Desenvolvimento da unidade central de controlo para uma plataforma Humanóide

Relatório Final de Projecto

Daniel José Figueiredo Baptista Nº28703

Orientação:

Prof. Dr. Filipe Silva (DETI-IEETA)
Prof. Dr. Vítor Santos (DEM-TEMA)

Universidade de Aveiro
Departamento de Electrónica, Telecomunicações e Informática, IEETA
Licenciatura em Engenharia Electrónica e Telecomunicações

Julho de 2007

Índice

Índice de Figuras

Índice de Tab1elas

1. Introdução

Resumo:

Este projecto visa o desenvolvimento de um robô humanóide de baixo custo. A grande ambição é a participação no concurso *RoboCup*, na modalidade *The Penalty Kick.* Futuramente, irão ser construídos mais dois humanóides para entrar na classe de *Team kid*.

Neste trabalho pretende-se estudar os vários sensores do humanóide, sendo necessário verificar os vários comportamentos destes e implementá-los no robô humanóide. Adicionalmente, é necessário projectar uma unidade central de processamento, para processar imagens vídeo e controlar um robô humanóide de baixo custo. Pretende-se ainda projectar as câmaras de visão, com os algoritmos de processamento de imagem e os de controlo.

No final deste projecto espera-se que o robô tenha atingido os objectivos propostos.

No século passado, grandes visionários redigiram livros e realizaram filmes de robôs, seres artificiais, capazes de, além de ajudar a desempenhar tarefas, pensarem e aprenderem coisas por si mesmos, interagirem com o ser humano, expressarem emoções, terem uma consciência própria... isto é, possuírem características puramente humanas.

Hoje em dia essa visão passou à realidade, com o grande desenvolvimento da tecnologia. A concepção de um robô humanóide, um ser artificial antropomórfico semelhante ao homem, é possível. Mas, por enquanto, com grandes limitações, pois o ser humano é a forma de vida mais complexa conhecida até hoje, tornando-se um grande desafio para o nosso engenho imitar essa forma de vida. Alguns protótipos já estão muito desenvolvidos como por exemplo o *QRIO* da *Sony*, o *ASIMO* da *Honda* entre outros, estes já caminham, dançam, ou pegam em objectos, e até interagem com o ser humano.

Fig. 1.1 - QRIO da Sony

Fig. 1.2 - ASIMO da Honda

Fig. 1.3 - HOAP-2 da Fujitsu

Neste projecto, o nosso grande desafio é a concepção e desenvolvimento de um robô humanóide. Assim, neste relatório, é possível encontrar o enquadramento do projecto e objectivos do trabalho, o estudo dos diversos sensores e das unidades centrais, o processamento da imagem, bem como as conclusões tiradas após estes estudos, e as referências bibliográficas consultadas na sua elaboração.

1.1. Enquadramento do projecto

Grandes companhias de desenvolvimento de produtos tecnológicos apareceram com protótipos de várias plataformas humanóides com grandes capacidades humanas. Foi este, o principal móbil que levou um grupo do DEM, em colaboração com DETI, da Universidade de Aveiro, a construir o primeiro robô humanóide em Portugal. Os primeiros objectivos aquando da concepção do robô, eram entrar em duas competições: o *RoboCup* e o *FIRA*, duas organizações internacionais que realizam anualmente competições na classe dos robôs humanóides.

A construção do robô foi iniciada em 2003. Actualmente este já tem toda a estrutura externa

construída e alguns algoritmos de controlo aplicados, que permitem ao robô fazer coisas básicas como manter o equilíbrio. A unidade de processamento actualmente utilizada e que se pretende substituir é um PC. Com a construção do robô humanóide já neste ponto, é necessária a remoção dos cabos de ligação ao PC e substituir este por uma unidade central de processamento de pequenas dimensões para que o robô seja totalmente autómato e capaz de desempenhar determinadas tarefas sem intervenção humana.

A Unidade Central de Processamento é responsável pela gestão global dos procedimentos incluindo as seguintes tarefas: cálculo das configurações que as juntas devem adoptar com base em directivas de alto nível, processamento de imagens vídeo, interacção com computador externo para permitir monitorização, *debuging* ou *tele-operação*.

A motivação para a investigação desta unidade, na área da robótica humanóide é encontrada em diversos protótipos do tipo e em novos componentes electrónicos do mercado.

1.2. Objectivos do trabalho

A primeira fase deste trabalho consiste, em estudar os diversos sensores do humanóide sendo estes os extensómetros, os acelerómetros/inclinómetros e o giroscópio. Para os estudos dos diversos sensores foi necessário um estudo prévio dos controladores de baixo nível implementados. Esta fase está dividida em:

1. Estudo da melhor estratégia de controlo aplicada nas juntas de modo a realizar comportamentos relativos à locomoção: estudo do comportamento de um servomotor de modo a analisar qual a melhor estratégia de controlo;
2. Estudo dos comportamentos especiais relativos ao equilíbrio estático e dinâmico: como por exemplo equilibrar uma perna na vertical na variação do plano de suporte;
3. Estudo dos sensores dos pés (extensómetros) para o equilibrar numa perna na vertical na variação do plano de suporte;
4. Estudo do comportamento em regime estático/dinâmico dos acelerómetros de dois eixos e formas de os combinar para melhorar a sua resposta;
5. Avaliação do potencial de aplicação do acelerómetro como inclinómetro, pretendendo-se estimar em tempo real o ângulo de inclinação do pé de suporte em relação ao plano horizontal isto é, imaginar o robô a mover-se em terrenos irregulares ou a caminhar num plano inclinado;
6. Estudo do giroscópio, análise da resposta do sensor e avaliação do potencial de aplicação no robô humanóide;
7. Anexo 1: Sistema energético do robô;

Na segunda fase deste trabalho, pretende-se estudar o estado da arte, investigar as opções do

mercado de *embedded systems* e escolher a melhor solução para a unidade central de processamento, tendo em consideração o controlo do humanóide, os algoritmos de processamento de imagem e ainda as comunicações entre vários humanóides que irão ser construídos futuramente. Esta tarefa vai ser dividida em:

1. Análise do estado actual de desenvolvimento do sistema e compreensão dos requisitos tecnológicos envolvidos:
2. - Avaliação do desempenho ao nível do controlo, visão, planeamento e percepção, tendo em conta os requisitos físicos e funcionais colocados pela participação no *RoboCup*;
3. Investigação e procura da unidade central de processamento em função dos requisitos tecnológicos para que o humanóide seja totalmente autónomo;
4. Escolha da melhor solução para a unidade central de processamento tendo em conta todos os requisitos pedidos e o seu custo;
5. Compra da unidade central de processamento tendo em conta os vários requisitos propostos;
6. Instalação do sistema operativo mínimo, devido às baixas capacidades da placa;
7. Desenvolvimento da unidade central de processamento para o sistema de visão específico do robô, em que este tem de fazer o *tracking* de um objecto em movimento (por exemplo, uma bola de determinada cor).
8. Numa fase posterior, introduzir os melhoramentos possíveis ao sistema existente.

Por isso, o relatório está organizado em duas partes principais:

- Capítulos 2, 3 e 4 – Estudam a compreensão e implementação dos vários sensores do humanóide.
- Capítulos 5, 6 e 7 – Estudam a unidade central de processamento, escolha, descrição, implementação dos algoritmos de comunicação e de processamento de imagem.

1.3. Descrição da plataforma humanóide

A plataforma humanóide possui um conjunto de 22 graus de liberdade, distribuídos da seguinte forma:

- 2 em cada pé (2x2);
- 1 em cada joelho (1x2);
- 3 em cada anca (3x2);
- 2 no tronco (2x1);
- 3 em cada braço (3x2);
- 2 no suporte da câmara (cabeça) (2x1).

A estrutura é constituída essencialmente por alumínio e aço nos eixos e outros pequenos

componentes, pesando um total de 6 Kg com as baterias incluídas, e medindo cerca de 60cm. Estes valores foram estabelecidos de acordo com as regras impostas pelo *RoboCup*, baseando-se no pressuposto de que para valores superiores a estes, o uso de servomotores de baixo custo poderá tornar-se inviável dada a impossibilidade de conciliar binários de motores e pesos dos equipamentos e acessórios como as baterias.

Por razões de estética e de acomodação de componentes, foi adoptada uma estrutura em forma de exoesqueleto (carapaça) dotando assim o sistema de módulos ocos onde são alojados os motores, sensores, cablagens, placas de controlo, ente outros.

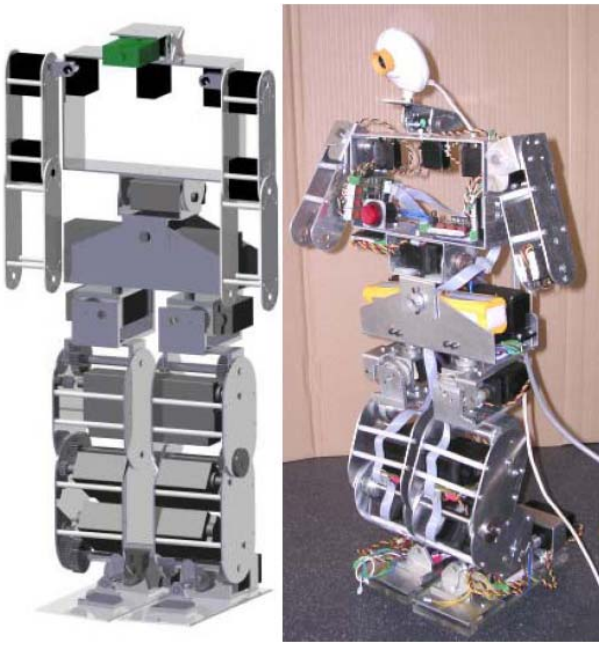


Fig. 1.4 - Modelo 3D do robot e implementação actual.

Preocupações com a autonomia energética do sistema, exigiram a escolha de baterias de elevada capacidade de corrente, até porque é necessário alimentar 22 motores de binário apreciável (embora uns mais do que outros). Tendo em conta esses aspectos, utilizaram-se duas baterias de 7,4V ligadas em paralelo com uma capacidade máxima de 9600mAh.

No que respeita ao controlo, foram assumidas, desde logo, as vantagens de uma arquitectura distribuída e modular baseada num *bus* CAN, responsável por permitir a troca de informação entre as diversas unidades de controlo a partir de uma unidade mestre que faz a gestão da rede e que está directamente ligada a uma unidade primária de decisão que, neste momento, é um computador vulgar. Posteriormente substituir-se-á o computador por uma *embedded motherboard* que será discutida no Capítulo 4 e posteriores, com as mesmas funcionalidades que um PC, no que respeita à comunicação e ao processamento de imagem, com as vantagens de dimensões reduzidas e de baixo custo.

Neste momento, a plataforma é constituída por 9 unidades, uma mestre que gere as informações de actuação ou sensoriais e controlo de tráfego na rede CAN, 8 de controlo local dos actuadores e

sensores. As unidades de controlo local estão distribuídas de forma a agrupar conjuntos de três actuadores relativos a um determinado membro, como é o caso das pernas ou dos braços. Com uma arquitectura deste género, pretende-se que o hardware que constitui estes módulos seja idêntico e também um software idêntico para cada um deles. 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*

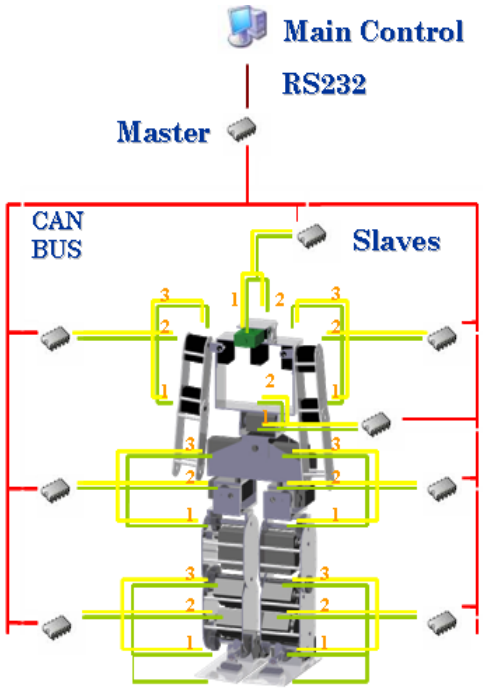


Fig. 1.5 - Arquitectura distribuída da plataforma.

Na percepção destacam-se os sensores proprioceptivos, alguns inérciais e visuais:

1. Potenciómetros de medição da posição das juntas;
2. Sensores de força, para medição das forças de reacção dos pés;
3. Acelerómetros/Inclinómetros, para medição da aceleração gravítica
4. Giroscópios para a medição da velocidade angular
5. Câmara para o processamento de imagem

Os sensores do tipo 2 e 3 serão designados por sensores especiais, dada a sua natureza específica, e podem ser ligados um conjunto máximo de quatro sensores a cada unidade de

controlo local. Tal compromisso deve-se a questões de organização da informação trocada entre as diversas unidades de controlo.

2. Sensores de força

Resumo:

Neste capitulo é feito um estudo aos sensores de força, pois estes são fundamentais para conhecer as forças aplicadas ao solo e conhecer as forças de reacção do solo aplicadas à plataforma. Sendo isto, essencial para o equilíbrio do robô humanóide. No caso da marcha, sempre que uma perna não necessite de realizar controlo na locomoção, como é o caso da perna de suporte, é importante manter o equilíbrio da plataforma, de modo a que o centro de massa projectado sobre o solo se mantenha sempre centrado sobre o pé de suporte.

2.1. Introdução

A principal função dos sensores de força é medir as forças de reacção do solo no pé da plataforma humanóide e medir a força gravítica desta plataforma sobre o solo. Conseguindo medir as forças aplicadas directamente no pé do humanóide, pode-se estimar o equilíbrio do robô.

No estudo da marcha da plataforma humanóide, apenas uma das pernas possui a tarefa de realização da trajectória correspondente à execução de um meio passo, sendo que a outra fica responsável pela manutenção do equilíbrio. Em seguida, as tarefas são trocadas, ou seja, a perna de suporte passa a ser a perna a executar a trajectória e a outra passa a ser a perna de equilíbrio da estrutura, de modo a que esta mantenha a sua posição vertical. Note-se, que esta estratégia, é usada pelo ser humano, com a percepção das forças aplicadas nos pés e pelo nível de tensão aplicado nos músculos, procurando assim, o equilíbrio do corpo. Com a adaptação dos tornozelos, a flexão dos joelhos e a inclinação do tronco é possível encontrar o ponto de equilíbrio mesmo que este seja perdido por algum momento.

Dado que, no caso do robô humanóide, o seu centro de massa localiza-se na região da cintura, logo este é simétrico e o maior peso está sobre a cintura e as pernas sendo o tronco mais leve até esta altura.

Posteriormente, a estrutura do tronco vai ter que ser refeita e calculado o seu novo centro de massa, uma vez que a unidade de processamento central vai ser lá colocada e o seu peso vai influenciar o centro de massa da estrutura. Nesta fase, o movimento do tronco poderá ser dispensado por este ser simétrico bastando mover as juntas das pernas.

O processo de controlo, pensado e implementado, baseia-se na medição de vários sensores de força localizados nas extremidades da base de cada pé, que medem as forças de reacção do solo e gravítica aplicadas sobre eles. Ao fazer a leitura dos sensores pode-se estimar o ponto onde se localiza o centro de pressão que, para baixas velocidades, coincidirá com a projecção do centro de massa da estrutura sobre os pés. Ora, sabe-se que, na situação de equilíbrio, o centro de pressão deve coincidir com a posição central de cada pé ou muito perto desta, pelo que o controlador de equilíbrio deve utilizar esta posição como ponto de referência a atingir.

2.2. Medição das forças de reacção

2.2.1. Versão anterior do circuito de acondicionamento de sinal

O circuito de condicionamento de sinal encontrado[1], tinha a particularidade de melhorar o

[1] Referência bibliográfica 9

procedimento de calibração dos extensómetros. A introdução do potenciómetro em paralelo com uma resistência, tem um efeito que provoca pouca variação da resistência, mas com elevada resolução.

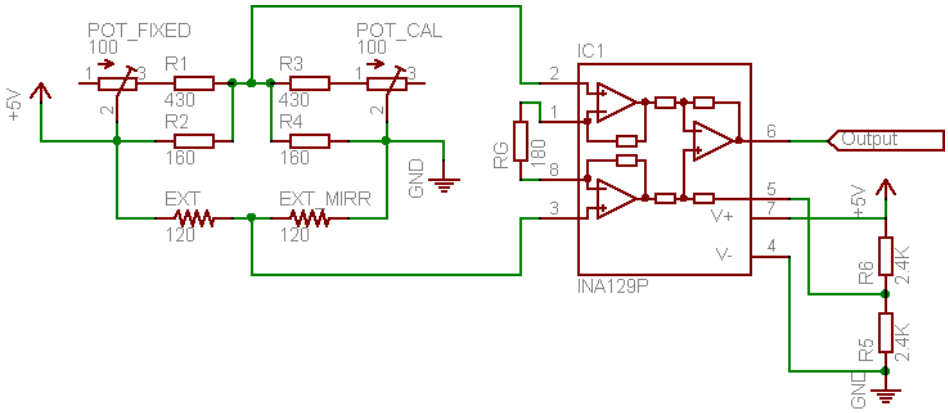


Fig. 2.1 - Ponte de Wheatstone com compensação de interferências externas.

O circuito da Fig. 2.1 encontrava-se montado na placa banca (Fig. 2.2), por sua vez circuito estava conectado aos sensores da plataforma e ligado a um microcontrolador também implementado uma placa branca.

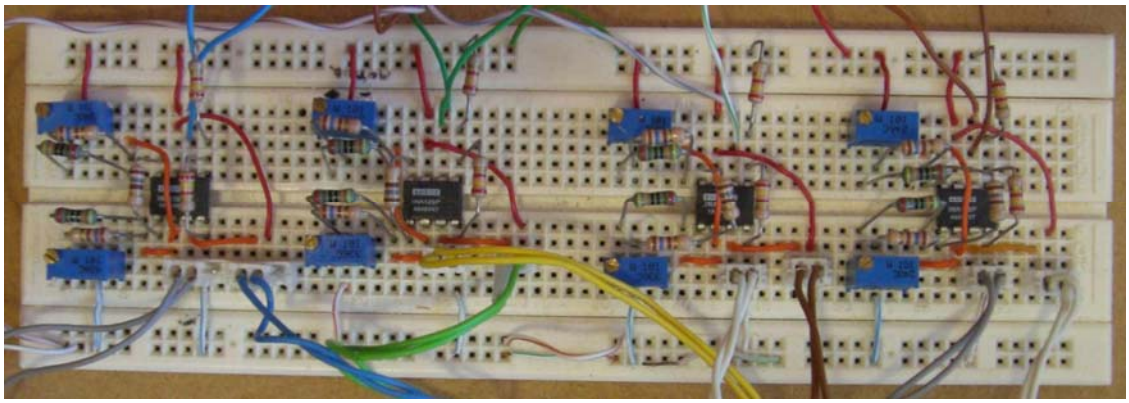


Fig. 2.2 - Montagem da ponte *Wheatstone* e acondicionamento de sinal

2.2.2. Versão actual do circuito de acondicionamento de sinal

Verificou-se que o circuito da Fig. 2.1, podia ser melhorado, para que a ponte ficasse mais balanceada e com um menor consumo de corrente. Uma nova configuração, em que apenas se muda a localização dos componentes, torna o circuito mais robusto. Esta nova configuração (Fig. 2.3), permite limitar a corrente consumida de modo a poupar energia das baterias e a limitar a dissipação por efeito de Joule nas resistências da ponte.

Segundo a configuração da Fig. 2.1 a corrente consumida é de cerca de 42mA por cada circuito, o que para um conjunto de 8 sensores (para dois pés) teríamos um consumo de 333mA, sendo este

um consumo acentuado só para os extensómetros.

Para o novo circuito (Fig. 2.3) é possível limitar a corrente em ambos os ramos da ponte de *Wheatstone*, incluindo a corrente consumida por cada extensómetro.

Para a redução da corrente em ambos os ramos é suficiente aumentar a resistência R_{CAL} e R_{FIXED} na mesma proporção:

$$I_{EXT} = \frac{V_{DD}}{EXT_{MIRR} + R_{CAL}} + \frac{V_{DD}}{EXT + R_{FIXED}} = 2 \cdot \frac{V_{DD}}{EXT + R_{FIXED}}$$

Note-se, contudo, que com este procedimento, a excursão do sinal de saída da ponte diminui, exigindo o aumento do ganho do amplificador de instrumentação:

$$v_3 - v_2 = \frac{R_{FIXED}}{R_{FIXED} + EXT} \cdot V_{DD} - \frac{R_{CAL}}{R_{CAL} + EXT_{MIRR}} \cdot V_{DD} \Leftrightarrow \frac{\Delta v}{V_{DD}} = \frac{1}{1 + \frac{EXT}{R_{FIXED}}} - \frac{1}{1 + \frac{EXT_{MIRR}}{R_{CAL}}}$$

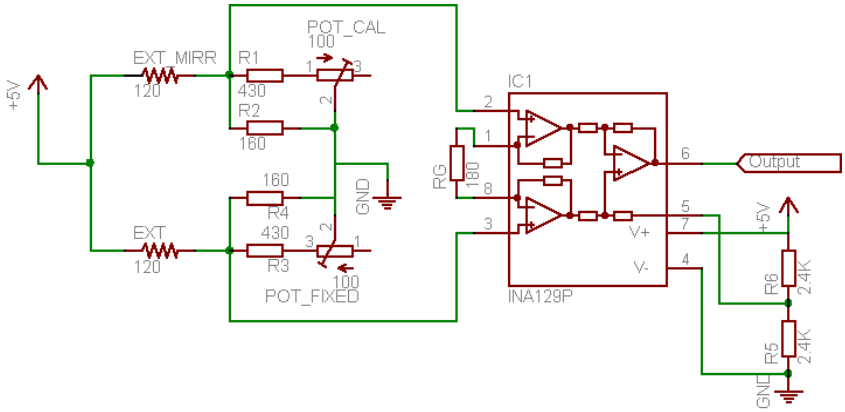


Fig. 2.3 - Circuito de acondicionamento de sinal com simetria completa.

O circuito da Fig. 2.3, foi implementado uma placa branca, mas este circuito tinha problemas de contactos entre os extensómetros e o circuito de acondicionamento de sinal. Mesmo neste último existiam maus contactos quando se tentava ajustar os potenciómetros para balancear a ponte.

A placa branca tem o problema de os contactos serem feitos por duas patinhas simétricas, que seguram os fios que se introduzem (Fig. 2.4); os contactos só são feitos em dois pontos muito pequenos, e ao mexer os condutores estes vão introduzir muito ruído no circuito. Esse ruído posteriormente vai ser amplificado pelo amplificador de instrumentação, e destruirá completamente os sinais provenientes da ponte de *Wheatstone*.

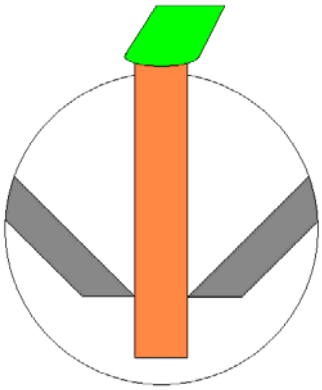


Fig. 2.4 - Contactos na placa branca

Tal como aconselhado pelos orientadores do projecto, a melhor solução para resolver este problema foi implementar os quatro circuitos de acondicionamento de sinal numa placa perfurada e soldar todos os componentes do circuito, para que estes não se movam e não tenham maus contactos. Com esta solução os problemas relacionados com a placa branca foram superados.

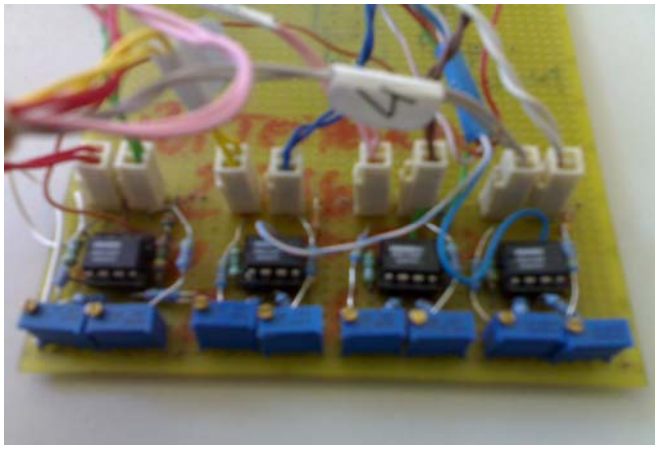


Fig. 2.5 - Circuito de acondicionamento de sinal em placa perfurada

Porém, esta não vai ser a solução definitiva para o circuito de acondicionamento de sinal dos extensómetros. A placa de acondicionamento de sinal para ser implementado na plataforma humanóide tem que ser mais compacta, robusta e funcional. Tudo isto implica a implementação com componentes SMD (surface mount devices), com placas de face dupla, com pistas muito estreitas, planos de massa para proteger os sinais contra o ruído e fabricada em locais especializados. Adicionalmente, esta vai conter o barramento piggy-back para ser montada mas placas slaves, de modo que os sinais sejam amostrados pelo microcontrolador. O desenho da placa foi feito em Cadence Orcad 10.5 (ver pequeno tutorial no final deste relatório). Pode-se ver o resultado final nas Fig. 2.6 e Fig. 2.7 do circuito desenhado pelo Cadence Orcad 10.5 e a aparência final montado na slave na Fig. 2.8.

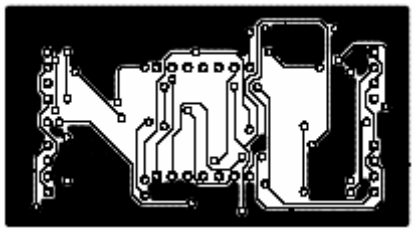


Fig. 2.6 - PCB dos sensores de força - Booton

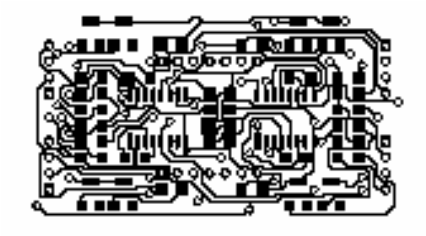


Fig. 2.7 - PCB dos sensores de força -Top

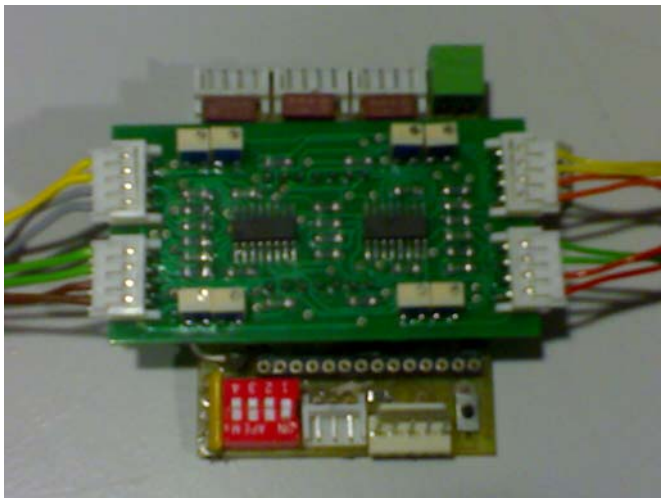


Fig. 2.8 - Placa Slave com sensores de pressão.

2.3. Resultados experimentais

2.3.1. Setup experimental

Para o teste dos extensómetros utilizou-se a estrutura do pé indicada na Fig. 2.9 com os seus quatro sensores ligados à placa de acondicionamento de sinal (Fig. 2.5) usando o circuito da Fig. 2.3. Com o multímetro e com o osciloscópio foi feita a leitura de cada sensor. A experiência foi realizada do seguinte modo: colocando vários pesos, previamente medidos, sobre cada sensor testou-se cada sensor de forma independente.

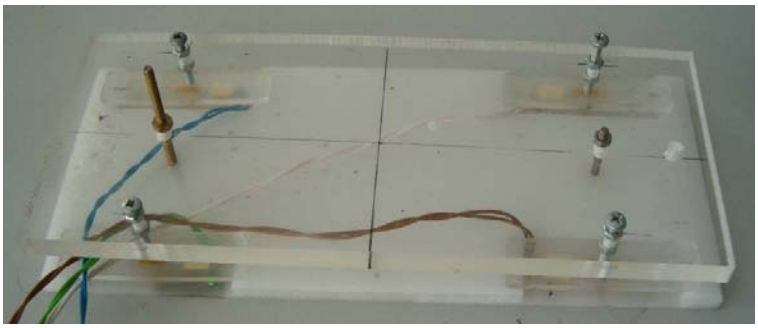


Fig. 2.9 - Estrutura de testes do pé

A lista de pesos medidos é indicada a seguir, na Tabela 2.1. Os pesos 1 a 4 correspondem a cilindros de aço. Usaram-se estes pesos de forma isolada e combinados uns com os outros de modo a obter uma massa máxima de cerca de 2,0Kg.

Carga	*Massa (g)*	*Peso (N)*
1	247	2,4206
2	454	4,4492
3	713	6,9874
4	664	6,5072
1+2	701	6,8698
1+3	960	9,4080
1+4	911	8,9278
2+3	1167	11,4366
2+4	1118	10,9564
3+4	1377	13,4946
1+2+3	1414	13,8572
1+2+4	1365	13,3770
1+3+4	1624	15,9152
2+3+4	1831	17,9438
1+2+3+4	2078	20,3644

Tabela 2.1 - Lista de pesos usados no teste dos sensores de força.

Fig. 2.10 - Teste do extensómetro

Fig. 2.11 - Localização dos sensores no pé.

Os valores registados na experiência, para cada sensor, estão indicados na Tabela 2.2, com o vermelho e o amarelo localizados na parte da frente do pé e o cinzento e rosa na parte traseira do pé (Fig. 2.11).

2.3.2. Medidas experimentais dos sensores aplicados em acrílico

Para esta experiência todos os sensores foram calibrados com o valor 1,00V na ausência de carga, com apenas estrutura de teste a pressionar os sensores.

	Pesos															
Sensores	**0**	**1**	**2**	**3**	**4**	**1+2**	**1+3**	**1+4**	**2+3**	**2+4**	**3+4**	**1+2+3**	**1+2+4**	**1+3+4**	**2+3+4**	**1+2+3+4**
Vermelho	1,00	1,10	1,18	1,30	1,29	1,29	1,43	1,42	1,52	1,51	1,62	1,64	1,58	1,75	1,89	2,13
Amarelo	1,00	1,14	1,27	1,44	1,38	1,43	1,60	1,51	1,71	1,69	1,83	1,81	1,89	2,01	2,21	2,34
Cinzento	1,00	1,10	1,21	1,36	1,35	1,35	1,48	1,47	1,61	1,60	1,73	1,69	1,72	1,93	2,04	2,25
Rosa	1,00	1,14	1,24	1,39	1,36	1,34	1,45	1,41	1,51	1,49	1,64	1,67	1,60	1,71	1,75	1,88

Tabela 2.2 - Resultados experimentais para cada sensor.

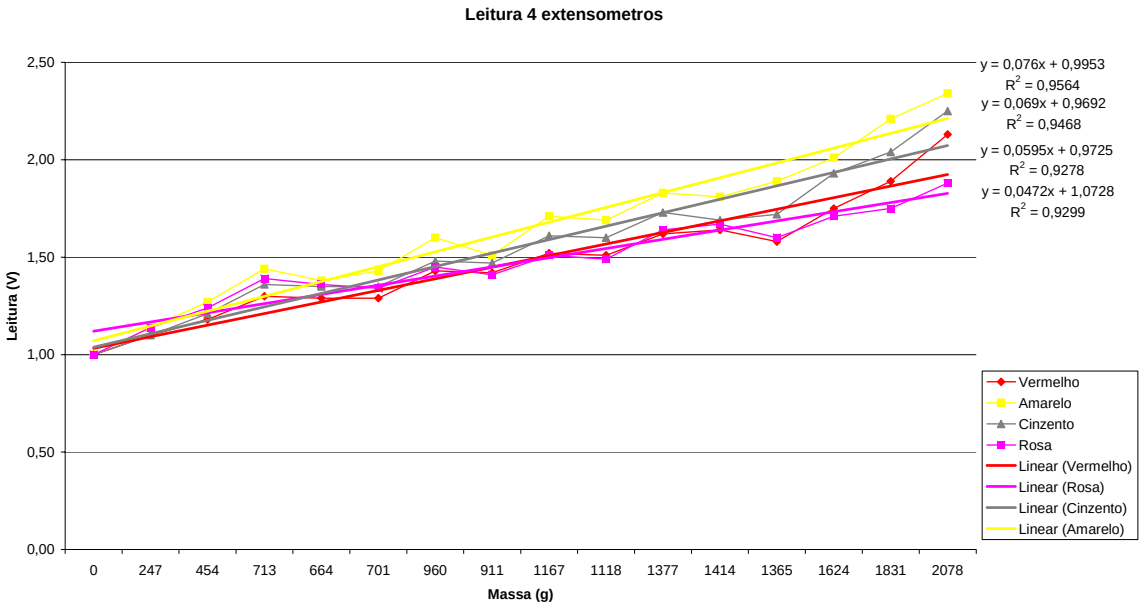


Fig. 2.12 - Leituras dos extensómetros para as várias massas.

Parâmetro da regressão y = m×x + b	Sensor Vermenho	Sensor Amarelo	Sensor Cinzento	Sensor Rosa
Declive m	0.0595	0.076	0.069	0.0472
Ordenada na origem b	0.9725	0.9953	0.9692	1.0728
Coeficiente de correlação R^2	0.9278	0.9564	0.9468	0.9299

Tabela 2.3 - Parâmetros das regressões lineares traçadas.

Como se pode constatar, a Fig. 2.12 representa graficamente os dados registados para cada sensor testado isoladamente, em que se confere a relação entre a força gravítica aplicada sobre cada extensómetro e a deformação medida. A relação entre a força gravítica e a leitura dos extensómetros é aproximadamente linear. As rectas traçadas correspondem à regressão linear dos valores retirados da experiência. Como se pode constatar, existe dispersão de valores, daí conclui-se que estes não são totalmente lineares. Note-se, que na Tabela 2.3 os coeficientes de correlação vão variando de sensor para sensor, o que reflecte um desvio entre sensores. Quando os quatro sensores estão em conjunto a calcular o centro de pressão este pode tender para o sensor mais sensível, o que traz grande inconvenientes no controlo de equilíbrio.

Para cargas superiores a 20N, verifica-se um desvio mais acentuado destes, pelo que é totalmente aconselhado a não utilizar cargas superiores a 20N pois o comportamento destes não é linear.

2.3.3. Medidas experimentais dos sensores aplicados em aço

Para os testes dos extensómetros colocados em laminas de aço, foi utilizado o mesmo

procedimento descrito no ponto anterior.

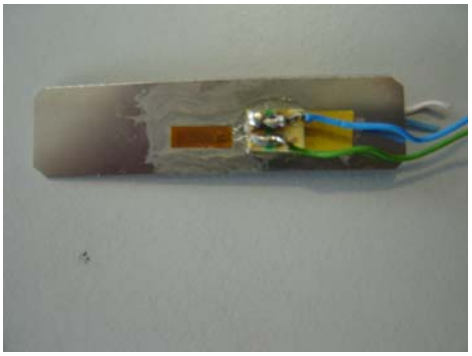


Fig. 2.13 - Sensor aplicado numa capa de aço deformável.

Neste caso, foi feito um primeiro teste à variação de resistência do sensor com a deformação deste.

Peso	0	1	2	3	4	1+2	1+3	1+4	2+3	2+4	3+4	1+2+3	1+2+4	1+3+4	2+3+4	1+2+3+4
Aço	119,6	119,7	119,8	119,8	119,8	119,8	119,9	119,8	119,9	119,9	120	120	119,9	120	120	120,1

Tabela 2.4 - Resultados experimentais para o sensor.

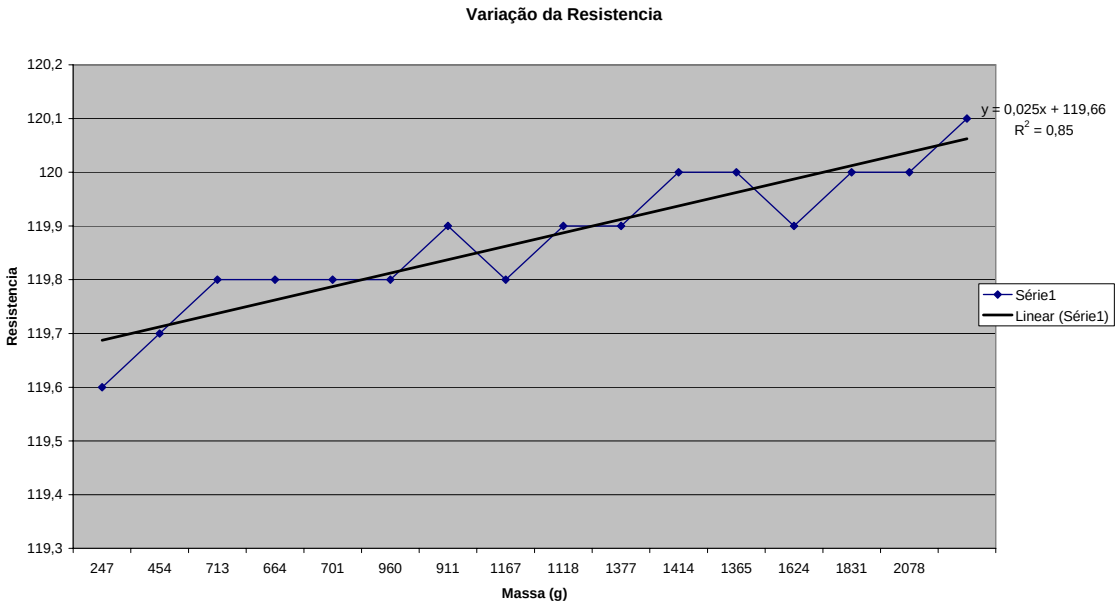


Fig. 2.14 - Leitura dos extensómetro para as várias massas.

A Fig. 2.14 representa graficamente, a variação da resistência para o sensor com a massa aplicada. Como se constata pelos dados retirados, a deformação do aço não é tão linear como a do acrílico, pois existem valores mais dispersos nesta experiência do que no caso anterior, o que confirma o coeficiente de correlação. Contudo, estes valores podem ser equívocos, pois, a precisão do multímetro para ler as resistências é apenas de 0.05Ω, o que pode introduzir erros na leitura do extensómetro.

Como na secção anterior, para esta experiência, o sensor foi calibrado inicialmente com o valor

de 1,00 V na ausência de carga, mas agora com dois sensores um em cima da chapa de aço e um por baixo, um em distensão e o outro em contracção, como mostra a Fig. 2.15.

Fig. 2.15 - Dois sensores aplicado numa capa de aço deformável.

Peso	0	1	2	3	4	1+2	1+3	1+4	2+3	2+4	3+4	1+2+3	1+2+4	1+3+4	2+3+4	1+2+3+4
Aço	1	1,12	1,21	1,34	1,29	1,32	1,42	1,41	1,49	1,49	1,55	1,57	1,58	1,6	1,68	1,73

Tabela 2.5 - Resultados experimentais para o sensor.

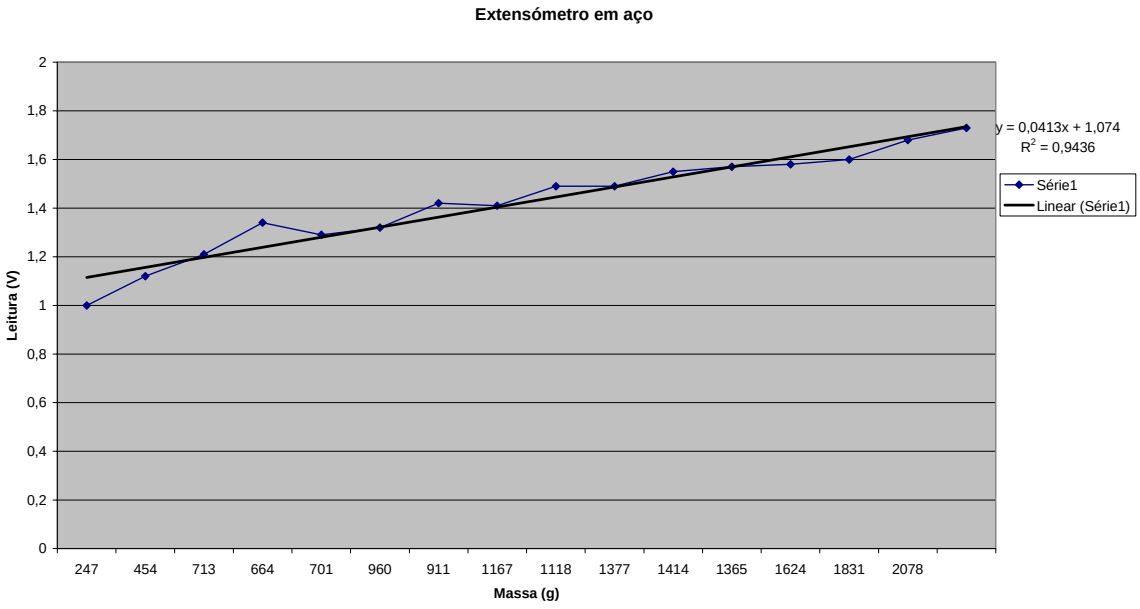


Fig. 2.16 - Leitura dos extensómetro para as várias massas

Como podemos analisar pela Fig. 2.16 e pela Tabela 2.5, com a utilização de dois extensómetros um em deformação e o outro em contracção, verifica-se que os sensores são pouco sensíveis a variações com a massa aplicada. Quanto à linearidade, os resultados foram muito semelhantes ao caso em que os extensómetros estavam aplicados numa lâmina de acrílico. Para tentar resolver a questão da pouca deformação da chapa de aço, foi feita uma procura nas oficinas do departamento de mecânica, mas não foi encontrada uma chapa de aço que satisfizesse os requisitos desta aplicação, pois ou eram muito espessas e tinham pouca deformação ou eram tão

finas que dobravam. Em discussão com os orientadores e com recursos humanos ligados ao projecto, foi enunciada que a melhor solução para este caso era o uso de "aços elásticos", visto que estes se deformam e voltam sempre à sua posição original.

Adicionalmente, seria necessária uma alteração da estrutura dos pés: os pontos de contacto entre a estrutura superior e as placas deformadoras teriam de ser alterados e ainda seria necessário comprar novos extensómetros para ser possível a implementação destas chapas de aço. Por fim, com todos estes problemas, esta ideia foi posta de parte. Espera-se que, quando for encontrada a chapa de aço ideal, seja possível esta implementação, visto que o acrílico é um polímero (e encontra-se sempre no estado líquido) e ao fim de algum tempo este vai sofrendo fadiga e os sensores de força vão variando ao logo do tempo.

3. Sensores inérciais: Acelerómetros / Inclinómetros

Resumo:

Neste capítulo é feito um estudo aos acelerómetros/ inclinómetros e giroscópio. Os acelerómetros/inclinómetros são fundamentais ao equilíbrio do robô, dado que é necessário manter a verticalidade do tronco em várias tarefas a realizar pela plataforma humanóide. O giroscópio é necessário ao equilíbrio da plataforma humanóide, quando esta se encontrar em movimentos dinâmicos. Também quando o robô estiver em movimentos rotativos é necessário ter uma estimativa da velocidade angular, dado que a força centrípeta ou centrífuga pode provocar desequilíbrios na plataforma.

3.1. Introdução

A principal função dos acelerómetros é medir a aceleração de uma dada massa quando esta se encontra em movimento. Quando a massa está sobre o efeito da aceleração gravítica, então podemos medir a aceleração gravítica imposta na massa. Deste modo, podemos medir a inclinação de um plano quando este vai variando a sua perpendicularidade em relação à força gravítica. Assim sendo, é possível utilizar um acelerómetro como um inclinómetro, isto é, medindo a aceleração da gravidade segundo o eixo do acelerómetro pode-se estimar inclinações. O acelerómetro pode medir acelerações dinâmicas (aceleração de movimentos muito rápidos) ou medições estáticas, (medição da aceleração da gravidade dado que a sua aceleração é sempre constante).

Na comparação com o ser humano, os acelerómetros têm a mesma função que os ouvidos, pois estes são os principais responsáveis pelo equilíbrio do tronco. A mesma função vai ter o acelerómetro: este vai estimar a inclinação do tronco em relação à gravidade.

Quando a plataforma humanóide se encontra em marcha, o seu tronco tem que se encontrar na posição vertical, sendo o acelerómetro responsável pela indicação da verticalidade ou quando a plataforma cair, o acelerómetro vai dar a indicação que a plataforma se encontra no solo.

No caso desta plataforma humanóide, o seu centro de massa localiza-se perto na região da cintura, é neste ponto que deve situar-se o acelerómetro ou inclinómetro, sendo neste sitio o principal ponto de equilibro da plataforma. Enquanto o centro de massa estiver em equilíbrio automaticamente toda a plataforma se encontrará em equilíbrio.

3.2. Acelerómetros: princípios e modo de funcionamento

A função do acelerómetro é medir a aceleração de um movimento de um determinado objecto. Existem muitos e variados tipos de acelerómetros, mas para o caso da plataforma humanóide, são utilizados acelerómetros de muito pequena dimensão, em que a massa destes é chamada por massa sísmica. Para acelerómetros de reduzida dimensão, a massa sísmica é tão pequena que uma pequena vibração facilmente é detectada, por mais atenuada que seja. Pela segunda Lei de Newton em que $F=ma$, quanto menor a massa, menor é a força para ter a mesma aceleração, logo, para os acelerómetros electrónicos, como a massa sísmica é tão reduzida, pois estes detectam as mais pequenas acelerações aplicadas.

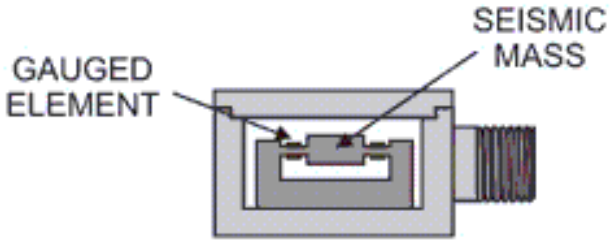


Fig. 3.1 - Interior do acelerómetro.

Para a conversão da aceleração em sinais eléctricos são usados extensómetros de reduzida dimensão, menores que a massa sísmica. O suporte da massa sísmica é constituído por um material deformador, pois a massa sísmica tem que ficar relativamente livre para sofrer acelerações (Fig. 3.1). No suporte deformador são assentes dois extensómetros em eixos perpendiculares, que medem a deformação do suporte. Deste modo, quando existe uma aceleração sobre a massa sísmica, esta vai sofrer um deslocamento e os suportes da massa sísmica são ser deformados segundo o eixo da força sofrida, e assim, é possível medir a aceleração da massa sísmica, em dois eixos.

Para o acelerómetro funcionar como inclinómetro, os seus dois eixos têm que ficar perpendiculares com a aceleração da gravidade, isto é, estes estão a sofrer a máxima aceleração. Para a posição horizontal, a aceleração gravítica é máxima, porque os seus eixos não estão alinhados com a aceleração gravítica (estes estão perpendiculares à gravidade), assim, define-se que para esta aceleração máxima o ângulo de inclinação é zero. Para as medidas de acelerações gravíticas mínimas (+1*g* ou -1*g*) os eixos do acelerómetro já se encontram alinhados com a aceleração gravítica, o que implica a não deformação dos extensómetros. Dado que estes estão paralelos com a gravidade logo, a aceleração medida é mínima, deste modo, define-se que inclinação é máxima num ou no outro sentido.

3.3. Medição das inclinações do plano

3.3.1. Interface entre o acelerómetro e o microcontrolador

Como foi descrito anteriormente, com um acelerómetro é possível medir o plano de inclinação deste, em relação à aceleração gravítica. Para o caso do hardware existente nas placas *slaves* é necessário amostrar os sinais provenientes do acelerómetro através do microcontrolador, para que se obtenha o valor dos sensores e se possa equilibrar o tronco da plataforma humanóide.

3.3.2. Acelerómetro ADXL202E da *Analog Devices*

O ADXL202E é um acelerómetro de dois eixos (X e Y), com uma escala de medida de ±*2 g*, isto é, pode medir no máximo até duas vezes a aceleração da gravidade nos dois sentidos.

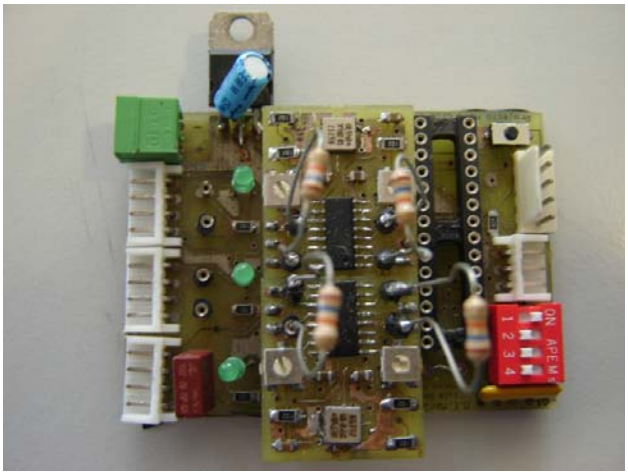


Fig. 3.2 - Acelerómetro ADXL202E da *Analog Devices*

O acelerómetro ADXL202E, permite a medição de acelerações dinâmicas (impactos, vibrações, acelerações instantâneas), e ainda, acelerações estáticas (aceleração da gravidade sendo esta constante no plano perpendicular à terra). Como foi descrito anteriormente, a principal função do acelerómetro é funcionar como inclinómetro, medindo a aceleração da gravidade segundo os seus dois eixos. Para usar os acelerómetros de modo a medir a inclinação do seu plano em relação à gravidade, é necessário fazer a conversão pelas fórmulas:

$$Pitch = ASIN\,(Ax/1g)$$
$$Roll = ASIN\,(Ay/1g)$$

Onde Ax e Ay são as leituras de aceleração nos dois eixos do sensor.

Quando os eixos dos acelerómetros estão orientados segundo a aceleração da gravidade, isto é, quando as leituras estão próximos de +1*g* ou -1*g* a variação da aceleração medida na saída por degrau é muito pequena. Quando o acelerómetro está perpendicular com a gravidade, a saída medida é próxima de 17.5 mg por degrau, mas quando este está a ±45 degraus a variação é apenas de 12.2 mg por degrau, e vai decaindo até chegar aos ±90 degraus (±1g). Dadas estas características, as fórmulas apresentadas anteriormente são perfeitamente justificadas para retirar a verdadeira inclinação do plano, dado que estas compensam as não lineares do acelerómetro.

3.3.3. Circuito de acondicionamento de sinal

O acelerómetro ADXL202E tem dois tipos de saídas, uma digital em *Duty Cycle Modulated* (DCM), e uma saída analógica. A saída DCM (nos pinos, X_{OUT}, Y_{OUT}), em que variação do *duty*

cycle e é proporcional à aceleração. O período de DCM é ajustável entre 0.5 e 10 ms usando para o efeito uma resistência (R_{SET}). A largura do impulso do DCM é ajustável com condensadores C_X e C_Y nos pinos X_{FILT}, Y_{FILT}. As saídas analógicas também são nos pinos X_{FILT}, Y_{FILT}, têm uma variação de aproximadamente 0 a 5V e uma impedância de 32KΩ. Para o uso das saídas analógicas, é aconselhável o uso de um amplificador para o aumento da impedância de saída, isto, para que a leitura feita pela ADC do microcontrolador não tenha efeitos de carga, e ainda, para o melhor ajuste da amplitude de onda proveniente dos acelerómetros.

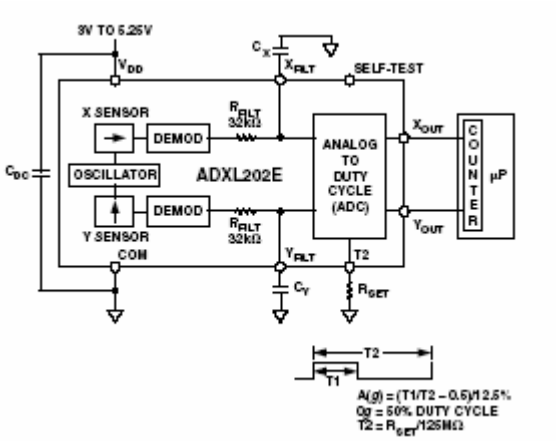


Fig. 3.3 - Diagrama funcional do ADXL202E

Dadas as características das *slaves*, em que estas só estão preparadas para receber sinais analógicos provenientes dos sensores, foi optado á dois anos[2], pelas saídas analógicas (X_{FILT}, Y_{FILT}).

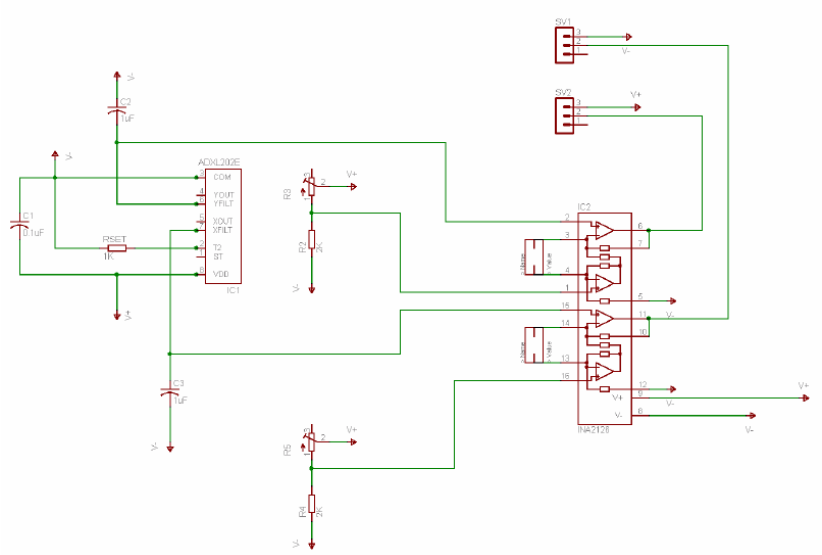


Fig. 3.4 - Circuito de acondicionamento de sinal para o ADLX202E

Verifica-se que o circuito de acondicionamento de sinal, tem um problema de amplificar sempre

[2] Referencia Bibliográfica 14

o sinal proveniente dos acelerómetros, dado que a fórmula do amplificador de instrumentação é G = 1+ 49.4 KΩ / R_G , e por maior que seja R_G, nunca é possível reduzir o ganho do amplificador para valores inferiores a um. Verifica-se que esta situação é inconveniente, pois quando existem leituras próximas de +1g ou -1g, o sinal dos acelerómetros vem saturado, passando acima dos 5V. Assim, ao fazer a amostragem pela a ADC do microcontrolador, a característica do sinal vem alterada, consequentemente, perde-se informação relevante. Neste circuito, era preferível uma menor amplitude do sinal proveniente do amplificador de instrumentação, mas com toda a sua amplitude, e a amplificação ser feita digitalmente pelo microcontrolador.

Verifica-se pelo circuito da Fig. 3.4 que é possível medir inclinações em dois eixos, e para cada eixo é possível medir no sentido positivo e no negativo. Para ser efectuada a leitura nos dois sentidos, é necessário regular o potenciómetro do circuito da Fig. 3.4, de modo a que o sinal de entrada do amplificador tenha o valor de 2,5V, para a inclinação de 0 graus. Só assim é possível medir nos dois sentidos de um eixo.

3.4. Resultados experimentais

3.4.1. Setup experimental

Para os testes dos acelerómetros, foi utilizada uma *slave* com as placas dos acelerómetros montadas no barramento *piggy-back*, conforme se pode verificar na Fig. 3.6, em que os sinais provenientes do circuito da Fig. 3.4 são amostrados pela ADC do microcontrolador de 20 em 20 ms.

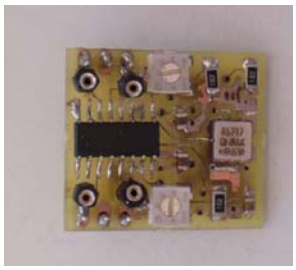


Fig. 3.5 - Placa de acondicionamento de sinal dos acelerómetros

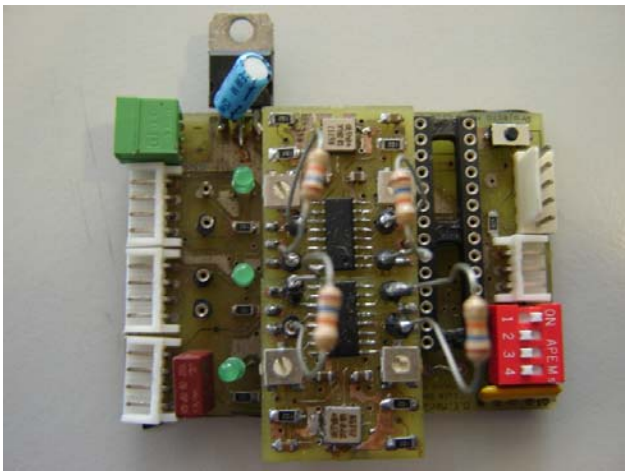


Fig. 3.6 - Slave com os acelerómetros no piggy back

Para variar o plano de inclinação dos acelerómetros, foi utilizado um actuador da plataforma. A *slave* que continha a placa dos acelerómetros foi aparafusada pelo regulador de tensão, que por sua vez, foi aparafusado num braço de um actuador, como se pode verificar pela Fig. 3.7. Actuando uma sequência linear de ângulos no servo, assim, se pode variar o ângulo de inclinação do plano dos acelerómetros. A *slave* de leitura dos acelerómetros é a mesma que controla o actuador, pois ao variar o ângulo do servo, pode ao mesmo tempo amostrar os valores dos acelerómetros, para cada ângulo actuado. Para esta experiência de variação de ângulo dos acelerómetros, foi confiado que o sensor de posição do actuador medisse o ângulo correcto de inclinação, de modo a poder-se comparar com o valor lido dos inclinómetros.

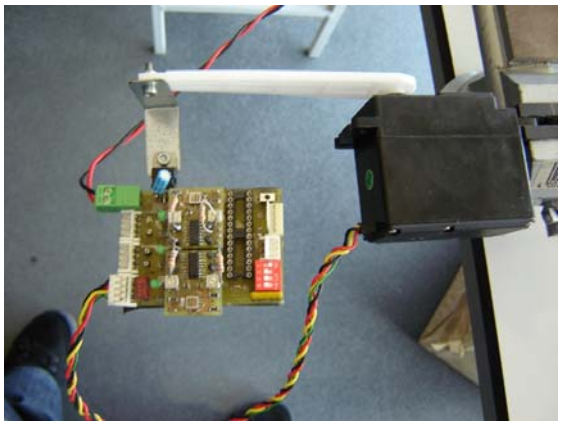


Fig. 3.7 - Teste dos acelerometros (posição horizontal 0º)

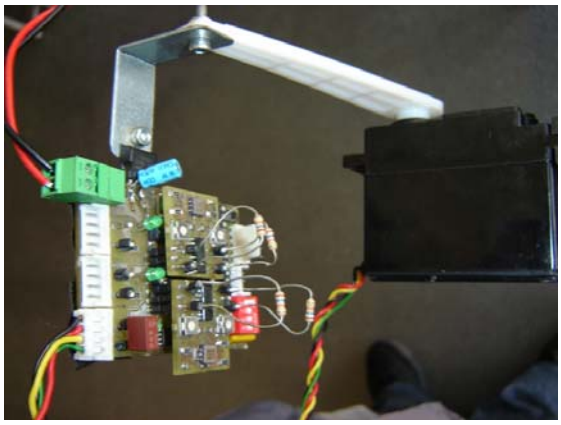


Fig. 3.8 - Teste dos acelerómetros (posição vertical -1g ou +90º)

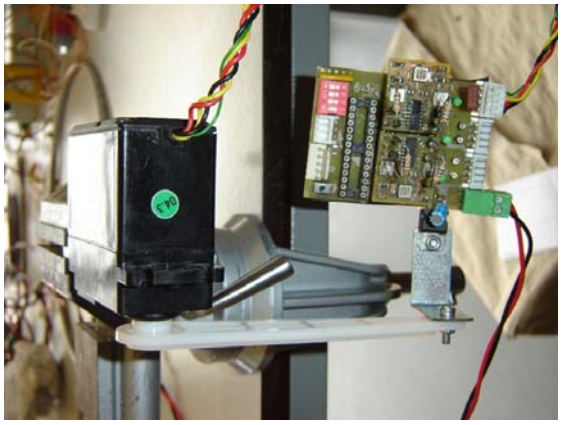


Fig. 3.9 - Teste dos acelerómetros (posição vertical +1g ou -90º)

Os testes realizados foram para dois acelerómetros aplicados na mesma *slave*, mas estes estão colocados de forma a que os eixos dos acelerómetros estejam invertidos, um em relação ao outro tanto em X como em Y. Nesta situação, quando um dos acelerómetro está a medir no sentido positivo o outro no mesmo eixo, está a ler no sentido negativo e vice-versa.

3.4.2. Medidas dos acelerómetros de-90 a 0 graus e de 0 a 90 graus

Como foi descrito anteriormente, com o uso de um actuador de uma *slave* e de dois acelerómetros, foi elaborada uma experiência. Esta consiste em variar o ângulo do braço actuador progressivamente de -90º a 0 e de 0 a +90º. Em seguida amostra-se os acelerómetros para cada grau de inclinação aplicado ao actuador, como pode ser visível na Fig. 3.7 e Fig. 3.9.

Ao aplicar um ângulo no actuador, foi assumido que o sensor de posição do actuador medisse o ângulo correcto de inclinação, de modo a que o servo aplicasse o ângulo correcto ao braço.

Regime estático

Nesta experiência, foi aplicado um ângulo ao actuador e feita uma espera de 0.5s antes da amostragem. Isto para ter a certeza que o movimento do actuador já tivesse terminado, de modo a que as leituras não sofressem a aceleração da actuação.

- Leituras segundo o eixo X

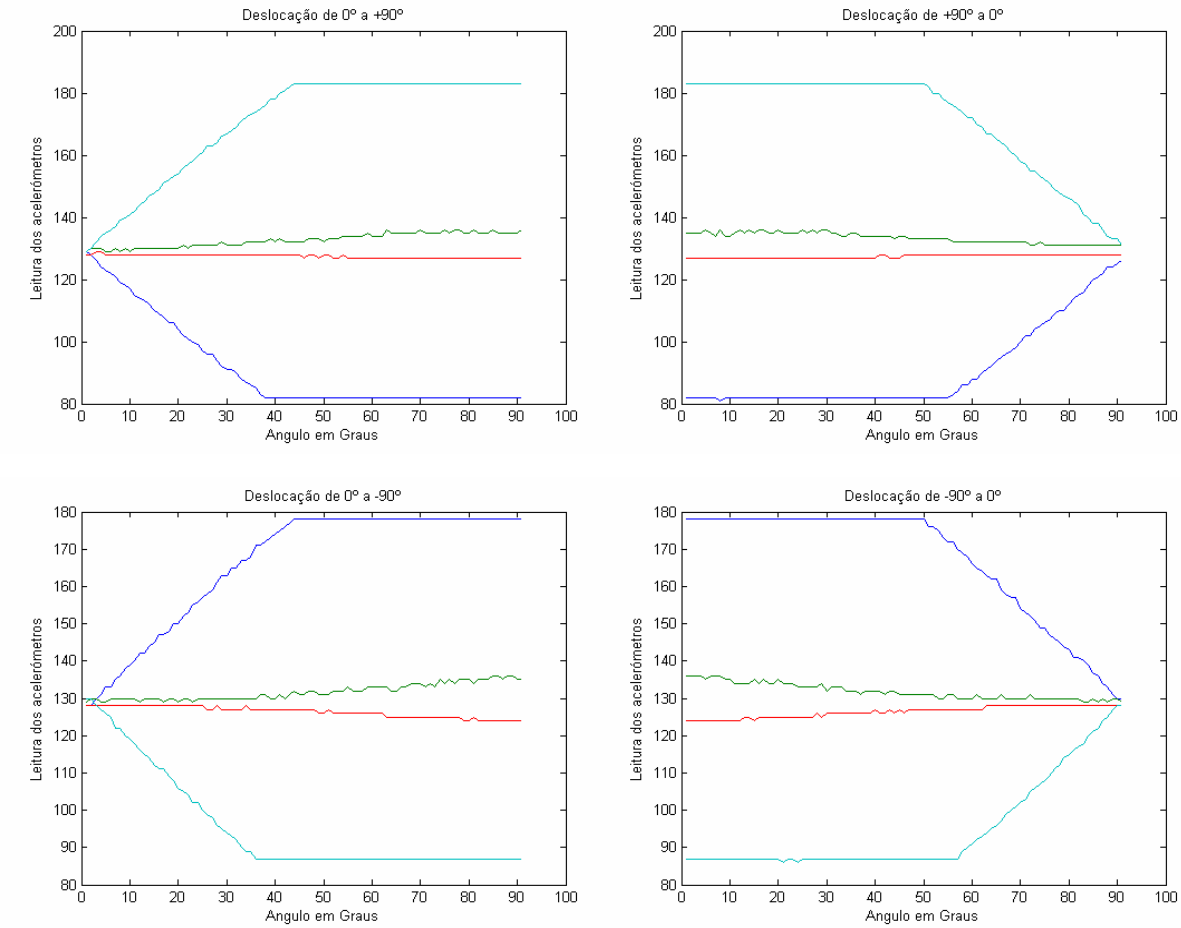


Fig. 3.10 - Aceleração da gravidade no eixo X

- Leituras segundo o eixo Y

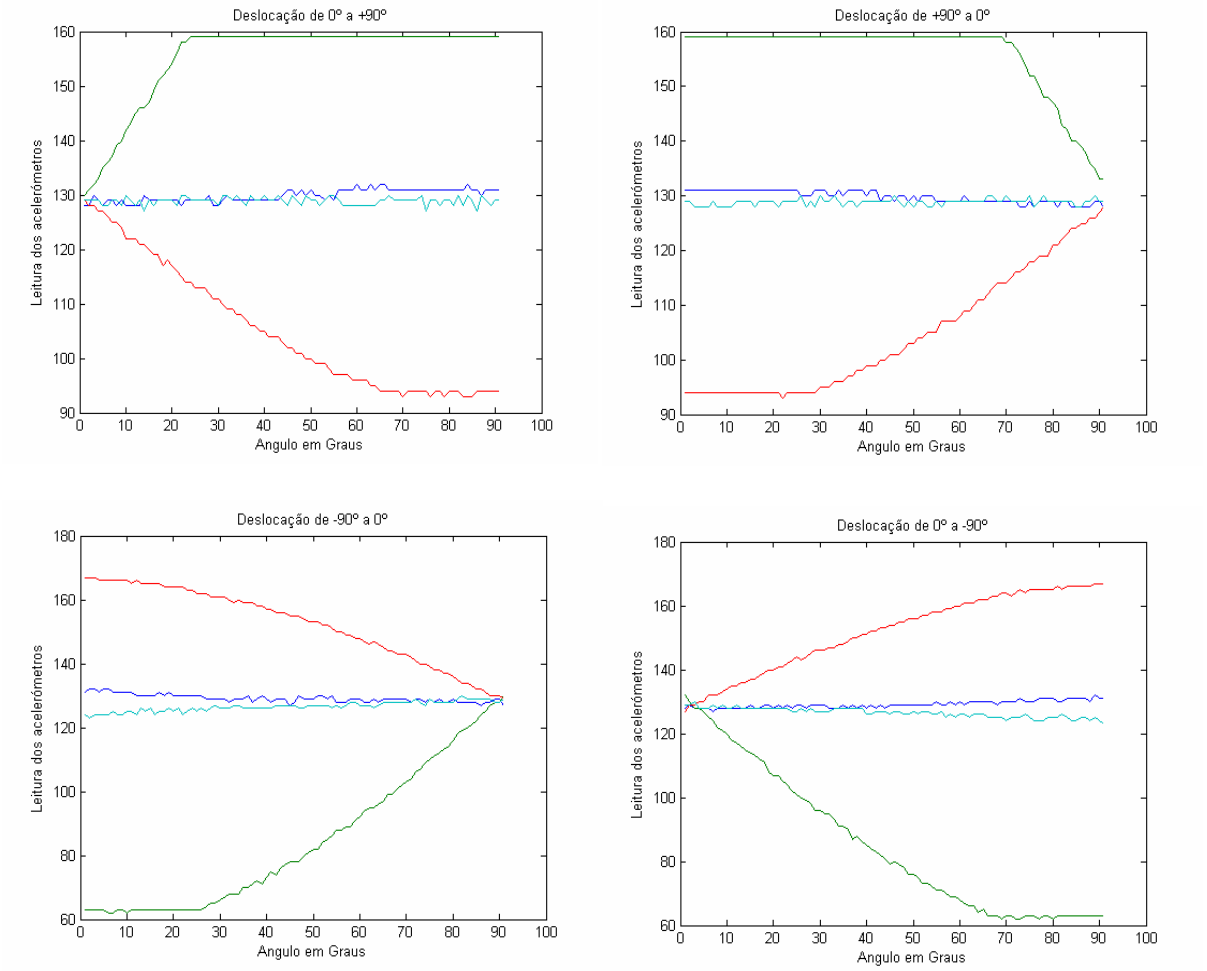


Fig. 3.11 - Aceleração da gravidade no eixo Y

Como se pode verificar pelas Fig. 3.10 e Fig. 3.11, os acelerómetros não são totalmente simétricos um em relação ao outro, isto é, basta um pequeno desvio na regulação dos potenciómetros dos dois eixos, dos dois acelerómetros, e como a posição referencial dos eixos já é diferente de um para o outro, vão existir desvios nas leituras. Adicionalmente, os componentes discretos escolhidos (condensadores e resistências) têm variações de uns para os outros, mesmo que tenham precisamente o mesmo valor, vão existir variações na resposta de um acelerómetro para o outro.

Verifica-se ainda que os acelerómetros têm só uma variação com a inclinação aproximadamente até ±40 graus, isto é, para 0 graus (horizontal, paralelo ao plano da terra) até ±40 graus de inclinação. Como se pode verificar depois dos ±40 graus existe uma saturação, mas esta situação pode ser corrigida com a aplicação da fórmula para o cálculo da inclinação, e ainda pela multiplicação por um factor de escala de correcção.

Regime dinâmico

Para o regime dinâmico foi utilizado o mesmo procedimento que no ponto anterior, mas agora a leitura é sequencial, sem esperas. Neste caso, leitura dos acelerómetros é influenciada pela a aceleração dinâmica do movimento aplicado.

- Leituras segundo o eixo X

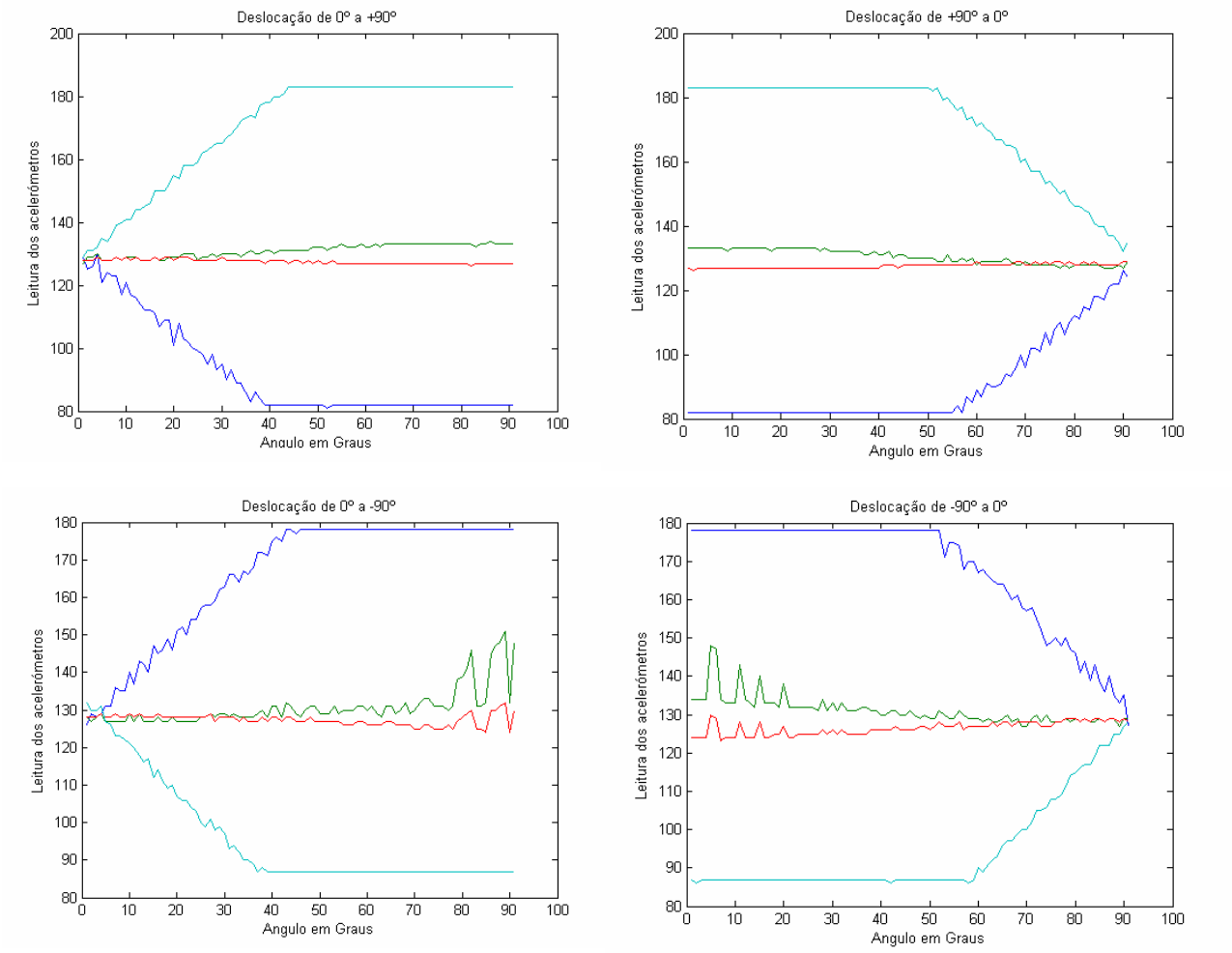


Fig. 3.12 - Aceleração da gravidade mais aceleração dinâmica no eixo X

- Leituras segundo o eixo Y

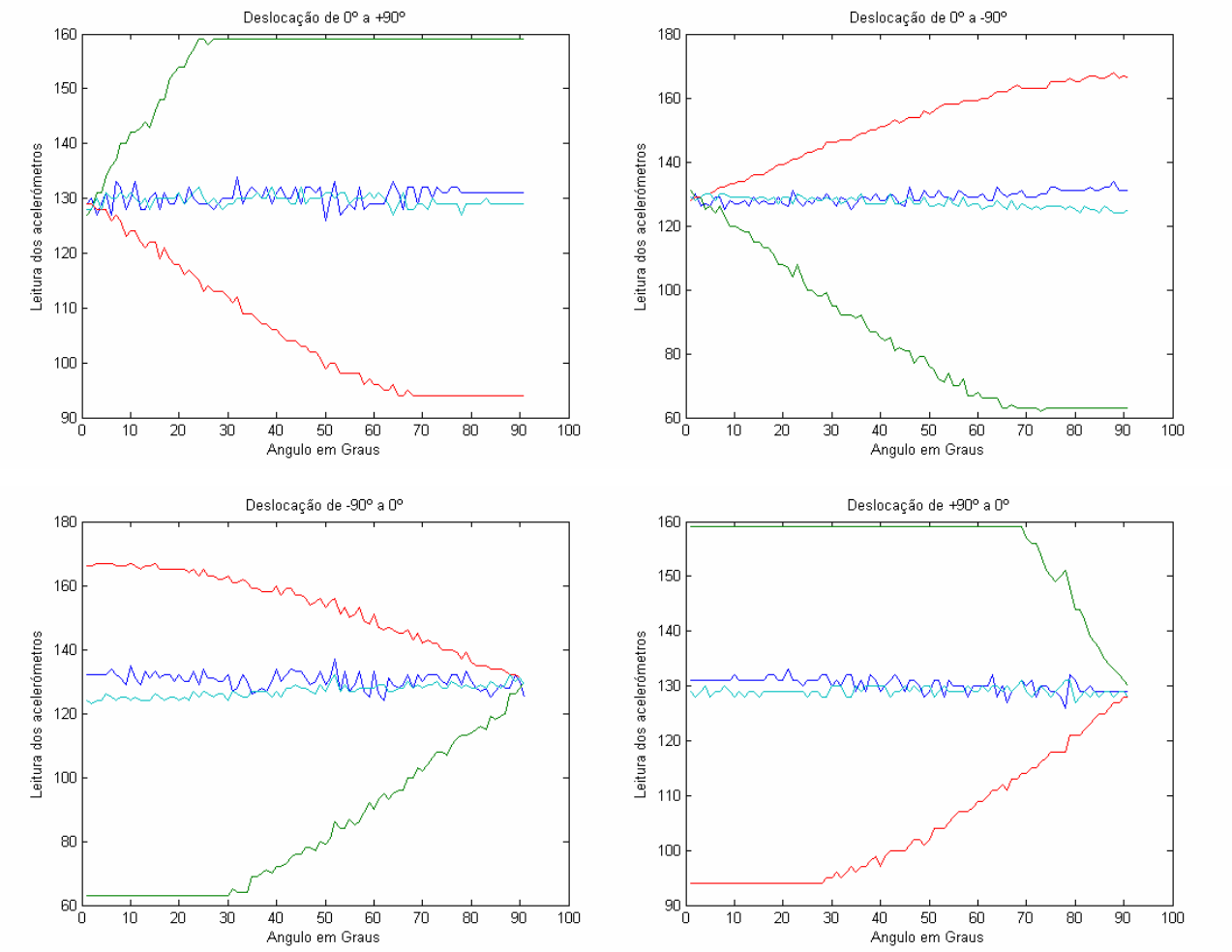


Fig. 3.13 - Aceleração da gravidade mais aceleração dinâmica no eixo Y

Como podemos verificar pelas Fig. 3.12 e Fig. 3.13, além de os acelerómetros medirem a aceleração da gravidade, adicionalmente, são medidos picos de acelerações dinâmicas, provenientes da aceleração do movimento do braço do actuador e das vibrações mecânicas dos actuadores.

Conclui-se que os acelerómetros são muito sensíveis a acelerações dinâmicas, o que pode trazer inconvenientes na altura do controlo da plataforma humanóide, em que os actuadores estão sempre activos, o que por si só, faz com que existam vibrações a percorrer toda a estrutura, o que vai influenciar na medida das acelerações estáticas, provocando erros de leitura desta.

3.4.3. Medidas dos acelerómetros de -90 a 90 graus

Esta experiência tem o mesmo fundamento que as anteriores, mas esta foi concebida para ser aplicado um intervalo progressivo de ângulos entre -90 a 90 graus como ilustram as Fig. 3.8 e Fig.

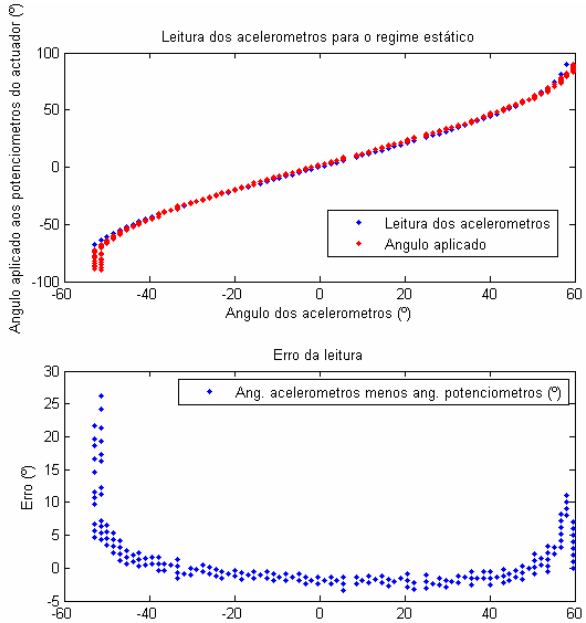
3.9.

Como nas experiências anteriores, foi confiado que o potenciómetro do actuador estava a medir o ângulo correcto em graus ao braço do actuador, para que o servo aplicasse o ângulo correcto ao braço, isto, para ter um ângulo de comparação entre a leitura do acelerómetro e o ângulo aplicado.

Regime estático

Para o regime estático, foi aplicado um ângulo ao actuador e feita uma espera de 0.5s antes da amostragem, para ter a certeza que o actuador se encontra parado, de modo a medir unicamente a aceleração da gravidade.

- Leituras segundo o eixo X



- Leituras segundo o eixo Y

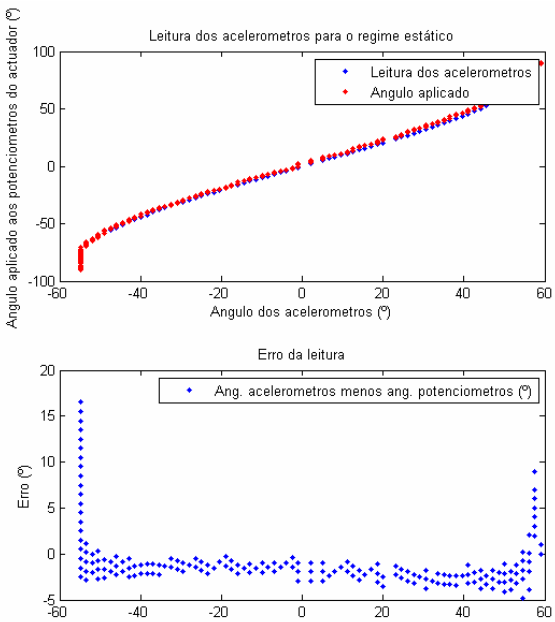


Fig. 3.14 - Leitura estática dos acelerómetros de +1*g* a -1*g*

Como se pode verificar pela Fig. 3.14, o ângulo aplicado ao actuador e o ângulo obtido pela expressão $Tilt=asin(A_X/1g)*K$ (em que A_X é o sinal proveniente do acelerómetro e K é um factor de escala) estão muito próximos um do outro. Verifica-se que, para leituras superiores a ±40 graus de inclinação, o erro começa a ser elevado, pelas várias razões já anteriormente descritas, como a saturação da leitura dos acelerómetros e a pouca sensibilidade dos acelerómetros perto de +90 e -90 graus. Contudo, as leituras entre 40 e -40 graus são totalmente fiáveis para a inclinação dos acelerómetros.

Regime dinâmico

- Leituras segundo o eixo X **- Leituras segundo o eixo Y**

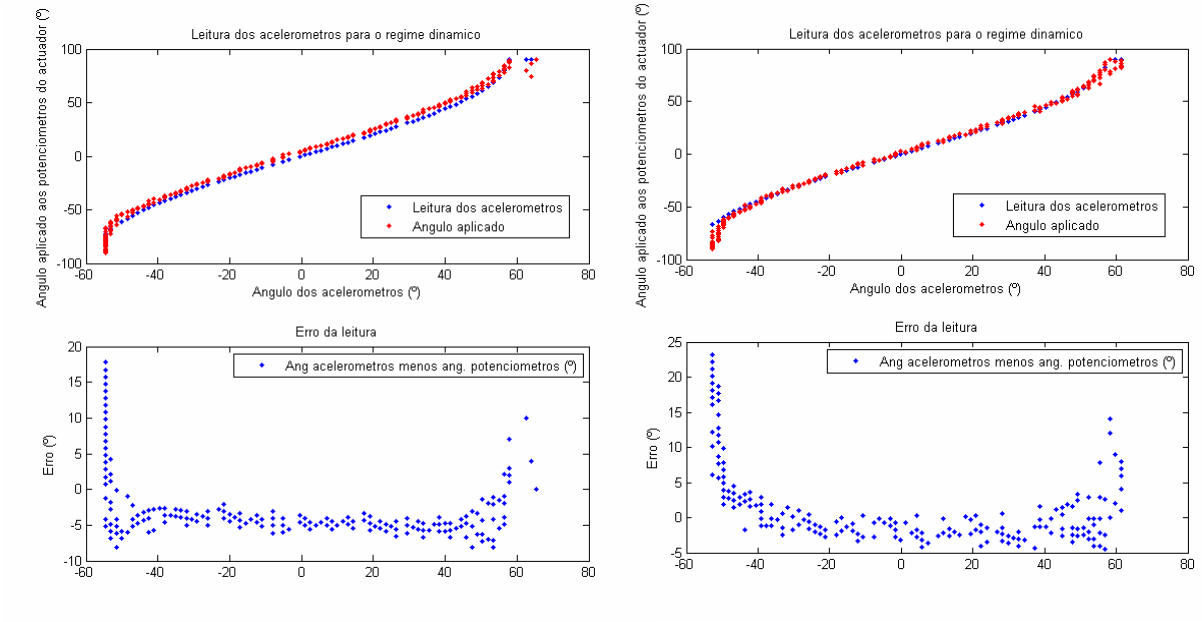


Fig. 3.15 - Leitura dinâmica dos acelerómetros de +1g a -1g

Para as acelerações dinâmicas verifica-se que o erro médio aumenta. Isto porque é introduzida aceleração do movimento do actuador à aceleração gravítica. Este desvio médio corresponde à aceleração imposta pelo actuador. Desta forma é possível estimar a aceleração dinâmica do actuador.

O erro para situações perto de -90 e +90 graus é aproximadamente igual à aceleração em regime estático. Conclui-se que com o presente circuito de acondicionamento de sinal dos acelerómetros destrói parte do sinal proveniente dos acelerómetros. Como foi descrito anteriormente, era necessário reduzir o ganho do amplificador (Fig. 3.4) para uma melhor leitura da curva característica dos acelerómetros.

4. Unidade central de processamento

Resumo:

Este capítulo descreve um estudo às unidades centrais de processamento. É necessário encontrar uma unidade central de processamento para a plataforma humanóide, que tenha dimensões reduzidas para ser transportada na plataforma. Foram pesquisadas várias soluções, desde *mainbroads*, *PC104* e *PDA's* para se escolher a melhor solução para a unidade central de processamento para a plataforma humanóide.

4.1. Introdução

A unidade de processamento actualmente utilizada e que se pretende substituir é um PC vulgar. Com a construção do robô humanóide já muito avançada, e estando este pronto para andar, é necessária a remoção dos cabos de ligação ao PC e a substituição deste por uma unidade central de processamento de pequenas dimensões para que o robô seja totalmente autómato e capaz de desempenhar determinadas tarefas sem intervenção humana.

A Unidade Central de Processamento é responsável pela gestão global dos procedimentos incluindo as seguintes tarefas: cálculo das configurações que as juntas devem adoptar com base em directivas de alto nível, processamento de imagens vídeo, interacção com computador externo para permitir monitorização, *debug* ou *tele-operação*.

Para a unidade central de processamento, foram estudados alguns tipos de soluções, em que estas são *mainbroads*, *PC104* e *PDA*. Entre estes três grandes tipos, dever-se-á escolher a solução mais adequada à plataforma, tendo em conta o nível de processamento exigido para controlar e processar as imagens.

4.2. Estado da Arte

4.2.1. Equipas participantes no RoboCup

Antes da procura da unidade central de processamento ideal para o projecto, foi feita uma pesquisa às várias equipas participantes no *RoboCup*. Foi necessário saber qual os processadores usados pelos vários participantes para se ter uma ideia do que seria necessário adquirir para a plataforma humanóide. Adicionalmente, foi necessário também saber qual o sistema operativo usado, o tipo de câmaras das diversas plataformas, o tipo de arquitectura de controlo, o tipo de actuadores usados e os graus de liberdade de cada plataforma. Assim, foi construída a tabela seguinte:

Nome da equipa	Processor Principal	Sistema Operação	Visão	Dof (º)	Actuadores	Arquitectura de Controlo
Abarenbou and DaoDan	Sony Clie PDA NR-70V 66MHz	Assembly	CMOS camera	17	Hitec and Futaba	Distribuído
Artisti Humanoid Team	VS-7054 board, SH2-7054 MCU@40MHz	Compilador	CMOS Camera OV7620 (OmniVision)	22	Sanwa Hyper ERG-VB	Centralizado
BreDoBrothers	Dell AximX50v and Fujitsu Siemens PocketLoox 720 PDA.	Microsoft Windows Mobile	LiveView FlyCam CF 1.3M	17	Kondo KRS-784ICS	Distribuído
Darmstadt Dribblers and Hajime Robots (3º prémio)	Fujitsu-Siemens Pocket PC 420 and Acer n50 Premium	Windows CE	Philips ToUCam Pro	24	Robotis Dynamixel DX117	Distribuído

Team Humboldt	PDA Fujitsu-Siemens Pocket Loox 720,	Microsoft Windows Mobile	Conrad CCD Color Cam,	21	Robotis Dynamixel AX-12	Distribuído
JEAP Team	PC 104 PNM-SG3, SH2	?? (UML)	CCD camera Quickcam	24	Robotis Dynamixel DX117	Distribuído
NiciCo	ATMega 128 16MHz	Compilador	CMU camera	18	KONDO, KRS-768 ICS	Distribuído
NimbRo (2º prémio)	Fujitsu Siemens Computers Pocket Loox 720	Microsoft Windows Mobile	Lifeview FlyCam 1.3M CF	19	Futaba S9152	Distribuído
Pioneros México	ATMEL 89C52 24MHz	Assembly	CMU Camera	16	Futaba S3003 HITEC HS-5645MG	Centralizado
Robo-Erectus	SONY VAIO VGN-U8G 900MHz	Microsoft Windows	CMU Camera	23	Hitec HSR-5995TG Hitec HS-5945MG	Distribuído
RO-PE	Kontron MOPSlcd7-Mobile Intel Pentium III CPU	Microsoft Windows	Omni-Directional Vision System VS-C14N	17	Robotis Dynamixel DX117	Centralizado
KMUTT	ARM7 TDMI-S 60MHz	Compilador	CMUcam	22	HITEC HS5995TG	Distribuído
Team Osaka (1º prémio)	GeodeLX800 @ 0.9w 400MHz		Logicool usb omnidirectional camera	23	Robotis Dynamixel DX117	Distribuído
TH-MOS	MEGA128 8MHz	Compilador	Acroname, Inc. CMOS camera	20	KONDO KAGAKU	Distribuído
TKU	CycioneEP1C12F324C8 50MHz	Compilador	Digital compass CMOS sensor	24	KONDO KRS-2350ICS	Distribuído
Toin Phoenix	PowerPC 400MHz	Compilador	CMOS camera	19	Robotis Dynamixel DX117	Distribuído

Tabela 4.1 - Características das várias equipas participantes no RoboCup

Na Tabela 4.1[3] verificamos que as unidades centrais de processamento mais usadas são os *PDA's* e os sistemas de operação mais utilizados são o *Windows* e sistemas compilados para microcontroladores. As câmaras mais utilizadas são as omnidireccionais (mas estas no entender das organizações humanóides, não têm semelhança com o Homem) e as CMOS. Os actuadores mais comuns são *Robotis Dynamixel* e os *Kondo* e, por fim, a arquitectura de controlo é normalmente distribuída.

4.2.2. Mainboards

Com o aumento da elevada capacidade computacional dos processadores e com a redução dimensões destes, começam a ser utilizados em dispositivos altamente compactos com elevadas capacidades de processamento. Estes processadores extremamente compactos são especialmente fabricados para dispositivos móveis de reduzidas dimensões, de baixo consumo e sem nenhum padrão estabelecido.

A utilização de uma *mainboard* não implica obrigatoriamente a utilização de um disco rígido

[3] Referência bibliográfica 8

para o armazenamento dos dados, dado que os sistemas operativos podem ser instalados numa *flash memory* já inserida na *mainboard* pois esta já tem uma zona para o armazenamento de dados específica.

CPU Type	Nano-ITX	Mini-ITX	Intel XScale PXA270	AMD Geode LX800	MPC8271	VIA C3/C7 CoreFusion	Intel XScale PXA255	AMD Geode CS1200	Pentium III	AMD Elan SC520
CPU Speed (MHz)	1000	600	520	500	400	1000	400	300	1260	133
Memory	1.0GB DDR400	128MB SDRAM	128MB SDRAM	1.0GB DDR	128MB SDRAM	256MB DDR	128MB SDRAM	128MB SDRAM	128MB SDRAM	128MB SDRAM
Display interface	VGA	RCA TV out	VGA	LCD	LCD	VGA	VGA	LCD	VGA	LCD
Video input port	S-Video	S-Video	Direct camera	1	none	S-Video	none	none	none	none
Flash Disk		32MB	512 MB	512 MB	512 MB	512 MB	512 MB	512 MB	512 MB	128 MB
USB	4	4	4	3	2	4	2	3	2	2
COM(serial) Ports	1	1	1	1	1	1	1	1	1	2
PCI	1 nini PCI	2 PCI	1 PCI	none	none	1 PCI	none	1 PCI	1 PCI	1 PCI
Hard Disk interface	2 IDE	2 IDE	1 IDE	1 IDE	1 IDE	1 IDE	1 IDE	1 IDE	1 IDE	1 IDE
10/100Mb Ethr Ports	2	2	2	2	2	2	2	2	2	2
PCMCIA	none	none	1 slot	none	1 slot	none	1 slot	none	none	none
Wi-Fi	none	none	WiFi 802.11	none	none	none	none	none	none	none
O/S Support	Win CE XP Em and Linux	Win CE XP Em and Linux	Win CE XP Em and Linux	Win CE XP Em and Linux	Win CE XP Em and Linux	Win CE XP Em and Linux	Win CE XP Em and Linux	Win CE XP Em and Linux	Win CE XP Em and Linux	Win CE XP Em and Linux
Dimensions	12cm X 12cm	17cm X 17cm	6.8cm X 5.8cm	6.8cm X 5.8cm	6.8cm X 5.8cm	7.0cm X 8.6cm	6.6cm X 4.4cm	6.8cm X 5.8cm	9.1cm X 9.6cm	7.8cm X 6.0cm
Supply Voltage	+3.3V +5V +12V IDE	+3.3V +5V +12V IDE	+3.3V +5V	+3.3V	+3.3V +5V	+3.3V +5V	+3.3V +5V	+3.3V +5V	+3.3V +5V	+3.3V +5V
Power Consumption	14.0W	6.0W	2 W	5 W	3W	12W	2 W	5 W	40W	4 W
Price	$332	$145	$100 to $200	$150 to $300	$120 to $200	$200 to $300	$100 to $200	$120 to $200	$250 to $500	$100 to $200

Tabela 4.2 - CPU mainboards

Como podemos ver ao analisar a Tabela 4.2[4] as diversas características das *mainboards*, estas poderão ser uma das melhores soluções para a unidade central de processamento. A utilização de uma *micro-mainboard* tem muitas vantagens como, por exemplo, uma enorme capacidade de processamento e a utilização de um sistema operativo que vai facilitar muitas funções a implementar.

Na Tabela 4.2 é de notar que quanto maior for a velocidade de processamento da placa maior é o consumo desta. Assim, verificamos logo que o Pentium III não pode ser uma solução para o nosso projecto pois tem um enorme consumo de energia.

[4]Referências bibliográficas 1 e 4

4.2.3. PC 104 e PC 104 plus

A PC 104 foi inicialmente criada para que todos os fabricantes de componentes para PC's tivessem módulos standard para aplicações embutidas, para que estes pudessem comunicar uns com os outros, e tirar partido de todas as funções dos vários módulos.

As grandes vantagens de uma PC 104 sobre um vulgar PC são as dimensões da placa (que na PC 104 são 3.6" x 3.8" ou 96mm x 91mm), o modo único e standard de retirar o barramento da placa, os pinos e os *socket*, que são de, exactamente, 60 e 40 contactos (macho/fêmea) para interligação das placas, e o barramento com correntes máximas de 6 mA para que cada módulo não tenha um consumo superior a 1 ou 2 Watts.

A grande diferença entre a PC 104 e a PC 104 plus é que esta última inclui um barramento PCI.

CPU Type	Intel XScale PXA270	Intel XScale PXA255	AMD Geode	VIA C3/C7 C3/ESP/C7	Celeron or Pentium-III	Intel XScale IXP425
CPU Speed (MHz)	300-520	200-400	133-500	300-1000	600-1260	533
Instruction set	ARM	ARM	X86	X86	X86	X86
DRAM Size (MB)	16-128	16-64	32-1000	64-256	32-384	16-64
Flash Disk Size (MB)	1-512	1-512	1-512	32-512	32-512	32-512
Display Type	LCD & CRT	LCD & CRT	LCD & CRT	LCD,CRT,TV	LCD &CRT	LCD & CRT
Display Res. (max)	1280x1024	800x600	1280x1024	1600x1200	1600x1200	800x600
Display Color (max bpp)	16	16	800x600	24	24	16
10/100Mb Ethr Ports	2	2	2	2	2	2
COM (serial) Ports	2	2	3	2	2	3
USB Ports	4	2	3	4	2	4
PC Card / Card Bus Slot	2	2	2	2	2	2
PC/104+ Compatible	Yes	Yes	yes	yes	Yes	Yes
Hard Disk interface	1 IDE	1 IDE	1 IDE	1 IDE	1 IDE	None
General Purpose I/O	30	30	20	4	20	20
O/S Support	Linux Win CE	Linux Win CE	Linux Win CE Win XP Em	Linux Win CE Win XP Em	Linux Win CE Win XP Em	Linux
Size (mm)	96x91x12 111x91x12	96x91x12	96x91x12	96x91x25 111x91x25	96x91x30	96x91x25
Supply Voltage	+3.3V +5V	+3.3V +5V	+3.3V +5V	+3.3V +5V	+3.3V +5V	+ 5V
Power Consumption	5W	4W	6W	10W	40W	6W
Power Down mode	yes	yes	yes	yes	yes	None
Price	$70 to $200	$70 to $200	$70 to $150	$70 to $200	$70 to $200	$200

Tabela 4.3 - PC 104 plus

Como podemos ver ao analisar a Tabela 4.3[5], todas as soluções aqui apresentadas para a unidade central de processamento são PC 104 plus. Esta é uma boa solução para fugir aos preços elevados das *mainboards*, mas perde-se capacidade de processamento.

Verificamos que o Pentiun III em versão PC 104 tem maior velocidade de processamento, maior RAM disponível e todas as conexões necessárias para o projecto. Contudo, este tem uma potência consumida exagerada o que o retira logo das opções.

[5] Referências bibliográficas 1, 2, 3, 6 e 7

4.2.4. PDA's

Originalmente, o PDA foi fabricado para ser uma agenda de bolso electrónica. Actualmente, este é mais que uma agenda, é um computador com elevado grau de processamento para as suas dimensões muito reduzidas. As suas funções são um pouco limitadas, mas não deixa de ser uma solução interessante.

Name	Pocket Loox 720	e830	MyPal A730	iPAQ hx4705	Cassiopeia DT-10	Axim X50v
Manufacturer	Fujitsu-Siemens	Toshiba	Asus	HP	Casio	Dell
Dimensions	4.8" x 2.8" x 0.7" / 122 x 72 x 15.2mm	5.3" x 3.0" x 0.7" / 135 x 77 x 16.7mm	4.6" x 2.9" x 0.6" / 117 x 73 x 17mm	5.17" x 3.03" x 0.59" / 131 x 77 x 15mm	5.5" x 3.2" x 1.0" / 140 x 80 x 25mm	4.7" x 2.9" x 0.7" / 119 x 73 x 16mm
Weight	6.0oz / 170g	7.0oz / 200g	6.0oz / 170g	6.6oz / 187g	10.6oz / 300g	6.2oz / 175g
Processor	520MHz (Intel XScale)	520MHz (Intel XScale)	520MHz (Intel XScale)	624MHz (Intel XScale)	416MHz (Intel XScale)	624MHz (Intel XScale)
Graphics Processor	integrated	integrated intel chipset	integrated intel chipset	ATI 3220	none?	Intel 2700G (Marathon w/extended memory)
Screen	3.6" VGA (640x480) transreflective TFT	4.0" VGA (640x480) transreflective TFT	3.7" VGA (640x480) transreflective TFT	4" Transflective VGA (640x480) TFT	3.7" VGA (640x480) TFT	3.7" VGA (640x480) transreflective TFT
RAM	128MB (123MB available to user)	128MB (124MB available to user)	64MB (46MB available to user, 128MB RAM on A730w model)	64MB (55MB available to user)	128MB (?MB available to user)	64MB (55MB available to user)
ROM	64MB (28MB available to user)	64MB (24MB available to user)	64MB (29MB available to user)	128MB (80MB available to user)	64MB (20MB available to user)	128MB (80MB vailable to user)
Body	Plastic	Plastic	Plastic	Magnesium alloy	Shock resistant, drop proof, dust proof	Plastic
Battery	Removable 1640Ah lithium-ion	Removable 1320Ah lithium-ion	Removable 1100Ah lithium-ion	Removable 1800mAh lithium-ion	Removable 2300mAh lithium-ion	Removable 1100Ah lithium-ion
Extended Battery	unknown	2640mAh available	1800mAh available	3600mAh battery available	none	2200mAh available
Wi-Fi	802.11b	802.11b	None (802.11b on A730w model)	802.11b	802.11b	802.11b
Bluetooth	Bluetooth 1.2	Bluetooth 1.2	Bluetooth 1.1	Bluetooth 1.2	Bluetooth 1.2	Bluetooth 1.2
Infrared	IrDA (consumer infrared)	IrDA (115kbps)	IrDA (115kbps)	IrDA FIR (fast infrared)	IrDA 1.3 (4Mbps)	IrDA (115kbps)
CompactFlash	Type II card slot	Type II card slot	Type II card slot	Type II card slot	Type II card slot	Type II card slot
Secure Digital	SDIO, SD, and MMC	SDIO, SD, and MMC	SDIO, SD, and MMC	SDIO, SD, and MMC	SDIO, SD, and MMC	SDIO, SD, and MMC
Camera	1.3 megapixel with integrated flash	none	1.3 megapixel camera with integrated flash	None	None	None
Sync Methods	USB 1.1 and serial	USB (1.1?) and serial	USB 1.1 and serial(?)	USB 2.0 and serial	USB 1.1 and serial	USB 1.1
Navigation	5 way navigation button & jog dial	5 way navigation button & jog dial	5 way navigation button	Touchpad	5 way navigation button	5 way navigation Button
Available Now	Yes (in Europe, not US)	Yes (in Europe Canada, and Japan, not US or UK)	Yes (in Asia and US)	Yes	No (Available from November in Japan)	Yes (in Europe Canada, and US)
Retail Price	about $700	$667	$499.95	$649.99	$404.10	$499

Tabela 4.4 - PDA's

Analisando a Tabela 4.4[6] verificamos que todos os PDA's têm características muito similares entre si. Destacam-se alguns, por possuírem características que podem ser mais adequadas ao projecto, como câmara integrada, maior velocidade de processamento, maior memória, entre outras. A principal desvantagem do *PDA's* é o preço destes.

4.2.5. Escolha da unidade central de processamento

De todas a soluções aqui apresentadas a mais relevante é a *nano-ITX*, pois esta é uma placa de dimensões muito reduzidas, de grande capacidade de processamento e memória, com todos os conectores standard e fixos na placa. As desvantagens da *nano-ITX* são o seu custo e a sua potência consumida.

A segunda melhor opção é o *AMD Geode LX800*, em versão mainboard ou PC 104 plus. A sua capacidade de processamento é menor do que a da *nano-ITX*, e as ligações terão que ser tiradas do barramento para interligar os periféricos, no entanto, compensa em termos de potência consumida, custo e nas vantagens subjacentes ao facto de ser uma PC 104 (se for esta a versão escolhida).

A terceira melhor opção é a *Intel XScale PXA270*, em versão mainboard ou PC 104 plus, tendo uma velocidade de processamento razoável, e todas as conexões para interligar os periféricos necessários ao projecto, sendo de baixo custo e baixa potência consumida.

4.2.6. Conclusão da pesquisa

É fundamental que a solução para a unidade central de processamento tenham um ambiente flexível e versátil para a programação desta. Este ambiente implica a utilização de um monitor, teclado e rato para a interacção com o utilizador e um sistema operativo para abstracção do hardware. Esta solução necessitará ainda de capacidades para o processamento das imagens vídeo e o controlo total de todas as juntas do humanóide assim como os parâmetros para essas juntas.

A *nano-ITX* da *VIA Technologies*, traz todo o software necessário para criar um ambiente flexível e versátil. Nas soluções *AMD Geode LX800* e *Intel XScale PXA270* também existe software disponível para a interacção com o utilizador e os diversos *device drivers*, para interligar os periféricos.

Em relação aos PDA's, destaca-se o *Fujitsu-Siemens Pocket Loox 720*. Apesar de os PDA's terem custos muito elevados para o projecto, tendo em conta que todas as funcionalidades existentes já estão configuradas e a correr sobre um sistema operativo, basta só criar um programa que interligue todos os dispositivos, que processe as imagens vídeo e controle o robô.

[6] Referência bibliográfica 5

Considerando estas capacidades do *PDA* chegamos à conclusão que até pode ser uma boa alternativa.

É de notar que nenhuma das soluções apresenta a porta *FireWire IEEE-1394*. Contudo, apesar deste inconveniente, todas as placas têm portas USB e através destas é possível, com adaptadores, ligar os diversos periféricos como câmaras de vídeo, *wireless pen's*, *bluetooth pen's* …

Visto que a porta *FireWire IEEE-1394* é importante para o projecto, podemos inclui-la através de diversos módulos externos. Por exemplo, a PC 104 e as *mainboards* (com controladores) têm porta PCMCIA onde se pode ligar um conversor *PCMCIA-Firewire*. Também existe um módulo de portas *FireWire IEEE-1394* para o barramento PCI, que pode ser uma solução para a placa escolhida.

4.3. Unidade central adquirida

Por motivos comerciais e económicos, a unidade central de processamento adquirida foi o *AMD Geode* em versão PC 104 plus, pois foi a melhor solução encontrada nos representantes das diversas marcas de venda de *embedded motherboards*. Mais especificamente, foi a *PC/104 PM-LX-800* (Fig. 4.3) e o módulo *PC/104 Dual PCMCIA Module* (Fig. 4.4) da *IEI Technology Corp.*

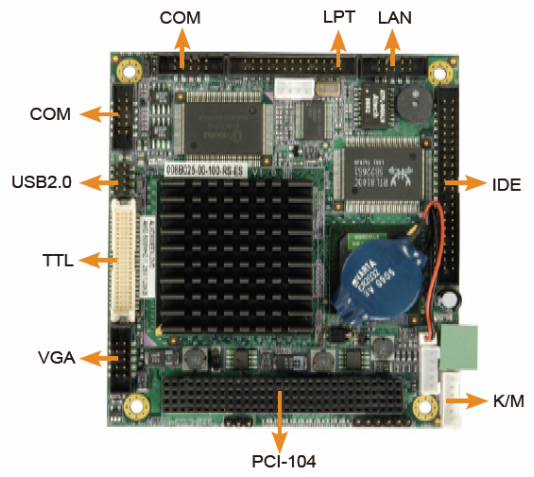


Fig. 4.1 - PM-LX-800 da IEI Technology Corp - Top

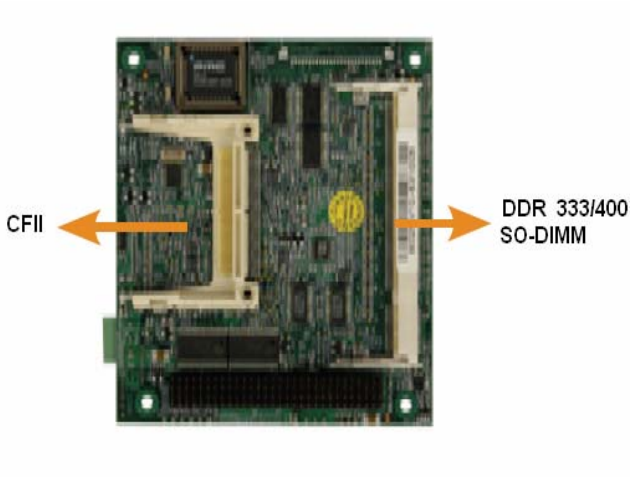


Fig. 4.2 - PM-LX-800 da IEI Technology Corp - Bottom

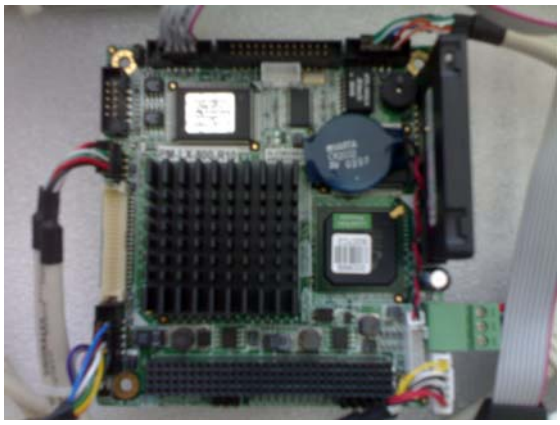



Fig. 4.3 - PM-LX-800 da IEI Technology Corp

Na tabela seguinte, estão as especificações esta placa.

PM-LX-800 R10 SPECIFICATION	
CPUs Supported	AMD® Geode LX-800 @ 500 MHz
Cache Memory	64K I/ 64k D L1 cache, 128K L2 cache
System Chipset	AMD® CS5536
I/O Controller	AMD® CS5536
Memory	One 200-pin DDR 333/400MHz SO-DIMM SDRAM with a maximum size of 1GB.
PCI Bus Interface	Revision 2.2
Super IO	W83627EHG
Display	CRT integrated in AMD® Geode LX800
TTL/ LVDS	24 bit TTL integrated in AMD LX800 18 bit LVDS
HDD Interface	IDE channel support
Power Support	AT/ATX power suppor
Power Consumption	+5V @ 0.92ª (DDR333 256MB)
Watchdog Timer	Software programmable supports 1~255 sec. system reset
I/O Interfaces	2 x USB 2 .0 1 x LPT 1 x CFII 1 x IDE
Ethernet	10/100Base-T RTL8100C
BIOS	AWARD
Physical Dimensions	91mm x 95mm
Weight	GW:0.65Kg; NW: 0.25Kg
Operating Temperature	Minimum: 0ºC (32°F) Maximum: 60°C (140°F)

Tabela 4.5 - Especificações da PM-LX-800 R10

Adicionalmente, com esta placa vêm os diversos cabos de ligação, alguns manuais e CD, como se pode verificar na Tabela 4.6.

Packing List	
1 x PM-LX single board computer	1 x LAN cable
1 x Mini jumper pack	1 x Power cable
1 x IDE flat cable	1 x KB/MS cable
2 x RS232 cable	1x Utility CD
1 x USB cable	1 x QIG
1 x VGA cable	

Tabela 4.6 - Componentes vindos com a PM-LX-800 R10

A placa para a ligação da câmara *FireWire* e para a ligação *wireless*, foi adquirido um *Dual PCMCIA Module* pois esta é capaz de suportar duas placas PCMCIA, e através de conversores

PCMCIA pode-se ligar os vários periféricos necessários. A ligação *wireless* será útil, futuramente, para se poder comunicar entre vários humanóides ou mesmo para *tele-operação*. A ligação *FireWire* é necessária para aquisição das imagens provenientes da câmara.

Fig. 4.4 - Modulo dual PCMCIA

Para o total funcionamento destes dois módulos foi necessário adquirir uma memória *DDR RAM* de 256 Mb, com 200 pinos (Fig. 4.5) e um *Solid State Disk* IDE de 44 pinos (Fig. 4.6) para o armazenamento do sistema operativo e dos ficheiros de projecto.

Fig. 4.5 - DDR RAM

Fig. 4.6 - Solid State Disk

Estes vários módulos vão ser fixos ao tronco da plataforma humanóide e alimentados por baterias, para que este seja totalmente autónomo. A estrutura do tronco terá que ser refeita para poder ser colocada a unidade central de processamento, logo, a alteração ao seu peso vai influenciar no centro de massa da estrutura. No entanto, como o tronco será totalmente refeito, será

necessário recalcular o novo centro de massa da estrutura, para que todos os algoritmos de locomoção sejam refeitos com o centro de massa correcto, permitindo o perfeito funcionamento do robô, à semelhança do que acontece agora.

5. Arquitectura das comunicações implementadas em Linux

Resumo:

Este capítulo descreve a organização distribuída da plataforma humanóide e o modo como é realizada a comunicação via RS-232 entre a unidade central de processamento e o *master*. Como o sistema operativo implementado na unidade central de processamento é o Linux, foi elaborado um conjunto de funções e classes para se proceder à comunicação entre a unidade central e o *master*.

5.1. Arquitectura do sistema

O sistema implementado na plataforma humanóide é distribuído e constituído por três tipos de unidades, uma de alto nível (unidade central de processamento), outra unidade para a gestão das comunicações (*master*) e, por fim, uma unidade de baixo nível (*slave*), para controlo de actuadores e leituras sensoriais. O sistema distribuído implementado é baseado numa configuração *master*/*slave*.

- A **unidade central de processamento** é responsável pela gestão global dos procedimentos, efectuando o cálculo das configurações que as juntas têm de adoptar em função dos valores dos sensores e do processamento de imagem.
- A **unidade Master (mestre)** tem como principal tarefa distribuir os comandos provenientes da unidade principal pelas diversas unidades locais (*slaves*), bem como direccionar os dados sensoriais provenientes dos *slaves* para a unidade principal.
- As **unidades Slave (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.

Entre os diversos nós são utilizados como meios de comunicação:

- Linha série ponto-a-ponto, baseada na norma **RS-232**, entre a unidade central de processamento e a unidade *Master*: acesso assíncrono byte a byte a um *baudrate* de 115200 bps.
- **CAN** (*Controller Area Network*) entre a unidade *master* e as unidades *slave*: é utilizada a versão *full*CAN 2.0A a uma taxa de transmissão/recepção de 1Mbps.

O sistema operativo implementado na unidade central de processamento é o Linux. Foi necessário criar os diversos *device drives* para se estabelecer a comunicação RS-232 entre a unidade central de processamento e o *master*.

Para as unidades de controlo local (*master*/*slave*), a escolha recaiu sobre os microcontroladores PIC da série 18F da *Microchip* – PIC18F258(0) – por possuírem diversos periféricos e interfaces para redes de comunicações, incluindo o CAN (

Fig. 5.3).

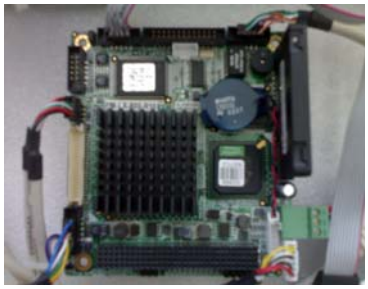


Fig. 5.1 - PC-104 do robô

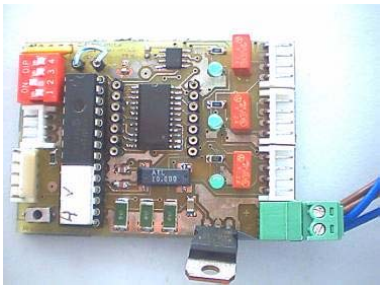


Fig. 5.2 - Placa de controlo Master/Slave

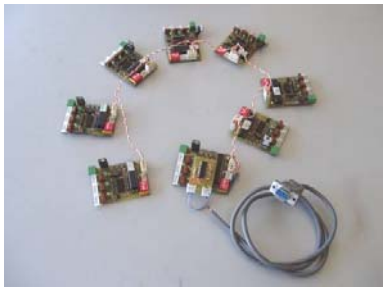


Fig. 5.3 - Rede completa de Microcontroladores

A estrutura implementada é constituída por uma unidade central de processamento, que vamos designar por CPU (*Central Processing Unit*), uma placa *master,* que designaremos por MCU (*Master Control Unit*), que efectua a interface entre a unidade principal e as unidades *slave*, e oito placas *slave*, designadas por SCU (*Slave Control Unit*), que efectuarão o controlo local até três actuadores através da geração de uma onda de impulso modulado em largura (PWM), e a aquisição de até 16 sinais analógicos usando um *multiplexer* (Fig. 5.2)

A organização enunciada na Fig. 1.5 tem como objectivo agrupar as juntas que estão directamente relacionadas, como é o caso das juntas do tornozelo e do joelho que possuem um controlador dedicado que, por aquisição dos sinais analógicos dos sensores de força instalados nos pés, pode controlar o equilíbrio por compensação em malha fechada. Desta forma obtém-se 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 podem ser de quatro origens diferentes:

- Potenciómetros internos aos servomotores, indicativos da sua posição;
- Extensómetros presentes na base de cada pé, para medição da força aplicada, para posterior cálculo do centro de pressão exercido;
- Acelerómetros/inclinómetros que, pela medição da aceleração da gravidade em duas componentes ortogonais, medem a inclinação do objecto onde se encontram localizadas;
- Giroscópios para medição da velocidade angular, para compensação das forças dinâmicas exercidas sobre o robô.

Estes valores são convertidos e registados localmente em cada unidade *slave* sendo depois enviados via CAN para o controlador *Master*. Os *Slaves* estão preparados também para receber mensagens via CAN. Estas mensagens consistem, basicamente, nos dados de actuação a aplicar sobre cada unidade local:

- Posições finais que os actuadores têm de tomar;
- Velocidade a que têm que se mover até atingir a posição final;

- Tipo de controlador em acção sobre as três juntas;
- Parâmetros de compensação para os algoritmos de compensação;
- Flags booleanas de controlo (ex.: PWM activados).

O controlador *master* tem a tarefa de receber a informação enviada pelos *slaves* via CAN e registá-la, para que esteja disponível para ser enviada para a unidade central de controlo quando solicitada. Este controlador mantém, por isso, uma representação do estado actual das juntas (actuadores e sensores) que disponibilizará ao controlo central sempre que este o pedir. O processo é bidireccional e o controlador *master* também recebe as ordens da unidade central e despacha para o *slave* respectivo.

Unidade Central de Processamento	**Unidade central de processamento com porta série RS-232** • Sitema operativo de suporte: Linux • Device Drivers de comunicação série
Unidades de controlo local *Master/Slave*	**PIC18F258 da Microchip** • Memória de programa: 2 MB • Memória de dados: 4 KB • Velocidade de processamento: 10 MIPS (f_{osc}=40MHz com a PLL activa) • Instruções de 16 bits e *datapath* de 8 bits • Definição de dupla prioridade nas interrupções. • Diversos periféricos: *timers*, módulos CCP, interfaces para redes de comunicação, ADC, etc…

Tabela 5.1 - Descrição do sistema

5.1.1. Protocolos de Comunicação

Desenvolveu-se o protocolo de comunicação para a linha série RS-232, entre a unidade central de processamento e a unidade *master*, em cooperação com o CAN entre a unidade *master* e os *slaves*, para que se possa trocar dados sensoriais e de actuação entre o CPU e as unidades *slave*.

Entre os dados sensoriais podem-se enumerar (para um SCU):

- Posição dos três servomotores (em graus);
- Velocidade correspondente (em graus/s);
- Corrente consumida por cada servomotor;
- Valores dos sensores de força de cada pé (quatro sensores por pé);
- Saída dos Giroscópios (em graus/s);
- Saída dos inclinómetros.

Quanto aos dados de actuação:

- Posição final a atingir por cada servomotor;
- Velocidade média para a realização do movimento;
- Parâmetros de compensação para o controlo local (K_P, K_I e K_D).

Considera-se que o controlo de actuação deve ser feito de forma isolada a cada uma das três juntas, por isso as diversas juntas podem realizar movimentos com diferentes velocidades e diferentes parâmetros de compensação. Tal é muito útil, pois cada junta pode estar sujeita a diferentes esforços, e por isso, é rentável aplicar diferentes parâmetros de controlo a cada uma.

5.2. Comunicação RS-232 entre a CPU e o Master

A comunicação RS-232 entre o CPU e a unidade *master* é efectuada assincronamente e atomicamente, e é orientada ao byte (um *start* bit + 8 *data* bits + um *stop* bit), ou seja, é transmitido um byte de dados em cada transmissão/recepção. Pretende-se que numa única mensagem esteja contida toda a informação relativa a um parâmetro a ler/actuar das três juntas de um SCU, o que implica que cada mensagem seja constituída por vários bytes. Por exemplo, para ordenar o SCU *X* a colocação das juntas nas posições *A*, *B* e *C*, são necessários no mínimo seis bytes: três para as três posições *A*, *B* e *C*; e um indicando a identificação do SCU, um de controlo e um de verificação de dados.

Estrutura de um pacote RS-232 (transmissão/recepção de um byte)

5.2.1. Comunicação CPU→*Master*

As mensagens no sentido CPU→*Master* são constituídas por seis bytes. O primeiro byte sinalizará a mensagem como sendo um comando de solicitação à unidade *master* – MESSAGE_REQ (ver Tabela 5.3), o segundo byte conterá um código (*opcode*) indicativo da operação a realizar, do SCU alvo e de parâmetros adicionais, os três bytes seguintes conterão parâmetros a atribuir às três juntas, no caso de um comando de actuação, e finalmente o byte BCC indicará a validade da mensagem (como não tendo sofrido corrupção) (ver Tabela 5.2).

byte 1	*byte 2*	*byte 3*	*byte 4*	*byte 5*	*byte 6*
SOF: *0x*FF	OpCode	Joint 1	Joint 2	Joint 3	BCC

Mensagem USART CPU→Master de actuação.

byte 1	*byte 2*	*byte 3*	*byte 4*	*byte 5*	*byte 6*
SOF: *0x*FF	OpCode	0	0	0	BCC

Mensagem USART CPU→Master de leitura sensorial

Pacotes	***Função:***	***Valores possíveis:***
SOF	*(Start Of Frame)* Indica o tipo da mensagem.	(Ver Tabela 5.3).
OpCode	Código que indica o que é solicitado e a quem se destina.	(Ver Tabela 5.4)
Joint 1/2/3	Dados de actuação.	• Mensagem de actuação: Dados de actuação a atribuir a cada componente de um determinado SCU. • Mensagem de leitura sensorial: campos nulos.
BCC	*Block Check Code* Verificação da integridade da mensagem.	$BCC = \sum_{0}^{N} bytes \Vert 256 \Vert$

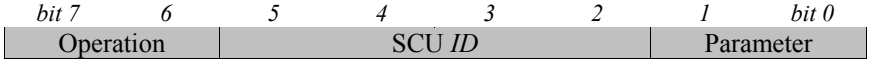
Tabela 5.2 - Campos das mensagens CPU→*Master* via USART.

A Tabela 5.3 descreve quais são as possibilidades do SOF (S*tart* O*f* F*rame*) sendo este o primeiro byte da trama a ser enviado. Este parâmetro vai descrever o tipo de mensagens a que a trama se refere.

Tipo de Mensagem	**Designação**	**Código**	**Descrição**
Mensagem de solicitação	*MESSAGE_REQ*	*0x*FF	Solicitação de dados do PC para o *Master*.
Mensagem de teste	*MESSAGE_TEST*	*0x*FE	Envio de uma mensagem de teste das comunicações.
Mensagem de sucesso	*MESSAGE_SUCESS*	*0x*FD	Resposta a uma mensagem de solicitação (MESSAGE_REQ), indicando que o comando foi executado com sucesso.
Parâmetros inválidos	*MESSAGE_INVREQ*	*0x*FC	Pedido com parâmetros inválidos.
Mensagem inválida	*MESSAGE_ERROR*	*0x*FB	Mensagem de estrutura inválida (BCC incorrecto).

Tabela 5.3 - Tipos de mensagens USART (primeiro byte de cada frame).

A Tabela 5.4 descreve a estrutura do byte *OpCode*, bem como todos os seus valores possíveis. Os três bytes de dados (*Joint* 1/2/3) dizem respeito à operação indicada neste byte.



Byte OpCode

SCU id	***Operação***	***Parâmetro***
SCU alvo relativo à operação (endereçável a 15 SCU's)	*OP_APPLY_JOINT* (*0b*00) Mensagem de actuação sobre as três componentes do SCU alvo (SCU id).	*PARAM_POSITION* (*0b*00) Posição referência a ser atingida.
		PARAM_VELOCITY (*0b*01) Velocidade média do movimento a efectuar.
		PARAM_SPECIAL (*0b*11) Des/Activação do PWM aplicado aos motores e da filtragem dos valores sensoriais do servo.
	OP_APPLY_CONTROL (*0b*01) Mensagem de actuação sobre as três juntas do SCU alvo com a definição dos parâmetros dos controladores.	*PARAM_KI* (*0b*00) Ganho da componente integral do controlador.
		PARAM_KP (*0b*01) Ganho da componente proporcional do controlo.
		PARAM_K (*0b*10) Ganho do controlador de primeiro nível.
		PARAM_CONTROLON (*0b*11) Tipo de controlo (de primeiro nível) a aplicar na junta.
	OP_READ_SENSORS (*0b*10) Mensagem de leitura dos sensores.	*PARAM_POSITION* (*0b*00) Posição actual de cada junta.
		PARAM_VELOCITY (*0b*01) Velocidade estimada de cada junta.
		PARAM_CURRENT (*0b*10) Corrente drenada por cada servo.
		PARAM_SPECIAL (*0b*11) Saída dos sensores especiais.
	OP_READ_EXTBUFF (*0b*11) Leitura do *buffer* externo do *master*.	*Status* do barramento CAN.

Tabela 5.4 - Campos do pacote *OpCode* nas mensagens CPU→*Master* via USART

De notar que a posição referência e a velocidade média da trajectória depende do controlador de primeiro nível activo (seleccionado através do parâmetro *PARAM_CONTROLON*). Os vários tipos de controlador estão indicados na

Tipo de Controlo sobre as juntas	Designação	PARAM_CONTROLON
Sem Controlo de primeiro nível	*NO_CONTROL*	*0b*00
Controlo de Centro de Pressão	*COP_CONTROL*	*0b*01
Controlo de Inclinação	*INC_CONTROL*	*0b*10
Controlo de velocidade angular	*GIRO_CONTROL*	*0b*11

Tabela 5.5 - Tipo de controlo a seleccionar no campo PARAM_CONTROLON.

Nota-se que a actuação é feita directamente às três juntas/componentes numa única mensagem, mas com a desvantagem de apenas se poder actuar num parâmetro de cada vez (posição final, velocidade média ou um parâmetro do controlador).

5.2.2. Comunicação *Master* → CPU

Na resposta, o *master* responde com uma mensagem de 7 bytes, cuja estrutura é a seguinte:

- *SOF*: indicação da mensagem;
- *OpCode*: *opcode* utilizado pela mensagem original PC→*Master*;
- Data 1-4: dados sensoriais no caso de um pedido de consulta sensorial;
- BCC: Validação da mensagem.

byte 1	*byte 2*	*byte 3*	*byte 4*	*byte 5*	*byte 6*	*byte 7*
*0x*FD	OpCode	Data 1	Data 2	Data 3	Data 4	BCC

Formato geral de uma mensagem de resposta Master→CPU.

Esta estrutura é geral e pode assumir várias formas de acordo com a operação envolvida:

- No caso de uma operação de actuação (*OP_APPLY_**), os bytes 2-5 possuem o mesmo valor que a mensagem original, com o sexto byte nulo;
- Numa leitura sensorial (*OP_READ_SENSORS*), o byte *OpCode* é igual ao da mensagem original, com os bytes *Data* 1-4 contendo a informação sensorial pedida. Se os dados sensoriais concernem aos servomotores, três bytes são utilizados para conter a informação relativa a cada um deles, e o último é utilizado para transmitir informação de *status* do SCU em causa. Se concernem aos sensores especiais, são utilizados todos os quatro bytes para conter a informação de um dos conjuntos dos sensores – sensores de força (de um pé), inclinómetros ou giroscópios.

- Se for solicitada a leitura do *buffer* externo do *master* (*OP_READ_EXTBUFF*), o estado do barramento CAN (percepcionado pelo *master*) é devolvido.

As mensagens de resposta podem ter diferentes formatos, apresentados em seguida (== significa que o byte é o mesmo do da mensagem de solicitação):

byte 1	*byte 2*	*byte 3*	*byte 4*	*byte 5*	*byte 6*	*byte 7*
SOF: *0x*FD	==	==	==	==	0	BCC

Mensagem de actuação aplicada com sucesso

byte 1	*byte 2*	*byte 3*	*byte 4*	*byte 5*	*byte 6*	*byte 7*
SOF: *0x*FD	==	Joint 1	Joint 2	Joint 3	Status	BCC

Mensagem de leitura sensorial das juntas

byte 1	*byte 2*	*byte 3*	*byte 4*	*byte 5*	*byte 6*	*byte 7*
SOF: *0x*FD	==	Data 1	Data 2	Data 3	Data 4	BCC

Mensagem de leitura dos sensores especiais

byte 1	*byte 2*	*byte 3*	*byte 4*	*byte 5*	*byte 6*	*byte 7*
SOF: *0x*FD	==	Error Code	# TX errors	# RX errors	0	BCC

Mensagem de leitura do buffer externo do Master (status do barramento CAN)

bit 7	*6*	*5*	*4*	*3*	*2*	*1*	*bit 0*
PWM	Calib	Deadline	FinAll	FinOne	FinServo3	FinServo2	FinServo1

Conteúdo do Byte de Status nas mensagens de leitura sensorial das juntas

Bit	***Campo***	***Descrição***
7	*PWM*	Activo se todos os motores possuem o PWM ligado.
6	*Calib*	Calibração dinâmica da posição dos servos activada.
5	*Deadline*	Ocorrência de um erro de violação da largura de banda disponível.
4	*FinAll*	Todos as juntas terminaram a trajectória.
3	*FinOne*	Pelo menos uma das juntas terminou a trajectória.
2	*FinServo3*	A junta 3 terminou a trajectória.
1	*FinServo2*	A junta 2 terminou a trajectória.
0	*FinServo1*	A junta 1 terminou a trajectória.

Tabela 5.6 - Significado dos bits presentes no byte de status nas mensagens de leitura sensorial das juntas.

De notar que, na leitura sensorial das juntas, na mesma mensagem são transportados os valores dos três servos. No entanto, apenas um parâmetro pode ser lido – posição, velocidade média ou corrente.

Quanto à leitura dos sensores especiais, note-se também o limite de quatro valores. Daí a limitação imposta anteriormente de apenas usar quatro linhas analógicas dedicadas a este género de sensores: ou o conjunto dos sensores de força, ou inclinómetros, ou giroscópios.

No entanto, podem ocorrer situações anómalas na recepção do comando por parte do *Master*, podendo-se discernir duas situações possíveis:

- Mensagem de pedido inválido: os parâmetros solicitados pelo CPU não fazem sentido (ex.: SCU alvo não existente). Neste caso uma mensagem de SOF de código *MESSAGE_INVREQ* (*0x*FC), com todos os restantes bytes iguais à da mensagem original, é retornada ao CPU.
- Mensagem corrompida: o código BCC não está de acordo com a estrutura da mensagem. Neste caso uma mensagem de SOF igual a *MESSAGE_INVALID* (0xFB) é retornado com os restantes bytes iguais aos da mensagem recebida.

byte 1	*byte 2*	*byte 3*	*byte 4*	*byte 5*	*byte 6*	*byte 7*
SOF: *0x*FC	==	==	==	==	0	BCC

Mensagem sinalizadora de um pedido inválido.

byte 1	*byte 2*	*byte 3*	*byte 4*	*byte 5*	*byte 6*	*byte 7*
SOF: *0x*FB	==	==	==	==	0	BCC

Mensagem sinalizadora de inconsistência no byte BCC.

Teste das comunicações RS-232 entre CPU e Master:

Para confirmação da correcta comunicação entre o CPU e o *Master* pode ser enviada uma mensagem de teste igualmente de 6 bytes com o SOF como MESSAGE_TEST (0xFE).

byte 1	*byte 2*	*byte 3*	*byte 4*	*byte 5*	*byte 6*
SOF: *0x*FE	X2	X3	X4	X5	BCC

Mensagem USART CPU→Master de teste.

Os bytes 2 a 5 podem possuir qualquer valor, desde que o byte BCC esteja de acordo com eles. Por sua vez, o *master* deverá responder com uma mensagem de 7 bytes possuindo os mesmos valores que a primeira:

byte 1	*byte 2*	*byte 3*	*byte 4*	*byte 5*	*byte 6*	*byte 7*
SOF: *0x*FE	X2	X3	X4	X5	0	BCC

Resposta Master→CPU a uma mensagem de teste.

5.3. Device drivers e funções para a comunicação

5.3.1. Funções de comunicação do CPU

Para a utilização da porta série em Linux, foram escritos funções de comunicação na forma de

funções e classes C/C++, para se poder comunicar com o master.

Ficheiro	*Função*	*Descrição*
SerialPort.cpp / SerialPort.h	`int initcom(void);`	Inicializacao da porta série
SerialPort.cpp / SerialPort.h	`int killcom(void);`	Fecho da porta série
SerialPort.cpp / SerialPort.h	`int writecom(unsigned char msg);`	Escrever na porta série
SerialPort.cpp / SerialPort.h	`int readcom(unsigned char *msg);`	Ler da porta série
SerialPort.cpp / SerialPort.h	`int resetcom(void);`	Limpar os buffers da porta

Tabela 5.7 - Lista de device drivers da unidade principal.

initcom

Estabelecimento de uma nova ligação via RS-232.

```
int initcom(void);

 Entradas:
  gate          -> String da pota ("/dev/ttyUSB0")
 Saídas:
  handler       -> ID da linha série
                -> Código de erro
                -> String descritiva do erro
```

killcom

Término de uma ligação RS-232 existente.

```
int killcom(void);

 Entradas:
  handler       -> ID da linha série
 Saídas:
                -> Código de erro
                -> String descritiva do erro
```

writecom

Escrever na porta serie

```
int writecom(unsigned char msg);

 Entradas:
  msg           -> Mensagem a escrever na porta serie de 8 bits
 Saídas:
                -> Código de erro
                -> String descritiva do erro
```

readcom

Leitura de uma mensagem da porta serie

```
int readcom(unsigned char *msg);

 Entradas:

 Saídas:
  * msg          -> Endereço para a mensagem lida
                 -> Código de erro
                 -> String descritiva do erro
```

resetcom

Limpar os buffers da porta série

```
int resetcom(void);

 Entradas:

 Saídas:
                 -> Código de erro
                 -> String descritiva do erro
```

5.3.2. Funções de actuação e sensoriais

Para automatizar o processo de leitura/escrita dos dados sensoriais/actuadores, foram escritas funções e classes em C/C++, sendo estas de nível superior, e mais fácil compreensão para o utilizador. São elas:

Ficheiro	*Função*	*Descrição*
message.cpp / message.h	`int testcom(void);`	Teste da porta série
message.cpp / message.h	`int readcanstate (void);`	Leitura do estado do barramento CAN entre *slaves*
message.cpp / message.h	`int readjoint (signed char *joint, unsigned char scu_id,unsigned char param);`	Leitura de um parâmetro sensorial dos servos de um SCU
message.cpp / message.h	`int readspecial (signed char *sensor, unsigned char scu_id);`	Leitura dos sensores especiais de um SCU
message.cpp / message.h	`int readorient(signed char *sensor);`	Leitura dos sensores de orientação (inclinómetro e giroscópio)
message.cpp / message.h	`int applyjoint (unsigned char`	Aplicação de uma ordem de posição ou

	scu_id,unsigned char param,signed char *servos);	velocidade a cada motor de uma junta
message.cpp / message.h	int applycontrol (unsigned char scu_id,unsigned char param,signed char *servos);	Ajuste dos parâmetros do controlador PID para o posicionamento dos servo, Jacobiano e Proporcional

Tabela 5.8 - Funções de actuação e leitura

testcom

Pedido de envio de uma sequencia de teste.

```
int testcom(void);

 Entradas:

 Saídas:
                -> Código de erro
                -> String descritiva do erro
```

readcanstate

Leitura do estado do barramento CAN.

```
int readcanstate (void);

 Entradas:

 Saídas:
                -> [estado de erro, #erros de transmissão, #erros de recepção]
                -> Mensagen de baixo nivel recebida
                -> Código de erro
                -> String descritiva do erro
```

readjoint

Leitura de um parâmetro sensorial de um SCU.

```
int readjoint(signed char *joint, unsigned char scu_id, unsigned char
param);

 Entradas:
  suc_id        -> Indentificador do SCU alvo
  param         -> Parametro a ler (Tabela 5.9)
 Saídas:
   *joint       -> Parâmetros sensoriais relativos a cada sensor
   state        -> Bits de estado dos servos (Tabela 5.10)
                -> Mensagen de baixo nivel recebida
                -> Código de erro
                -> String descritiva do erro
```

Campo param		*Descrição*	*Campo servos*	*Unidades*
PARAM_POSITION	0	Posição angular de cada junta.	[pos_1, pos_2, pos_3]	Graus
PARAM_VELOCITY	1	Velocidade estimada de cada junta.	[vel_1, vel_2, vel_3]	Graus/100ms
PARAM_CURRENT	2	Corrente drenada por cada servo.	[$curr_1$, $curr_2$, $curr_3$]	% do T de PWM

Tabela 5.9 - Valores possíveis do parâmetro param na função readjoint.

Status devolvido:

Valor 1	*2*	*3*	*4*	*5*	*6*	*7*	*Valor 8*
PWM	Calib	Deadline	FinAll	FinOne	FinServo3	FinServo2	FinServo1

Conteúdo do vector Status retornado pela função readjoint (ver Tabela 5.6)

Elemento	*Campo*	*Descrição*
1	*PWM*	Activo se todos os motores possuem o PWM ligado.
2	*Calib*	Calibração dinâmica da posição dos servos activada.
3	*Deadline*	Ocorrência de um erro de violação da largura de banda disponível.
4	*FinAll*	Todos as juntas terminaram a trajectória.
5	*FinOne*	Pelo menos uma das juntas terminou a trajectória.
6	*FinServo3*	A junta 3 terminou a trajectória.
7	*FinServo2*	A junta 2 terminou a trajectória.
8	*FinServo1*	A junta 1 terminou a trajectória.

Tabela 5.10 - Valores presentes no vector *status* retornado pela função *readjoint*.

readspecial

Leitura dos sensores especiais de um SCU.

```
int readspecial (signed char *sensor, unsigned char scu_id);

 Entradas:
  suc_id         -> Indentificador do SCU alvo
 Saídas:
   Sensor        -> Valores dos sensores especiais (4 valores)
                 -> Mensagen de baixo nivel recebida
                 -> Código de erro
                 -> String descritiva do erro
```

readorient

Leitura dos sensores de orientação.

```
int readorient(signed char *sensor);
```

```
Entradas:

Saídas:
  Sensor          -> Valores dos sensores de orientação (4 valores)
                  -> Mensagen de baixo nivel recebida
                  -> Código de erro
                  -> String descritiva do erro
```

applyjoint

Aplicação de uma ordem de actuação a cada componente de um SCU.

```
int applyjoint (unsigned char scu_id,unsigned char param,signed char
*servos);

 Entradas:
  suc_id          -> Indentificador do SCU alvo
  param           -> Parametro a ler (Tabela 5.11)
  *servos         -> Dados a aplicar
 Saídas:
                  -> Mensagen de baixo nivel recebida
                  -> Código de erro
                  -> String descritiva do erro
```

Campo param		*Descrição*	*Campo servos*	*Unidades*
PARAM_POSITION	0	Posição referência a ser atingida.	[ref_1, ref_2, ref_3]	Graus *ou* outro
PARAM_VELOCITY	1	Duração do movimento a efectuar.	[vel_1, vel_2, vel_3]	Ciclos de 20ms
PARAM_SPECIAL	3	Activação/desactivação do PWM aplicado aos motores.	[0/1, 0, 0]	(Valor booleano)

Tabela 5.11 - Valores possíveis do parâmetro param na função applyjoint.

applycontrol

Selecção do controlador de primeiro nível, e ajuste dos seus parâmetros bem como dos do controlador local.

```
int applycontrol (unsigned char scu_id,unsigned char param,signed char
*servos);

 Entradas:
  suc_id          -> Indentificador do SCU alvo
  param           -> Parametro a ler (Tabela 5.12)
  *servos         -> Dados a aplicar
 Saídas:
                  -> Mensagen de baixo nivel recebida
                  -> Código de erro
                  -> String descritiva do erro
```

Campo *param*		*Descrição*	Campo *servos*
PARAM_KI	0	Ganho da componente integral (*Ki*) do controlador local.	[Ki_1, Ki_2, Ki_3]
PARAM_KP	1	Ganho da componente proporcional (*Kp*) do controlo local.	[Kp_1, Kp_2, Kp_3]
PARAM_K	2	Ganho (*K*) do controlador de primeiro nível.	[K_1, K_2, K_3]
PARAM_CONTROLON	3	Tipo de controlo de primeiro nível (*Type*) a aplicar em cada junta (Tabela 5.13).	[$Type_1$, $Type_2$, $Type_3$]

Tabela 5.12 - Valores possíveis do parâmetro *param* na função *applycontrol*.

Tipo de Controlo (*Type*)		*Descrição*
NO_CONTROL	0	Sem Controlo de primeiro nível
COP_CONTROL	1	Controlo de Centro de Pressão
INC_CONTROL	2	Controlo de Inclinação
GIRO_CONTROL	3	Controlo de velocidade angular

Tabela 5.13 - Tipos de controladores de primeiro nível (a aplicar com o parâmetro PARAM_CONTROLON na função *applycontrol*).

5.3.3. Utilização das funções

Estabelecimento/Término de uma Linha de Comunicações

Para criar uma linha de comunicações série é utilizada a função *initcom*, com recurso a uma classe *Message* que contém todas a funções em C++. Será aberto um canal utilizando a porta série a um ritmo de transmissão de 115k2 bps.

```
Message port;
port.initcom();
```

Para verificar se as comunicações estão a decorrer sem problemas nenhuns, pode-se utilizar a função *testcom*.

```
port.testcom();
```

Para fechar o mesmo canal deve-se, utilizar a função *killcom*, função "oposta" à apresentada anteriormente:

```
port.killcom();
```

Activação/Desactivação dos Sinais de PWM

Para controlar a presença (ou não) dos sinais de PWM sobre os servomotores é utilizada a função de actuação *applyjoint* com o valor 3 no parâmetro *param* (PARAM_SPECIAL), tal como descrito na Tabela 5.11. Por sua vez o parâmetro *servos* será composto por um *array* de três elementos, cujo primeiro elemento corresponderá a um valor booleano que indica se o *slave* de endereço *scu_id* deverá ter os seus sinais de PWM activos ou não. A título de exemplo, para activar os PWMs do *slave* de endereço 1, seria enviado a seguinte função:

```
signed char val[3];
val[0]=1;
signed char scu_id=1;

port.applyjoint((unsigned char) scu_id ,(unsigned char)3,val);
```

Leituras Sensoriais

Para ler os diversos sensores são utilizadas as funções *readjoint* e *readspecial*, conforme se pretenda ler os sensores associados aos servomotores ou os sensores ligados via *piggy-back* sobre a placa controladora respectiva (Tabela 5.14).

Tipo de sensores	***Função associada***
Sensores associados aos servomotores	`int readjoint (signed char *joint, unsigned char scu_id,unsigned char param);`
Sensores via *piggy-back*	`int readspecial (signed char *sensor, unsigned char scu_id);`

Tabela 5.14 - Funções para leitura sensorial.

Com a função *readjoint* (parâmetros sensoriais dos servomotores) é possível ler a posição angular, a velocidade estimada e a corrente drenada que cada servo possui, através do parâmetro *param* (ver Tabela 5.9). Um vector de três elementos (*sensors*) é retornado com o resultado para cada um deles, tal como um vector de 8 elementos descritivos do estado do SCU consultado (a estrutura e o significado dos seus valores pode ser consultado na Tabela 5.10).

Parâmetro a obter	***Syntax da função***
Posição angular	`int readjoint (signed char *joint, unsigned char scu_id,(unsigned char) 0);`
Velocidade angular estimada	`int readjoint (signed char *joint, unsigned char scu_id,(unsigned char) 1);`
Corrente consumida	`int readjoint (signed char *joint, unsigned char scu_id,(unsigned char) 2);`

Tabela 5.15 - Sintaxe da função *readjoint* na leitura dos diversos parâmetros sensoriais.

Através da função *readspecial* é possível obter os ditos dados sensoriais "especiais", ou seja, as saídas provenientes dos sensores conectados via *piggy-back* à unidade *slave*. Eles são:

- Quatro extensómetros localizados em cada pé, que medem a força exercida (sensores de força);
- Acelerómetros/Inclinómetros que medem a inclinação do robô nas componentes *x* e *y*;
- Giroscópios para medição da velocidade angular sobre as três componentes *x*, *y* e *z*.

Qualquer destes conjuntos de sensores pode ser ligado às entradas analógicas (num máximo de quatro) fornecidas na interface *piggy-back*, sendo devolvidas à unidade central as suas quatro saídas digitalizadas em torno do valor 128, utilizando a gama de 0 a 255. O vector *sensor* devolvido por esta função contém estes quatro valores:

```
int readspecial (signed char *sensor, unsigned char scu_id);
```

Como exemplos temos a leitura sensorial da posição angular e dos sensores "especiais" do SCU de endereço 1:

```
signed char joint[4];
signed char scu_id=1;

port.readjoint (joint , (unsigned char) scu_id,(unsigned char) 0);

      joint=[1 0 -69]
      state=0x02

signed char sensor[4];
signed char scu_id=1;

port.readspecial (*sensor, (unsigned char) scu_id);

      sensor=[128 129 126 130]
```

Se, por qualquer motivo, os valores retornados forem inconsistentes, pode sempre ser feita a recalibração dos sensores. Para tal é executado o seguinte procedimento:

6. Desligar os PWMs

```
signed char val[3]={0, 0, 0};
unsigned char scu_id=1;
port.applyjoint((char) scu_id ,(unsigned char)3,val);
```

7. Reactivar os PWMs

```
signed char val[3];
val[0]=1;
unsigned char scu_id=1;
port.applyjoint((char) scu_id ,(unsigned char)3,val);
```

Controlador de Primeiro Nível

Cada unidade *slave*, possui dois controladores em cascata:

8. Controlador de primeiro nível, responsável por realizar trajectórias de vários parâmetros de actuação diferentes como, por exemplo, posição angular ou centro de pressão;
9. Controlador local, responsável pela garantia da aplicação da posição angular solicitada.

Estão implementados quatro tipos de controladores de primeiro nível, o que se pode constatar através da Tabela 5.16:

Tipo de controlador	***Parâmetro de actuação***	***Sensores utilizados***
Sem controlador	Posição angular do servo	Potenciómetro do servomotor
Controlo do Centro de Pressão	Centro de Pressão	4 Extensómetros
Controlo de Inclinação	Inclinação	Acelerómetros/Inclinómetros
Controlo de Velocidade Angular	Velocidade Angular	Giroscópios

Tabela 5.16 - Controladores de primeiro nível (consultar Tabela 5.13).

Se não se seleccionar nenhum controlador, as trajectórias efectuadas terão como base apenas posições angulares referência, relativas aos próprios servomotores. Caso contrário, um dos conjuntos de sensores "especiais" será utilizado, com o controlador a executar a tarefa de calcular a posição angular a atribuir aos motores, de modo a alcançar o valor-referência.

A função *applycontrol* é destinada à escolha deste controlador, com o seu parâmetro *param* igual a 3 (ver Tabela 5.12), e o parâmetro *servos* com a selecção do controlador para cada junta (vector de três elementos) (ver Tabela 5.13):

```
int applycontrol (unsigned char scu_id,unsigned char param,signed char *servos);
```

Por exemplo, para seleccionar o controlador de centro de pressão para as três juntas do SCU 1, é necessário as seguintes funções:

```
for( i=0;i<3;i++)
{
      servos[i]=1;
}
port.applycontrol (1, 3,servos);
```

Além da selecção do controlador, é necessário também configurar o seu ganho para ajustar a sua reactividade a variações do sinal de erro na sua entrada. Este ganho é controlável também através da função *applycontrol*, mas com o parâmetro *param* igual a 2 (Tabela 5.12). O parâmetro *servos* é, assim, usado para a definição do ganho para cada junta. Note que se definir diferentes

controladores para as várias juntas, cada ganho será aplicado ao controlador correspondente.

```
int applycontrol (unsigned char scu_id,2,signed char *servos);
```

Para a definição de um ganho de 30 para as três juntas, deve ser executado o seguinte:

```
for( i=0;i<3;i++)
{
      servos[i]=30;
}
port.applycontrol (1, 3,servos);
```

Controlador Local

O controlador local é responsável por garantir a aplicação da posição angular solicitada pelo controlador de primeiro nível, com base na posição medida através do potenciómetro interno ao servomotor. Caso o controlador de primeiro nível esteja desactivado, a posição de actuação enviada pela unidade central é aplicada directamente neste controlador.

Esta componente apenas se trata de um compensador do tipo PI que, a partir da posição referência e da medida pelo potenciómetro (sinal de erro), calcula a compensação a atribuir ao servo para que o sinal de erro se anule. Para o ajuste deste compensador são usados os ganhos K_I e K_P para as componentes integradora e proporcional, respectivamente. Para a definição destes ganhos é usada igualmente a função *applycontrol* com o parâmetro *param* igual a 0 para o ajuste de K_I e 1 para o K_P (Tabela 5.12). O parâmetro *servos* possuirá o valor numérico do ganho a atribuir a cada junta (vector triplo).

Ganho a controlar	***Syntax da função***
K_I	`int applycontrol (unsigned char scu_id,0,signed char *servos);`
K_P	`int applycontrol (unsigned char scu_id,1,signed char *servos);`

Tabela 5.17 - Sintaxe da função *applycontrol* na definição dos ganhos KI e KP do controlador local.

Para a definição de um K_I igual a 5, e um K_P igual a 20 às três juntas do SCU 1, usar-se-á a sintaxe:

```
for( i=0;i<3;i++)
{
      servos[i]=5;
}
port.applycontrol (1, 0,servos);

for( i=0;i<3;i++)
{
      servos[i]=30;
}
port.applycontrol (1, 1,servos);
```

Caso se atribua K_I e K_P nulos, o controlador local será desactivado e as posições angulares retornadas pelo controlador de primeiro nível são aplicadas directamente sobre os servomotores:

```
for( i=0;i<3;i++)
{
      servos[i]=0;
}

port.applycontrol (scu_id, 0,servos);
port.applycontrol (scu_id, 1,servos);
```

Execução de Trajectórias

Para a definição dos *setpoints* para uma trajectória, a função de actuação *applyjoint* é utilizada, com o parâmetro *param* a 0, para a atribuição da posição referência, e a 1 para a velocidade.

```
int applyjoint (unsigned char scu_id,unsigned char param,signed char
*servos);
```

Parâmetro a controlar	***Syntax da função***
Referência	`int applyjoint (unsigned char scu_id,0,signed char *servos);`
Velocidade	`int applyjoint (unsigned char scu_id,1,signed char *servos);`

Tabela 5.18 - Sintaxe da função *applyjoint* na definição da posição-referência e da velocidade.

A velocidade é indicada através da duração da trajectória em ciclos de 20ms (período de PWM), pelo que, se se pretender uma duração de 2s, o valor 100 deve ser dado. No que respeita à posição referência, este parâmetro depende do controlador de primeiro nível em acção. A Tabela 5.16 refere a relação entre o parâmetro de actuação dado nesta função e o controlador activo, por isso, antes da definição deste valor é necessário seleccionar previamente o controlador através da função *applycontrol*, e só depois modificar este parâmetro usando a função *applyjoint*. O mesmo se aplica à velocidade, podendo definir-se valores diferentes de posição e velocidade para cada controlador – a mudança do controlador carrega automaticamente os valores da sua utilização anterior.

Para a definição de uma trajectória de inclinação à primeira junta do *slave* 1, até +40º segundo uma duração de 2s, deve efectuar-se o seguinte conjunto de passos:

10. Seleccionar o controlador de inclinação:

```
for( i=0;i<3;i++)
{
      servos[i]=0;
}
port.applycontrol (1, 0,servos);
```

11. Verificar se o ganho do controlador é nulo:

```
for( i=0;i<3;i++)
{
      servos[i]=0;
}
port.applycontrol (1, 2,servos);
```

12. Definir a inclinação de referência final:

```
servos[0]=40;
servos[1]=0;
servos[2]=0;
port.applyjoint (1, 0,servos);
```

13. Definir a duração da trajectória:

```
servos[0]=100;
servos[1]=0;
servos[2]=0;
port.applyjoint (1, 1,servos);
```

14. Iniciar o movimento ao atribuir um ganho não nulo ao controlador de inclinação:

```
servos[0]=100;
servos[1]=0;
servos[2]=0;
port.applycontrol (1, 2,servos);
```

Note-se que, primeiro, é feita a selecção do controlador de interesse pois, desta forma, a definição do seu ganho e as instruções de actuação serão agulhadas para o parâmetro de controlo associado. Uma estratégia usada neste exemplo, para garantir que a trajectória não é iniciada durante a definição da inclinação final e da velocidade, é a inicialização prévia do ganho do controlador a zero para forçá-lo ao "estacionamento". Só após a atribuição de um ganho válido ele retoma o seu funcionamento normal.

6. Sistema de visão

Resumo:

Este capítulo descreve como é implementado o sistema de visão na plataforma humanóide. É descrito o processamento de imagem e o controlo da estrutura do tronco, a ser efectuado pela unidade central de processamento. Para o processamento de imagem utilizou-se a biblioteca OpenCv, pois esta é das mais desenvolvidas para o pretendido. Para o controlo da plataforma humanóide, recorreu-se às funções de actuação e sensoriais descritas no capítulo anterior.

6.1. Introdução

Como a grande ambição deste projecto é a participação no concurso *RoboCup*, na modalidade *The Penalty Kick*, o robô vai ter que visualizar a bola e a baliza, para poder enquadrar-se com a baliza e chutar a bola. Para a realização desta tarefa, é necessária uma câmara para processamento de imagem, de modo a o robô possa reconhecer a bola e a baliza e chutar a bola na direcção certa. Para o processamento de imagem é utilizada a biblioteca OpenCv pois esta está muito completa e desenvolvida para a aquisição e processamento de imagens.

Para o controlo da plataforma, foram utilizados os *Devices Drives* implementados no capítulo anterior. Adicionalmente, foi necessário implementar algoritmos básicos de controlo, pois o tempo para processamento de imagem e controlo é muito curto, e a unidade central de processamento tem recursos limitados comparado com um computador normal.

6.2. Câmara digital Unibrain FireWire

Para a aquisição de imagem é utilizada uma câmara digital FireWire IEEE 1394 de 400 Mbps, visto que, estas câmaras são facilmente configuráveis e utilizáveis em Linux. Com o seu modo simples de utilização e de configuração, estas são extremamente fiáveis e robustas, e ainda são de baixo custo. Para esta câmara, o *Frame Rate* máximo suportado pela câmara à resolução 640x480 pixeis é de 30fps, o que é suficiente para esta aplicação específica.

Fig. 6.1 - Câmara digital Unibrain FireWire

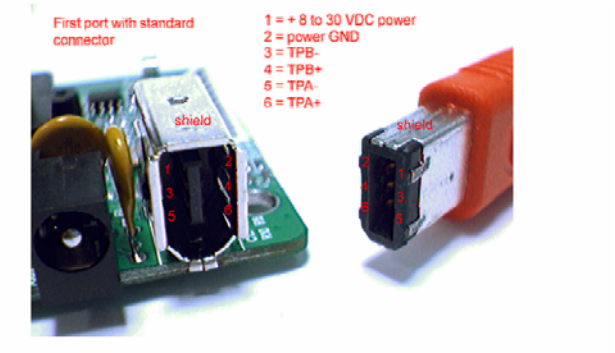


Fig. 6.2 - Conexão FireWire

6.3. Processamento de Imagem

Como já foi referido anteriormente, para o processamento de imagem é utilizada a biblioteca OpenCv. Esta é muito completa e poderosa para captura e processamento de imagens, e as suas funções são simples de implementar. Antes de começar a capturar imagens da câmara com a biblioteca OpenCv, é aconselhável o uso da aplicação *Coriander*, pois este programa está preparado para lidar com as câmaras *FireWire* e verificar se tudo se encontra em correcto funcionamento. No final deste relatório existe um pequeno tutorial de *Coriander*.

Captura de imagem

Na captura de uma imagem é feita a sua aquisição, gravação e, por fim, conversão, de modo a torná-la manipulável.

```
//Captura da imagem do indice da porta vídeo
CvCapture* cvCreateCameraCapture ( int index );

// Gravação da captura realizada
int cvGrabFrame( CvCapture* capture );

// Conversão da capura gravada numa estrutura do tipo IplImage
IplImage* cvRetrieveFrame( CvCapture* capture );
```

Visualização de imagens

Para visualizar as imagens capturadas é necessário criar uma janela onde serão visíveis as referidas imagens.

```
//Criar uma janela com um nome
int cvNamedWindow( const char* name, unsigned long flags );

// Mostrar na janela a imagem
void cvShowImage( const char* name, const CvArr* image );
```

Deste modo é possível visualizar a primeira imagem capturada da câmara, como se pode ver na Fig. 6.3.

Fig. 6.3 - Imagem original capturada

Conversão RGB para HSV

Para se poder criar um filtro por cor é necessário converter a imagem capturada em RGB para HSV. O sistema HSV de cores formadas pelas componentes *Hue* (tonalidade), *Saturation* (Saturação) e *Value* (Valor). O sistema HSV também é conhecido como HSB, *Hue* (Tonalidade), *Saturation* (Saturação) e *Brightness* (Brilho).

Por tonalidade entende-se a cor específica (este sistema abrange todas as cores do espectro, desde o vermelho até o violeta).

A saturação, também pode ser chamado de "pureza". Quanto menor for, mais cinza aparecerá a imagem; quanto maior for, mais clara ou "pura" é a imagem.

O valor, ou brilho, define o brilho da cor, ou seja, a sua luminosidade. Quanto maior este valor, mais clara será a imagem. No extremo máximo, esta tornar-se-á totalmente branca.

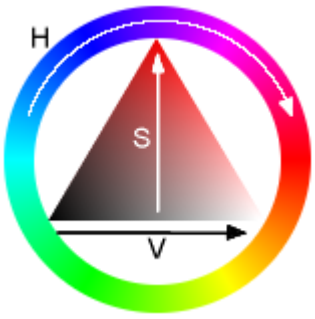


Fig. 6.4 - Sistema HSV

```
//Converter a imagem
void cvCvtColor( const CvArr* src, CvArr* dst, int code );

// Separar as várias componetes HSV
void cvSplit ( const CvArr* src, CvArr* dst1, CvArr* dst2, CvArr* dst3,
CvArr* dst4 );
```

Como se pode verificar pelas Fig. 6.5, a imagem original em RGB, convertida em HSV e separada nas suas componentes.

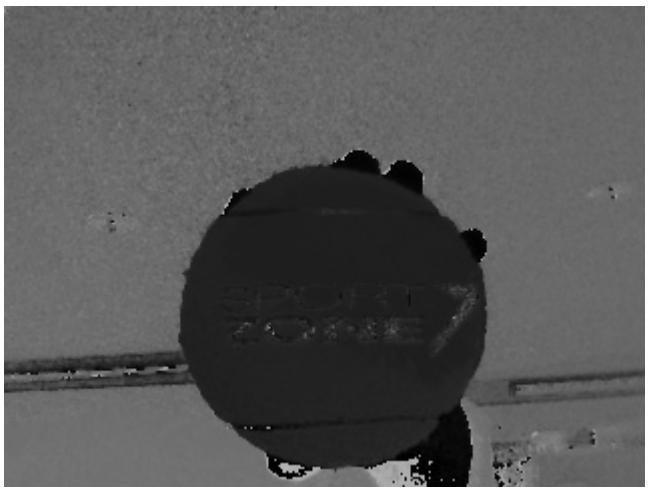


Fig. 6.5 - Imagem H

Fig. 6.6 - Imagem S

Fig. 6.7 - Imagem V

Filtro de cor

Com a imagem convertida em HSV e separada nas várias componentes, é possível filtrar por cor, sendo devolvida uma máscara para filtro. Para isso basta utilizar as seguintes funções:

```
//Comparar com escalar
void cvCmpS( const CvArr* src, double S, CvArr* dst, int cmp_op );

// Comparar dentro de dois limites
void cvInRangeS( const CvArr* src, CvScalar SL, CvScalar SU, CvArr*
dst);
```

Como se pode verificar pelas Fig. 6.10, foi criada uma mascara binária da filtragem nas diversas componentes H, S e V.

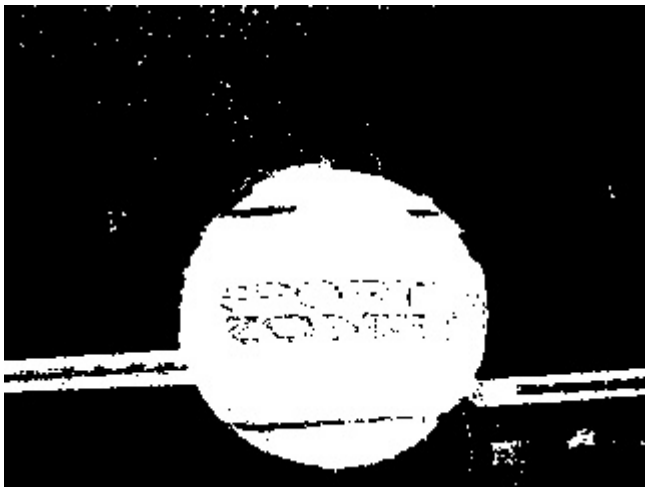


Fig. 6.8 - Imagem filtrada em H

Fig. 6.9 - Imagem filtrada em S

Fig. 6.10 - Imagem filtrada em V

Para ter o filtro completo, basta multiplicar as três máscaras criadas por comparação. Para a multiplicação das máscaras utiliza-se a função seguinte:

```
//Multiplacação binária
void cvAnd( const CvArr* dst, const CvArr* src1, CvArr* src2, const
CvArr* mask=0 );
```

Fig. 6.11 - Multiplicação das três filtragens

Cálculo do centro de massa

Com a imagem filtrada é necessário calcular o seu centro de massa, para saber qual o centro da imagem. Deste modo, posteriormente, pode-se controlar os vários actuadores para que a câmara se centre na imagem.

Para calcular os momentos são utilizadas as seguintes funções:

```
//Cálculo dos Momentos da Imagem
void cvMoments( const CvArr* arr, CvMoments* moments, int isBinary=0 );

//Cálculo dos Momentos de primeira ordem
double cvGetSpatialMoment( CvMoments* moments, int j, int i );
```

Fig. 6.12 - Calculo do centro de massa

Cálculo de círculos na imagem

Uma vez que a imagem que estamos a processar é uma bola (de forma circular) pode-se detectar círculos na imagem e os círculos dão-nos os contornos da bola. Para calcular círculos na imagem são utilizadas as seguintes funções:

```
//Retiro das transições bruscas na imagem
void cvSmooth( const CvArr* src, CvArr* dst,int smoothtype=CV_GAUSSIAN,
int param1=3, int param2=0 );

//Cálculo dos circulos
CvSeq* cvHoughCircles( CvArr* image, void* circle storage, int method,
double dp, double min_dist, double param1=100, double param2=100 );
```

O processamento de imagem anteriormente descrito pode ser verificado na Fig. 6.13, onde está representado o círculo detectado pelas funções anteriores e o centro de massa calculado da imagem filtrada.

Fig. 6.13 - Centro de massa e circulo

Template Match

A técnica de *Template Match* é muito utilizada para encontrar um objecto específico na imagem, no entanto tem a desvantagem de ser sensível à escala. Ainda assim, esta técnica pode ser útil para reconhecer objectos sempre à mesma distância da câmara, por exemplo o pé que está junto da bola, pronto para chutar.

```
//Função de template match
void cvMatchTemplate( const CvArr* I, const CvArr* T, CvArr* result, int
method );
```

Como se pode ser na Fig. 6.15, a técnica de *Template Match* foi aplicada à imagem original, podendo também verificar-se que esta técnica é sensível à escala.

Fig. 6.14 - Template

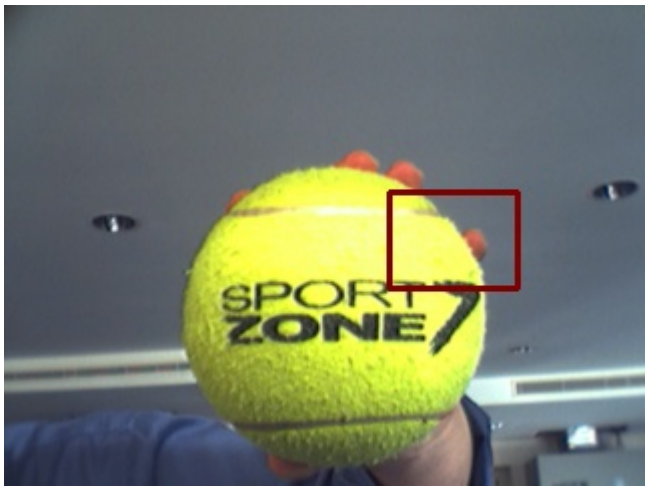


Fig. 6.15 - Template match

6.4. Controlo Pan & Tilt

Para o controlo de Pan e Tilt é utilizada a distância em pixeis do centro da bola ao centro da câmara. Com este erro é possível controlar os actuadores para que o centro da bola coincida com o centro da câmara. No controlo de Pan e Tilt, vão existir situações em que não será possível fazer coincidir o centro da bola com o centro da imagem, isto devido, a existem limitações físicas (das juntas do pescoço). Como se pode verificar pela

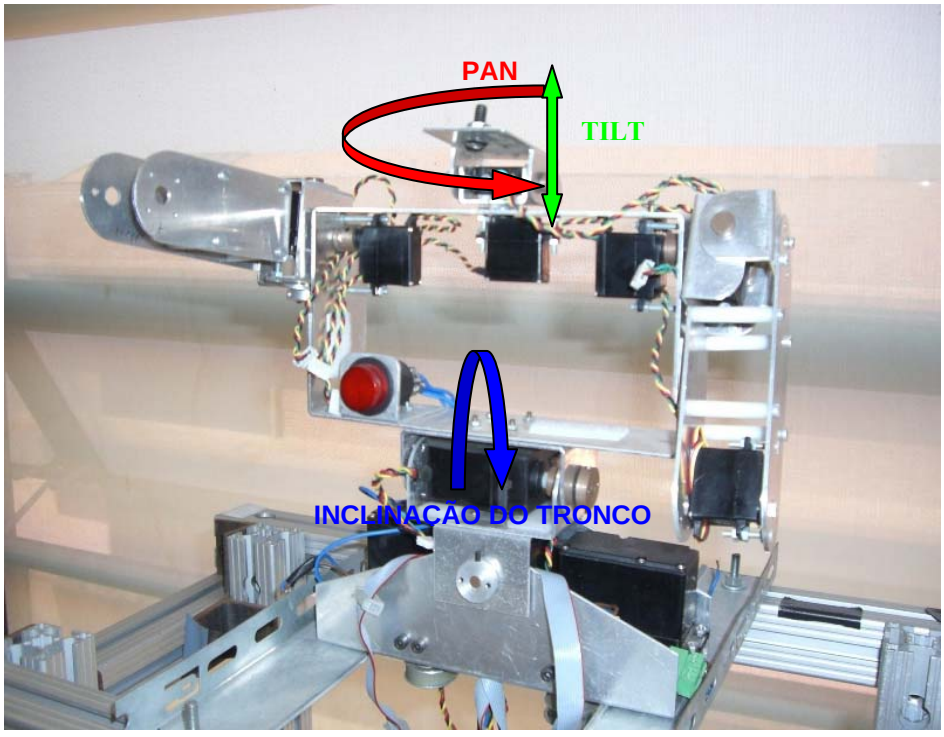


Fig. 6.16 -Unidade de pan &Tilt

Como se pode ver pelo diagrama de blocos da Fig. 6.17, antes do controlo da plataforma é

necessário processar as imagens para se calcular o erro entre a posição central da câmara e a posição central da bola. Com este erro é possível calcular novos ângulos para os actuadores de *Pan e Tilt* e também para um terceiro grau de liberdade que é a inclinação do tronco, de modo a corrigir a posição central da bola com a da câmara. Deste modo, é possível fazer *tracking* de uma bola com três graus de liberdade. O controlador para o *Pan e Tilt* é o seguinte:

$$Pan\_Ang\_New = (Centro\,Bola_{Pan} - Centro\ Câmara_{Pan}) \times K + Pan\_Ang$$
$$Tilt\_Ang\_New = (Centro\,Bola_{Tilt} - Centro\ Câmara_{Tilt}) \times K + Tilt\_Ang$$

É possível fazer o *tracking* da bola em movimento de um modo simples: multiplica-se a diferença de pixeis entre a posição central da bola e a da câmara por um factor (que traduz o número de pixeis na imagem, para o ângulo aproximado a aplicar nos actuadores) e somam-se os ângulos já aplicados anteriormente.

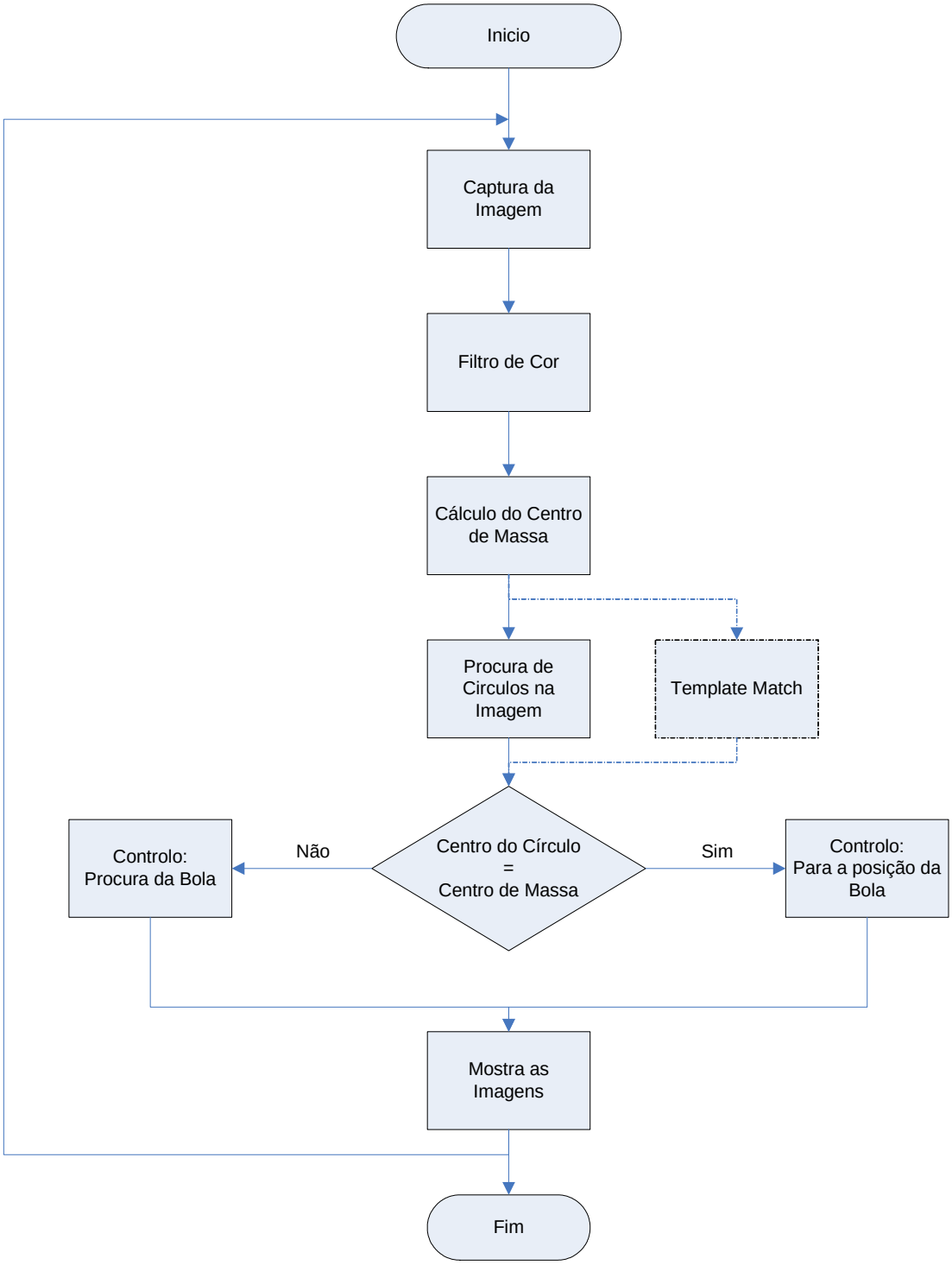


Fig. 6.17 - Estrutura do software do CPU

7. Notas finais

7.1. Agradecimentos

Agradece-se a todas as pessoas ligadas de alguma forma a este projecto, pela sua cooperação empenho e ajuda.

Agradece-se à disponibilidade de recursos e toda a ajuda fornecida por parte do Departamento de Mecânica para a realização deste projecto.

Agradece-se aos colegas de projecto deste ano e de anos anteriores, e a todos os colegas presentes diariamente no LAR, que muito ajudaram a superar dificuldades.

Por fim, agradece-se em especial aos orientadores do projecto, professor Filipe Silva e professor Vítor Santos que tanto trabalharam para que este projecto fosse possível.

7.2. Bibliografia

[1] http://www.compulab.co.il/, Cumpulab, 27/09/2006

[2] http://www.arcom.com/, Arcom, 27/09/2006

[3] http://www.embeddedARM.com/, ARM, 28/09/2006

[4] http://www.via.com.tw/, VIA Technologies, 28/09/2006

[5] http://www.vgapocketpc.com/, Vga pocket PC, 25/09/2006

[6] http://www.ieiworld.com/, IEI technology Corp, 25/09/2006

[7] http://www.pc104.org/technology/plus_info.html, PC 104, 20/09/2006

[8] http://www.humanoidsoccer.org/teams.html, RoboCup

[9] Milton Ruas, "Desenvolvimento de Algoritmos de Controlo para Locomoção de um Robot Humanóide" Relatório final de projecto 2005/06 – Departamento de Electrónica, Telecomunicações e Informática.

[10] Milton Ruas, Filipe M. T. Silva, Vítor M. F. Santos, "Techniques for Velocity and Torque Control of RC Servomotors for a Humanoid Robot", aceite para publicação nos proceedings da 9th International Symposium on Climbing and Walking Robots and Associated Technologies, 11-14 September 2006

[11] Milton Ruas, Filipe M. T. Silva, Vítor M. F. Santos, "A Low-Level Control Architecture for a Humanoid Robot", submetido à International Conference on Humanoid Robots 2006.

[12] Vítor M. F. Santos, Filipe M. T. Silva, "Engineering Solutions to Build an Inexpensive Humanoid Robot Based on a Distributed Control Architecture", proceedings of the IEEE International Conference on Humanoid Robots 2005.

[13] Vítor M. F. Santos, Filipe M. T. Silva, "Development of a Low-Cost Humanoid Robot: Components and Technological Solutions", proceedings of the CLAWAR 2005.

[14] Luís Gomes, Mauro Silva, "Concepção e Desenvolvimento de Unidades de Percepção e Controlo para um Robô Humanóide", Relatório final de projecto 2004/05 –

Departamento de Mecânica

[15] Nuno Beça, Ângelo Cardoso, "Desenvolvimento e Integração das Sub-estruturas Inferior e Superior para a Locomoção de uma Plataforma Humanóide", Relatório final de projecto 2004/05 – Departamento de Mecânica.

Anexo 1: Sistema energético do robô

Resumo:

Na plataforma humanóide ao activar todos os PWM de todos os actuados, verifica-se a ocorrência de picos de corrente que fazem com que a tensão desça a limites críticos. Quando a tensão desce ao limiar mínimo provoca um *reset* dos microcontroladores, o que muitas das vezes transtorna o controlo da plataforma humanóide. Este capítulo vem descrever como solucionar este problema.

Introdução

Na activação dos *PWM* para o controlo da plataforma humanóide, e na actuação dos servos da referida plataforma, verifica-se por vezes um *reset* dos microcontroladores ou um mau funcionamento destes. Verificou-se que quando o *PWM* dos actuadores (três actuadores por *slave* vezes oito *slaves*) era activado, e principalmente quando os actuadores tinham uma carga considerável, os microcontroladores desligavam. Conclui-se que o consumo de corrente era muito elevado e fazia com que a tensão de alimentação descesse a níveis inferiores a 5V, quando se activava os *PWM's*. A solução encontrada para este problema, foi separar a parte de potência da parte electrónica. Isto imlplica alimentar toda a electrónica independente dos actuadores, para que não exista nenhum erro no controlo dos actuadores.

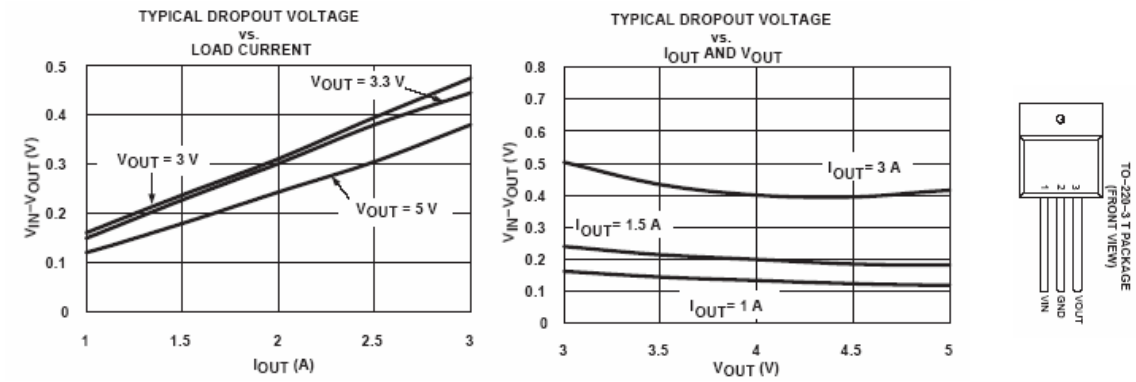
Alimentação da plataforma humanóide

Alimentação dos actuadores

Para os actuadores a alimentação é directa de duas baterias de 7.4V ligadas em paralelo com uma capacidade máxima de 9600mAh.

Regulador linear com baixa queda de tensão UCC283-5 3ª da *Texas Instrument*

Para alimentar toda a electrónica, a unidade central de processamento e os servos de menor dimensão, pois estes só funcionam a 5V, foi calculada uma corrente aproximada de 4ª. Com a utilização de TL7805 eram precisos pelo menos oito reguladores para satisfazer a corrente necessária, e ainda o inconveniente de estes precisarem de pelo menos 1,5V da entrada para a saída para funcionarem correctamente. Numa procura para a solução deste problema, foram encontrados os reguladores *Low Dropout Linear Regulator*, pois estes são a escolha ideal para a alimentação da plataforma, dado que esta utiliza baterias. Como a tensão vai descendo nas baterias até ficarem sem carga, com este regulador podemos tirar o máximo rendimento das baterias, e com uma grande passagem de corrente através do regulador, aproximadamente de 3A.

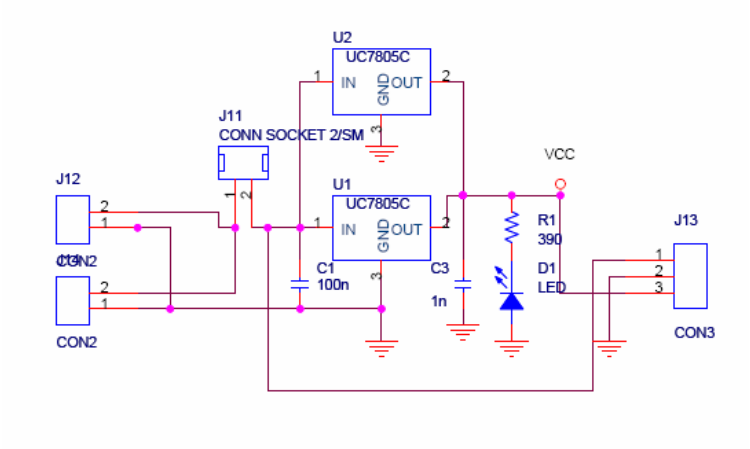


Curva característica do regulador de tensão

Como se pode verificar pela anterior, basta que a tensão de entrada seja superior à de saída em 0,5V, para se ter a capacidade máxima de corrente do regulador. Mesmo que a tensão de estrada seja só superior em 0,1V, o regulador pode pôr na saída a tensão regulada com 1ª de corrente. Tudo indica que a utilização de dois reguladores em paralelo suporte a corrente necessária à plataforma humanóide. Por isso, foi construído uma placa para esses reguladores.

Circuito de alimentação da electrónica

O circuito desenhado para alimentação da electrónica foi o seguinte:



Circuito de alimentação da electronica

Como se pode verificar pelo circuito, existem duas baterias ligadas em paralelo para alimentar a electrónica e, também é possível alimentar os actuadores de maior dimensão através deste circuito, se for necessário. Para este circuito foi feita a placa de circuito impresso desenhada em *OrCAD 10.5* como se pode ver na figura seguinte.

Placa de circuito impresso para a alimentação da electrónica

Alimentação da câmara FireWire

No último mês do projecto, foi encontrado o problema da alimentação da câmara *FireWire*. Esta necessita de 12V para o seu correcto funcionamento, mas actualmente não existe através das baterias gerar os 12V de alimentação da câmara.

Ficam aqui algumas soluções que no próximo ano devem ser debatidas com os orientadores do projecto para a resolução desta implementação. Uma primeira solução é o uso duas baterias em série, pois estas conjugadas fazem um total de 15V e em seguida regulá-las para 12V. No entanto, creio que não será a melhor solução pois é acrescentado muito peso à plataforma e um desperdício de energia só para a alimentação da câmara. Uma solução possível é o uso de uma nova bateria de 12V de pequenas dimensões só para alimentar exclusivamente a câmara. Uma outra solução seria utilizar um circuito de duplicação de tensão e em seguida regular a tenção para os 12V necessários. Normalmente estes circuitos de duplicação de tensão têm problemas com a corrente de saída, pois esta é muito baixa. Uma última solução é o uso de um *step-up switching converter* de 7,4V para 12V ou até mesmo de 5V para 12V. Como por exemplo o ADP1610 da *Analog Devices*, que converte de 2,5V a 5,5V para 12V com 1,2ª de saída. No entanto, o uso de um *step-up switching* implica o uso de uma bobina, o que por vezes não é simples de implementar. Adicionalmente, as fontes *switching* ainda têm o problema de introduzirem variações de tensão de elevada frequência na alimentação.

Uma solução mais prática, é a utilização de uma câmara USB, pois esta já não necessita de alimentação, mas é necessário instalar os *drives* na unidade central de processamento.

Esta implementação será resolvida no próximo ano, partindo-se já das potenciais soluções acima apresentadas.

Anexo 2:Giroscópio

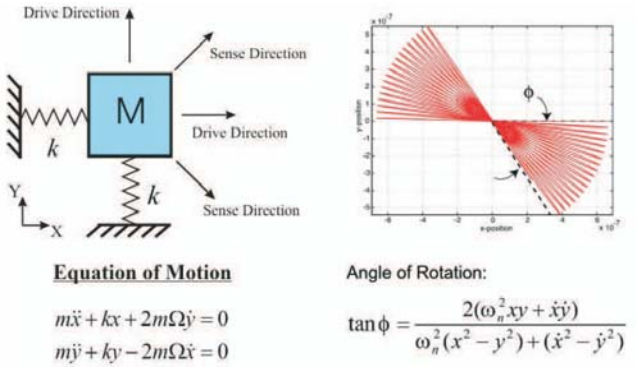
O giroscópio mede a medição de velocidades angulares.

Giroscópio: princípios e modo de funcionamento

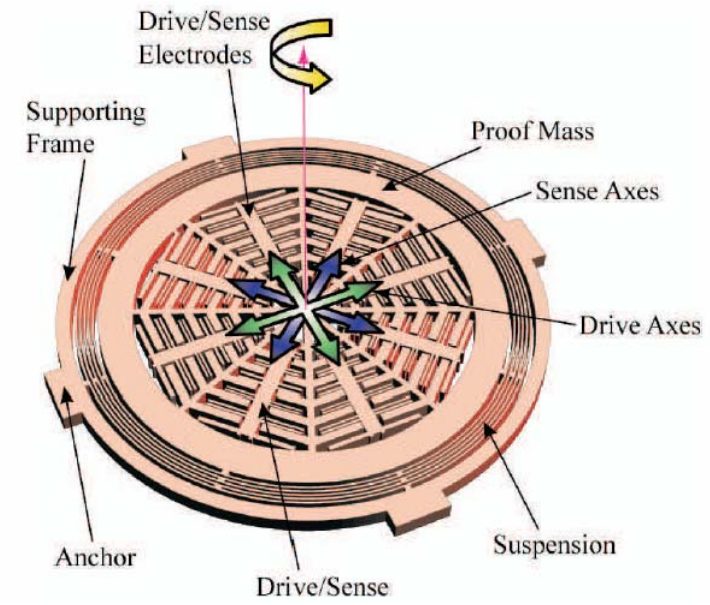
A principal função do giroscópio é a medicação de velocidades angulares. Os giroscópios mecânicos consistem essencialmente numa roda livre, ou em várias rodas a girar em alta velocidade, em qualquer direcção, mas com uma propriedade. Esta propriedade baseia-se na primeira Lei de Newton, o princípio da inércia, pois quando a roda livre está a girar a alta velocidade no seu eixo, esta opõe-se a qualquer tentativa de mudar a direcção do seu eixo original. Deste modo, mantendo a roda livre a girar no seu eixo, quando existe um movimento em qualquer direcção esta fará uma oposição ao movimento, medindo esta oposição ao movimento é possível saber a velocidade angular ou o ângulo de deslocamento.

No caso dos giroscópios electrónicos, normalmente, é usado o princípio de *Coriolis*. O princípio da força de *Coriolis* fictícia, ou inercial, esta só existe num referencial acelerado rotativo, em que o movimento tem de ser circular e submetido a uma aceleração centrípeta. Neste caso, a força de *Coriolis* é similar à força centrífuga, e como se pode verificar, força centrífuga só se manifesta em referenciais em rotação. No entanto, a força de *Coriolis* depende da velocidade do corpo em movimento, e é nula, por definição, no caso de um corpo imóvel num referencial em rotação. A força centrífuga, por sua vez, depende da posição do corpo em relação ao centro de rotação. Pode-se, assim, dizer que a força centrífuga é o componente estático da força inercial manifestando-se no referencial em rotação, enquanto que a força de *Coriolis* é o componente dinâmico.

Deste modo, é possível construir um giroscópio, de muito pequenas dimensões, em que existe uma massa *proof*, não em rotação mas em vibração que vai sofrer igualmente da força de *Corialis*, e esta irá mover-se de forma centrifuga, em relação ao eixo de rotação com a velocidade angular aplicada na massa.



Equação dinâmica do giroscópio



Micro maquinagem do giroscópio

O giroscópio electrónico é feito de silício e consiste numa massa *proof* montada uma estrutura de suspensão. Esta massa é atravessada por uma corrente eléctrica através da estrutura de apoio. Ao fazer com que a corrente atravesse a massa *proof* de uma maneira específica, faz com que esta vibre de maneira previsível. A vibração imposta á massa *proof* tem o mesmo efeito que teria a massa do giroscópio mecânico a girar. Quando o giroscópio sofre uma velocidade angular em torno do seu eixo, as alhetas afastam-se ou aproximam-se devido à força de *Coriolis*, consequentemente, o efeito capacitivo vai variar entre as várias alhetas da estrutura. Desta forma é possível a medição de velocidades angulares.

Giroscópio gyrostar (ENJ-03JA) da Murata

O giroscópio gyrostar da marca *Murata* (ENJ-03JA) segue o mesmo principio de funcionamento descrito na secção anterior, que permite medir a velocidade angular usando o fenómeno da força de *Coriolis*, quando aplicada a um corpo em vibração. Este giroscópio vem já de anos anteriores, que permite medir velocidades até 300 graus/s

Giroscópio gyrostar (ENJ-03JA)

As características deste giroscópio são as seguintes:

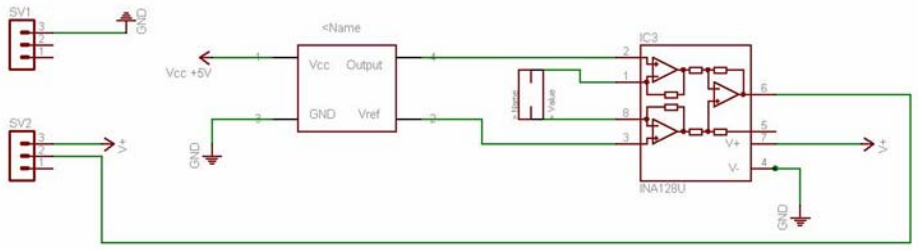
Part Number	ENC−03J
Supply voltage (Vdc)	+2.7 to +5.5
Current consumption (mA max.)	5
Max. angular velocity (˚/s)	±300
Output (at angular velocity=0) (Vdc)	+1.35
Scale factor (mV/˚/s)	0.67
Temp. coefficient of scale factor (%)	±20
Linearity (%FS)	±5
Response (Hz max.)	50
Operating temperature range (˚C)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

Características do giroscópio gyrostar (ENJ-03JA) da Murata

Como se pode verificar pela anterior, este giroscópio tem uma saída analógica e permite medir velocidades até 300 graus/s, o que satisfaz os requisitos da aplicação.

Circuito de acondicionamento de sinal do giroscopio

O circuito desenvolvido nos anos anteriores foi o seguinte:

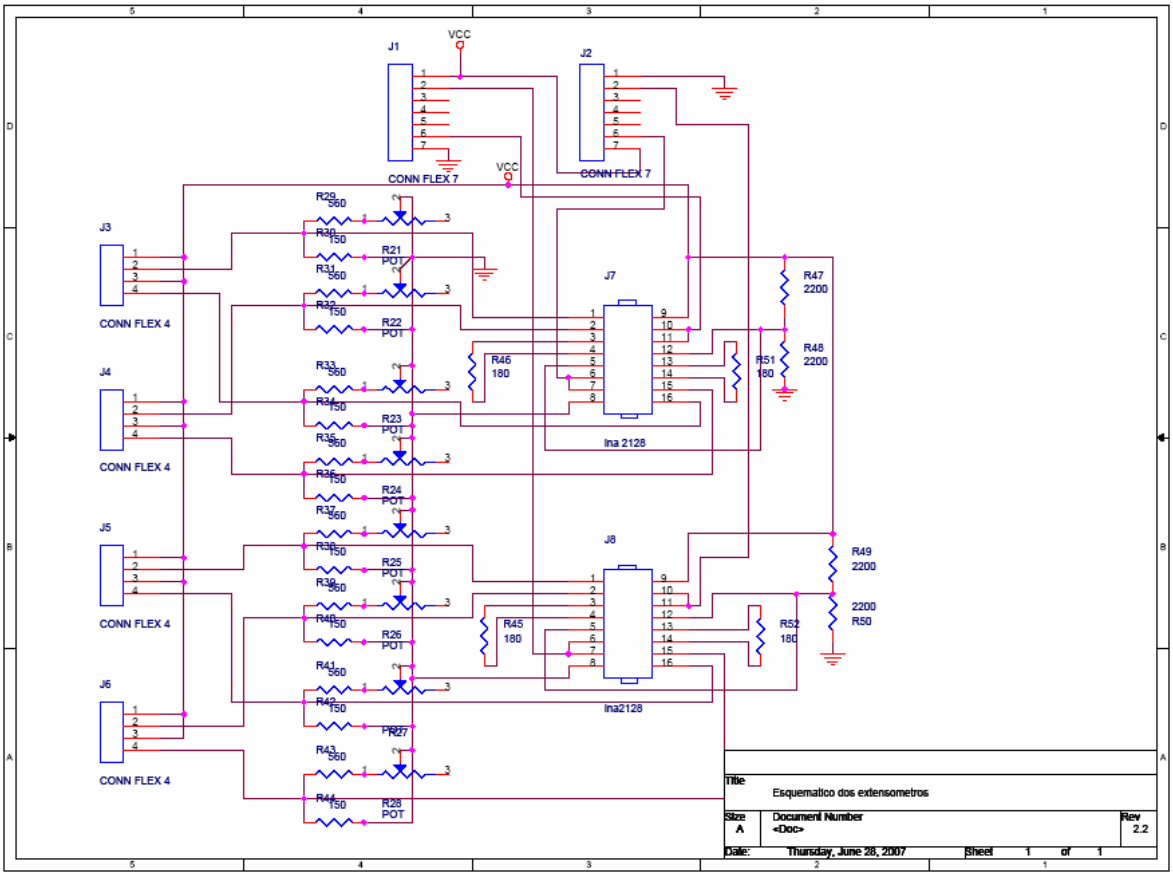


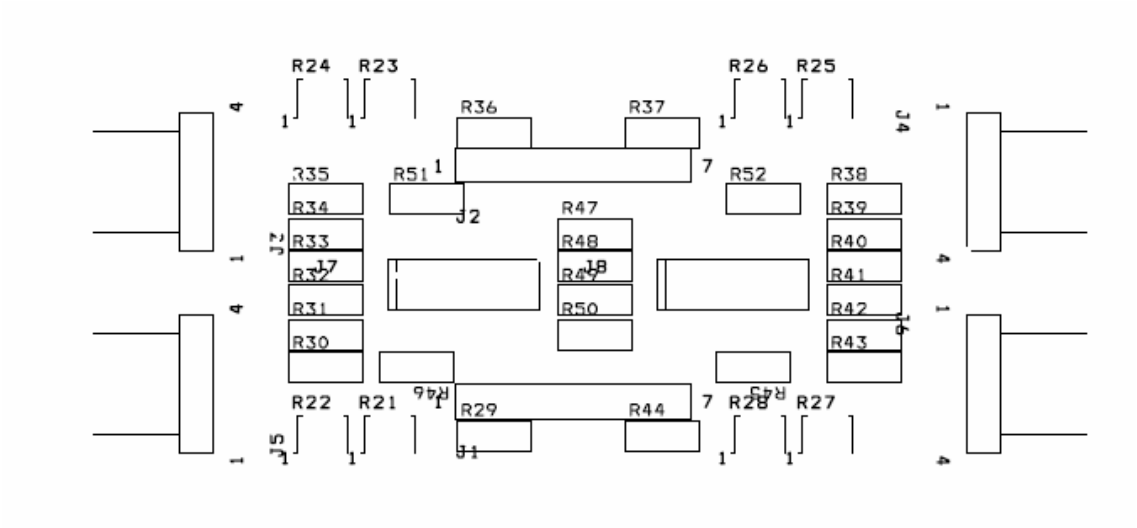
Circuito de acondicionamento de sinal do giroscópio

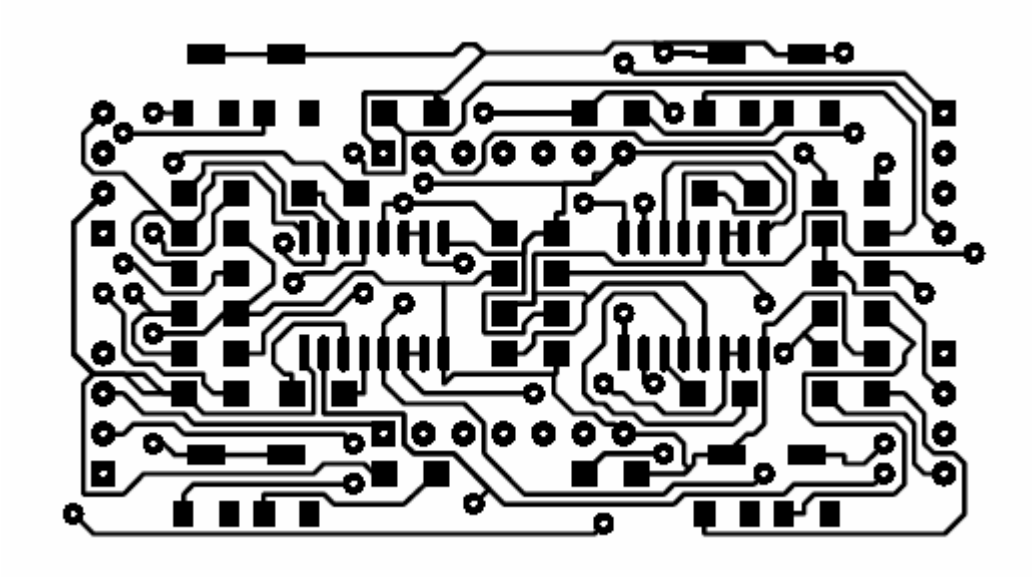
Segundo testes de anos anteriores, em que foi usado um motor rotativo onde era possível colocar no veio o giroscópio, ao rodar o motor foi-se medindo a variação do sinal conforme o aumento e diminuição da velocidade angular, o sinal aumentava e diminuía conforme a velocidade imposta. No presente ano, foi feito um teste aos giroscópios mas estes estavam avariados. Dada esta situação, procurou-se um novo giroscópio e encontrou-se o ADXRS300ABG da *Analog Devices* contudo, não foi possível comprar este giroscópio.

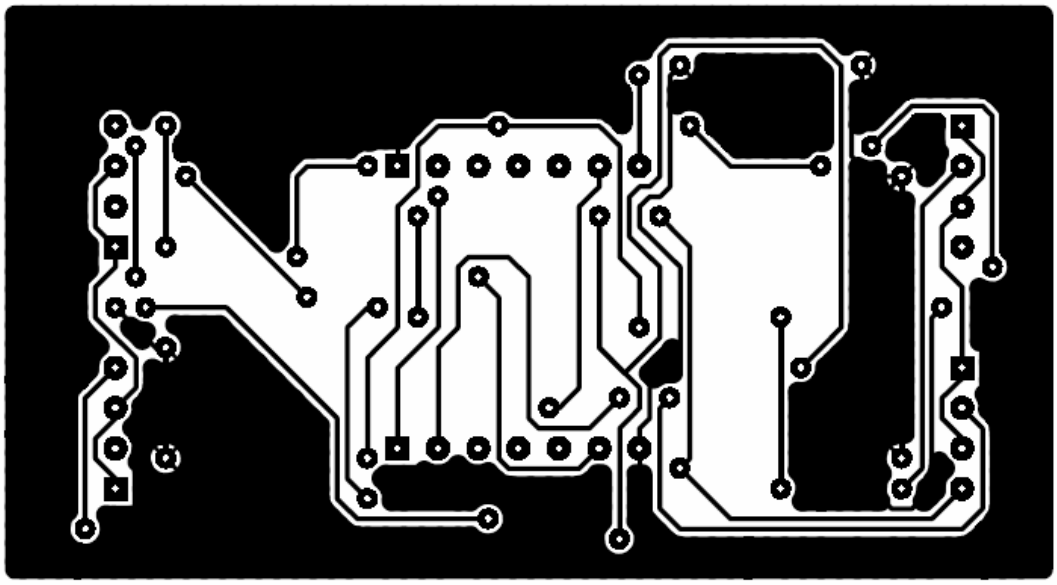
Anexo 3: Esquemas em Orcad

Esquemático dos extensómetros

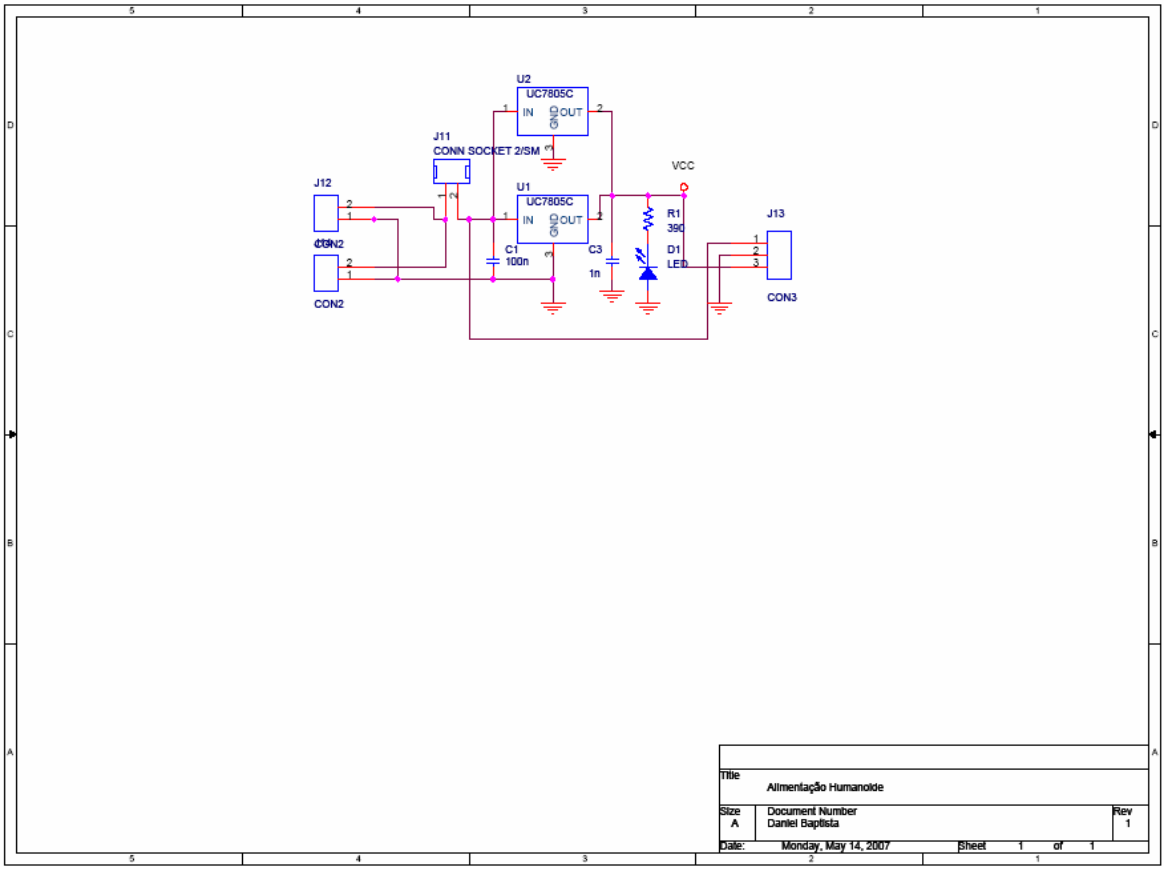


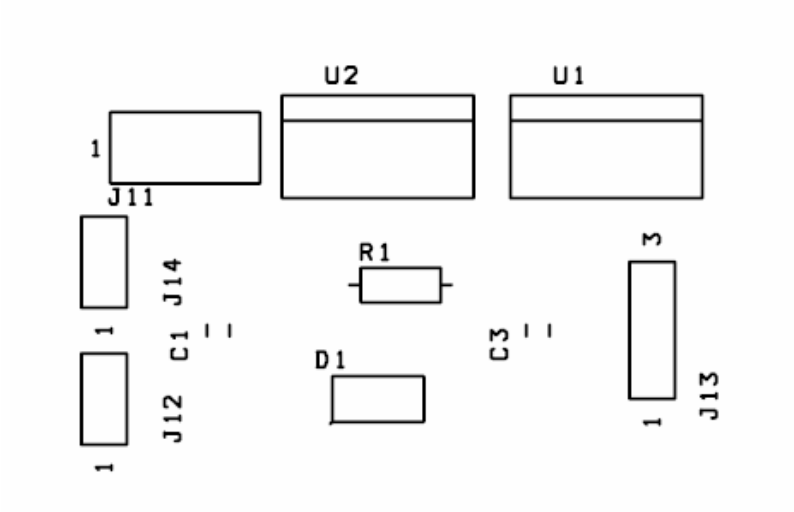

R24
R23
R26
R25
R36
R37
R51
R52
R38
R35
R34
R39
R47
R33
R48
R40
R32
R41
R31
R50
R42
R30
R43
R22
R21
R29
R44
R28
R27
J2
J5

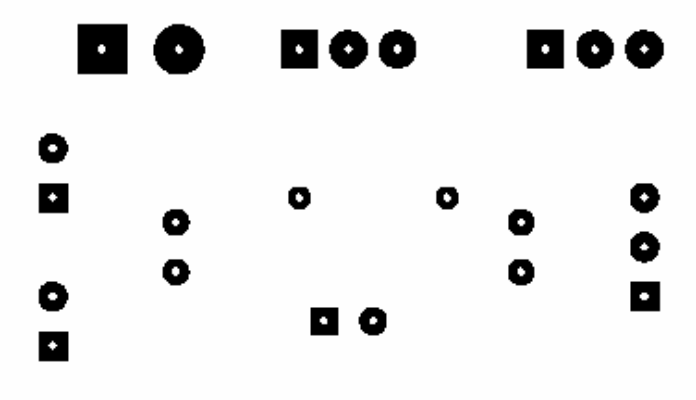


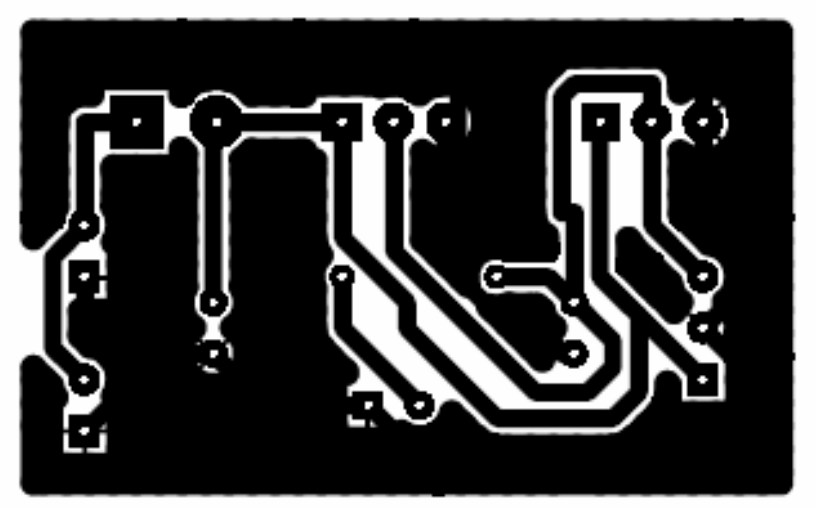


Esquemático das placas de alimentação




U2
U1
1
J11
R1
3
J14
1
C1
C3
D1
J12
1
J13
1





Anexo 4: Tutorial de OrCAD 10.5

Cadance OrCAD 10.5

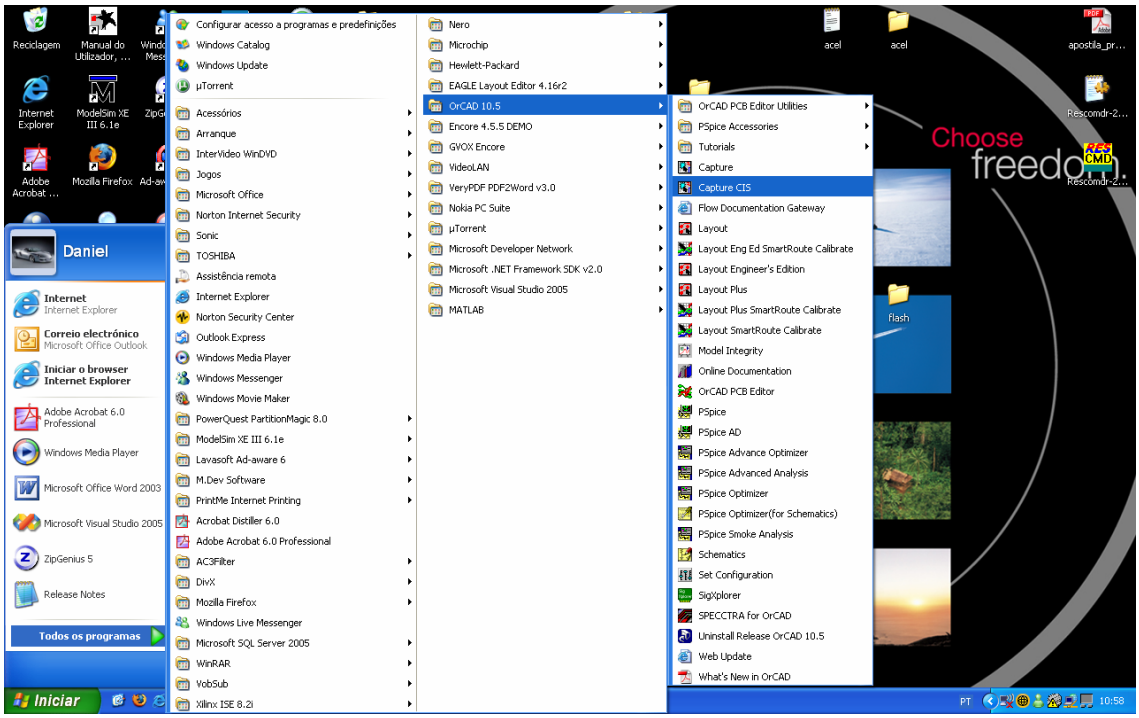
Layout Plus e Capture CIS

O *Cadance OrCAD* é uma das ferramentas mais conceituadas e utilizadas pelas empresas tanto para fabrico de circuitos impressos, como para simulação de circuitos electrónicos. As empresas de elaboração de componentes electrónicos, têm normalmente á disposição dos utilizadores as bibliotecas dos seus componentes para o *OrCAD*, o que não acontece para outros softwares disponíveis.

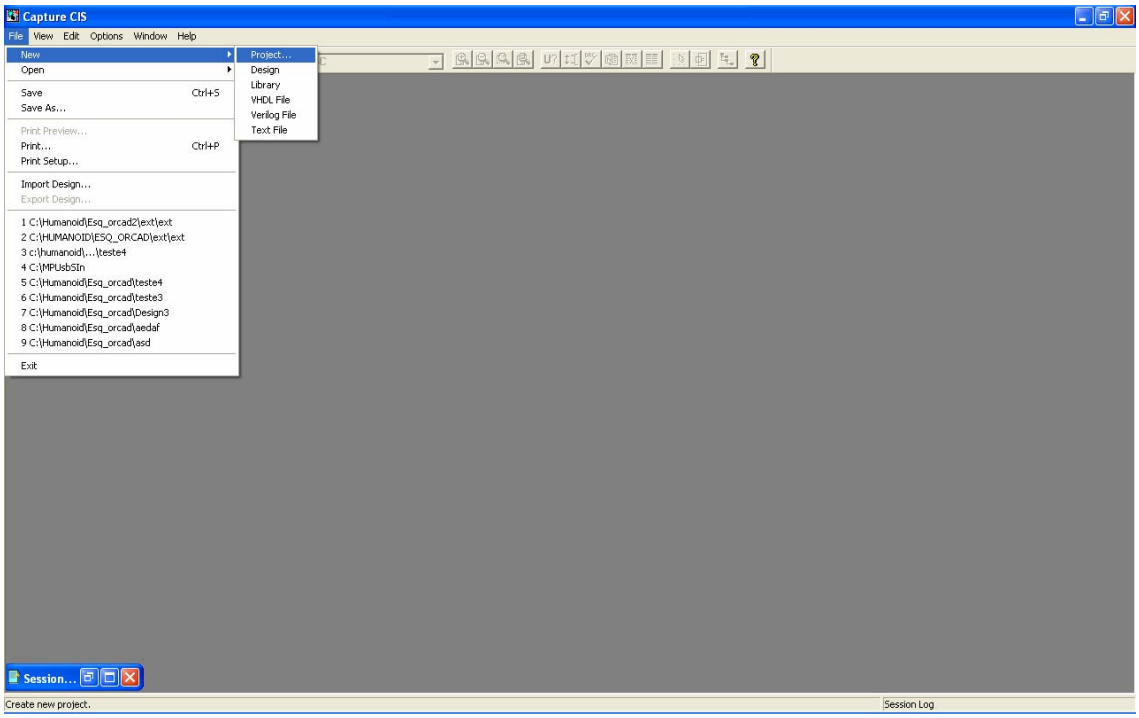
O *OrCAD* traz um pacote de software para diagnostico, simulação e confecção de placas de circuito impresso (*layout's*). É destinado à elaboração de layouts de PCB's.

A ferramenta de edição de esquemas de circuitos eletrônicos é o *Capture CIS*, onde a partir deste, pode-se construir o *layout* de qualquer circuito previamente desenhado, pode-se simular o circuito com o *PSpice AD*, ou ainda importar um circuito desenhado em PSpice para *Capture CIS* para que possa ser construído o PCB.

Para a criação de um novo esquemático abra o *Capture CIS*:

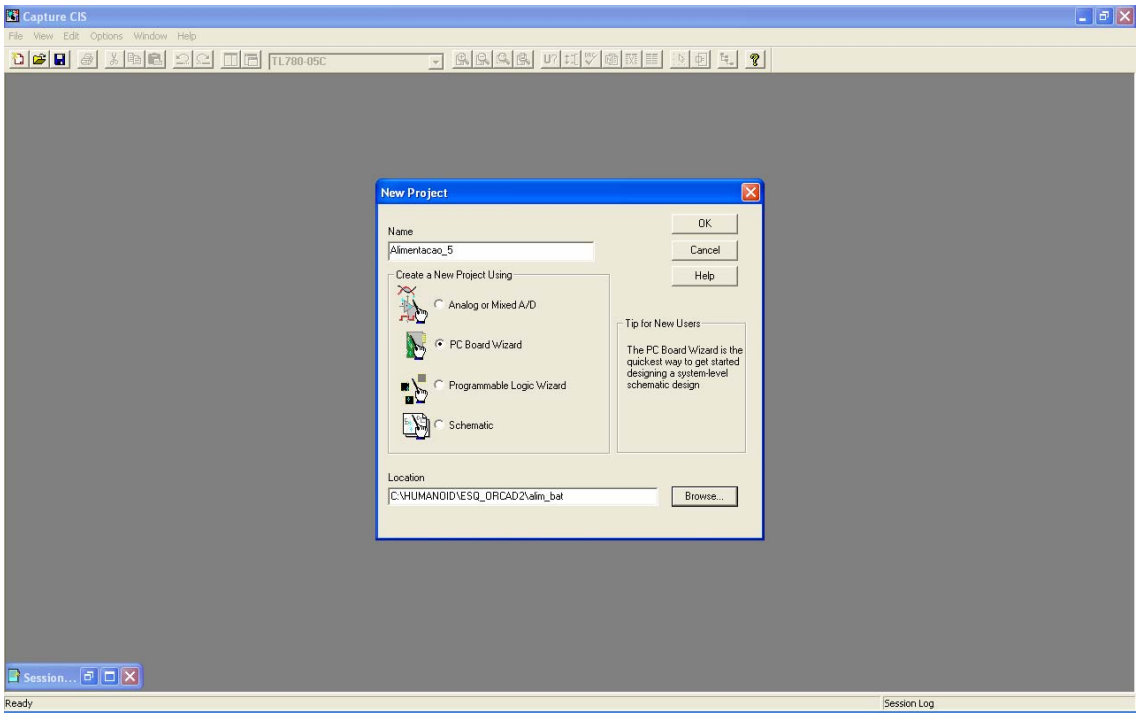


Para a criação de um novo projecto e execute os seguintes passos, *"File"* → "*New"* → "*Project…"*.

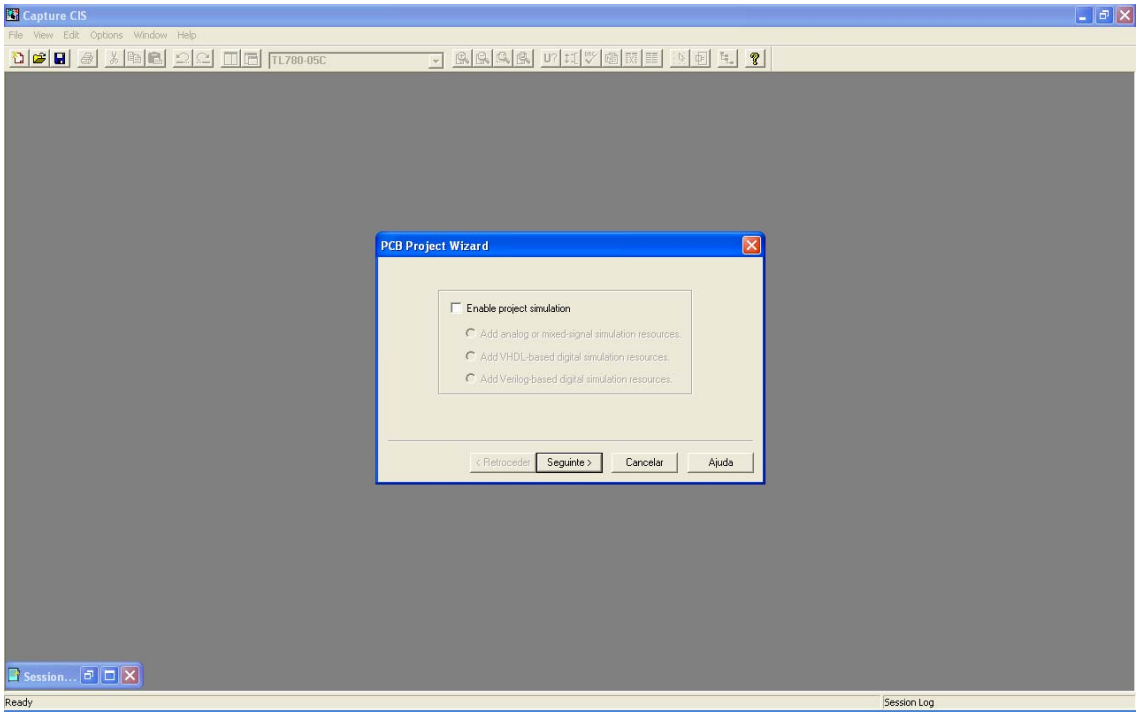


Escolha do tipo de projecto:

- Para simulação seleccione "*Analog or Mixed A/D"*;
- Para construção de uma PCB seleccione "*PC Board Wizard"*;
- Para programação lógica seleccione "*Programable Logic Wizard"*;
- Para criação de um esquemático seleccione "*Schematic"*;

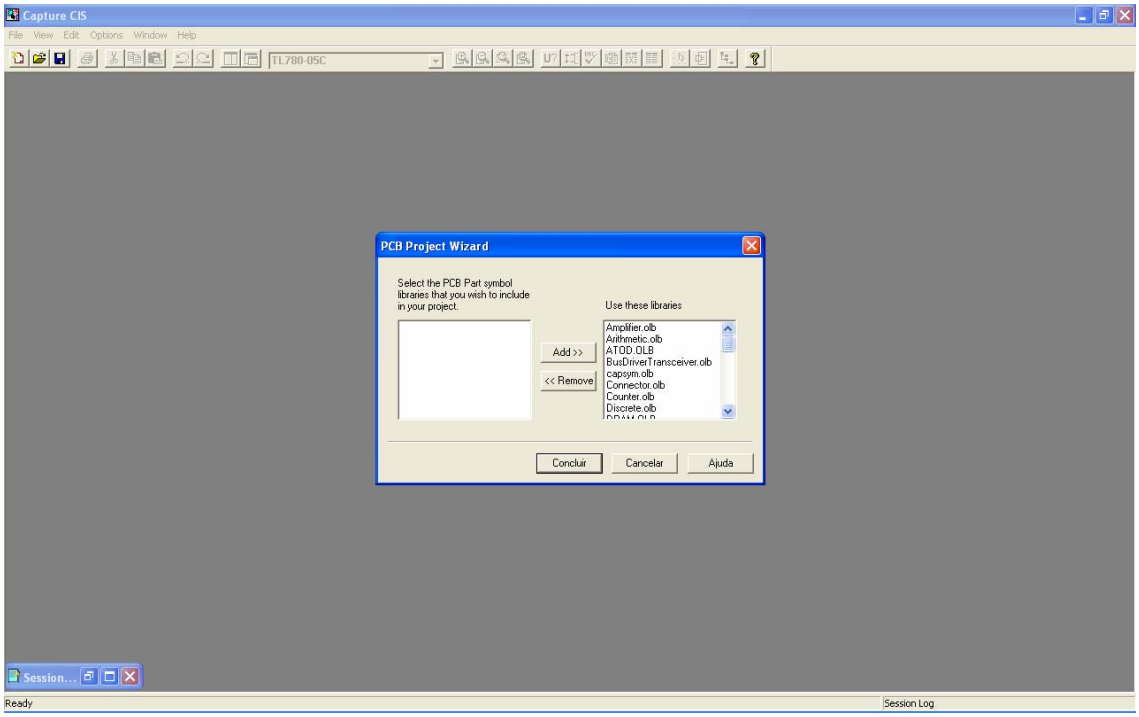


Ao criar um esquemático para construção de uma PCB, podemos contudo simular o esquemático para verificar se tudo está correcto com o circuito, para isso seleccione; "*Enable Project simulation*" → "*Add analog or Mixed signal simulation resources*" →"*Seguinte*".



Podemos adicionar as livrarias disponíveis do OrCAD na configuração do projecto, mas é possível adicionar as livrarias durante a elaboração do esquemático. Contudo, é aconselhável a adição das livrarias aqui disponíveis para a construção do esquemático.

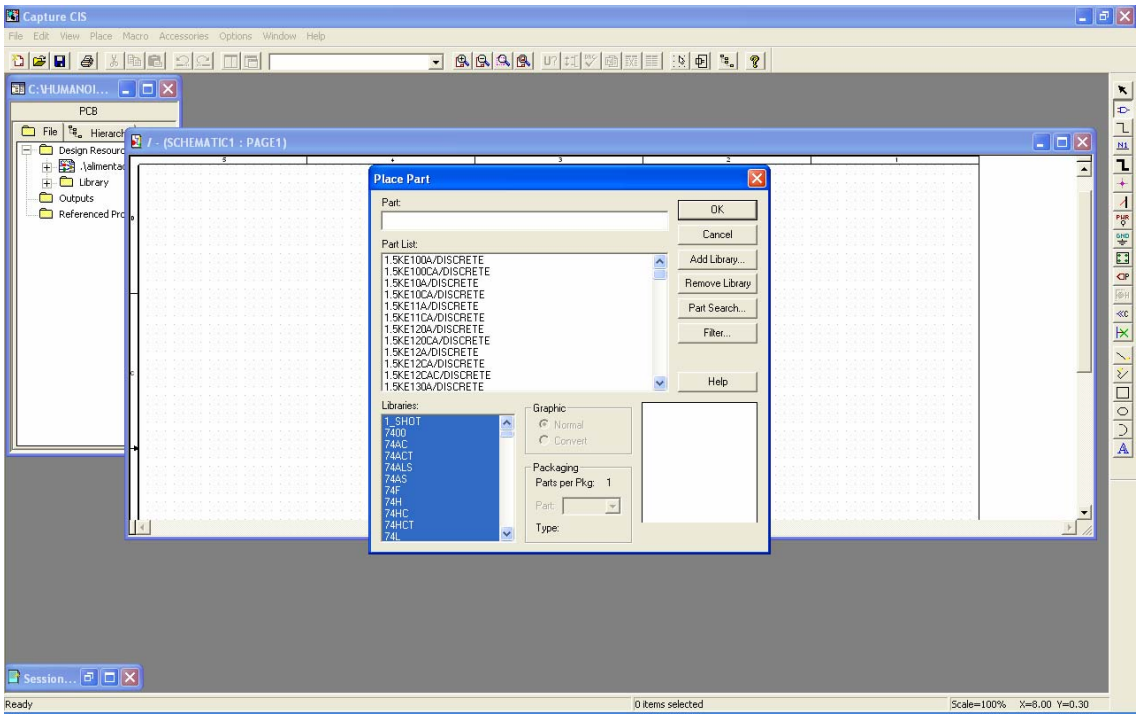
Para concluir, seleccione as livrarias e clique em "*Add>>*" → "*Concluir*".



Do lado direito do *Capture CIS* encontram-se os ícones das diversas ferramentas para permitir o desenho dos circuitos, em que as principais são:

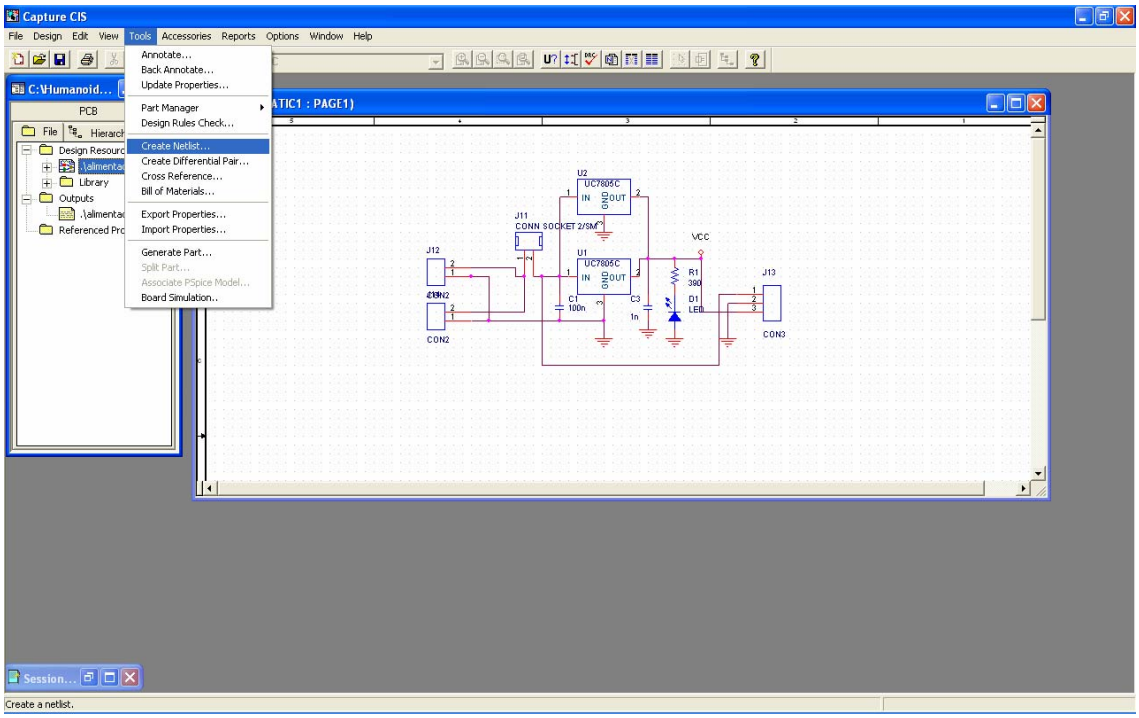
	*Select* – Passar de uma ferramenta parra o modo rato
	Place part – Permite adicionar componentes da livraria
	Place Wire – Introduzir Ligação
	Place Net Alias – Dar nome a uma ligação
	Place Bus – Introduzir um bus de ligações
	Place junction– Introduzir uma junção
	Net – Introduzir ligação do bus
	Pwr – Introduzir uma ligação a uma tensão
	Gnd – Introduzir uma ligação á massa
	Place hierarchial block – Criação de um sub-cricuito
	Place Port – Porta de ligação entre vários circuitos
	Place Pin – Inserir pino um circuito
	Place off-page connector – Conector de mudança de pagina
	Place on connect – Não ligar uma conexão
	Place line – Desenho de uma linha recta
	Place polyline – Desenho de uma linha
	Place rectangle – Desenho de um rectângulo
	Place ellipse – Desenho de uma elipse
	Place acr – Desenho de um arco
	Place text – Inserção de texto

Para introduzir um componente, clique no ícone *Place part* que vai abrir uma nova janela que contém a livraria de componentes, seleccionar o componente pretendido e clique *"OK"*. Para ligar pinos dos componentes, seleccione o ícone *Place Wire,* e com auxílio do rato lique os pinos que pretende.



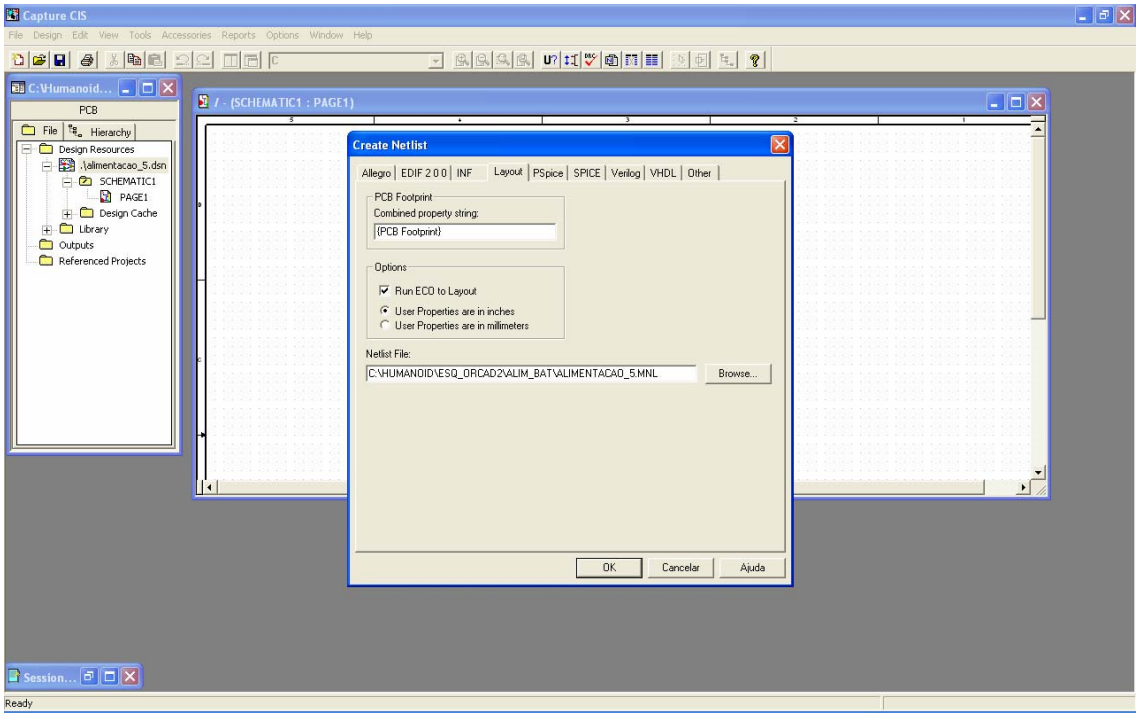
No fim do esquemático concluído, tem que ter pelo menos uma conexão á *Gnd* (massa), para que se possa fazer um pano de massa para a protecção contara o ruído. Neste ponto é necessário a criação de um ficheiro para, posteriormente, se criar a placa PCB.

No programa *Capture CIS*, aceda ao gestor de projectos e seleccione o ficheiro *"*.dsn"* na barra de ferramentas faça: *"Tools"* → *"Create NetList"*.



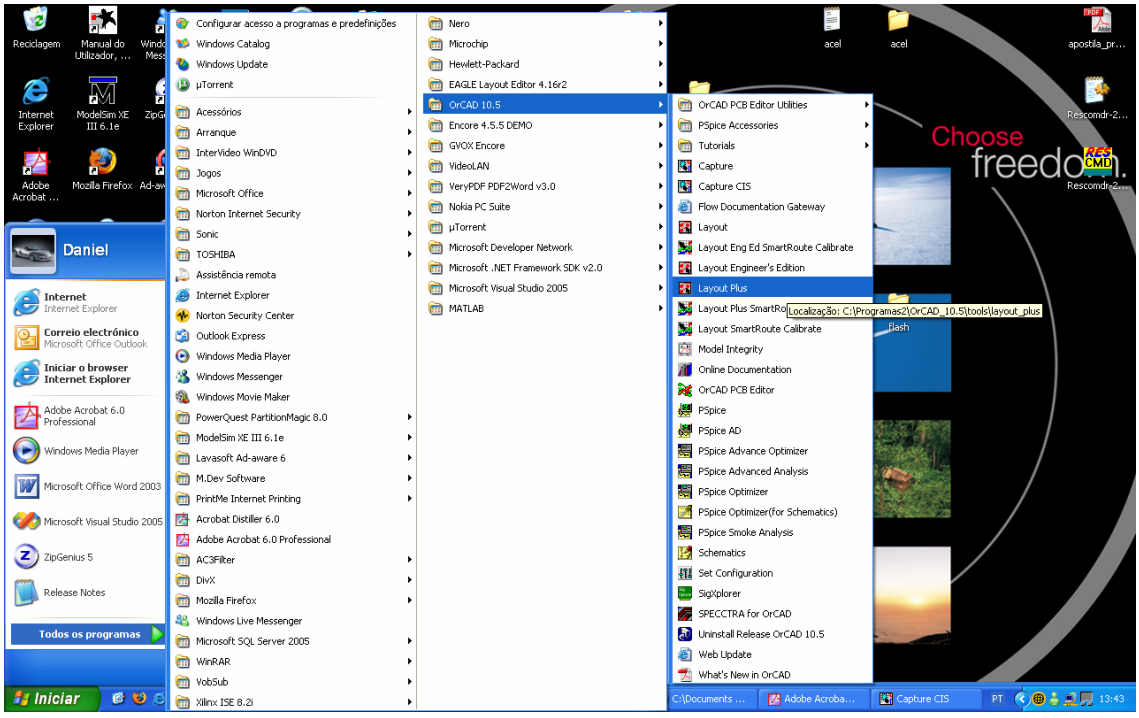
Na janela que será mostrada seleccione o separador *Layout*, localizado na parte superior da janela. Na caixa *Options*, certifique-se que o sistema de unidades esteja em *"inches"* e seleccione *Run ECO to Layout*. Observe o local onde o arquivo *NetList "*.mnl"* será guardado. Por definição, o nome e o local do ficheiro *NetList* é o mesmo do directório de

projecto do esquemático e chique em *"OK"*. Caso apareça alguma mensagem de aviso (informando que as alterações no esquemático serão salvas) e chique *"OK"*.



Observe que na pasta *Outputs* do gestor de projectos aparecerá um ficheiro *"*.mnl"* *NetList* criado. Este é o principal ficheiro para a passagem do *Capture CIS* para os outros programas do *OrCAD*, este ficheiro tem todos os componentes utilizados e as suas ligações.

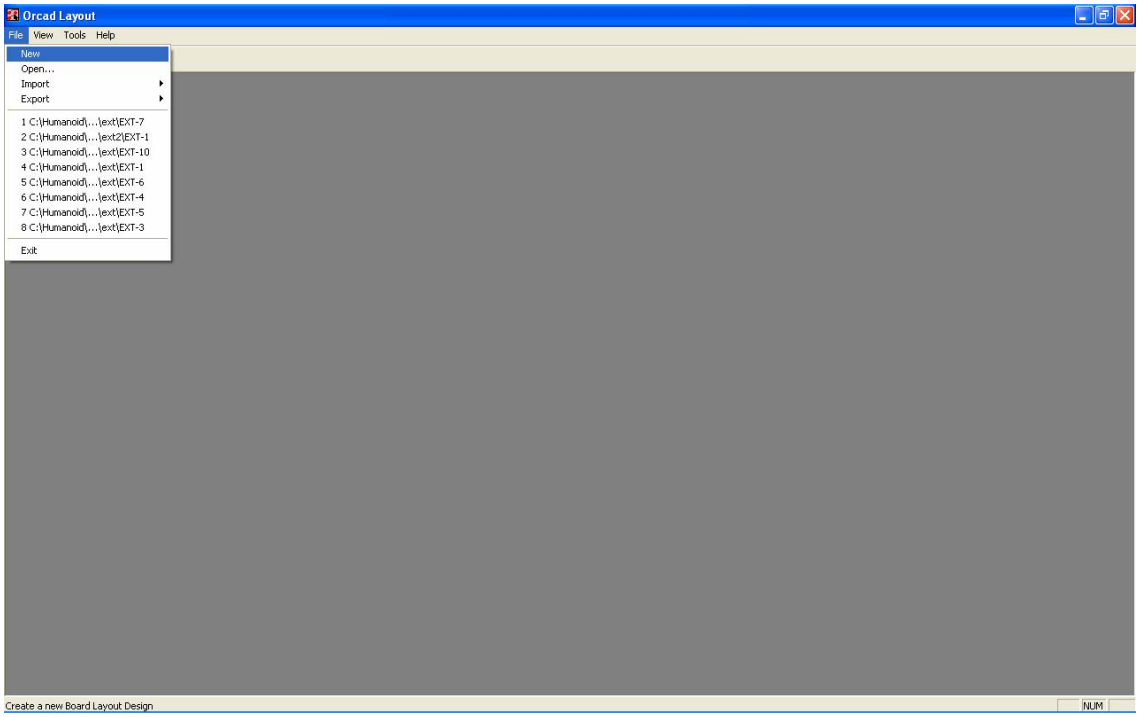
Neste ponto passamos para o *Layout Plus*.



Antes de tudo, é extremamente importante que o projectista defina quais os componentes que ira utilizar, o seu encapsulamento e qual a biblioteca que utilizou na criação do

circuito. Para projectistas menos experientes, ter em mãos todos os componentes do circuito como o seu *datasheet*, pode evitar inconvenientes e erros quanto à escolha dos *footprints* (desenho do componente na placa).

No OrCAD Layout clique em: "File" → "New"



Na primeira caixa de texto com o nome *"Input Layout TCH or TPL or MAX file"*, é para inserir a regra de desenho em que podem ser as seguintes:

- 1BET_ANY.TCH – Usada para placas com furos ou de montagem em superfície; o roteamento é standard para IC DIP pinos; as definições para IC DIP pinos para esta regra são: 62 mils de espessura da placa e de 38 mils para o furo; os ilhoses são de 25 mils e o espaçamento entre eles são de 100 mils; o espaçamento entre vias de roteamento é de 12 mils.
- 2BET_SMT.TCH – Usada para placas de montagem em superfície e tecnologia mista; o roteamento é reservado para standard IC DIP pinos; as definições para IC DIP pinos para esta regra são: 54 mils de espessura da placa e de 34 mils para o furo; os contactos são de 8 1/3 mils; e o espaçamento entre eles são de 50 mils; o espaçamento entre vias de roteamento é de 8 mils.
- 2BET_THR.TCH – Usada para placas com furos; o roteamento é reservado para standard IC DIP pinos; as definições para IC DIP pinos para esta regra são: 52 mils de espessura da placa e de 34 mils para o furo; os ilhoses são de 20 mils e o espaçamento entre eles são de 100 mils; o espaçamento entre vias de roteamento é de 8 mils.
- 3BET_ANY.TCH – Usada para placas de montagem em superfície e para montagem em superfície; o roteamento é reservado para standard IC DIP pinos; as definições para IC DIP pinos para esta regra são: 50 mils de espessura da placa e de 34 mils para o furo; os contactos são de 12 ½ mils; e o espaçamento entre eles são de 50 mils; o espaçamento entre vias de roteamento é de 6 mils.
- CERAMIC.TCH – Usado para chips cerâmicos.
- DEFAULT.TCH – Usado para tecnologia mais variada no fabrico de PCB's. Onde a tecnologia usada pode ser de vários tipos e ser integrada toda na mesma

placa, sem ter uma regra rigorosa e especifica.

- 386LIB.TCH – Usada como tecnologia definida e moldes depois da biblioteca PCB386 de ficheiros. Esta tecnologia, mantem todos os modelos e definições das cores das varias bibliotecas.
- HYBRID.TCH – Usada para chips híbridos.
- JUMP5535.TCH – Usada para placas de uma face com 55 mils de vias e com 35 mils para o furo.
- JUMP6035.TCH – Usada para placas de uma face com 60 mils de vias e com 35 mils para o furo.
- JUMP6238.TCH – Usada para placas de uma face com 62 mils de vias e com 35 mils para o furo.
- JUMP6238.TCH – Usada para múltiplos módulos chip.
- METRIC.TCH – Usada para placas com métrica (normalmente em milímetros).

Para o nosso caso é recomendável o uso da regra *default.tch*.

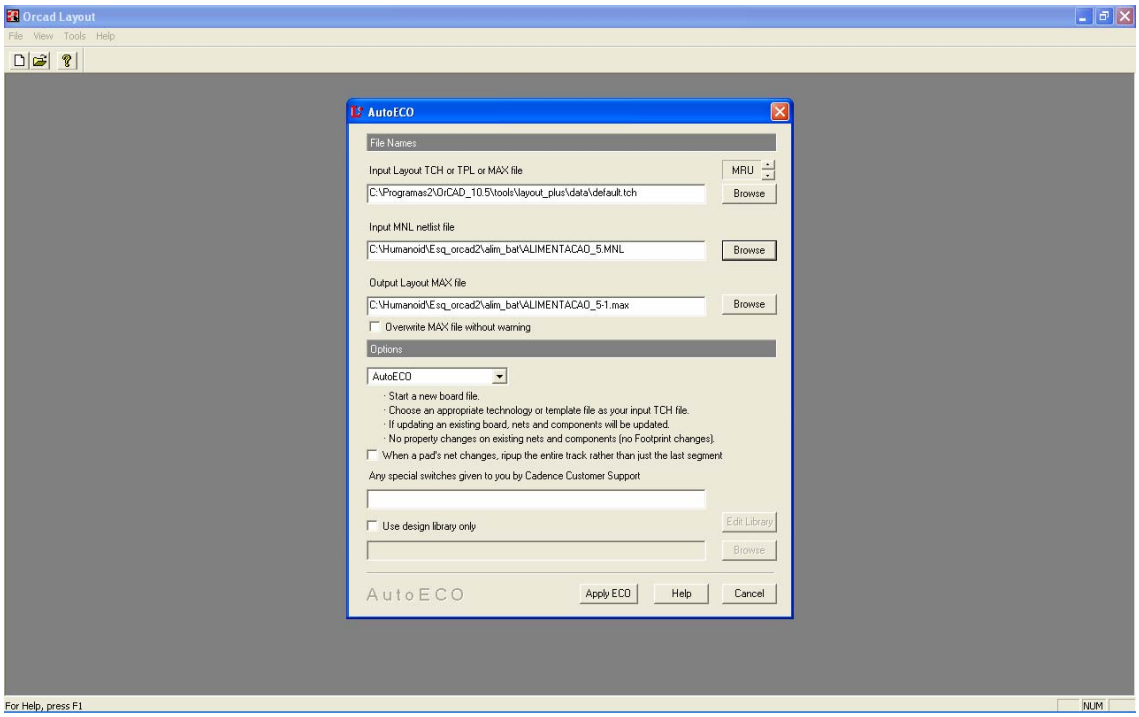
Na segunda caixa de texto com o nome *"Input MNL netlist file"*, é para inserir o ficheiro *NetList "*.mnl"* que foi criado no *Capture CIS* (o local do ficheiro *NetList* deve ser o mesmo do directório de projecto do esquemático).

Na teceira caixa de texto com o nome *"Output Layout MAX file"*, este é o ficheiro(*"*.max"*) de saída da placa PCB. Por definição este fica situado na mesma dirctoria do projecto do esquemático.

Por fim clique em *"Apply ECO"*.

O *NetList* previamente gerado pelo editor de esquemáticos, que ao ser carregado pelo *OrCAD Layout Plus*, fará com que um programa secundário chamado *AutoECO* seja executado. Este programa tem a função de, a partir do *NetList*, encontrar os componentes na biblioteca do *Layout Plus* e, após encontra-los, efectuar as ligações existentes entre os mesmos.

Obs: Conversão entre milímetros (mm), e polegadas (pol) e milésimos de polegadas (mils): 1pol = 25,4mm, 1mm ≈ 0,03937pol, 1pol = 1000mils

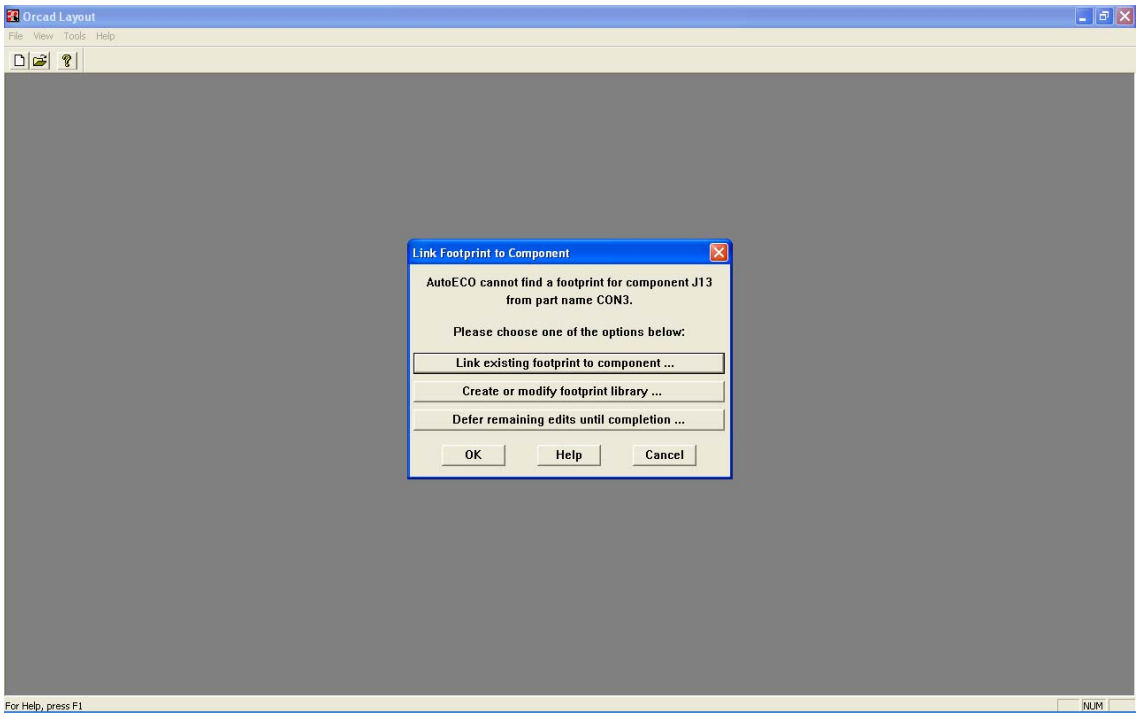


O *AutoECO* será executado e pode-se verificar que este executa uma série de operações. Estas operações resumem-se em inserir os componentes descritos no *Netlist* no *Layout Plus*, bem como suas ligações.

No caso, em que o programa não encontre um *footprint* (desenho do componente na PCB) de algum componente do editor de esquemático na biblioteca, será apresentada uma caixa de diálogo com algumas opções.

- *Link existing footprint to component*: Indicar manualmente o componente que deve ser utilizado o que foi apresentado no título. Em que deve saber qual o *footprint* adequado para este componente.
- *Create or modify footprint library*: Esta opção permite que crie um novo componente na biblioteca do *Laytou Plus* ou modifique um já existente.
- *Defer remaining edits until completion*: Esta opção sugere que salte este componente para que mais tarde se possa seleccionar um *footprint* na biblioteca.

È a conselhavel que clique em "Link existing footprint to component".



Neste ponto pode-se escolher o *footprint* mais adequado ao componente. (Existem vários componentes, como se pode verificar, com mesmo *footprint* em que se pode utilizar de uns para os outros). Neste ponto é conveniente ter os *datasheets* dos componentes para sabermos quais os tipos de *footprint* mais adequados, mas posteriormente, pode ser alterado como vamos ver mais á frente. Vamos dar alguns exemplos para se perceber como funciona os *footprints*.

Resistências:

-Biblioteca TM_AXIAL.
-Componente padrão: AX/.400x.125/034.
-Onde: 400 é a distância dada em mils (milésimos de polegada) entre os pinos da resistência e 125 é a sua largura (também em mils). 034 é o diâmetro do furo na placa.

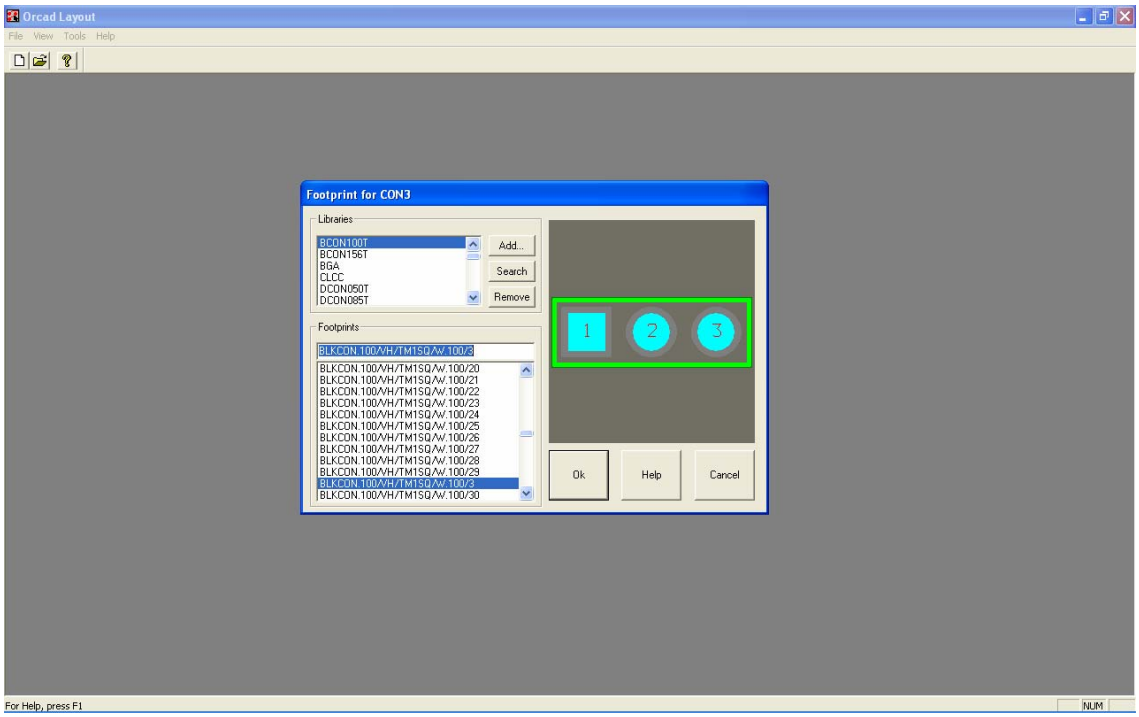
Condensadores Electrolíticos:

-Biblioteca: TM_CYLND.
-Exemplo: CYLND/D.400/LS.200/034
-Onde: D.400 é o diâmetro docondensador e LS.200 é a distância entre seus pinos, 034 é o diâmetro do furo na placa.

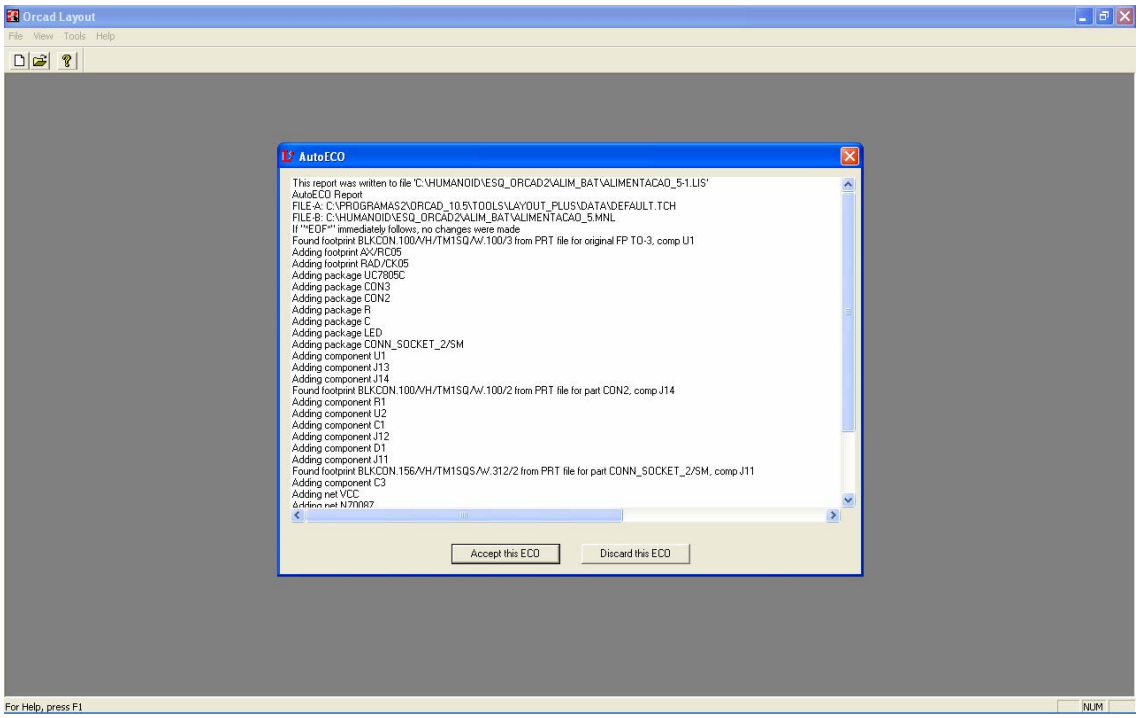
Circuitos Integrados:

-Biblioteca: DIP100T
-Exemplo: DIP100/14/W.300/L.700
-Onde: 100 é a distância entre os pinos em mils, 14 é o número de pinos, W.300 é a distância, em mils, entre os eixos dos pinos (largura do CI) e L.700 é o comprimento

do componente.

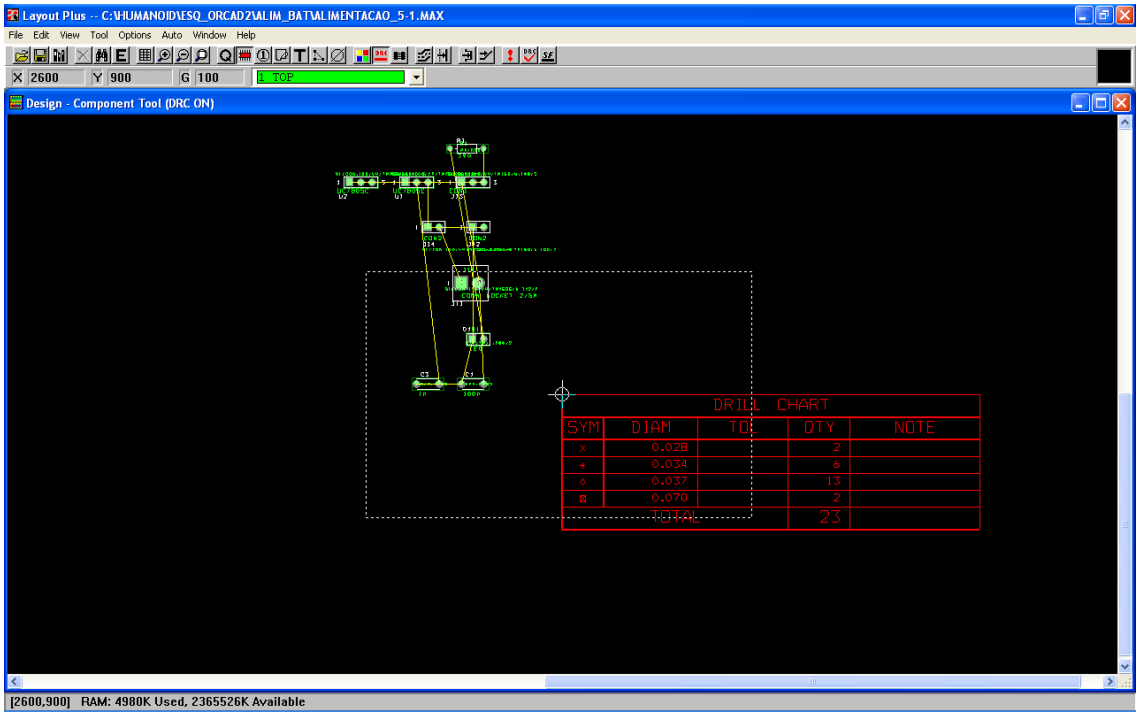


No fim de todos os componentes terem um *footprint* associado basta clicar em *"Accept this ECO"* e está concluída a fazer dos *footprints*.

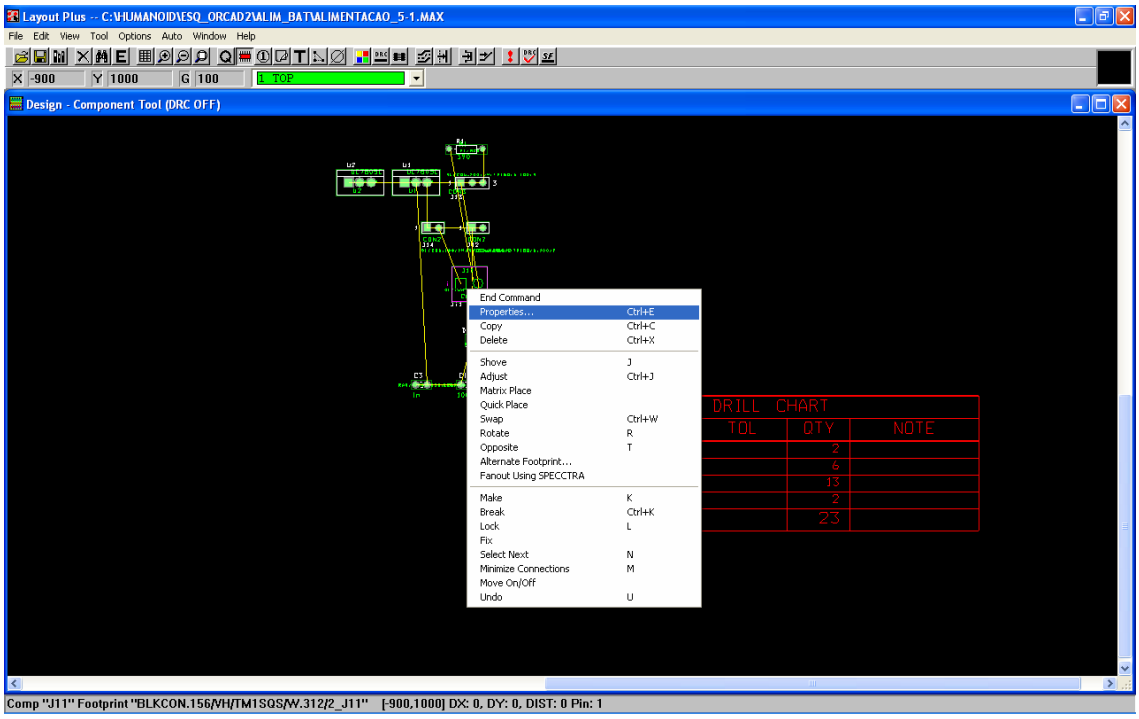


Agora com o *Layout Plus* aberto. As principais ferramentas são:

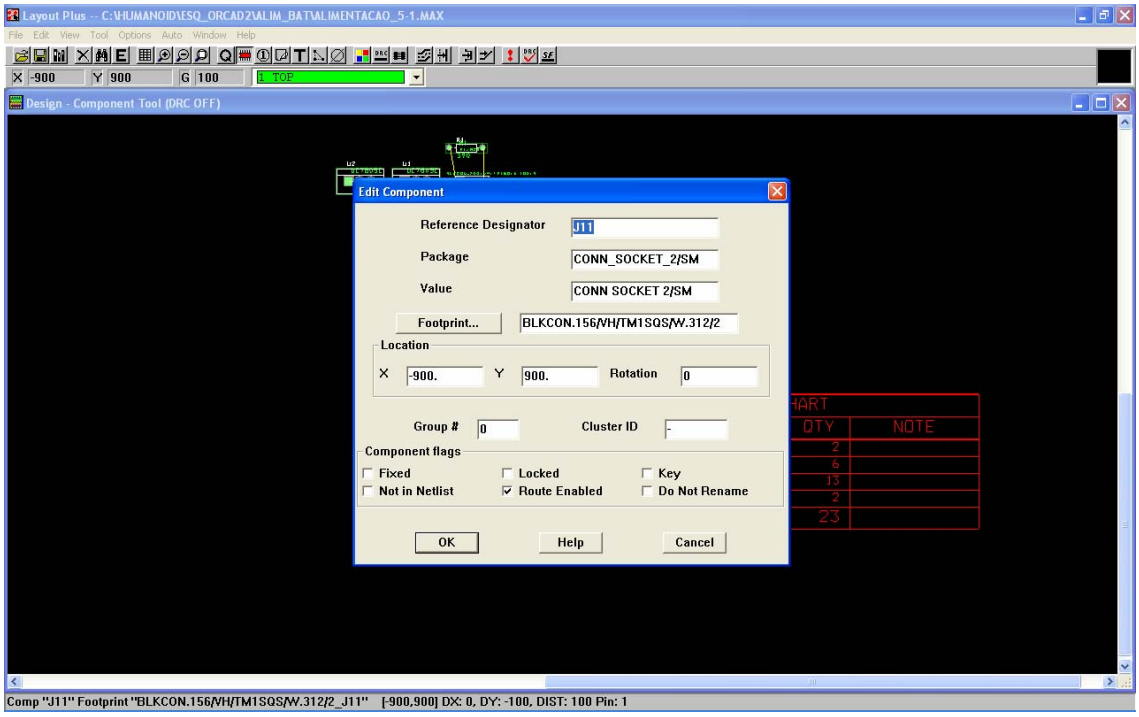
View Spreadsheet – Listas de opeções de roteamento
Zoom in – Aumento da imagem
Zoom out – Diminuição da imagem
Zoom all – Ver todos os componentes no ecrã
Query – Ver as características todas de um componente seleccionado
Component Tool – Selecção de um componente
Pin Tool – Selecção de um pino
Obstacle Tool – Criação e selecção de obstáculos
Text Tool – Escrita da texto
Connection Tool – Conexão manual de componentes
Error Tool – Seleção de erros
Color Settings – Descrição de todas as cores
Online DRC – Ligar/Desligar verificação e erros na placa
Reconnect Mode – Modo de conexão
Autopath Route Mode – Modo de roteamento manual
Shove Track Mode – Mudar a linhas de roteamento
Edit Segment Mode – Edição de um segmento de linha
Add/Edit Route Mode – edição dpo modo de roteamento
Refresh All – Actualização
Design Rule Check – Verificação da regra de desenho
Selection Filter – Filtro de selecção



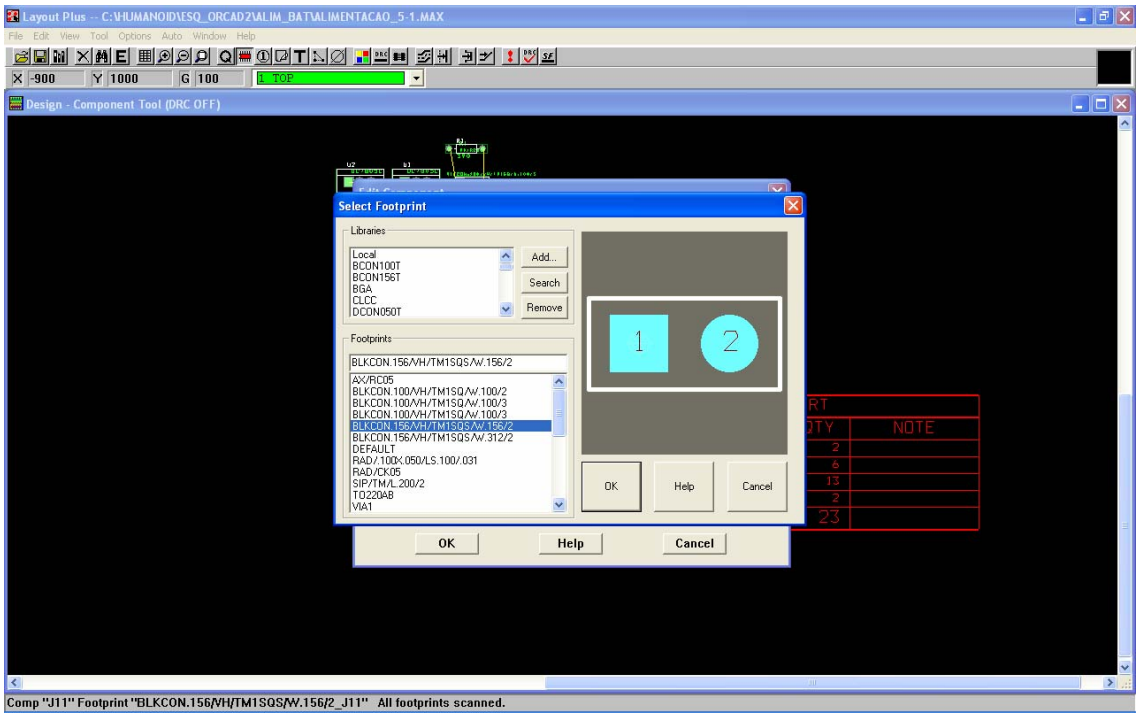
Pode verificar se todos os *footprints* são adequados aos componentes. Pois que na biblioteca de *footprints* não temos uma noção visual do tamanho destes. Caso, seja, necessário a alteração do *footprint,* seleccione a ferramenta *"Component Tool"* e seleccione o componente que deseje alterar o *footprint*. Com o rato em cima do componente clique com o botão do lado direito, e seleccione *"Properties…"* do componente.



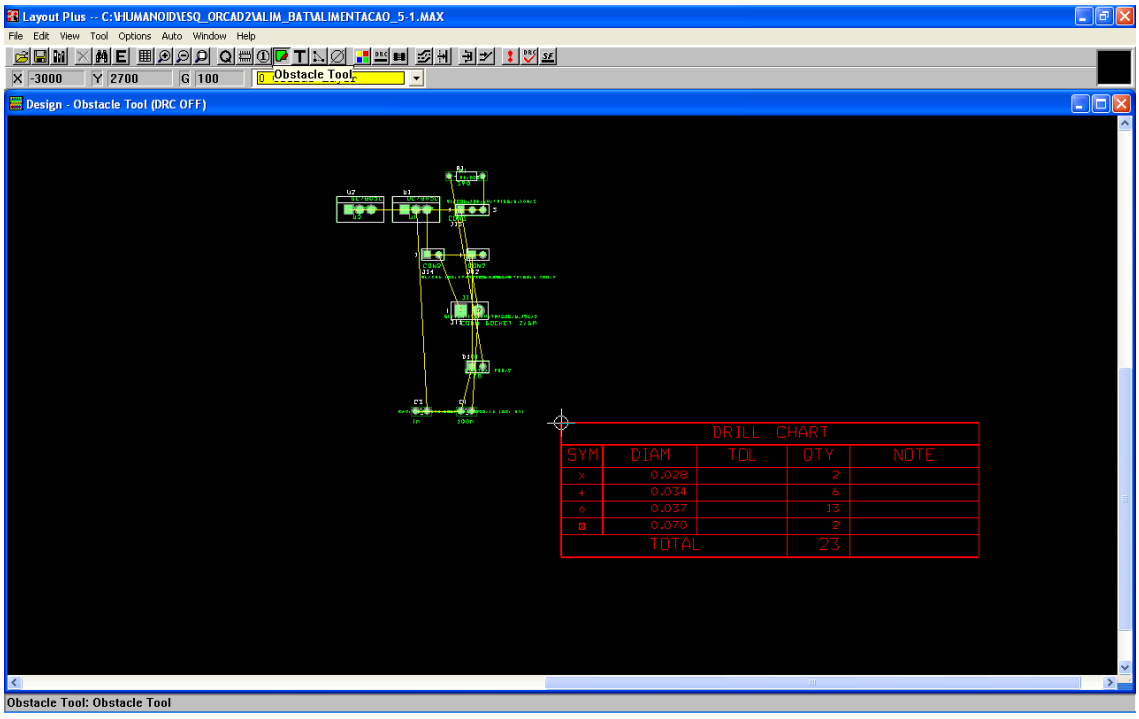
Aparecerá a seguinte caixa de dialogo, a clique em *"Footprint…"*.



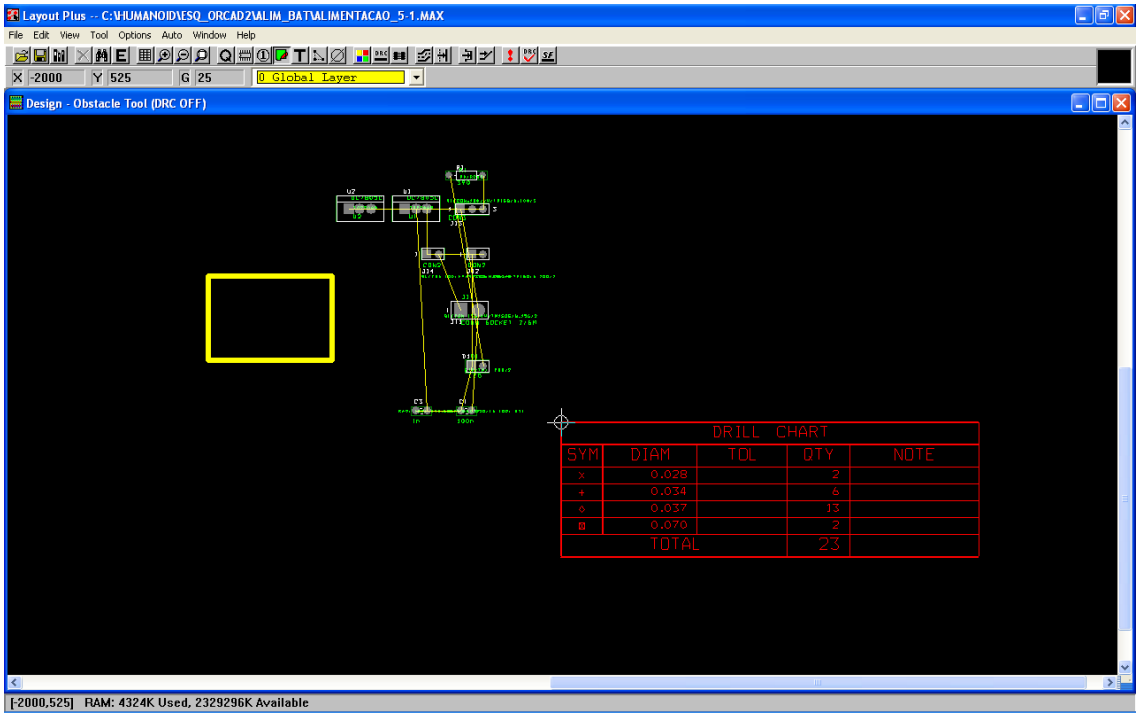
Aparecerá de novo a biblioteca de *footprints* e escolho o mais adequado para o seu componente e clique *"OK"*.



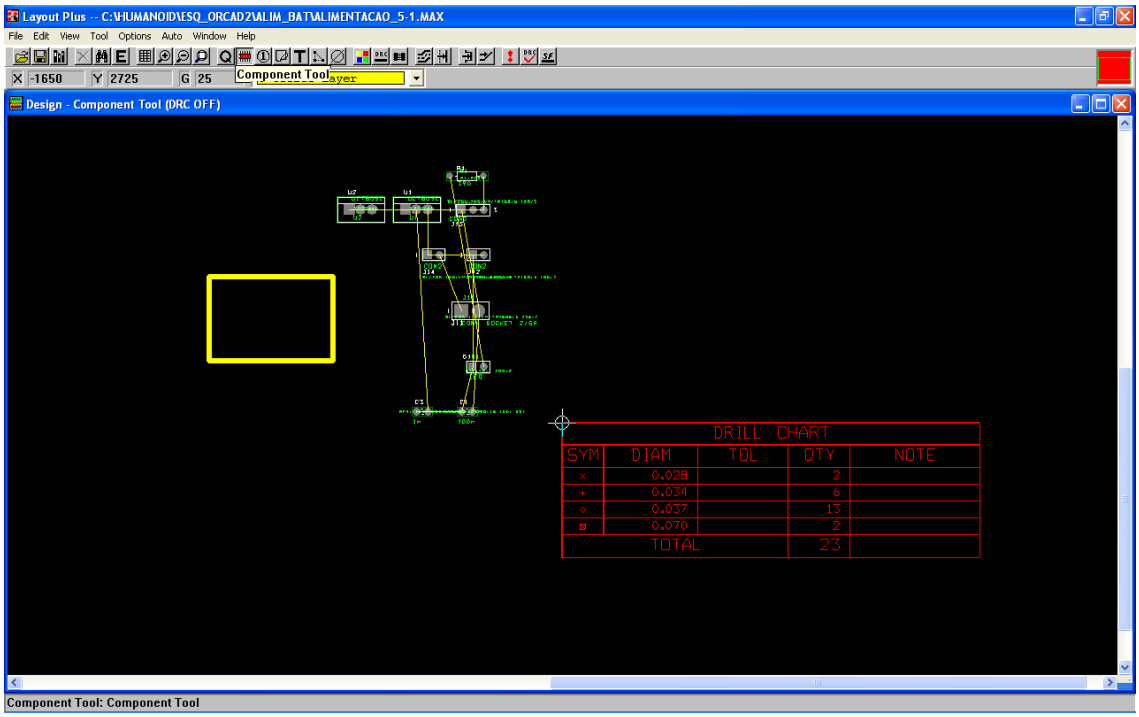
O próximo passo é criar a linha de "obstáculo", isto é, a linha de contorno da placa. Para isso, seleccione a *"0 Global Layer"* que deve aparecer a amarelo. Seguindo seleccionando a ferramenta *"Obstacle Tool"* e traça-se os contornos da placa.



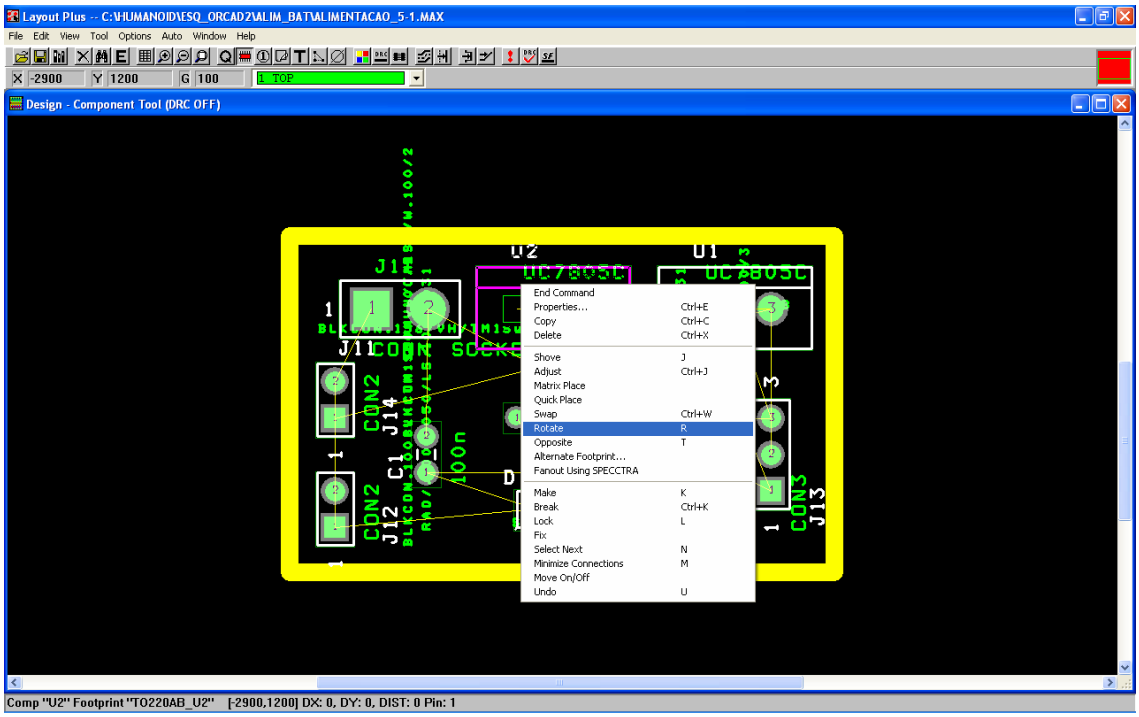
Ao fim de concluído deve ter o seguinte aspecto. Dimensões da placa podem ser medidas fazendo *"Tool"*→ *"Dimension"*, e reajustar as dimensões da placa.



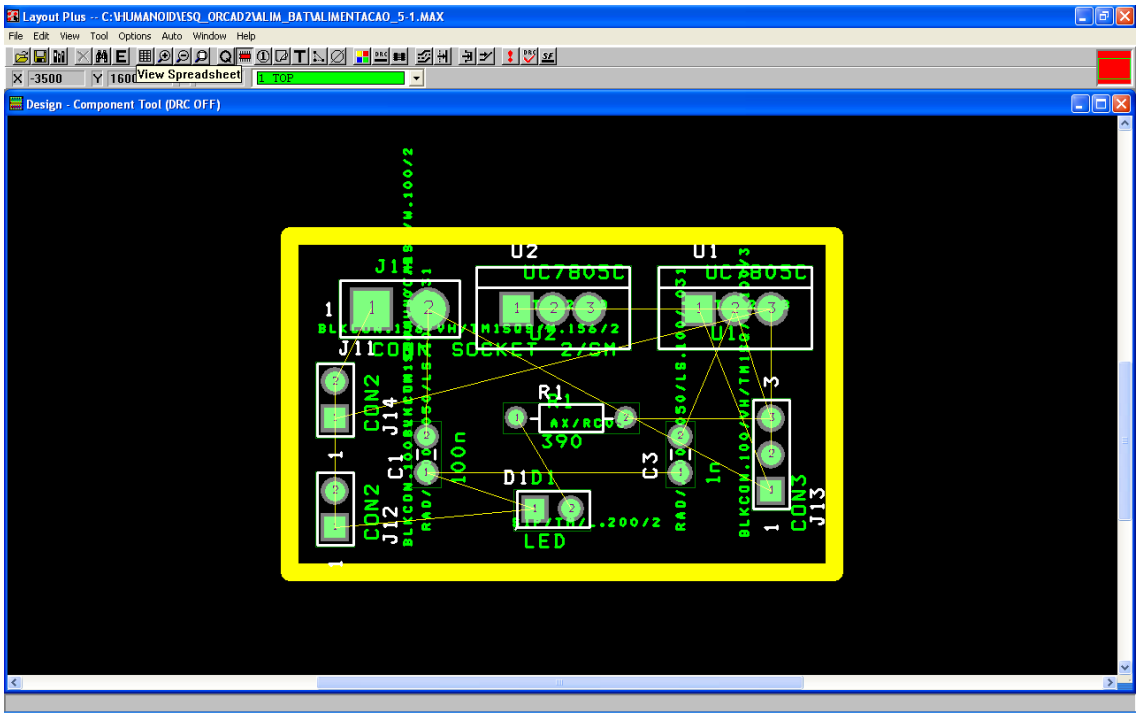
Agora temos que colocar os componentes dentro das dimensões da placa para isso, seleccionamos a ferramenta *"Componente Tool"*, clicamos nos componentes e arrastamos para dentro da placa e fazemos *Esc* para largar os componentes.



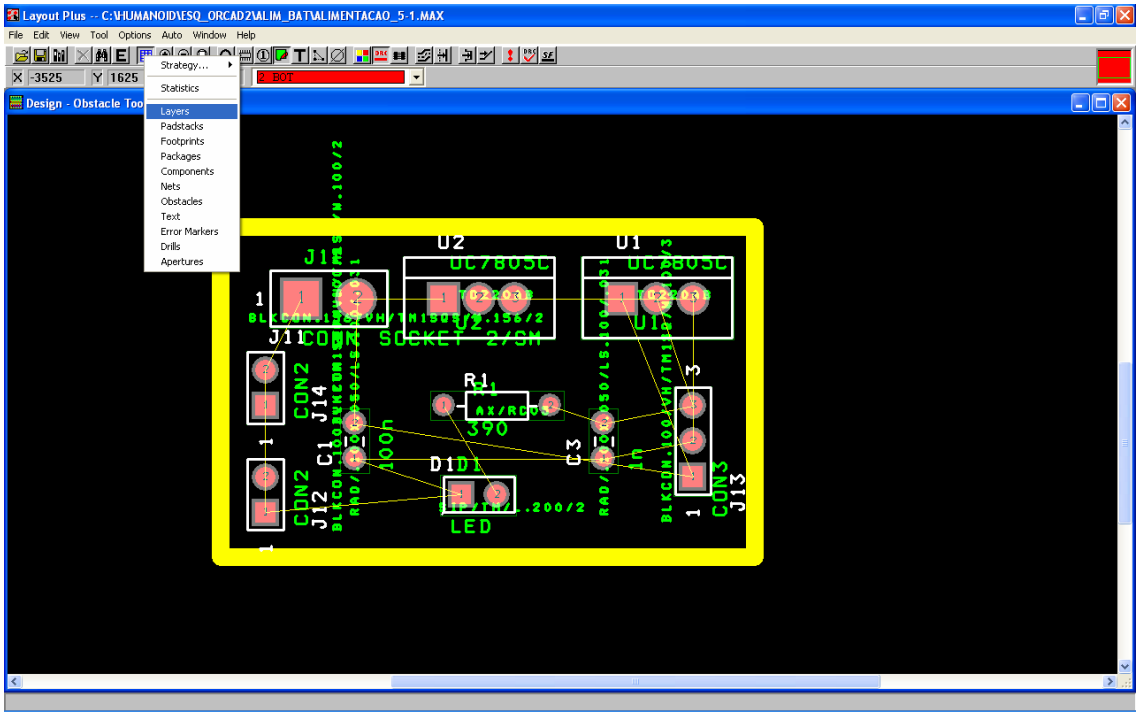
Os componentes podem ser rodados, ajustados, mudar o sentido, … Para isso com a ferramenta de "*Component Tool*" seleccionada basta clicar sobre o componente com o rato com o botão direito e aplicar o efeito desejado.



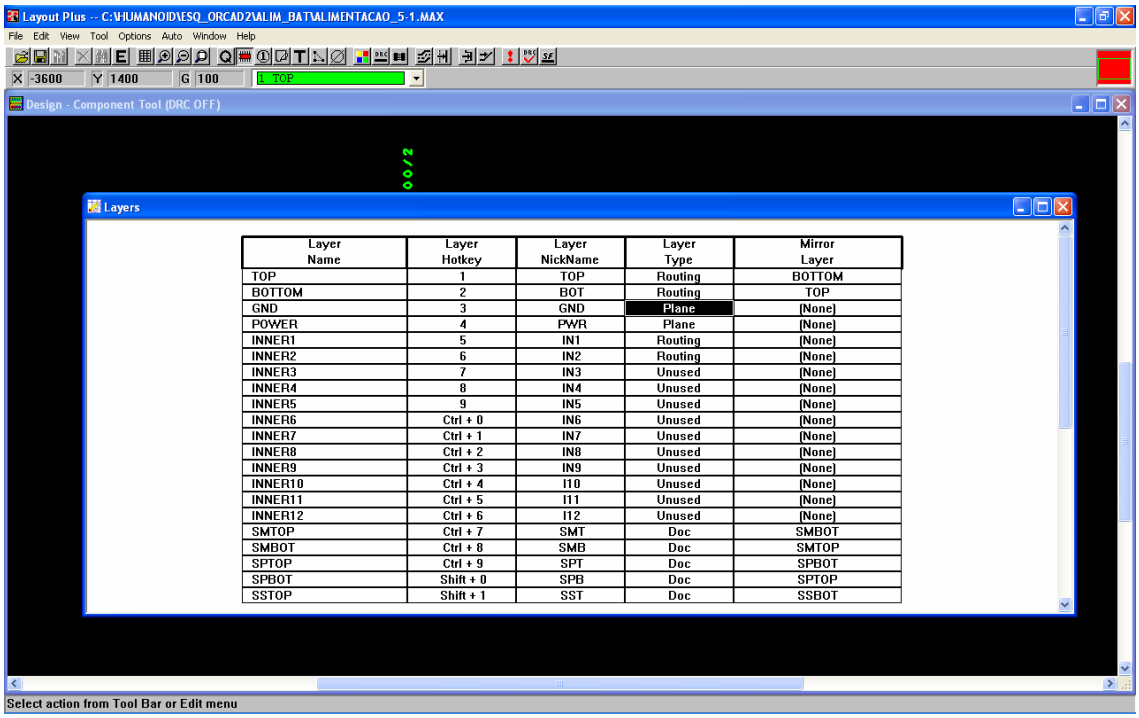
No fim, de todos os componentes estarem no sítio correcto, vamos ver as definições de roteamento, isto é, ver quantas faces a utilizar, ver a largura da via, o espaço entre vias… Para isso clica-se na ferramenta *"View Spreadsheet"*.



O primeiro passo é ver as layers. Clica-se *"View Spreadsheet"* → *"Layers"*

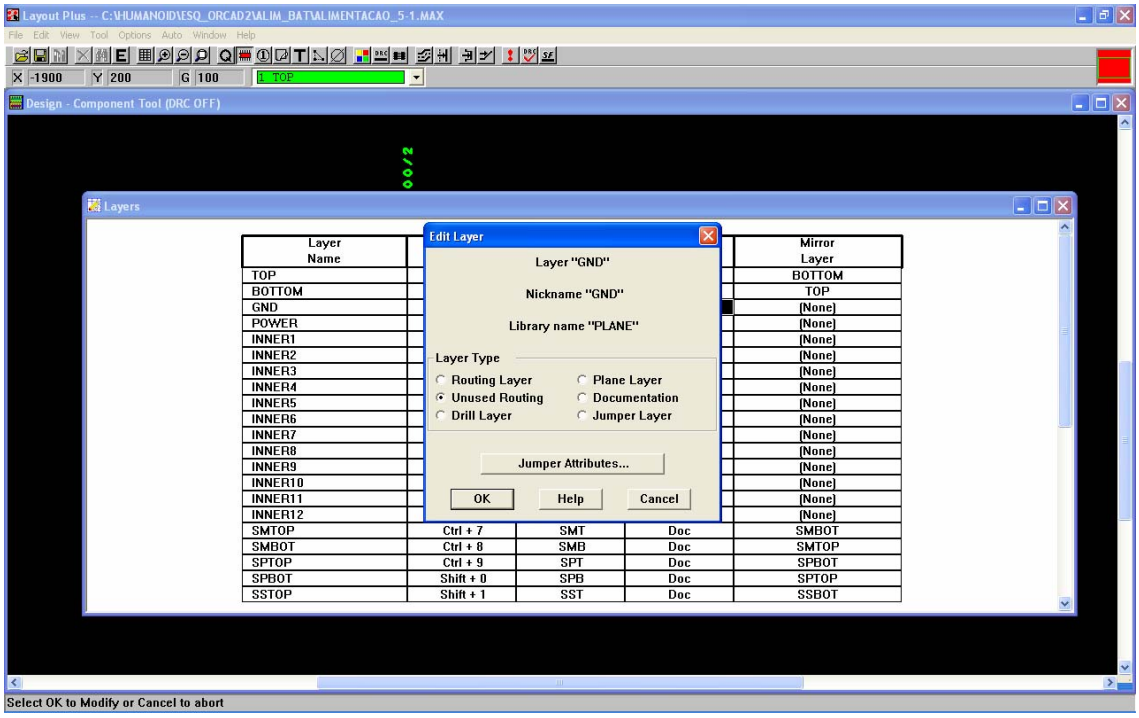


Para uma placa de duas camadas só o topo e fundo é que é roteado, logo todos os outros modos não são roteados. Para alterar, clica-se duas vezes em cima dos modos.

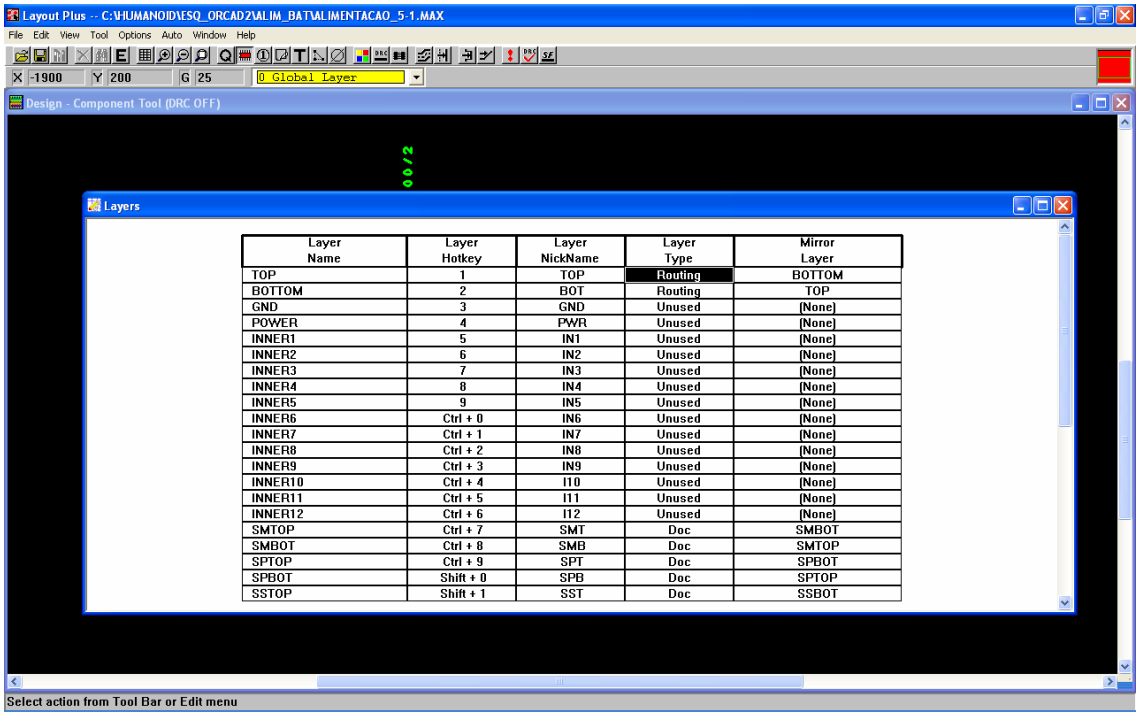


Layer Name	Layer Hotkey	Layer NickName	Layer Type	Mirror Layer
TOP	1	TOP	Routing	BOTTOM
BOTTOM	2	BOT	Routing	TOP
GND	3	GND	Plane	[None]
POWER	4	PWR	Plane	[None]
INNER1	5	IN1	Routing	[None]
INNER2	6	IN2	Routing	[None]
INNER3	7	IN3	Unused	[None]
INNER4	8	IN4	Unused	[None]
INNER5	9	IN5	Unused	[None]
INNER6	Ctrl + 0	IN6	Unused	[None]
INNER7	Ctrl + 1	IN7	Unused	[None]
INNER8	Ctrl + 2	IN8	Unused	[None]
INNER9	Ctrl + 3	IN9	Unused	[None]
INNER10	Ctrl + 4	I10	Unused	[None]
INNER11	Ctrl + 5	I11	Unused	[None]
INNER12	Ctrl + 6	I12	Unused	[None]
SMTOP	Ctrl + 7	SMT	Doc	SMBOT
SMBOT	Ctrl + 8	SMB	Doc	SMTOP
SPTOP	Ctrl + 9	SPT	Doc	SPBOT
SPBOT	Shift + 0	SPB	Doc	SPTOP
SSTOP	Shift + 1	SST	Doc	SSBOT

Desactive todas as *layers* para *"Unused Routing"* e clique em *"OK"*.

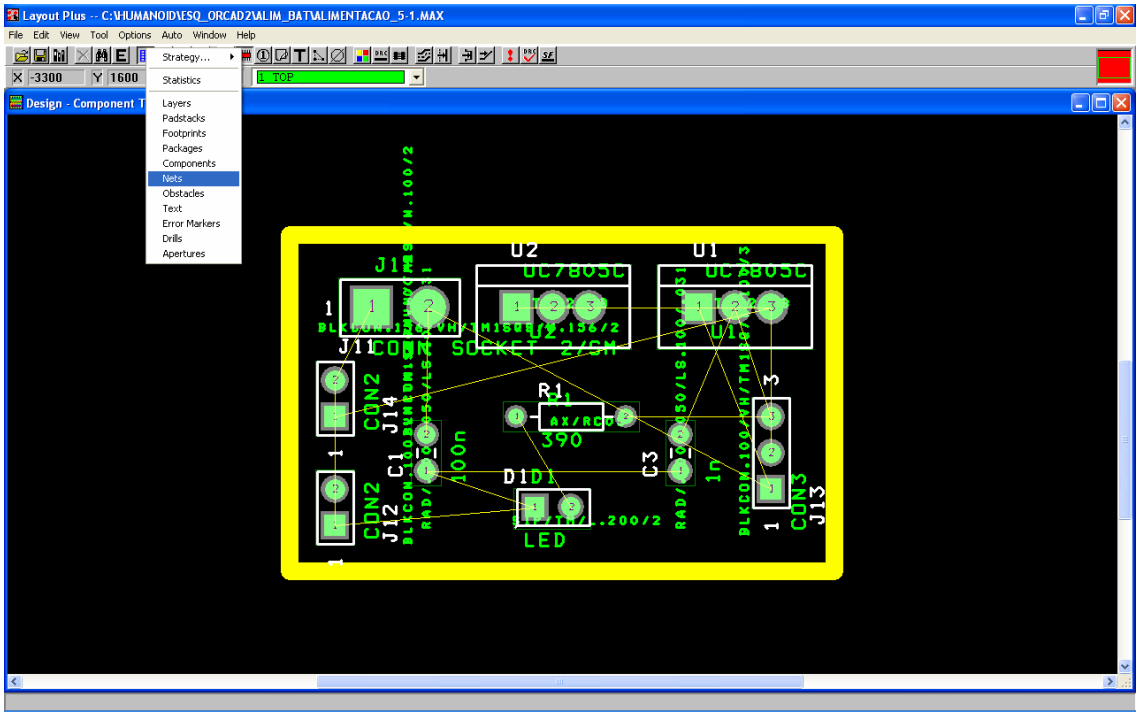


No fim de estar tudo desactivo deve ter o seguinte aspecto. Só o topo e fundo é que estão para routeamento.

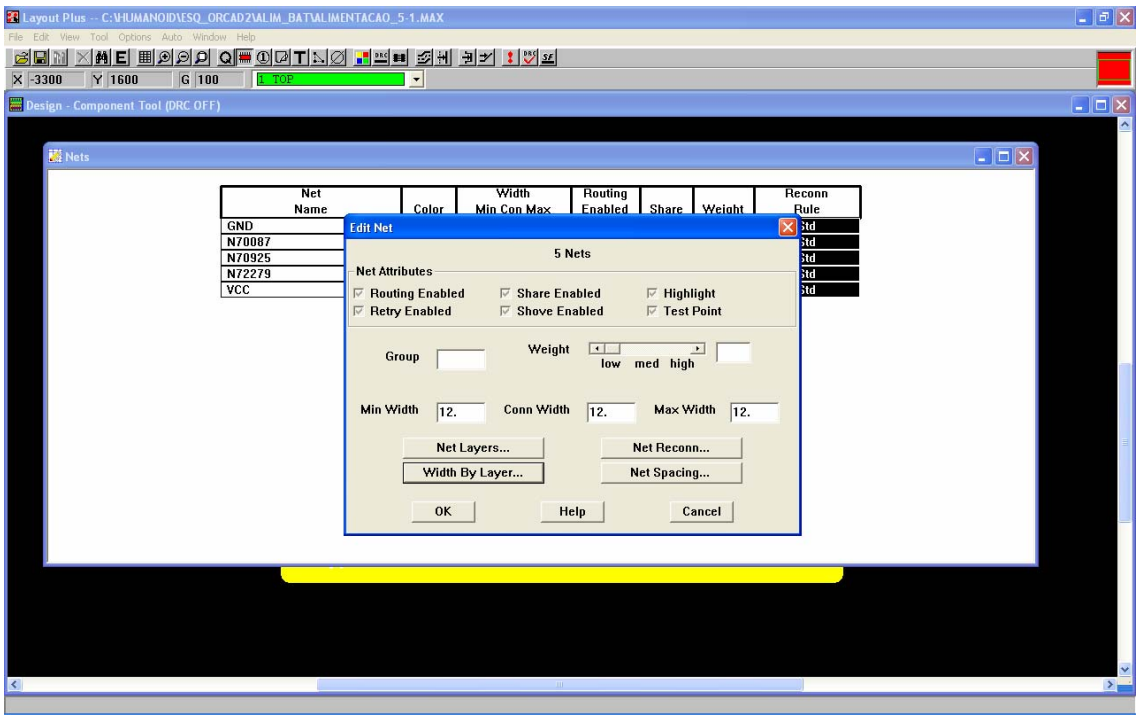


Layer Name	Layer Hotkey	Layer NickName	Layer Type	Mirror Layer
TOP	1	TOP	Routing	BOTTOM
BOTTOM	2	BOT	Routing	TOP
GND	3	GND	Unused	(None)
POWER	4	PWR	Unused	(None)
INNER1	5	IN1	Unused	(None)
INNER2	6	IN2	Unused	(None)
INNER3	7	IN3	Unused	(None)
INNER4	8	IN4	Unused	(None)
INNER5	9	IN5	Unused	(None)
INNER6	Ctrl + 0	IN6	Unused	(None)
INNER7	Ctrl + 1	IN7	Unused	(None)
INNER8	Ctrl + 2	IN8	Unused	(None)
INNER9	Ctrl + 3	IN9	Unused	(None)
INNER10	Ctrl + 4	I10	Unused	(None)
INNER11	Ctrl + 5	I11	Unused	(None)
INNER12	Ctrl + 6	I12	Unused	(None)
SMTOP	Ctrl + 7	SMT	Doc	SMBOT
SMBOT	Ctrl + 8	SMB	Doc	SMTOP
SPTOP	Ctrl + 9	SPT	Doc	SPBOT
SPBOT	Shift + 0	SPB	Doc	SPTOP
SSTOP	Shift + 1	SST	Doc	SSBOT

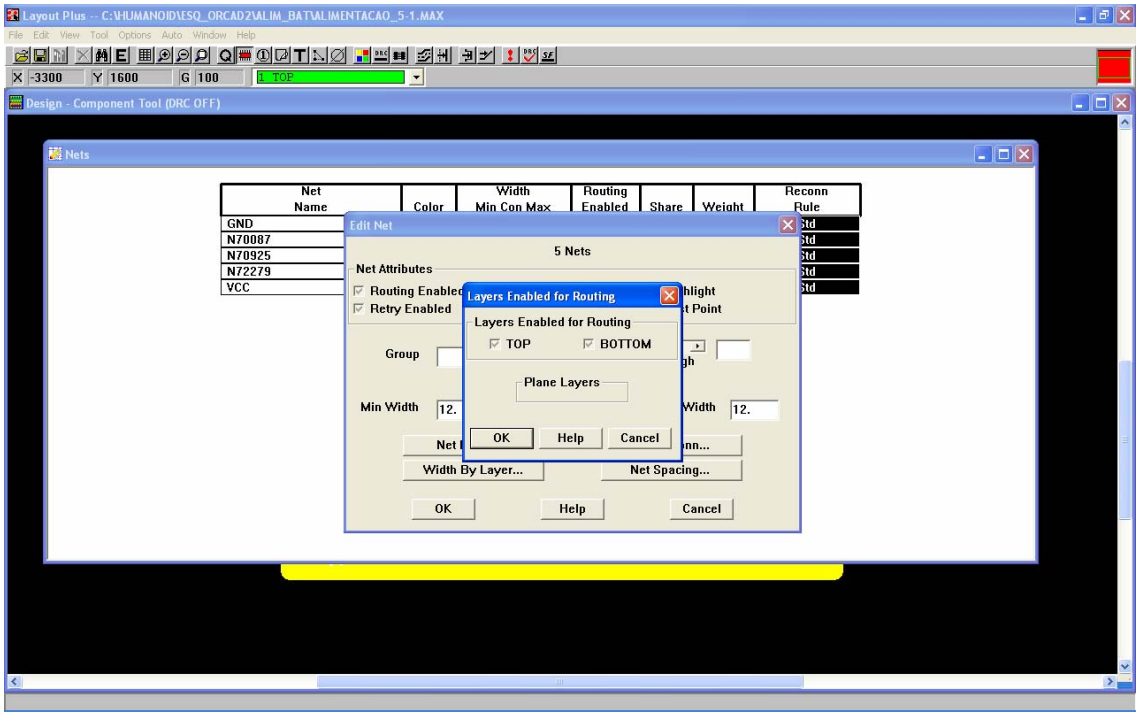
O segundo passo é a configuração das vias de routeamento. Para isso clica-se em *"View Spreadsheet"* → *"Nets"*.



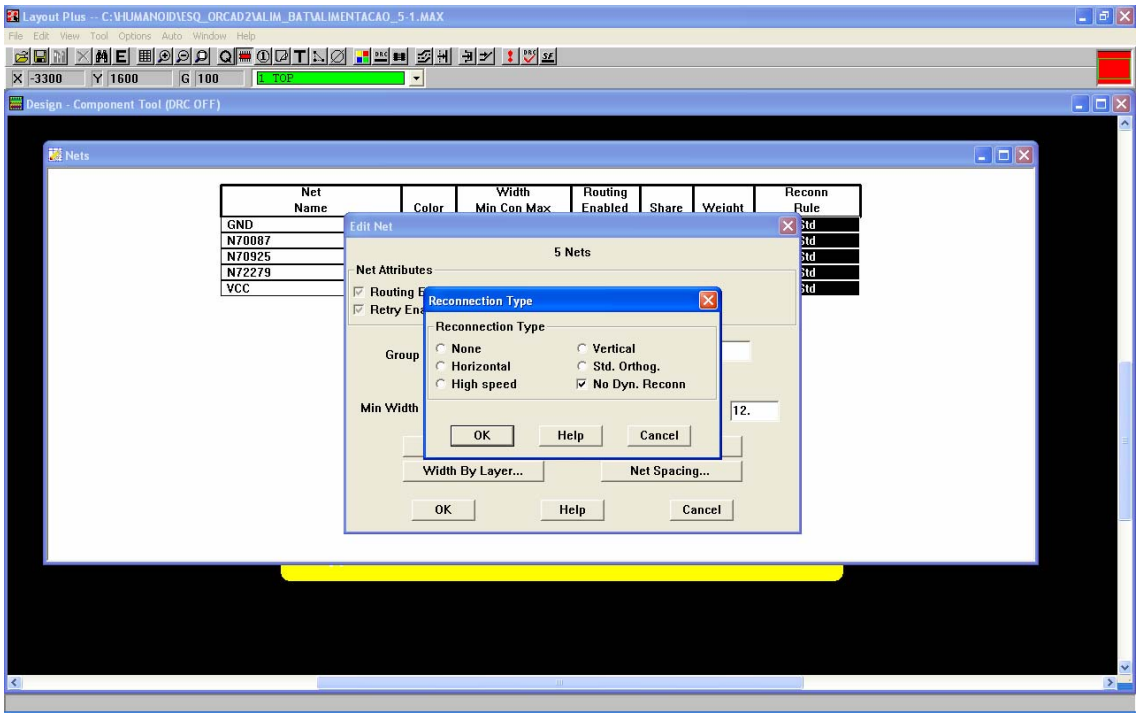
Clique duas vezes em cima do *"Net Name"*.
A espessura da vias é definida aqui com 12 mils.



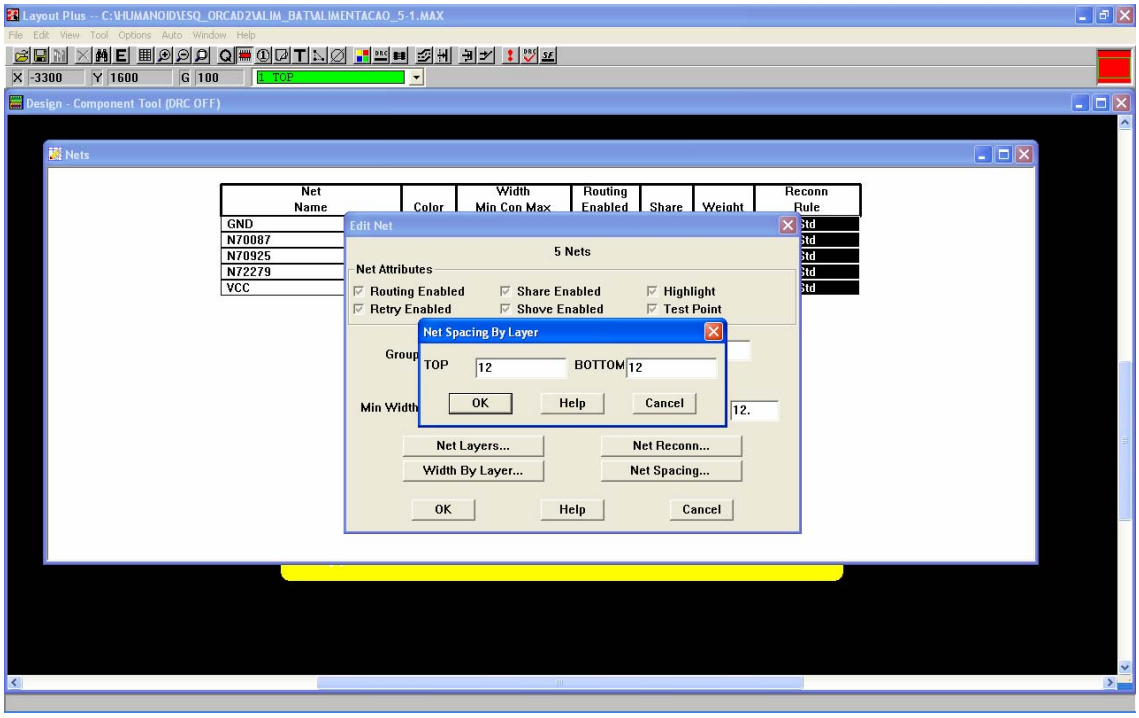
Clique em *"Net layers…"* e verifique que estão para roteamento o *TOP* e *BOTTOM*.



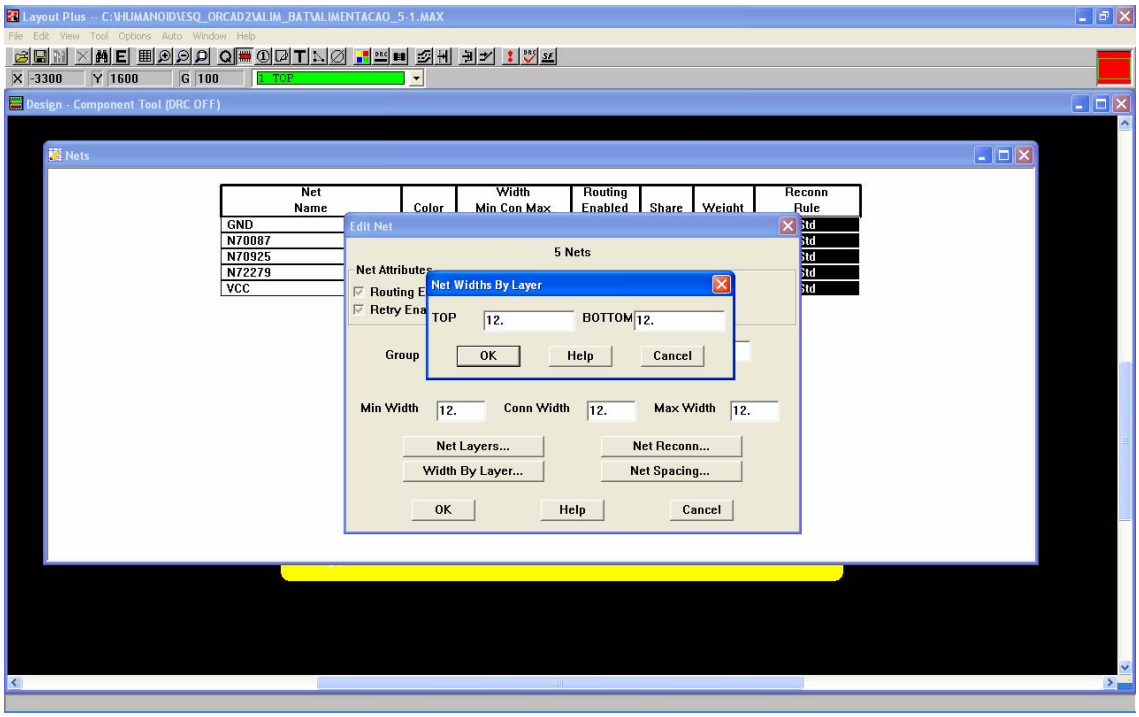
Clique em *"Net reconn…"* reconhecimento dinâmico de deve estar desactivo.



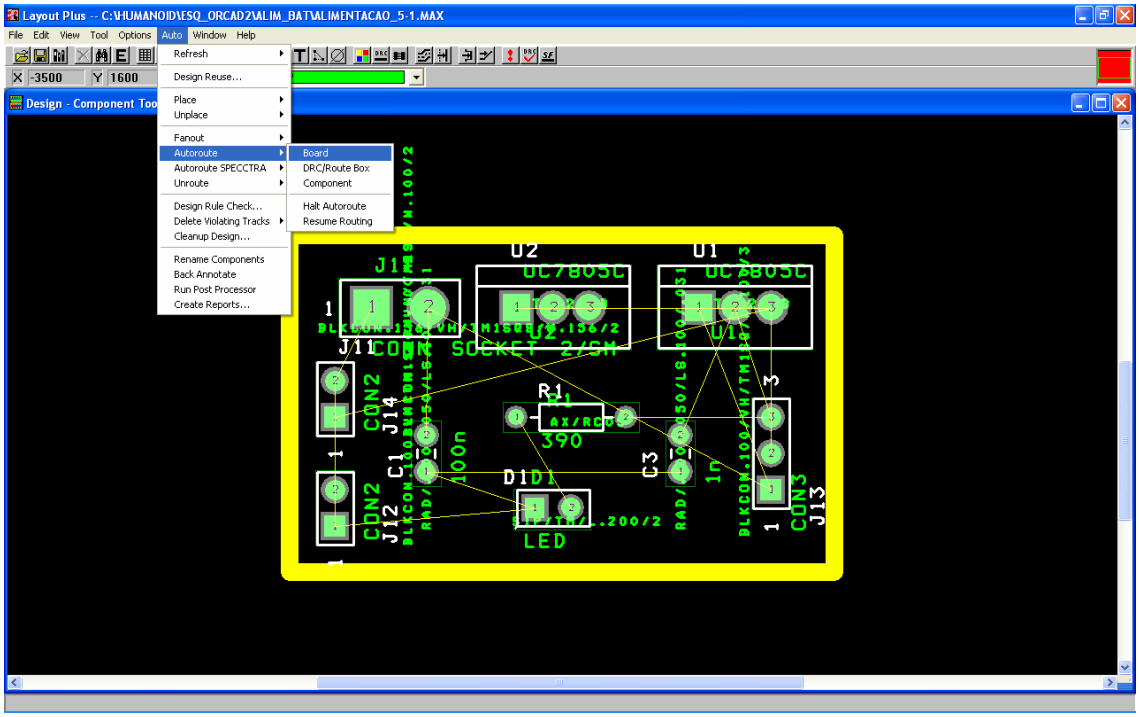
Clique em *"Net Spacing…"*em que o espaçamento entre *Layers* deve ser de 12mils.



Clique em *"Width By Layer…"* aqui introdu-se o espaçamento entre ente vias pontos de ligação que deve ser de 12 mils. Por fim clique em *"OK"*.

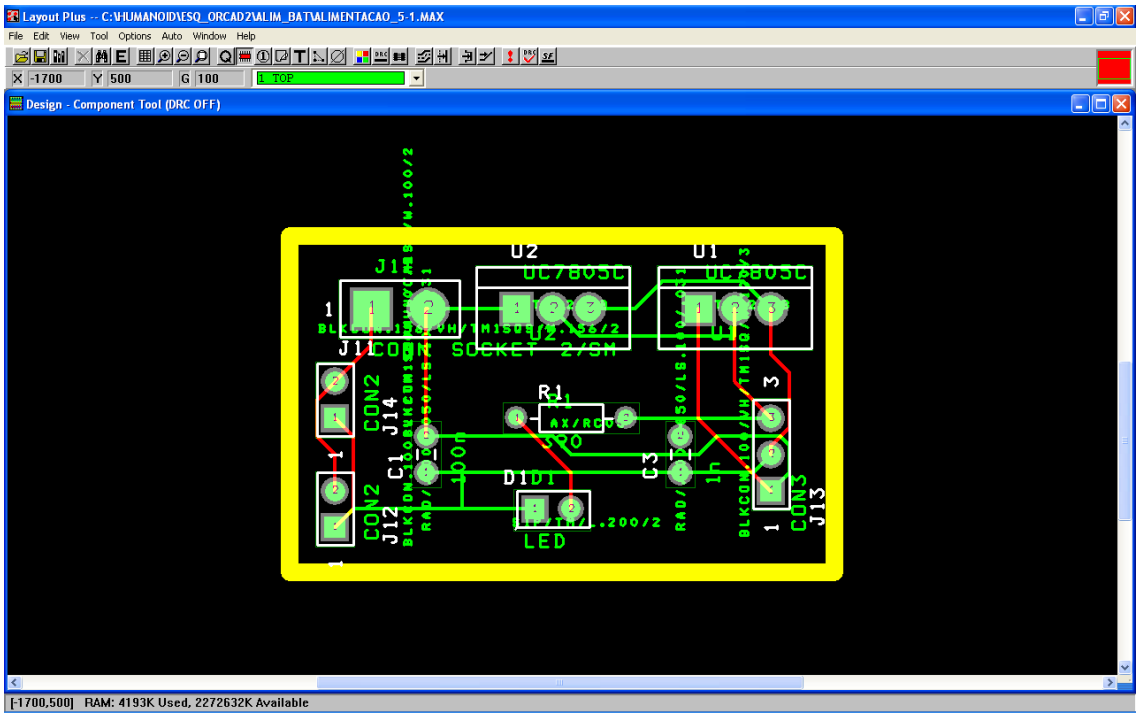


Agora pode-se fazer o roteamento da placa, basta fazer *"Auto"* → *"Autoroute"* → *"Board"*.

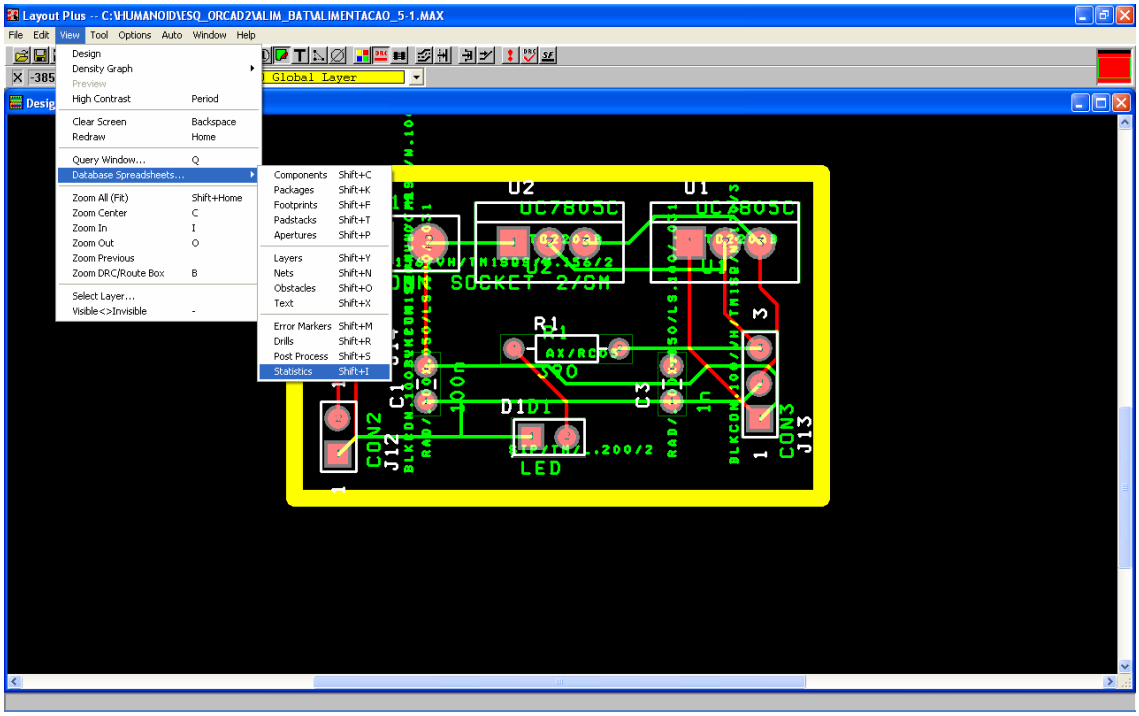


Ao fim de roteada a placa deve ter o seguinte aspecto.

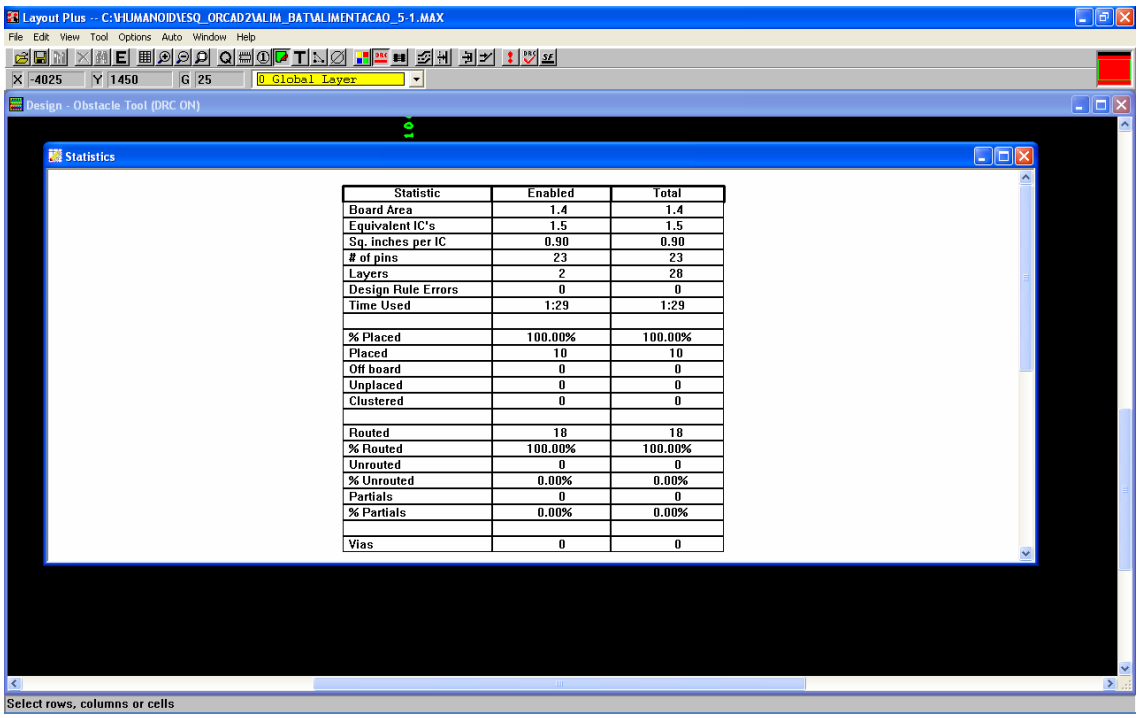
Deve-se clicar na ferramenta *"Design Rule Check"* para ver se não existe nenhum erro de desenho na placa.



Deve-se verificar de a placa ficou totalmente roteada, fazendo *"View"* → *"Database Spreadsheets"* → *"Statistics"*.

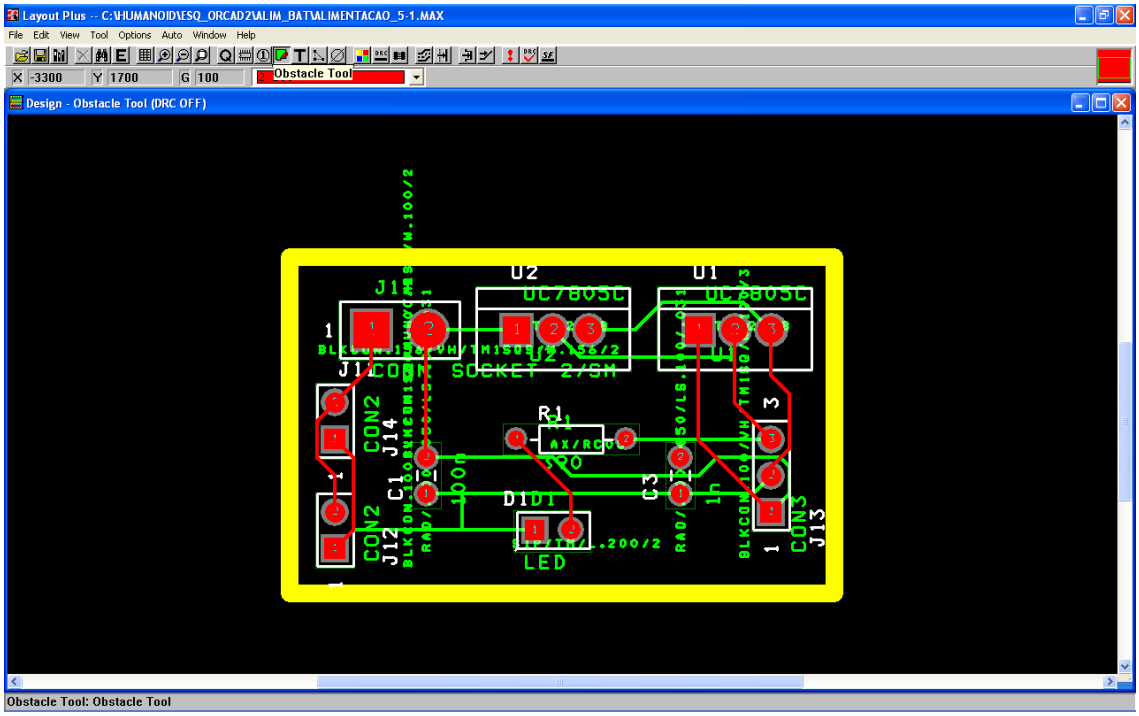


Caso a placa tenha ficado totalmente roteada o campo *%Routed* fica a 100.00%. Caso este campo não esteja a 100.00%, deve fazer *"Auto"* → *"Unroute"* → *"Board"* e voltar a repetir este processo (*"Auto"* → *"Autoroute"* → *"Board"*) até que se encontre completamente roteada. Mesmo assim, se não for possível o roteamento altere as disposições dos componentes e aumente a dimensão da placa o que pode ajudar.

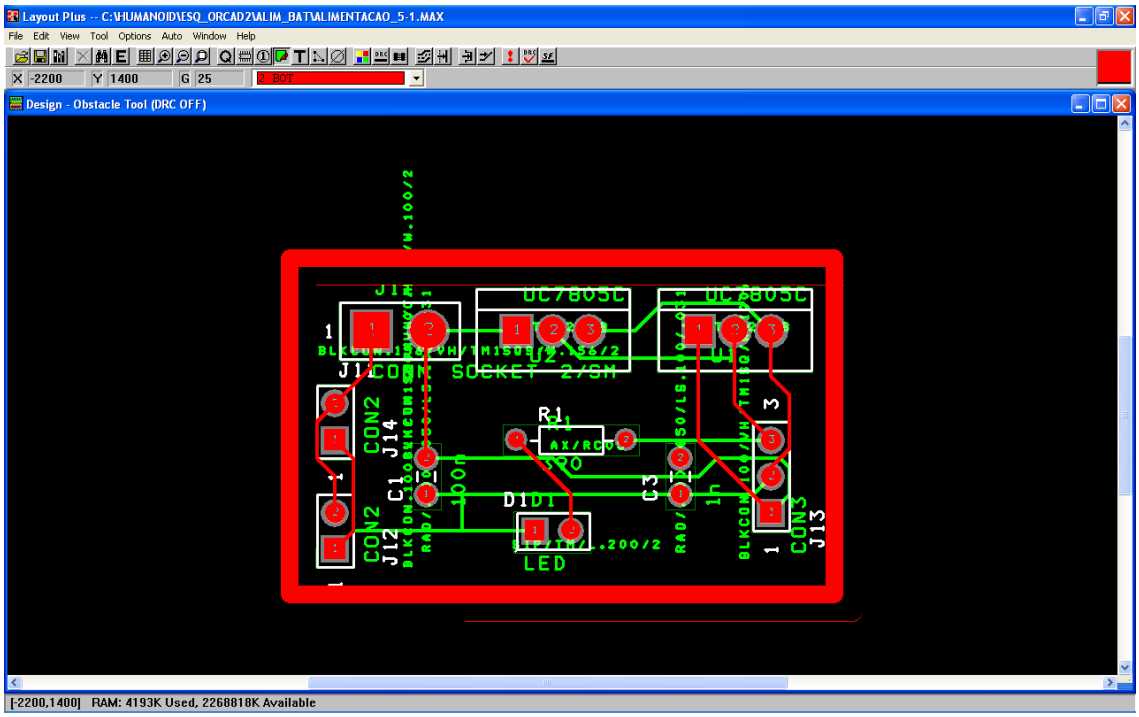


Statistic	Enabled	Total
Board Area	1.4	1.4
Equivalent IC's	1.5	1.5
Sq. inches per IC	0.90	0.90
# of pins	23	23
Layers	2	28
Design Rule Errors	0	0
Time Used	1:29	1:29
% Placed	100.00%	100.00%
Placed	10	10
Off board	0	0
Unplaced	0	0
Clustered	0	0
Routed	18	18
% Routed	100.00%	100.00%
Unrouted	0	0
% Unrouted	0.00%	0.00%
Partials	0	0
% Partials	0.00%	0.00%
Vias	0	0

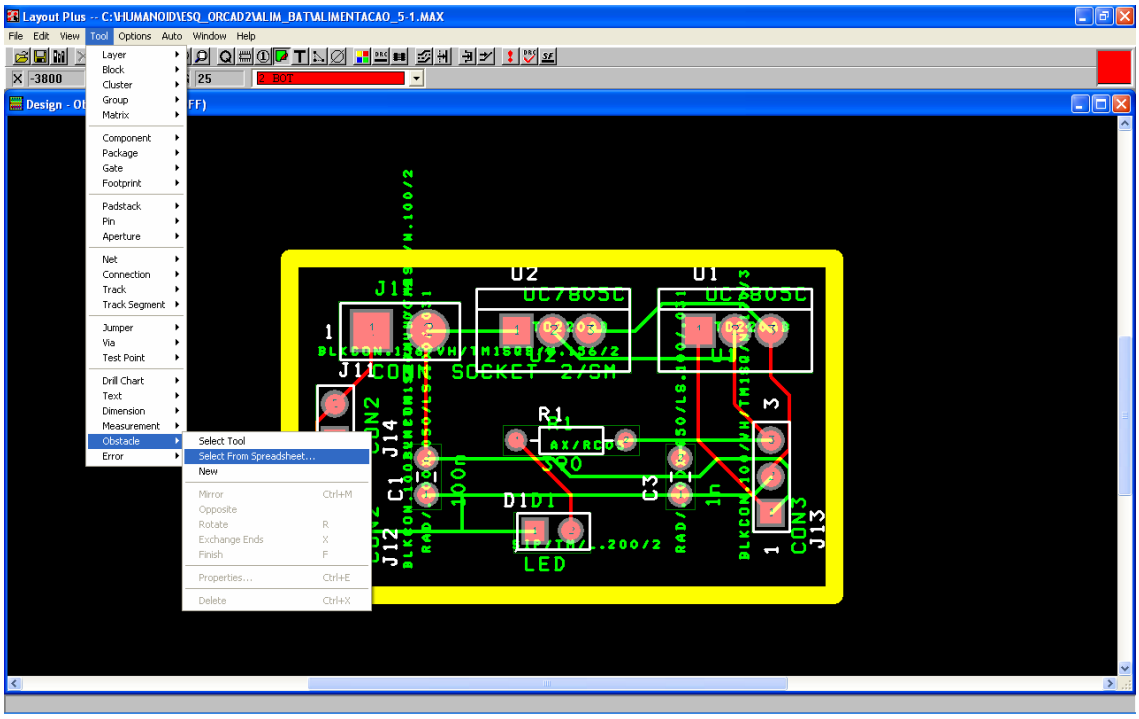
Para a criação de um plano de massa, temos que construir um obstáculo para o plano de massa seleccionando *"2 BOT"* que deve aparecer a vermelho, seleccionando a ferramenta *"Obstacle Tool"* e traça-se os contornos da placa em cima da *" Global Layer"* a amarelo.



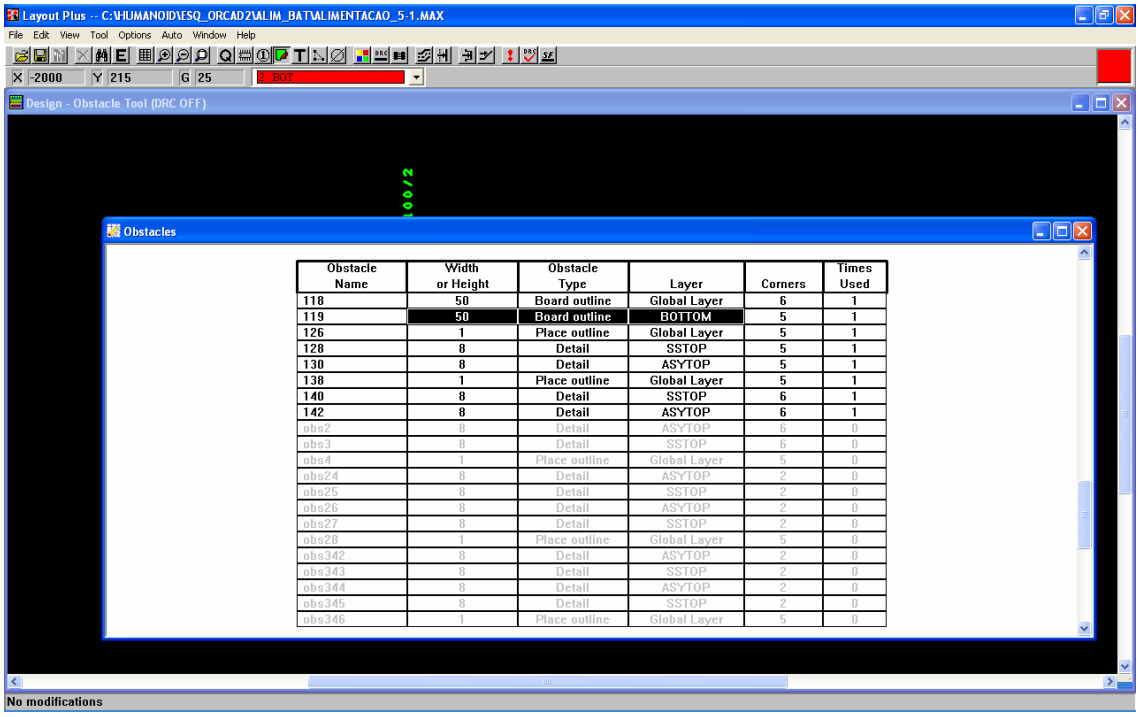
A placa deve ter o seguinte aspecto.



Agora, selecciona-se o *"ObstacleTool"* do plano de massa fazendo os seguintes comandos *"Tool"* → *"Obstacle"* → *"Select From Spreadsheet…"* .

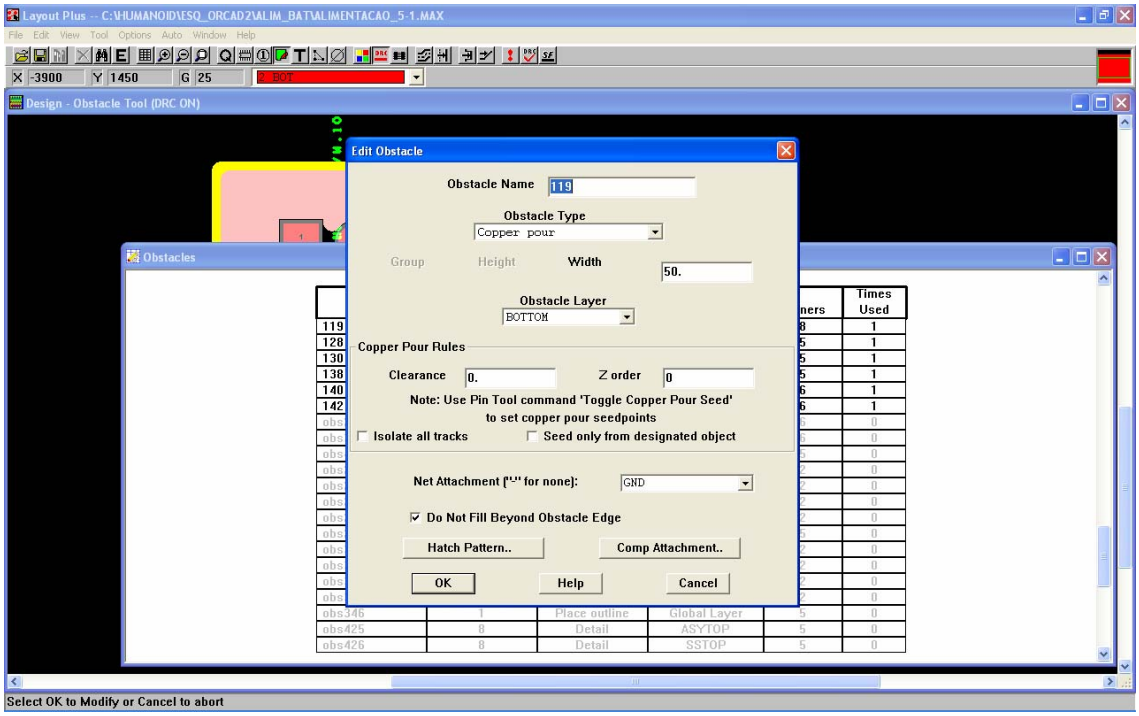


Clique duas vezes sobre "Board outline" "BOTTOM".

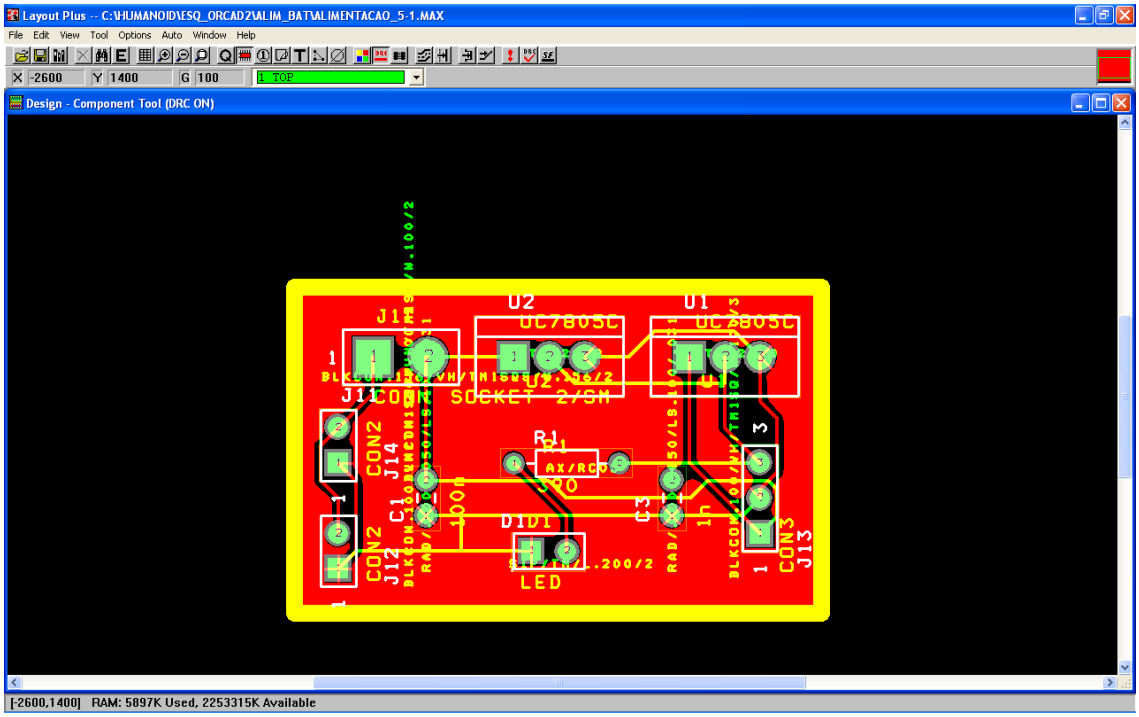


Obstacle Name	Width or Height	Obstacle Type	Layer	Corners	Times Used
118	50	Board outline	Global Layer	6	1
119	50	Board outline	BOTTOM	5	1
126	1	Place outline	Global Layer	5	1
128	8	Detail	SSTOP	5	1
130	8	Detail	ASYTOP	5	1
138	1	Place outline	Global Layer	5	1
140	8	Detail	SSTOP	6	1
142	8	Detail	ASYTOP	6	1
obs2	8	Detail	ASYTOP	6	0
obs3	8	Detail	SSTOP	6	0
obs4	1	Place outline	Global Layer	5	0
obs24	8	Detail	ASYTOP	2	0
obs25	8	Detail	SSTOP	2	0
obs26	8	Detail	ASYTOP	2	0
obs27	8	Detail	SSTOP	2	0
obs28	1	Place outline	Global Layer	5	0
obs342	8	Detail	ASYTOP	2	0
obs343	8	Detail	SSTOP	2	0
obs344	8	Detail	ASYTOP	2	0
obs345	8	Detail	SSTOP	2	0
obs346	1	Place outline	Global Layer	5	0

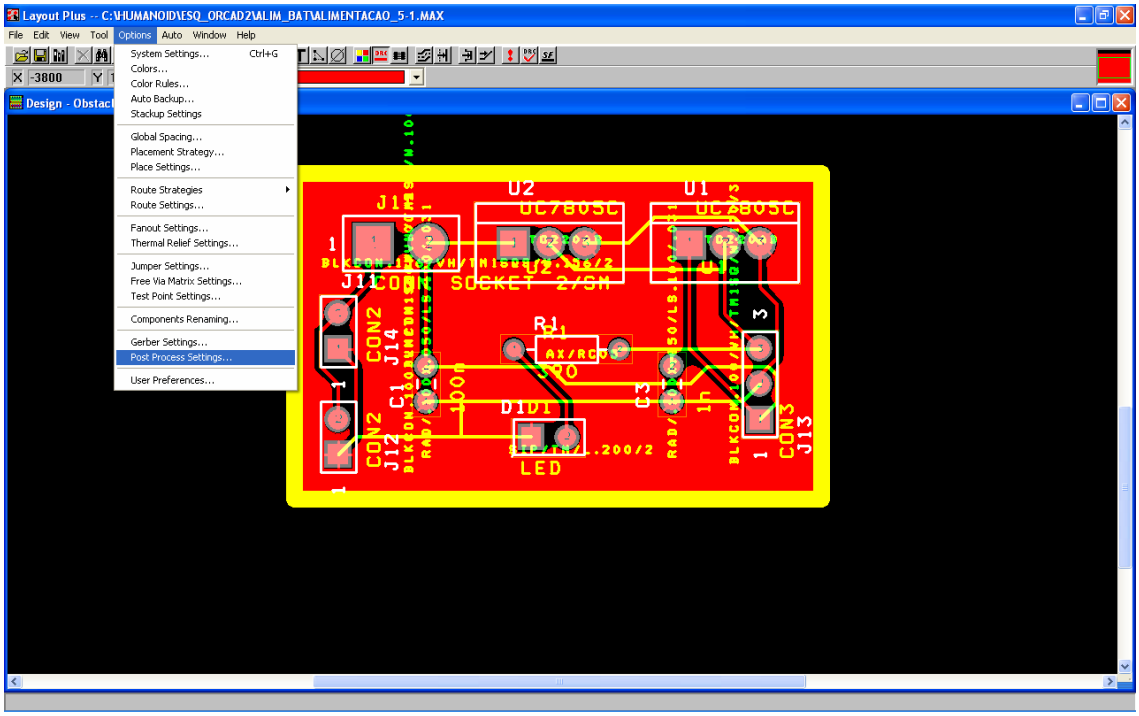
Altere o campo *Obstacle Type* para *"Copper pour"*, o campo *Width* fica com 50 mils, o *Net Attachment ("-" for none)* passa para "*GND"* e clique em *"OK"*.



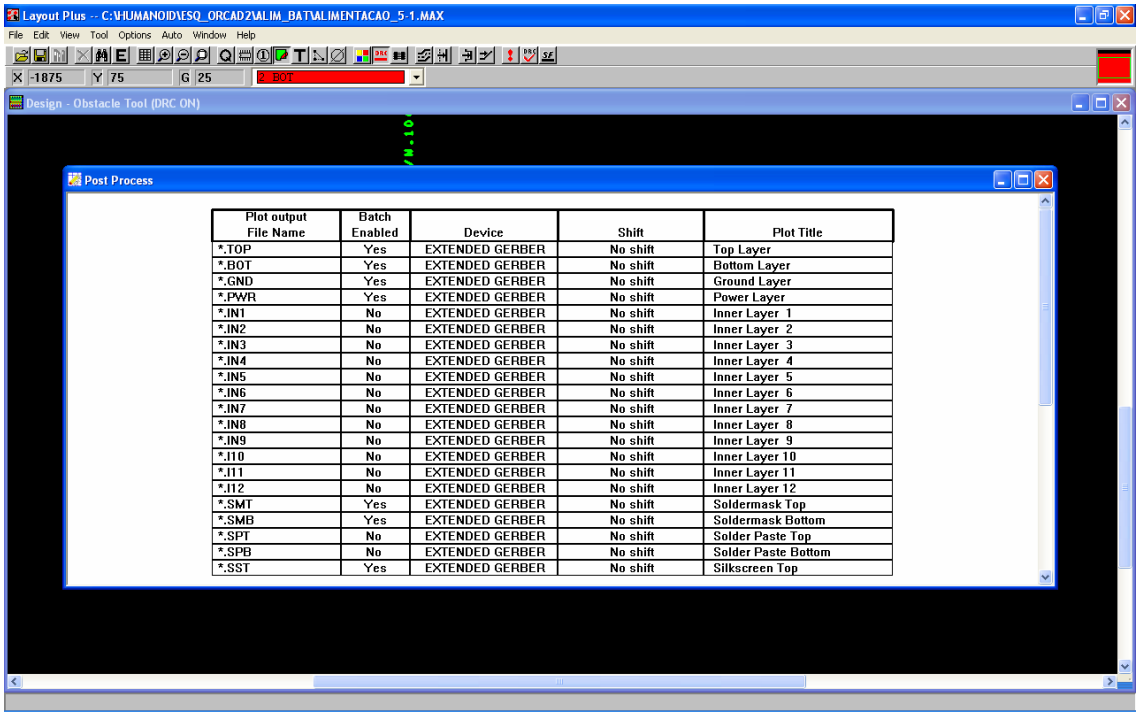
A placa ficará com o seguinte aspecto, os pinos ligados á massa ficam também ligados ao plano de massa.



Por fim, basta criar os ficheiros *gerber* para se poder fazer a placas automaticamente. Dando os ficheiros *gerber* a maquinas especificas de fabrico de PCB's estas fazem automaticamente as placas sem intervenção humana. Contudo, também se pode imprimir a placa e fazer as placas manualmente. Fazendo "Options" → *"Post Process Settigs…"* edita-se as características do dispositivo.

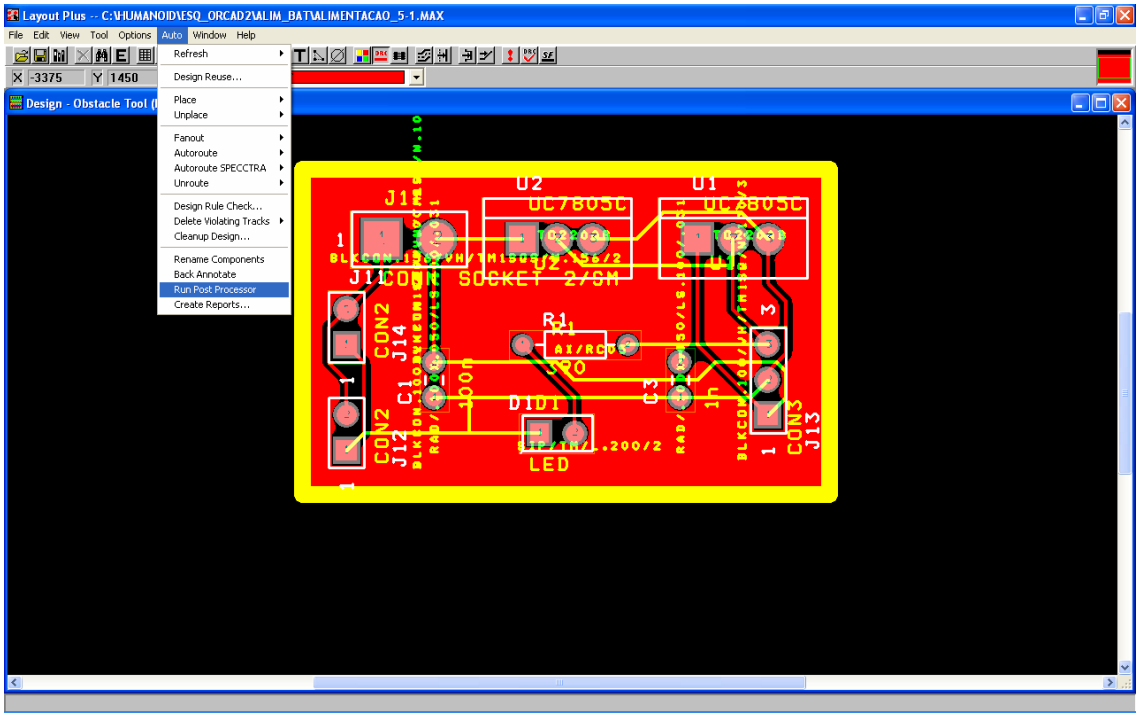


Verifica-se, que estão seleccionados os ficheiros *EXTENDED GERBER* (RS-274X) pois são estes os mais utilizados na produção industrial de PCB's, também se pode utilizar os ficheiros GERBER RS-274D para isso basta clicar duas vezes no campo *device*. Por fim feche a janela.

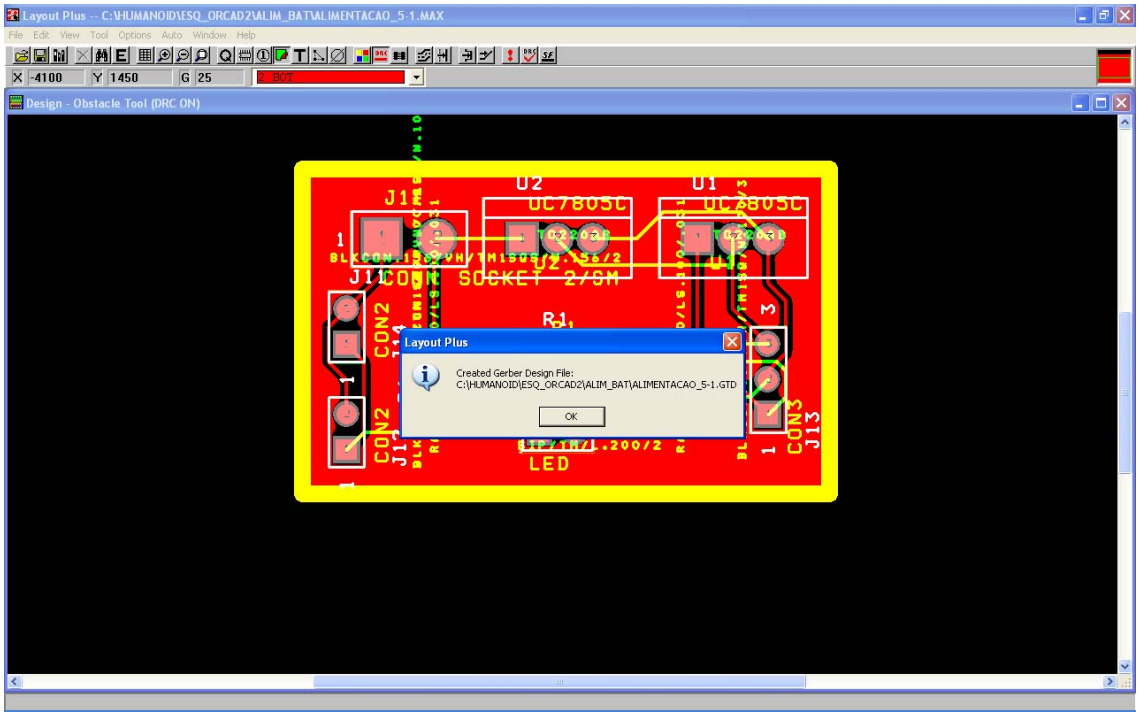


Plot output File Name	Batch Enabled	Device	Shift	Plot Title
*.TOP	Yes	EXTENDED GERBER	No shift	Top Layer
*.BOT	Yes	EXTENDED GERBER	No shift	Bottom Layer
*.GND	Yes	EXTENDED GERBER	No shift	Ground Layer
*.PWR	Yes	EXTENDED GERBER	No shift	Power Layer
*.IN1	No	EXTENDED GERBER	No shift	Inner Layer 1
*.IN2	No	EXTENDED GERBER	No shift	Inner Layer 2
*.IN3	No	EXTENDED GERBER	No shift	Inner Layer 3
*.IN4	No	EXTENDED GERBER	No shift	Inner Layer 4
*.IN5	No	EXTENDED GERBER	No shift	Inner Layer 5
*.IN6	No	EXTENDED GERBER	No shift	Inner Layer 6
*.IN7	No	EXTENDED GERBER	No shift	Inner Layer 7
*.IN8	No	EXTENDED GERBER	No shift	Inner Layer 8
*.IN9	No	EXTENDED GERBER	No shift	Inner Layer 9
*.I10	No	EXTENDED GERBER	No shift	Inner Layer 10
*.I11	No	EXTENDED GERBER	No shift	Inner Layer 11
*.I12	No	EXTENDED GERBER	No shift	Inner Layer 12
*.SMT	Yes	EXTENDED GERBER	No shift	Soldermask Top
*.SMB	Yes	EXTENDED GERBER	No shift	Soldermask Bottom
*.SPT	No	EXTENDED GERBER	No shift	Solder Paste Top
*.SPB	No	EXTENDED GERBER	No shift	Solder Paste Bottom
*.SST	Yes	EXTENDED GERBER	No shift	Silkscreen Top

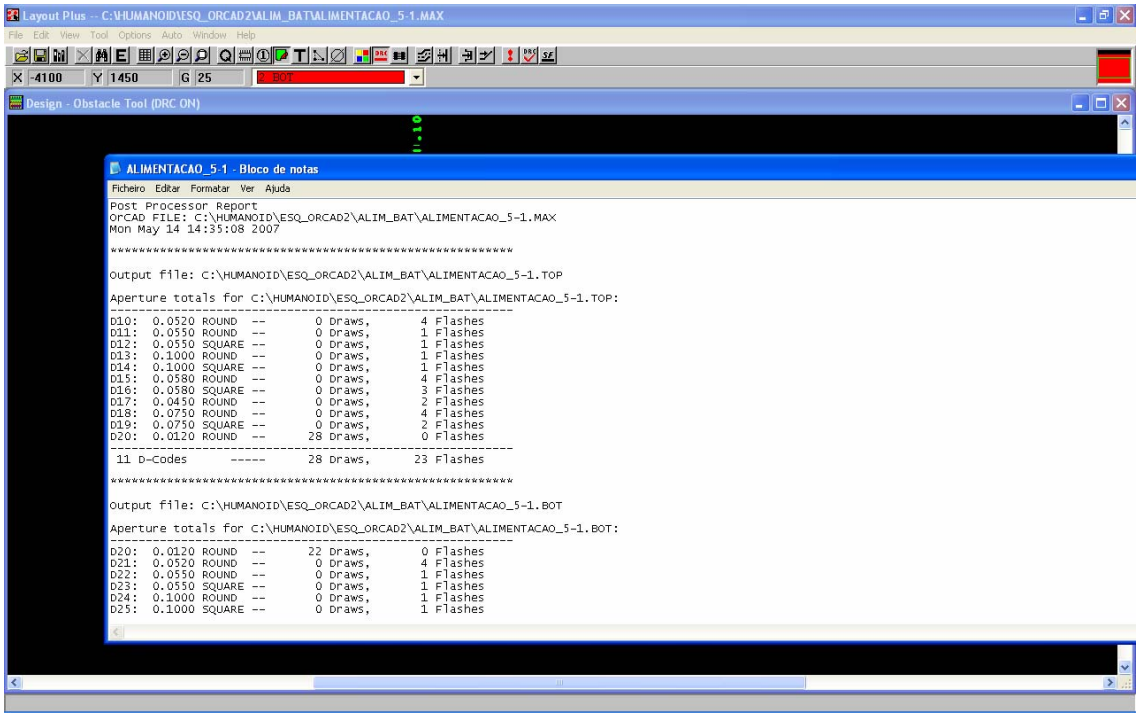
Agora tem que executar o as defenições alteradas, fazendo *"Auto"→ "Run Post Processor"*.



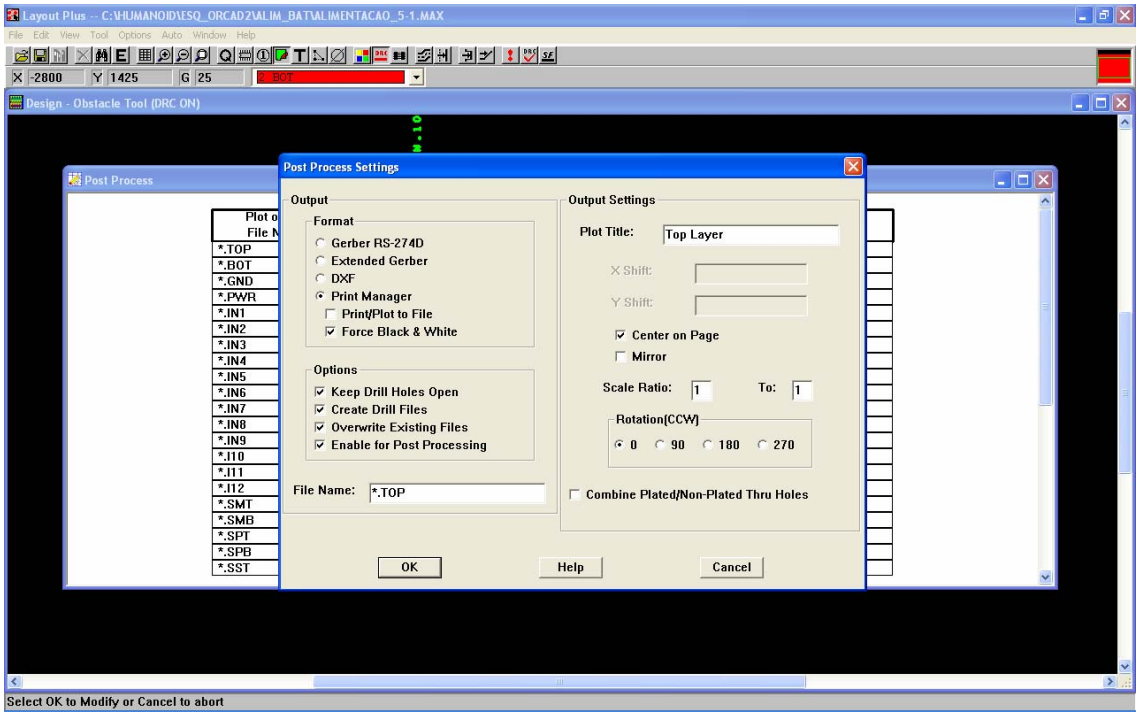
Este vai dizer a onde os ficheiros vão ser guardados. Clique *"OK"*.



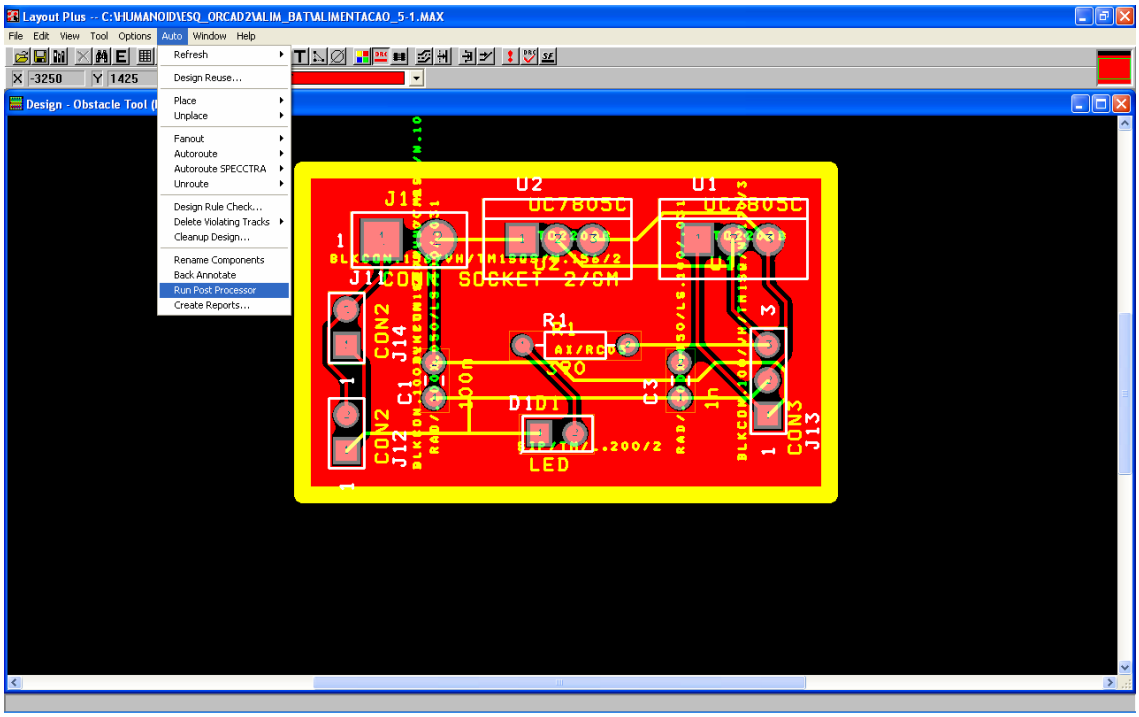
E aparecerá o relatório do ficheiro *gerber* criado.



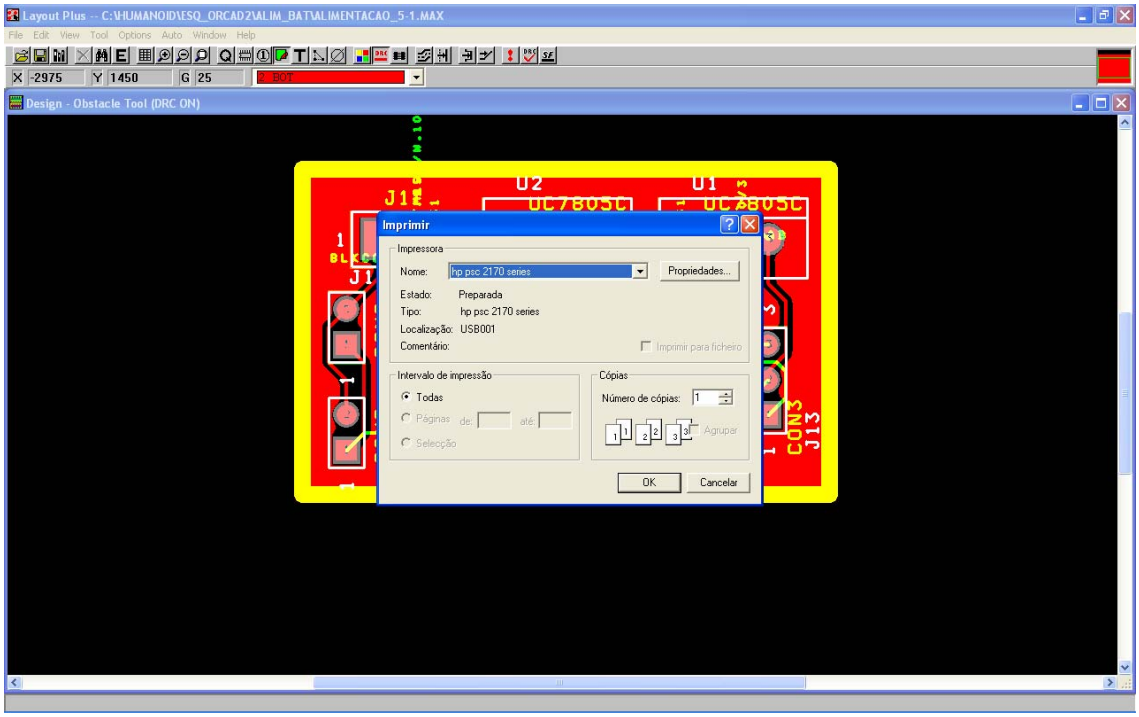
Para imprimir uma folha de acetato por exemplo faça *"Options"* → *"Post Process Settigs…"*, clique duas vezes sobre o *layer* a imprimir e defina *"Print Manager"* → *"Force Black & White"* → *"OK"*.



Finalmente, faça "Auto"→ "Run Post Processor".



E devem aparecer as características da impressora, verifique as "*Propriedades…"* e clique *"OK"*.



Anexo 5: Tutorial de KDevelop

KDevelop para Linux

O programa KDevelop teve início em 1998 com o intuito de se construir um IDE Integrated Development Enviroment (Ambiente de Desenvolvimento Integrado) com extrema fácilidade de usar para o KDE. Desde então, o IDE KDevelop está disponível publicamente sob a licença GPL (Gnu Public License - Licença Pública Gnu) e suporta várias linguagens de programação.

Actualmente, o KDevelop tem vindo a ter uma grande importância no desenvolvimento e construção de projectos em Linux. Este tem uma enorme simplicidade na gestão de ficheiros, na edição do código fonte, na criação de ficheiros executáveis, na criação ficheiros de instalação e ainda traz ferramentas de integração de controlo de versões.

Adicionalmente a este existem o Automake, o Autoconf e o Libtool que tem de ser instalados com o KDevelop. Estes têm por objectivo os criar os ficheiros de compilação e configuração automática sem intervenção do utilizador. A plataforma QMake chama-se a si próprio "é uma ferramenta para criar *Makefiles* para os vários compiladores e plataformas".

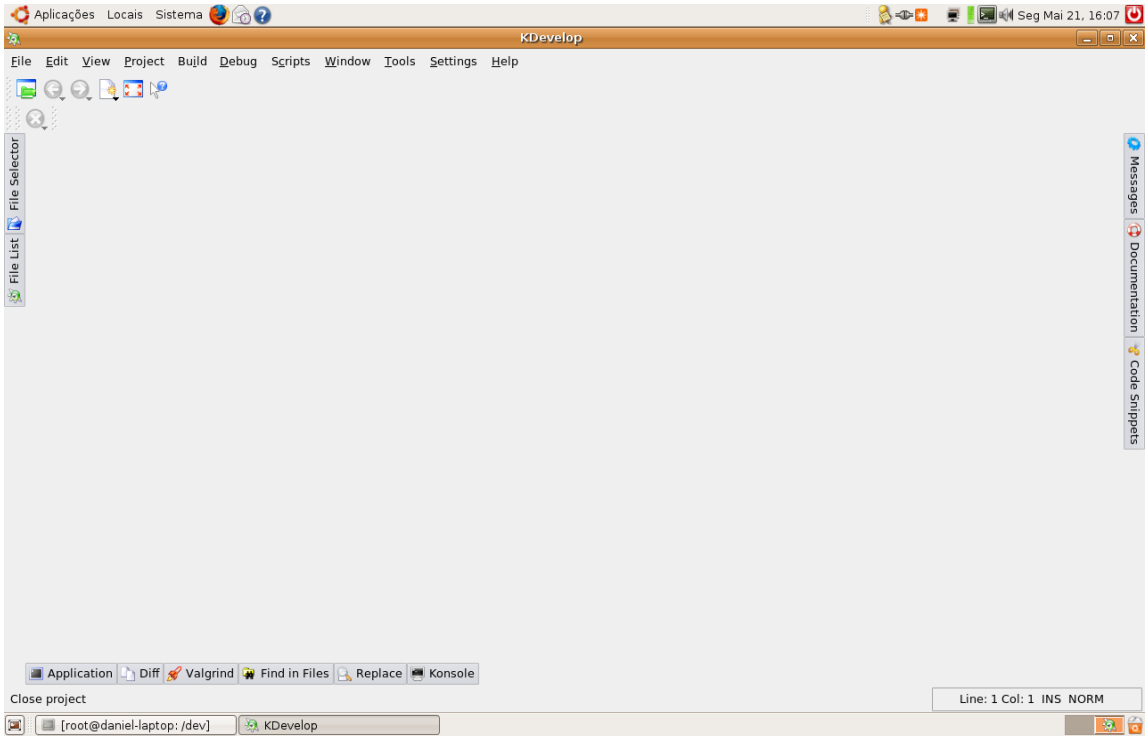
Antes da utilização instale:

- Automake
- Autoconf
- Libtool
- g++
- gcc
- Qt3 ou Qt4
- KDevelop

É aconselhável, instalar o Obuntu pois este sistema operativo traz um gestor de pacotes, onde, se pode seleccionar todos os pacotes requeridos de maneira simples e sem grandes riscos para o sistema.

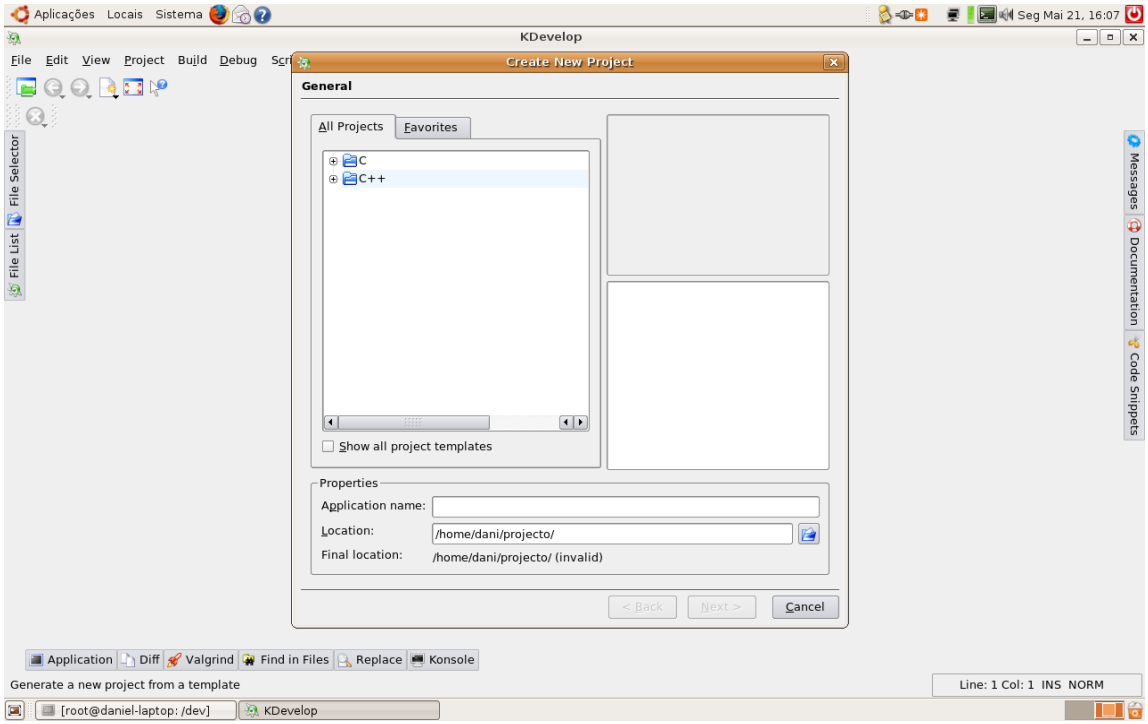
Execute os seguintes passos:

"Aplicações" → *"Desenvolvimento"* → *"Kdevelop: C/C++"*

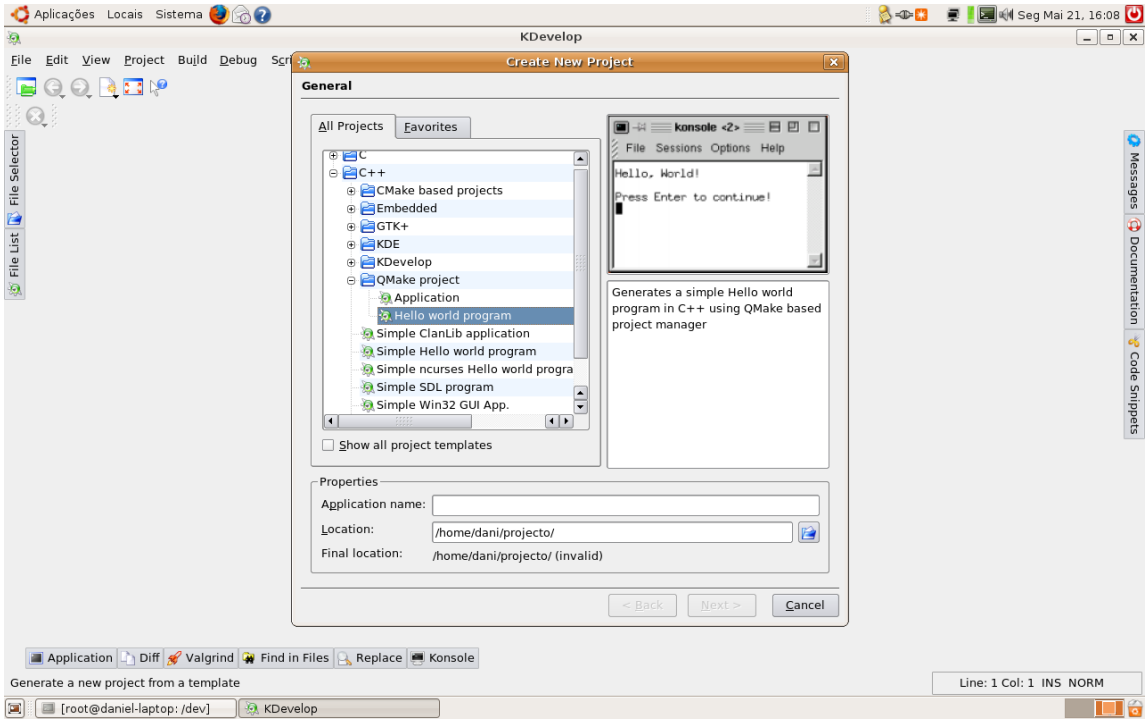


Seleccione a linguagem de programação:

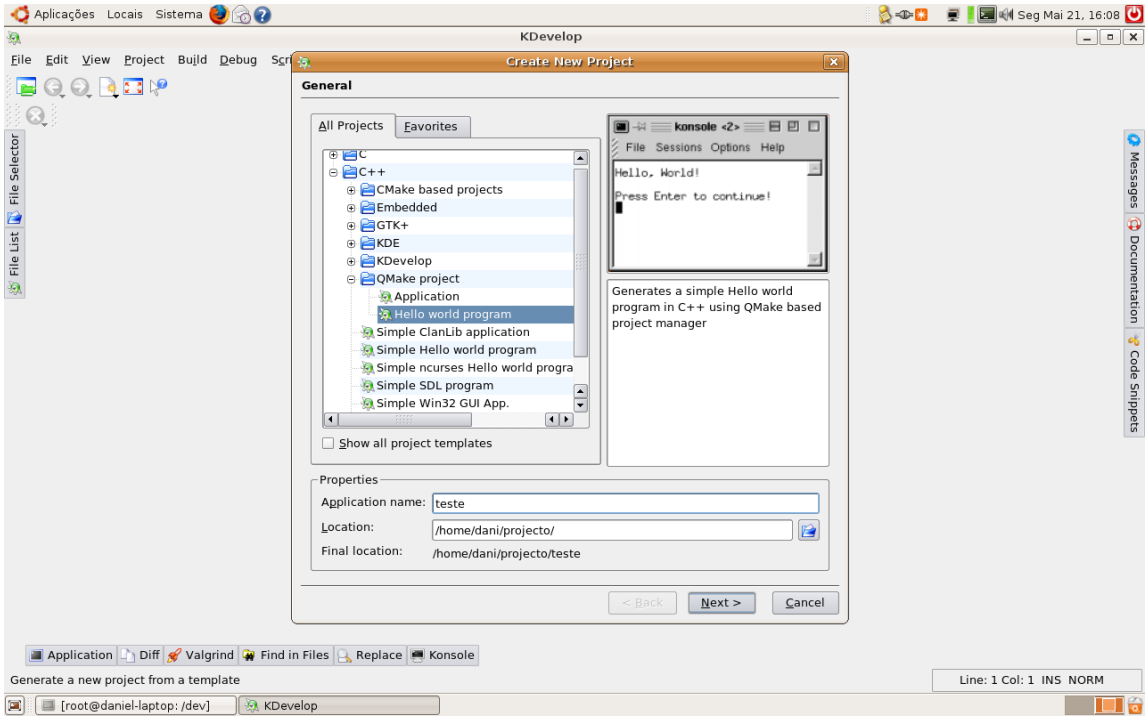
Clique no expansor de *"C++"*, caso seja C++ a linguagem de programação.



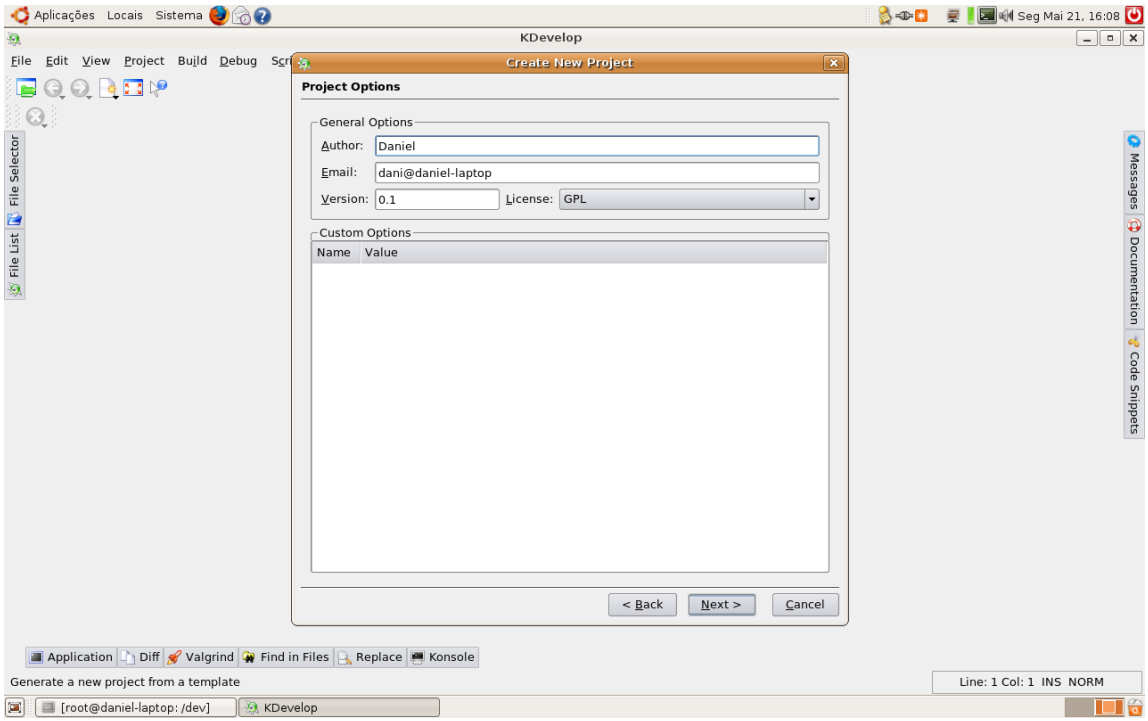
Clique no expansor de "QMake project" →"Hello wordl program".



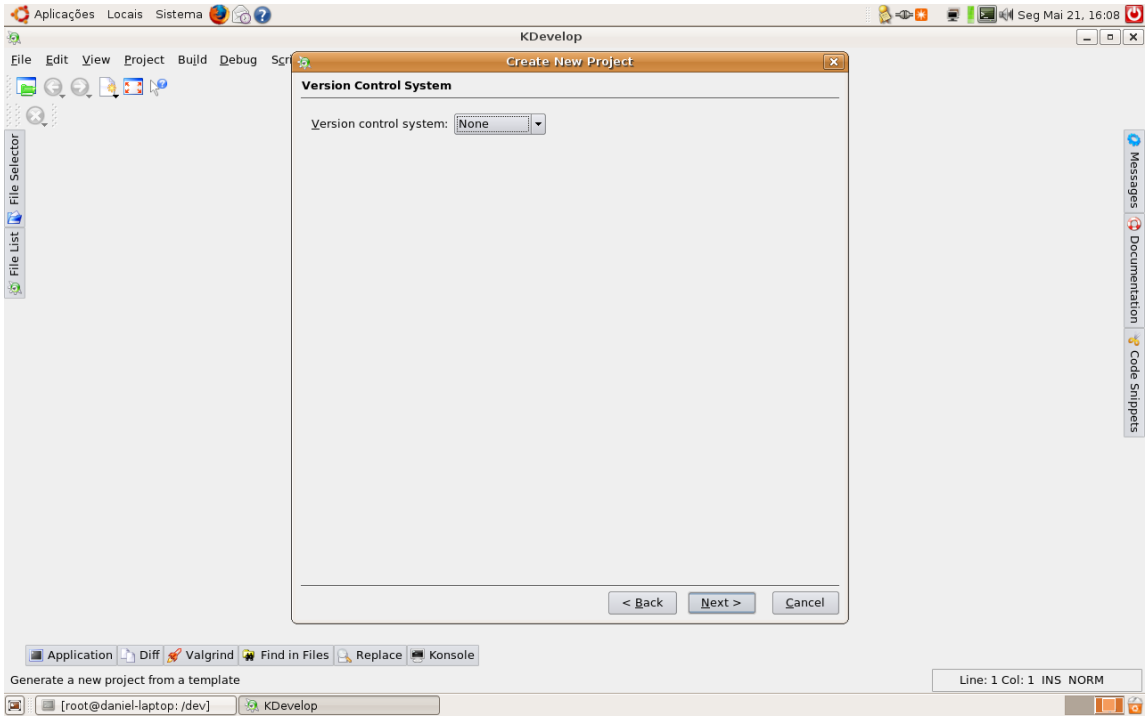
Digite o nome do projecto no campo *"Application name"* e clique *"Next >"*.



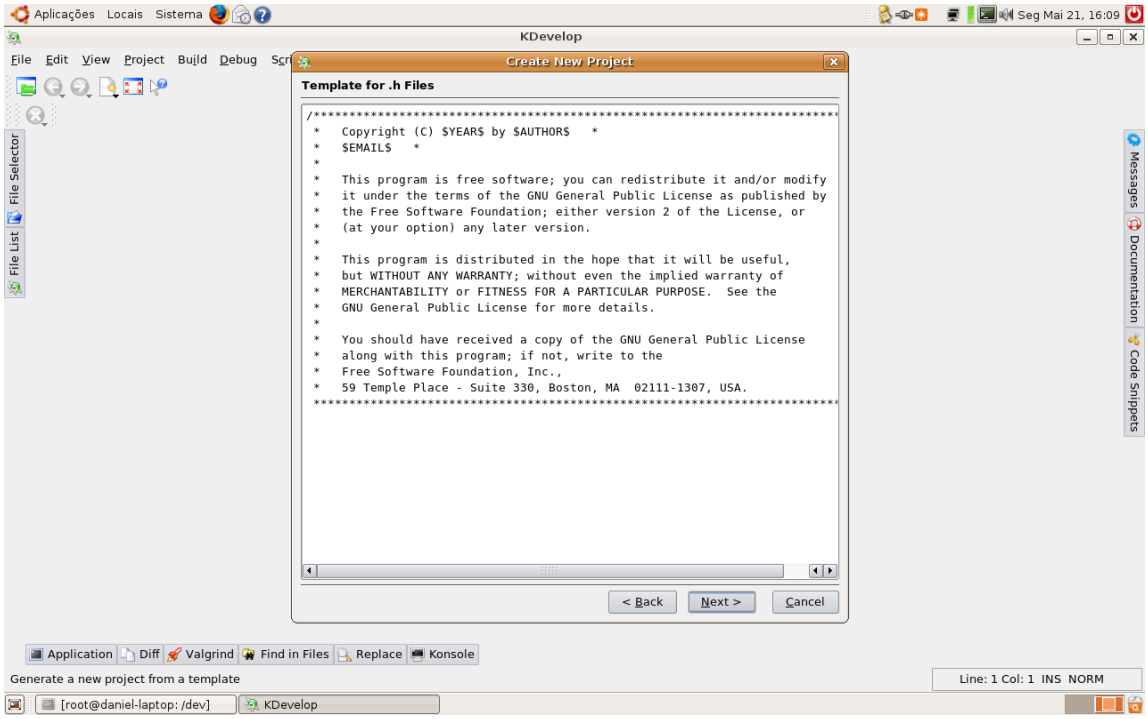
Nesta janela pode introduzir o Autor, Email e a versão e clique em *"Next >"*.



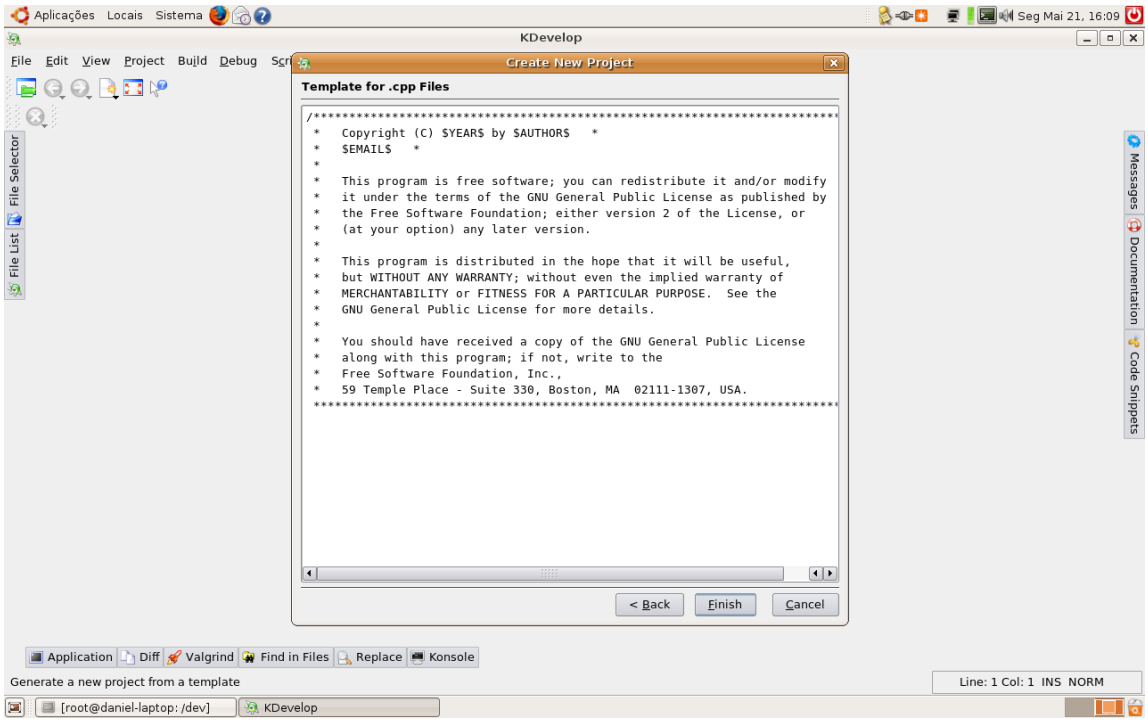
Nesta janela pode introduzir o tipo de controlo de versões clique em *"Next >"*.



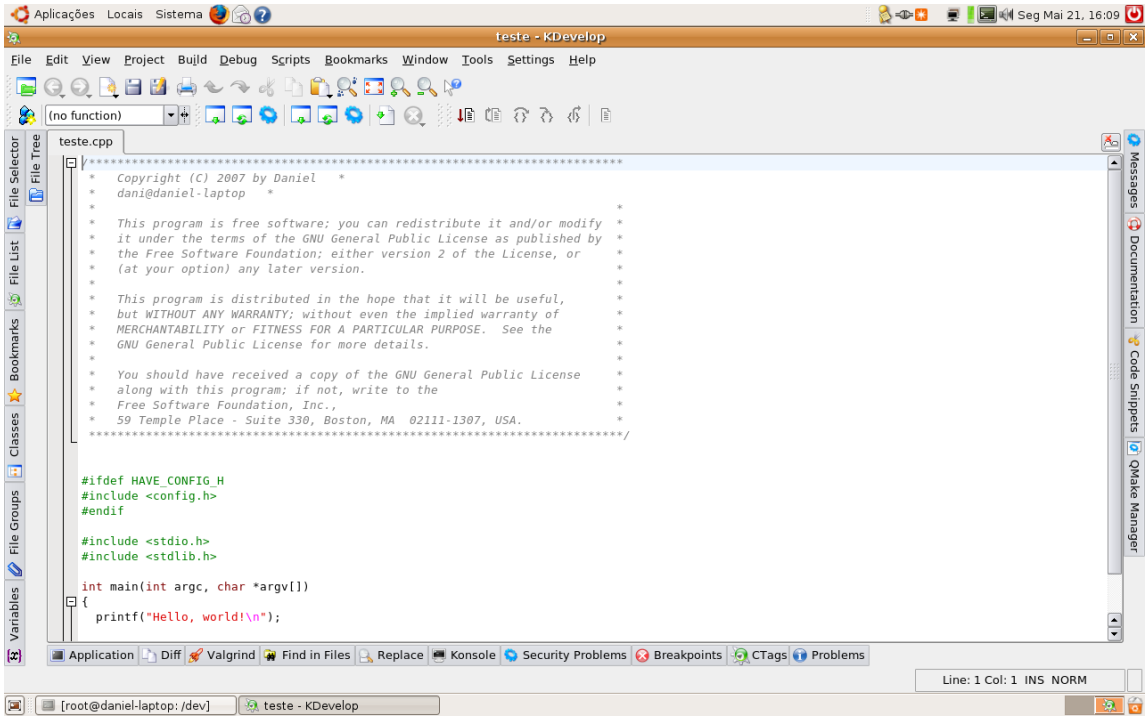
Nesta janela pode introduzir o cabeçalho dos ficheiros *"*.h"* clique em *"Next >"*.



Nesta janela pode introduzir o cabeçalho dos e descrição ficheiros *"*.cpp"* clique em *"Next >"*.

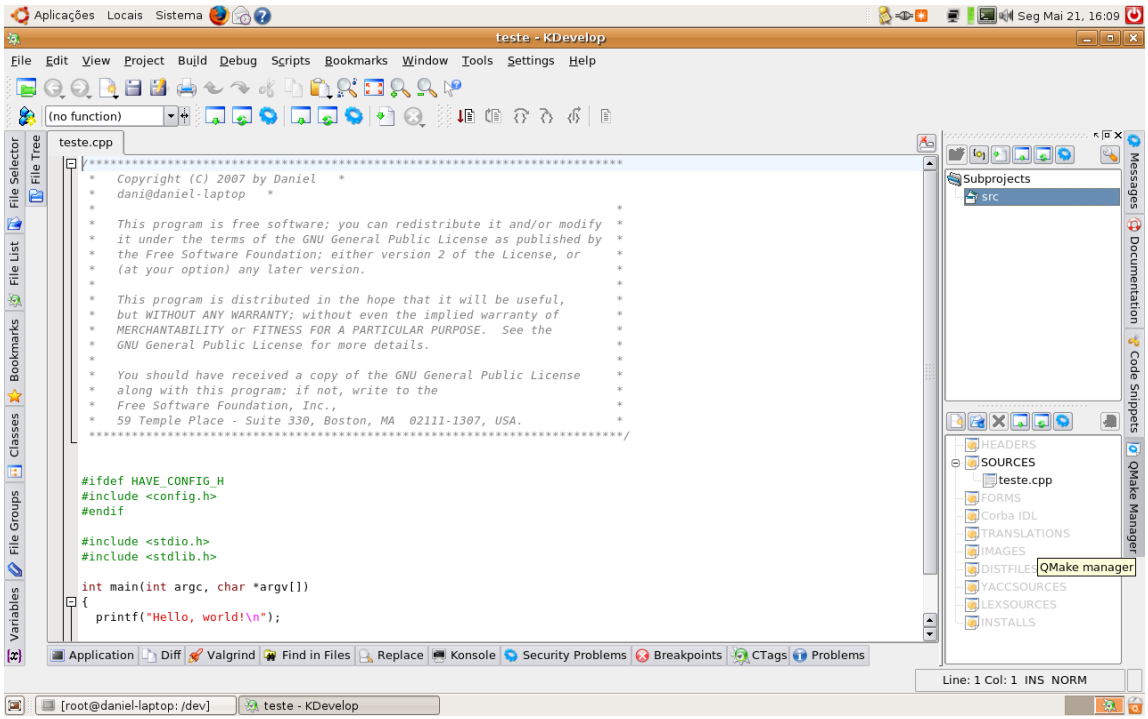


Neste ponto, está ponto para executar o projecto. Faça *"Bulid"* → *"Bulid Project"* e *"Bulid"* → *"Execute Main Program"*.

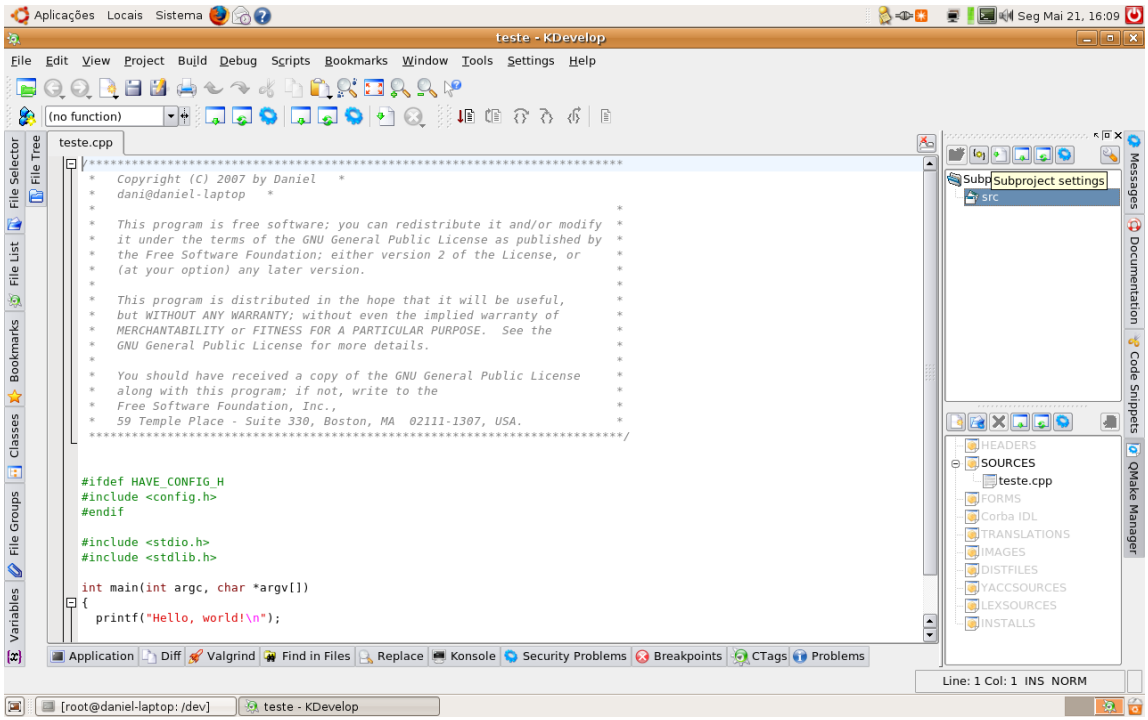



```
/***************************************************************************
 *   Copyright (C) 2007 by Daniel   *
 *   dani@daniel-laptop   *
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 2 of the License, or     *
 *   (at your option) any later version.                                   *
 *                                                                         *
 *   This program is distributed in the hope that it will be useful,       *
 *   but WITHOUT ANY WARRANTY; without even the implied warranty of        *
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the         *
 *   GNU General Public License for more details.                          *
 *                                                                         *
 *   You should have received a copy of the GNU General Public License     *
 *   along with this program; if not, write to the                         *
 *   Free Software Foundation, Inc.,                                       *
 *   59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.             *
 ***************************************************************************/


#ifdef HAVE_CONFIG_H
#include <config.h>
#endif

#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
  printf("Hello, world!\n");
```

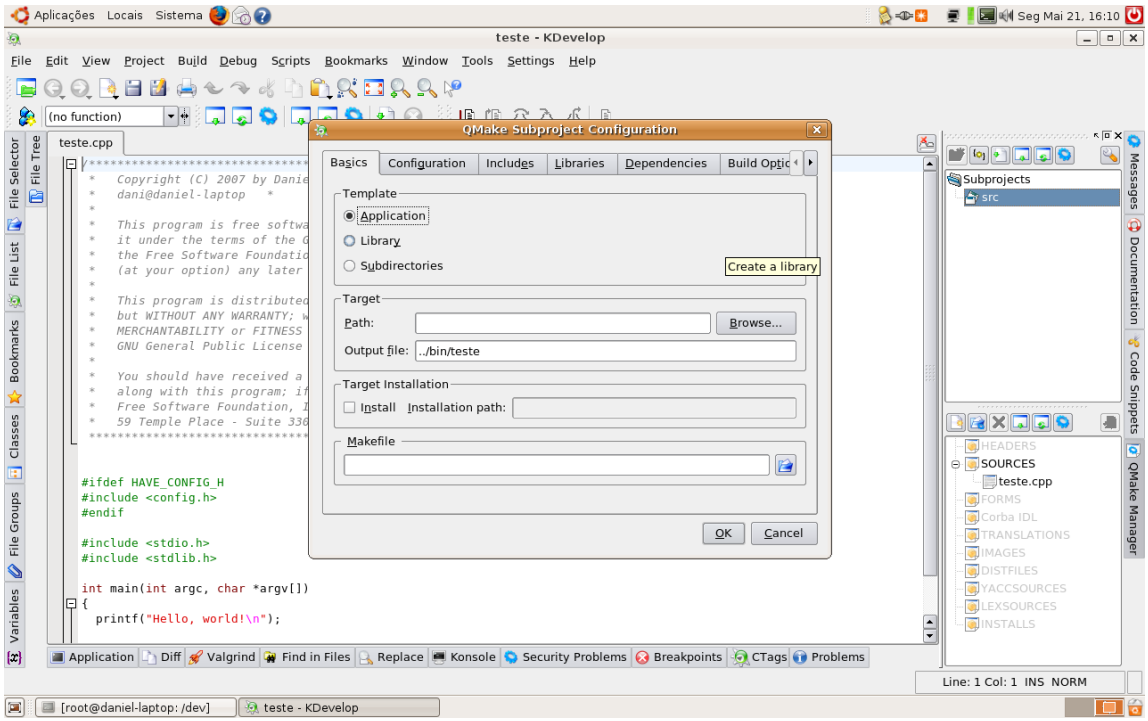
Caso seja necessário, adicionar livrarias ao projecto ou novos ficheiros ao projecto, abra o *"QMake Manager"* que se encontra do lado direito do KDevelop.



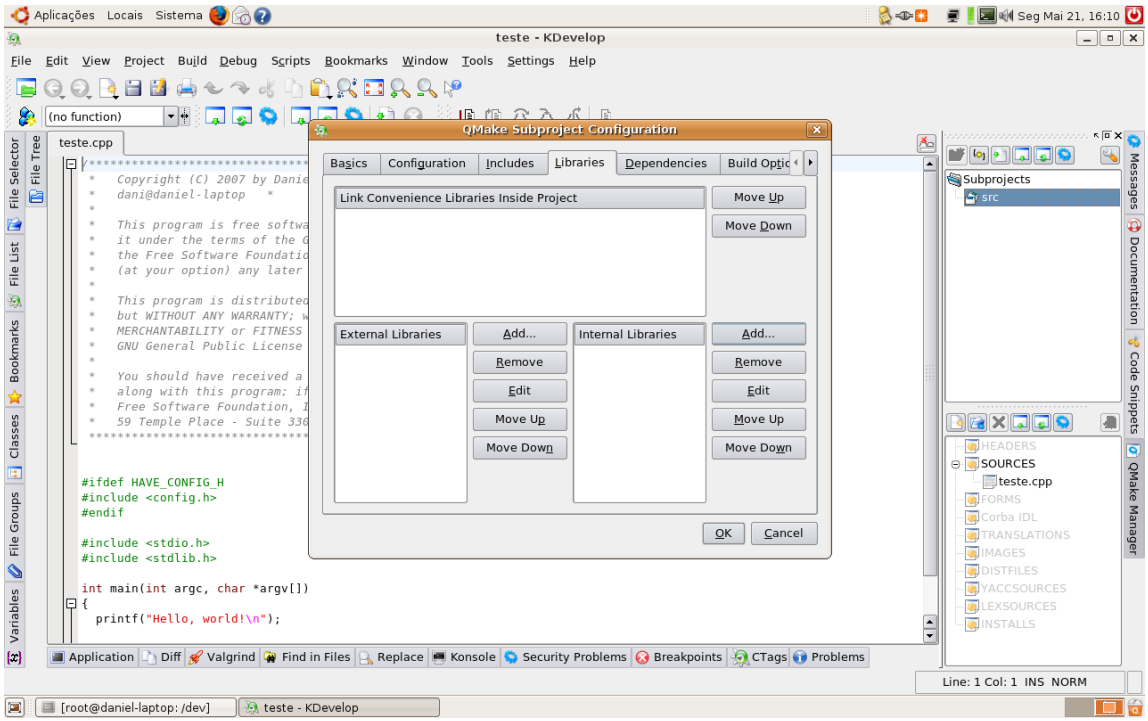
Para adicionar livrarias ao projecto, clique na ferramenta *"Subproject settings"* (com o símbolo de uma chave de bocas).



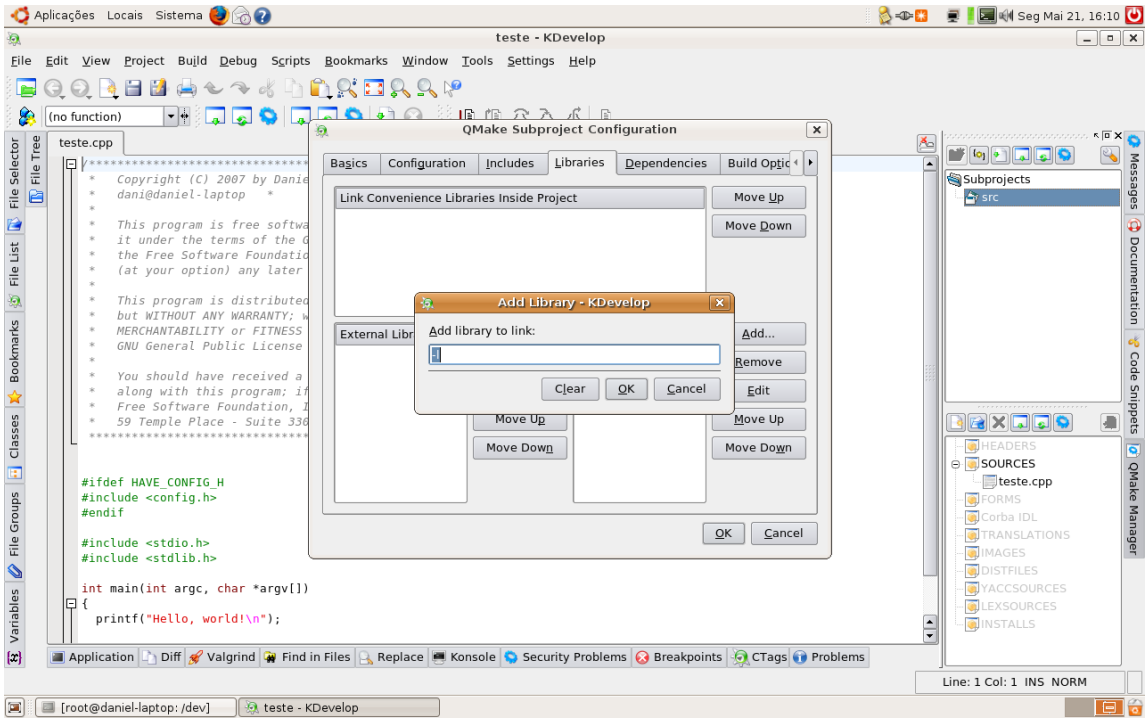
Aparecerá uma janela com o seguinte formato. Clique no separador *"Libraries"*.



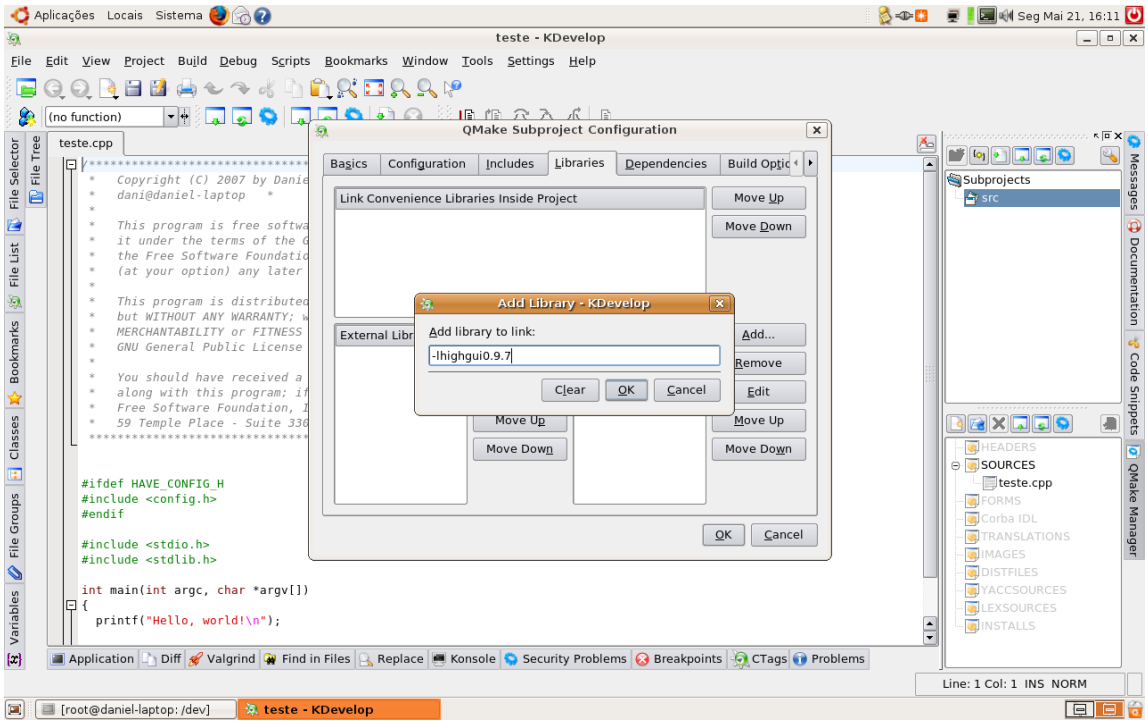
Para adicionar uma livraria clique no botão *"Add..."* do lado direito da janela.



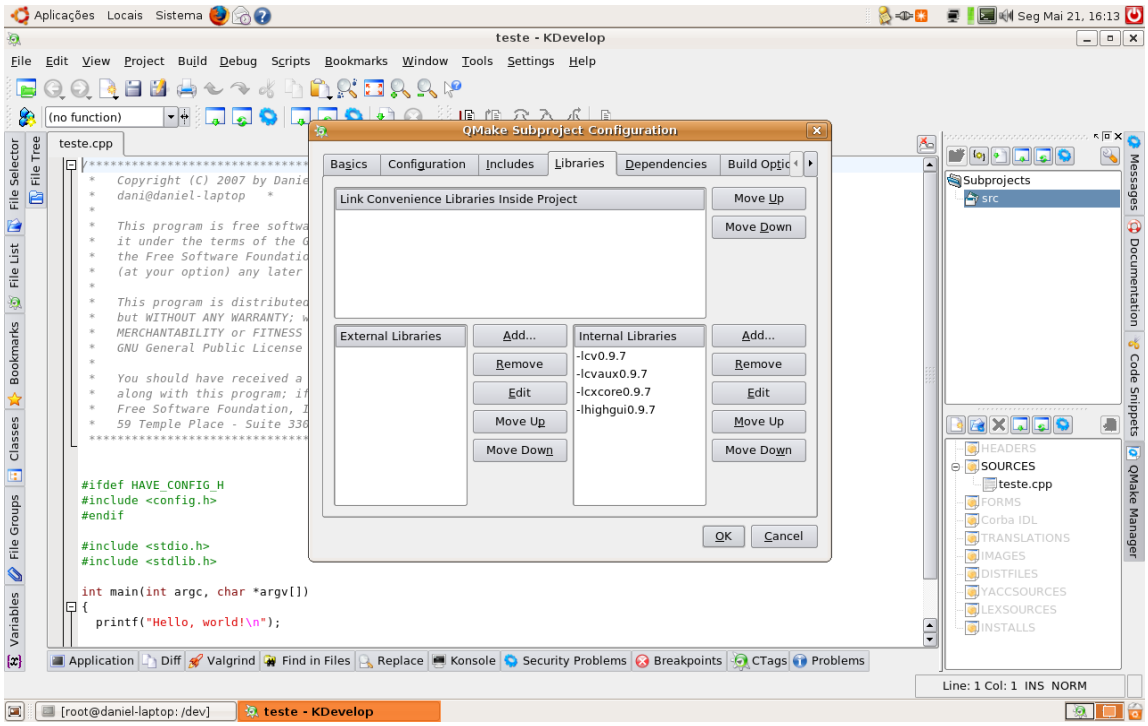
Introduza o nome da livraria na caixa de texto apresentada.



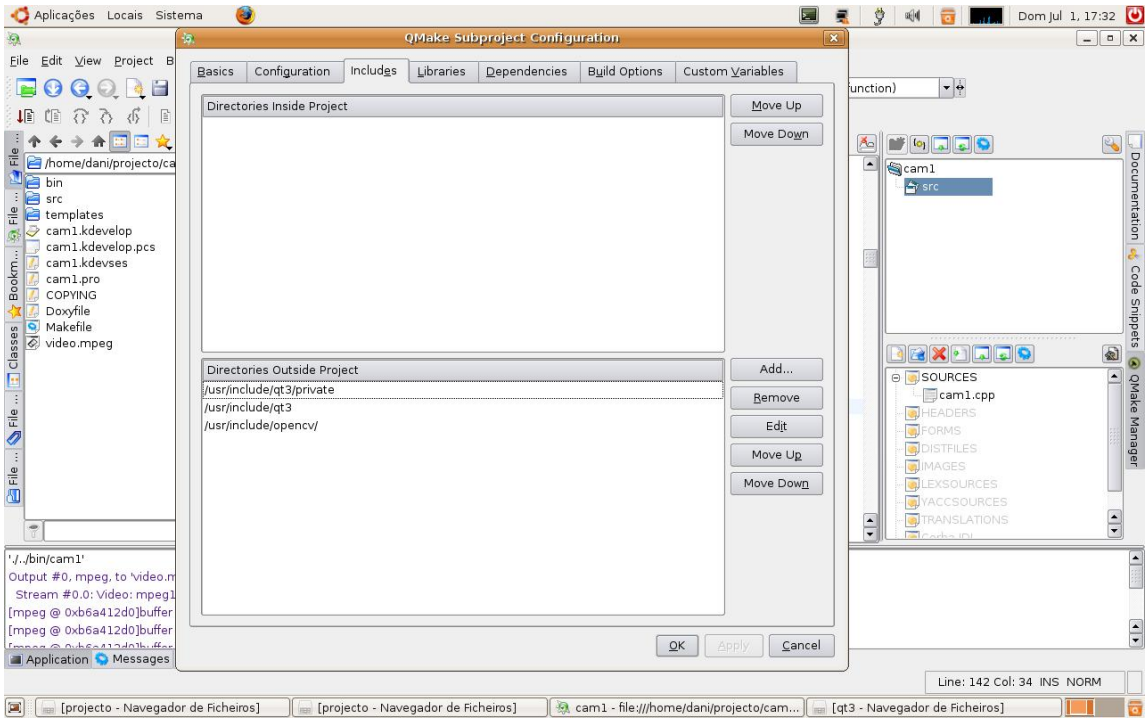
Neste exemplo, foi utilizada uma livraria do OpenCv.



No fim de todas as livrarias incluídas no projecto, clique em *"OK"*.

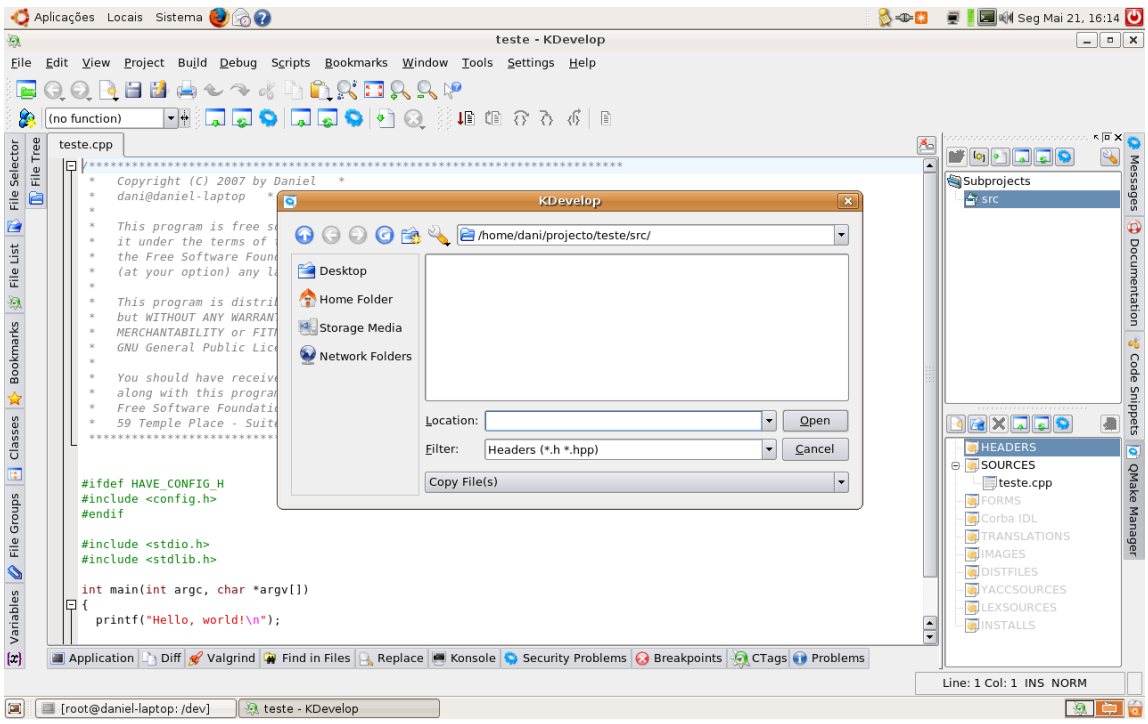


Agora é necessário incluir a localização dos ficheiros cabeçalhos no projecto. No separador *Incluudes* e no campo *Directories Outside Project*, faça *Add*...e insira os caminhos das livrarias.

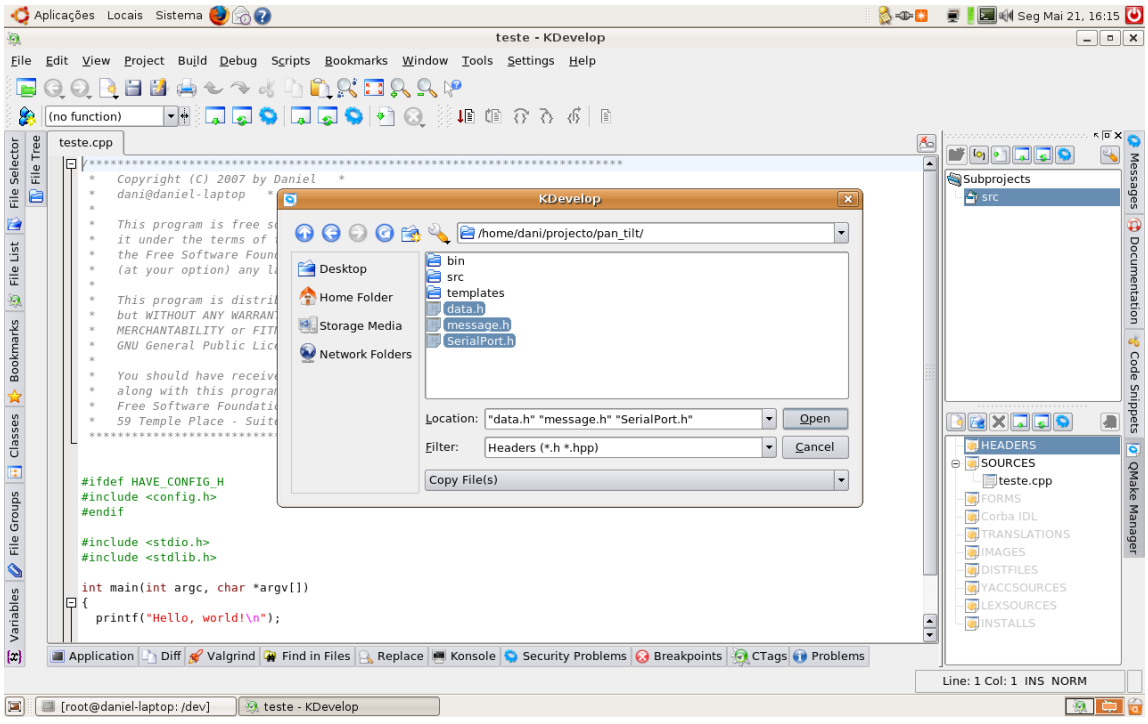


Para adicionar novos ficheiros ou ficheiros já existentes ao projecto, clique no *"QMake*

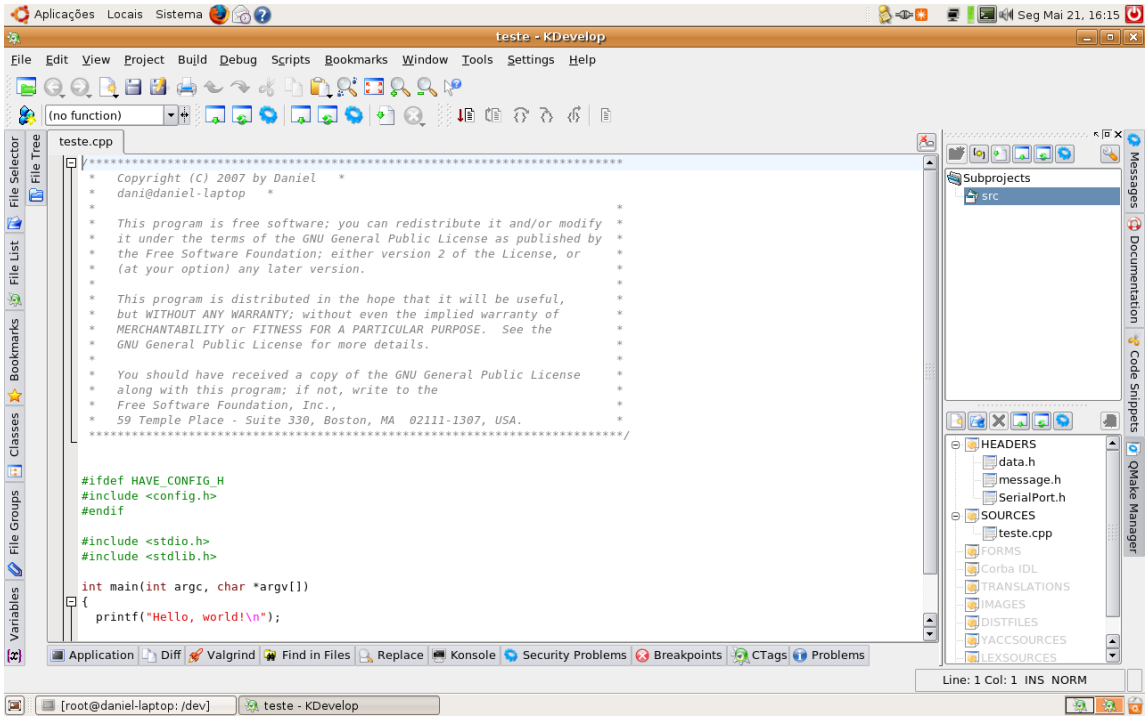
Manager" e clique em cima de *"HEADERS"* ou em *"SOURCES"* com o botão do rato do lado direito e aparecerá *"Create New File"* e *"Add Existing Files"*. Escolha um dos casos, que seja mais adequado ao seu projecto.



Este é o caso em que adicionamos ficheiros ao projecto, basta seleccionar os ficheiros e clicar em *"open"*. Os ficheiros existentes são automaticamente adicionados ao projecto.



Já com os ficheiros adicionados, recompilamos o projecto e estes ficheiros são automaticamente adicionados às *makefiles* do projecto.



Neste momento, o executável está criado e pode executar o projecto numa consola.

Anexo 6: Tutorial de Coriander

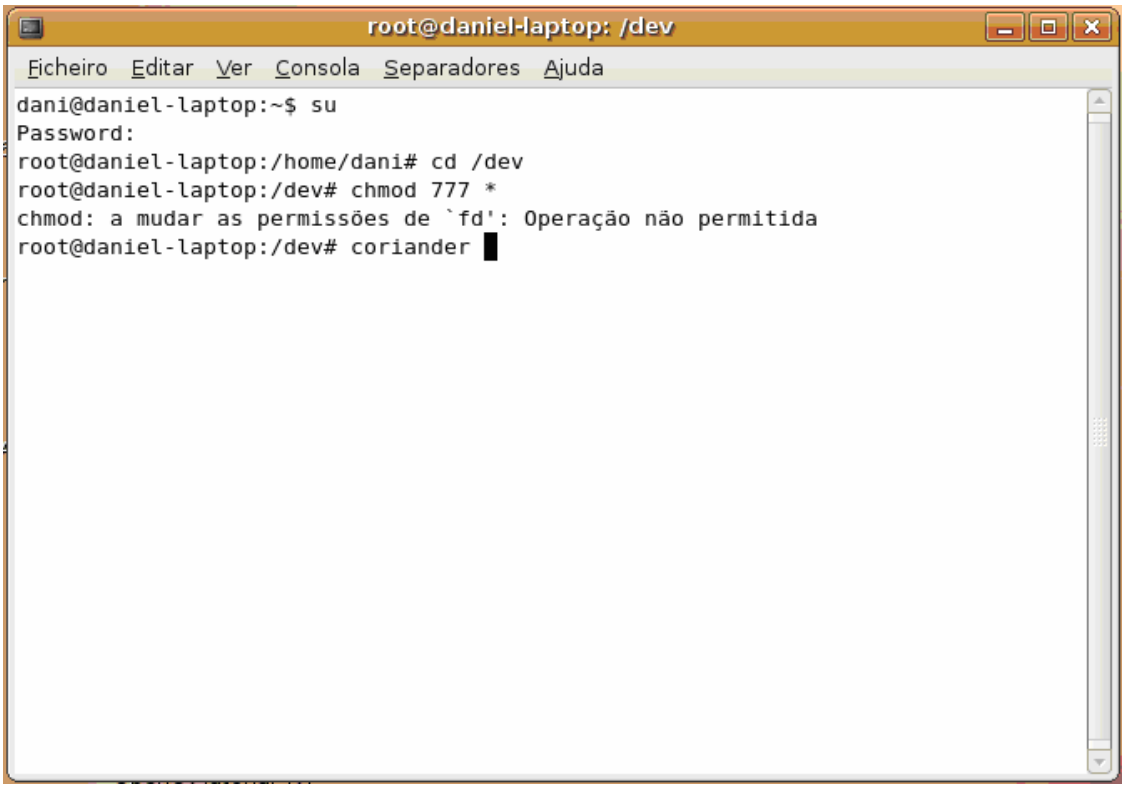
Coriander

O coriander é uma aplicação simples para visualização de imagens. Esta aplicação, foi desenvolvida para interface com cameras, principalmente *firewire,* onde este as reconhece automaticamente e ainda pode ajustar os parâmetros da camera remotamente através deste.

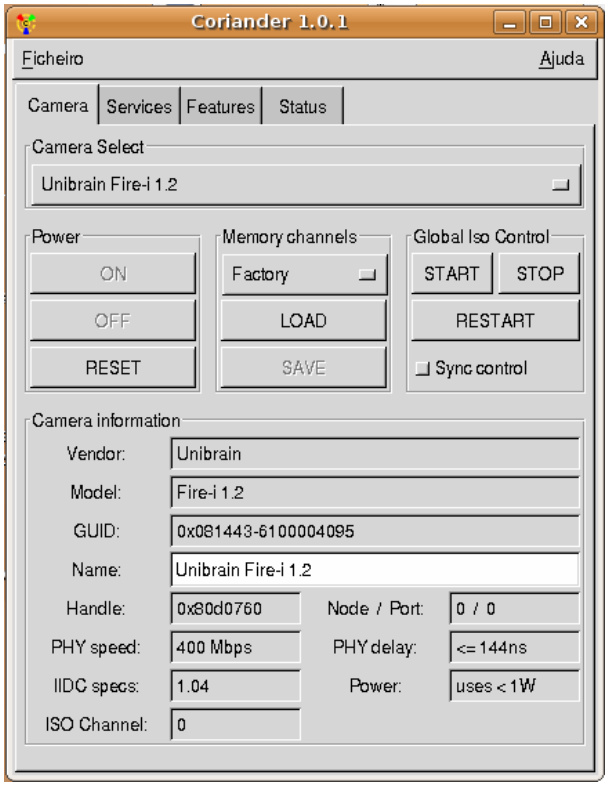
Para a utilização das cameras *FireWire* da *Unibrain* é preciso instalar no Linux.

- Libdc1394;
- Libraw;
- Libraw1394;
- Coriander;

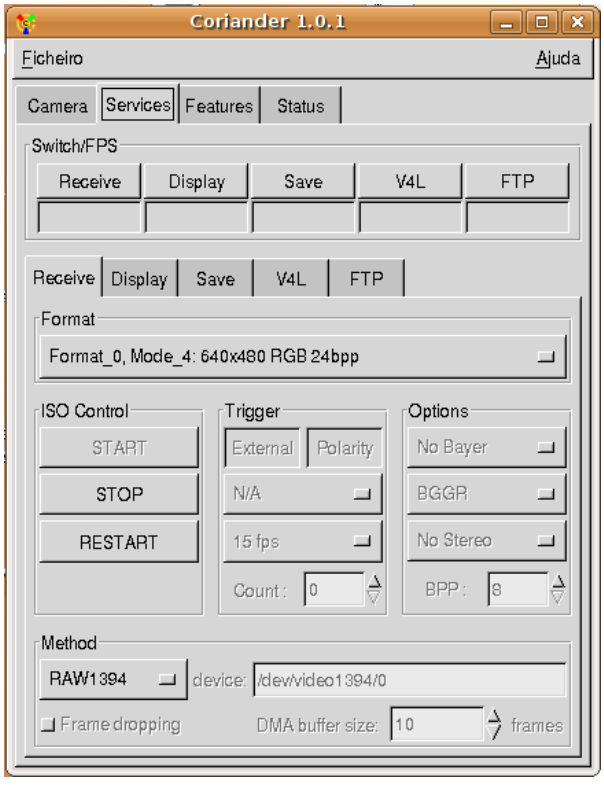
Após, a instalação desta aplicação e destas livrarias e a câmara ligada, abra uma *consola* e em modo de super utilizador, entre no directório *"cd /dev/"* e mude as permissões desta pasta para escrita, leitura e execução com o comando *"chmod 777 *"* . Por fim, execute o *Coriander* conforme na figura seguinte.



Aparecerá uma janela do *Coriander*.

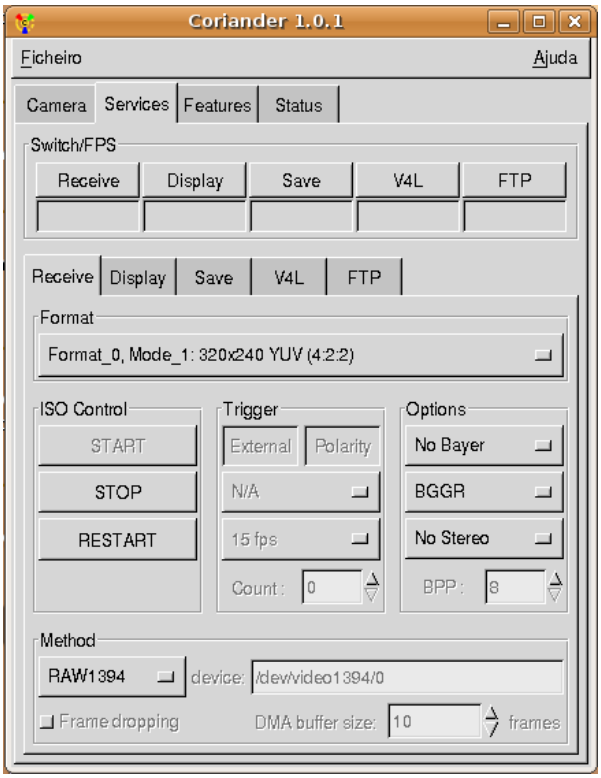


No separador principal, seleccione *"Services"*.

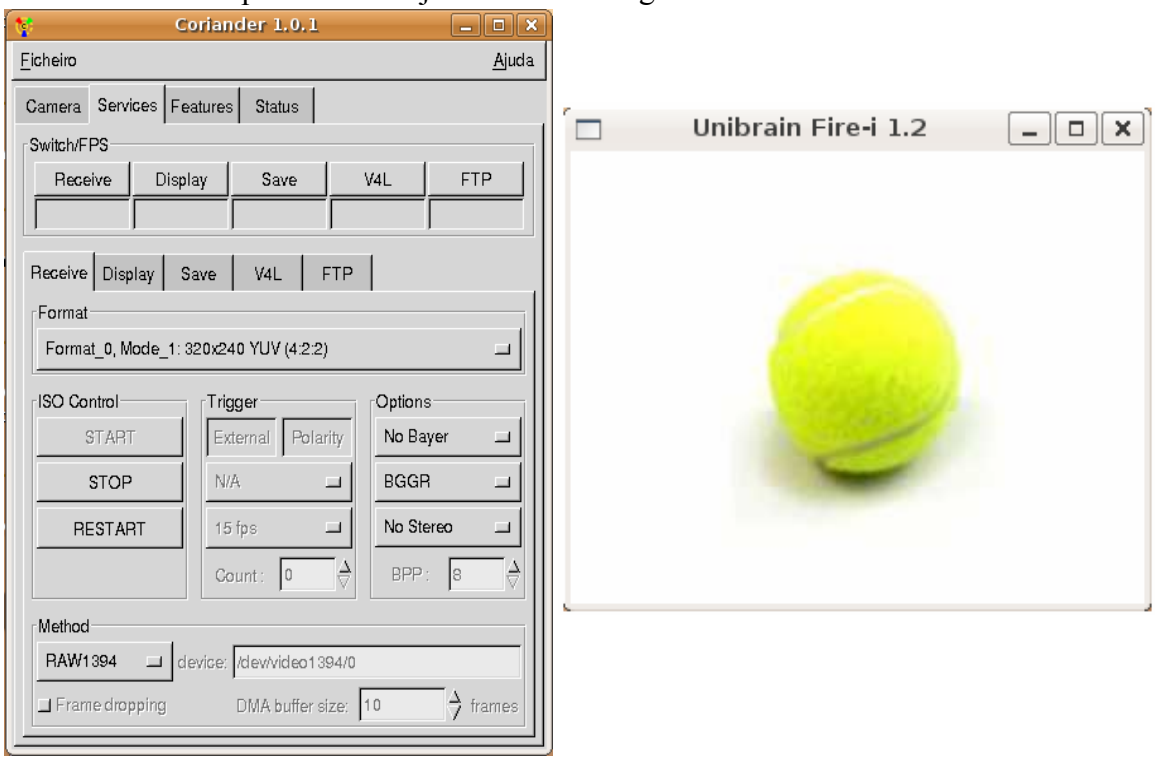


É visível no campo *Method* que existe um device em "/dev/video1394/0", pois existe uma câmara ligada *FireWire* com o índice 0.

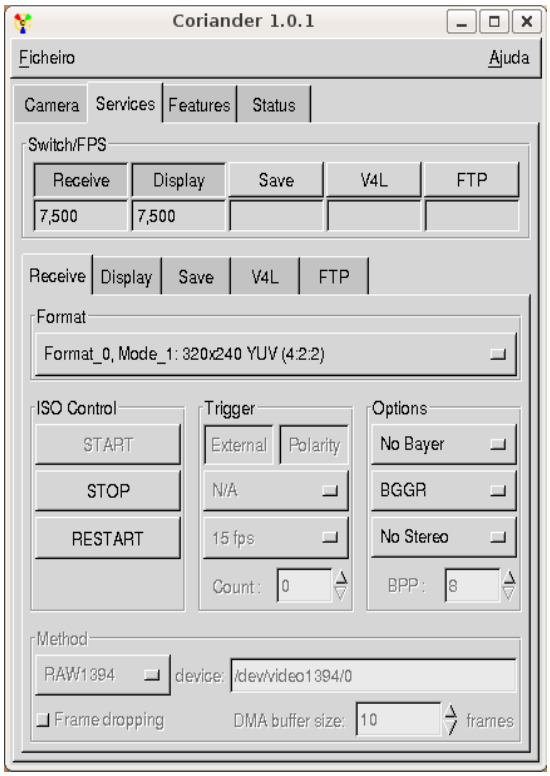
Clique em "Recive" → "Display" → "START".



Neste momento aparecerá uma janela com a imagem da camera.

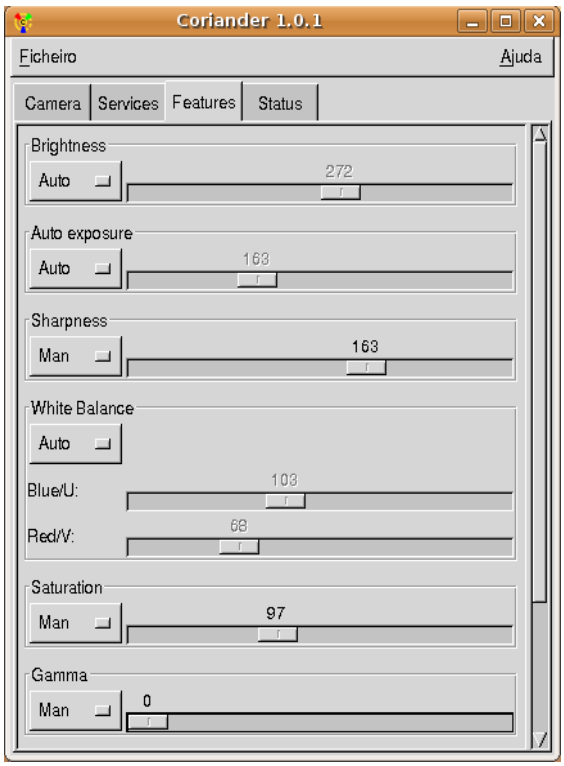


Pode visualizar os vários modos de saída da camera clicando no compo "*Format*".

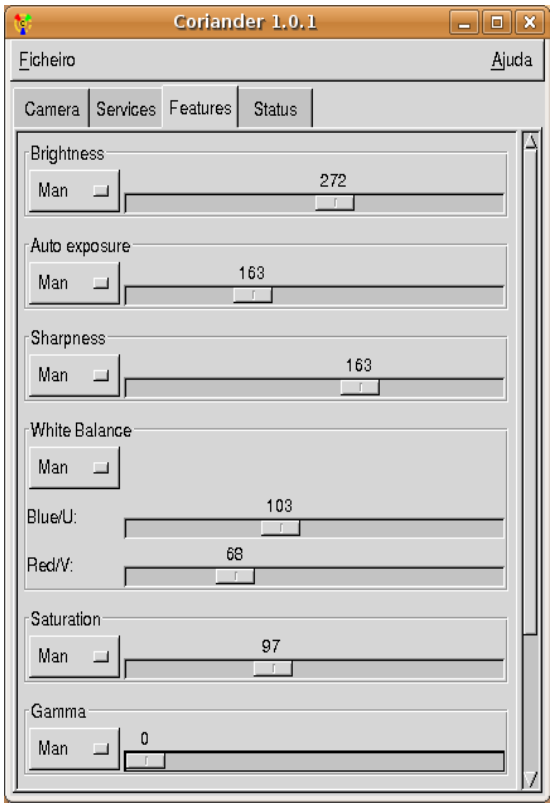


Para mudar as definições da de captura da imagem como o brilho, contraste saturação…

No separador seleccione *"Features"*, como pode ver estas então automáticas mas pode passa-las para o modo manual clicando em *"Auto"*.



Agora deslizando cada uma das barras de ajuste, pode adaptar a sua imagem de modo a ter melhores qualidade de imagem. Esta é uma ferramenta importante no ajuste da imagem, porque as condiçoes de luminosidade estão sempre a variar. E para processamento de imagem com filtros de cor esta é uma tarefa inessencial para adaptação. Sem ser necessário andar sempre a alterar os filtros de imagem.



Anexo 7: Como instalar Linux através de um dispositivo USB

Instalar o Linux a partir de um dispositivo USB

Este pequeno tutorial vem explicar, como se pode instalar o Linux a partir de um dispositivo USB. Este tutorial vem no propósito, que teve ser instalado o Linux um *embedded system* sem ter acesso a um leitor de DVD ou CD. Mas este tutorial pode ser aplicado a todos os computadores com a opção no BIOS de arranque por dispositivos USB. Pois pode-se evitar o grande transtorno, de andar sempre a gravar novos CD's sempre que sai uma nova versão do Linux, e por muitas vezes os CD não são mais utilizados, e por fim são destruídos.

Para a instalação do Linux através de um dispositivo USB é necessário um PC com o Linux já instalado para se poder criar a imagem a imagem de boot. Assume-se que o dispositivo USB está a funcionar em */dev/sda*, mas para quem tem o Windows instalado o dispositivo deve estar em */dev/sdb* mas pode ser verificado como comando *cat /proc/partitions*. Neste tutorial assume-se que vamos instalar o Debian stable (não foram testadas outras distribuições, no entanto as mesmas deverão funcionar também). Não é necessário dizer, que o dispositivo USB deve-se encontrar limpo.

- Primeiro faça o unmount da flash pen (memory stick):
  ► sudo umount /dev/sda
- Agora o download da imagem boot.img.gz que é necessária para poder arrancar com o dispositivo USB (bootable):

► wget ftp://ftp.debian.org/debian/dists/stable/main/installer-i386/current/images/hd-media/boot.img.gz

- Extraia esta imagem e grave-a no seu dispositivo USB:
  ► sudo zcat boot.img.gz > /dev/sda

Neste caso está a ser usada uma imagem da Debian Stable net-install com a versão mínima do Linux. Pode-se usar também uma versão completa ou ainda um iso business card. No entanto deve garantir que usa a mesma versão da imagem "ISO" como image.tar.gz que foi usada anteriormente.

- Agora monte o volume em /mnt:
  ► sudo mount /dev/sda /mnt
- Faça download da ISO para o dispositivo USB:
  ► cd /mnt/

► sudo wget http://cdimage.debian.org/debian-cd/4.0_r0/i386/iso-cd/debian-40r0-i386-netinst.iso

- Agora pode fazer o 'unmount' do volume:
  ► sudo umount /dev/sda

Pode remover o dispositivo USB e retirá-lo do sistema. E ensira-o no PC que pretende instalar o linux. É claro que deve obrigar a BIOS a arrancar a partir da USB.