erro (Erro1..3) entre a posição desejada e a posição actual. Em função do sinal desse erro a posição dos servomotores (dutycicle1..2) é incrementado positiva ou negativamente.
Os incrementos na posição podem ser efectuados a uma taxa máxima de 50Hz. A esta taxa já se verifica um atraso considerável dos servomotores. Mas é possível baixar esta taxa de incrementos e assim baixar ainda mais a velocidade dos servomotores. Basta ter um contador (control_vel1..3) que vai incrementando cada ciclo até obtermos o valor igual à velocidade (atraso) pretendida que vem em forma de um valor de zero a cinco
(velocidade1..3). O valor zero será o atraso mínimo o valor cinco corresponde ao atraso máximo.
O controlo que se efectua com este código é em malha fechada dado que se utiliza o valor vindo dos potenciómetros para decidir se se deve efectuar uma alteração na posição ou não. No entanto verificou-se que em funcionamento no robô esta abordagem não era a mais estável uma vez que os servomotores se descontrolavam com frequência. Uma alternativa consiste substituir a linha de código:

```
Erro1 = pwm1 - (byte)(100 + 130.0 * (new_value_1 - 35)/49.0 );
```

Pela seguinte:

```
Erro1 = pwm1 – duticicle1;
```

Esta versão é muito mais estável no entanto não temos o controlo em malha fechada.

void ComIsr(void);) – Rotina de Serviço aos interrupts de baixa prioridade

Esta função é a rotina de serviço dos *interrupts* de baixa prioridade. Os *interrupts* que estão configurados como sendo de baixa prioridade no caso dos slaves são os *interrupts* da ADC e dos *buffers* de recepção das comunicações CAN. Segue-se a descrição detalhada desta rotina.

Aquisição de sinais analógicos

Como já foi referido anteriormente é necessário que os controladores sejam capazes de adquirir sinais analógicos para se poder obter informação dos diversos sensores do sistema. Cada placa controladora terá no máximo de adquirir sete sinais analógicos, é o caso das duas placas responsáveis pela aquisição dos sinais dos sensores dos pés. Cada pé terá quatro extensómetros acrescentando ainda os três potenciómetros que indicam as posições das juntas têm-se então sete sensores. As restantes placas controladoras (4) têm um numero inferior de sensores a este.
O ***Pic 18f258*** possui somente 5 canais de conversão A/D. Optou-se pela utilização de um *multiplexer* de 16 canais (*HEF4067B CNV 3*) que permite assim que todas as 7 conversões sejam feitas por um único canal do microcontrolador. As comutações entre os diferentes canais no *multiplexer* é efectuada por quatro sinais digitais, estes são gerados pelo ***Pic*** de forma a que seja efectuado o registo correcto dos valores sensoriais na matriz *ValSens*.
O módulo AD deste ***Pic*** permite a conversão de um sinal analógico no seu correspondente digital de 10 bits. No total possui quatro registos, dois são de configuração (ADCON0, ADCON1) os outros dois (ADRESH, ADRESL) são os registos nos quais são escritos o valor digital resultante da conversão. A seguir explica-se como foi efectuada a configuração.

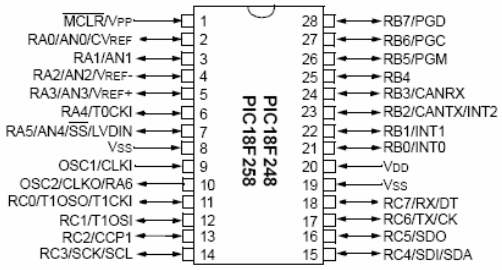


Figura 9 – Pin out do microcontrolador

Como é possível verificar os canais de aquisição analógica (AN0, AN1, AN2, AN3, AN4) neste ***Pic*** estão localizados no *Port A*. Optou-se por utilizar o canal AN0 para efectuar as aquisições e os pinos (RA1,RA2,RA3, RA5) como saídas para comutar os canais do multiplexer. Os pinos RA1, RA2 , RA3 e RA5 estão ligados fisicamente aos pinos A0, A1, A2 e A3 do *Mutliplexer*. Para pormenores acerca do circuito eléctrico por favor consultar os esquemas eléctricos.

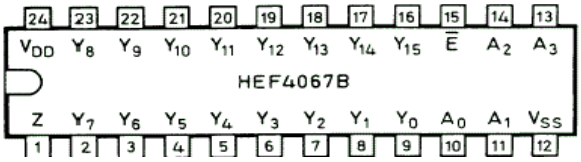


Figura 10 – Multiplexer utilizado

INPUTS					CHANNEL
$\overline{E}$	A_3	A_2	A_1	A_0	ON
L	L	L	L	L	$Y_0 - Z$
L	L	L	L	H	$Y_1 - Z$
L	L	L	H	L	$Y_2 - Z$
L	L	L	H	H	$Y_3 - Z$
L	L	H	L	L	$Y_4 - Z$
L	L	H	L	H	$Y_5 - Z$
L	L	H	H	L	$Y_6 - Z$
L	L	H	H	H	$Y_7 - Z$
L	H	L	L	L	$Y_8 - Z$
L	H	L	L	H	$Y_9 - Z$
L	H	L	H	L	$Y_{10} - Z$
L	H	L	H	H	$Y_{11} - Z$
L	H	H	L	L	$Y_{12} - Z$
L	H	H	L	H	$Y_{13} - Z$
L	H	H	H	L	$Y_{14} - Z$
L	H	H	H	H	$Y_{15} - Z$
H	X	X	X	X	none

Note
1. H = HIGH state (the more positive voltage)
L = LOW state (the less positive voltage)
X = state is immaterial

Figura 11 – Tabela de binários para comutar

O código implementado comuta o canal antes de se efectuar a aquisição do sinal respectivo. Este microcontrolador permite a configuração de um *interrupt* que ocorre sempre que seja efectuada uma conversão. Este *interrupt* está configurado como sendo de baixa prioridade, os pormenores da configuração do *interrupt* e da sua prioridade estão apresentados na secção dedicada aos *interrupts* utilizados deste texto.
A rotina de escrita do respectivo valor no *array ValSens* que é efectuada quando ocorre o *interrupt* está por isso na zona de rotinas de baixa prioridade. È necessário referir aqui um pequeno pormenor, verificou-se que a conversão do valor analógico no ***Pic*** era mais rápida do que comutação do multiplexer e por isso inicialmente eram adquiridos valores de tensão que correspondiam a sinais transitórios e não aos valores que se pretendia obter. Esta situação corresponde às linhas vermelhas da figura.

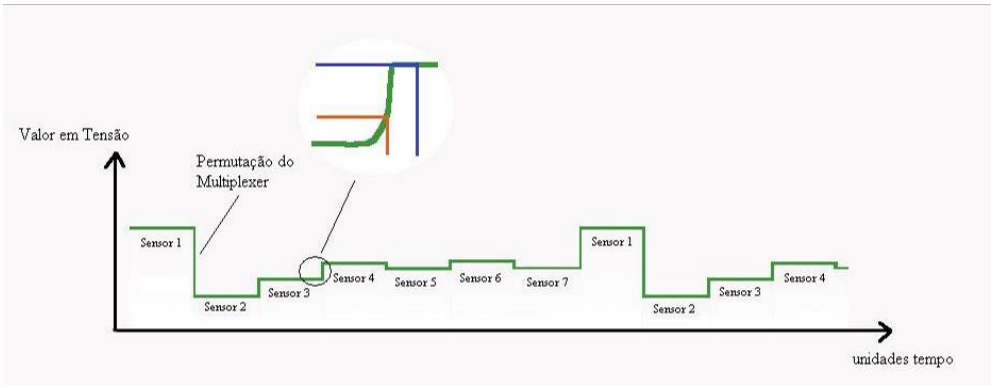


Figura 12 - Erro na aquisição dos sinais sensoriais

A solução encontrada consiste basicamente em atrasar a escrita dos resultados da conversão no *array ValSens*. Utiliza-se um *if* que só permite o registo do resultado após efectuar-se 20 vezes a aquisição. Esta abordagem não é muito elegante no entanto permite resolução do problema sem recorrer *a um outro timer*.

Canal de aquisição	Binário a ser colocado no Port A
pontenciometro 0 (canal mult-Y0)	00000000
pontenciometro 1 (canal mult-Y1)	00000010
pontenciometro 2 (canal mult-Y2)	00000100
medicao corrente 0 (canal mult-Y3)	00000110
medicao corrente 1 (canal mult-Y4)	00001000
medicao corrente 2 (canal mult-Y5)	00001010
extensometro 0 (canal mult-Y6)*	00001100 (00101100) *
extensometro 1 (canal mult-Y7)*	00001110 (00101110) *
extensometro 2 (canal mult-Y14)	00101100
extensometro 3 (canal mult-Y15)	00101110

Como se pode verificar no código apresentado em baixo a leitura dos canais analógicos é efectuada de maneira diferente consoante o número da placa do Slave. Na placa número dois e três, que correspondem às placas que se encontram localizadas nos pés são adquiridos quatro canais (Y6, Y7, Y14, Y15). Fora estes dois casos são só adquiridos nas restantes placas os canais Y6 e Y7. As restantes placas só devem ter como placas de sensores o inclinómetro ou giroscópio. Estas placas de aquisição sensorial só necessitam de dois canais de aquisição e dado que se verificou que o multiplexer não lida bem com canais no ar optou-se por efectuar a aquisição dos canais que estarão obrigatoriamente sempre ocupados. Para efeito definiu-se que a posição a ocupar pelas placas de aquisição sensorial do inclinómetro e do giroscópio é a apresentada na imagem. Nesta altura ainda não se tem certeza quais são as placas controladoras que não vão efectuar aquisição sensorial mas logo que se conheça o código será alterado para que os canais que estejam no ar não sejam lidos pelo microcontrolador.
Pode se verificar que após um registo válido é efectuada a comutação do canal do *multiplexer* para a próxima aquisição. A troca de um canal do *multiplexer* consiste basicamente em activar as saídas do ***Pic*** que estão conectadas aos pinos de controlo do *multiplexer* de forma a obter-se o canal desejado conforme apresentado na tabela acima.

```
if(PIR1bits.ADIF)                       //rotina de leitura dos sinais analogicos
{//-------------------------------------------------------------------------------------------------------------------

        byte ADC;

        ADC = ADRESH;

        n++;                            //este contador associado ao if seguinte permite que o
multiplexer tenha tempo
        if(n > 20)                      //estabelisar apos uma mudança de canal
        {       n = 0;

                ValSens[cont] = ADC;    //escrita do valor do sinal no array ValSens
                switch (cont)
                                {
                        case 0: PORTA = 0b00000010; //pontenciometro 1 (canal mult-Y1)
                                break;
                        case 1: PORTA = 0b00000100; //pontenciometro 2 (canal mult-Y2)
                                break;
                        case 2: PORTA = 0b00000110; //medicao corrente 0 (canal mult-Y3)
                                break;
                        case 3: PORTA = 0b00001000; //medicao corrente 1 (canal mult-Y4)
```

```
                        break;
                case 4: PORTA = 0b00001010; //medicao corrente 2 (canal mult-Y5)
                        break;
                case 5: if ((HostMcu == 2)|(HostMcu == 3))
                                PORTA = 0b00001100; //extensometro 0 (canal mult-Y6)
                                else
                                PORTA = 0b00101100;
                        break;
                case 6: if ((HostMcu == 2)|(HostMcu == 3))
                                PORTA = 0b00001110; //extensometro 1 (canal mult-Y7)
                                else
                                PORTA = 0b00101110;
                        break;
                case 7: PORTA = 0b00101100; //extensometro 2 (canal mult-Y14)
                        break;
                case 8: PORTA = 0b00101110; //extensometro 3 (canal mult-Y15)
                        break;
                case 9: PORTA = 0b00000000; //pontenciometro 0 (canal mult-Y0)
                        break;
        }
            cont++;
            if(cont>9)
            {       cont = 0;
            }
    }

            ADCON0bits.GO = true;
            PIR1bits.ADIF = false;
};
```

Comunicações CAN

A recepção de mensagens via CAN também está configurada como sendo de baixa prioridade. Como foi descrito anteriormente sempre que os Slaves recebem mensagens pelo CAN respondem com o envio de duas mensagens contendo os valores sensoriais actualizados.
Dado que se está a utilizar a livraria que implementa as comunicações CAN fornecida pela Microchip basta utilizar a função de leitura dos *Buffers* CANReceiveMessage para ter acesso à mensagem recebida. Se não ocorrer erros só resta actualizar as varáveis locais do Slave.

```
if(PIR3 & 0x03)                 //rotina de leitura de mensagens vindas pelo CAN
{//-------------------------------------------------------------------------------------------------


    acabou = CANReceiveMessage(&ID, MsgRecCan, &CompMsg, &Avisos);
    if (acabou)
    {
            //Actualizacao valores pwm

            pwm1 = MIN_PWM + Relacao*max(min(MsgRecCan[0],MAX_ANG),MIN_ANG);

            pwm2 = MIN_PWM + Relacao*max(min(MsgRecCan[1],MAX_ANG),MIN_ANG);

            pwm3 = MIN_PWM + Relacao*max(min(MsgRecCan[2],MAX_ANG),MIN_ANG);

            velocidade1 = MsgRecCan[3];
```

```
                velocidade2 = MsgRecCan[4];

                velocidade3 = MsgRecCan[5];



        }
        else
                ERRO_CAN = true;//ocorreu um erro no bus

        switch (Avisos & CAN_RX_FILTER_BITS)
        {
        case CAN_RX_OVERFLOW:
                {
                ERRO_CAN = true;
                COMSTAT &= 0b00111111;//retirar o overflow
                };
        case CAN_RX_INVALID_MSG:
                {
                ERRO_CAN = true;
                PIR3bits.IRXIF = false;
                }
        };

EnviaValSens(MasterID, HostMcu, ValSens);
                                //sempre que chega algo pelo CAN e enviada a matriz com os valores
                                //sensoriais para o mestre


};
```

Imediatamente a seguir a esta actualização evoca-se uma função EnviaValSens que trata do envio dos valores sensoriais para o Master. O envio propriamente dito é efectuado por uma função CANSendMessage da livraria fornecida pela Microchip.

```
void EnviaValSens(byte Destino, byte HostMcu, byte *ValSens)
{                                               //envio dos valores sensoriais via CAN para o mestre

#define Msg_Motores 0;
#define Msg_Outros_Sens 128;

byte MsgSendCan[10], i;
byte tmp=123;


        MsgSendCan[0] = HostMcu;
MsgSendCan[1] = Msg_Motores;

        for(i = 2; i<8; i++)
                MsgSendCan[i] = ValSens[i-2];

        CANSendMessage((unsigned long)Destino, MsgSendCan, 8, CAN_TX_FLAGS);

        MsgSendCan[0] = HostMcu;
        MsgSendCan[1] = Msg_Outros_Sens;

        for(i = 2; i<6; i++)
                MsgSendCan[i] = ValSens[i+4];
```

```
        MsgSendCan[6] = 0;
        MsgSendCan[7] = 0;

        CANSendMessage((unsigned long)Destino, MsgSendCan, 8, CAN_TX_FLAGS);

}
```

3.1.2 Enumeração e explicação de todas as variáveis

Enumeração de todas as variáveis utilizadas no código do Slave

A seguir apresentam-se as variáveis utilizadas e uma breve discrição da sua função no código.

#define MAX_VEL 0	Define o atraso mínimo dos servos, ou seja este valor corresponde à velocidade máxima.
#define MIN_VEL 6	Define o atraso máximo dos servos, ou seja este valor corresponde à velocidade mínima. Atrasos superiores a este são possíveis no entanto a resposta dos motores não é desejável uma vez que se já se verifica um "para-arranca" acentuado.
#define MAX_ANG 170.0	Define o ângulo máximo descrito pelos servomotores.
#define MIN_ANG 5.0	Define o ângulo máximo descrito pelos servomotores.
const int MIN_PWM = 100;	Constante que define o pwm máximo com que se consegue controlar os motores. O valor 100 corresponde a 5º. Definiu-se este valor assim para proteger o servomotor.
const int MAX_PWM = 230;	Constante que define o pwm mínimo com que se consegue controlar os motores. O valor 230 corresponde a 170º. Definiu-se este valor assim para proteger o servomotor.
const int MAX_CONT = 231;	Limite de segurança que é utilizado para desligar os interrupts que geram a frequência de resolução do pwm.
const int MAX_SENS = 83;	Constante que define o valor máximo do potenciómetro dos servomotores.
const int MIN_SENS = 32;	Constante que define o valor mínimo do potenciómetro dos servomotores.
CAN_TX_FLAGS 0b11000111	
ERRO_CAN PORTBbits.RB4	Quando o CAN dá um erro este pino do Port B é utilizado para indicar o mau funcionamento do CAN.

int pwm1, pwm2, pwm3;	Variáveis nos quais são registados os valores desejados de posição vindos pelo CAN.
int velocidade1, velocidade2, velocidade3;	Variáveis nos quais são registados os valores desejados da velocidade vindos pelo CAN.
int control_vel1, control_vel2, control_vel3;	Contadores que são utilizados para controlar o atraso da velocidade dos motores.
int contador = 0;	Contador responsável pelo dutycicle do pwm. O valor máximo deste contador é o valo MAX_CONT.
int dutycicle1, dutycicle2, dutycicle3;	Variáveis nas quais é registado o dutycicle que será enviado aos motores.
int Erro1, Erro2, Erro3;	Nesta variável é registado registado o erro actual entre a posição pretendida e a posição desejada dos servomotores.
byte temporario;	Esta variável é temporária e só serve para transferir o estado das entradas do PORT C que representam o endereço actual da placa para a variável HostMcu. Após o endereço estar registado nestavariável o CAN já pode ser inicializado.
int erro_14_23 = 0; int erro_12_34 = 0;	Estas variáveis foram utilizadas para o controlo de força dos Pés. Erro_14_23 representam por exemplo o erro do valor dos extensómetros 1 e 4 em relação aos extensómetros 2 e 3. Actualmente o código que permite o controlo de força dos pés está comentado.
int Nova_Correcao1 = 0; int Nova_Correcao2 = 0;	Variável necessária para os cálculos do controlo de força dos pés.
int Correcao1 = 0; int Correcao2 = 0;	Variável necessária para os cálculos do controlo de força dos pés.

int new_value_1 = 44;
int new_value_2 = 44;
int new_value_3 = 44;
	Variáveis utilizadas para registar o valor actual dos potenciómetros dos servomotores. Estas variáveis são depois utilizadas para calcular o erro entre a posição desejada e aposição actual dos servomotores.

byte ValSens[10] = {1,2,3,4,5,6,7,8,9,10};	Array no qual é registado os valores sensoriais.

ValSens[0..2]	Valor dos potenciómetros.
ValSens[3..5]	Valor da tensão nas resistências de potência para calcular o

		consumo dos servomotores.
	ValSens[6..9]	Valor dos restantes sensores (sensores força ou inclinómetro ou giroscópio).
float Relacao = 0.0;		Esta variável serve para registar o factor de escala com o qual se converte os valores angulares em valores para o *dutycicle* dos *pwm*.
float relacao_si = 0.0;		Esta variável serve para registar o factor de escala com o qual se efectua a conversão entre os valores de tensão dos sensores de força e os valores numéricos para os cálculos.
byte HostMcu;		Variável na qual se regista o endereço da placa que é definido por um *switch* que se encontra ligado ao PORT C.
byte MasterID;		Variável na qual se regista o endereço do Master.

bool acabou;	Variável do Can
byte CompMsg;	Variável do Can
unsigned long ID;	Variável do Can
int n;	Variável do Can
static byte cont = 0;	Contador que serve para indexar o *array* ValSens na aquisição sensorial.
static byte MsgRecCan[6];	*Array* do Can
#define Msg_Motores 0;	1º Tipo de Mensagem que é enviado pelo Slave para o Master.
#define Msg_Outros_Sens 128;	2º Tipo de Mensagem que é enviado pelo Slave para o Master.
byte MsgSendCan[10];	*Array* no qual é escrito a mensagem a enviar pelo Can para o Master.
byte tmp=123;	Variável temporária
int Si12, Si34, Si14, Si23;	Variáveis necessárias para os cálculos no controlo de força.

Dá-se com isto por concluída a explicação do código dos controladores escravos. Passa-se então á explicação do código do controlador Master.

3.2 Documentação do Código do microcontrolador Mestre

3.2.1 Enumeração e explicação de todas as funções

Todas funções de configuração e inicialização do microcomtrolador são idênticas às funções utilizadas no código dos microcontroladores Escravos pelo qual não voltam a ser discutidas nesta secção.

	Funções contidas neste ficheiro:	Descrição da função:
 MasterPrinc Ficheiro C 10 KB		
	void UsartIsr(void)	Rotina de serviço dos interrupts de alta prioridade.
	void CanTimerIsr(void)	Rotina de serviço dos interrupts de baixa prioridade.
 MasterSec Ficheiro C 9 KB		
	void AnalisaMsgUsart (PernaBraco *, byte *, byte *, byte *, byte *, byte *, byte *, byte *);	Analisa as mensagens recebidas por comunicação série e executa os comandos reconhecidos.
	void HomePosition (byte *, byte *);	Coloca p robô na home position.
	static void EnviaSensPes (byte linha, byte *SensPe)	Monta a mensagem com os valores sensoriais adicionais de cada placa e envia os por comunicação série para o Mestre.
	Static void EnviaArtPos (byte NumMCU, byte NumServo, byte *ValSens)	Monta a mensagem com os valores de posição dos motores e envia os por comunicação série para o Mestre.
 MasterTrd Ficheiro C 4 KB		
	void AnalisaMsgCan (byte *, byte, byte *, byte *, byte *,	Analisa as mensagens vindas por CAN e efectua os registos dos valores sensoriais nos arrays correctos.

	byte)	
	static void EscrValSensPe (byte linha, byte *Msg, byte *Pes)	Função de registo par os valores sensoriais dos sensores.

MASTER

Como anteriormente no Código do Slave antes de se passar a comentar e a descrever a rotina principal do programa tem que abordar as funções de configuração do microcontrolador. No caso da função de iniciação do CAN InitCan não existem diferenças é o mesmo código que se utiliza.

No caso da função que inicializa os periféricos do Microcontrolador existem algumas diferenças sobre tudo na configuração da prioridade dos *interrupts*.

void InitiPic()

A principal diferença será a ausência da configuração do módulo ADC uma vez que não é necessário que o Master efectue aquisições de sinais analógicos. Outra diferença será a configuração de um só *timer* dado que também só é necessário um para gerar a frequência de envio de mensagens via CAN para os Slaves.

Configuração do Timer0 e do Timer1

Este *timer* foi configurado para gerar uma frequência de 11KHz. Será esta a frequência com que as mensagens serão enviadas para os slaves.

T0CON REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0
bit 7							bit 0

TMR0ON - coloca em funcionamento o *timer*
T08BIT - 1= 8 bits
0 = 16 bits
T0CS - 0 = escolhe osc. Interno como relógio
T0SE - escolha high-to-low ou low-to-high (osc. Externo)
PSA - activa *prescaler*
T0PS2 - escolha *prescaler* (vêr tabela)
T0PS1 - escolha *prescaler* (vêr tabela)
T0PS0 - escolha *prescaler* (vêr tabela)

111	1:256
110	1:128
101	1:64
100	1:32
011	1:16
010	1:8
001	1:4
000	1:2

Configuração adoptada neste trabalho:

T0CON = 0b11001000;

- *Timer0* activado
- *Timer* a 8 bits
- *Clock* interno
- *Prescaler* activado
- Relação do *prescaler* 1:2

Código que configura o timer é apresentado a seguir.

```
T0CON = 0b11001010;          //Activa Timer0
                             //Timer a 8 bits
                             //Clock interno
                             //prescaler 1:8
```

Configuração dos Interrupts e das suas prioridades

As linhas de código que se seguem configuram-se as prioridades dos interrupts necessários para o funcionamento de desejado do microcontrolador Master. RCON = 0b10000000; //Activa os Interrupts de alta prioridade

```
IPR1 = 0b00100000;           //Interrupts relacionados com a USART
IPR3 = 0b00000000;           //estão declarados como de alta prioridade.
                             //Interrupt Timer0 baixa prioridade

INTCON2 = 0b00000000;        //Interrupt de "overflow" do Timer 0
                             // defenido como sendo de baixa prioridade.
```

O próximo passo será a activação os *interrupts* propriamente ditos. È necessário ter alguma atenção em colocar esta activação sempre no final da função de configuração do *Pic*. Isto porque o microcontrolador começa logo a atender aos *interrupts* podendo não completar eventuais configurações que fossem efectuadas após a activação dos *interrupts*.

```
PIE1 = 0b00100000;           //Activa Interrupt caso cheguem dados
                             //pela USART.


INTCON = 0b11100000;         //Activação global dos Interrupts
                             //Activação global dos Interrupts periféricos
                             //Activação do Interrupt do Timer 0
```

A seguir são apresentados os registos de configuração utilizados em detalhe.

IPR1: PERIPHERAL INTERRUPT PRIORITY REGISTER 1

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
PSPIP(1)	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP
bit 7							bit 0

Neste registo atribui-se o tipo de prioridade do *interrupt* do *Buffer* de recepção da USART.

- *Interrupt Timer1* alta prioridade

INTCON REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
bit 7							bit 0

No registo INTCON configura-se, entre outros, o funcionamento do *interrupt* do *timer0*, os seguintes bits são utilizados para esse efeito:

TMR0IE - activa o funcionamento do *interrupt* do *timer0*
TMR0IF - *overflow interrupt flag bit*, sempre que ocorre um *interrupt* passa ao estado 1

Relativamente ao timer0 efectuou-se a activação do seu *interrupt*, são efectuadas ainda as seguintes configurações neste registo:

- Activação global dos *Interrupts*
- Activação global dos *Interrupts* periféricos

PIE1: PERIPHERAL INTERRUPT ENABLE REGISTER 1

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
bit 7							bit 0

Neste registo é activado *interrupt* associado ao *buffer* de chegada de dados via USART (RCIE).

void main(void)

Após se ter apresentado as duas funções anteriores passa se à descrição do funcionamento da rotina principal.
Na rotina principal do código do microcontrolador Master é evocada a função HomePosition(). Esta função serve para inicializar a matriz de posição de todos os servomotores do sistema NovPosServ com os valores de posição que correspondem à

home position do robô. O código desta função não será aqui apresentado dado não haver nenhuma particularidade em especial que mereça ser mencionada.
Após esta inicialização é efectuada a configuração do CAN e dos restantes periféricos do Microcontrolador.
Como já anteriormente no Slave a rotina principal do programa Master também termina com um ciclo *While*() infinito. O Microcontrolador fica por assim dizer preso neste *While* enquanto não seja necessário atender às rotinas de serviço dos diversos *interrupts*.

void UsartIsr(void)

Esta é rotina na qual se efectua a leitura do *buffer* de chegada da USART. Dado que se definiu o *interrupt* do *buffer* de chegada da USART como sendo de alta prioridade esta será a rotina que terá prioridade em relação às restantes. O código com os respectivos comentários é apresentado em anexo uma vez ser extenso para o fazer nesta secção.
A funções utilizadas nesta rotina são as seguintes:

- AnalisaMsgUsart
- EnviaSensPes
- EnviaArtPos

AnalisaMsgUsart

Esta função foi desenvolvida para interpretar as mensagens enviadas via comunicação série pela unidade principal de controlo. A estrutura principal desta função é constituída por um *switch* que permite em função do comando enviado executar a tarefa pretendida.
Esta função tem como argumentos de entrada as seguintes variáveis.

- MsgRecebUsart
- NovPosServ[0][0]
- NovVelServ[0][0]
- ValSens[0][0]
- ValSensForca[0][0]
- SensPe[0][0]
-

Esta função não tem argumentos de saída.

EnviaSensPes

Esta função foi desenvolvida para retirar os valores adquiridos das matrizes de registo de um dado SLAVE e enviar estes mesmos valores pela USART para a unidade principal de controlo. Esta função é evocada pela anterior sempre que se envie o respectivo comando de leitura.
Os argumentos de entrada desta função as seguintes variáveis.

- NumMCU
- SensPe

Esta função não possui argumentos de saída.

EnviaArtPos

Esta função envia os valores de posição de uma dada articulação pela USART para unidade principal de controlo. Esta função é evocada pela função AnalisaMsgUsart sempre que se envie o respectivo comando.
Os argumentos de entrada desta função são as seguintes variáveis.

- NumMCU
- NumServo
- ValSens

Esta função não possui argumentos de saída.

void CanTimerIsr **(void)**

Nesta rotina é efectuado o envio de mensagens do Master para os Slaves a uma frequência de 11 KHz. Para além do envio é efectuada também a recepção de mensagens vindas dos Slaves.

As funções utilizadas nesta rotina são as seguintes:

- CANReceiveMessage
- AnalisaMsgCan
- CANSendMessage

CANReceiveMessage

Esta função vem incluída na livraria de comunicações CAN da Microchip e implementa todos os eventos necessários para obtermos a mensagem pretendida.
Os argumentos de entrada desta função são as seguintes varáveis.

- RemoteID
- MsgRecebCan,
- CompMsgCan,
- FlagsMsg

Esta função não possui argumentos de saída.

AnalisaMsgCan

Esta função interpreta as mensagens recebidas pelo CAN e efectua os registos nos arrays correctos.
Os argumentos de entrada desta função são as seguintes variáveis.

- MsgRecebCan

- CompMsgCan
- ValSens[0][0],
- ValSensForca[0][0]
- SensPe[0][0]
- MCU_PE

Esta função não possui argumentos de saída.

CANSendMessage

Esta função vem incluída na livraria de comunicações CAN da Microchip e implementa todos os eventos necessários para enviarmos a mensagem pretendida.
Os argumentos de entrada desta função são as seguintes varáveis.

- McuActual
- MsgEnvCan
- CAN_TX_FLAGS

Esta função não possui argumentos de saída

3.2.2 Enumeração e explicação de todas as variáveis

#define FREQ_OSC 40	Define a frequência do oscilador em Megahertz
#define BAUD 19200	Define a baud rate para a comunicação série.
#define MCU_PE (byte)2	Define o número do primeiro microcontrolador da rede de comunicações.
byte HostMcu = 0x01	Número do actual microcontrolador.
byte McuActual = 2	Contador que vai indexando os Escravos a serem contactados.
matriz1 NovPosServ	Array que contem os valores de posição de todos os motores do sistema.
matriz1 NovVelServ	Array que contem os valores de velocidade de todos os motores do sistema.
matriz1 ValSens	Array que contem os valores dos potenciómetros dos motores de todo o sistema.
matriz2 SensPe	Array que contem os valores dos sensores adicionais de todo o sistema.

4. Sensores e Percepção

4.1 Desenvolvimento de um pé sensível á força (Sensores de força)

Com vista a dar seguimento ao trabalho desenvolvido no ano anterior, propôs-se o desenvolvimento dos sensores de força, aproveitando a característica dos materiais em deformarem-se proporcionalmente quando se lhes é aplicada uma determinada força. A proporcionalidade, na verdade, só é garantida na zona elástica do material, como podemos ver no gráfico tensão/extensão na fig.1. Só dentro desta zona garantimos que o material recue sempre ao seu estado inicial após uma deformação e que assegure que o valor da deformação será sempre o igual para a mesma força imposta. Como se torna obvio, ao medir a deformação conseguimos medir a força.

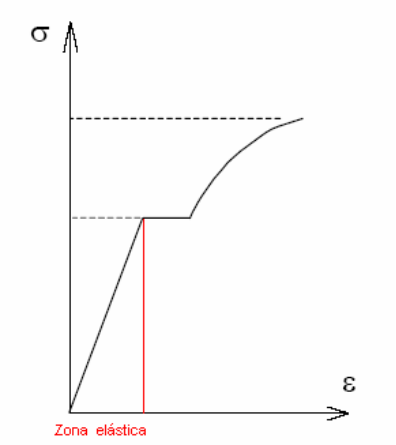


Figura 13 - Gráfico Tensão-Extensão

Para medir a deformação optou-se por usar extensómetros. Extensómetros são resistências cujo valor ohmico varia quando sujeitas a alongamentos. O método usado para medir este aumento é através de uma ponte de *Wheatstone* onde uma das resistências é substituída pelo extensómetro. A ponte permite medir uma diferença de potencial aplicada nos seus extremos. De notar que os extensómetros normalmente deformam-se cerca de 2 a 5% do seu valor, estando assim limitados a medir pequenas deformações. Sendo assim necessário a amplificação á saída da ponte.

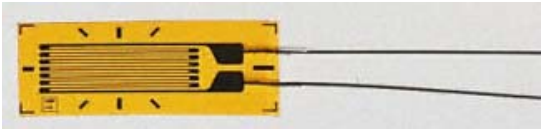


Figura 14 - Extensómetro

4.1.1 Estado dos sensores

Visto estarmos a dar continuidade a um projecto já existente, acabamos por herdar já algum material que foi objecto de estudo. Numa primeira análise ao que encontramos, havia já sido desenvolvido:

- Uma placa electrónica em que se conseguia obter um sinal de saída já com amplificação;

- Um extensómetro colado numa placa de metal - este extensómetro fazia parte da ponte montada na placa, e já se conseguia ver que ao deforma-mos o material o sinal de saída variava;
- Havia já modelado um primeiro protótipo para acoplar á plataforma, com quatro zonas de medida das forças (nas quatro extremidades dos pés).

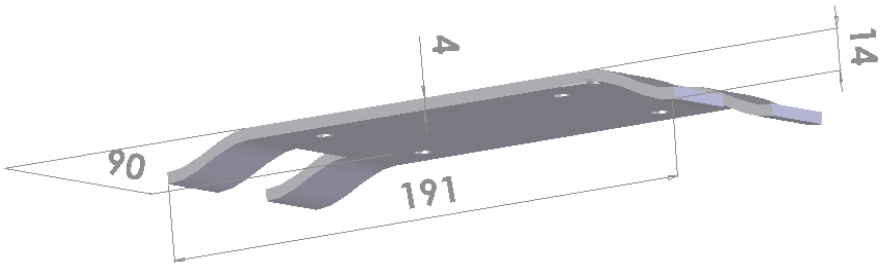


Figura 15 – Protótipo modelado no ano anterior

Após efectuar algumas experiências com o material desenvolvido, tomou-se a decisão de abandonar a maioria das metodologias anteriormente iniciadas. A electrónica desenvolvida, denotou-se com algumas deficiências ao nível do afinamento do sinal de saída, para alem que o protótipo por eles apresentado revelava algumas reservas:

- Como ser feito em metal (elevado peso);
- Complexa geometria (difícil de obter);
- Cara manutenção (bastava uma pequena danificação que era necessário substituir todo o sensor).

Contudo, a análise efectuada acabou por ser de grande utilidade, já que permitiu pôr a descoberto as principais questões a ser objecto de resolução e estudo.

- Solução de baixo peso;
- Geometria de formas fáceis de obter;
- Flexibilidade na substituição de diferentes peças;
- Rectificação de alguns pormenores a nível electrónico;
- Necessidade de quatro sensores em cada pé, para melhor leitura dos valores das forças ao longo do pé.

4.1.2 Desenvolvimento do protótipo

Após o abandono da solução anterior, optou-se por idealizar um novo protótipo de características diferentes, de modo a dar resposta as questões levantadas com o antigo.

A primeira ideia que ocorreu foi ter uma pequena peça apoiada nas suas extremidades, a força era aplicada no seu centro deformando assim o material. O extensómetro era colocado na face oposta á aplicação da força, de modo a permitir a medição da deformação. Sendo o pé de forma rectangular, a ideia era ter uma placa com a forma do pé com quatro rasgos nas extremidades, como se pode ver na figura abaixo, de modo a fixar quatro das placas com os extensómetros apoiadas nas extremidades dos rasgos. A figura abaixo ilustra a primeira ideia.

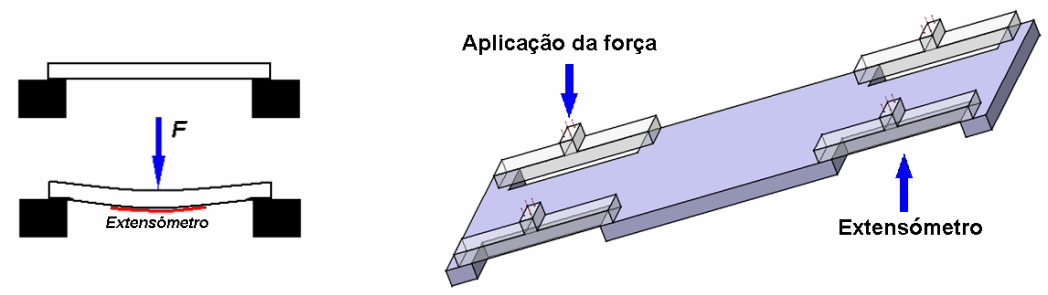


Figura 16 - 1º Ideia do sensor

A questão que se colocava a seguir era qual seria os materiais a usar nos dois tipos de peças. No caso das pequenas placas que levariam os extensómetros, a escolha começou por recair em polímeros devido à sua elevada rigidez e pequeno peso. Após a análise de alguns polímeros de forma muito primitiva (exercendo pequenas forças com os dedos), a primeira escolha recaiu sobre um acrílico existente no L.A.R, sem contudo termos grandes indicações técnicas. Uma pequena pesquisa na Internet sobre as características de alguns acrílicos revelou que a maioria dos acrílicos se encontravam dentro das características pretendidas. O primeiro estudo incidiu em colar um extensómetro numa pequena placa com as dimensões 60x15x4 e com as próprias mãos deformar o material e ver a variação do sinal. Constatou-se que com pequenas forças era possível deformar o material e obter bons resultados mediante os ganhos usados no amplificador.

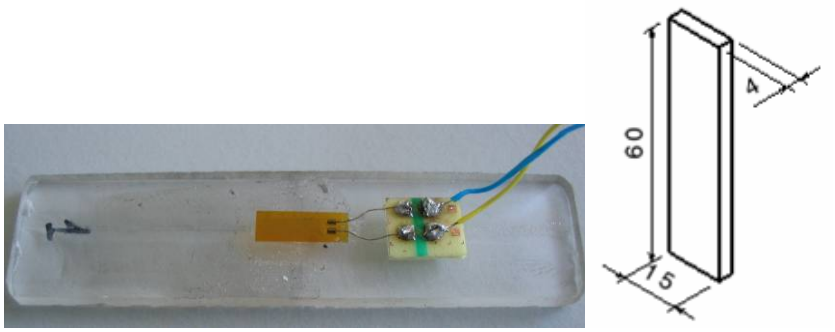


Figura 17 – Extensómetro colado na placa de acrílico

No caso da peça que iria suportar as placas de acrílico (o pé mais propriamente), não havia grandes restrições à sua escolha, apenas que fosse de uma boa rigidez e de baixo peso. Optou-se por usar o mesmo polímero usado pelos nossos colegas na construção dos elos.
Nesta altura tinha sido idealizado um primeiro protótipo, faltando a construção e melhoramento da electrónica existente bem como a construção de uma placa electrónica para efectuar testes.
Na construção do protótipo tendo como base a primeira ideia, o estudo incidiu nas seguintes questões:

- Sistema de acoplamento á plataforma;
- Estudo da transmissão da força para as pequenas placas de acrílico;
- Estudo das dimensões das placas de acrílico mais apropriadas para obtenção de resultados contundentes.

O primeiro passo foi a modelação recorrendo ao software CATIA, como se pode ver nas figuras 19 e 20. Após a modelação e obtenção dos desenhos técnicos procedeu-se à construção nas oficinas do DEM.
O protótipo desenvolvido dando é formado por duas partes, a primeira é a estrutura superior do pé onde são colocados quatro parafusos (nas extremidades) que transmitem a força à segunda parte que é constituída pelas chapas de acrílico (uma em cada extremidade), que estão encaixadas num "segundo pé" construído para o efeito.
Contudo, a solução dos parafusos não foi a primeira. Primeiramente tinha-se um sistema de transmissão da força feito com umas pequenas peças coladas na própria placa de acrílico. Esta solução revelou-se imprópria devido ao aumento da resistência da placa, este aumento da força era incutido pela pequena peça colada na superfície. Logo, houve a necessidade de procurar novas soluções, recaindo na escolha pelos parafusos, visto ser uma solução de rápida implementação e flexibilidade.

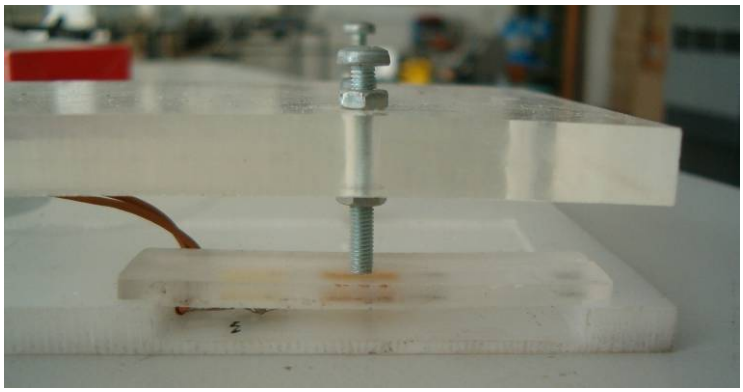


Figura 18 – pormenor do sistema de transmissão de força do protótipo construído

Após alguns testes com pesos conhecidos, as dimensões das placas de acrílicos também se mostraram incorrectas, o sistema não apresentava a sensibilidade necessária às forças necessárias medir. Para isso, foi necessário reduzir um milímetro de espessura nas placas.

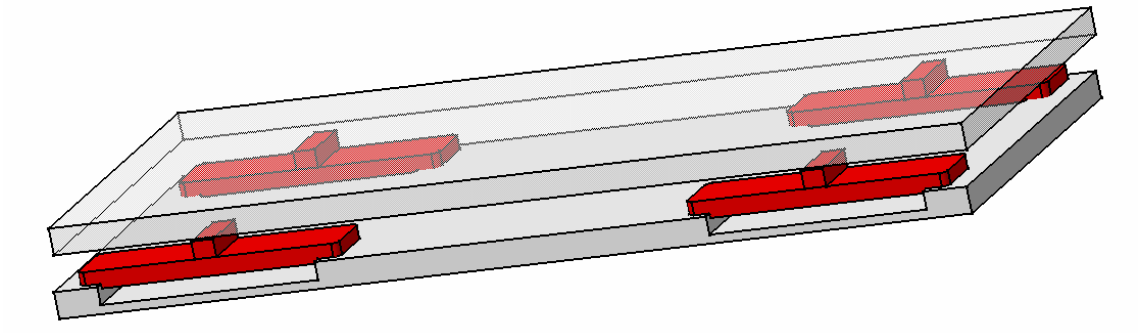


Figura 19 – modelação (primeiro protótipo construído)

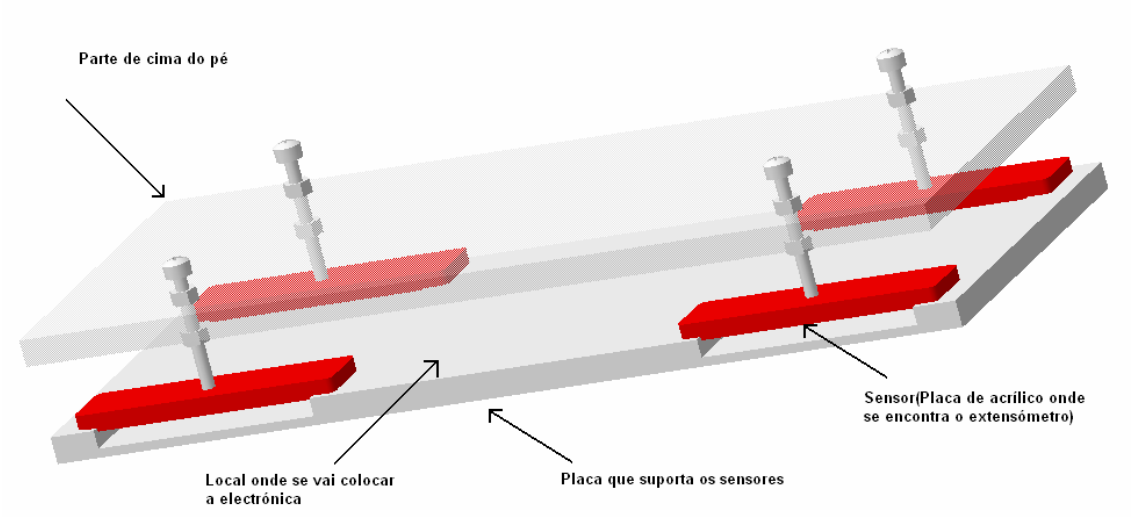


Figura 20 - modelação protótipo final

Esta solução foi testada com resultados satisfatórios, ou seja, com pequenas variações de pesos conhecidos, o sistema já apresentava a sensibilidade pretendida.

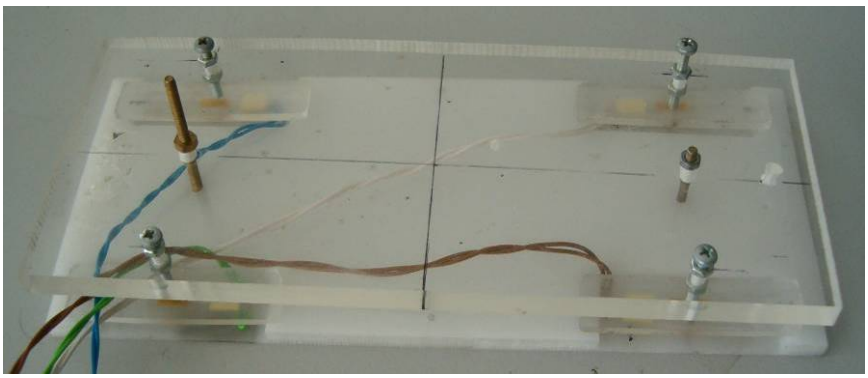


Figura 21 – Foto protótipo construído

4.1.3 Circuito Electrónico

Como já foi referido anteriormente, houve necessidade de refazer o circuito electrónico, as principais alterações foram:

- Substituição do amplificador A620 pelo INA219, apesar de ambos terem os mesmos requisitos, INA129 apresentou melhor desempenho.
- Toda a construção da ponte de *Wheastone* foi modificada, optou-se apenas por colocar um potenciómetro em série com umas das resistências, para possibilitar ajustar o sinal à saída do amplificador.

A vantagem do uso do INA129/A620 é permitirem usar uma resistência externa de regulação do ganho. No INA129 a formula do ganho é: $G = \frac{1 + 49.4}{R_G}$, e permite ganhos até 10000. O uso de diferentes ganhos permite conseguir regular mais facilmente a gama de valores que se quer medir.
Pode-se ver nas figuras abaixo as diferenças entre o circuito herdado e o que se usa actualmente.

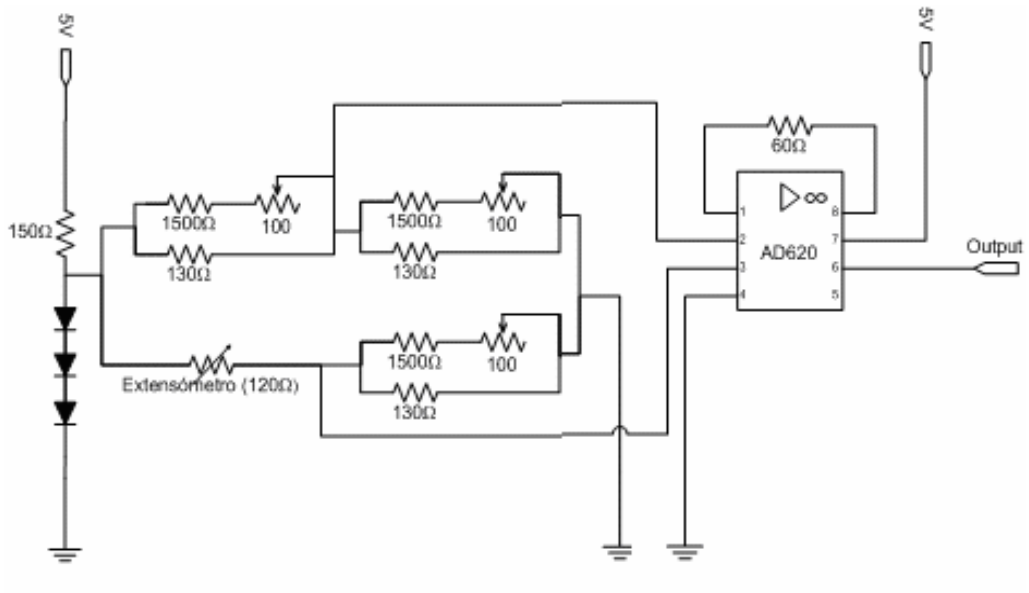


Figura 22 – Esquema eléctrico Ponte de wheatstone e Opamp(AD620), 2003_2004

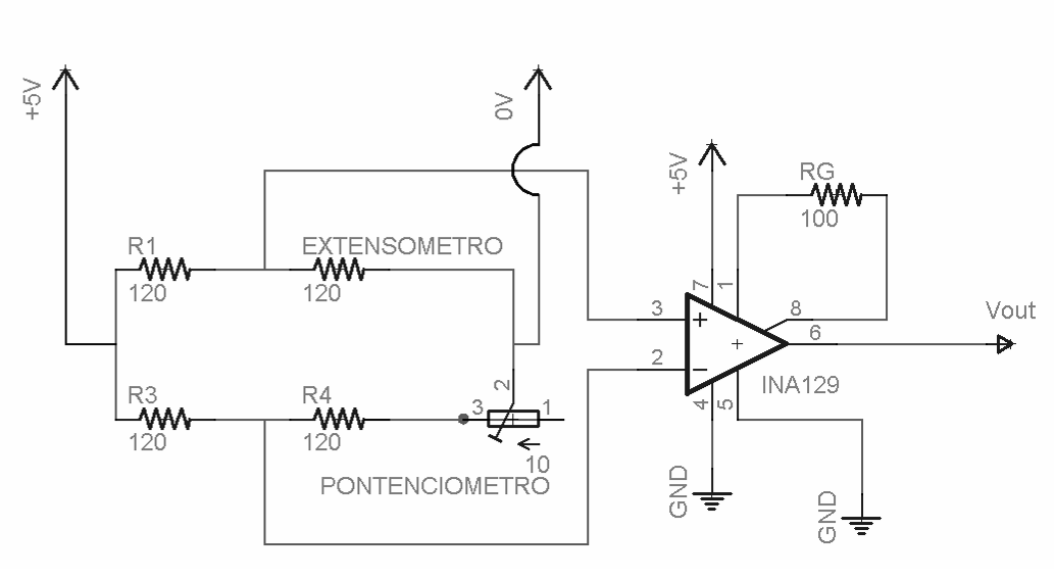


Figura 23 – Esquema eléctrico Ponte de wheatstone e Opamp(INA129)

Construiu-se uma placa electrónica que contêm quatro circuitos acima mencionados, para conseguir adquirir os valores provenientes de quatro sensores. Esta placa teve uma construção pensada, os cuidados a ter foram:

- Dimensões limitadas ao espaço onde iria ser colocada placa electrónica, entre a parte debaixo do pé, que suporta os sensores e a parte de cima onde é transmitida a força;
- A colocação dos potenciómetros de modo a fácil acesso de regulação;
- Foram usados uns pequenos pinos para facilmente substituir as resistências do ganho.

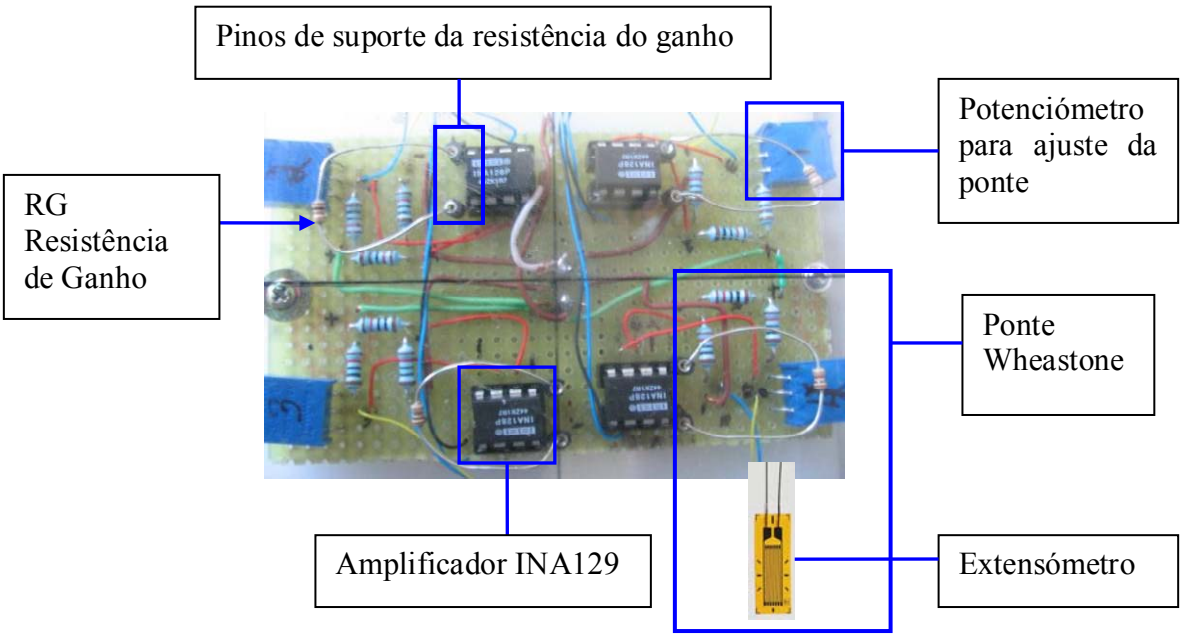


Figura 24 – Foto da Electrónica do circuito amplificador dos 4 Extensómetros

4.1.4 Processo de Calibração experimental

Antes de iniciar propriamente a calibração era necessário encontrar qual o melhor ganho a usar. O aspecto principal a ter em conta era o peso final da plataforma, permitindo assim ter uma indicação da força máxima que o sensor necessita de medir. Após a realização de alguns testes com diferentes ganhos optou-se por um ganho de 500 (Resistência de $100\,\Omega$).

Constatou-se que para o bom funcionamento do sensor era necessário que a posição de repouso (sem forças aplicadas) as placas de acrílico tivessem um pré-esforço, só assim se conseguia obter resultados contundentes. O método usado foi através da fixação da parte de cima á parte de baixo, em que o uso de parafusos possibilita aplicar uma pequena pré carga sobre as placas.

Para a calibração, foram usados diversos pesos conhecidos para os quais se registaram os valores para os quatro sensores, estes valores foram obtidos com o auxílio de um microcontrolador. O microcontrolador encarregou-se de converter os sinais analógicos em digitais, assim conseguimos integrar a parte de percepção com a parte de aquisição de dados. Acabou por ser um teste a ambas as partes, executada com sucesso.

Os dados foram adquiridos por um PC vindos do microcontrolador (via Rs232) recorrendo a uma aplicação em MatLab, esta aplicação permitiu adquirir para cada peso 10 medidas diferentes calculando de imediato a média dos valores.

Os gráficos seguintes ilustram os resultados obtidos para os quatro extensómetros.

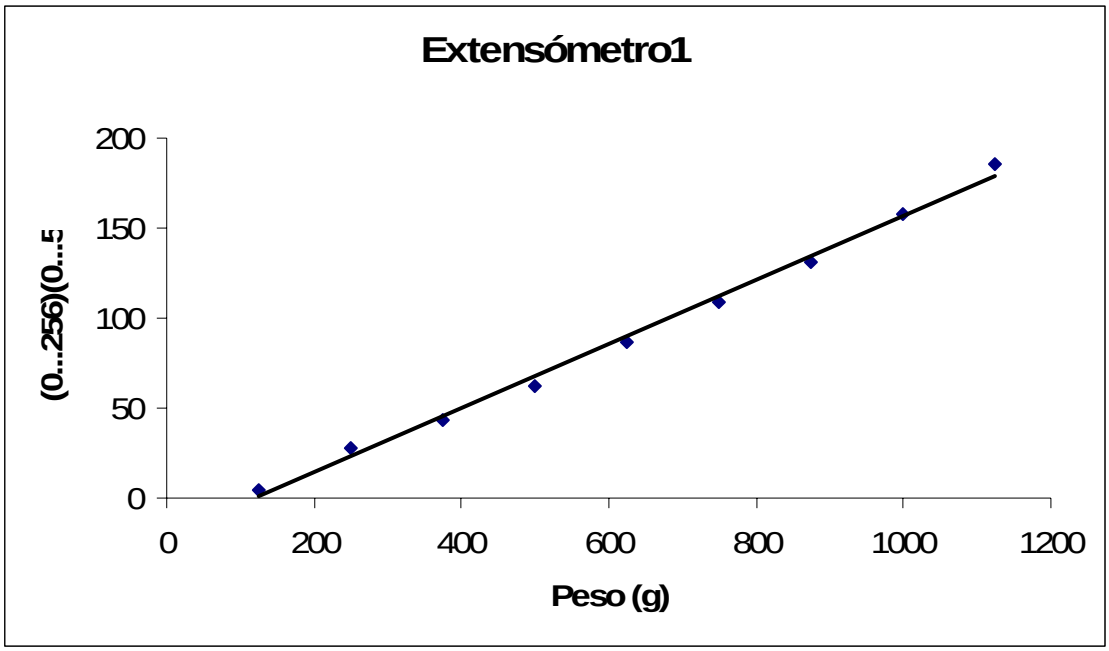


Gráfico1

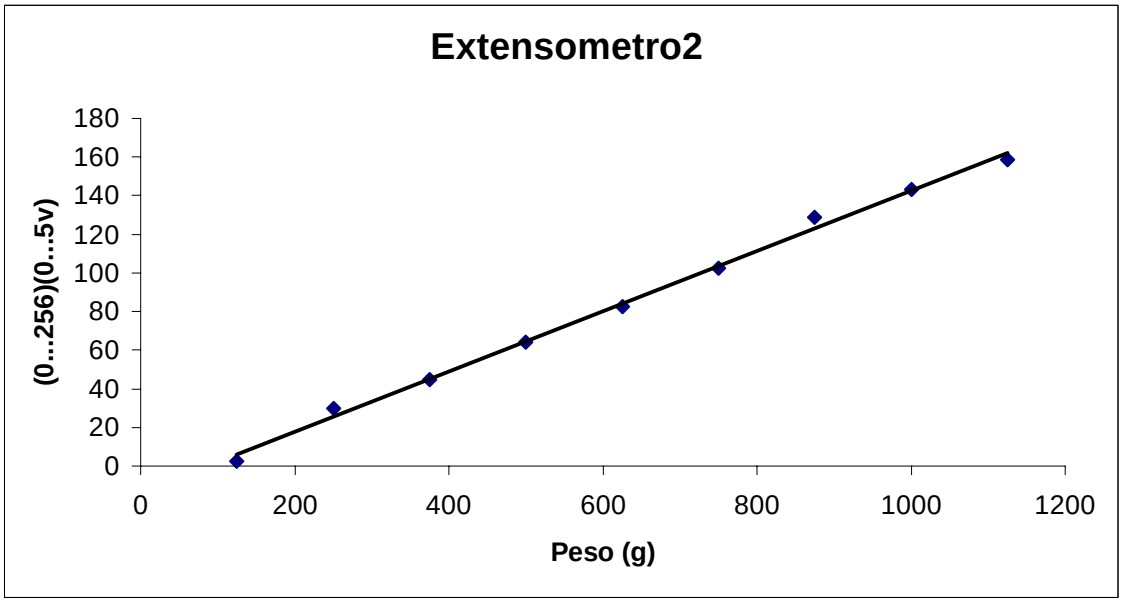


Gráfico 2

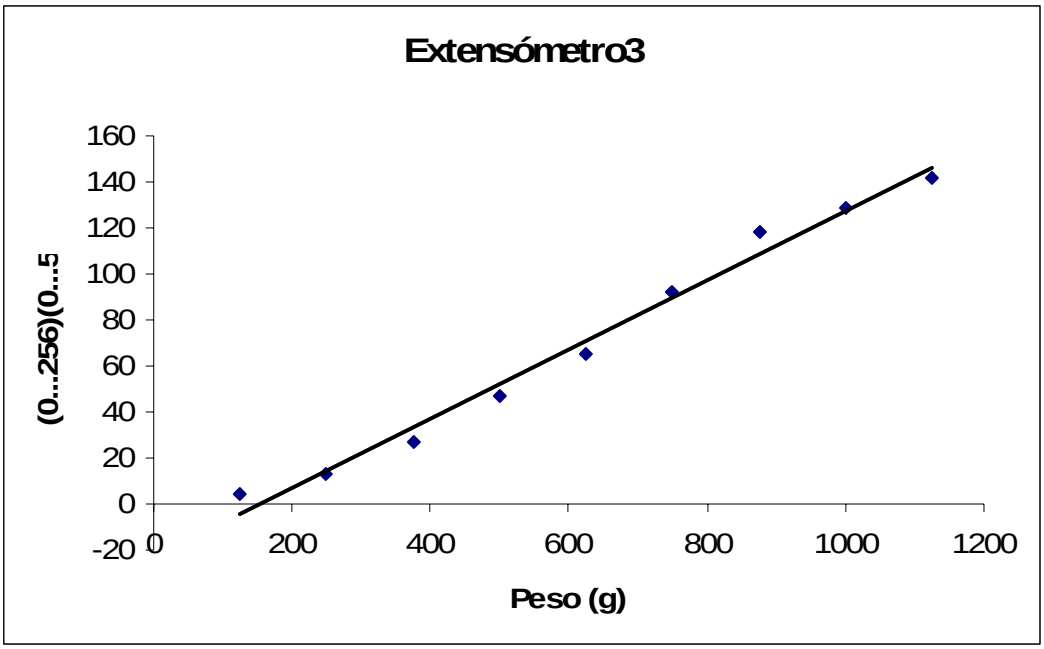


Gráfico 3

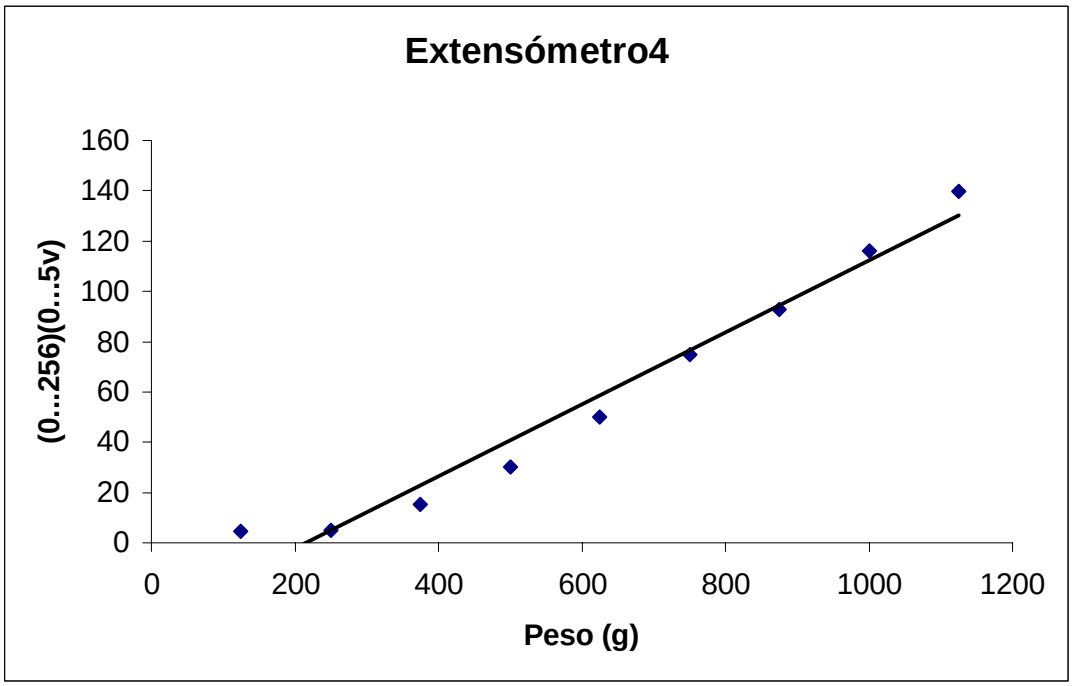


Gráfico 4

4.1.5 Conclusões da calibração

Pela análise dos gráficos constata-se que existe uma relação praticamente linear entre o peso (força) e os valores de tensão, ou seja, o material sofre uma deformação proporcional à força aplicada. Os coeficientes de correlação para os 4 extensómetros variaram entre 0.983 e 0.998 como podemos verificar na seguinte tabela. Estes resultados vêm demonstrar a validade do sensor, bem como todas as metodologias usadas para o desenvolver.

Extensómetro	Coeficiente de correlação
1	R=0.99736
2	R=0.99823
3	R=0.99318
4	R=0.98352

4.2 Inclinómetro

Na realidade não vamos usar um inclinómetro mas sim um acelerómetro. Uma das aplicações mais conhecidas dos acelerómetros é a possibilidade de poder medir inclinações. O acelerómetro usa a força da gravidade como um vector de entrada para determinar a orientação de um objecto no espaço. Um acelerómetro é tanto mais sensível à inclinação, quando a sua linha central é perpendicular à força da gravidade, isto é, paralela à superfície da terra.
No ano transacto já havia sido efectuado uma pesquisa e escolha do acelerómetro a usar, *ADXL202E* da *Analog Devices* foi o seleccionado. Também foram iniciados alguns estudos que foram continuados este ano.

4.2.1 Características do ADXL202E

O *ADXL202E* é um acelerómetro de 2 eixos (X e Y), com uma escala de medida de ±2 *g*. Permite a medição de acelerações dinâmicas (e.g vibrações) e acelerações estáticas (e.g gravidade).

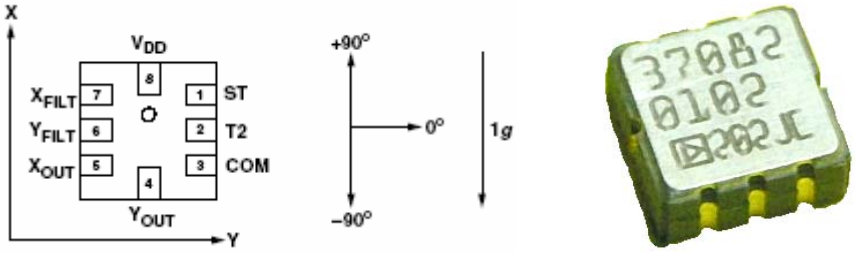


Figura 25 – Inclinómetro

Para o uso dos acelerómetros como inclinómetros é necessário fazer a conversão pelas fórmulas:
Pitch = *ASIN* (*Ax*/1 *g*)
Roll = *ASIN* (*Ay*/1 *g*)
Onde Ax e Ay são as leituras de aceleração nos dois eixos do sensor.

O *ADXL202E* tem dois tipos de saídas, *Duty Cycle Modulated* (DCM), e saídas analógicas.
DCM (pinos, X_{OUT} Y_{OUT}) é uma saída em PWM (pulsewidth modulation), em que varia o *duty cicle* e é proporcional à aceleração. O período *duty cicle* é ajustável entre 0.5 e 10 ms usando para o efeito uma resistência (R_{SET}). As saídas analógicas X_{FILT} e Y_{FILT} têm uma impedância à saída de 32KΩ em geral carecem de amplificação. A largura de faixa é ajustável com condensadores C_X e C_Y nos pinos X_{FILT} e Y_{FILT}.

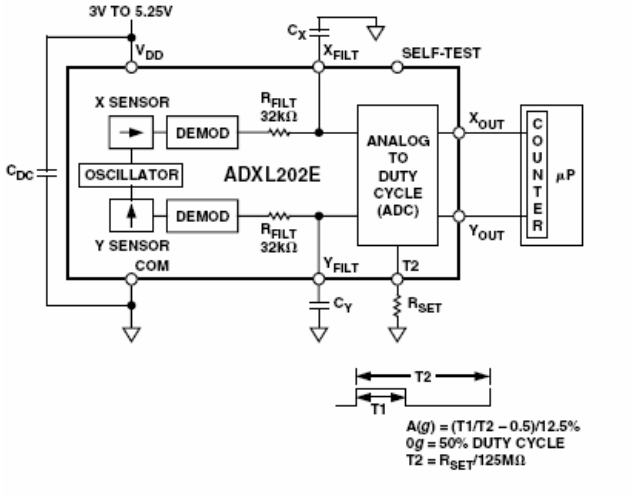


Figura 26 – Diagrama funcional

4.2.2 Desenvolvimento do circuito electrónico

Optou-se pelas saídas analógicas (X_{FILT} Y_{FILT}) com C_X e C_Y iguais a 1μF e R_{set} a 1KΩ.
Os sinais de saída (X_{FILT} e Y_{FILT}) para 0º é da ordem dos ±2.5V, havendo por isso necessidade de utilizar um divisor tensão com um potenciómetro, para que a diferença á entrada do amplificador seja mínima na posição 0º assim consegue-se melhores leituras de valores.

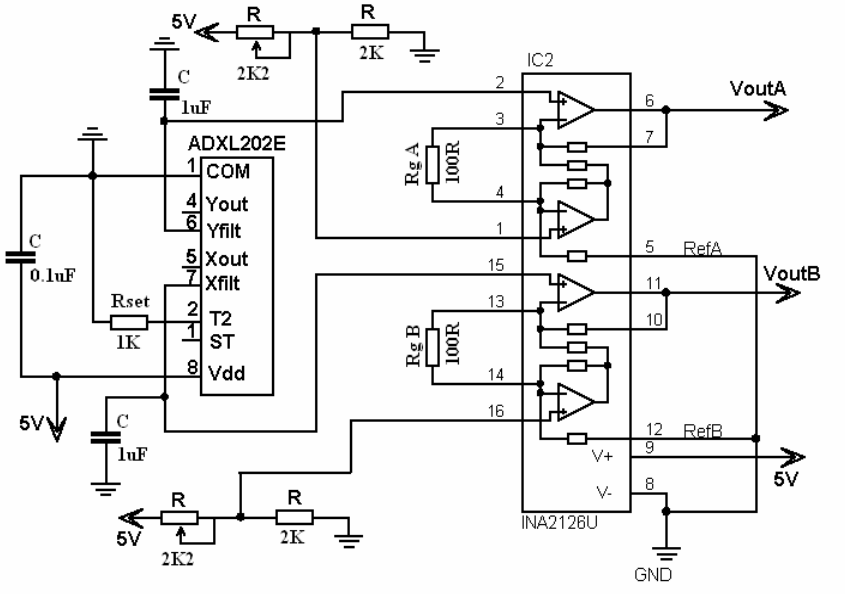


Figura 27 – Esquema eléctrico usado para adquirir valores do ADLX202E

Com este esquema construí-se uma placa e realizou-se algumas experiências. Ao inclinar a placa o sinal aumenta e diminui conforme a inclinação. De notar que temos inclinações em dois sentidos, no positivo e no negativo para cada eixo. Quando se afina

o sinal de entrada no amplificador convêm que o de saída a 0º tenha o valor de 2,5 volts, isto é necessário para se poder medir as inclinações nos dois sentidos.

4.3 Giroscópio

O giroscópio permite medir a velocidade angular usando o fenómeno da força de Coriolis, que é gerada quando é aplicada uma velocidade angular a um corpo em vibração.
Os colegas do ano anterior optaram por adquirir o giroscópio gyrostar (ENJ-03JA) da marca Murata que permite medir velocidades até 300 graus/s.

Especificações:

Part Number	ENC−03J
Supply voltage (Vdc)	+2.7 to +5.5
Current consumption (mA max.)	5
Max. angular velocity (˚/s)	±300
Output (at angular velocity=0) (Vdc)	+1.35
Scale factor (mV/˚/s)	0.67
Temp. coefficient of scale factor (%)	±20
Linearity (%FS)	±5
Response (Hz max.)	50
Operating temperature range (˚C)	−5 to +75
Storage temperature range (˚C)	−30 to +85
Size (mm)	15.5×8.0×4.3
Weight (g max.)	1.0

Foi desenvolvido um circuito electrónico de acordo com alguma pesquisa efectuada pela Internet juntamente com algum do material herdado.

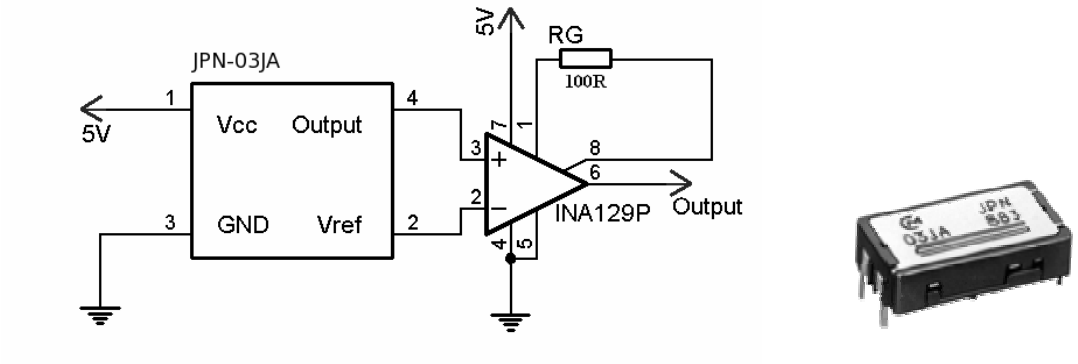


Figura 28 – circuito eléctrico do giroscópio (JPN-03A)

Uma placa foi construída e efectuados alguns testes. Foi usado um motor rotativo onde era possível colocar no veio o giroscópio, ao rodar o motor foi-se medindo a variação do sinal conforme o aumento e diminuir da velocidade angular, o sinal aumentava e diminuía conforme a velocidade imposta.
Contudo os 2 giroscópios que foram comprados avariaram e não foi possível adquirir novos, já que não se consegui encontrar um revendedor em Portugal.

4.4 Potenciómetros

Para conhecer as posições das juntas, tendo em conta que todas elas são rotacionais, optou-se por usar os potenciómetros internos dos motores. Foi necessário abri-los e extrair o sinal para o exterior uma vez que não está disponível no sistema base. Os sinais que saem destes potenciómetros variam entre os 0.85V e 1.75V.

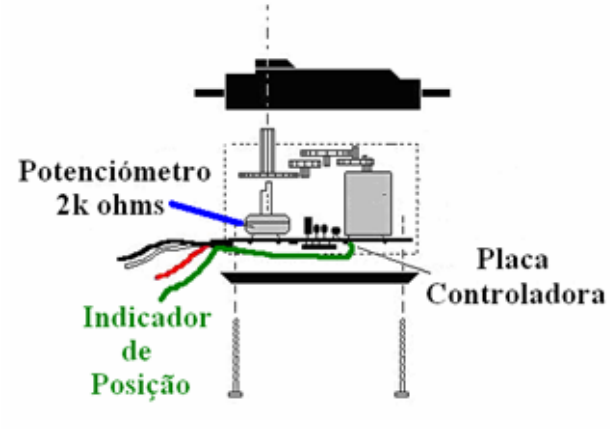


Figura 29 - Esquema do potenciómetro do motor

A informação destes potenciómetros serve para o controlo em velocidade e também como indicador da posição da junta. Embora os servomotores implementarem, internamente, rampas de aceleração e desaceleração, a sua velocidade natural é excessiva para o controlo das juntas, o que obriga a algum tipo de controlo de velocidade.

4.5 Monitorização corrente

Na construção da placa final foi introduzido a possibilidade de medir a corrente através de resistências de potência de baixo valor (0.7Ω) em série com a alimentação do motor, e monitorar essa queda de tensão pelo microcontrolador. Esta informação pretende ser usada para calcular o binário exercido por cada motor. Contudo esta abordagem ainda carece de uma futura calibração.

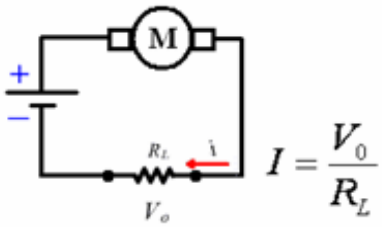


Figura 30 – Circuito monitorização corrente

5. Desenho e concepção das placas de controlo e sensores em PCB

Para desenvolver as placas recorreu-se ao software **EAGLE 4.13r1.** Este software permite o desenho dos circuitos electrónicos e gera automaticamente as pistas dos PCBs. Contêm uma livraria bastante completa, permitindo mesmo assim o desenho de novos componentes.

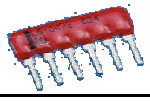
Houve um período aprendizagem e familiarização de modo a explorar ao máximo todas as potencialidades. Em anexo encontra-se um quick start.

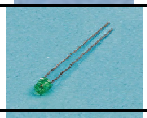
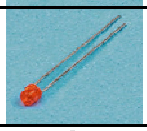
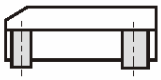
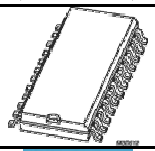
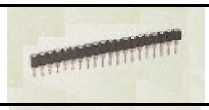
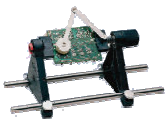
Foram desenvolvidas 6 tipos de placas:

- Placa principal;
- Placa expansão sensores força;
- Placa expansão inclinómetro;
- Placa expansão giroscópio;
- Placa expansão Master;
- Placas Alimentação.

5.1 Componentes usados nas placas

As placas tinham de ser construídas com a menor dimensão possível, o uso de componentes de montagem de superfície era exigido. Houve um processo de selecção dos componentes necessários, para isso recorreu-se ao catálogo da distribuidora *RS Amidata*, onde também foram adquiridos os componentes. Na tabela seguinte encontra-se todos os componentes adquiridos.

Designação	Código Amidata	Referencia fabricante	Imagem
Resistência em Chip 120R	RS 223-2136	1206	
Resistência em Chip 1K	RS 223-2265	1206	
Resistência em Chip 4K7	RS 223-2350	1206	
Resistência em Chip 100R	RS 223-2120	1206	
Array de resistências 10K	RS 241-6435	Série L05 - 1S	
Trimmer 100R, montagem superficial	RS 177-425	Série 3224W	
Trimmer 2K, montagem superficial	RS 100-1149	Série 3314G	
Resistência em Chip de baixo valor ohmico, R47	RS 223-1060	Série RL73	
Fusível 2A miniaturizado	RS 226-0816	Wickmann 395	
Condensador 100nF, superficial, formato 0805	RS 220-7966	0805	

Condensador 1uF, superficial, formato 0805	RS 220-7994	0805	
Condensador 27pF, superficial, formato 0805	RS 237-6810	0805	
Led 3mm, verde	RS 247-1410	L-1334G	
Led 3mm, vermelho	RS 247-1353	L-1334IT	
Interruptor miniaturizado, montagem superficial	RS 228-4547	norma IP67	
Cristal 10MHz, montagem superficial	RS 249-3606	86SMX/SXA	
Multiplexer, SOIC24, 16-1 Analog Mux/DeMux	RS 355-8956	HEF4067BT	
Terminal macho, 4 vias, lateral, passo 2,5mm	RS 311-6388	LR20812 Ou 60389	
Terminal fêmea, 4 vias, passo 2,5mm	RS 311-6221	LR20812 Ou 60389	
Contacto de crimpar para terminais de PCB passo 2,5mm	RS 311-6439	LR20812 Ou 60389	
Pin torneado 32 vias	RS 267-7416	W35532	
Terminal de componentes SIL	RS 227-2639		
Terminal fêmea para PCB, 2 vias, passo 3,81 mm	RS 220-4658	E60425 ou LR13631	
Terminal macho para PCB, horiz., 2 vias, passo 3,81 mm	RS 220-4737	E60425 ou LR13631	
Suporte para PCB's	RS 162-8766		
Manipulador por ventosa	RS 538-886		

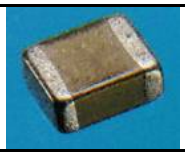
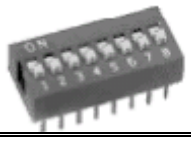
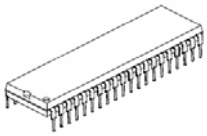
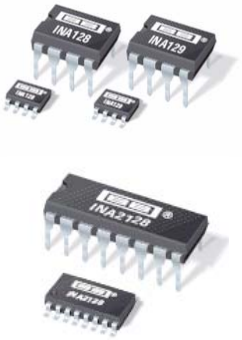
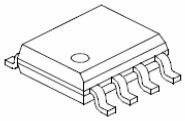
Condensador, série GRM, alta capacidade, 10uF	RS 312-3083	1206	
dip switct	RS 377-8712		

Tabela com os componentes adquiridos na RS Amidata

Foram também adquiridos através da solicitação de amostras, amplificadores, microcontroladores, drives de Can e acelerómetros. A seguinte tabela ilustra os componentes solicitados:

Designação	Marca	Imagem
Microcontrolador ➢ Pic 18f258 ➢ pic 18fl258	Microchip	
Acelerómetro *ADXL202E*	*Analog Devices*	
Amplificadores: ➢ INA129U – SOIC - DIP ➢ INA12UA – SOIC - DIP ➢ INA2128UA- SOIC - DIP ➢ INA2126U- SOIC - DIP ➢ INA2126UA- SOIC - DIP	Texas Instruments	
Drives CAN ➢ MCP2551 - SOIC ➢ MCP2551 - DILL	Microchip	

5.2 Placas PCB desenvolvidas

5.2.1 Placa principal

Esta placa é a principal de todo o sistema de controlo, optou-se por torna-la a mais multi-funcional possível dando assim mais ênfase à metodologia da hierarquia de controlo distribuída um dos emblemas de todo o projecto.

O desenvolvimento desta placa passou por diversas etapas: foi estudado o controlo de 3 motores, comunicação Rs232 entre o pic e o PC, comunicação CAN entre as diversas placas e aquisição de sinais analógicos provenientes de sensores através do uso de um multiplexer. Esta parte foi objecto de estudo ao longo de todo o ano estando portanto toda a electrónica estudada e validada.
Foi proposto que a placa pode-se acomodar placas de expansão, através de um sistema dominado piggy back. Este sistema não é mais que o encaixar de uma placa noutra, para isso usa-se pinos específicos, estes pinos no nosso caso são usados para distribuir alimentação e disponibilizar diversas entradas analógicos às placas de expansão.
Foi também introduzido uma resistência de baixo valor (0.7Ω) em série com a alimentação do motor, de modo a monitorizar a corrente consumida como já foi referido no capítulo acima. Para protecção de cada motor foi introduzido um fusível de 2A, consequentemente introduziu-se um led para monitorizar o seu estado. Para o endereçamento das placas como slaves usou-se um Dip Switch ligado ao microcontrolador. As baterias usadas fornecem uma tensão ±7,5 V, foi então necessário à entrada de cada placa introduzir um regulador de tensão (7805) de modo a alimentar a electrónica com 5V, a placa ficou com dois circuitos de alimentação distintos um para os motores (antes do regulador) e um para a electrónica. Como se previa que os motores consumissem bastante corrente quando sujeitos a grandes binários, houve a preocupação nas pistas de alimentação dos motores aumentar a sua secção, de modo a proteger as pistas de se danificarem com o calor. Introduziu-se um botão de pressão de modo a fazer o reset ao microcontrolador.

As características da placa são:

- Microcontrolador pic 18f258;
  - Com módulo CAN que permite ligar até 16 placas com endereços distintos na rede de controlo.
  - Possibilidade de geração de 3 PWM para controlo de posição e velocidade até três juntas;
- 16 Entradas analógicas possibilitadas pelo uso de um multiplexer;
  - Monitorização da posição e consumo de corrente para cada servomotor;
  - 10 Canais para aquisição de valores sensoriais das placas de expansão (piggy back).
- Comunicação Série Rs232 a uma *baudrate* de 19200.
  - Possível equipar a placa com um módulo para comunicação série.
- Alimentação: 7.5 Volt (Com regulador incorporado 7305, para alimentação da electrónica a 5V).

Após todas as características requeridas estarem listadas foi concebido o desenho electrónico da placa. Desenhou-se um primeiro PCB para testar o circuito.
Apesar de o departamento já ter condições para a produção de PCB, com a nossa técnica não foi possível construir o PCB em questão, a sua complexidade e o facto de ser de dupla face foram um grande entravo. Recorreu-se assim ao departamento de Engenharia Física que construiu a primeira placa. Após soldados todos os componentes foram efectuados os testes necessários para validar o funcionamento do circuito, com resultados positivos.

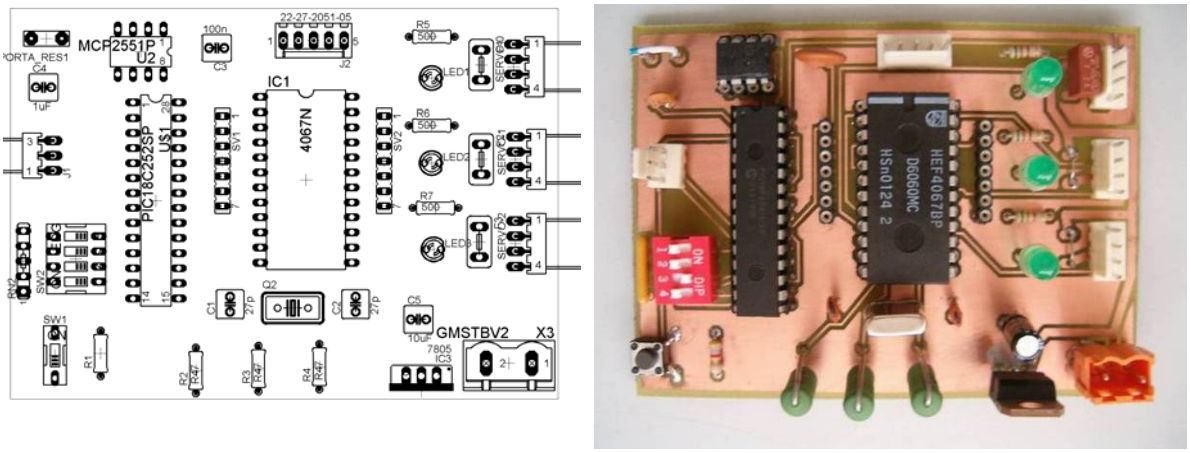
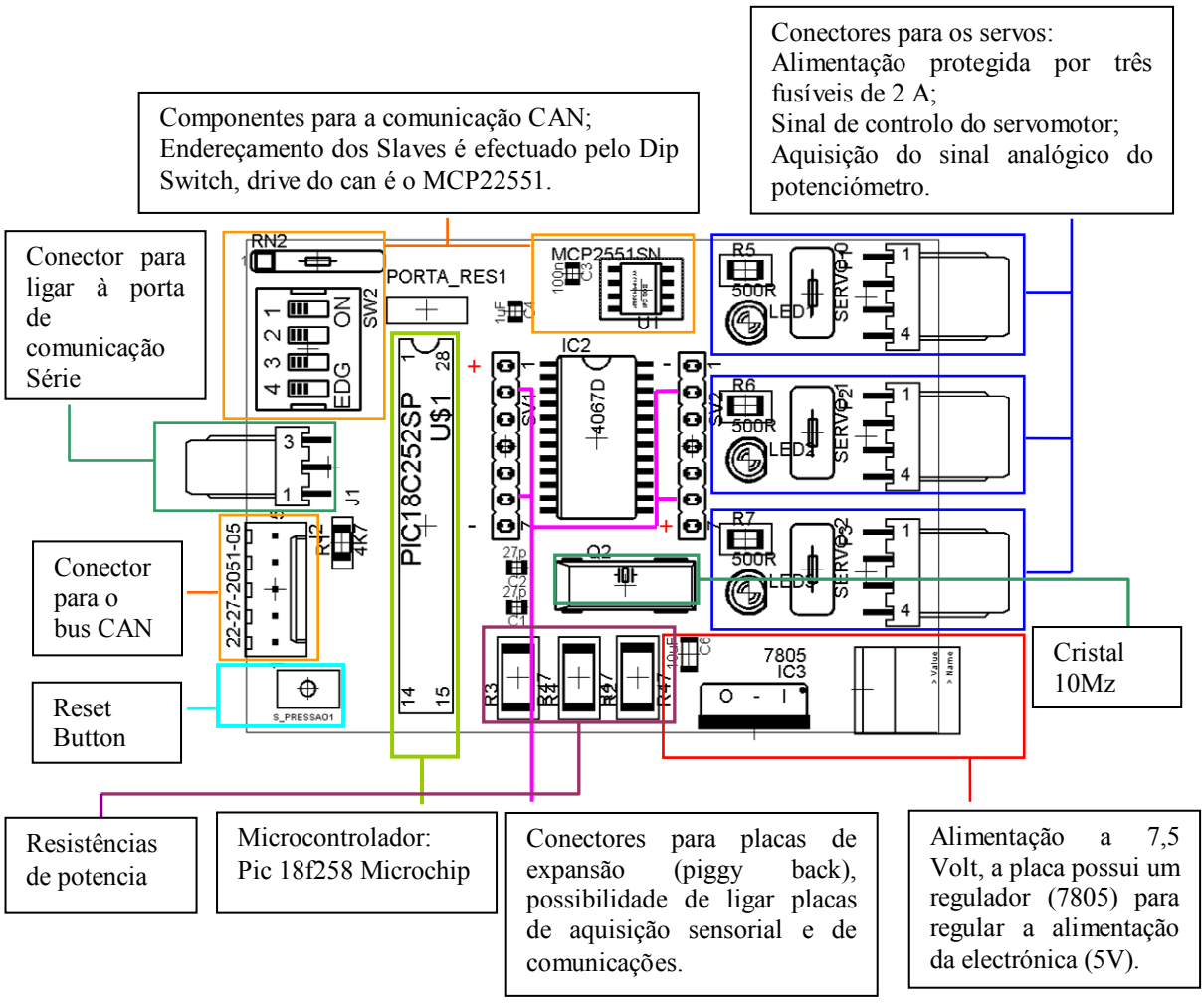


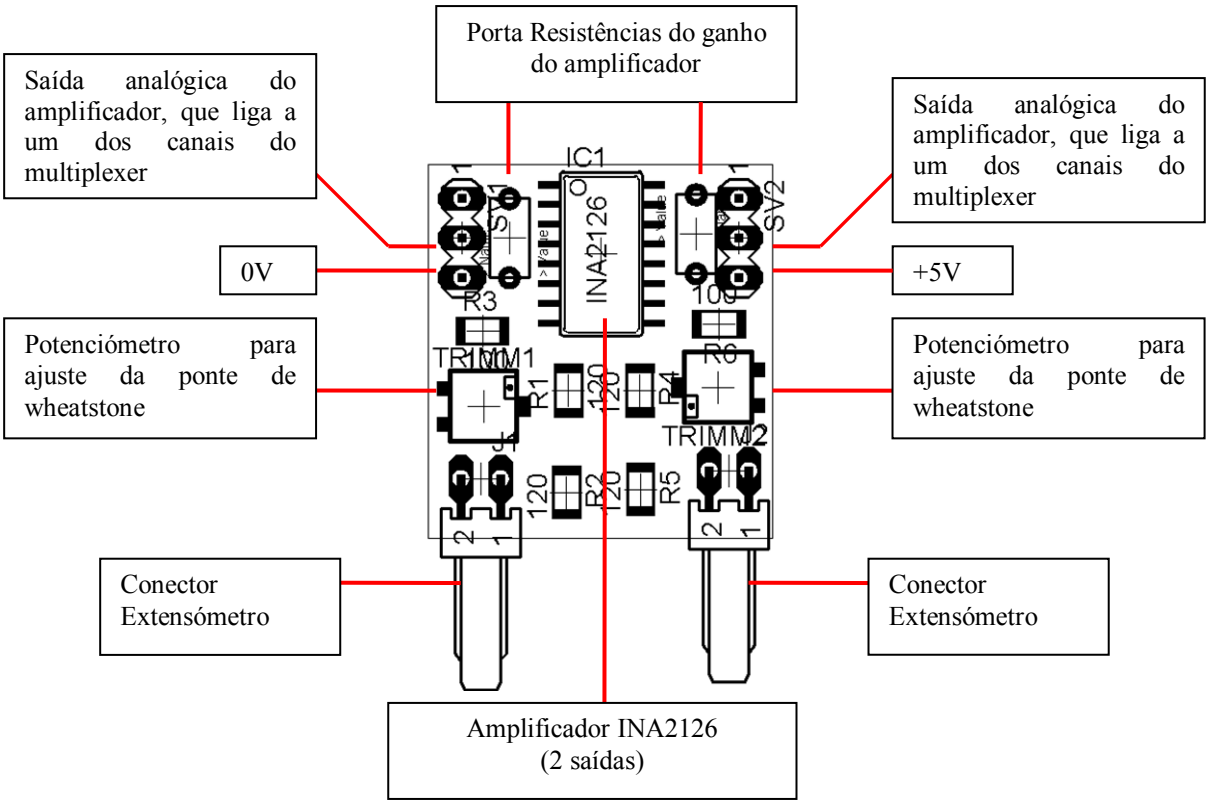
Figura 31 - Placa principal

Com o circuito electrónico validado, desenhou-se a placa final de acordo com os componentes escolhidos, com o uso dos novos componentes conseguiu-se que a placa teve-se a dimensão mínima de 65x50. Ao desenhar a placa teve-se em consideração a distribuição dos componentes de modo a ficarem o mais bem ordenados possível, para a receberem as placas de expansão e para se ligarem os diversos conectores da melhor forma.



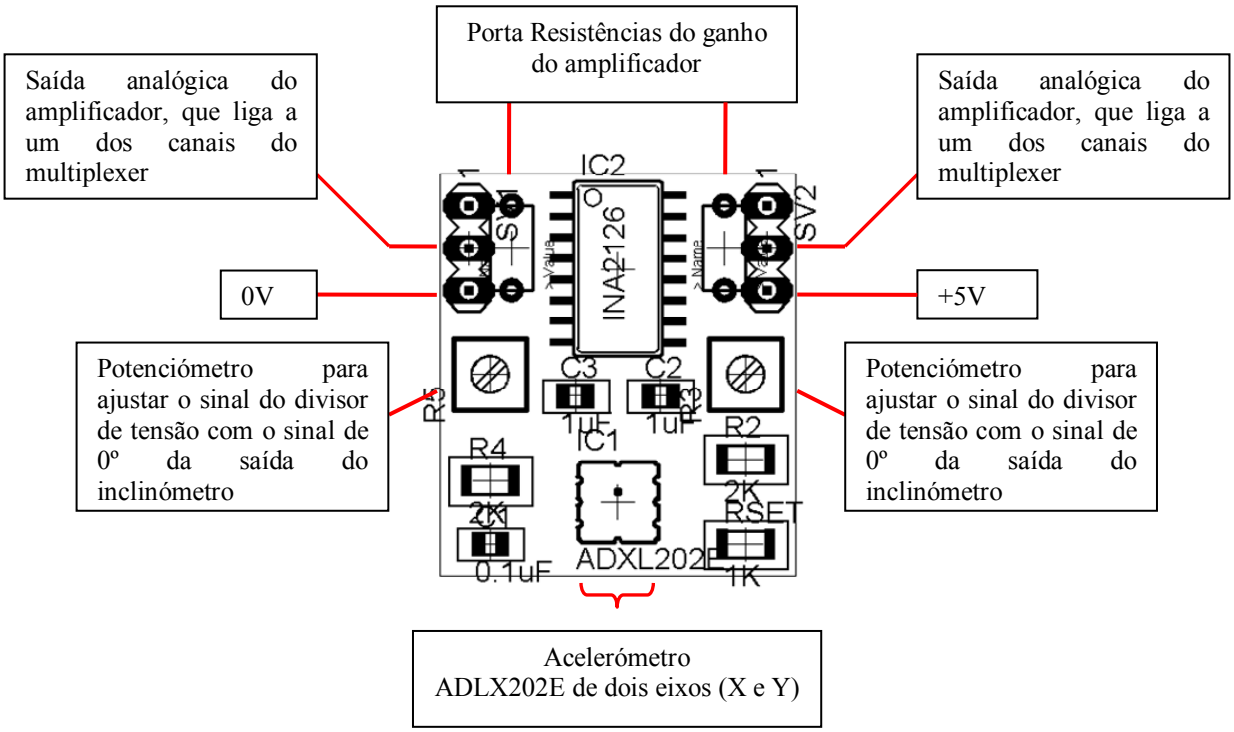
5.2.2 Placa expansão sensores força

A placa foi desenvolvida com base no circuito electrónico desenvolvido para este sensor. Cada placa permite adquirir dois sinais (cada uma tem duas pontes de wheatstone), usou-se um único amplificador com duas entradas distintas. Teve-se em consideração as dimensões de modo a permitir que a placa principal conseguisse albergar duas destas placas. O desenho do circuito electrónico encontra-se em anexo.



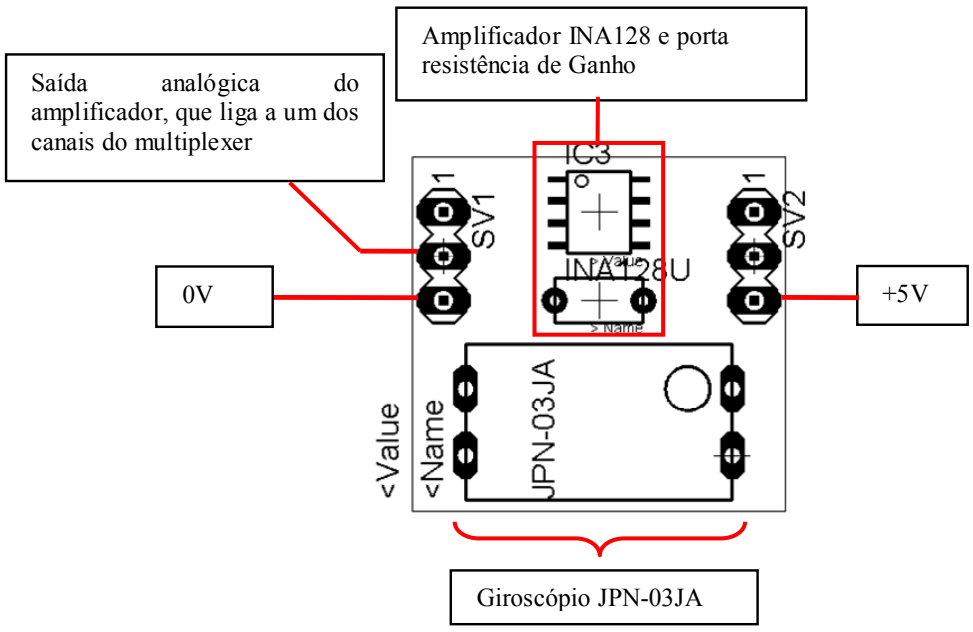
5.2.3 Placa expansão inclinómetro

A placa foi desenvolvida de acordo com o circuito estudado. De referir que o inclinómetro mede inclinações ao longo de dois eixos logo com dois sinais de saída (X e Y). O desenho do circuito encontra-se em anexo.



5.2.4 Placa expansão giroscópio

Esta placa foi a única não construída pelas razões já acima mencionas. Mesmo assim foi elaborado o desenho do PCB.

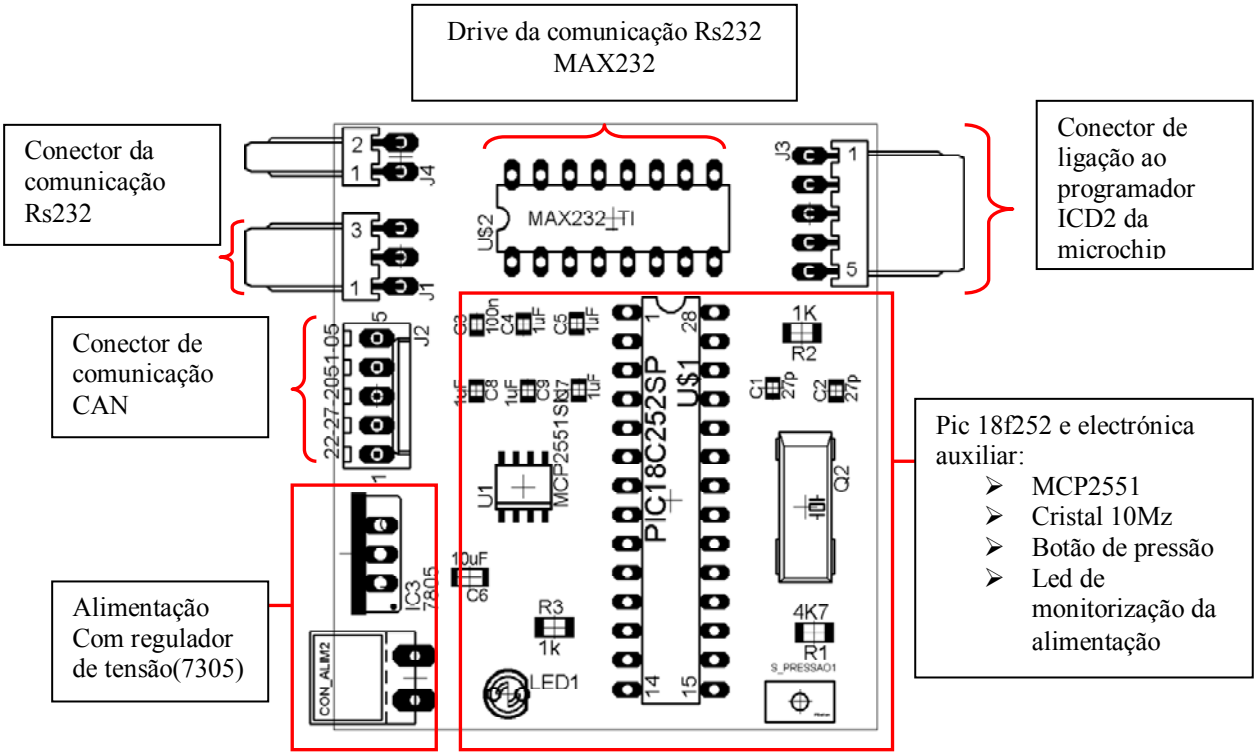


5.2.5 Placa Master

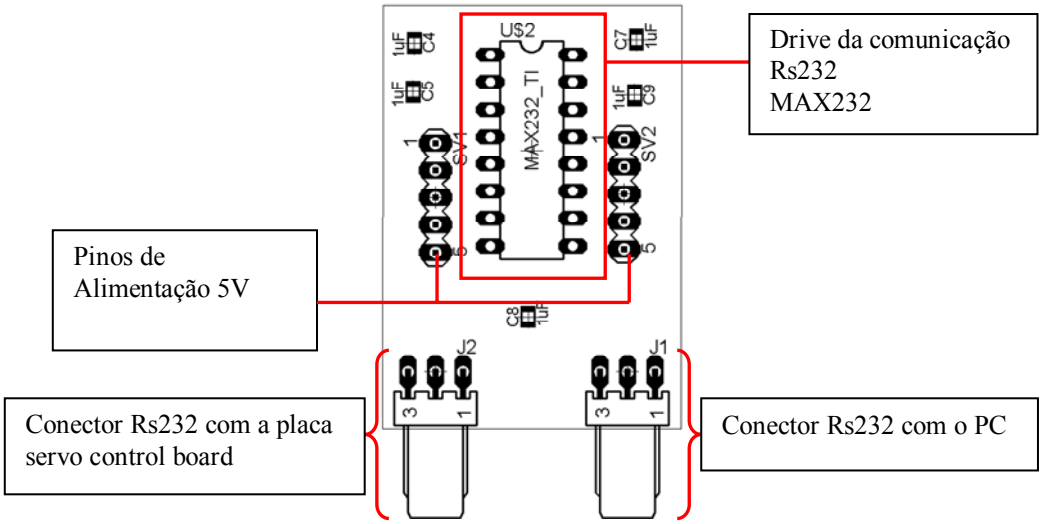
Numa primeira fase desenhou-se uma placa única com o microcontrolador incorporado. Esta placa permitia

- Comunicação Rs232 com o PC;
- Comunicação CAN entre as placas servo control board;
- Permitia fazer o debugging com o programador ICD2 da microchip.

A figura abaixo ilustra o desenho da placa Master desenvolvida.

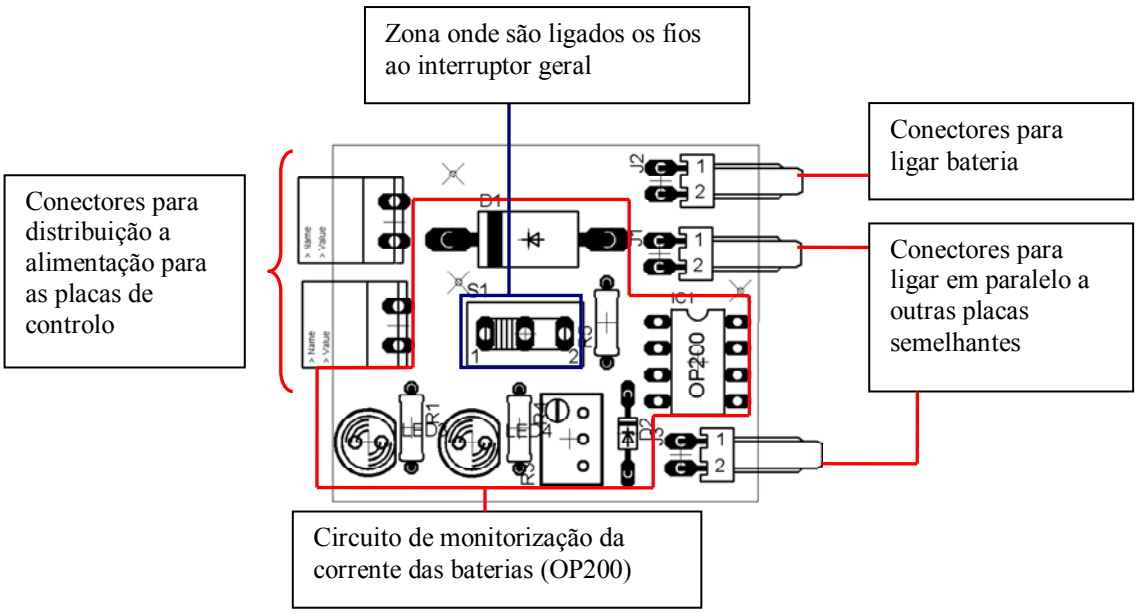


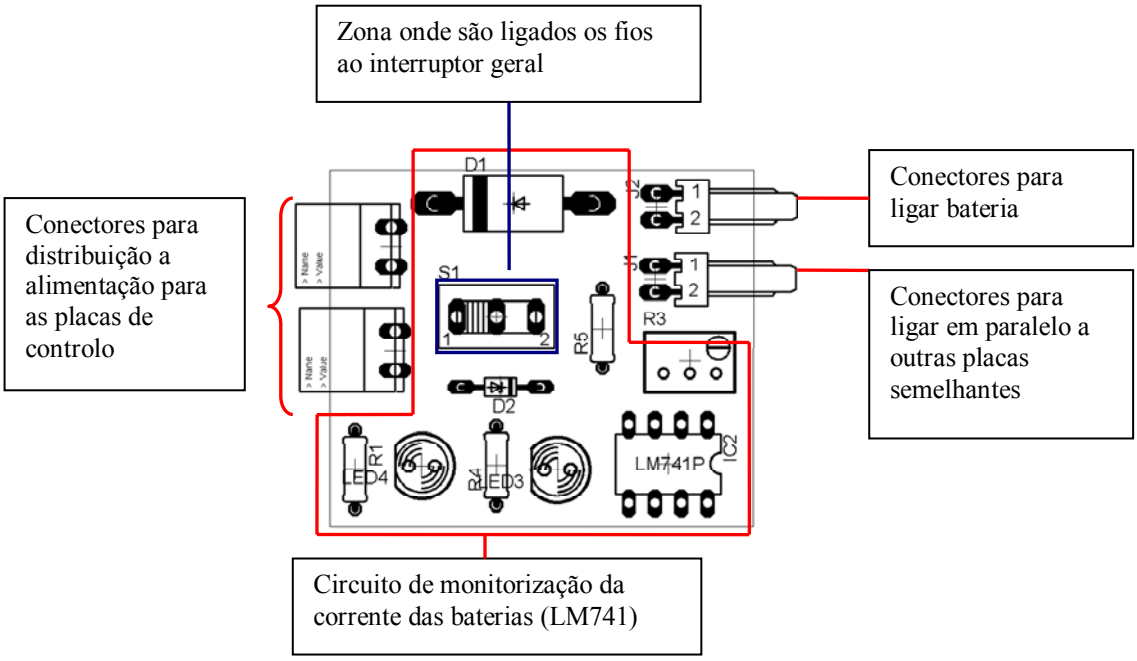
O nosso processo de construção de PCB não nos permitiu realizar esta placa, visto ser uma placa de dupla face e de dimensões reduzidas algo que foi um grande entravo. Após muitas tentativas e desenhos, optou-se por envergar por uma outra alternativa, desenhou-se então uma placa de expansão de modo a permitir juntamente com a placa principal (com o programa do master no microcontrolador) actuassem como master. Esta opção revelou-se um sucesso e veio dar mais ênfase ás opções tomadas do piggy back e multi-funcionalidade desta placa. Contudo não invalida que a placa acima mencionada não venha a ser construída, já que em relação a esta solução tem a vantagem de permitir o debugging pelo ICD2 da microchip.
Os desenhos dos circuitos electrónicos de ambas as placas encontram-se em anexo.

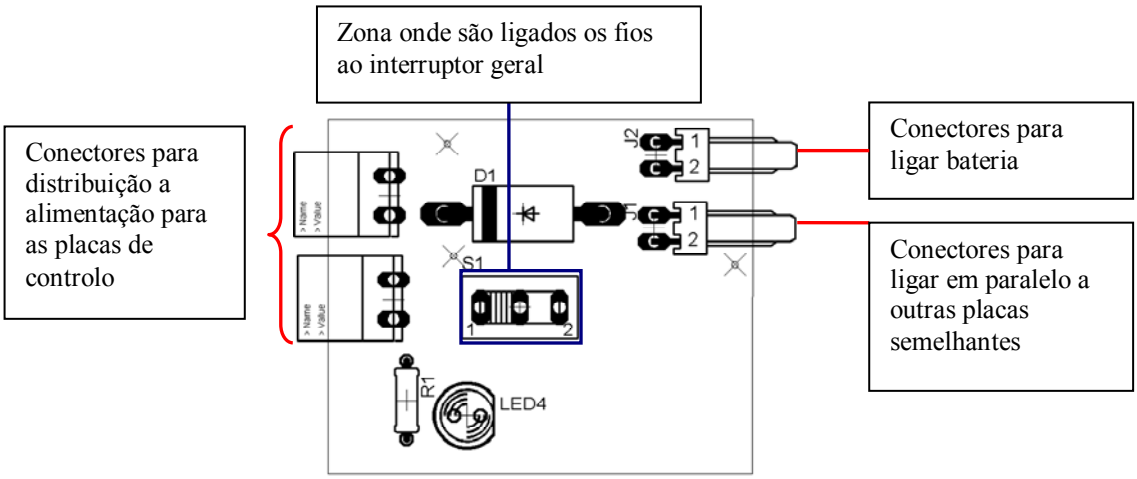


5.2.6 Placas Alimentação

As placas de alimentação foram desenvolvidas com o intuito de proporcionarem a distribuição da alimentação proveniente das baterias para todas as outras placas. Tem um circuito que permite após regulação de um potenciómetro acender um led quando as baterias estão abaixo da tensão mínima admitida. Estão ligadas a um interruptor que permite ligar e desligar a alimentação geral da plataforma. Foram desenvolvidas três placas, duas com o circuito de monitorização da tensão e distribuição (uma usa o amplificador OP200 e a outra LM741P) e uma de distribuição de alimentação que está ligada em paralelo com qualquer uma das outras. Os desenhos dos circuitos encontram-se em anexo.







5.3 Fabricação das placas PCB

Algumas das placas acima mencionadas foram fabricadas pelo processo desenvolvido pelo departamento. No início, todas as placas eram para ser fabricadas com este processo, contudo, após alguns testes com algumas das placas, principalmente com as de dupla fase e com pistas muito finas, o processo revelou-se ineficaz. Após aplicar o processo, variando as diferentes variáveis (tempo de exposição ao raios ultra violetas e concentração do revelador), sistematicamente partes das placas obtidas apresentavam zonas defeituosas como ilustram as imagens seguintes.

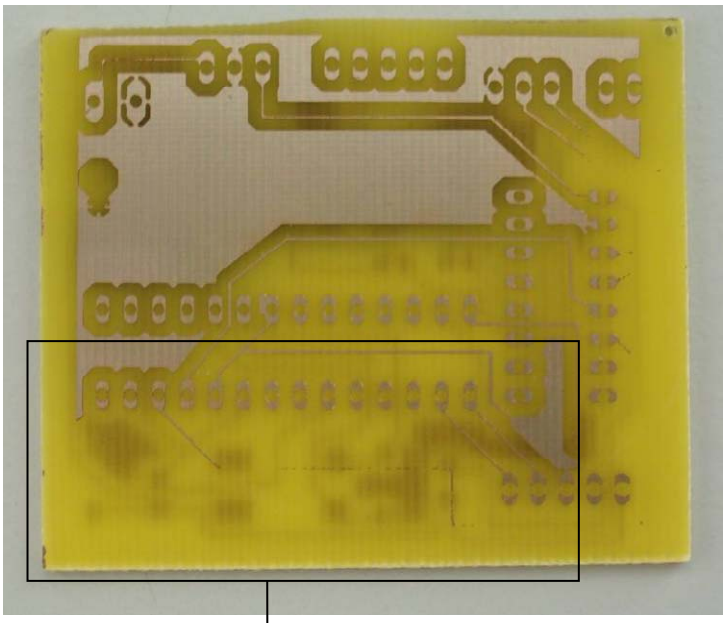


Figura 32 – Imagem de uma placa com defeitos

Não foi possível tornear estes defeitos, optando-se então por outra solução. No entanto, para as placas com pistas mais largas o processo foi viável. As placas fabricadas foram: placa expansão Master e placas de alimentação.

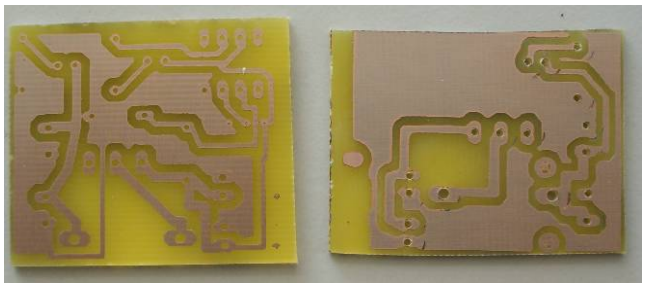


Figura 33 – PCCs fabricados no departamento

As restantes foram mandadas fazer a uma empresa especializada, mais concretamente na empresa CircuiTotal, situada em Aveiro. Todas são de dupla face e têm furos passantes para as pistas. Estas placas, trouxeram uma grande vantagem para a soldadura: o facto de terem furos metalizados evita que se tenha de soldar certos componentes nas duas faces. Em algumas placas que se soldou sem este tipo de furos, frequentemente certos componentes ficavam com maus contactos tornando o processo de colocação em funcionamento bastante penoso. Foi o caso das placas de expansão dos sensores de força: estas placas foram possíveis de obter com o nosso processo, contudo o processo de soldadura e colocação em funcionamento embateu nestes problemas.

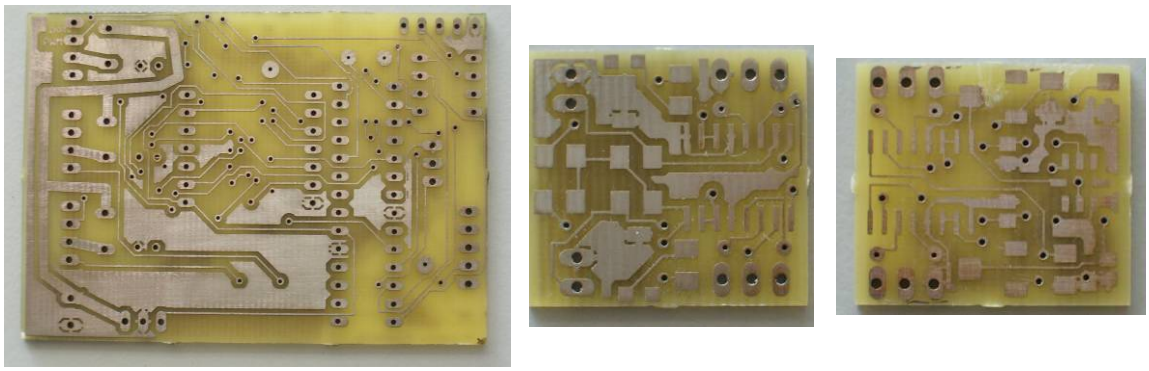


Figura 34 – PCBs fabricados fora do Departamento

5.3.1 Descrição do processo de fabrico de PCB

1º Passo – Desenhos em acetatos

Após a obtenção do desenho da placa, o desenho têm que ser imprimido numa folha de acetato. Na maioria dos casos, uma só impressão não é suficiente devido ás imperfeições da impressão. A técnica usada para contornar este problema, é imprimir uma nova folha e sobrepor uma sobre a outra, agrafando-as de maneira a que fiquem perfeitamente concordantes. A impressão pode ser a laser ou jacto de tinta: o mais importante é ser a mais perfeita possível.

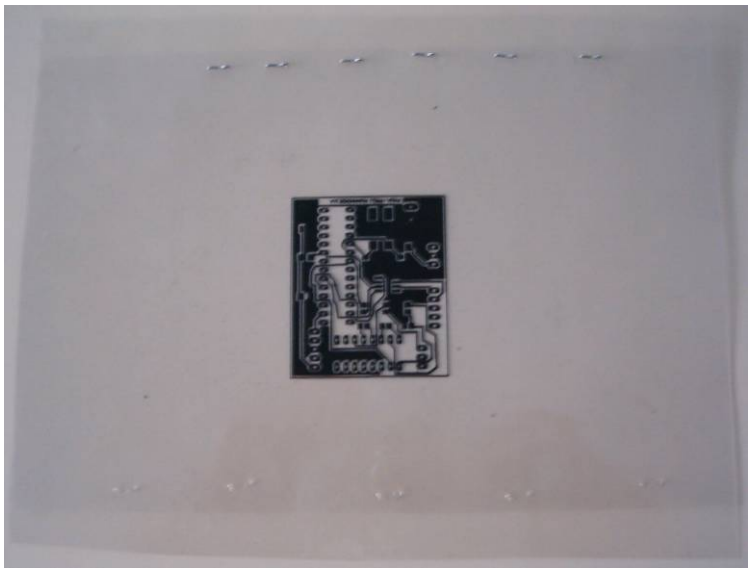


Figura 35 – Folha de Acetato

2º Passo – Sensibilização das placas

Existem placas sensibilizantes especiais em cobre, estas têm que ser previamente cortadas á medida da placa que se pretende obter. O objectivo neste passo é passar o desenho da folha de acetato para estas placas.

Esse trabalho foi executado utilizando lâmpadas que geram radiação ultravioleta. A radiação ataca o verniz que se encontra á superfície da placa, a tinta do desenho evita a passagem dos raios, aquando da revelação estas zonas não atingidas vão ficar reveladas. A folha de acetato onde se encontra o desenho é encostada á placa sensibilizante, normalmente fixa-se uma à outra por intermédio de fita-cola ou mesmo clipes, o importante é durante o tempo de exposição não sair do sítio. O tempo de exposição foi determinado de modo experimental, o mais apropriado é de 1 minuto e meio. A exposição a estas lâmpadas foi feita por intermédio de um dispositivo desenvolvido no departamento.

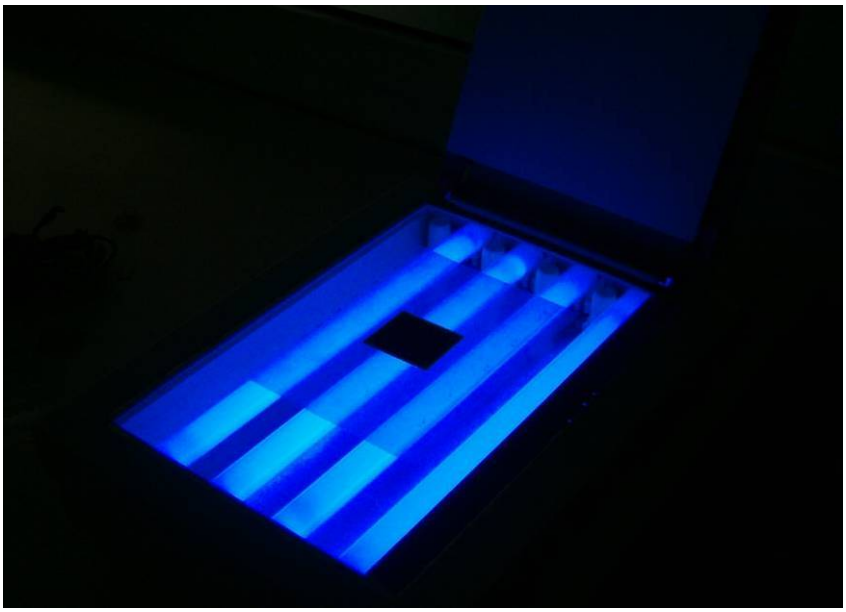


Figura 36 – Dispositivo com lâmpadas ultravioletas do Departamento

3º Passo – Revelação

Este processo vai revelar o desenho das pistas na placa. A placa após sensibilizada tem que ir a um banho para retirar o verniz que foi destruído pela radiação. Numa primeira fase, usou-se um banho de um produto existente no mercado, mas como se acabou, a solução encontrada foi usar um banho de uma solução de hidróxido de sódio (Na OH) mais conhecida vulgarmente por soda cáustica. Embora a concentração usada tenha sido obtida experimentalmente, o aconselhado é 7g de soda cáustica para um litro de água. O tempo do banho é variável, dependendo do tempo de exposição e da concentração do revelador, mas passado alguns segundos o desenho vai aparecendo na placa. Quando se tornar completamente nítido deve-se retirar e passar por água. Se por acaso passar ± os 2 minutos é sinal que algo foi mal executado. O revelador fica colorido em tons de verde, mas pode ser guardado para nova utilização. No caso da soda cáustica se tratar de um produto barato, o melhor é usar sempre um banho fresco.

Figura 37 – Banho no Revelador

4º Passo – Remoção do cobre

Esta operação é executada recorrendo a um banho de percloreto de ferro. É necessário aquecer o percloreto previamente e mantê-lo o mais quente possível durante o processo, a técnica usada para manter o banho quente é através do bem conhecido banho-maria. O tempo do banho é determinado pela total remoção do cobre ficando apenas as pistas em cobre como se pretende. Os problemas que podem surgir durante a operação de destruição do cobre são principalmente devidos a erros de exposição.

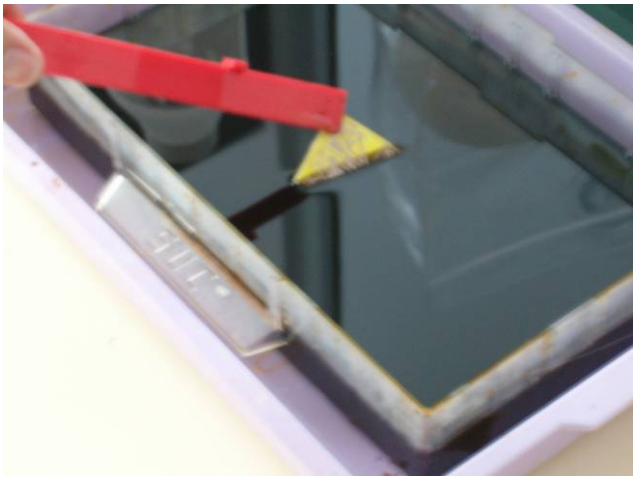


Figura 38 – Banho de percloreto

Cuidados a ter para protecção pessoal:
No segundo passo deve-se evitar que os raios ultravioletas incidam directamente com os olhos. Nos passos 3 e 4, deve-se evitar ao máximo o contacto com a pele dos produtos químicos usados. Em caso de contacto lavar imediatamente com água.

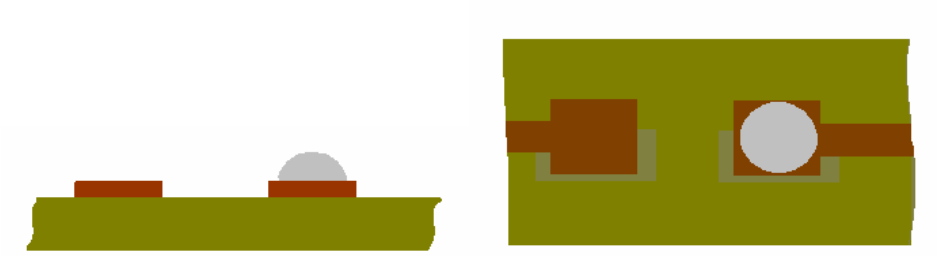
5.4 Descrição da técnica de soldadura de componentes de superfície

A técnica que a seguir vai ser descrita usada para a soldadura das placas PCB apresentadas no Relatório.

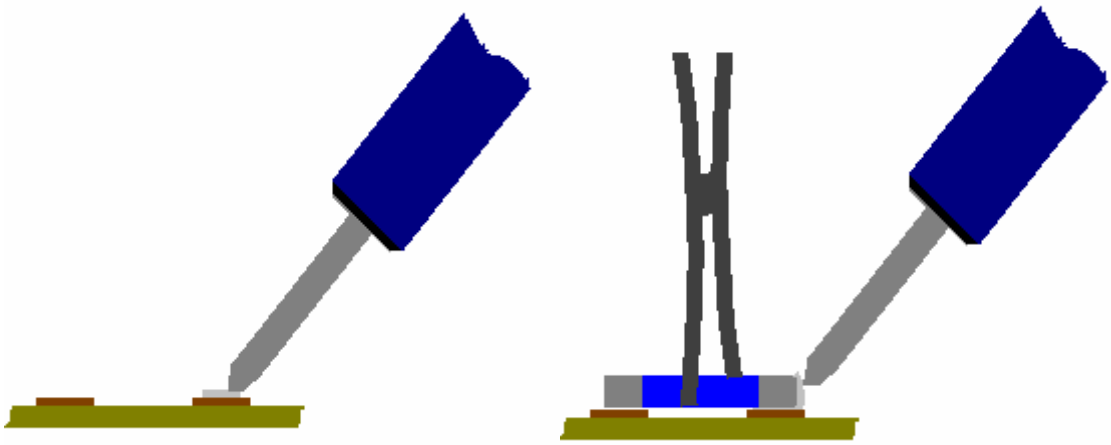
5.4.1 Soldadura de Resistências e Condensadores SMD

Para a soldadura de destes componentes deve-se seguir os seguintes passos:

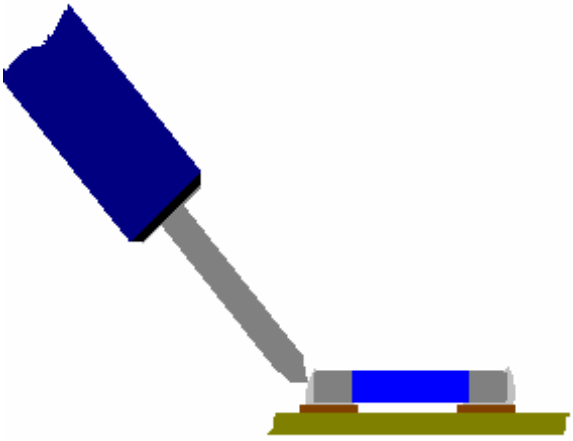
1º Passo – depositar numa das patas em cobre da placa uma pequena quantidade de estanho.



2º Passo – Aquecer o estanho depositado e com auxílio de uma pinça colocar o componente no lugar, como o estanho esta quente agarra-se à pata do componente e ao cobre da pista, retirar o ferro e esperar que arrefeça. No final uma das patas fica soldada.



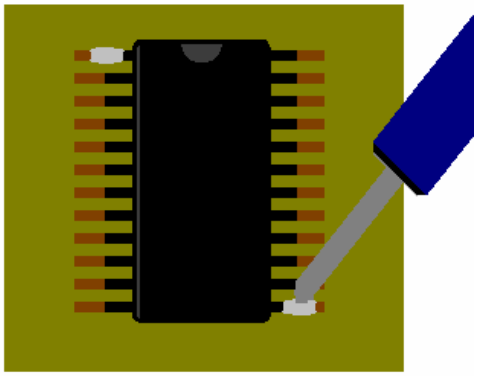
3º Passo – Neste passo solda-se a pata restante.



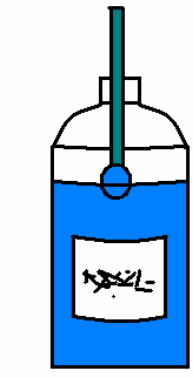
5.4.2 Soldadura de componentes SOIC

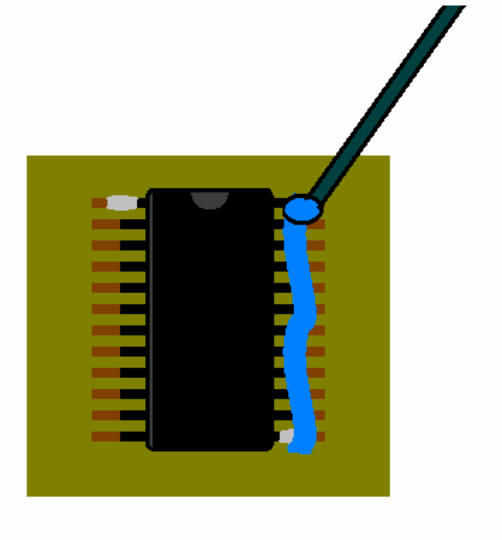
Para a soldadura de destes componentes deve-se seguir os seguintes passos:

1º Passo – Por intermédio de uma pinça ou manipulador por ventosas colocar o componente na posição correcta, em seguida solda-se uma das patas da ponta, depois deixar arrefecer e soldar outra pata do outro lado em sentido oposto. Esta operação permite que o componente fica seguro à placa e não mais é necessário segura-lo. Durante a operação ter bastante cuidado para que o componente não escape do lugar.

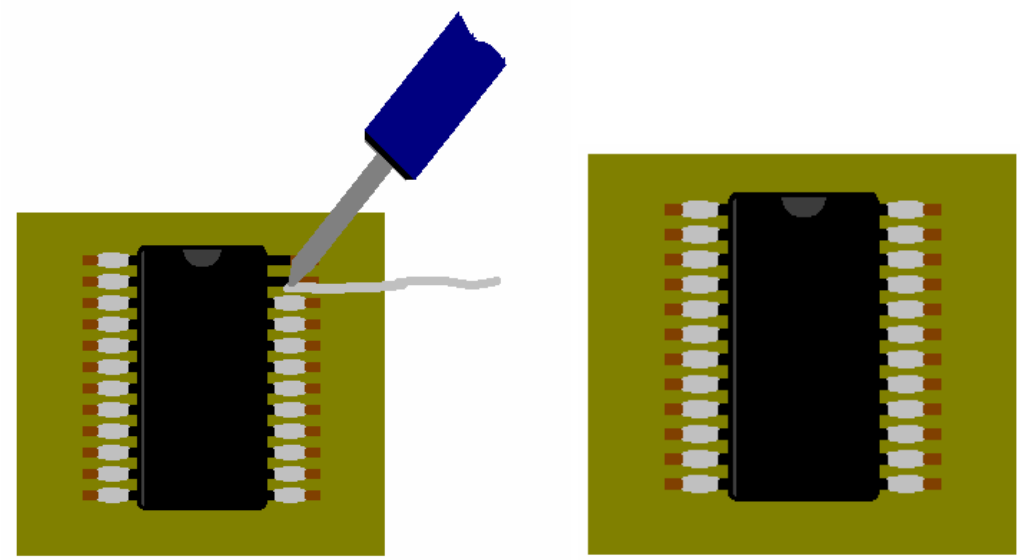


2º Passo – É aplicado sobre todas as patas um líquido especial, este líquido vai evitar que a solda se a agarre nos espaços onde não há metal. Para aplicar este líquido deve-se usar um cotonete ou algo semelhante, mergulhar no líquido e depois passar por todas as patas como ilustra a figura.





3º Passo – Neste último passo derrete-se solda para cima das patas, com o liquido aplicado a solda aloja-se nas patas e nas pistas do cobre soldando-as assim umas às outras.



5.5 Placas obtidas após soldadura

A seguir apresenta-se as imagens das placas finais fabricadas. Em análise aos resultados obtidos com a técnica de soldadura pode-se dizer que foi assimilada com sucesso, as placas em geral apresentam soldaduras com bom aspecto.

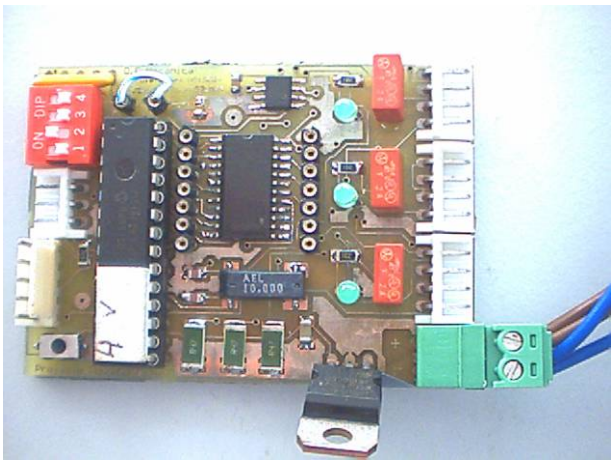


Figura 39 – Placa Principal

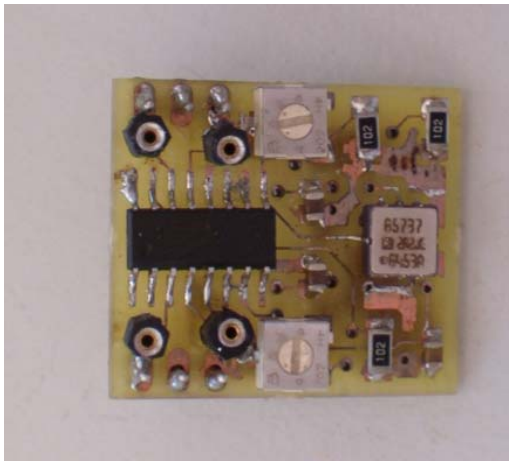


Figura 40 – Placa Expansão Inclinómetro

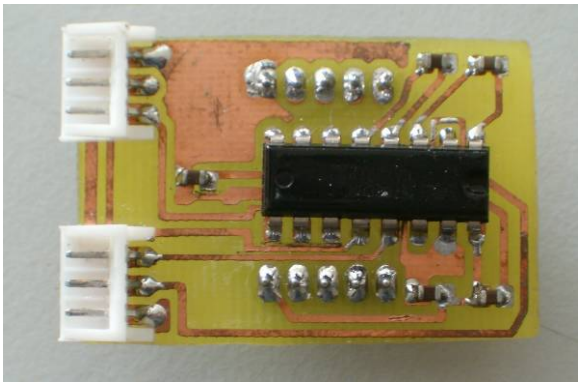


Figura 41 – Placa Expansão Master

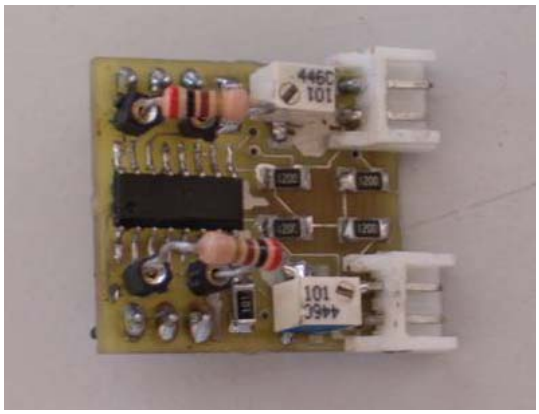


Figura 42 – Placa Expansão sensores força

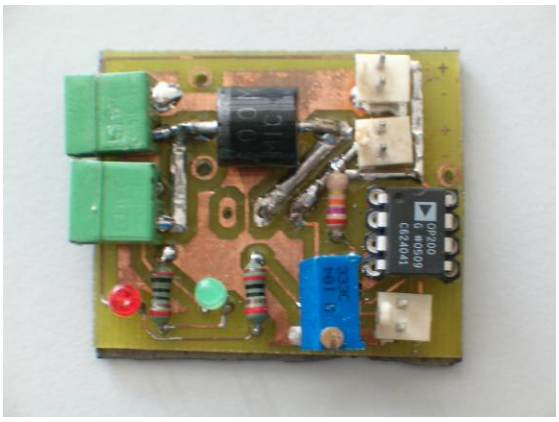


Figura 43 – Placa Alimentação

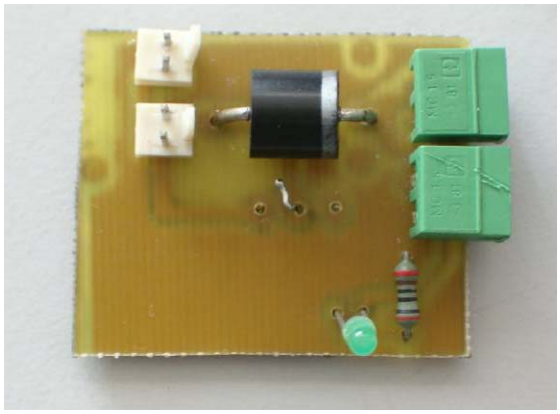


Figura 44 – Placa Alimentação

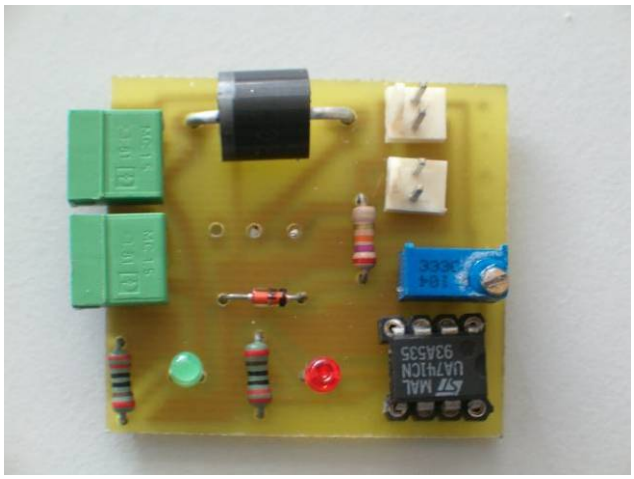


Figura 45 – Placa Alimentação

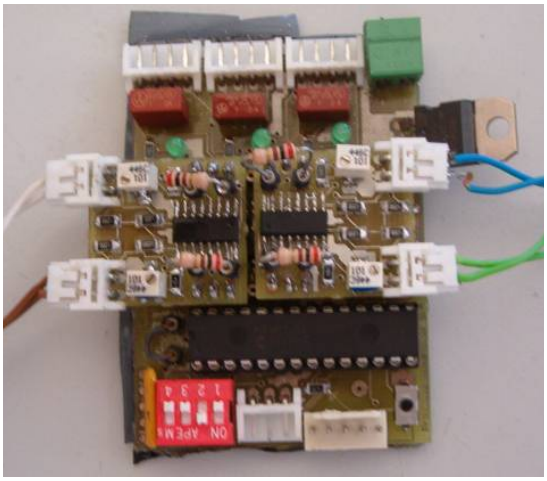


Figura 46 – Piggy Back, placas expansão sensores força

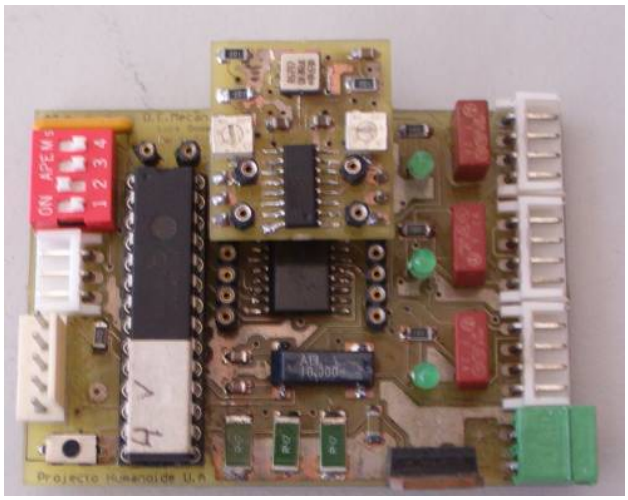
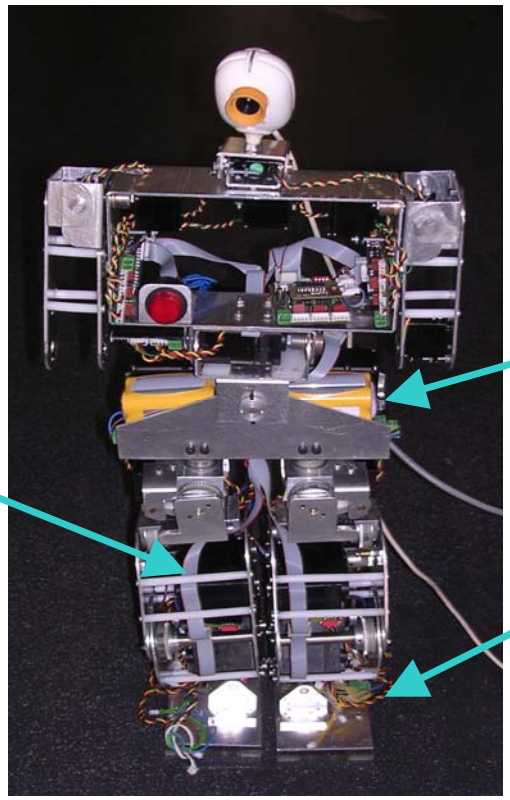


Figura 47 - Piggy Back, placa expansão inclinómetro

6. Integração

Na fase de integração foi necessário efectuar um conjunto de tarefas que possibilitassem o funcionamento de toda a plataforma. Como já foi referido este ano houve outro grupo de trabalho que desenvolveu a estrutura mecânica da plataforma. Durante a concepção existiu troca de informação entre os dois grupos de trabalho para possibilitar a instalação de toda a electrónica de controlo e percepção. Com os locais de instalação das placas de controlo definidos foi necessário elaborar toda a cablagem necessária para colocar em funcionamento o robô. Esta tarefa só foi possível após a estrutura ter sido disponibilizada pelo outro grupo de trabalho.



Os trabalhos consistiram basicamente em preparar todos os condutores com os respectivos conectores para possibilitar a ligação às placas.
Durante estes trabalhos surgiu a necessidade de isolar as placas uma vez que a maioria das superfícies de apoio na plataforma são de alumínio provocando curto circuitos nas placas. Sugere-se para no futuro seja aplicado nas placas um polímero entre a superfície de apoio e as placas de controlo.
Uma vez concluído este trabalho tentou-se ligar o sistema. Inicialmente pretendia-se utilizar as fontes de alimentação existentes no LAR ligadas em paralelo. Com esta configuração seria possível fornecer teoricamente 20 Amperes. O que se verificou na prática foi efectivamente que era impossível iniciar toda a plataforma em simultâneo pois verificou-se que na fase de arranque os servomotores possuem um pico de consumo de corrente elevadíssimo. Sem outra alternativa passou-se a utilizar as baterias que estavam destinadas para o robô. Foi então possível pela primeira vez colocar todo o sistema em funcionamento para efectuar os primeiros testes.

Verificou-se mais uma vez que a arquitectura distribuída apresenta claras vantagens para uma plataforma deste tipo. As falhas que surgiram durante os trabalhos foram localizadas de forma relativamente expedita. Sempre que houve necessidade de intervenção era possível trocar a placa por outra de reserva, assim só era necessário configurar o endereço correcto para a comunicação CAN.
Relativamente à disposição dos motores pelas placas das redes é apresentada a seguir uma ilustração que indica em que conectores se têm de ligar os motores

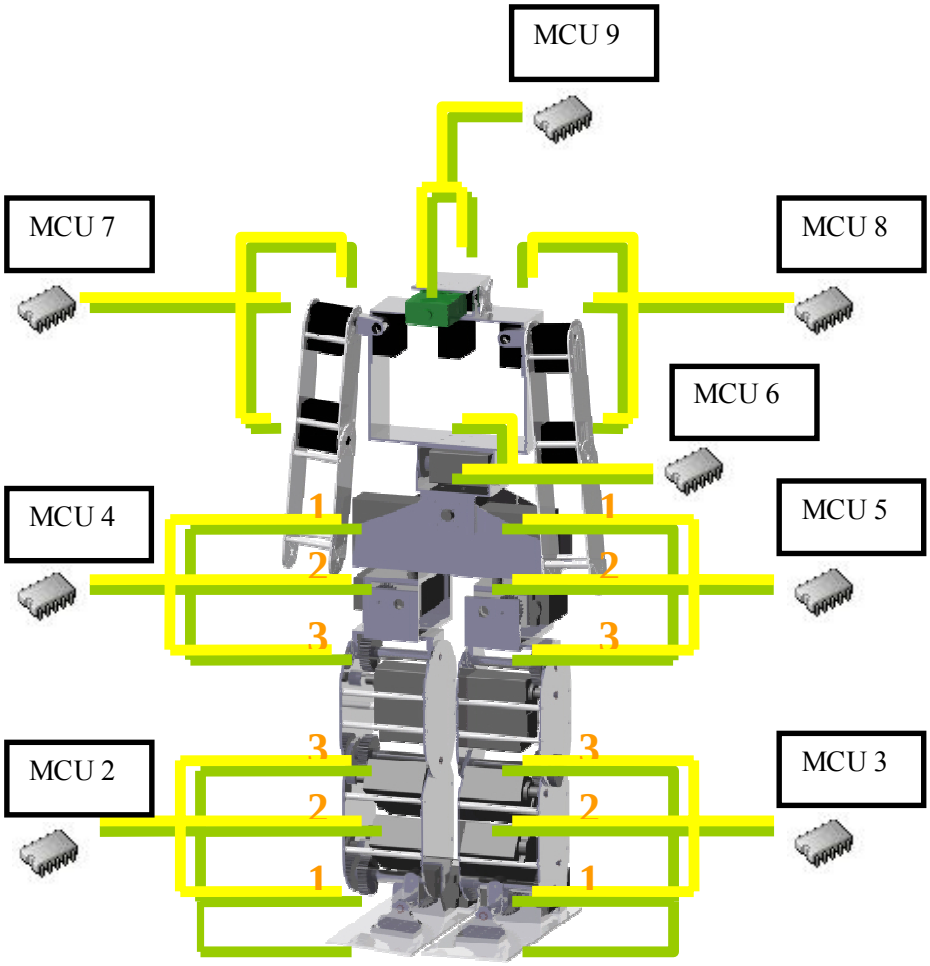


Figura 48 - Distribução dos Motores pelas placas

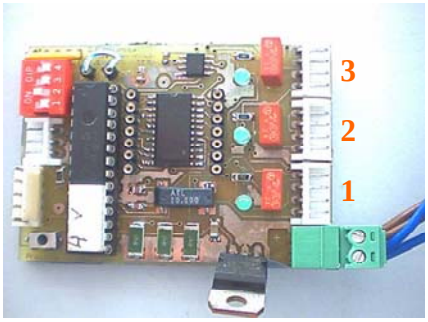
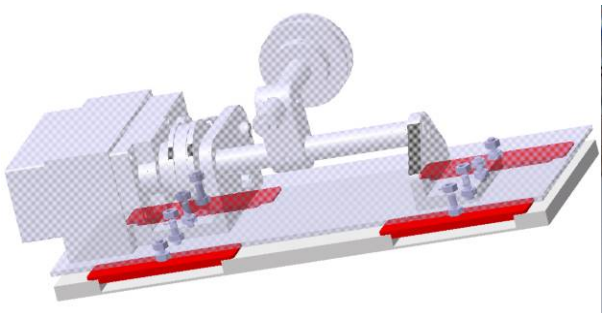


Figura 49 - Numeração dos Conectores

Actualmente só se tem essa distribuição implementada para os motores das pernas.

pé direito, ao lado	1
pé direito, à frente	2
joelho direito	3
pé esquerdo, ao lado	1
pé esquerdo, ao lado	2
joelho esquerdo;	3
anca direita, virar	1
anca direita, ao lado	2
anca direita, à frente	3
anca esquerda, virar	1
anca esquerda, ao lado	2
anca esquerda, à frente	3

Após os nossos colegas concluírem a estrutura, foi possível modelar e construir um novo protótipo para acoplar á plataforma. Recorreu-se a uma máquina de CNC de forma a dar rigor dimensional a todas as peças, algo que não se conseguiu no protótipo anterior. Foram efectuados testes com a plataforma em movimento e adquiridos valores, contudo os valores não são totalmente conclusivos. A solução do piggy back demonstrou que foi uma óptima escolha, estando todas as placas em perfeito funcionamento.



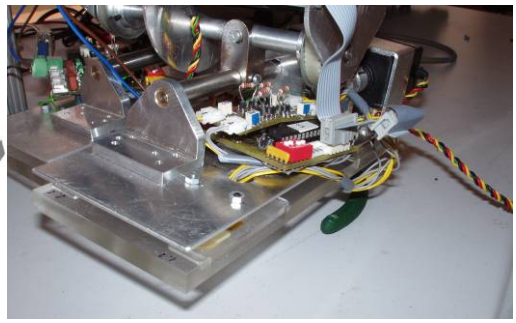


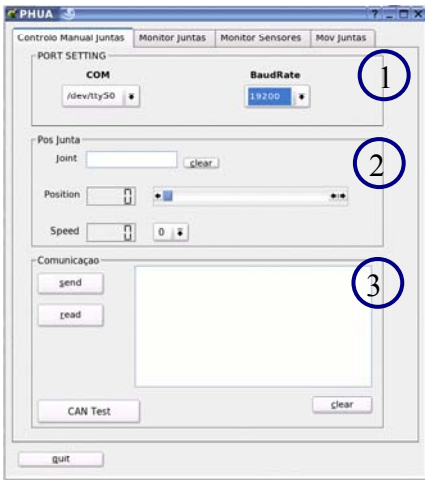
Figura 50 – Protótipo final construído

7. Controlo de alto Nível

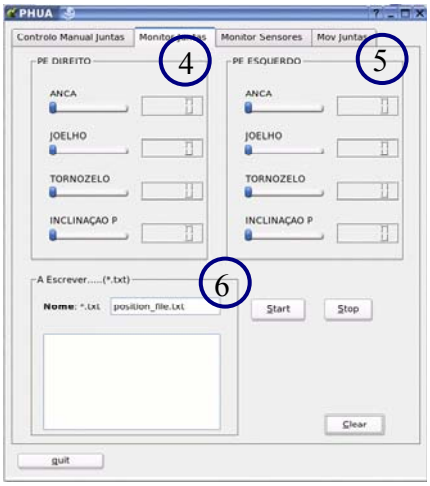
Foram desenvolvidos diversas aplicações de controlo e monitorização de alto nível em diferentes programas (MatLab, Qt designer em ambiente linux). Estes programas foram desenvolvidos com base no protocolo desenvolvido da comunicação Rs232 entre o microcontrolador e a unidade de controlo principal, actualmente um PC. Com as aplicações foi permitido testar as comunicações intensivamente de modo a corrigir as falhas existentes. Também foram desenvolvidas algumas aplicações de monitorização de sensores e para adquirir dados para calibração e estudos.
Contudo com novos desenvolvimentos nos estudos da robustez das comunicações algumas aplicações carecem de pequenas actualizações. A seguir apresenta-se resumidamente as aplicações desenvolvidas.

Em Qt Designer:

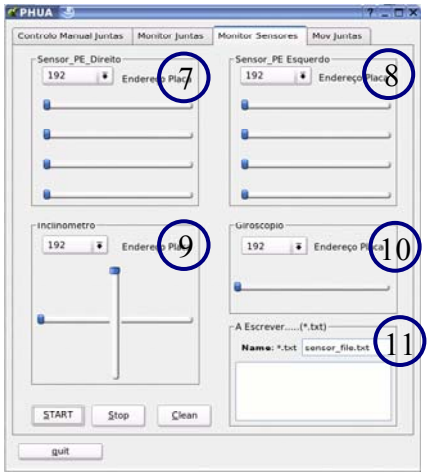
Esta aplicação foi desenvolvida no âmbito da disciplina de Robótica autónoma e móvel e permite o controlo das juntas das pernas e a monitorização dos diversos sensores desenvolvidos. È também a aplicação mais completa de todas, devido as enumeras tarefas que cumpre, as figuras abaixo ilustram o seu aspecto gráfico com comentário das diferentes funções.



1 Configurações da porta RS232;
2 Controlo para movimentar qualquer junta individualmente;
3 Envio e Recepção de comandos pela porta, Visualização do que é recebido;



4 Monitorização das juntas da Perna Direita;
5 Monitorização das juntas da Perna Esquerda;
6 Nome do ficheiro txt onde os dados são guardados e como;



7 Monitorização dos valores dos sensores de força na Perna Direita;
8 Monitorização dos valores dos sensores de força na Perna Esquerda;
9 Monitorização dos valores do Inclinómetro;
10 Monitorização dos valores do Giroscópio;
11 Nome do ficheiro txt onde os dados são guardados e como;

12 alores das juntas da Perna direita;
13 alores das juntas da Perna esquerda;
14 Possibilidade de ler de um ficheiro os valores das juntas das pernas;
15 Nome do ficheiro txt onde os dados são guardados e como;

Aplicações em MatLab:

O MatLab é uma ferramenta que permite desenvolver aplicações num ambiente de grande simplicidade e flexibilidade no envio e tratamento de dados, foi por isso a escolhida para adquirir dados para calibração dos sensores. Foram também desenvolvidas aplicações de monitorização de valores sensoriais nomeadamente dos sensores de força e inclinómetros. As figuras abaixo ilustram o aspecto gráfico destas aplicações.

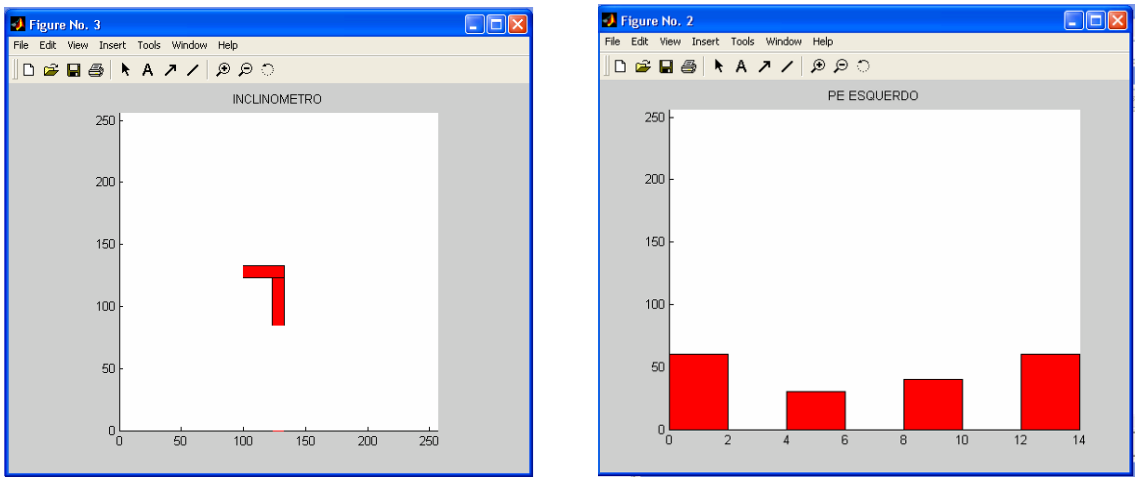


Figura 51 – Aplicações em MatLab, à esquerda o inclinómetro e à direito do sensores força

Todas as aplicações são fornecidas no CD que acompanha o Relatório.

8. Conclusão

Relativamente ao controlo serão abordadas as funcionalidades mais importantes tentando deixar um conjunto de notas que possam ter utilidade para a continuação dos trabalhos neste projecto.

Controlo dos servomotores

As alterações que se efectuaram no controlo dos servomotores corresponderam da melhor forma. Para além de se ter conseguido a estabilidade necessária para actuar a plataforma construída é possível adaptar o controlo a servomotores que tenham um sinal de controlo PWM de características diferentes sem que se tenha de alterar o hardware. Pois a geração da frequência é efectuada unicamente por software podendo a qualquer momento ser alterada para um valor diferente.
O controlo de velocidade encontra-se igualmente implementado e permitiu a actuação da plataforma. No entanto é necessário referir que a maior parte dos testes que se efectuaram não se utilizou o controlo em malha fechada mas sim em malha aberta. Ambos os controlos estão aptos para serem aplicados na plataforma, somente se utilizou o controlo em malha aberta porque na altura o controlo em malha fechada não se encontrava plenamente implementado. A utilização dos potenciómetros internos dos servomotores para efectuar este controlo é possível mas pensa-se que a melhor opção será aplicar potenciómetros externos. Uma vez que ao ligar-se um conector em paralelo com o potenciómetro interno para adquirir o seu sinal pode muito facilmente afectar o controlador interno do servomotor. A utilização de um potenciómetro externo será uma solução muito mais fiável e robusta.

Aquisição de valores sensoriais

A aquisição sensorial encontra-se num estado de evolução avançado não necessitando de intervenção futura. Verificaram-se bons resultados uma vez que foi utilizada para calibrações de diversos sensores. A utilização de um multiplexer para adquirir um elevado números de valores sensoriais verifica-se de grande utilidade e encontra-se plenamente desenvolvido o seu controlo. A nível dos Escravos todo o processo de aquisição, registo e envio via CAN para o Mestre encontra-se exaustivamente testado confirmando-se o seu bom funcionamento.

Comunicação CAN

A troca de informação do Mestre com o escravo já se encontrava plenamente desenvolvida no inicio dos trabalhos deste ano lectivo. Unicamente foi-lhe acrescentado mais dois bytes nas mensagens do Mestre para os escravos para possibilitar o controlo de velocidade por parte da unidade principal de controlo.
A troca de informação dos Escravos com o Mestre encontra-se testada confirmando-se igualmente o seu bom funcionamento. Os registos dos valores sensoriais no Master encontram-se desenvolvidos e fiabilizados.

Comunicação Série

A comunicação série do PC com o Mestre adquiriu um elevado grau de fiabilidade depois de se ter introduzido os bits de sincronização.

Arquitectura Distribuída Geral

Para este tipo de plataforma verificou-se que a arquitectura de controlo distribuída foi uma boa opção pois permite uma fácil expansão do sistema global utilizando os módulos desenvolvidos. Verificou-se ao longo das várias semanas de testes que o sistema já atingiu um nível robustez considerável. Os módulos desenvolvidos são de enorme valor dado que podem ser aplicados em diversos tipos de tarefas sendo até possível a sua utilização em outras plataformas robóticas. A verdadeira vantagem deste sistema reside no facto de se poder adicionar módulos com uma configuração mínima .

Integração

Conseguiu-se actuar todos os motores da plataforma possibilitando-se a execução de diversas sequências de movimentos permitindo ainda a aquisição em simultâneo de sinais sensoriais.

Sensores e percepção

Os sensores dos pés estão completamente desenvolvidos e montados na plataforma, espera-se que num futuro próximo venham a ser de grande utilidade para o controlo da plataforma. As novas placas de expansão apresentam um bom comportamento dentro daquilo que lhes é exigido. Já foram efectuados alguns testes ao novo protótipo, aparentemente funciona como previsto, apenas teve de ser corrigido um pequeno erro de construção. Faltam efectuar ainda bastantes testes até que se possa usar definitivamente o sensor, mas as perspectivas futuras são bastante animadoras.
Os inclinómetros apresentam um estado de completo desenvolvimento. As placas de expansão contudo precisam de uma pequena rectificação, o potenciómetro actualmente usado de uma só volta dificulta o afinamento do sinal. A solução recomendada para este problema é trocar este potenciómetro por um de multi-voltas.
Os potenciómetros já estão a ser utilizados para implementação do controlo de velocidade, mas como já foi dito acima provavelmente vão ter que ser substituídos por um potenciómetro externo.
Toda a parte de desenvolvimento do giroscópio está concluída, com o desenho da placa de expansão disponível falta apenas o fabrico. No futuro será necessário adquirir novos giroscópios, os que existem actualmente encontram-se avariados. Os resultados obtidos até a sua avaria foram conclusivos para afirmar que esta é uma solução valida.
Os medidores de corrente dos motores foram incorporados na placa principal. Falta no futuro efectuar a sua calibração.

Módulos Desenvolvidos

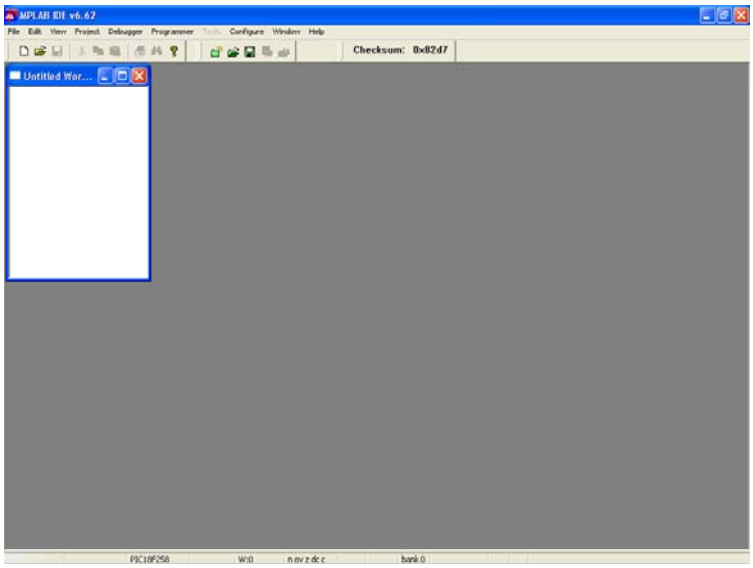
Os módulos desenvolvidos apresentam em geral um bom desempenho. Contudo após serem usadas intensivamente chegou-se á conclusão que se pode efectuar alguns melhoramentos nomeadamente na placa principal.

No início da integração teve-se alguns problemas ao nível da alimentação das placas, verificava-se que os motores ficavam completamente fora de controlo, causando danos na estrutura. Para evitar este tipo de problemas no futuro propõem-se que as placas tenham transístores ou relés ligados aos motores para assim se poder ter controlo destas situações. As placas quando sujeitas a grandes consumos de corrente aquecem demasiado, a solução passa por incluir dissipadores nas resistências de potências já que são as que mais aquecem. O filtro passo abaixo que se implementou no multiplexer têm de ser retirado visto influenciar negativamente os valores de tensão á saída aquando da comutação de canal efectuado pelo multiplexer.

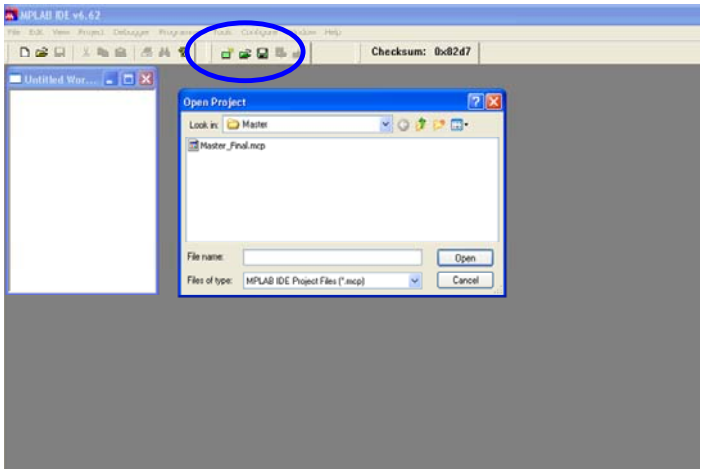
ANEXO I

O Ambiente de desenvolvimento MpLab IDE v6.62

Como ferramenta de desenvolvimento do código dos microcontroladores utilizou-se o MpLab IDE v6.62 da Microchip. A linguagem de programação utilizada foi o C, como compilador utilizou-se o MCC18 da Microchip. Este compilador permite ser integrado no ambiente de desenvolvimento.
Aspecto gráfico do ambiente de desenvolvimento do MpLab.



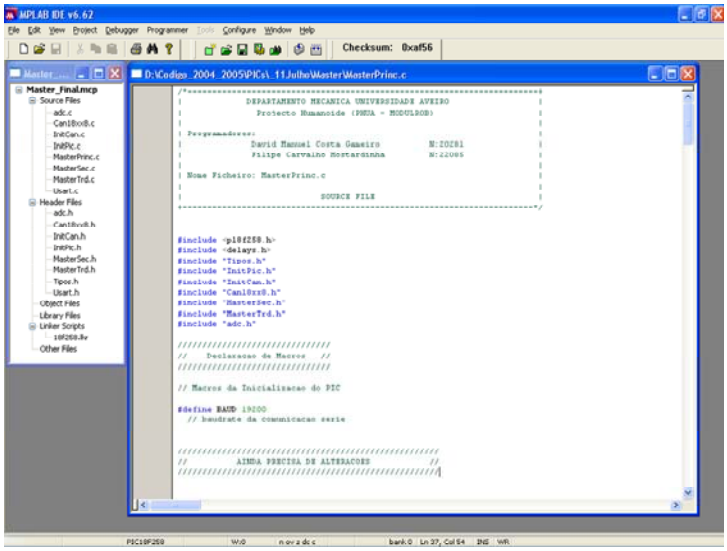
Anexo - Figura 1



Anexo - Figura 2

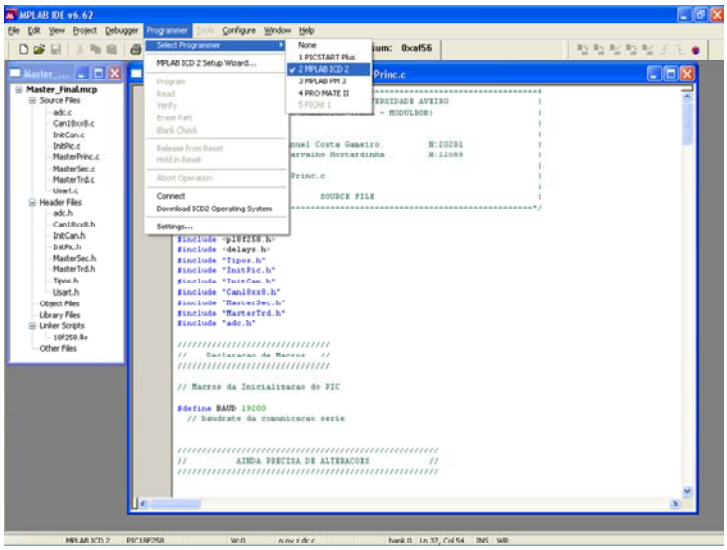
A seguir será descrito o processo necessário para abrir um projecto existente.

Para abrir um projecto existente basta escolher o botão com a pasta verde na barra de ferramentas. Para abrir ficheiros sem que se queira abrir todo projecto utiliza-se o outro botão com a pasta amarela.
Ao abrir um projecto temos acesso a todos os ficheiros que fazem parte do projecto.



Anexo - Figura 3

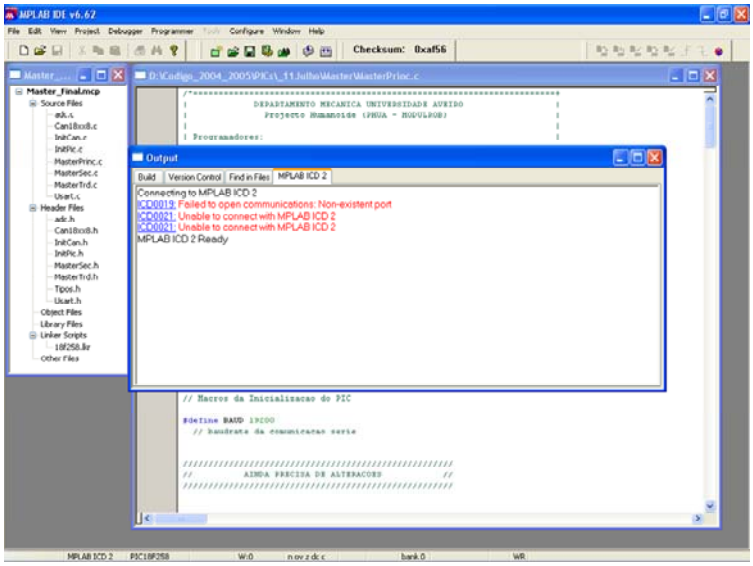
A seguir será descrito o processo necessário para programar um microcontrolador utilizando o código de um projecto desenvolvido utilizando o programador da MpLab ICD2.
Na barra de menus do MpLab existe o menu Programer. Neste menu escolhe-se o programador que queremos utilizar, neste caso o MpLab ICD2.



Anexo - Figura 4

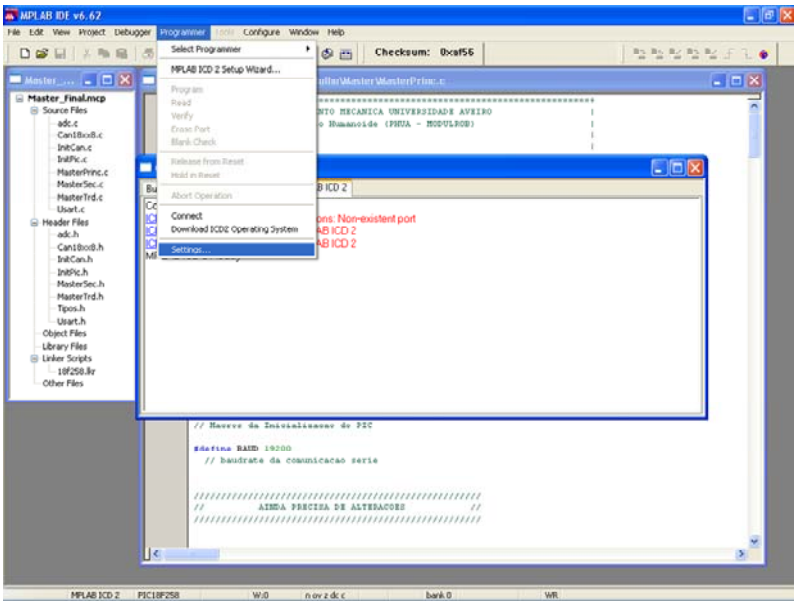
Como se pode ver na figura aparece uma nova janela designada por Output. Esta janela é muito importante pois todas as mensagens que o MpLab devolve para o utilizador são

apresentadas nesta janela. Como se pode por exemplo ver na figura são devolvidas mensagens de erro ao ligar-se pela primeira vez o ICD2.



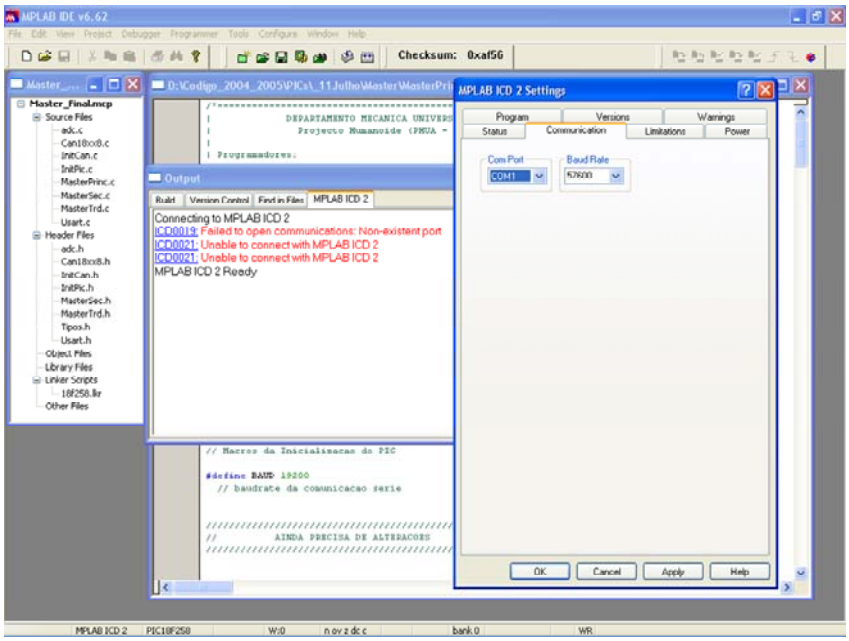
Anexo - Figura 5

As mensagens indicam que não foi possível comunicar com o ICD2. Para estbelecer a ligação correcta é necessário escolher o submenu Settings do menu Programer.

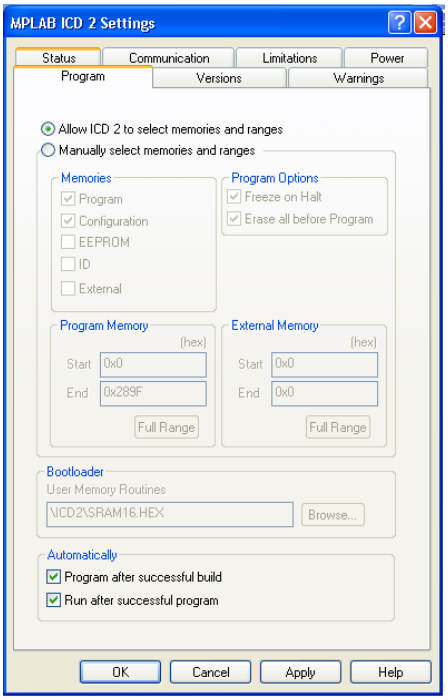


Anexo - Figura 6

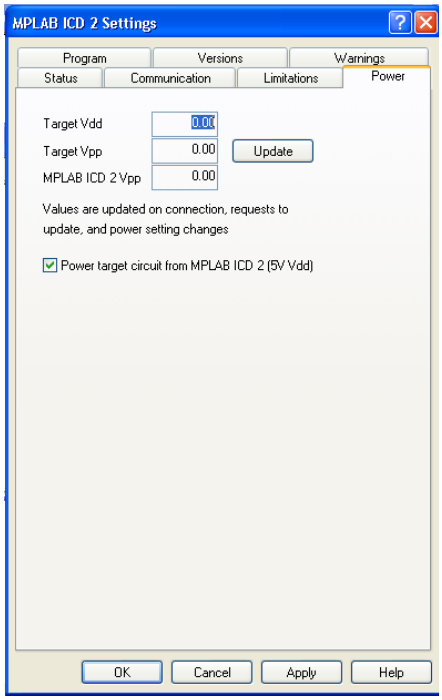
Dentro deste submenu de configurações é necessário o frame Communication. Aqui pode-se escolher o Port de comunicações e a baud rate a que se comunica. A comunicação por USB nunca foi utilizada pelo qual se aconselha a utilização da comunicação Série.



Anexo - Figura 7



Anexo - Figura 8

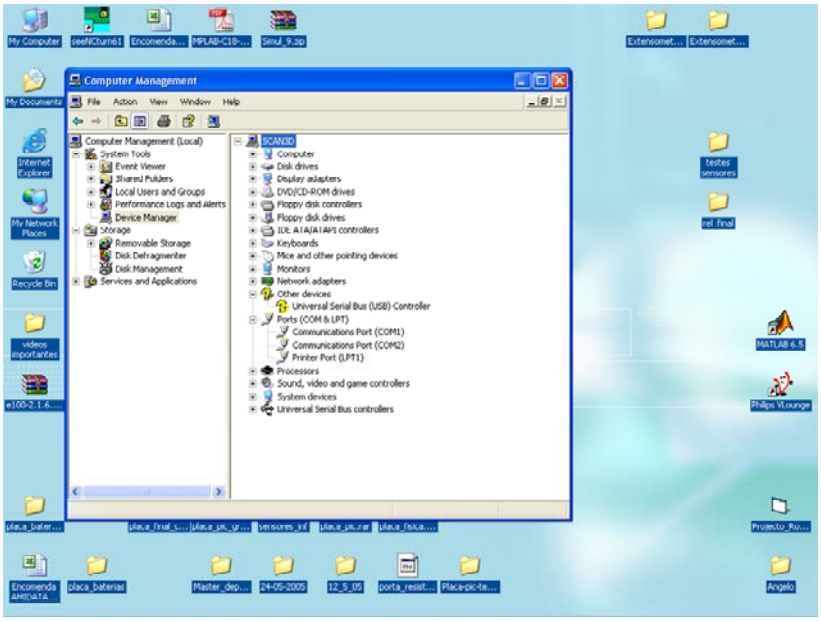


Anexo - Figura 9

Nas restantes configurações são utilizadas as opções originais. Basta então carregar em Apply e OK. Ao efectuar se isto o MpLAb devolve o seguinte aviso.

Anexo - Figura 10

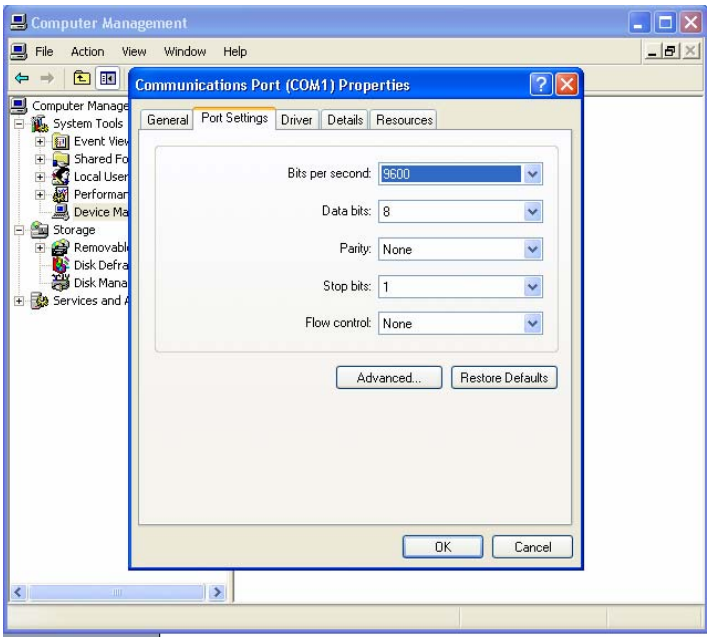
O Windows tem por defeito os FIFO buffers da comunicação série activados. Temos que ir ao My Computer desctivar os serial FIFO buffers. A maneira mais rápida é carregar com a tecla direita do rato sobre o icon do meu computador. Ao efectuar-se isto aparece o menu apresentado na figura. Escolhe-se o submenu Manage.



Anexo - Figura 11

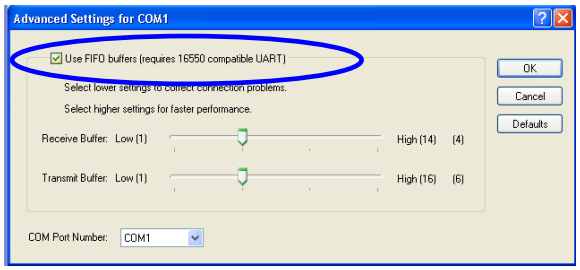
Nesta janela escolhe-se o dispositivo de comunicação série que se pretende alterar a comunicação.

Escolhe-se na nova janela o frame Port Setting. E selecciona-se o botão Advanced.



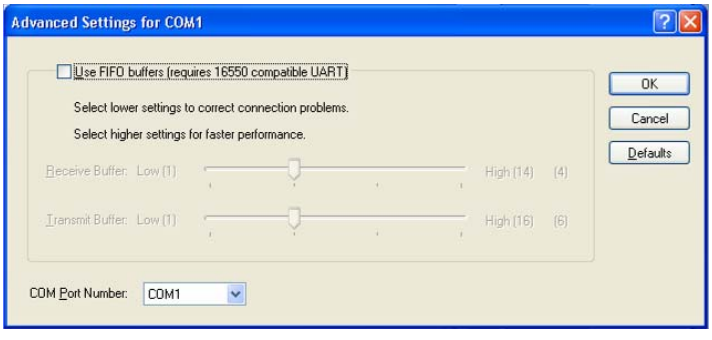
Anexo - Figura 12

Na janela Advanced Settings desactiva-se o FIFO buffers da Porta de comunicações.



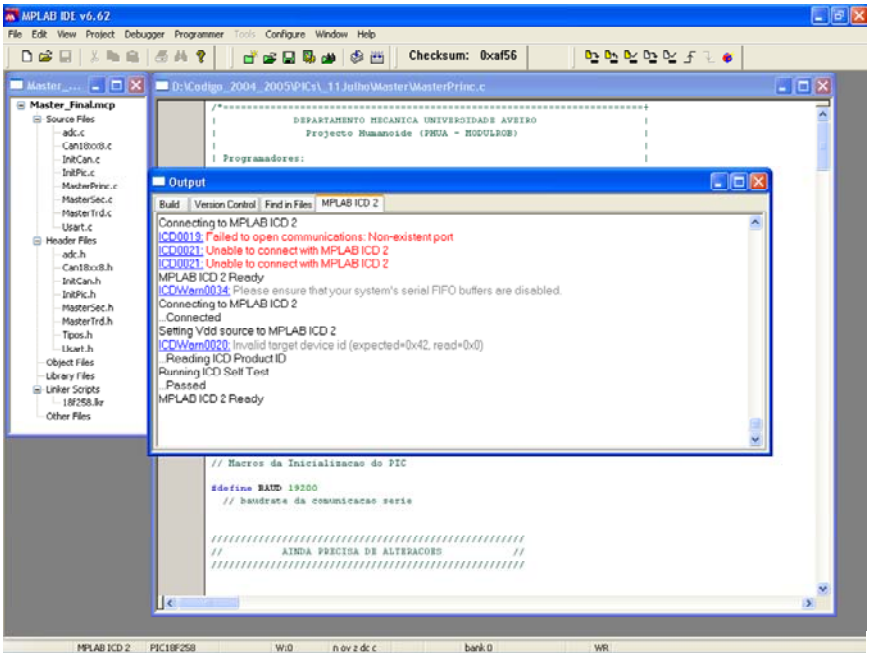
Anexo - Figura 13

Neste momento o MpLab já deve de ser capaz de estabelecer comunicação com o ICD2.



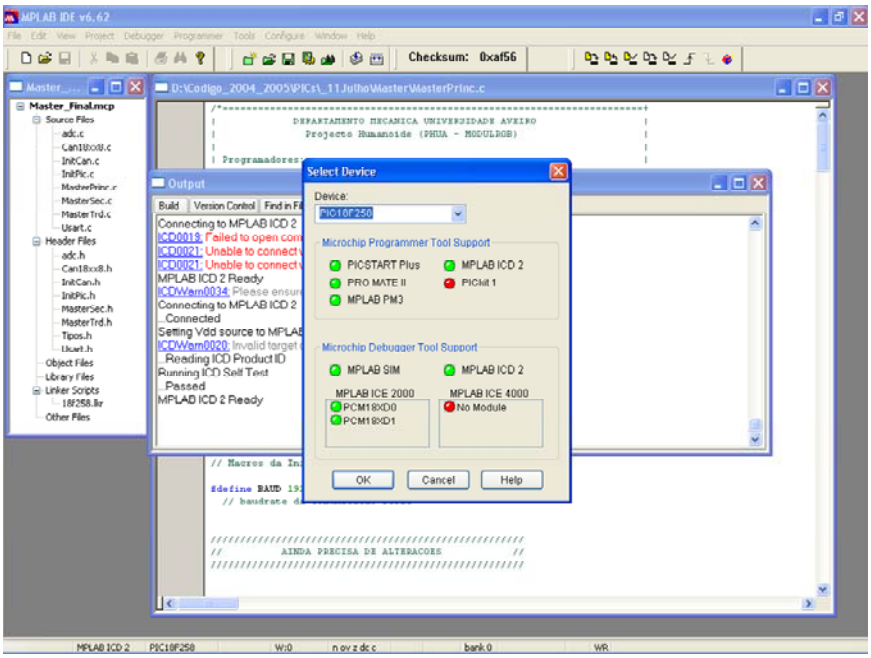
Anexo - Figura 14

Como é possível verificar as comunicações foram estabelecidas correctamente.



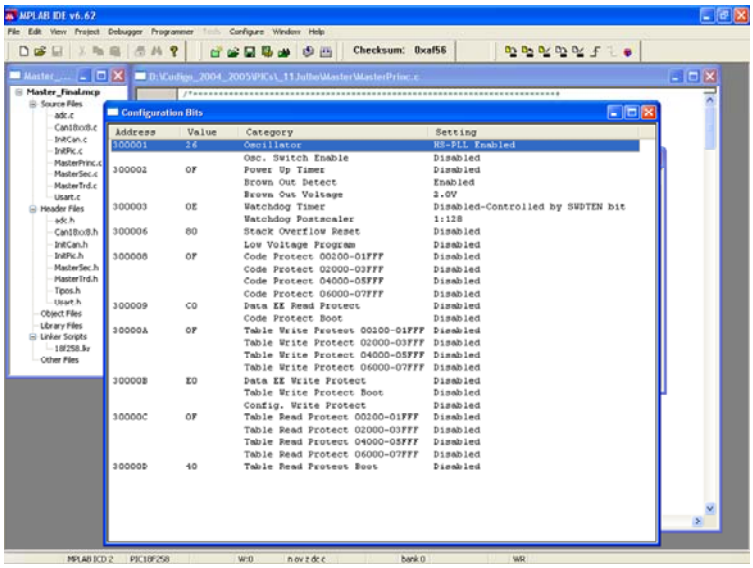
Anexo - Figura 15

No entanto continua a aparecer um Warning a indicar que o ID do microcontrolador não é válido. Este Warning pode várias ter várias causa. A primeira a indicar será a falta de alimentação do microcontrolador. Verificar a placa que se está utilizar e confirmar se a montagem está de acordo com o esquema que se encontra na documentação do ICD2.
A segunda causa pode ser um problema de configuração. Sempre que se cria um projecto temos que escolher para qual microcontrolador é destinado o código. Caso isso não se tenha feito é possível voltar escolher o microcontrolador que se pretende programar. Para isso tem-se que escolher o menu Configure e no sub-menu Select Device escolher o microcontrolador que se está a utilizar.



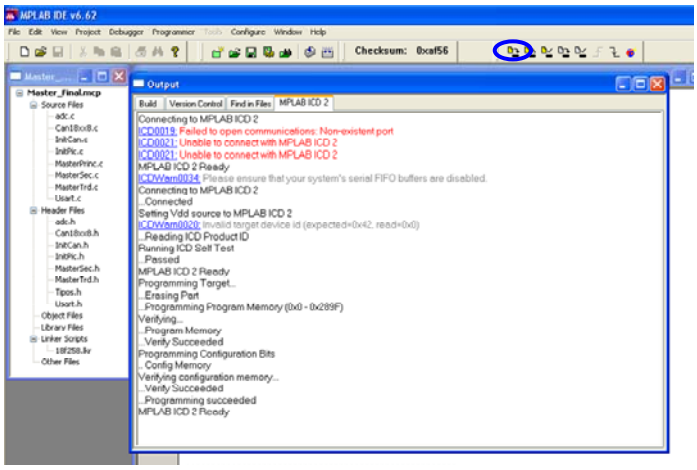
Anexo - Figura 16

Para além disto é necessário seleccionar os bits de configuração do microcontrolador. Apresenta-se na figura a configuração correcta.



Anexo - Figura 17

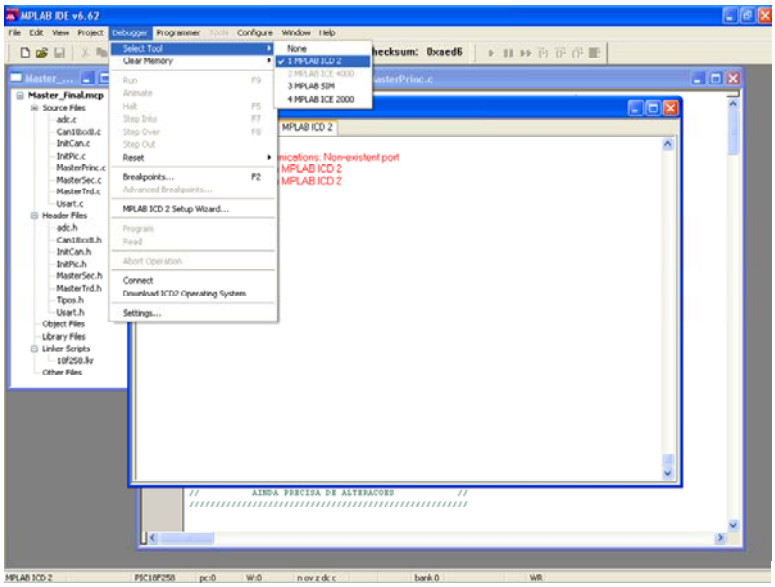
Após este procedimento deve se ter todo o sistema configurado e as comunicações devem estar estabelecidas. Podendo se compilar o código e programar o pic utilizando o botão que se indica na figura.



Anexo - Figura 18

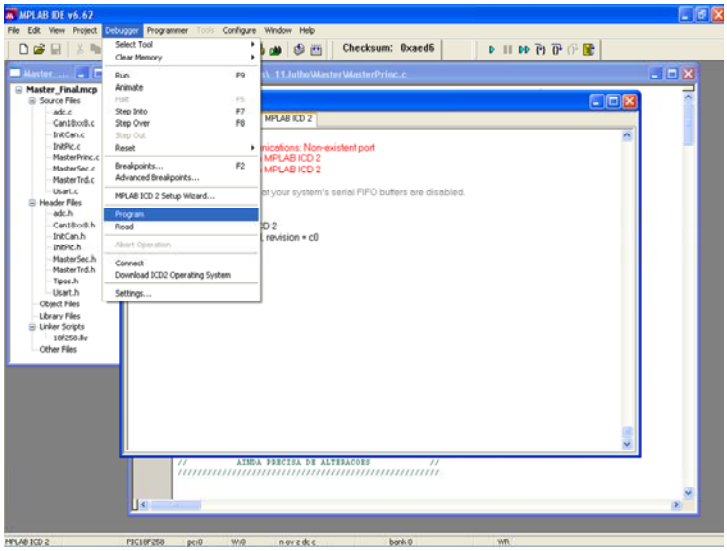
<u>A seguir será descrito o processo necessário para utilizar o ICD2 para efectuar o debugging de código a correr no microcontrolador.</u>

Na barra de Menus do MpLab existe o menu Debugger. No submenu Select Tool escolhemos o MpLAb ICD2.
Caso o MpLab não reconheça o ICD2 tem-se que efectuar o procedimento descrito anteriormente.



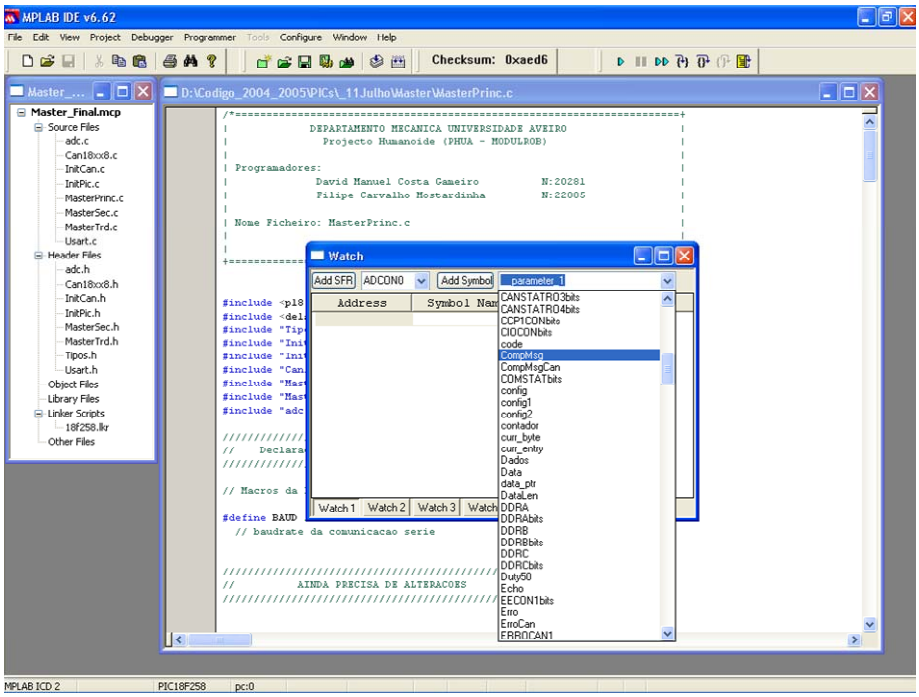
Anexo - Figura 19

Para o In Circuit Debugger (ICD) funcione é necessária uma rotina especifica que é progrmada em conjunto com o código que se desenvolveu no projecto. Esta rotina serve para entre outras coisas controlar todo o funcionamento do microcontrolador em modo de debugging. O MpLab encarrega-se de efectuar isto por nós. Basta escolher a opção Program no menu Debugger que o MpLab programa o microcontrolador com tudo o que é necessário.

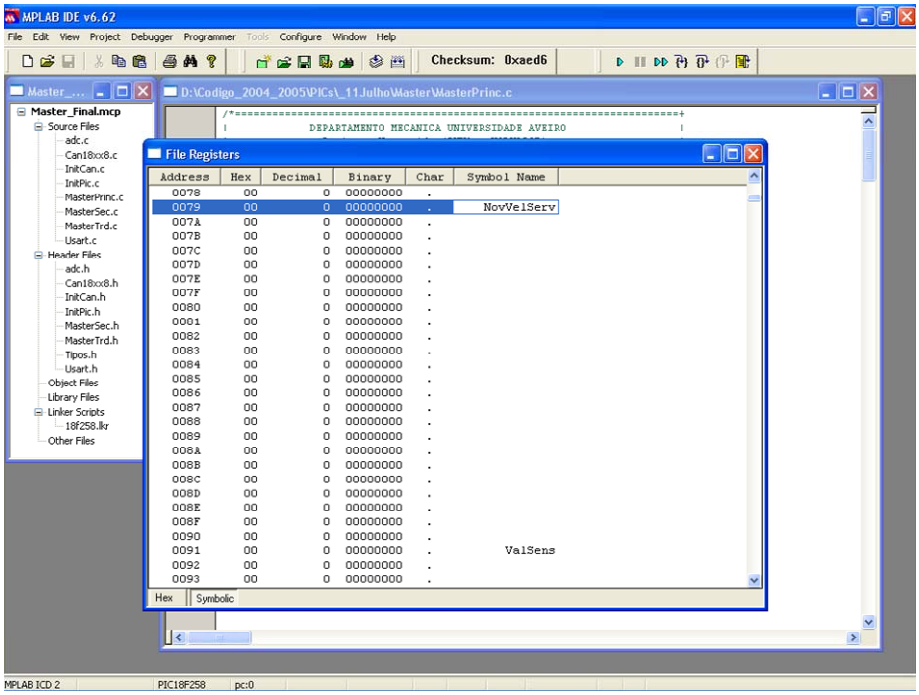


Anexo - Figura 20

Uma vez escolhido o debugger e progrmado o código aparece a barra de ferramentas do debugger. Aqui pode se colocar em ou interromper o funcionamento do microcontrolador para analisar o estado das variáveis e da memória na altura da interupção Basta no menu View escolher o submenu Watch ou File Register para se se visualizar todas as variáveis do código do microcontrolador.

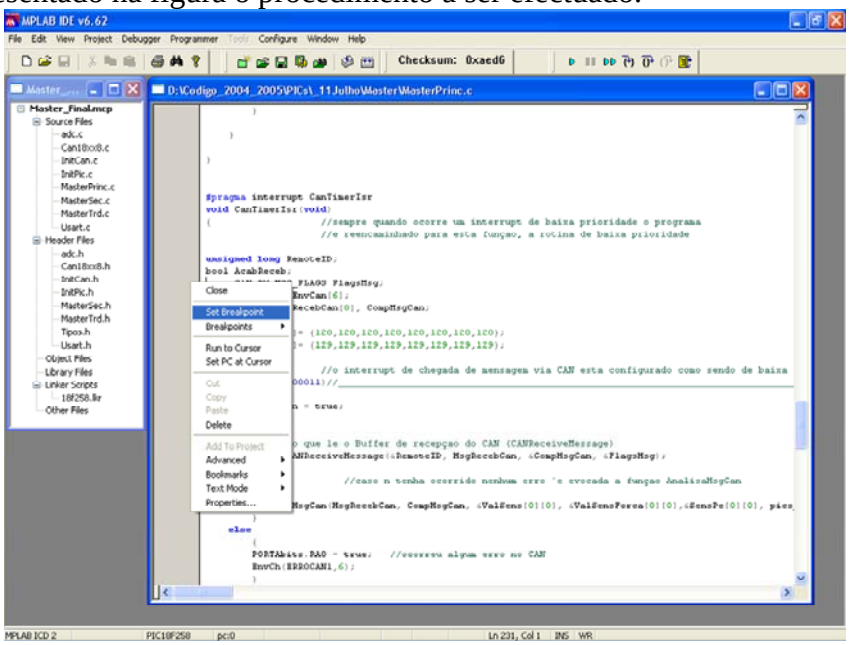


Anexo - Figura 21

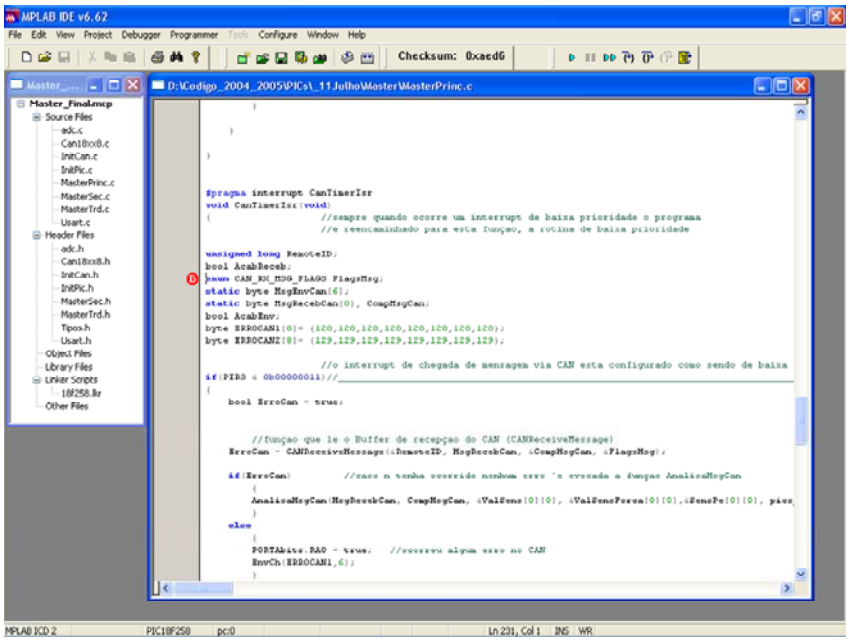


Anexo - Figura 22

Neste ambiente de desenvolvimento existe também a possibilidade de colocar break points no código. È apresentado na figura o procedimento a ser efectuado.



Anexo - Figura 23



Anexo - Figura 24

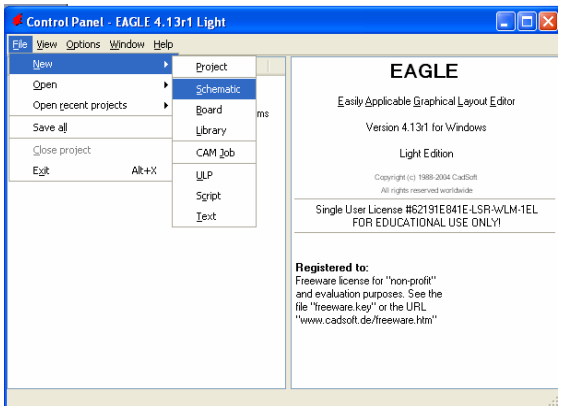
ANEXO II

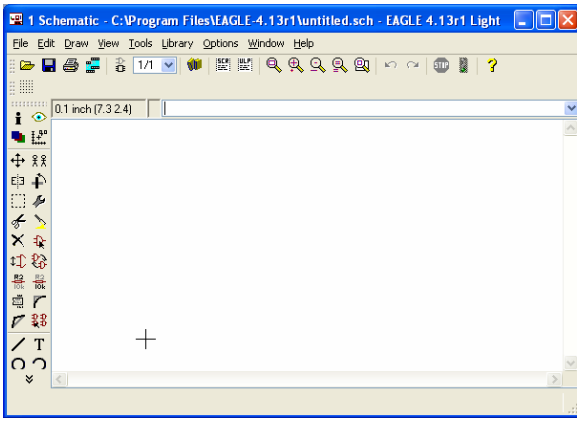
Quick Start Eagle Layout Editor

O editor de Eagle é uma ferramenta fácil de usar-se, poderosa para projecção de placas de circuito impresso (PCBs).

A seguir vão ser apresentados os passos necessários para elaborar uma placa em PCB.

1º Passo – criar um novo *schematic*, o Eagle abre uma nova janela onde se vai desenhar o circuito electrónico.

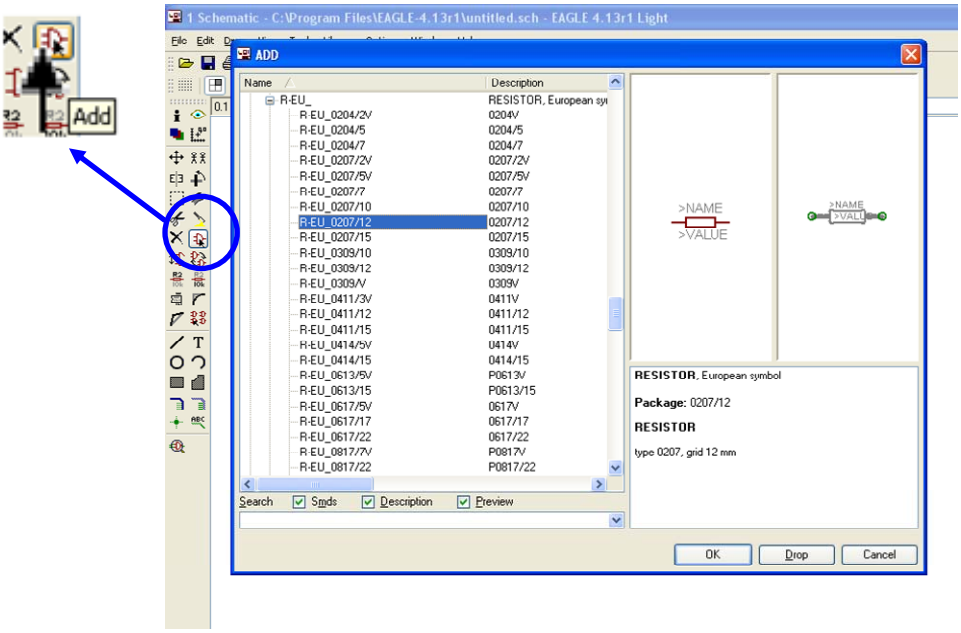




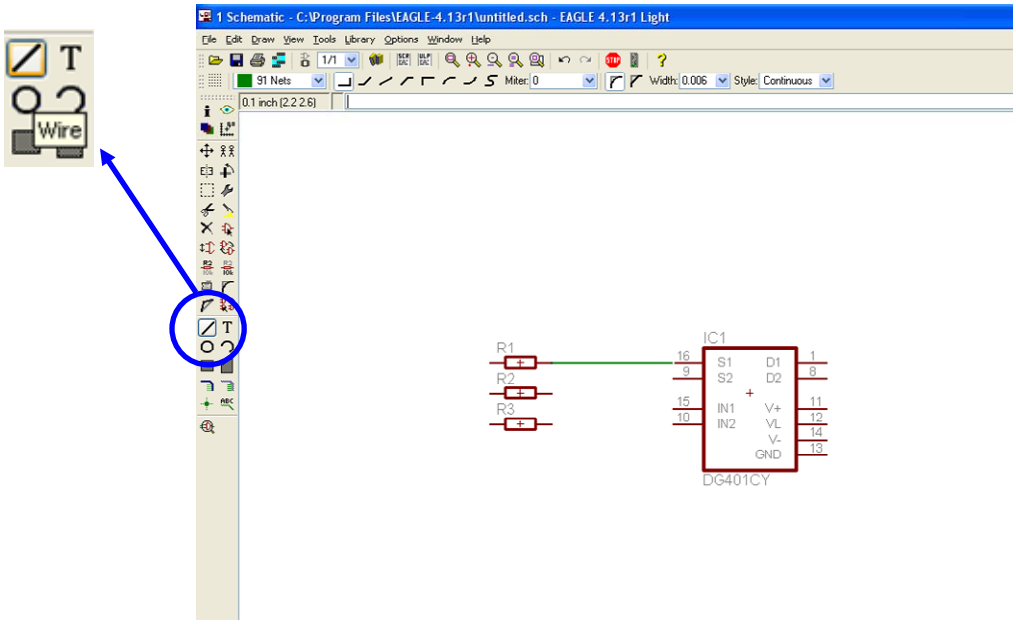
2º Passo – Desenhar circuito
Do lado esquerdo encontram-se os ícones das diversas ferramentas para permitir o desenho dos circuitos, as principais são:

	*Move* – Mover o seleccionado
	*Copy* – Copiar componentes
	*Change* – Permite alterar diferentes propriedades dos componentes e pistas
	*Delete* – apagar o seleccionado
	*Rotate* – Rodar o seleccionado
	*Name* – Dar nome ao componente ou ligação
	*Value* – dar Valor ao componente
	*Add* – Permite adicionar componentes da livraria
	*Bus* – Introduzir um bus de ligações
	*Wire* – Introduzir Ligação
	*Net* – Introduzir ligação do bus
	*Erc* – Extria uma lista de erros cometidos no desenho de circuito

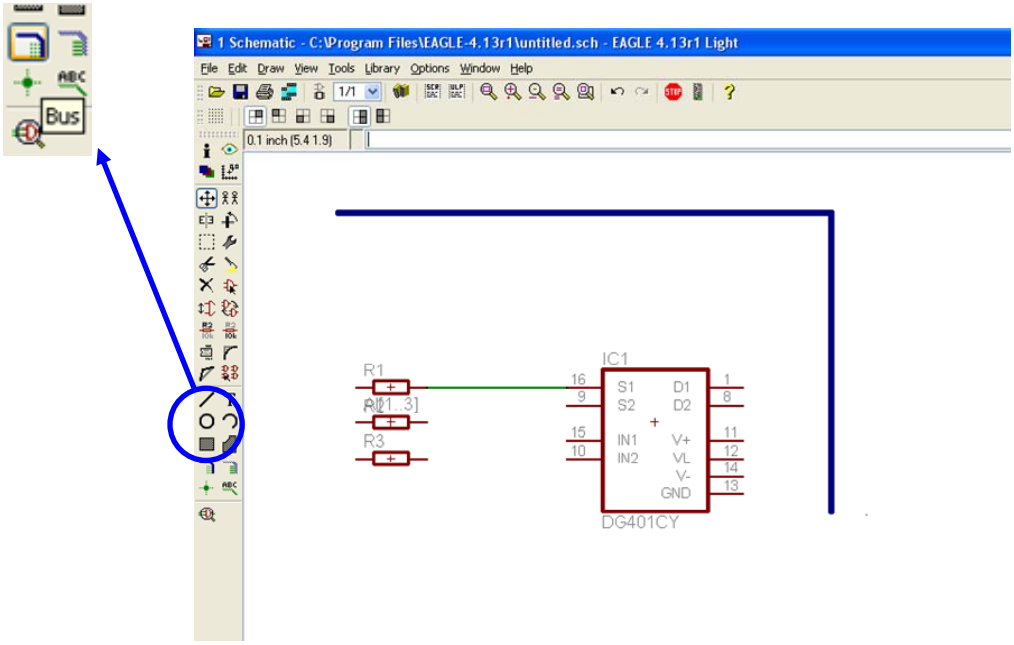
- *Introduzir um componente*, seleccionar o ícone ADD que vai abrir uma nova janela que contém uma livraria de componentes, seleccionar o componente pretendido.



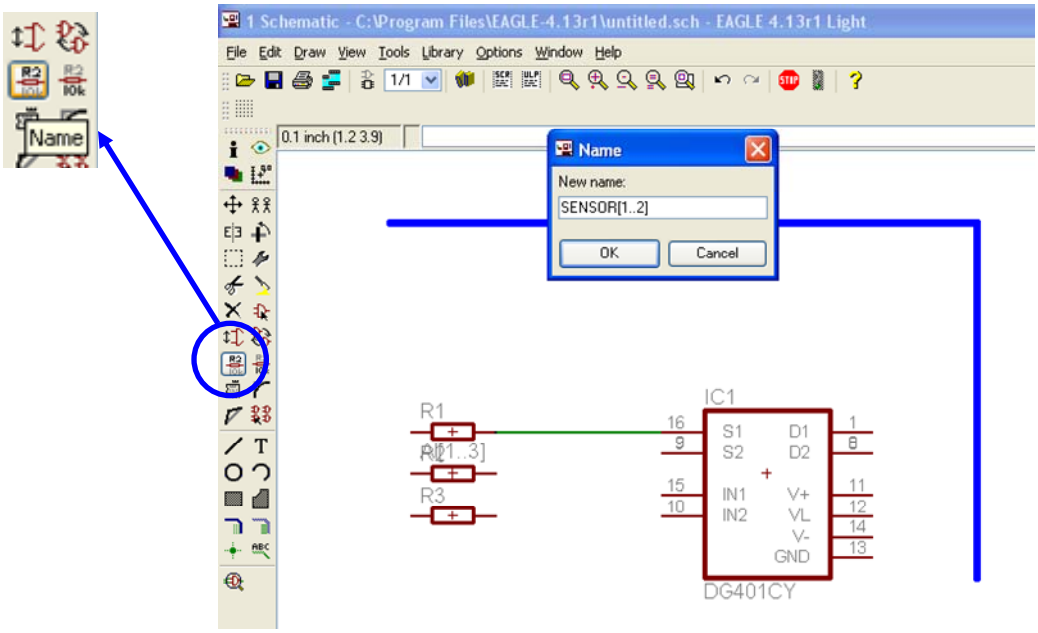
- *Ligar pinos dos componentes*, seleccionar o ícone WIRE e ligar com auxílio do rato os pinos que se pretendem ligar.



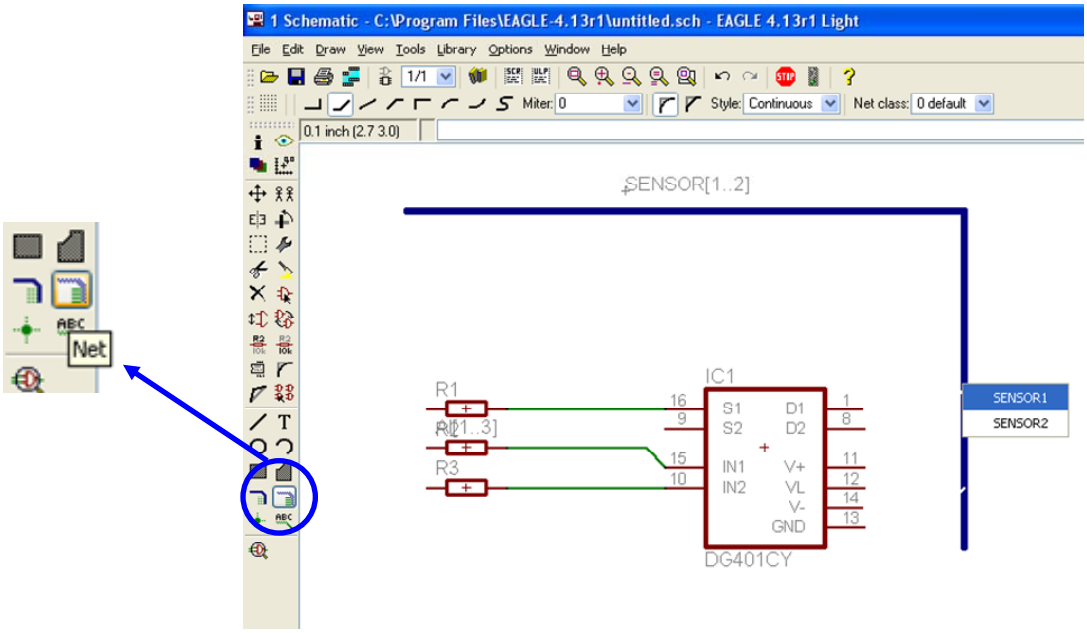
- *Criar um bus de ligações*, seleccionar o ícone BUS e desenhar o bus.



- *Dar nome ao Bus e definir o número de ligações que contêm*, para isso selecciona-se o ícone NAME e clica-se sobre o bus, aparece uma nova janela, primeiro escrevesse o nome e depois entre parênteses rectos o número inicial e final das ligações como ilustra a figura.

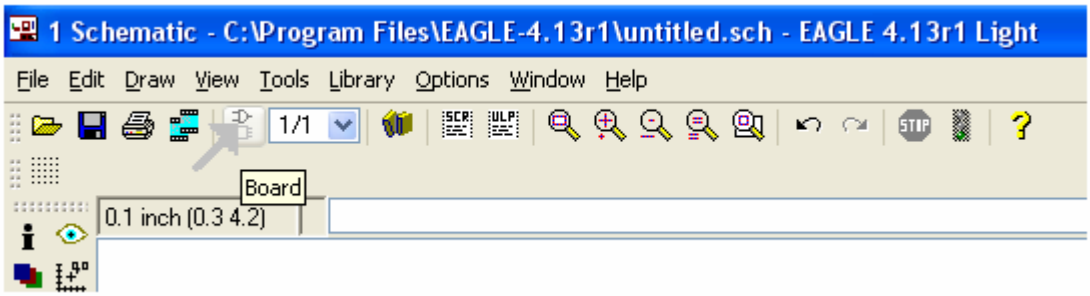


Em seguida seleccionar o ícone Net, depois clicar sobre o Bus, vai aparecer uma lista com as ligações escolher a que se pretende efectuar a ligação.



3º Passo – Desenhar Placa

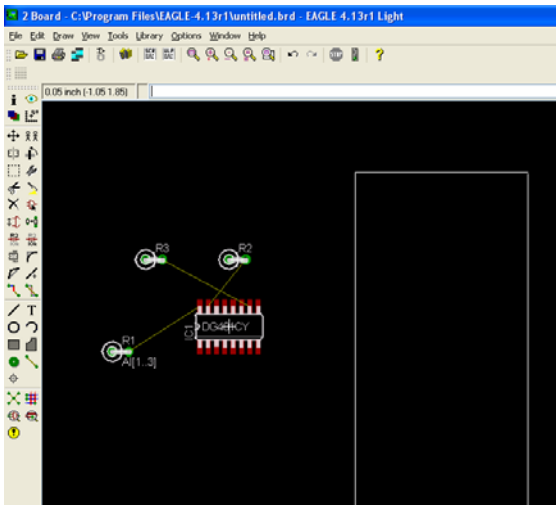
- *Passar para a janela BOARD*, seleccionar o ícone que se encontra na parte superior da janela *Schematic*, abre-se uma nova janela com um aspecto diferente e com novas ferramentas.

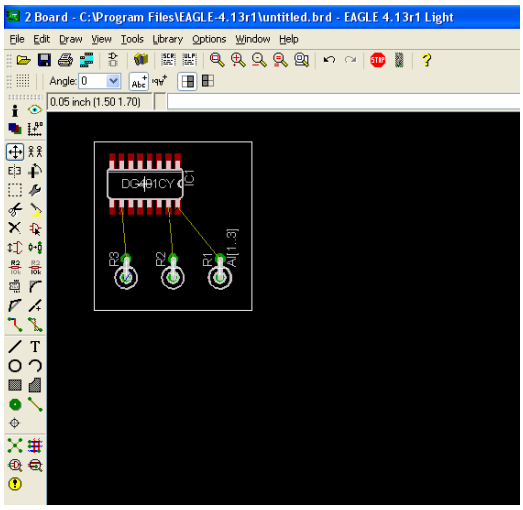


A seguir apresenta-se as ferramentas com maior utilidade.

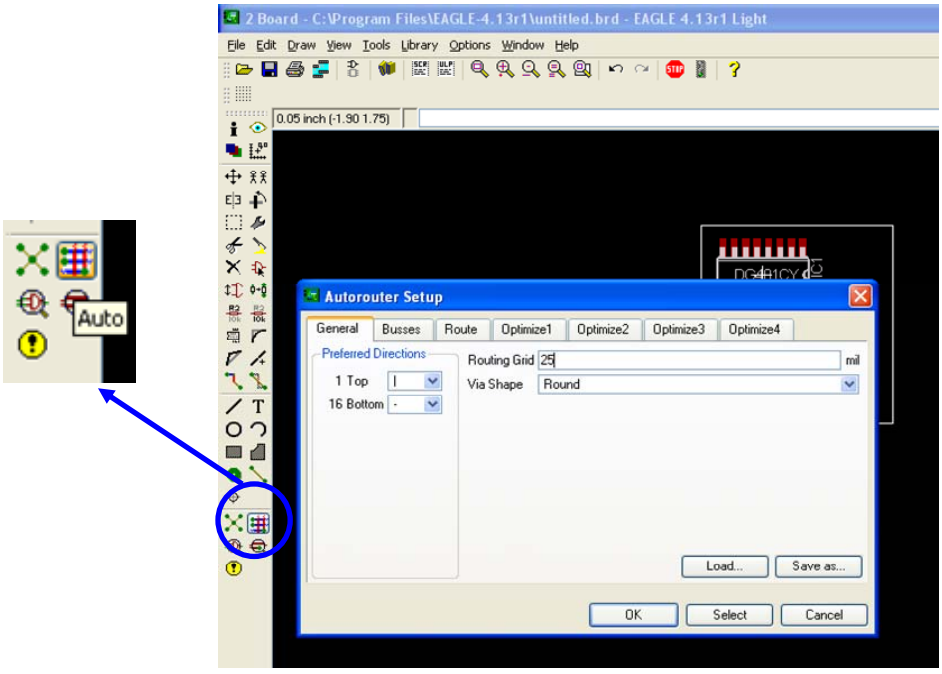
	Route – Desenha pista manualmente
	Ripup – Elimina pista manualmente
	Auto – Faz o auto route das pistas, ou seja, desenha todas as pistas da placa automaticamente.
	Ratsnest – re-arranjar as ligações
	Polygon – Cria um polígono, ferramenta usada para criar um plano de massa

Quando se entra nesta nova janela os componentes aparecem desordenados e fora dos limites da placa, nesta operação arrasta-se os componentes para dentro dos limites da placa ordenando-os, e dimensiona-se a placa.

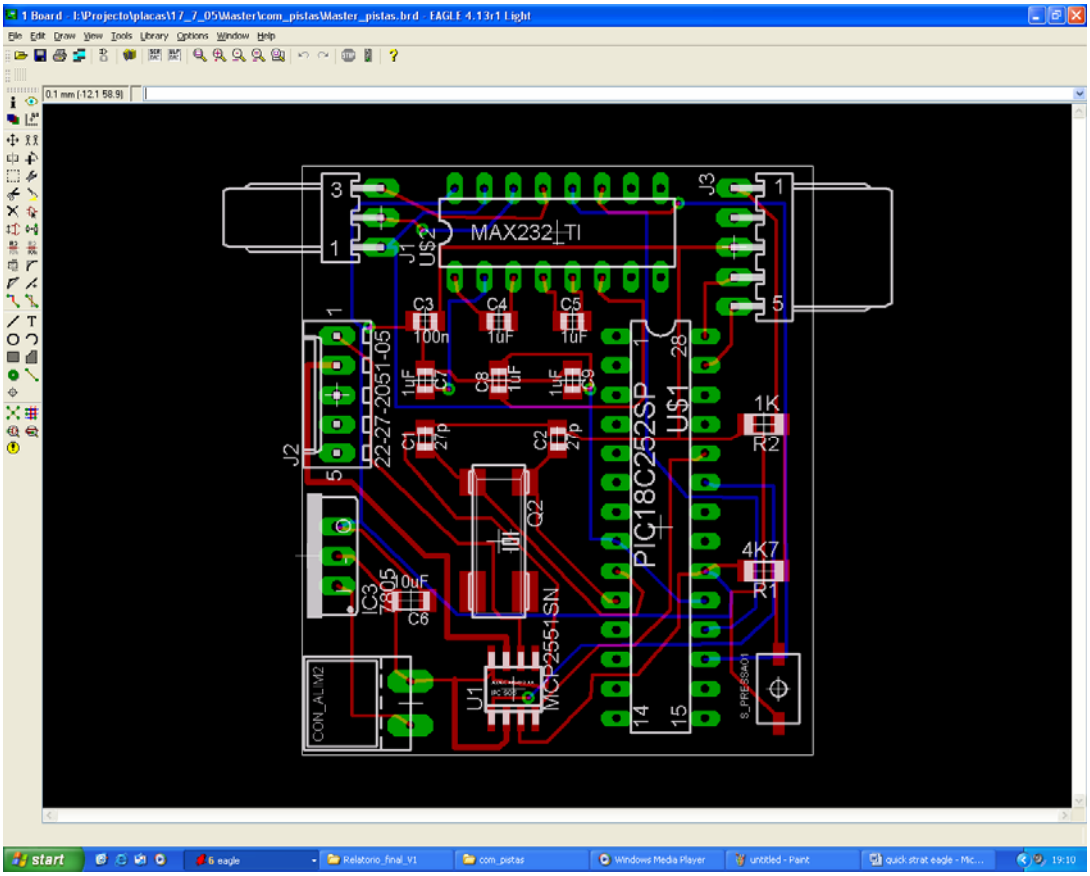




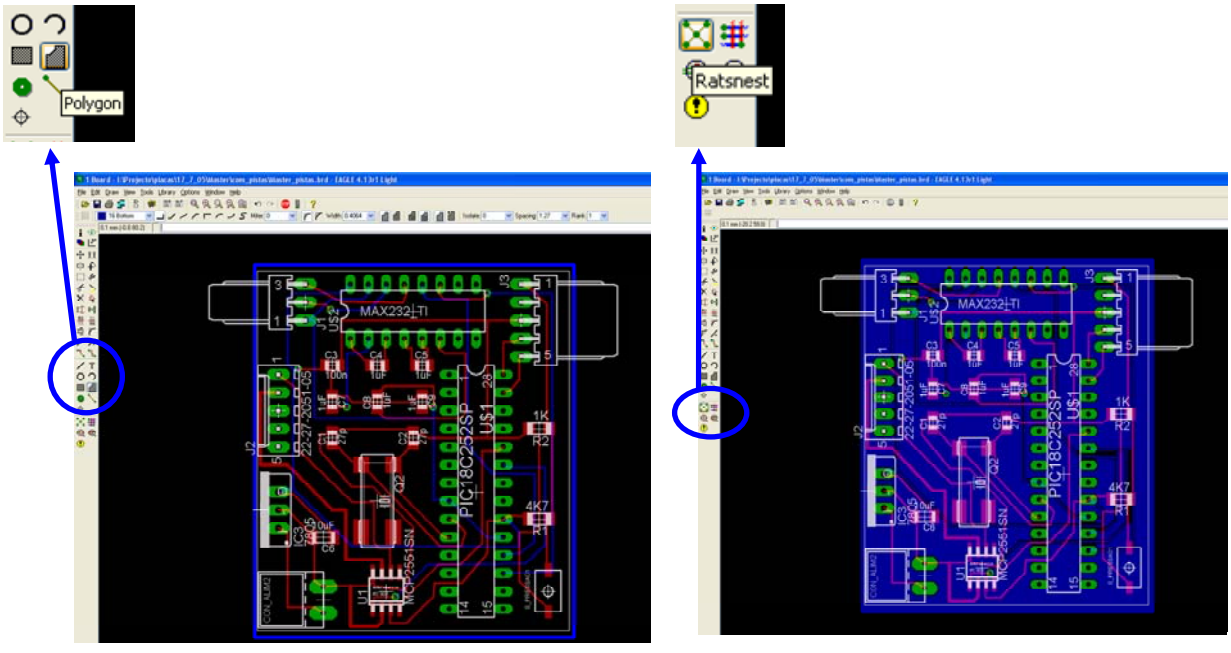
- *Auto Route*, nesta operação o Eagle cria as pistas automaticamente. Ao seleccionar o ícone AUTO abre-se uma nova janela. Nesta janela definem-se diversos parâmetros: largura das pistas, placa de face simples ou dupla etc.



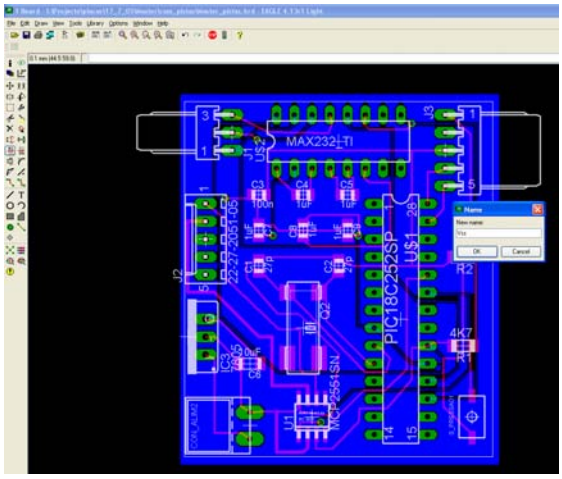
A figura abaixo ilustra uma placa após aplicação do Auto Route.

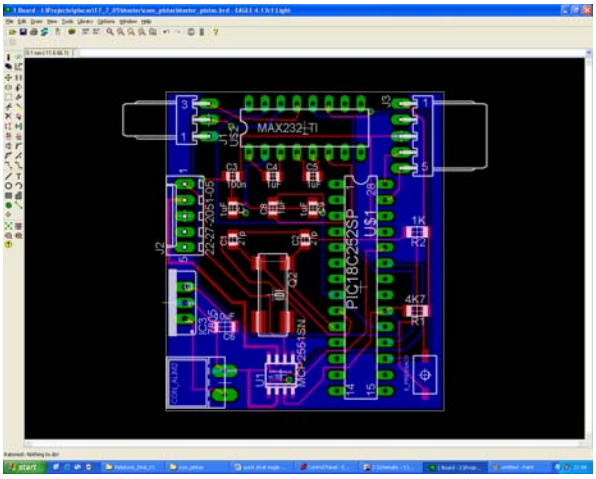


- *Criação de um plano de massa*, em primeiro é necessário desenhar um polígono á volta da placa, para isso clica-se sobre o ícone POLYGON e desenha-se, para que a operação fique concluída clica-se sobre o icono RATSNEST e este desenha um plano sobre a placa.



O último passo necessário é atribuir um nome ao plano, seleccionar o ícone NAME e clicar sobre o plano já criado, o nome que tem ser o que foi dado á massa no circuito.





ANEXO VI

Circuito Electrónico Placa Expansão sensores força

ANEXO VII

Circuito Electrónico Placa Expansão Inclinómetro

ANEXO VIII

Circuito Electrónico Placa Master

ANEXO IX

Circuito Electrónico Placa Expansão Master

ANEXO X

Circuito Electrónico Giroscópio

ANEXO XI

Circuito Electrónico Placa Alimentação I

ANEXO XII

Circuito Electrónico Placa Alimentação II

ANEXO XIII

Circuito Electrónico Placa Alimentação III

ANEXO XIV

Desenho técnico suporte sensores força

ANEXO XV

Desenho técnico placa de acrílico

ANEXO XVI

Código dos Microcontroladores

ANEXO III - Esquema Representativo do envio de mensagens do Master para os Slaves

Envio de um comando da unidade principal de processamento para o Master.
Comunicação Série Rs232

2º byte	3º byte	4º byte
0 1 0 0 0 0 1 1	Ângulo	Velocidade

O Master recebe uma mensagem com três bytes. O primeiro byte está reservado para identificar o tipo de mensagem, o segundo é utilizado geralmente para enviar a posição do servo que nós queremos movimentar e o terceiro é utilizado para enviar a velocidade a qual ele deve efectuar o movimento. Qualquer alteração de posição de uma dos servomotores é escrita nas matrizes NovPosServ ou NovVelServ.

O Master envia mensagens aos Slaves a uma frequência de 10kHz. As matrizes NovPosServ e NovVelServ são varridas ciclicamente. Sempre que tenha havido uma alteração nessas matrizes esses novos valores são enviados ao respectivo slave.

Array principal de registo das posições de todos os servomotores do sistema.
Nome: NovPosServ[8][3]

	Posição referencia dos Motores de cada Placa		
1ºSlave	Posição Servo 1	Posição Servo 2	Posição Servo 3
2ºSlave	Posição Servo 1	Posição Servo 2	Posição Servo 3
…	…	…	…
7ºSlave	Posição Servo 1	Posição Servo 2	Posição Servo 3
8ºSlave	Posição Servo 1	Posição Servo 2	Posição Servo 3

Array principal de registo dos valores sensoriais de do binário exercido pelos servomotores de todos os slaves no master.
Nome: NovVelServ[8][3]

	Velocidade referencia dos Motores de cada Placa		
1ºSlave	Velocidade Servo 1	Velocidade Servo 2	Velocidade Servo 3
2ºSlave	Velocidade Servo 1	Velocidade Servo 2	Velocidade Servo 3
…	…	…	…
7ºSlave	Velocidade Servo 1	Velocidade Servo 2	Velocidade Servo 3
8ºSlave	Velocidade Servo 1	Velocidade Servo 2	Velocidade Servo 3

Mensagem enviada pelo CAN do Master para o Slave

1ºByte	2ºByte	3ºByte	4ºByte	5ºByte	6ºByte	7ºByte	8ºByte
Posição Servo 1	Posição Servo 2	Posição Servo 3	Velocidade Servo 1	Velocidade Servo 2	Velocidade Servo 3	--	--

CAN BUS

2º Slave (por exemplo)

pwm1

velocidade1

ANEXO IV - Esquema Representativo do registo de valores Sensoriais no Master

Master

Array principal de registo dos valores sensoriais de posição de todos os slaves no master.
Nome: ValSens[8][3]

	Posição dos Motores de cada Placa		
1ºSlave	Valor do Potenciómetro 1.	Valor do Potenciómetro 2.	Valor do Potenciómetro 3.
2ºSlave	Valor do Potenciómetro 1.	Valor do Potenciómetro 2.	Valor do Potenciómetro 3.
…	…	…	…
7ºSlave	Valor do Potenciómetro 1.	Valor do Potenciómetro 2.	Valor do Potenciómetro 3.
8ºSlave	Valor do Potenciómetro 1.	Valor do Potenciómetro 2.	Valor do Potenciómetro 3.

Array principal de registo dos valores sensoriais de do binário exercido pelos servomotores de todos os slaves no master.
Nome: ValSensForca[8][3]

	Posição dos Motores de cada Placa		
1ºSlave	Valor do Servomotor 1.	Valor do Potenciómetro 2.	Valor do Potenciómetro 3.
2ºSlave	Valor do Potenciómetro 1.	Valor do Potenciómetro 2.	Valor do Potenciómetro 3.
…	…	…	…
7ºSlave	Valor do Potenciómetro 1.	Valor do Potenciómetro 2.	Valor do Potenciómetro 3.
8ºSlave	Valor do Potenciómetro 1.	Valor do Potenciómetro 2.	Valor do Potenciómetro 3.

Array principal de registo dos valores sensoriais.adicionais de todos os slaves no master. Nome: ValSens[8][3]

	Posição dos Sensores de cada Placa			
1ºSlave	Valor do sensor adicional 1	Valor do sensor adicional 2	Valor do sensor adicional 3	Valor do sensor adicional 4
2ºSlave	Valor do sensor adicional 1	Valor do sensor adicional 2	Valor do sensor adicional 3	Valor do sensor adicional 4
…	…	…	…	…
7ºSlave	Valor do sensor adicional 1	Valor do sensor adicional 2	Valor do sensor adicional 3	Valor do sensor adicional 4
8ºSlave	Valor do sensor adicional 1	Valor do sensor adicional 2	Valor do sensor adicional 3	Valor do sensor adicional 4

2º Slave (por exemplo)

Array local de registo de valores sensoriais .dos slaves Nome: ValSens[0..9]

ValSens[0]	Valor do Potenciómetro 1.
ValSens[1]	Valor do Potenciómetro 2.
ValSens[2]	Valor do Potenciómetro 3.
ValSens[3]	Tensão na resistência de potência 1
ValSens[4]	Tensão na resistência de potência 2
ValSens[5]	Tensão na resistência de potência 3
ValSens[6]	Valor do sensore adicional 1
ValSens[7]	Valor do sensore adicional 2
ValSens[8]	Valor do sensore adicional 3
ValSens[9]	Valor do sensore adicional 4

1.ª Mensagem enviada pelo CAN do Slave para o Mestre

1ºByte	2ºByte	3ºByte	4ºByte	5ºByte	6ºByte	7ºByte	8ºByte
HostMcu	Identificador de Mensagem	ValSens[0]	ValSens[1]	ValSens[2]	ValSens[3]	ValSens[4]	ValSens[5]

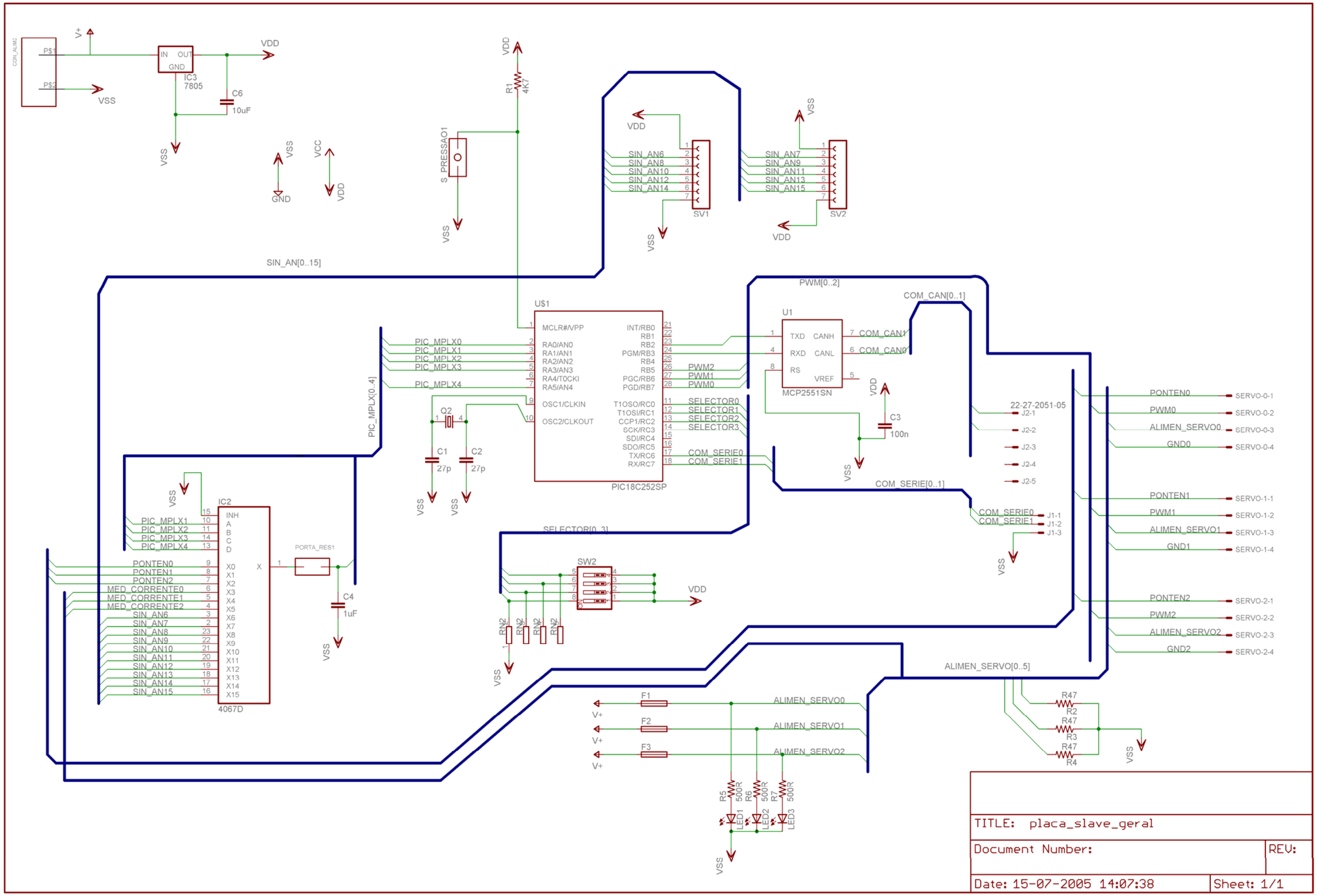
2.ª Mensagem enviada pelo CAN do Slave para o Mestre

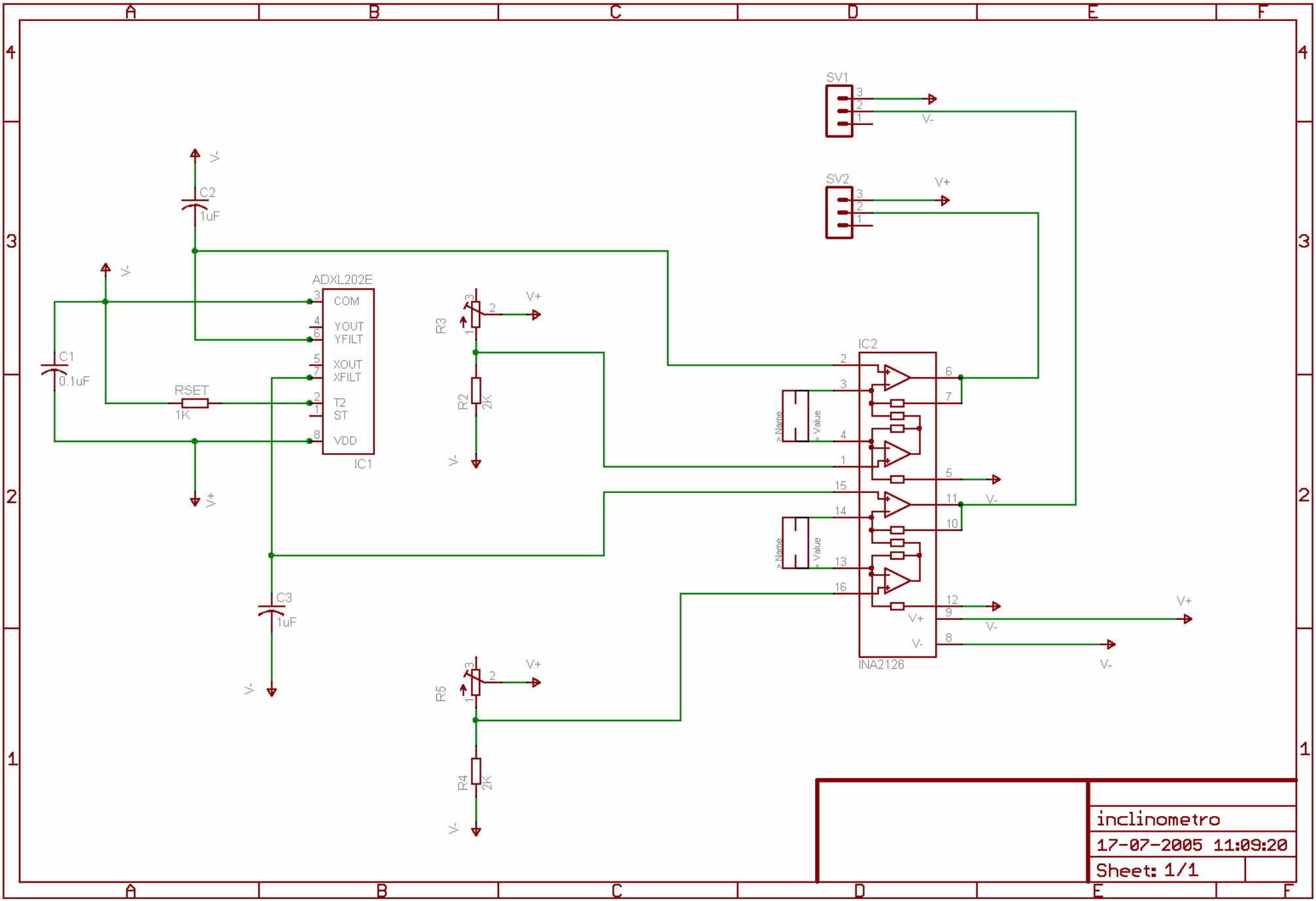
1ºByte	2ºByte	3ºByte	4ºByte	5ºByte	6ºByte	7ºByte	8ºByte
HostMcu	Identificador de Mensagem	ValSens[6]	ValSens[7]	ValSens[8]	ValSens[9]	--	--

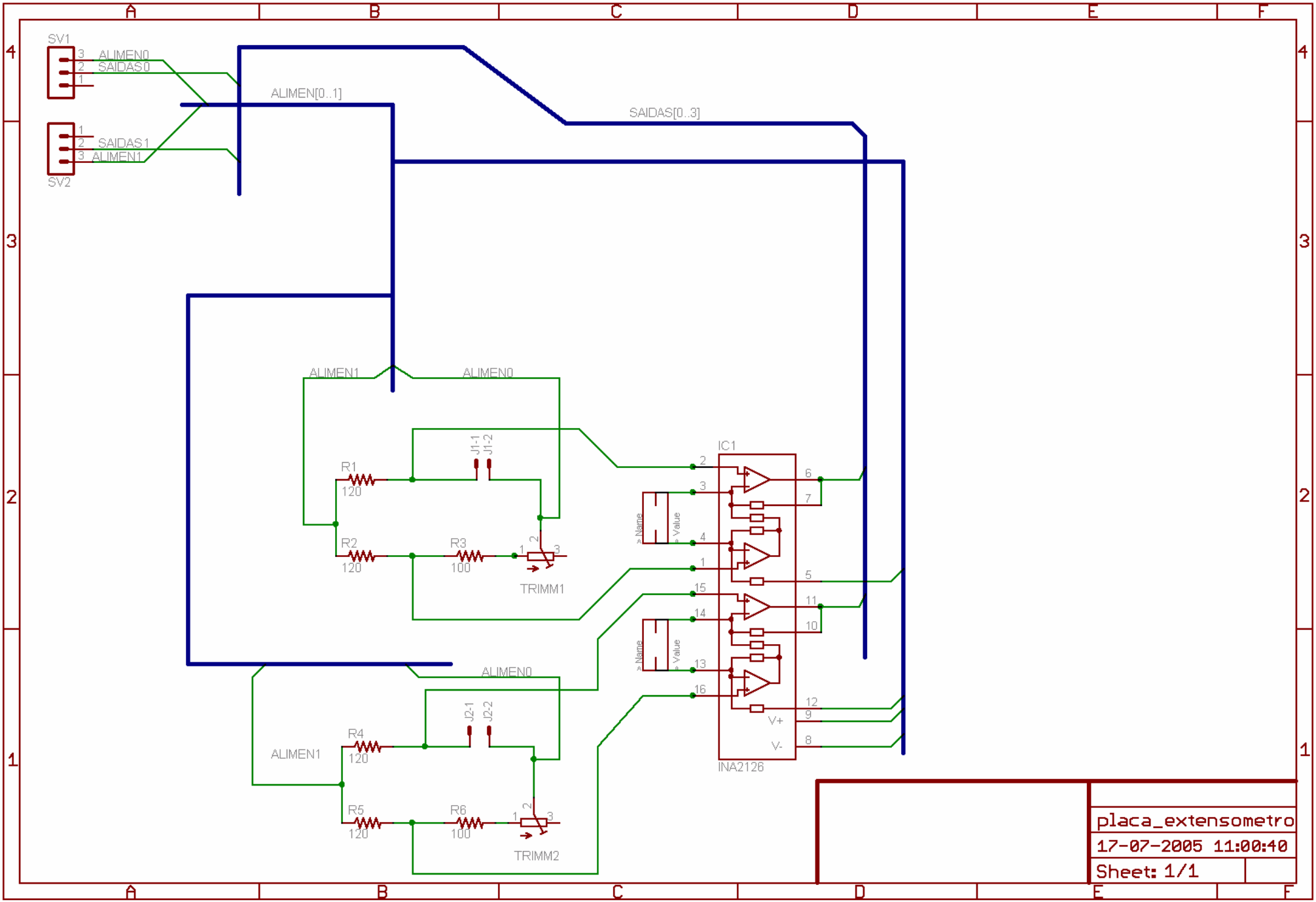
CAN BUS

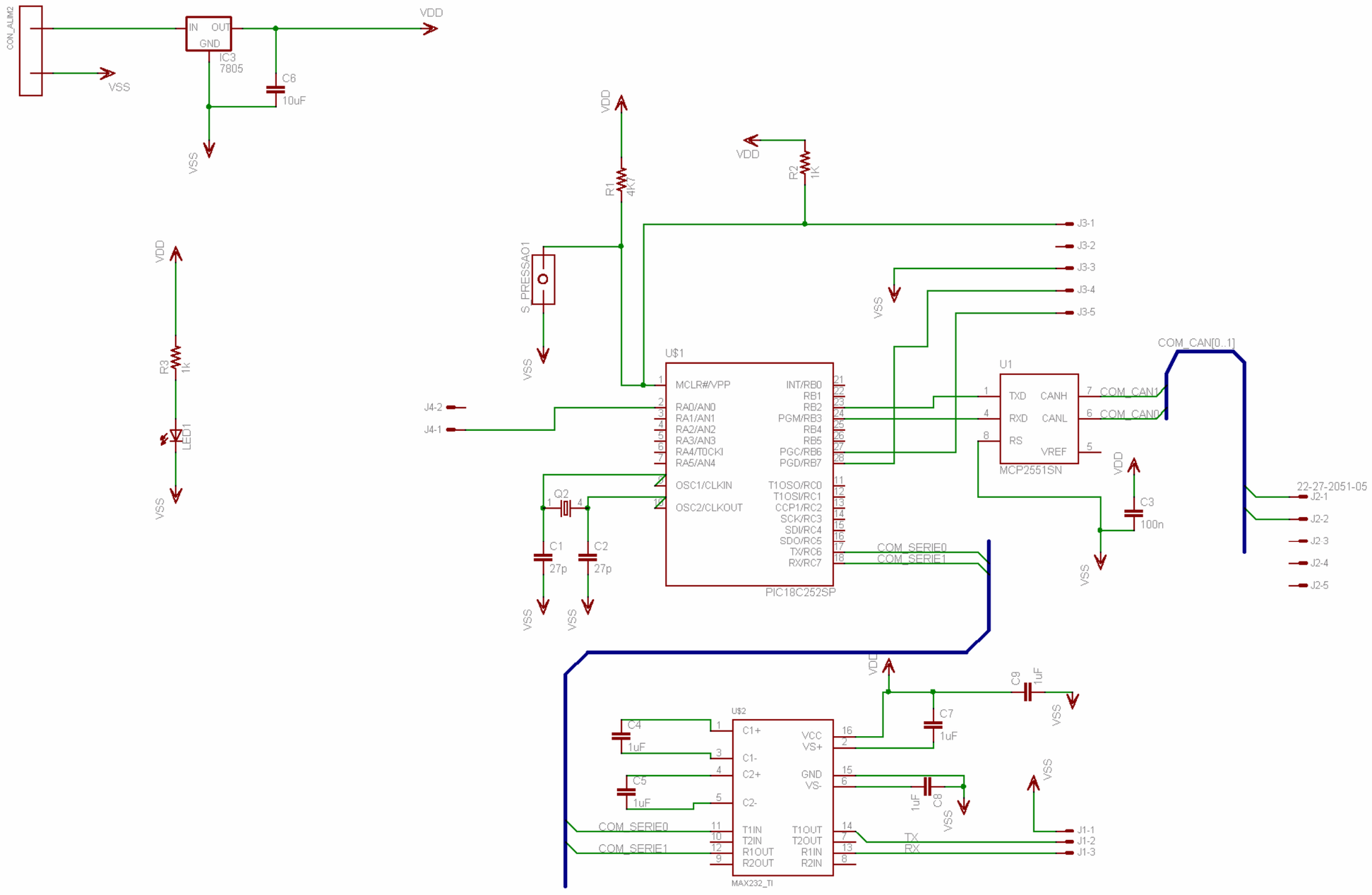
Sempre que um Slave recebe uma mensagem pelo CAN que seja destinada para ele, este responde com duas mensagens diferentes. A primeira mensagem que ele envia está relacionada com valores sensoriais dos motores. Após ter sido enviada a primeira mensagem com sucesso é enviada a segunda que contém os valores sensoriais dos sensores adicionais de cada placa.

ANEXO IV – Circuito Electrónico Placa Principal

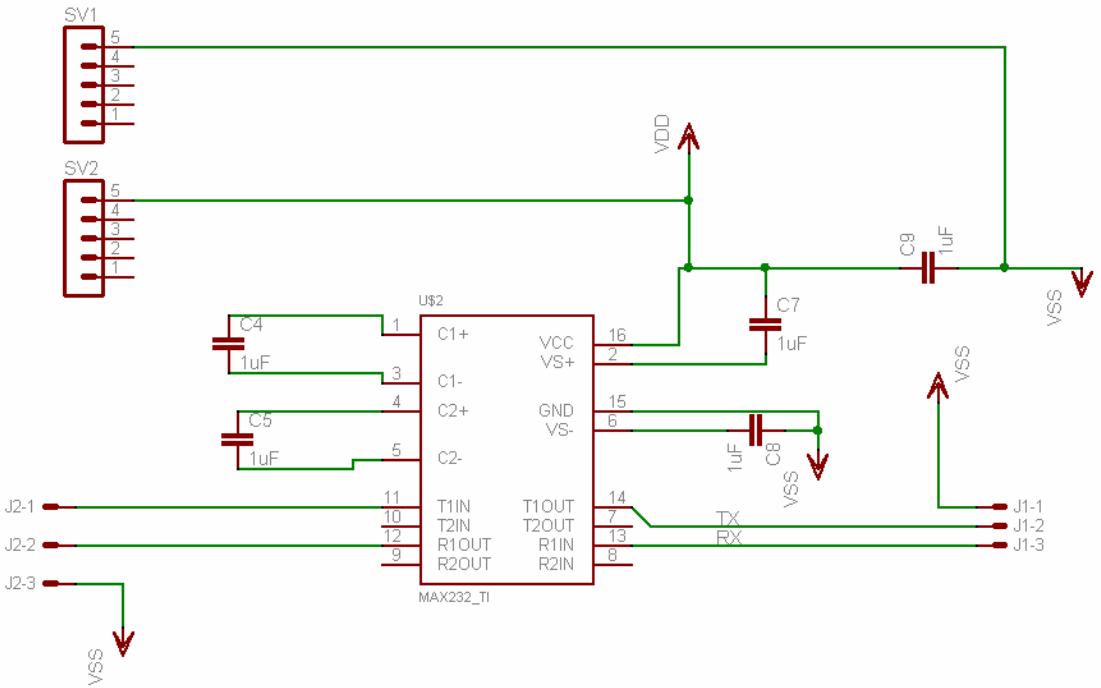



SV1
SV2
V-
V+
C2
1uF
C1
0.1uF
RSET
1K
ADXL202E
COM
YOUT
YFILT
XOUT
XFILT
T2
ST
VDD
IC1
R3
R2
2K
R5
R4
2K
C3
1uF
IC2
INA2126
inclinometro
17-07-2005 11:09:20
Sheet: 1/1

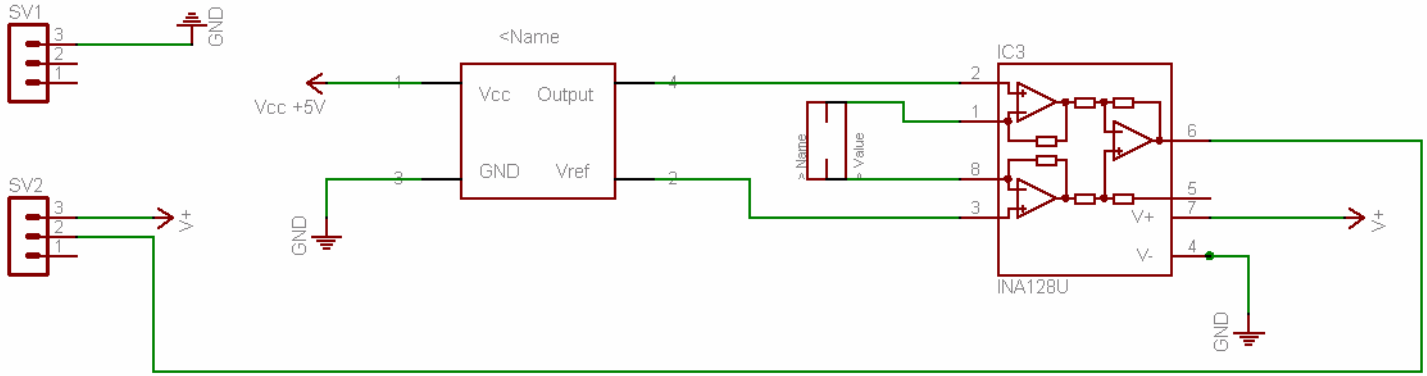

SV1
3 ALIMEN0
2 SAIDAS0
1
1
2 SAIDAS1
3 ALIMEN1
SV2
ALIMEN[0..1]
SAIDAS[0..3]
ALIMEN1
ALIMEN0
J1-1
J1-2
R1
120
R2
120
R3
100
TRIMM1
IC1
Name
Value
INA2126
V+
V-
ALIMEN0
ALIMEN1
J2-1
J2-2
R4
120
R5
120
R6
100
TRIMM2
placa_extensometro
17-07-2005 11:00:40
Sheet: 1/1



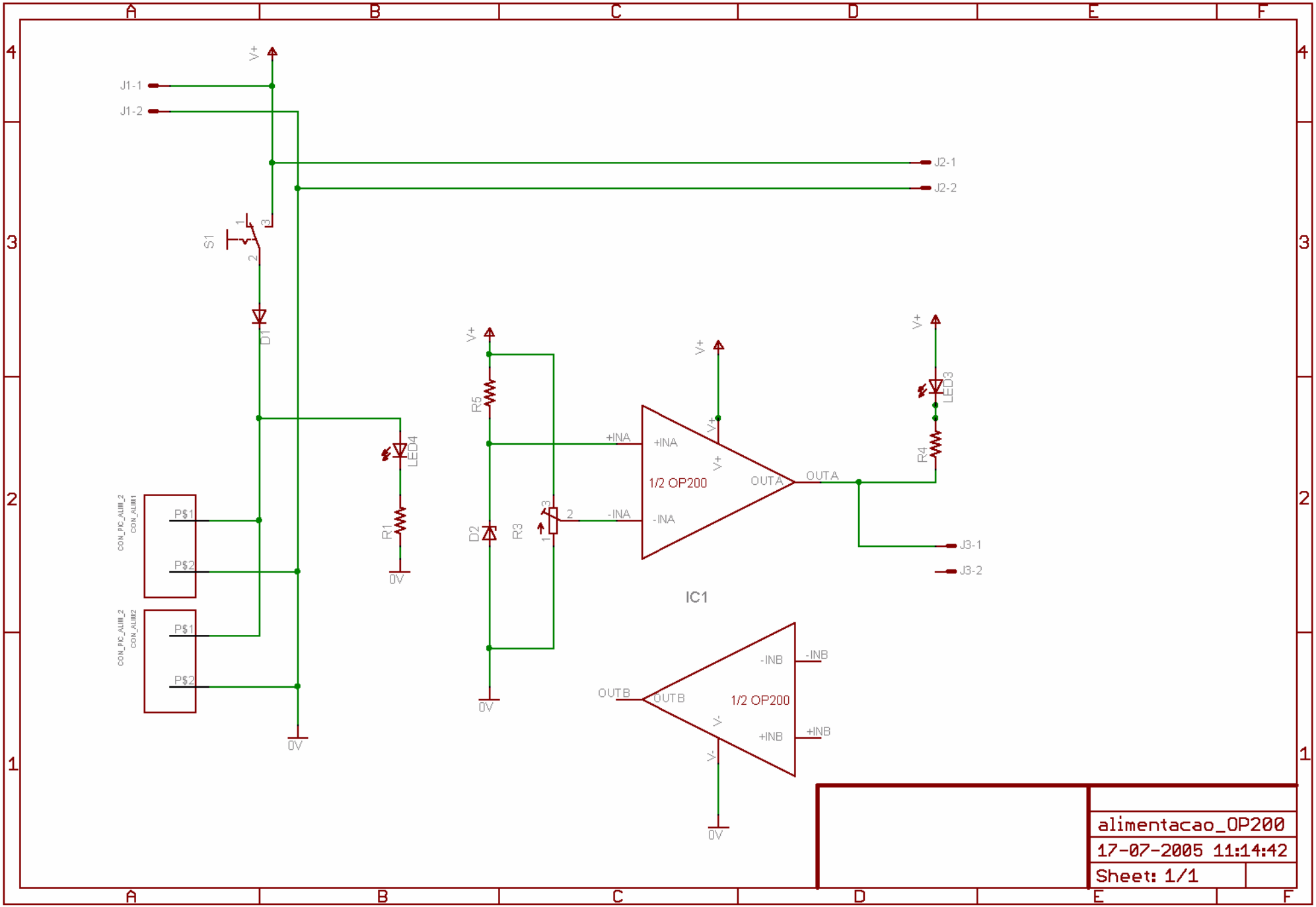
TITLE: Master	
Document Number:	REV:
Date: 17-07-2005 11:39:42	Sheet: 1/1

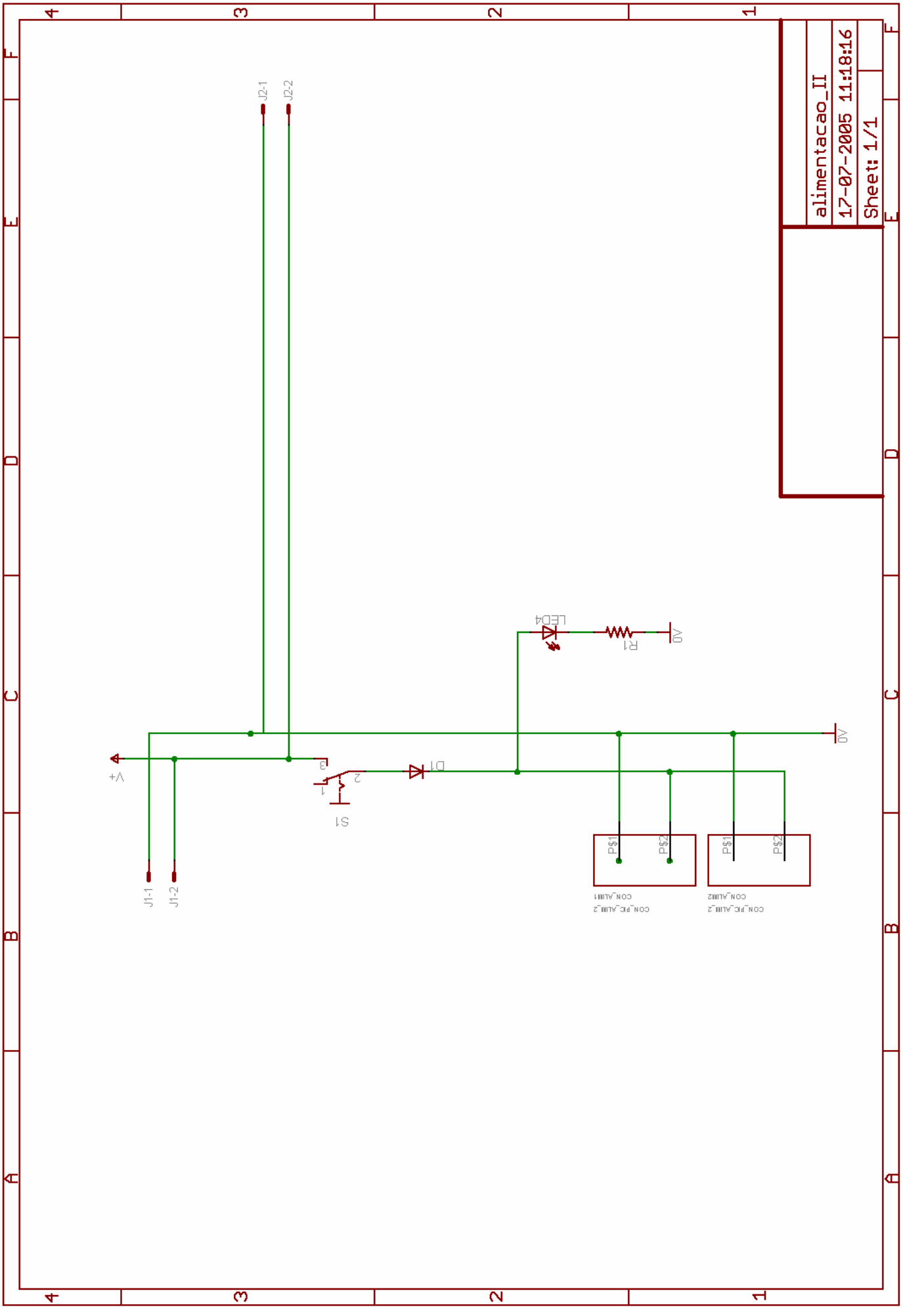


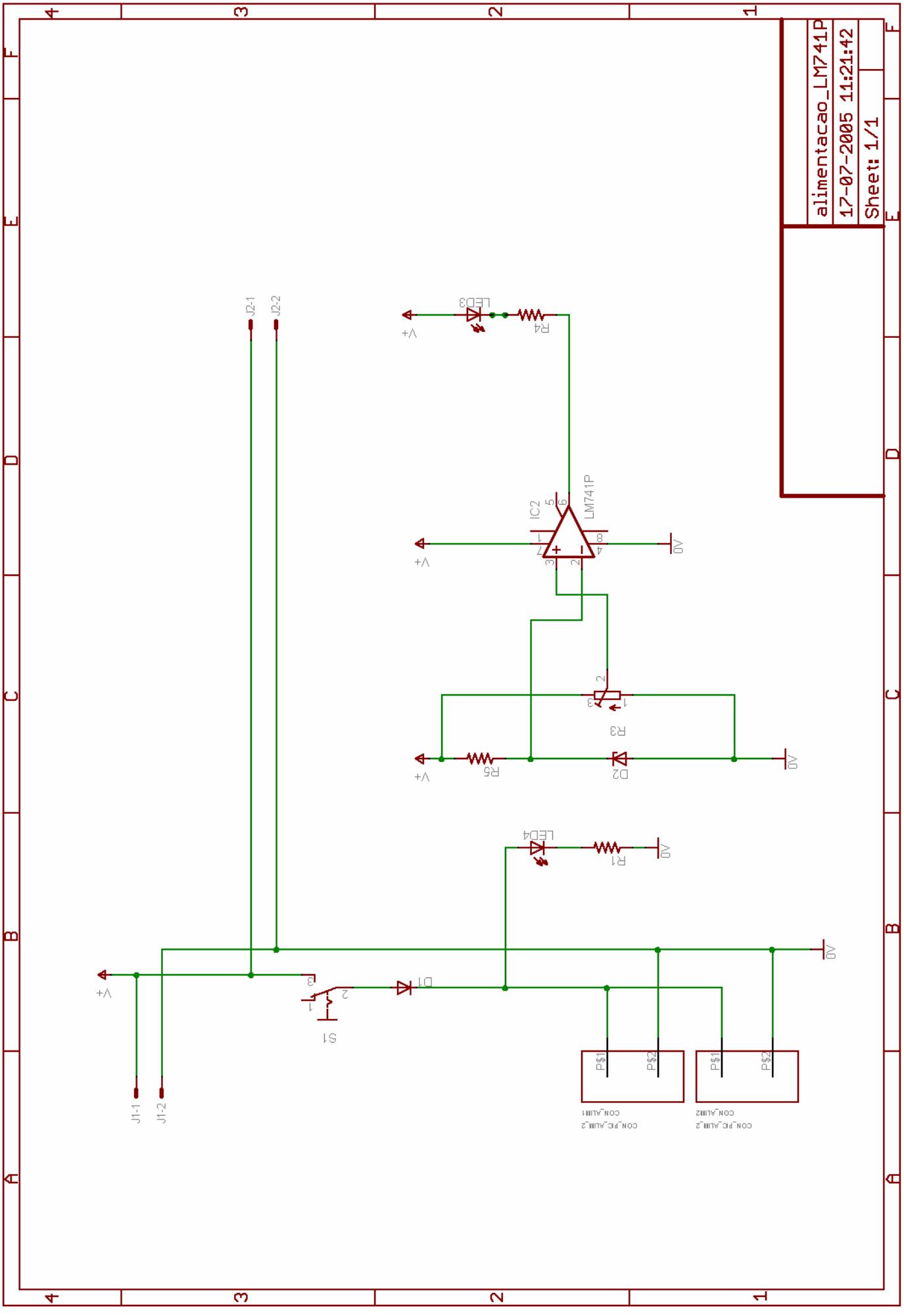
Master_18_5_05	
17-07-2005 10:27:16	
Sheet: 1/1	

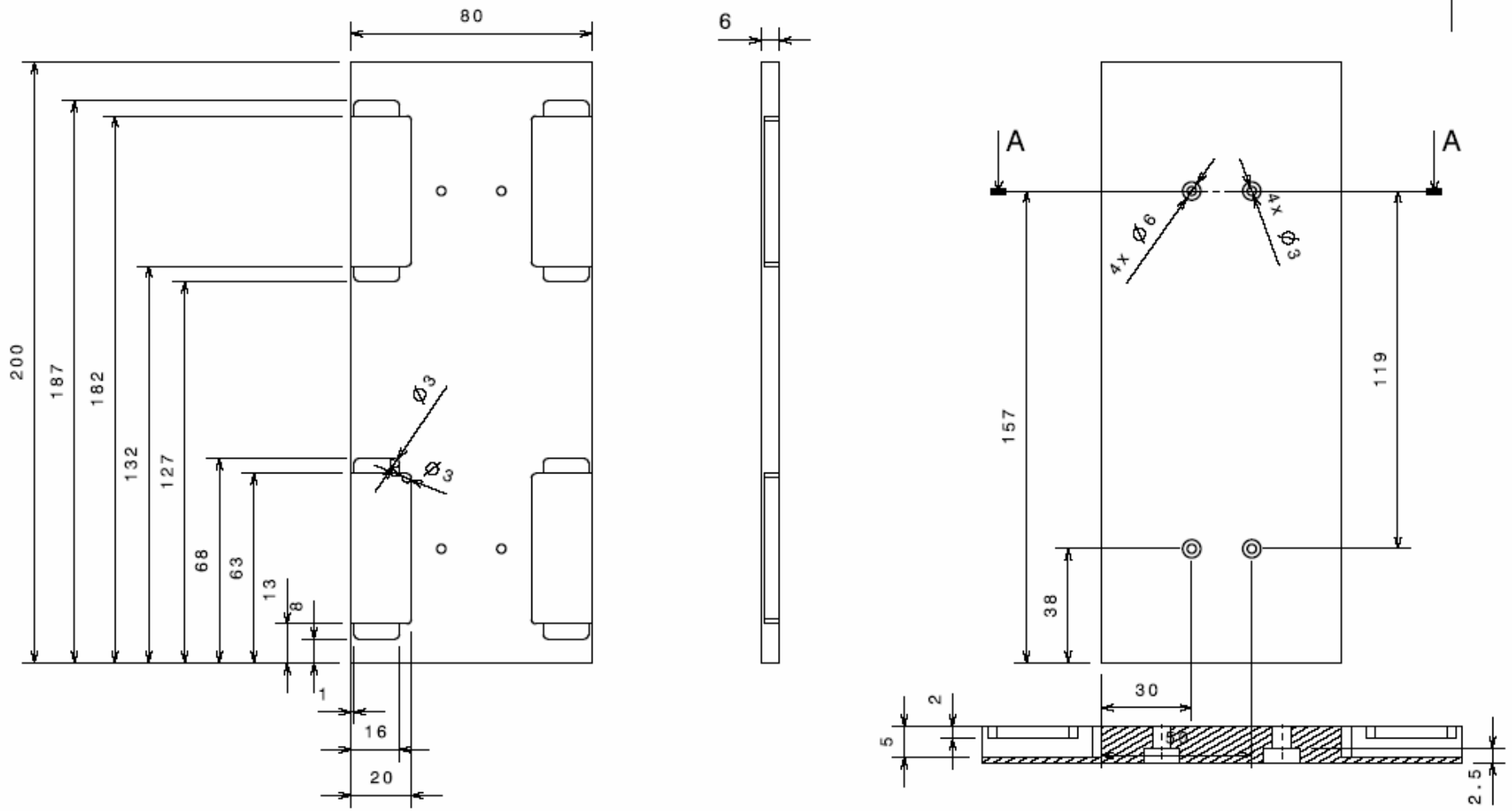


giroscopio	
17-07-2005 11:04:46	
Sheet: 1/1	

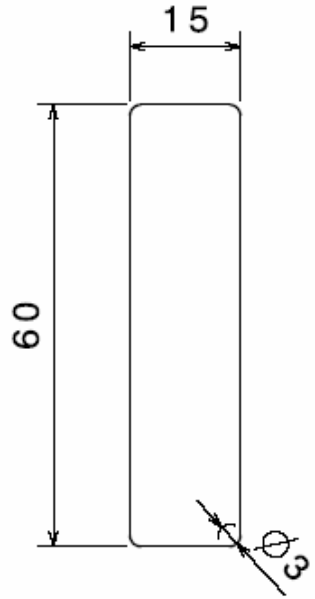

A
B
C
D
E
F
1
2
3
4
V+
J1-1
J1-2
J2-1
J2-2
S1
D1
LED4
R1
0V
CON_PIC_ALIM_2
CON_ALIM1
P$1
P$2
CON_ALIM2
R5
D2
R3
+INA
-INA
1/2 OP200
OUTA
IC1
LED3
R4
J3-1
J3-2
OUTB
-INB
+INB
alimentacao_OP200
17-07-2005 11:14:42
Sheet: 1/1


alimentacao_II
17-07-2005 11:18:16
Sheet: 1/1
J2-1
J2-2
LED4
R1
0V
V+
D1
S1
3
2
1
J1-1
J1-2
P$1
P$2
CON_PIC_ALIM_2
CON_ALIM1
P$1
P$2
CON_PIC_ALIM_2
CON_ALIM2

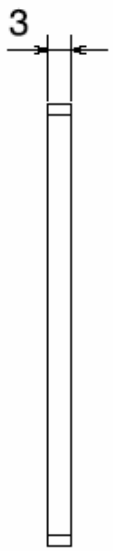

alimentacao_LM741P
17-07-2005 11:21:42
Sheet: 1/1
J2-1
J2-2
LED3
R4
V+
IC2
LM741P
0V
R3
R5
D2
LED4
R1
D1
S1
J1-1
J1-2
P$1
P$2
CON_PIC_ALIM_2



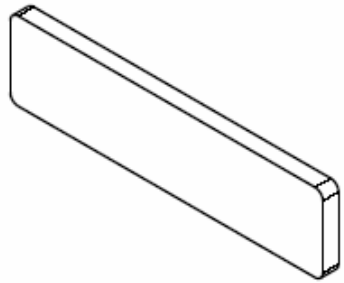
DESIGNED BY: PHUA DATE: 18-07-2005		
SIZE A4		DASSAULT SYSTEMES
SCALE 1:2	WEIGHT (kg) 0,00	DRAWING NAME pé suporte sensor



Right view
Scale: 1:1



Front view
Scale: 1:1



DESIGNED BY: PHUA DATE: 24-05-2005	material-acrilico		
CHECKED BY: DATE:			
SIZE A4	Placa Extensometro		
SCALE 1:1	WEIGHT (kg) 0,00	DRAWING NUMBER	SHEET 1/1

I	_
H	_
G	_
F	_
E	_
D	_
C	_
B	_
A	_