INSTITUTO POLITÉCNICO DE BRAGANÇA
Escola Superior de Tecnologia e Gestão

Desenvolvimento de Robô Manipulador Baseado em Componentes Bioloid

Luís Filipe Reis de Sousa

Relatório Final de Estágio Profissional apresentado à
Escola Superior de Tecnologia e de Gestão
Instituto Politécnico de Bragança
para obtenção do grau de Mestre em
Engenharia Industrial

Dezembro de 2012

INSTITUTO POLITÉCNICO DE BRAGANÇA
Escola Superior de Tecnologia e Gestão

Desenvolvimento de Robô Manipulador Baseado em Componentes Bioloid

Luís Filipe Reis de Sousa

Relatório Final de Estágio Profissional apresentado à
Escola Superior de Tecnologia e de Gestão
Instituto Politécnico de Bragança
para obtenção do grau de Mestre em
Engenharia Industrial

Orientador:
Prof. Dr. José Lima
Co-orientador:
Prof. Dr. Paulo Leitão
Supervisor na Empresa (Fundación Cartif):
Prof. Dr. Juan Carlos Fraile

Dezembro de 2012

Agradecimentos

Ao longo dos últimos anos, foram muitas as pessoas, desde professores a colegas, com quem me cruzei e que em muito contribuíram para a minha formação quer como futuro profissional, quer como pessoa. Um obrigado a todas elas. Gostaria, porém, de expressar um especial agradecimento aos meus pais, uma vez que sem eles, nada disto seria possível, pelo apoio que sempre me deram ao longo de toda a minha vida académica.

Gostaria de agradecer ao Professor José Lima, o meu orientador, ao Professor Paulo Leitão, o meu co-orientador, e ao Professor Juan Carlos Fraile, o meu supervisor na empresa, pelo grande apoio dado no desenvolvimento desta tese. Por fim, desejo expressar o meu agradecimento a todos os grandes amigos que fiz, ao longo destes anos, no IPB.

Resumo

Os manipuladores industriais são amplamente utilizados em vários domínios de aplicação, tais como soldadura, montagem, pintura e manipulação. O elevado custo associado a estes sistemas de automação, torna-os praticamente impossíveis de serem adquiridos por entidades educacionais, quer sejam universidades ou escolas profissionais; no entanto, estes dispositivos são necessários para apoiar a aprendizagem de actividades na área da robótica.

O presente documento descreve as diversas etapas no desenvolvimento de uma aplicação de controlo para um robô manipulador (denominada *Educational Manipulator Studio*), baseado em *kits* didáticos, cuja principal finalidade é criar uma plataforma de baixo custo que apoie o ensino de robótica. Esta ferramenta deve apresentar um sistema de controlo flexível e permitir ser facilmente melhorada.

Na construção do robô manipulador foi selecionado o *Bioloid Comprehensive Kit*, por ser um *kit* muito versátil, acessível, relativamente robusto, e que permite ser programado em diferentes linguagens amplamente conhecidas. O robô manipulador desenvolvido neste trabalho é dotado de seis servomotores *Dynamixel AX-12* e de várias peças estruturais contidas no *kit Bioloid*, permitindo criar um manipulador com quatro graus de liberdade e uma garra.

O sistema de controlo permite operar o manipulador de forma manual (movendo a posição da garra através da atuação eixo a eixo do manipulador) e elaborar programas de movimento através da programação de trajetórias a efetuar, que posteriormente podem ser executados automaticamente. O *software* disponibilizado pela *ROBOTIS* (o fabricante do *kit Bioloid*) foi apenas parcialmente utilizado para efetuar pequenos pré-ajustes, como sejam configuração dos identificadores dos servomotores. A plataforma de controlo inclui ainda várias opções adicionais que permitem aumentar a interação do utilizador com o manipulador, como seja a facilidade do utilizador em monitorizar e controlar o robô manipulador. Em particular a inclusão do simulador 3D permite emular um tipo de programação *off-line* do robô.

A aplicação desenvolvida, incluindo o robô manipulador, tem sido testada com sucesso em ambiente de aula e apresentada em diversos eventos, nomeadamente na *Feira de Educação, Formação, Juventude e Emprego - Qualific@* que decorreu em Abril na EXPONOR (Porto) e na *Noite Europeia dos Investigadores* sobre a temática "Robótica Descodificada" que decorreu no Centro Ciência Viva de Bragança.

Palavras Chave:Robô manipulador, Plataforma educacional, *Bioloid Comprehensive Kit*, simulador 3D.

Abstract

The industrial manipulators are widely used in industry for several domain applications, such as welding, assembly, painting and manipulation. The high price associated to these automation systems, makes almost forbidden their acquisition by educational entities, such as universities or professional schools; however, they are needed to support the learning activities of robotics topics.

This document describes the various stages in the development of a control application for a manipulator robot, based on educational kits, whose main purpose is to create a low-cost platform to support the teaching of robotics topics. This tool should provide a flexible control system and allow it to be easily improved.

In the construction of the manipulator robot was selected the *Bioloid Comprehensive Kit*, for being an very versatile kit, accessible, relatively robust and allows to be programmed in diferent languages widly know. The robot manipulator developed in this work is fitted with six servomotors *Dynamixel AX-12* and various structural components contained in the kit *Bioloid*, allowing to create a manipulator with four degrees of freedom and a gripper.

The control system allows to operate the manipulator in a manually mode (by moving the position of the gripper through the performance axis by axis at the manipulator robot) and elaborate programs of moviment by programming trajectories of the goal positions, which subsequently can run automatically. The software provided by *ROBOTIS* (the manufacturer of the kit *Bioloid*) was only partly used to make small presets, such as setting the identifiers of the servomotors. The control platform also includes several additional options that allow the user to increase the interaction with the manipulator, such as the user's ease in monitoring and controlling the robot manipulator. In particular the inclusion of a 3D simulator can emulate a type of off-line programming of the robot.

The developed application, including the manipulator robot, has been successfully tested in a classroom environment and presented in several events, including the *Feira de Educação, Formação, Juventude e Emprego - Qualific@* held in April in EXPONOR (Porto) and *Noite Europeia dos Investigadores* on the theme "Robótica Descodificada"("Decrypted Robotics") held at the Centro Ciência Viva of Bragança.

Keywords: Manipulator robot, Educational plataform, *Bioloid Comprehensive Kit*, 3D simulator.

Conteúdo

Índice de figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Contextualização

Nos dias que correm os robôs manipuladores industriais são cada vez mais usados pela indústria em diversos tipos de aplicações, tais como soldadura, assemblagem, pintura e manipulação dos mais variados objectos. Este contínuo crescimento proporciona um significativo aumento na procura de pessoal devidamente qualificado para gerir o seu funcionamento. Por sua vez, o referido incremento da procura de mão-de-obra especializada leva a que as instituições de ensino se adaptem ao mercado de emprego de modo a colmatar essa necessidade, aumentando a necessidade de formação na área da robótica.

Apesar de o equipamento robótico industrial ser crucial no auxílio à aprendizagem de actividades relacionadas com a robótica, o seu elevado preço, associado a estes sistemas automatizados, torna praticamente impossível a sua aquisição por entidades educacionais, tais como universidades ou escolas profissionais.

1.2 Objectivos

Este projeto surgiu da necessidade de conceber uma plataforma que permitisse estabelecer uma conexão mais aprofundada entre as aulas teóricas e as aulas práticas de disciplinas que abordem sistemas robotizados. De modo a colmatar essa carência emergiu o plano de desenvolver um robô manipulador de baixo custo que foi proposto através da colaboração entre a empresa Fundación CARTIF, a Universidade de Valladolid e o Instituto Politécnico de Bragança. Tendo em conta os trabalhos anteriormente desenvolvidos nesta área pelas referidas instituições, o objectivo deste trabalho foi o de construir um robô manipulador com pelo menos 3 graus de liberdade e um sistema de controlo aberto, com finalidade didática, flexível e que estivesse o mais próximo possível dos sistemas de controlo industriais. Deste modo foi utilizada apenas uma estrutura de um robô manipulador que foi utilizada em outros trabalhos semenhantes.

O desenvolvimento deste projecto decorreu numa primeira fase no Instituto Politécnico de Bragança e numa segunda fase em Valladolid (Espanha) no âmbito de um estágio na empresa Fundación CARTIF com o acompanhamento de docentes da Escuela Técnica Superior de Ingenieros Industriales pertencente à Universidade de Valladolid.

O uso deste manipulador permitirá melhorar significativamente a qualidade das aulas práticas de disciplinas da área de robótica, isto porque oferece aos alunos uma maior liberdade de exploração das diversas funcionalidades e permite aprender com eventuais erros sem recearem de danificar um equipamento dispendioso e eventualmente perigoso. Além disso, o desenvolvimento de *software* para modelar e controlar o robô a partir de um computador, permite o desenvolvimento de estratégias de controlo diferentes. Deste modo, o robô manipulador projetado poderá constituir um material didáctico auxiliar aos docentes para facilitar a consolidação de conhecimentos teóricos. Do mesmo modo permitirá aos alunos adquirirem novos conhecimentos através da sua manipulação para posteriormente aplicá-los em robôs industriais comerciais.

Uma tarefa importante neste projeto é o desenvolvimento de uma aplicação de *software* para programação *off-line* do robô, sendo para isso necessário construir um simulador tridimensional do próprio manipulador. Essa melhoria no *software* permite programar e analisar os movimentos do robô no PC, antes de serem executados no mesmo.

1.3 Estrutura do documento

Este relatório encontra-se sub-dividido em sete capítulos. A seguir a esta breve introdução sucede-se o capítulo dois, onde é abordada a arquitetura do sistema em causa e também é feita uma apresentação das noções básicas relativas a manipuladores industriais. Posteriormente, no capítulo três é apresentado o material requerido para a realização deste projeto e é, também, descrita a maneira como esse material é devidamente assemblado e conetado. O capítulo quatro é composto por uma aprofundada descrição de todo o *software* utilizado no decorrer deste trabalho, tal como, uma descrição todo o trabalho de desenvolvimento da aplicação melhorada em que é feita uma detalhada explicação de todas as melhorias que foram acrescentadas, com especial destaque para as opções avançadas. No capítulo cinco encontram-se em pormenor os passos a ter em conta para a construção do simulador tridimensional. No capítulo seis são apresentadas algumas demonstrações do uso da aplicação e é apresentada uma discussão dos resultados obtidos nos testes da aplicação.

Por fim, no capítulo sete são apresentadas as conclusões adquiridas ao longo do desenvolvimento e conclusão deste projeto. É ainda feita uma alusão a propostas para futuros trabalhos, de modo a evoluir ainda mais a aplicação desenvolvida.

Capítulo 2

Manipulador didático baseado em *Bioloid*

O presente capítulo aborda as noções principais associadas a manipuladores industriais, incluindo uma breve descrição do modo de caracterizar estes mesmos, e introduz a arquitetura do manipulador desenvolvido neste trabalho.

2.1 Robôs manipuladores industriais

Os robôs manipuladores industriais são amplamente utilizados pela indústria dos dias de hoje, mas o que é um robô manipulador? Segundo a Associação de Indústrias da Robótica (RIA) a sua definição para robô industrial é: "*Um robô industrial, ver Figura 2.1, é um manipulador reprogramável, multifuncional, projetado para mover materiais, peças, ferramentas ou dispositivos especiais em movimentos variáveis programados para a realização de uma variedade de tarefas*". Desta definição podemos concluir que um robô manipulador tem como principal objetivo deslocar materiais, podendo ser peças diversas, ferramentas que irão trabalhar sobre uma peça, sistemas de visão, entre outras possibilidades. [5, 11]

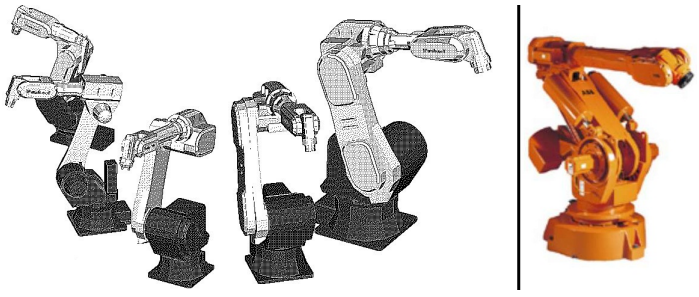


Figura 2.1: Exemplos de robôs manipuladores industriais (caso comum de braço manipulador)

Um robô manipulador é portanto uma máquina multi-aplicação e re-programável possuindo certas características antropomórficas muito à semelhança de um

membro superior humano.

As vantagens implícitas à utilização deste tipo de equipamento no desempenho de tarefas industriais são várias, entre as quais se destacam as seguintes:

- Factores de segurança acrescida na interface de operadores com máquinas industriais;
- Repetibilidade exaustiva da tarefa a realizar sem pausas;
- O ambiente de trabalho por vezes é prejudicial ao ser humano;
- Possibilita a manipulação de cargas elevadas que seriam difíceis ou impossíveis para operadores;
- Níveis de precisão altamente superiores aos do ser humano;
- Grande versatilidade no desempenho de tarefas;
- Utilização eficiente das unidades de produção intensiva.

Os robôs industriais podem ser classificados de acordo com o número de juntas, o tipo de controlo, o tipo de acionamento e a sua geometria. Normalmente classificam-se os robôs de acordo com o tipo de junta, ou, mais exatamente, pelas três juntas mais próximas da base do robô. Também podem ser classificados em relação ao espaço de trabalho (*workspace*), ao grau de rigidez, à extensão de controlo sobre o percurso do movimento, e de acordo com respetivsas aplicações. [1]

Os diferentes graus de liberdade de um robô podem ser encontrados em várias combinações de configurações rotacionais e lineares, dependendo da aplicação. Tais combinações são denominadas geometria do robô. As juntas tipicas podem ser descritas do seguinte modo (ver Figura 2.2):

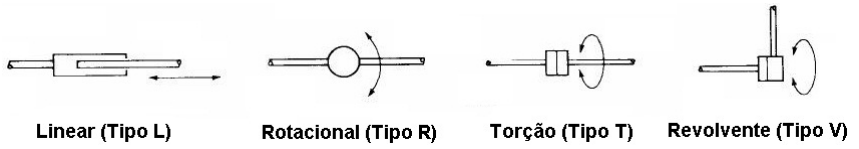


Figura 2.2: Tipos de juntas ou articulações

• Junta Linear (tipo L) - Permite o movimento das ligações de uma forma linear.

• Junta Rotacional (tipo R) - Movimento de rotação em relação ao eixo perpendicular à junção das duas ligações.

• Junta de Torção (tipo T) - Movimento de rotação em relação ao eixo paralelo à junção das duas ligações.

• Junta Revolvente (tipo V) - Movimento de rotação com eixo paralelo à ligação de entrada (a mais próxima da base) e perpendicular à de saída. [2]

Cada configuração pode ser representada pela sequência das diferentes juntas através da notação de letras de cada junta (L, R, T ou V). Começando pelo grau de liberdade mais próximo da base, podendo ser classicado como: Cartesiano (LLL), Cilíndrico (TLL), Esférico (TRL), Articulado (TRR) e SCARA

(VRL). Este tipo de caracterização denomina também o seu sistema geométrico coordenadas, isto porque descreve igualmente o tipo de movimento que o robô executa. Na Figura 2.3 encontram-se representadas as estruturas dos vários tipos de robôs e sua designação.

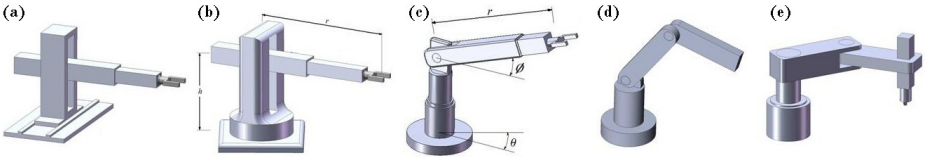


Figura 2.3: Representações de estruturas de robôs do tipo: *(a)* Cartesiano; *(b)* Cilindríco; *(c)* Esférico; *(d)* Articulado; *(e)* SCARA - *Selective Compliance Assembly Robot Arm.*

Associada à compra de um robô manipulador industrial é necessário comprar a respectiva unidade controladora. Este equipamento fornece a alimentação necessária para o robô manipulador e controla a sua atividade. Esta unidade é normalmente bastante volumosa, o que dificulta a gestão de espaço relacionada com a sua instalação. Por vezes é necessário adaptar novas instalações, o que proporciona a que os preços dos robôs manipuladores industriais atinjam facilmente montantes que não podem ser suportados pelas instituições de ensino.

2.2 Arquitetura da aplicação robotizada didática

O robô manipulador desenvolvido neste trabalho possui quatro eixos de liberdade e uma ferramenta do tipo garra, e assenta numa arquitetura que compreende diversos blocos, tal como é ilustrado na Figura 2.4.

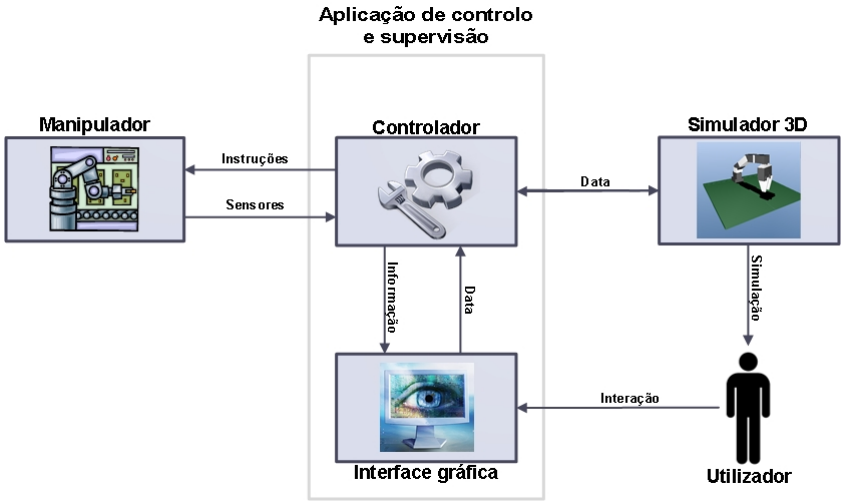


Figura 2.4: Esquemática da arquitetura do sistema

O bloco denominado de 'Manipulador' envolve toda a componente de hardware, composta pelo próprio robô manipulador construído principalmente com

componentes do *Bioloid Comprehensive Kit*, passando pelos componentes de comunicação com o seu controlador. O manipulador fornece constantemente informação relativa aos seus sensores, transmitindo dados sobre posição, velocidade e temperatura. Esta informação é processada pelo elemento *USB2Dynamixel* que por sua vez, comunica com o controlador de modo a estabelecer a troca de informação entre o robô e a aplicação.

O grupo 'Aplicação de controlo e supervisão' engloba os blocos 'Controlador' e 'Interface gráfica', em que neste segundo bloco representa os menus da aplicação desenvolvida para o utilizador. É nesta 'Interface gráfica' que o utilizador irá interagir de modo a controlar o robô manipulador. Este controlo pode ser feito acionando directamente os servomotores *Dynamixel* pela simples alteração na posição das barras deslizantes que controlam o ângulo de cada atuador. Outro modo de interagir com o manipulador é através da elaboração de um programa por aprendizagem. Ou seja criar um ficheiro que guarde os valores das posições de todos os *Dynamixels* e simultaneamente que lhe atribua uma velocidade de deslocação para esse mesmo ponto. Deste modo é permitido ao utilizador executar de um modo automático trajectórias previamente guardadas. O modo de funcionamento desta interface encontra-se explicado no capítulo 4.2.

Evidentemente, como não poderia deixar de ser, esta 'Interface gráfica' possui associada um sistema que a controla, em que na arquitetura se encontra denominado de 'Controlador'. Este sistema controlador é o núcleo de todas as operações envolvidas neste projeto, isto porque é ele que processa toda a informação proveniente do 'Manipulador' e da 'Interface gráfica' manipulando-a de forma a ser transmitida para o seu destino permitindo assim a interligação de todos os blocos e mantendo o bom funcionamento da aplicação. Este 'Controlador' é construído por linhas de código em linguagem Pascal, que funciona com base recursiva à evocação de procedimentos. Encadeando logicamente o conteúdo destes procedimentos tornou possível criar a aplicação de controlo e supervisão.

Por fim, este esquema inclui o bloco 'Simulador 3D', que tal como o próprio nome o indica, a aplicação fornece ao seu utilizador a hipótese de simular virtualmente o robô manipulador. Esta simulação é particularmente útil para o caso de o utilizador pretender programar previamente o robô manipulador na ausência do mesmo. Deste modo, o utilizador pode testar previamente os programas por ele criados sem correr o risco de danificar qualquer equipamento. Para a construção deste simulador foi necessário acrescentar uma componente gráfica ao programa de desenvolvimento da aplicação (Lazarus). Essa componente denomina-se *GLScene*, que permite desenvolver ambientes tridimensionais de forma relativamente rápida. A explicação pormenorizada deste componente encontra-se descrita no capítulo 5.

Capítulo 3

Construção física do manipulador

Este capítulo descreve o modo de construção do manipulador em termos de *hardware*. Para o efeito é descrita a sequência de construção, os materiais utilizados e a sua configuração/conexão. É apresentada uma breve descrição desses mesmos materiais e é ainda descrito o modo de como todas essas peças devem ser interligadas de modo a que o produto final (manipulador *Bioloid*), coincida com a montagem já testada. É de realçar a importância de conetar devidamente todos os elementos físicos requeridos neste projeto de acordo com as instruções referidas neste documento. Caso tal não seja verificado é altamente provável que o *software* (*Educational Manipulator Studio V1.0*), que foi desenvolvido especificamente para esta aplicação não funcione conforme o aceitável. Ainda assim existem alguns componentes que não vêm incluídos no '*Bioloid Comprehensive Kit*', nomeadamente a estrutura plana que servirá de base ao manipulador, o que permite alguma flexibilidade no material que pode vir a ser usado. Deste modo serão apresentadas algumas soluções práticas para a dissolução deste problema.

Na realização deste projeto iniciou-se por montar a estrutura do robô manipulador. De forma esquematizada pode-se ver na Figura 3.1 o modelo do robô que foi construído neste projeto. Os robôs manipuladores industriais tipicamente são do tipo TRR TRR, mas neste projeto pode-se caracterizar o sistema de juntas do manipulador em causa, como sendo do tipo TRRT.

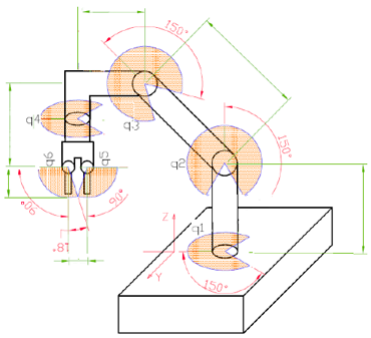


Figura 3.1: Modelo esquemático do robô manipulador

Para a elaboração deste projeto foram utilizadas várias peças que fazem parte do *Bioloid Comprehensive Kit*, sendo as peças em causa exibidas na Figura 3.2. De modo a facilitar a alusão dos materiais em causa, nesta mesma figura foram mantidas as referências fornecidas pelo fabricante (*ROBOTIS*). As quantidades necessárias de cada peça encontram-se também mencionadas nesta mesma figura. Assim sendo é mantido o modo de apresentação do material necessário tal como o fabricante indica permanentemente para a montagem dos seus *kits*. [8]

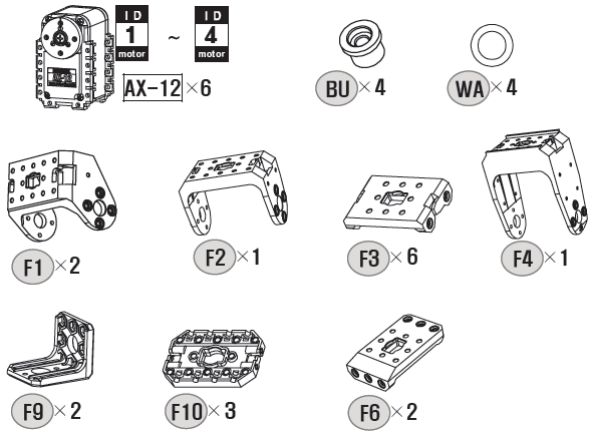


Figura 3.2: Peças necessárias para a construção do robô manipulador (com as referências do fabricante)

É de notar que na Figura 3.2 apenas constam as peças necessárias para montar o robô manipulador, nomeadamente os servomotores *Dynamixel AX-12* e as peças de estrutura. Contudo, é necessário adicionar à lista de materiais os elementos que as suportam e interligam, ou seja, os parafusos e fêmeas. A quantidade e dimensões destes pequenos elementos pode variar, se bem que é altamente aconselhável que seja usado o maior número possível de parafusos e respectivas fêmeas, isto porque, é uma mais valia possuir uma estrutura o mais sólida possível.

Enquanto que as estruturas de plástico não possuem grande descrição possível, já aos *Dynamixels* deve ser dada especial atenção quanto às suas características técnicas, de modo a melhor se compreender o funcionamento e limitações do robô manipulador.

Após uma breve observação da Figura 3.3 podemos constatar que a aparência destes servomotores é bastante simples e indutiva ao seu fácil manuseio. Possuem, então, um LED (vermelho) indicador de estado que pode ser facilmente programado para ser acionado conforme o pretendido. Possui duas entradas para estabelecer a sua alimentação e respectiva comunicação com outros elementos controladores ou actuadores. Nestes servomotores é ainda indicado um número identificativo para cada *Dynamixel*. Este número deve ser único para cada servomotor por estrutura, ou seja, num robô manipulador não podem existir dois *Dynamixels* com o mesmo ID. Caso isto aconteça os servomotores com numeração repetida irão replicar da mesma maneira os comandos fornecidos pelo respectivo controlador, que é uma situação não pretendida para um projeto desta natureza. [10]

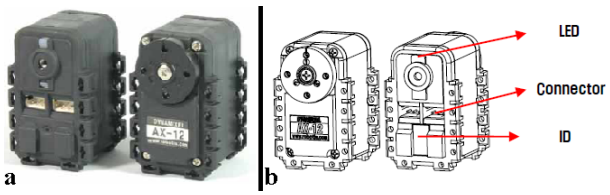


Figura 3.3: a)Servomotor *Dynamixel AX-12*; b)Esquemático do *AX-12* com legenda

Na Tabela 3.1 estão listadas as principais características técnicas dos servomotores *Dynamixel AX-12*. É aconselhável efetuar uma cuidada análise a estes dados antes de modo a compreender algumas das opções tomadas neste projeto. Desta mesma tabela tem especial interesse alguns destes parâmetros, que são: resolução, ângulo de funcionamento, voltagem, temperatura de funcionamento, tipo de sinal de controlo e a numeração de identificação. Embora o seu peso e binário não tenham sido mantidos muito em grande destaque, é de notar que estes parâmetros são deveras importantes para o cálculo de cargas que o robô pode manusear.

Tabela 3.1: Principais características do servomotor *Dynamixel AX-12* (fornecida pelo fabricante *ROBOTIS*)

Weight (g)	55
Gear Reduction Ratio	1/254
Final Max Holding Torque (Kgf.cm)	12 16.5
Sec/60degree	0.269 0.196
Resolution	0.35º
Operating Angle	300º, Endless Turn
Voltage	7V~12V (Recomended voltage:9.6V)
Max. Current	900mA
Operate Temperature	-5ºC ~ +85ºC
Command Signal	Digital Packet
Protocol Type	Half duplex Asynchronous Serial Communication (8bit,1stop,No Parity)
Link (Physical)	TTL Level Multi Drop (daisy chain type Connector)
ID	254 ID (0~253)
Communication Speed	7343bps ~ 1 Mbps
Feedback	Position, Temperature, Load, Input Voltage, etc.
Material	Engineering Plastic

Um dado muito importante retirado da Tabela 3.1 do qual se deve ter continuamente consciência é o ângulo de funcionamento destes servomotores. E como não basta apenas saber que este mesmo é de 300º, é também necessário tomar conhecimento quais são as posições e referências desse mesmo ângulo. Para tal deve-se observar cuidadosamente a Figura 3.4. Daqui podemos verificar onde se localiza a referência do ângulo (posição zero graus) e qual é a área que o

Dynamixel não atinge (assinalada a vermelho).

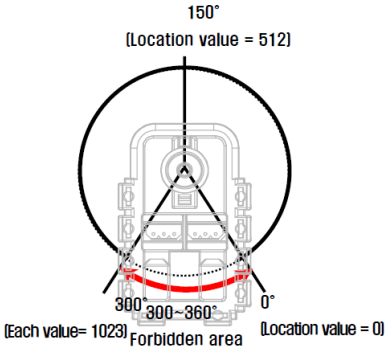


Figura 3.4: Ângulos de funcionamento do servomotor *AX-12*

O primeiro factor a ter em conta antes de se iniciar a montagem do robô manipulador é a posição em que se encontram os eixos dos servomotores *Dynamixel AX-12* em relação ao resto do corpo do servomotor. De modo a que a posição dos *Dynamixels* coincidam com a posição do modelo 3D contido no *Educational Manipulator Studio* é necessário alinhar a referência que se encontra assinalada no eixo dos *Dynamixels* com a referência existente no corpo do servomotor, tal com se encontra exemplificado na Figura 3.5.

Figura 3.5: Alinhamento entre a parte móvel e a estrutura do servomotor (*AX-12*)

Este alinhamento certifica que a posição dos eixos dos *Dynamixels* se encontra sempre a 150º, ou seja a meio do seu ângulo de funcionamento. Se todos os *Dynamixels* se encontrarem na posição de 150º, o robô manipulador deve exibir um aspecto semelhante ao da Figura 3.6.

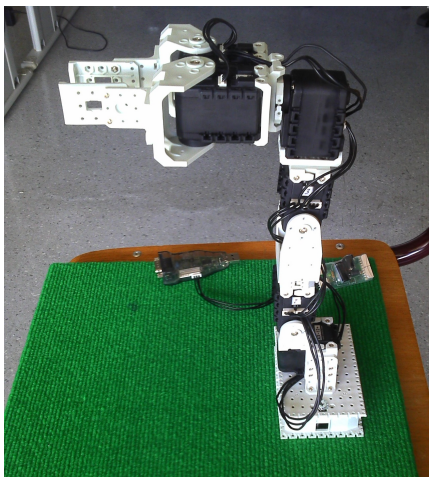


Figura 3.6: Braço manipulador com os servos alinhados

De forma a completar as peças que formam o manipulador é necessária a utilização de uma base e de um sistema de fixação para a mesma. Neste projeto utilizou-se para a base uma peça plana em madeira com dimensões aproximadas de 35cm x35cm x1cm. A utilização deste tipo de material é totalmente opcional, contudo, é necessário ter em atenção que esta peça necessita de possuir um peso e área suficiente de modo a que consiga manter o mais estável possível o robô manipulador e que lhe conceda uma área adequada de trabalho de acordo com as funções a desempenhar pelo mesmo. É ainda aconselhado que esta mesma peça que servirá de base seja revestida por um material que impeça os objectos a serem manipulados de se moverem livremente pela mesma.

Também as peças que fixam o robô manipulador à base ficam um pouco ao critério do utilizador, conforme se poderá verificar é possível utilizar peças do *Bioloid Comprehensive Kit*, nomeadamente as peças que se encontram na Figura 3.7. Caso a peça de fixação seja fabricada pelo utilizador, esta pode possuir as dimensões que lhe forem mais convenientes, uma vez que é possível ajustar nas configurações do programa de modo a que as dimensões da peça fabricada coincidam com o modelo 3D contido no *Educational Manipulator Studio*.

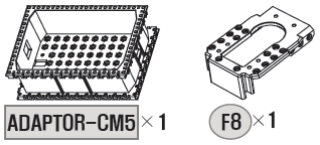


Figura 3.7: Peças opcionais para fixar o robô na base

De modo a iniciar a propriamente dita assemblagem do robô manipulador inicia-se então pela estrutura que o fixa à base. Na Figura 3.8 pode-se ver duas sugestões de fixação. Em ambos os casos é de ter em conta que as peças utilizadas têm uma altura considerável (~4cm), o facto de se usar este acentuado desnível em vez de se fixar directamente à base, tem a ver com o necessidade de que deste modo o robô manipulador possua uma área de trabalho mais prática de acordo

com as dimensões do mesmo. Embora, se assim for pretendido essa altura entre a base e o manipulador fique à descrição do utilizador e portanto pode até ser suprimida.

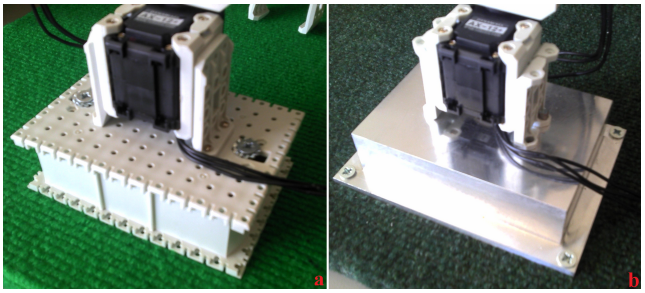


Figura 3.8: a)Adaptador *CM5* utilizado como suporte entre a base e robô; b)Mesmo conceito que em 'a', mas com peça fabricada em alumínio maciço

Seguidamente encontra-se exemplificado o modo de montagem de um *Dynamixel* (servomotor 2), com a respectiva peça de estrutura *F2*, ver Figura 3.9. Salienta-se novamente a importância de manter o eixo do servomotor sempre alinhado e a utilização do máximo de parafusos possível de modo a manter a montagem o mais sólida possível.

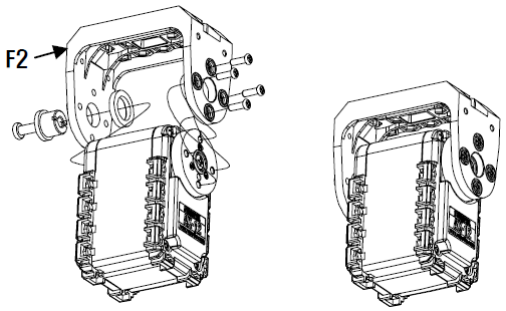


Figura 3.9: Exemplo de montagem de um servomotor (*AX-12*) com a respectiva peça de estrutura (F12)

Nas Figuras 3.10, 3.11 e 3.13 encontra-se pormenorizado o modo de como todos os elementos da montagem se encontram dispostos para formar o robô manipulador.

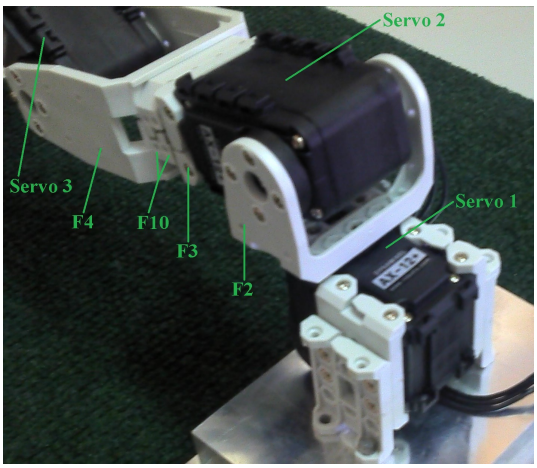


Figura 3.10: Montagem da zona junto à base

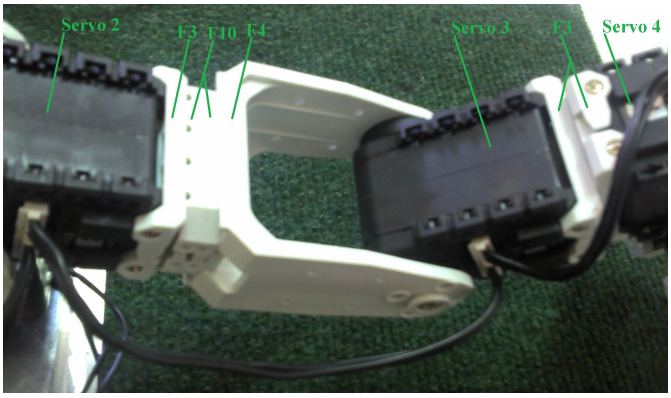


Figura 3.11: Montagem da zona média do manipulador

Para uma melhor aderência da garra/ferramenta com os objetos a serem manipulados foi colado nas extremidades interiores da garra um tipo de esponja suave que se adapta facilmente aos contornos dos objetos manipulados, ver Figura 3.12.

Figura 3.12: Garra com esponjas nas extremidades

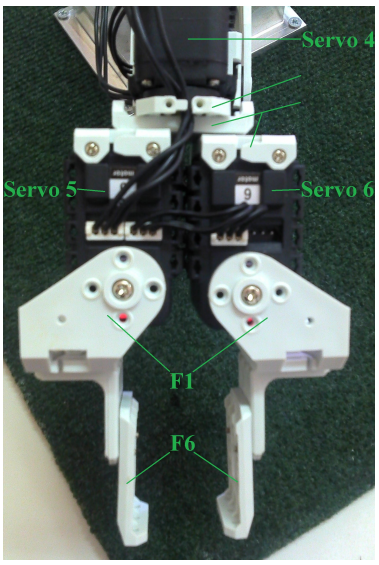


Figura 3.13: Montagem da zona da garra

Resta referir os materiais utilizados na alimentação e controlo dos *Dynamixels*. Assim sendo, e como não podia deixar de ser, são requeridos cabos que estabelecem a comunicação e fornecem a alimentação aos servomotores. Mais uma vez a *ROBOTIS* fornece o equipamento necessário, ver Figura 3.14. Opcionalmente estes cabos podem também ser fabricados pelo próprio utilizador.

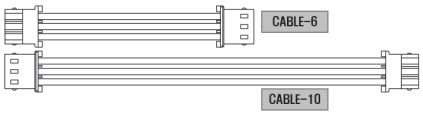


Figura 3.14: Cabos de alimentação e comunicação entre os servomotores e seu controlador

De modo a estabelecer a comunicação entre o PC e os *Dynamixels* foi utilizada o importantíssimo adaptador *USB2Dynamixel*. Este adaptador permite eliminar o uso da consola de programação *CM-5* e estabelecer uma conexão directa entre os *Dynamixels* e o computador por meio USB. Noutros casos esta ferramenta também permite transformar uma porta USB numa porta série. Este pequeno aparelho permite controlar diretamente uma vasta família de servomotores *Dynamixel*, sendo essa a razão de este se encontrar equipado com duas portas, *RS485* e *RS422* (de 3 pinos e outra de 4 pinos), tal como se pode constatar na Figura 3.15. Para este projeto foram utilizados unicamente servomotores *Dynamixel AX-12* o que apenas requereu a utilização da porta *RS485*. Como já foi constatado anteriormente a quando a descrição dos servomotores em causa estes comunicam através de um sinal de comando em formato *TTL*, assim sendo, é necessário comutar o interruptor de selecção de funções para a posição *TTL*. [9]

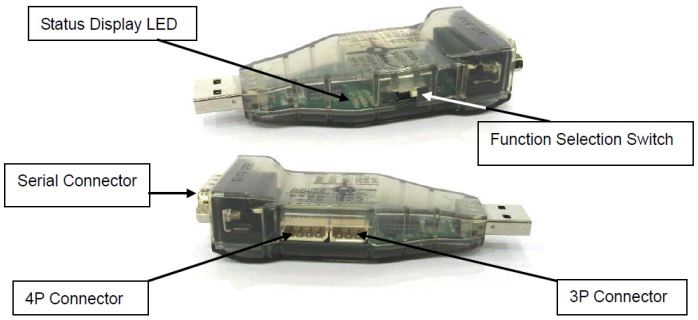


Figura 3.15: Adaptador *USB2Dynamixel*

De modo a alimentar os *Dynamixels* é necessário utilizar uma fonte de tensão adequada às características dos mesmos, devemos então utilizar uma fonte de alimentação igual ou semelhante ao da Figura 3.16.

Figura 3.16: *Power tranformer SMSP* - fonte de alimentação para os *Dynamixels* (12V)

Por fim é necessário um adaptador *SMSP2Dynamixel*, como o exibido na Figura 3.17. Esta pequena peça de equipamento permite fornecer a energia vinda do transformador de tensão para os servomotores *Dynamixels*. Este adaptador estabelece a conexão entre o *USB2Dynamixel* e os próprios *Dynamixels* sem alterar em nada as instruções de comando que por ele passam.

Figura 3.17: Adaptador *SMSP2Dynamixel*

Uma vez que a estrutura do robô manipulador se encontre completa é necessário estabelecer as comunicações entre os vários elementos. Conforme se pode verificar na Figura 3.18, o método de estabelecer as conexões eléctricas é bastante simples. Basta portanto conetar o *USB2Dynamixel* a uma porta USB de um computador que contenha o programa controlador, e através de um cabo série interligar a porta *RS485* (três pinos) do *USB2Dynamixel* a uma porta igualmente *RS485* do adaptador *SMSP2Dynamixel*, sendo que a outra porta *RS485*

do referido adaptador vai-se conetar (por questões práticas) ao *Dynamixel* que se encontra mais junto da base. Quanto aos outros servomotores as conexões deverão ser estabelecidas em série, preferencialmente na ordem da base em direção à ferramenta. Para a alimentação basta portanto conetar o transformador à rede e ligar o seu conector ao adaptador *SMSP2Dynamixel*.

Uma grande vantagem na utilização destes componentes é de que tanto nas conexões nos servomotores *Dynamixel* como no adaptador *SMSP2Dynamixel* as portas utilizadas não possuem ordem fixa de entrada e saída. Este facto dá ao utilizador uma grande flexibilidade para estabelecer as conexões conforme o que for mais conveniente.

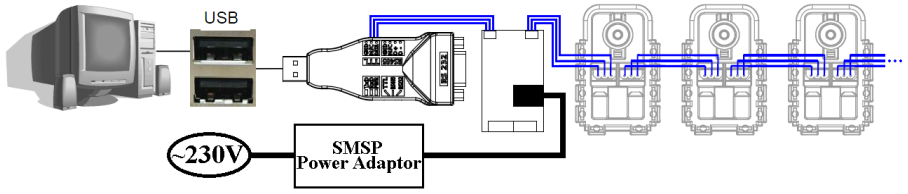


Figura 3.18: Esquema das ligações entre todos os componentes necessários

Neste projeto o resultado final da assemblagem do robô manipulador com peças do *Bioloid Comprehensive Kit* é apresentado na Figura 3.19.

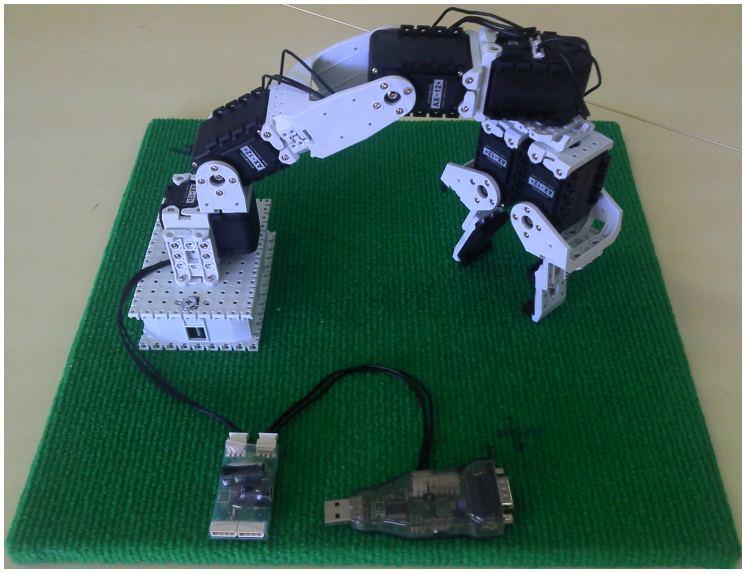


Figura 3.19: Montagem final

Uma vez completa a explicação de como montar o robô manipulador é necessário compreender como se pode interagir com este manipulador, para isso segue-se o capítulo 4, onde é aprofundada a explicação de como a palicação foi criada e quais são as suas funcionalidades.

Capítulo 4

Aplicação de controlo e supervisão

Neste capítulo é descrita a aplicação de controlo e supervisão do manipulador didático. Em particular são descritas de forma pormenorizada a configuração do sistema, o modo de operação e os modos de programação (manual e *RAPID*). Uma breve descrição do *software ROBOTIS* é também elaborada para suportar a configuração dos servomotores. A utilização do *Dynamixel Manager* é opcional pois este apenas foi utilizado para efetuar um pré-ajuste aos servomotores (*Dynamixels*). Embora, não seja estritamente necessário consumar estes ajustes, é aconselhado que sejam feitos meramente por questões de organização prática.

4.1 *Software* Robotis - DynamixelManager

Este *software*, desenvolvido e fornecido pela *ROBOTIS*, é uma ferramenta bastante útil para efetuar alguma pré-configurações. Uma vez que se execute o respectivo programa, é exibida uma janela denominada de *Dynamixel Search*. Para os casos mais simples com poucos componentes *Dynamixel* basta fazer uma busca rápida dos mesmos, para isso selecciona-se *Quick Search* e clica-se no botão Search. Em poucos segundos todos os *Dynamixels* conetados ao *USB2Dynamixel* serão encontrados e exibidos nesta mesma janela, sendo apenas identificados pelo seu número de identificação (ID), tal como se pode ver na Figura 4.1. Para o caso de que os *Dynamixels* já tenham sido encontrados pelo programa, não é necessário esperar até ao final de toda a procura, bastando para isso clicar em *Stop* e seguidamente em *OK*.

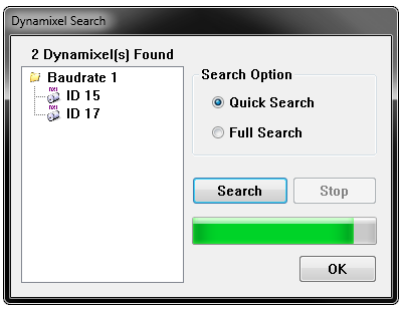


Figura 4.1: Procura de servomotores *Dynamixel*

Uma vez obtidos os *Dynamixels* conectados (neste caso exemplificativo, ID 15 e ID 17), tem-se acesso à janela do próprio programa *Dynamixel Manager*, ver Figura 4.2. Neste programa é oferecido um vasto leque de opções para a configuração independente dos *Dynamixels*, é ainda possível verificar as condições/estado actual em que se encontram os mesmos.

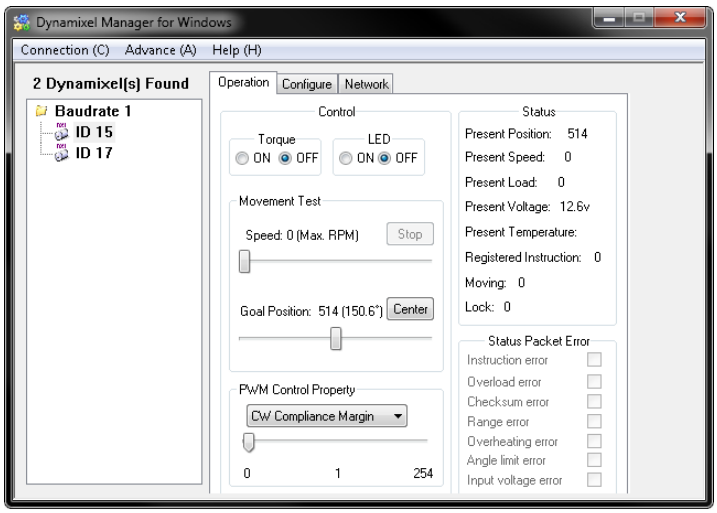


Figura 4.2: Modo de controlo de operações (*Operation*) do *Dynamixel Manager*

No separador *Configure* pode-se efetuar algumas limitações relativamente aos ângulos, voltagem, temperatura e binário. Embora o programa desenvolvido neste projeto (*Educational Manipulator Studio*) possa também fazer algumas dessas configurações, é na mesma possível efetuar previamente e independentemente para cada *Dynamixel* no *Dynamixel Manager*. Conforme consta na Figura 4.3 é possível configurar a função que pretendemos que a luz de alarme (*Alarm LED*) possua e qual a sua consequência. Uma característica bastante interessante neste programa, apesar de neste projeto não ter sido utilizada pois não se enquadra, é a opção de modo de operação (*Operating Mode*). Esta configuração tem duas opções, funcionamento do servomotor como junta, ou seja com 300^{o} de funcionamento, ou modo roda, em que o *Dynamixel* deixa de possuir limitações de ângulos e apenas controlamos sua a velocidade e direcção.

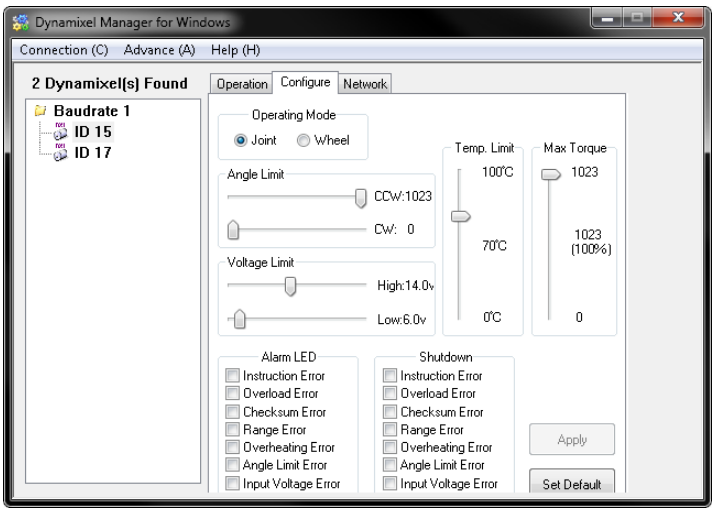


Figura 4.3: Modo de configurações (*Configure*) do *Dynamixel Manager*

Por fim, no separador *Network* (ver Figura 4.4) é permitido configurar a rede de trabalho. Se bem que neste projeto apenas se configurou os números identificativos dos *Dynamixels*. Esta operação de alteração dos IDs é totalmente aconselhada por questões práticas de modo a simplificar a enumeração atribuída aos *Dynamixels*. No caso deste trabalho os *Dynamixels* foram numerados de 1 a 6, contando da base em direcção à garra. Essa ordem numérica identificativa pode ser verificada no capítulo 3.3.

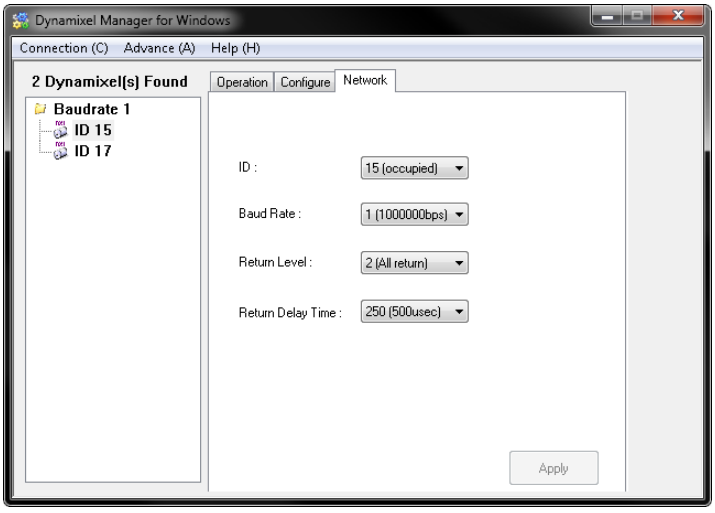


Figura 4.4: Modo de rede de trabalho (*Network*) do *Dynamixel Manager*

4.2 Desenvolvimento da aplicação

Neste sub-capítulo encontra-se descrito todo o desenvolvimento efetuado de modo a ser obtido o produto final pretendido. À parte deste desenvolvimento será realçado a adição expandida de opções avançadas. Este tópico confere ao utilizador costumizar ao seu agrado o modo de controlo do seu robô manipulador, o que permite uma maior versatilidade e flexibilidade ao programa em si. Deste modo, encontram-se apresentadas e exemplificadas essas mesmas opções para uma melhor compreensão das mesmas.

Uma vez explicado o significado do conteúdo do programa encontra-se, também, explicado o seu modo de funcionamento que consiste em dois modos de programação separados. Um desses modos é a básica programação ponto a ponto por aprendizagem e o outro é uma aproximação à programação em *RAPID*. Descrito como aproximação pois considera-se que este tipo de programação aqui utilizada ainda não se encontra completa, se bem que os seus conceitos básicos já se encontram presentes. Completar devidamente este modo de programação caberá como sugestão a ser desenvolvido futuramente.

4.2.1 Menu de configuração

No separador *Configuration*, ver Figura 4.5, encontram-se duas caixas separadoras de grupos. Na caixa da esquerda é possível alterar os números identificativos correspondentes a cada servomotor, ou seja, é necessário anunciar qual é o ID correspondente a cada servomotor *Dynamixel*. Já na caixa da direita encontram-se as configurações referentes à ferramenta/garra. É também possível declarar quais os IDs relativos a cada *Dynamixel* que compõem a garra. Neste mesmo espaço delimitado pode-se ainda definir quais são os ângulos que o utilizador pretende que a garra execute quando se encontra aberta e fechada, sendo ainda possível testar esses mesmos ângulos antes de se guardarem as configurações.

Realça-se o facto de ser sempre necessário guardar as configurações no botão *Save Configuration* para que estas surjam efeito.

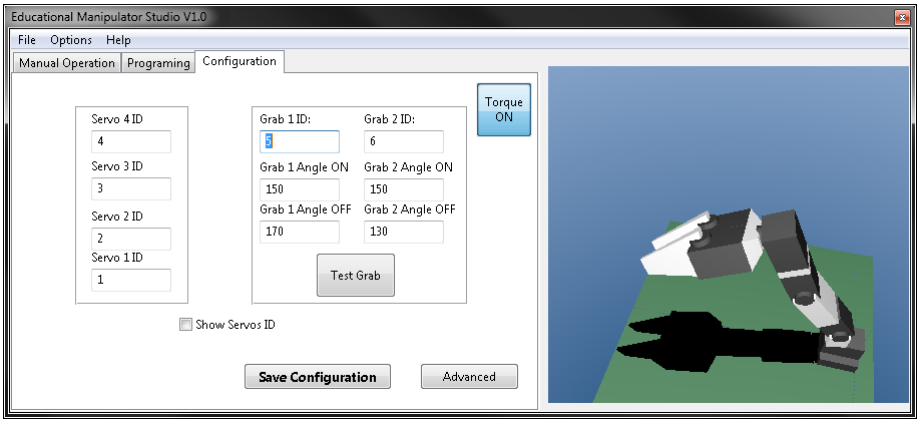


Figura 4.5: *Educational Manipulator Studio V1.0* - separador Configuration

Ainda no separador *Configuration* consta o botão *Advanced*, que caso seja pressionado a aplicação exibe uma nova janela que contém as configurações

avançadas. O conteúdo dessa janela encontra-se relatado mais à frente ainda neste capítulo.

Por fim, no mesmo separador existe uma *check box* (caixa de verificação), que permite selecionar se quer ou não visualizar os IDs correspondentes nos servomotores representados no modelo 3D. A Figura 4.6 exemplifica essa mesma situação de acordo com os dados expostos na Figura 4.5.

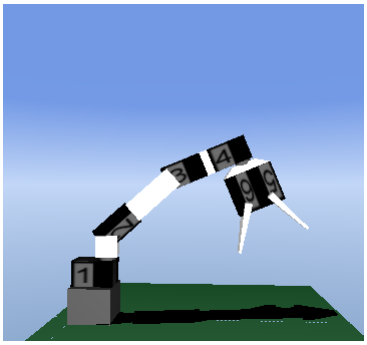


Figura 4.6: Aparência do modelo 3D quando se seleciona a opção *Show Servos ID*

Para um funcionamento correto da aplicação *Educational Manipulator Studio* os números exibidos nos servomotores do modelo 3D devem corresponder exactamente pela mesma ordem nos *Dynamixels* do robô manipulador.

Conforme se pode constatar pela observação das Figuras 4.11, 4.12 e 4.5, embora exteriormente aos separadores que a aplicação possui, existe um modelo 3D do robô manipulador. Este referido modelo acompanha permanentemente todos os movimentos do próprio robô manipulador, o que confere uma significativa melhoria visual a esta aplicação. O modo como este elemento tridimensional foi construído encontra-se descrito no seguinte capítulo 5.

A adição de opções avançadas a esta aplicação oferece ao utilizador desta, um maior controlo sobre o robô manipulador. Na Figura 4.7 podemos ver o aspecto da janela *Advanced*, que é relativo às configurações avançadas.

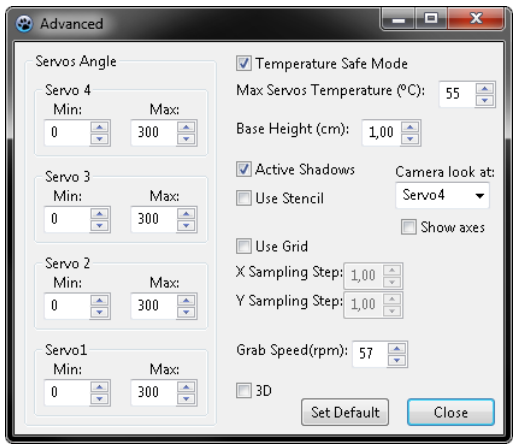


Figura 4.7: Janela das configurações avançadas

De modo a explicar devidamente o conteúdo desta janela, inicia-se por aclarar a funcionalidade desempenhada pela caixa de grupo designada por *Servos Angle* (ângulo dos servomotores). Na Figura 4.8 podemos ver a referida caixa de grupo e a sua ação desempenhada na aplicação. Esta configuração permite então limitar individualmente os ângulos máximo e mínimo dos servomotores, sendo para isso necessário alterar os valores contidos nas caixas de texto editáveis correspondentes. Cada alteração efetuada surge efeito gráfico imediato que se encontra visível nas barras deslizantes no separador *Manual Operation* da aplicação. Estas barras exibem a azul os ângulos que são permitidos de alcançar, tal como se encontra exemplificado na Figura 4.8, se observar atentamente a imagem verá que na tabela que mostra a posição actual dos ângulos contém apenas valores contidos entre os limites configurados nas opções avançadas. Esta configuração permite em certos casos evitar colisões com o ambiente que rodeia o manipulador ou com ele próprio.

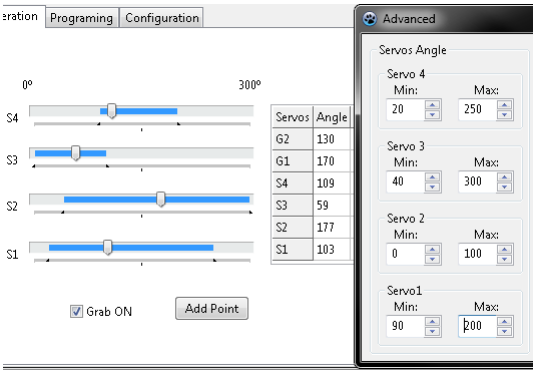


Figura 4.8: Limitar os ângulos máximo e mínimo dos *Dynamixels*

Para a elaboração deste trabalho foi pedido que tivesse em conta a temperatura dos *Dynamixels* de modo a que não fossem danificados, isto porque tal como foi comprovado com o uso do manipulador em fase de testes, estes

aqueciam demasiado, pondo em risco o seu funcionamento.

Deste modo criou-se um algoritmo de segurança que desativa de imediato o funcionamento dos *Dynamixels* caso estes excedam um limite de temperatura. A esta opção denomina-se de *Temperature Safe Mode*, que para além de poder ser ativo e desativo é possível atribuir o valor de temperatura em graus Celsius que se considere ser o limite máximo de funcionamento. Para o caso deste limite ser ultrapassado os *Dynamixels* são automaticamente desligados, a cor de fundo da tabela contida no separador *Manual Operation* muda para vermelho e é exibida uma mensagem de aviso, tal como se pode constatar na Figura 4.9. Só é possível fechar este mensagem de erro caso a temperatura do *Dynamixel* em causa retorne a valores aceitáveis, ou seja inferior ao imposto pelo utilizador.

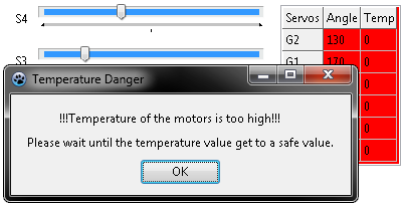


Figura 4.9: Mensagem de erro exibida quando a temperatura excede o limite máximo

A opção *Base Height* (altura da base) permite ao utilizador ajustar o modelo 3D ao robô manipulador relativamente à espessura da peça que interliga a base ao próprio robô. Esta altura pode variar de robô para robô conforme a maneira de como este foi fixo à base, sendo então possível introduzir nesta opção a distância em centímetros entre o primeiro servomotor e a peça que serve de base, no caso da Figura 4.10 a altura introduzida foi de 4cm.

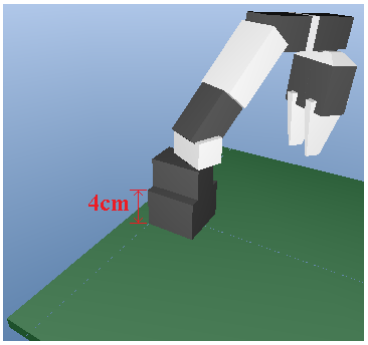


Figura 4.10: Alteração da altura da base para 4cm

É também nesta mesma janela, o único local, onde é possível alterar a velocidade de funcionamento da garra em *Grab Speed*(rpm). A programação do robô utiliza a velocidade para a garra que estiver indicada nesta caixa de texto, sendo a velocidade máxima de 114rpm.

Encontra-se ainda disponível a opção *Camera look at* (câmara olha para), em que simplesmente é possível escolher qual é a peça do modelo 3D que se pretende que a câmara se direcione, ou seja, o servomotor para o qual é pretendido que

se encontre sempre numa posição central da imagem produzida pelo modelo do robô.

A última opção disponível é *Show axes* (ver eixos), em que caso esta seja seleccionada pode-se ver um conjunto de eixos ortogonais XYZ. A cor destes eixos é Vermelha, Verde e Azul correspondentemente a XYZ. Este eixo encontra-se numa posição que é considerada a origem dos eixos do robô, que é junto à base e debaixo do servomotor 1.

Por fim resta referir a funcionalidade do botão *Set Default* (definir padrão), que caso o utilizador clique nele, todas as configurações avançadas são restauradas para valores pré-configurados, ou seja os valores que se encontram exibidos na Figura 4.7.

As outras opções relativas ao uso das sombras, da grelha e da visualização em 3D encontram-se descritas no capítulo 5.2, onde juntamente é feita uma descrição da construção do simulador 3D.

4.2.2 Modo de operação manual

Na Figura 4.11 o conteúdo do separador *Manual Operation*, esta divisória condensa os antigos separadores *Torque based record* e *Hapatic record (Manual)* e é o único local onde se tem acesso à tabela de estados dos *Dynamixels*. A esta referida tabela foi-lhe reduzido o seu conteúdo, sendo que passou a conter apenas a informação mais importante para o utilizador, ou seja o ângulo dos servomotores e a sua correspondente temperatura. Todas as posições dos *Dynamixels*, ou seja os ângulos, são agora exibidas em graus e não numa escala de 0 a 1023. Esta conversão foi estabelecida por uma conversão linear descrita na equação 4.2.1.

$$\hat{A}ngulo = \frac{300 * Posição}{1023} \tag{4.2.1}$$

Embora o utilizador veja sempre a posição dos servomotores em graus a aplicação trabalha sempre na escala própria dos *Dynamixels* (entre 0 e 1023), isto porque é mais fácil de processar essa informação quando se gravam ficheiros e se abrem em outro local, como será explicado seguidamente no capítulo 4.2.3.

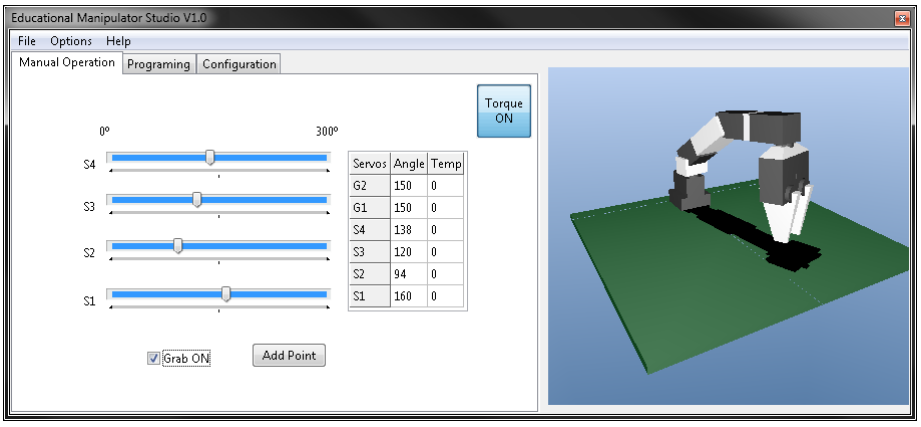


Figura 4.11: *Educational Manipulator Studio V1.0* - separador *Manual Operation*

As barras deslizantes contidas neste separador servem para alterar a posição do próprio robô manipulador e do seu modelo 3D. Caso o botão de *Torque* se encontre em *OFF* e se alterar-se manualmente as posições dos *Dynamixels*, estas barras acompanham individualmente a movimentação do robô, formando também um elemento visual de monitorização.

O separador *Programing* que consta na Figura 4.12, é o único local onde é possível alterar a velocidade de funcionamento do robô manipulador excluindo sua a ferramenta. Este controlo centralizado da velocidade do manipulador simplifica consideravelmente o modo de programação relativamente a velocidades. Essa velocidade que é exibida em *Servos Speed* vem em rpm em troca da antiga referência entre 0 e 1023. De um modo semelhante à conversão dos ângulos, foi realizado também através de uma conversão linear essa mesma conversão, a equação em causa é a equação 4.2.2.

$$Velocidade(rpm) = \frac{114 * Velocidade(Dynamixel)}{1023} \tag{4.2.2}$$

4.2.3 Modo de programação

Conforme se pode observar na Figura 4.12 este separador contém duas caixas que agrupam funcionalidades distintas dentro do módulo de programação. A caixa de cima agrupa a interface que apenas permite efetuar a dita programação e a caixa de baixo é relativa à execução dos programas construídos. Sendo que os únicos elementos que se encontram fora dela são o botão de *Torque ON/OFF* e o botão *Programing in RAPID*. Conforme se pode verificar a presença do botão de *Torque* encontra-se em todos os separadores, esta característica singular foi atribuída unicamente a este botão porque ele serve de interruptor de emergência, podendo deste modo ser desactivado em qualquer altura o binário exercido pelos *Dynamixels* evitando problemas de maior.

Quanto ao botão *Programing in RAPID* em caso de ser clicado é exibida uma nova janela, esse assunto encontra-se explicado mais pormenorizadamente no capítulo 4.2.3.2.

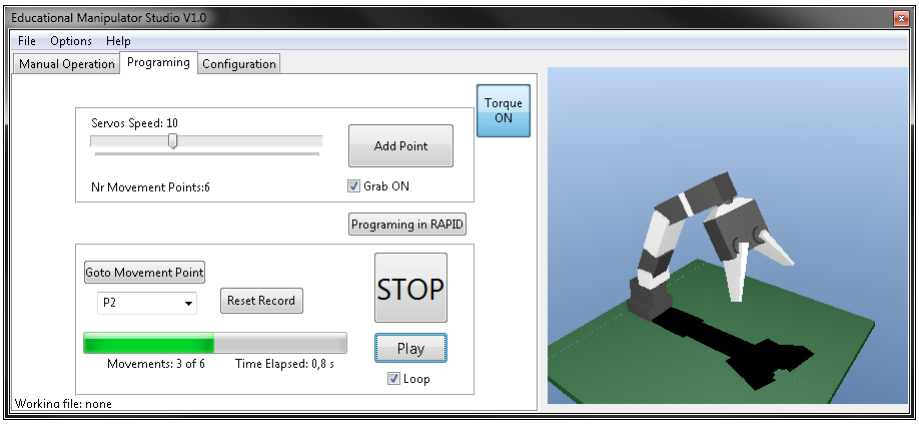


Figura 4.12: *Educational Manipulator Studio V1.0* - separador *Programing*

4.2.3.1 Programar por aprendizagem de pontos

Este modo de programação já se encontrava embutido na aplicação anterior a este projeto, sendo portanto um modo muito simples, mas eficaz, de programar os movimentos do robô manipulador. Actualmente o painel de controlo principal deste tipo de programação encontra-se no separador *Programing* da aplicação *Educational Manipulator Studio*, que se encontra exibido na Figura 4.12.

Numa fase inicial da programação é necessário adicionar as posições que se pretendem que o robô execute. Mas antes de o fazer, o utilizador deve ter em atenção os valores das velocidades que vai requerer, bem como a posição da garra. Uma vez que as velocidades dos movimentos e a posição da garra se encontrem calibradas nos valores eleitos pelo utilizador é necessário colocar o robô manipulador na posição que se intenciona gravar. Para isso e com o botão *Torque* em OFF, é possível regular essa mesma posição do robô manualmente alterando a posição dos *Dynamixels*. No caso de se desejar manobrar o robô manipulador com o binário ativo, botão *Torque* em ON, basta ir ao separador *Manual Operation* e alterar as posições das barras deslizantes para os valores requeridos.

De modo a gravar cada posição do robô manipulador, basta clicar no botão *Add Point*. Cada vez que este mesmo botão é premido é incrementado um ponto representado por Pi (em que i varia entre 0 até $+\infty$, teóricamente), na *Combo Box* (caixa de combinações), que se encontra do lado esquerdo do botão *Reset Record*. É portanto nesta mesma caixa onde se pode ter acesso a todos os pontos gravados.

De modo a construir um programa resta então ao utilizador adicionar quantos pontos forem do seu interesse. Para gravar essas posições é necessário aceder a *File* seguido de *Save* ou *Save as* e indicar o diretório onde se deseja gravar os pontos. Se o utilizador gravar um ficheiro com os pontos, a localização do ficheiro em que se encontra a trabalhar é exibida no fundo do separador *Programing* (em *Working file*), e só será alterada no caso de se utilizar a opção *Save as* e gravar num outro directório.

Uma vez completa a adição de posições, é possível correr esse mesmo programa premindo o botão *Play*. A execução do programa consiste em o robô adquirir as posições previamente gravadas, pela mesma ordem em que estas mesmas foram adquiridas. Uma barra de progresso permite ao utilizador visualizar em que estado a execução do programa se encontra. De modo a completar esta percepção do estado onde o programa se encontra é exibido também o número específico do movimento que o robô se encontra a executar, bem como o tempo que demora a sequência de movimentos até ao movimento actual.

Para o caso de o utilizador querer repetir ciclicamente a execução do seu programa, basta activar a opção *Loop* (ciclo), e o programa executa os movimentos que foram adicionados repetidamente até que o utilizador intervenha para parar. Essa é a função do botão *STOP*, que tal como o botão *Torque*, tem umas dimensões relativamente maiores de modo a servir também de botão de paragem de emergência.

Na eventualidade do utilizador pretender aceder diretamente a um ponto no meio da sua lista de pontos, apenas necessita de o seleccionar na anteriormente já referida *Combo Box* (onde os pontos são armazenados/exibidos), e clicar no botão *Got Movement Point*.

Por fim se o utilizador desejar apagar a lista de pontos/movimentos gravados,

tem ao seu dispor o botão *Reset Record*, que limpa definitivamente os pontos inseridos na lista de movimentos.

4.2.3.2 Programar em RAPID

Uma segunda opção de programação é o modo RAPID, em que para se ter acesso a esta janela, como já foi anteriormente referido, é necessário clicar no botão *Programing in RAPID* que se encontra no separador *Programing* da aplicação *Educational Manipulator Studio*. Uma vez premido o referido botão, aparece a janela exibida na Figura 4.13. Pelo facto de o utilizador estar a aceder a uma nova interface de programação, não significa que o resto da aplicação se encontre desativada, pelo contrário, todas as opções e interfaces serão alteradas normalmente pelo regular manuseio do robô manipulador.

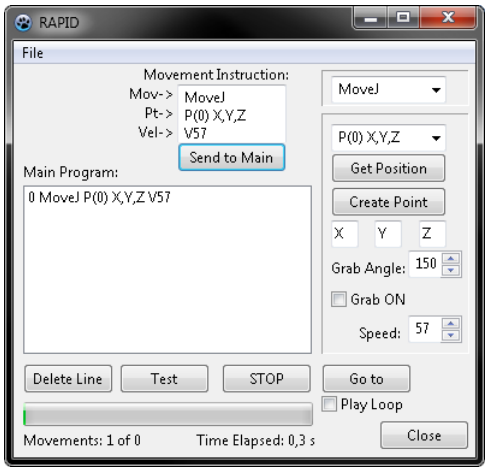


Figura 4.13: Janela de programação em *RAPID*

Embora esta janela se dedique à programação em *RAPID*, também tem incluído o modo de obtenção de pontos de forma idêntica à programação por aprendizagem descrita no Capítulo 4.2.3.1. O interesse de manter o mesmo modo de se obterem as posições dos *Dynamixels*, permite um intercâmbio de ficheiros que contém a informação desses mesmos pontos. Ou seja, quer os pontos sejam gravados na referida janela ou no resto da aplicação, eles podem ser utilizados em ambos os lados, sendo que para isso basta guardar e abrir o respectivo ficheiro onde se pretende trabalhar com ele.

Uma vez que a programação em *RAPID* se baseia em instruções de movimentos, existe nesta janela uma caixa de texto editável dedicada exclusivamente à construção destas mesmas instruções de comando, que se chama *Moviment Instruction*. Nesta caixa é necessário a inclusão de três principais variáveis pela ordem correcta, estas variáveis são movimento, ponto e velocidade, que se encontram abreviadas por *Mov*, *Pt* e *Vel* respectivamente. O modo de se construir as instruções de movimento tem obrigatoriamente de se iniciar por escolher um tipo de movimento em *Movements* (movimentos), os movimentos que se encontram disponíveis são *MoveJ*, *MoveL* e *MoveC*, que correspondem a movimentos do tipo aleatório, linear e circular respectivamente.

Ainda antes de prosseguir na construção da instrução de movimento, é de frisar, que o modo de como as trajectórias são executadas não é influenciado pela escolha destes movimentos, isto porque, infelizmente estas instruções ainda não se encontram devidamente programadas, assim sendo todos os movimentos serão do tipo *J*, o que significa que são trajectórias ponto a ponto. Fica então para trabalho futuro desenvolver um algoritmo que corresponda a cada tipo de movimento.

Uma vez selecionado o tipo de movimento este será exibido na caixa correspondente a *Movement Instruction* à direita da sinalética que o representa, ou seja *Mov*.

Seguidamente é requerido adicionar um ponto que corresponde a uma posição do robô. De modo a serem adquiridos os pontos basta carregar um programa em que já tenham sido previamente gravados ou então de um modo semelhante à programação por aprendizagem, as posições podem ser adicionadas em *Get Position*. Premindo este botão, um novo ponto é adicionado à lista de pontos contida em *Points* (pontos/posições), com a informação relativa às posições dos *Dynamixels* 1, 2 e 3, sendo que para o *Dynamixel* 4, vai possuir o valor exibido em *Grab Angle* (ângulo da garra). Por sua vez as posições dos servomotores que integram a própria garra ficam dependentes da selecção de *Grab ON*, que são os ângulos inseridos nas configurações. Por fim um valor de velocidade é adicionado. Sendo que quando se seleciona o ponto que se pretende utilizar na instrução de movimento este é imediatamente transferido para a caixa *Movement Instruction*, assim como o valor da velocidade que se encontra selecionado. Uma vez que a instrução de movimento se encontre devidamente construída, o utilizador tem acesso ao botão *Send to Main* (enviar para principal). Este botão permite inserir as instruções de movimento, devidamente construídas, na caixa não editável *Main Program* (programa principal). É então nesta caixa que a programação toma aspectos de instruções reais em *RAPID*. Por fim para guardar e abrir a programação dita principal basta aceder a *File* (ficheiro) e seleccionar a instrução que se pretende.

Por fim seria possível correr um programa contido em *Main Program*, bastando para isso carregar em *Test*, muito à semelhança do botão *Play* do separador *Programing*. Outras parecenças também extraídas do modo de programação por aprendizagem são os botões *STOP* e *Go to* e ainda a opção *Play Loop* que correspondem aos anteriores *STOP*, *Goto Movement Point* e *Loop* respectivamente. A função destes elementos é portanto a já descrita anteriormente no Capítulo 4.2.3.1.

Algumas funcionalidades desta janela, não se encontram ainda totalmente a funcionar, isto porque este trabalho serve de complemento do trabalho anteriormente realizado. Ou seja, esta plataforma que foi desenvolvida encontra-se preparada para suportar o trabalho paralelo que já foi desenvolvido. A quem trabalhar futuramente no desenvolvimento desta aplicação é sugerido que interligue as aplicações anteriores, desenvolvidas em Matlab, com esta que foi desenvolvida neste projeto.

4.2.4 Barra de menus

Ainda neste capítulo de desenvolvimento da aplicação de controlo e supervisão, pode-se ver na Figura 4.14, que também a barra de menus sofreu modificações.

Na opção *File* desta barra é possível guardar e abrir os programas relativos

ao modo de programação ponto a ponto (descrito no capítulo 4.2.3.1), encontra-se também disponível a opção *Exit* que tal como o *X* no canto superior direito da aplicação, em caso de ser seleccionado, a aplicação e todas as suas janelas associadas são encerradas saindo da aplicação.

A barra de menus contém a opção *Options*. Embora nesta versão da aplicação ainda não se encontre funcional, já existe o espaço reservado para pacotes de línguas, nomeadamente em Inglês, Português e Espanhol. Contudo a língua base é o Inglês e esta característica não pode, infelizmente, ser alterada no *Educational Manipulator Studio V1.0*, sendo portanto uma sugestão de futuro trabalho a ser desenvolvido.

Em *Options* dispõe-se ainda de uma calculadora e de um calendário. Destaca-se que estes elementos não foram desenvolvidos ao longo deste trabalho (mas sim adicionados), sendo o seu aspecto de acordo com o *Windows* instalado no computador.

O último elemento que consta em *Options* é deveras útil, denomina-se *ONLINE Mode*. Caso este elemento se encontre seleccionado o modelo 3D do robô acompanha todos os passos dados em simultâneo do manipulador. Por outro lado se esta opção não estiver verificada só é possível mover o modelo 3D através da alteração da posição das barras deslizantes contidas no separador *Manual Operation*. A enorme vantagem que esta opção permite é de que se torna possível construir um programa sem ter o robô conetado ao computador. Uma outra característica desta opção quando selecionada é visível no separador *Manual Operation* o botão *Refresh Angles*, este botão quando premido recebe a posição dos *Dynamixels* e actualiza a aplicação com esses valores. Se o *ONLINE Mode* não se encontrar seleccionado em vez do botão *Refresh Angles*, é visível um botão *Add Point* que facilita a programação *OFFLINE*.

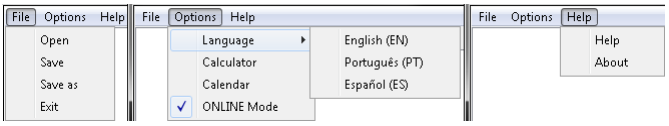


Figura 4.14: Barra de menus expandida

Por fim encontra-se disponível a opção *Help* (ajuda). Nesta opção, encontra-se novamente a opção *Help* que no caso de se aceder esta, a aplicação abre uma nova janela com um menu interactivo do tipo árvore de nós, conforme se pode observar na Figura 4.15. Tal como o resto da aplicação, a linguagem aqui utilizada é novamente o Inglês, esta escolha remete ao facto de esta língua ser considerada uma língua universal e portanto de fácil compreensão na área das engenharias. Embora este manual do utilizador não se encontre totalmente completo, nesta versão já é possível obter um guia para a iniciação à utilização desta aplicação. Fica então como sugestão de trabalho futuro um desenvolvimento das ajudas prestadas por esta opção.

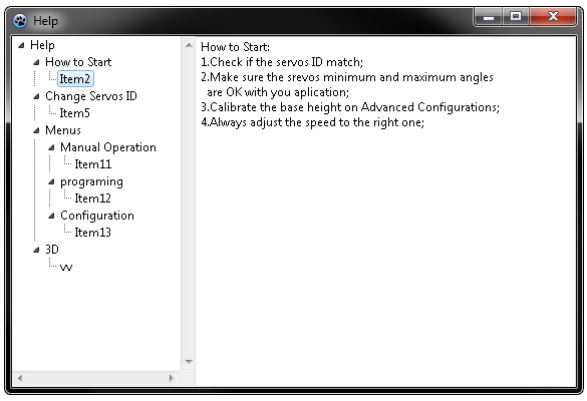


Figura 4.15: Janela *Help*

De modo a concluir a descrição da barra de menus, ainda na opção *Help*, pode-se encontrar a subopção *About*. Caso esta opção seja acedida aparece uma mensagem no canto superior do ecrã com informação relativa à versão da aplicação, instituição patrocinadora e os autores desta aplicação, ver Figura 4.16.

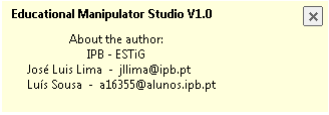


Figura 4.16: Mensagem *About*

Todas as descrições anteriores contidas neste capítulo são relativas à aparência da aplicação, que o utilizador final tem acesso. Contudo, estão ocultos alguns elementos cruciais para o funcionamento desta aplicação, tal como podemos ver na Figura 4.17 que é relativa à janela principal da aplicação em fase de edição.

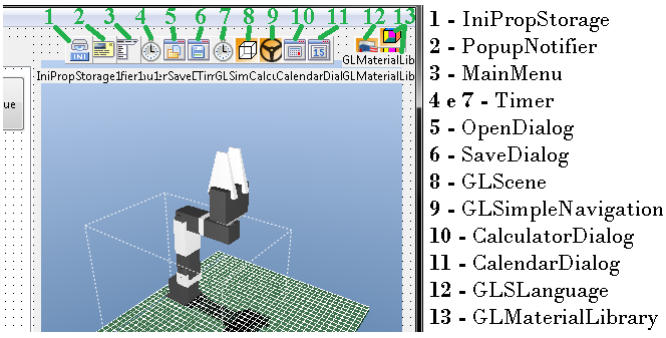


Figura 4.17: Elementos *Lazarus* e *GLScene* ocultos e sua legenda

Seguidamente, adiciona-se a este capítulo uma breve descrição da funcionalidade de estes elementos.

1. *IniPropStorage* - Este elemento encontra-se também presente na janela das configurações avançadas (*Advanced*), a função desempenhada por este

componente é a de guardar as últimas alterações efectuados pelo utilizador antes de fechar a aplicação. Para configurar este elemento é necessário aceder ao *Object Inpector* do *Lazarus* e adicionar os dados que se pretendem guardar de uma sessão para a outra em *SessionProperties*.

2. *PopupNotifier* - Este componente permite criar a caixa de texto que para este caso foi utilizada para fornecer ao utilizador alguma informação sobre a versão da aplicação, instituição patrocinadora e os autores da mesma, ver Figura 4.16.

3. *MainMenu* - Tal como o próprio nome indica, este elemento cria a barra de menus, sendo que os seus elementos foram adicionados manualmente na interface fornecida ao clicar no referido elemento, ver Figura 4.14.

4. *Timer* - É um relógio que conta de quanto em quanto tempo se faz a leitura dos sensores dos *Dynamixels*.

5. *OpenDialog* - Este componente fornece acesso à janela do *Windows* de abrir ficheiros, este também se encontra na janela *Programing in RAPID*, em que nesta mesma janela não abre unicamente ficheiros que contêm os pontos, como na janela principal da aplicação, mas abre também o texto do programa principal.

6. *SaveDialog* - À semelhança do *OpenDialog* este componente desempenha a função complementar não de abrir ficheiros mas sim de os guardar, o formato em que são gravados é editável e para este caso foi escolhido *.mvm.

7. *Timer* - É um relógio que permite contar o tempo de quando o programa se encontra em execução (*Play*).

8. *GLScene* - Este componente permite criar um ambiente 3D. Foi aqui que foi construído o modelo tridimensional do manipulador, a maneira de ser trabalhar no referido componente encontra-se anteriormente descrita neste capítulo.

9. *GLSimpleNavigation* - A adição deste elemento permite que o utilizador da aplicação possa mover a posição da câmara ao seu agrado, navegando facilmente pelo modelo 3D.

10. *CalculatorDialog* - Quando este componente é invocado aparece uma calculadora simples para uso diverso.

11. *CalendarDialog* - À semelhança da calculadora, quando solicitado, aparece um calendário.

12. *GLSLanguage* - Este elemento permite criar pacotes de línguas e alterá-las com a aplicação em funcionamento.

13. *GLMaterialLibrary* - Permite alterar a textura dos objectos 3D, neste caso aparecem números dos IDs escritos nos servomotores do modelo 3D.

Deste modo encontra-se concluída toda a descrição pormenorizada do modo de funcionamento da aplicação controladora do robô manipulador, no seguinte capítulo é descrito o modo de como foi desnvolvido o simulador 3D.

Capítulo 5

Simulador 3D

Este capítulo explica o que foi utilizado em termos visuais para o desenvolvimento da aplicação *Educational Manipulator Studio*. De modo a completar visualmente esta aplicação, foi necessário adicionar a componente gráfica em três dimensões (3D). Sendo que para isso foi necessário adicionar a componente *GLScene* ao programa *Lazarus*. Após uma breve descrição do que é o *GLScene*, encontra-se descrito em detalhe os passos que foram tidos em conta para a construção do simulador 3D

5.1 GLScene

O *GLScene* é uma biblioteca baseada em *OpenGL 3D*. Inicialmente este produto foi apenas desenvolvido para *Delphi*, sendo que nos dias de hoje pode ser executado em *C++ Builder*, *Kylix*, *Delphi* e *Lazarus*. Ele fornece componentes visuais e objectos permitindo a descrição e renderização de cenas 3D de uma forma fácil e sem complicações mas de uma forma poderosa. [4]

O *GLScene* não é apenas uma biblioteca *OpenGL* ou um mero utilitário, ele cresceu e tornou-se num conjunto de classes fundadoras focado num motor genérico 3D com *Rapid Application Development* (*RAD* - Aplicação de Desenvolvimento Rápido). Esta componente permite rapidamente criar e renderizar cenas 3D sem ter que aprender as complexidades de *OpenGL*, se o seu utilizador sabe como criar um *TForm*, então facilmente domina as operações básicas de *GLScene*. Esta biblioteca vem com uma larga gama de coleções de demonstrações mostrando a facilidade do seu uso, e demonstrando também que *RAD* não foi feito à custa de unidades de processamento de dados e gráficos muito potentes.

As demonstrações desta aplicação encontram-se em formato para *Delphi*, sendo que para os podermos abrir em *Lazarus* estes projetos/aplicações necessitam de ser convertidos. Para isso é necessário aceder às ferramentas do *Lazarus* e executar a conversão de projetos *Delphi* para projetos *Lazarus* (*Tools -> Convert Delphi project to Lazarus project*). Depois de se escolher qual o exemplo a ser convertido é necessário clicar em começar a conversão (*Start convesion*). Seguidamente, é exibida uma janela com as unidades em falta para o funcionamento do projeto, basta então sair dessa janela (*Comment Out*), abrir as *Form* relativas a essa demo e adicionar as unidades em falta. Na Figura 5.1 é possível

ver o ambiente de uma das demos criado pela componente *GLScene* no *Lazarus*. [6]

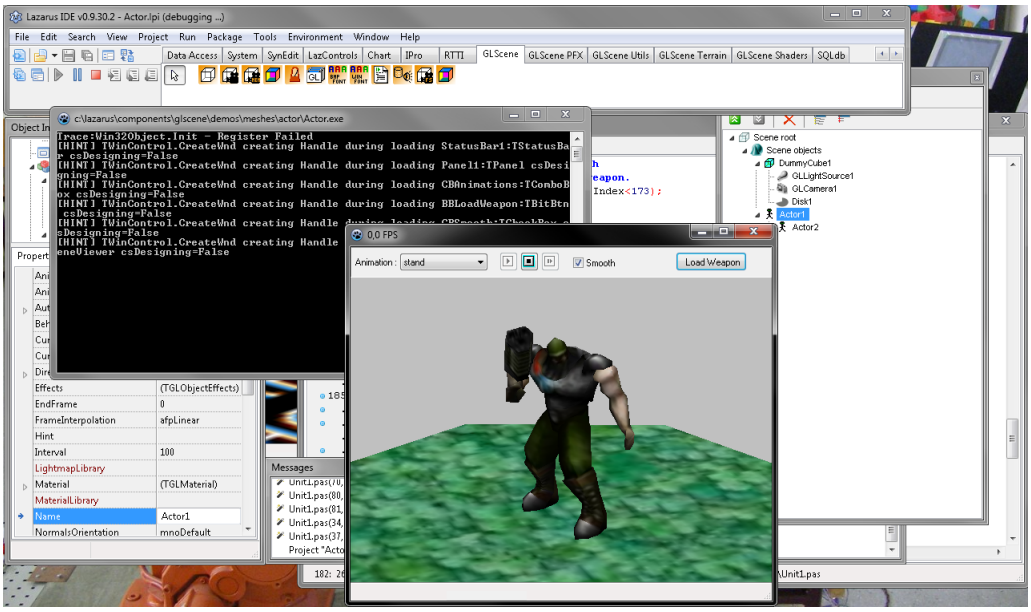


Figura 5.1: Demonstração *GLScene* (*meshes - actor*)

Uma análise cuidada destas demonstrações incluídas no pacote *GLScene* foi de extrema importância para o desenvolvimento deste projeto, isto porque fornece várias ideias daquilo que é possível construir numa aplicação em *Lazarus*.

No seguinte capítulo 5.2 é descrito aprofundadamente o modo de trabalhar com a referida componente.

O guia de instalação passo a passo desta componente no *Lazarus* encontra-se disponível no Apêndice B.

5.2 Construção do modelo 3D

Para construir o modelo tridimensional do robô manipulador é necessário iniciar por construir um cenário. Para isso, tal como se pode ver na Figura 5.2, é fundamental adicionar os elementos chave *GLScene* e *GLSceneViewer*. Sendo que este segundo elemento funciona como se fosse uma janela através da qual é possível ver o cenário. Quanto ao componente *GLScene*, tal como já foi descrito anteriormente neste documento, é o elemento que permite criar e editar o cenário e os seus elementos nele contidos.

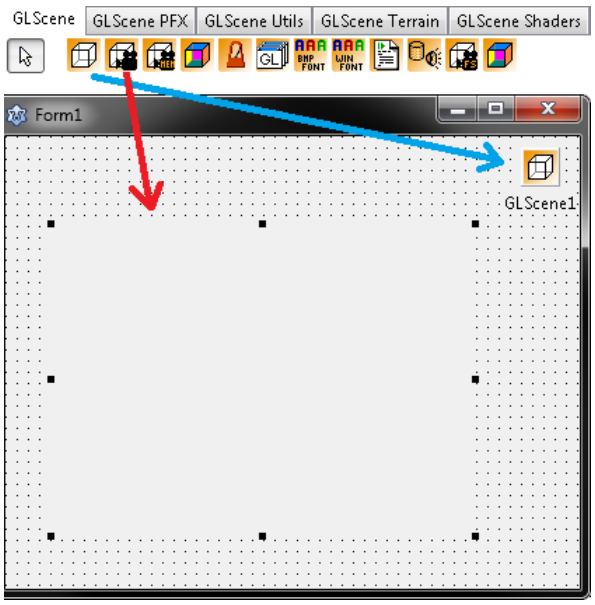


Figura 5.2: Adicionar uma *GLScene* e *GLSceneViewer*

Após a adição destes dois componentes, existem outros componentes que são também eles de extrema importância. É portanto à necessidade obrigatória de todos os cenários necessitarem de uma *Camera* (câmara). Para se adicionar este elemento, tal como se pode ver na Figura 5.3, temos em primeiro lugar de abrir o *GLScene Editor* (editor de cenários), sendo que para isso é preciso fazer duplo clique no ícone *GLScene*. Seguidamente, clicando com o botão direito do rato em *Scene objects*, aparece a janela de ferramentas indicada na mesma figura com o número 1. Para, então, adicionar uma câmara basta clicar em *Add camera*.

O *GLSceneViewer* necessita indispensavelmente de ter uma câmara associada, este passo faz-se rápidamente no *Object Inspector* do *GLSceneViewer*. Uma vez que este componente tenha uma câmara a ele associado já é possível ver o cenário criado na janela, se bem que se este se encontrar sem elementos tal facto não vai ser diferenciado.

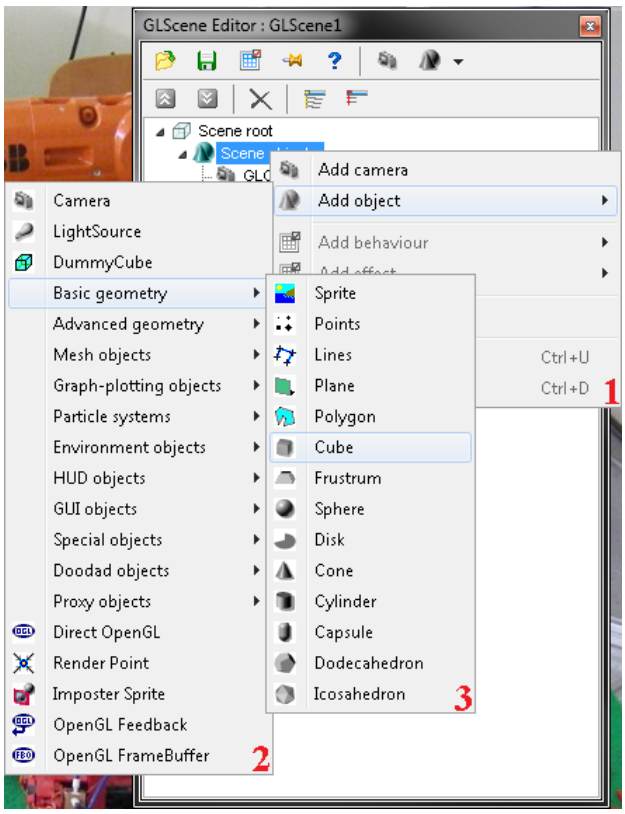


Figura 5.3: Como adicionar objectos

Embora não seja obrigatório é de boa prática adicionar-se um *DummyCube* (cubo manequim) aos cenários. Este cubo permite ter uma noção das dimensões dos objectos adicionados, a sua distância e ângulos relativamente ao eixo origem e permite ainda delimitar a área de trabalho, facilitando no início da construção do cenário ter a câmara constantemente direccionada para este cubo. Este cubo só aparece na janela desenvolvimento e nunca na aplicação.

De modo a melhorar o ambiente tridimensional é aconselhado adicionar uma ou várias *LightSource* (fontes de luz). A adição deste elemento confere ao cenário uma maior facilidade de compreender a forma e posição dos objectos 3D nele contidos.

Por fim, como não podia deixar de ser, é necessário adicionar os objectos. Para isso segue-se as instruções ilustradas na Figura 5.3, de modo a se adicionar um cubo. Todos os objectos adicionados necessitam de ser editados. em primeiro lugar deve-se atribuir as dimensões pretendidas, depois devemos associar no *GLScene Editor* a um outro elemento já existente (normalmente a outro objecto). Este relacionamento deve ser construído tendo em conta o mecanismo das peças físicas. Após a associação do novo objecto se encontrar correcta é necessário alinhar o posicionamento do novo objecto de modo a construir o modelo do robô.

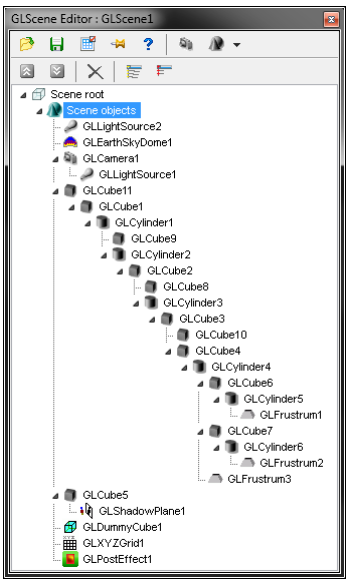


Figura 5.4: *GLScene Editor* finalizado

Depois de todos os objectos se encontrarem devidamente associados e na sua posição correta é possível adicionar alguns detalhes ao cenário criado. Os detalhes contidos neste projeto foram *GLEarthSkyDome*, *GLShadowPlane*, *GLXYZGrid* e *GLPosEffect*.

Tal como se pode verificar na Figura 5.5 a opção *Active Shadows* e *Use Stencil*, contidas na janela de opções avançadas, são meramente estéticas de modo a melhorar o ambiente gráfico do modelo do robô. O utilizador dispõe portanto de três modos de sombras. Sendo um deles o exibido na Figura 5.5 em *a* que é relativo ao uso de uma sombra normal, em *b* tem aquilo que se chama de *stencil*, isto é, a mesma sombra mas translucida. Finalmente existe também a possíbilidade de não mostrar qualquer sombra, tal como exemplificado em *c*. Este efeito foi conseguido através da associação do elemento *GLShadowPlane* à peça onde se pretende que vejamos a sombra do modelo do robô no *GLScene Editor*.

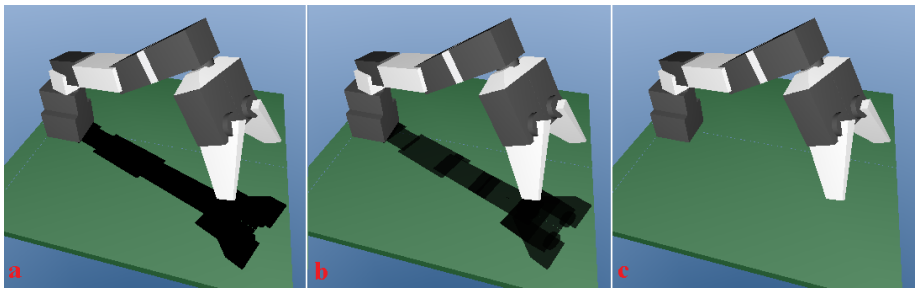


Figura 5.5: a)Modelo 3D com sombra; b)Modelo 3D com sombra em *stencil*; c)Modelo 3D sem sombra

Outra opção disponível nesta janela é *Use Grid* (usar grelha), que conforme

o nome sugere, tem-se acesso a uma grelha. Observando a Figura 5.6 encontra-se um exemplo do uso dessa mesma grelha, em que na referida imagem possui 2cm por 2cm de dimensão entre linhas. Estas dimensões Podem ser alteradas facilmente em *X Sampling Step* e *Y Sampling Step*. O uso desta grelha permite ao utilizador ter uma melhor noção das dimensões da área de trabalho e do consecutivo impacto dos movimentos do robô. Este detalhe foi alcançado com a adição do componente *GLXYZGrid* ao *GLScene Editor*, sendo que a sua localização encontra-se ligeiramente acima da superfície da base.

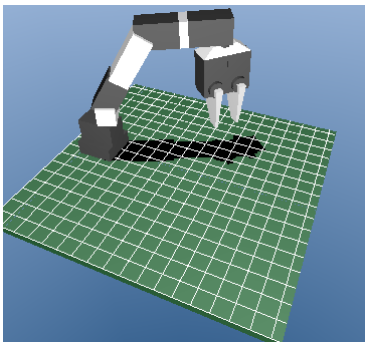


Figura 5.6: Uso da grelha para referência com um passo de 2cm para x e para y

Uma vez que nos dias correntes existe uma grande aposta em imagens em 3D, foi incluído nesta aplicação a opção de visualizar o modelo do robô, com o recurso a óculos próprios para este efeito (lentes azul e vermelho), em três dimensões com o objetivo de o utilizador adquirir uma noção melhor de profundidade. Na Figura 5.7 encontra-se um exemplo de uma posição do modelo do robô com o filtro 3D activo. Na realidade este 3D não é considerado um 3D 'verdadeiro', mas sim uma simulação ou uma ilusão do mesmo. O facto de a janela do ambiente gráfico ser de reduzidas dimensões e de o próprio cenário conter relativamente poucos elementos por vezes torna difícil a sua compreensão total de profundidade. Para desenvolver este filtro foi necessário adicionar o elemento *GLPosEffect* no *GLScene Editor*, posteriormente foi programado o efeito pretendido para este filtro de imagem.

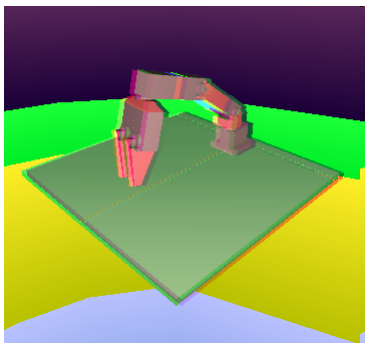


Figura 5.7: Exemplo de uma visualização do modelo 3D em ilusão de 3D de profundidade com auxilio a óculos apropriados

Quanto ao elemento *GLEarthSkyDome* deve ser associado directamente à raiz do cenário de modo a que o utilizador só o veja como tela de fundo do cenário tridimensional.

Para o componente *GLShadowPlane*, este tem de ser associado ao objecto que designa como base, pois é aí onde a sombra vai ser visível.

Neste projeto o aspecto final do editor de cenários é o que se encontra exibido na Figura 5.4. Nesta figura podemos comprovar a complexidade gerada pela interligação dos objectos. Quanto ao aspecto produzido na janela *GLSceneViewer* é dado pela Figura 5.8. Embora nesta imagem pareça que a posição do modelo não corresponda a uma posição que o robô possa adquirir, é de realçar que na aplicação esta posição não é atingida devido às limitações nos ângulos. Serve então esta pré-visualização para se ter uma ideia geral do produto final.

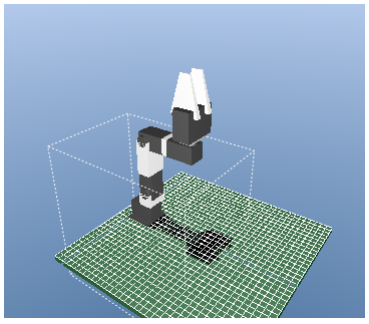


Figura 5.8: Aspecto final do modelo tridimensional do robô em fase de edição

Capítulo 6

Resultados experimentais

Neste capítulo serão expostos alguns exemplos experimentais retirados da utilização da aplicação desenvolvida. Estes exemplos dizem respeito à elaboração de programas construídos nos diferentes modos de programação contidos na aplicação. Seguidamente através de várias críticas e sugestões, são analisados os resultados obtidos ao longo da elaboração deste trabalho.

6.1 Exploração de demonstração

Na aplicação desenvolvida, e como já foi referido, é possível elaborar programas que guardam os movimentos das trajetórias. De modo a exemplificar a programação por aprendizagem de pontos, foi elaborada uma demonstração denominada 'Potes de Mel'. Na Figura 6.1 pode-se ver uma recriação tridimensional do ambiente da demonstração. É importante referir que nesta imagem apenas o modelo do robô e a sua respetiva base, são os elementos retirados da aplicação, sendo que os demais elementos que figuram na imagem foram posteriormente adicionados, apenas para uma melhor compreensão do exemplo. O ambiente de trabalho nesta demonstração contém dois pequenos potes de mel (A e B), e uma peça com forma cilíndrica. O ponto P representa a posição de segurança/descanso. Esta demonstração consiste em mover a peça cilíndrica do topo do pote de mel B para o topo do pote de mel A e seguidamente do pote de mel A para o pote de mel B.

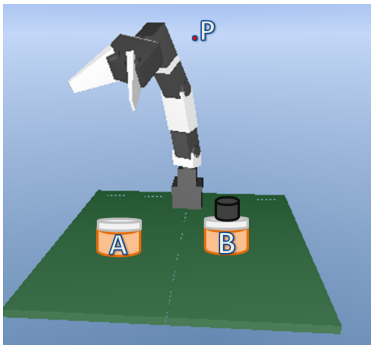


Figura 6.1: Simulação tridimensional (exemplo 'Potes de Mel')

Para uma melhor analise das trajetórias que o robô manipulador executa, encontra-se esquematizado na Figura 6.2 todos os passos efetuados pelo robô.

Numa primeira etapa da demonstração, a peça a deslocar encontra-se em cima do pote de mel B e deve ser deslocada para o pote A. Para isso o robô inicia o seu percurso na posição assinalada como ponto P e dirige-se para um ponto que se situa em cima da peça cilíndrica, de modo a efectuar a aproximação, depois apanha a peça e vai para uma posição em cima do pote A, onde faz a aproximação ao mesmo e larga a peça, por fim afasta-se ligeiramente da peça e retorna à posição inicial (ponto P). Numa segunda etapa da demonstração o comportamento é semelhante ao descrito na primeira etapa, com a diferença de que a peça cilíndrica se encontra no topo do pote de mel A e vai ser deslocada para o cimo do pote de mel B.

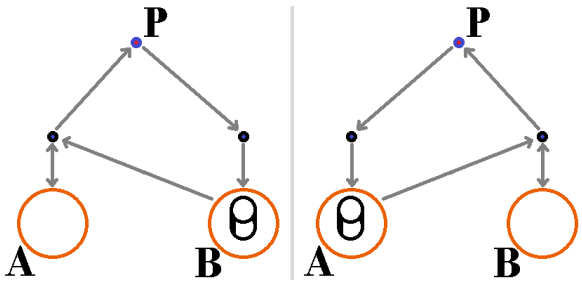


Figura 6.2: Esquema das trajetórias programadas (exemplo 'Potes de Mel')

Após se gravar o ficheiro que contenha os pontos da trajetória, estes são gravados num ficheiro de texto. O aspeto que esse ficheiro toma, se for aberto com o *Bloco de notas* do *Windows* é o apresentado na Figura 6.3 para o exemplo da demonstração 'Potes de Mel'. Tal como se pode observar na referida figura constam sete colunas com vários valores. Sendo que as primeiras quatro colunas representam os quatro eixos de liberdade do robô manipulador e as duas colunas seguintes a posição da garra de acordo com a sua posição, aberta ou fechada. Por fim, pode-se ver na última coluna um valor constante que representa a velocidade em que as trajetórias vão ser feitas quando este exemplo for executado.

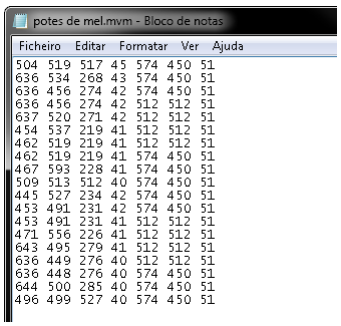


Figura 6.3: Ficheiro texto de pontos guardados (exemplo 'Potes de Mel')

Conforme se pode verificar, novamente, na Figura 6.3, os valores nela contidos não são de fácil percepção, isto porque, se encontram numa escala diferente da exibida na aplicação, ou seja, estão entre 0 e 1023, que corresponde aos comandos *Dynamixel*.

Conforme já foi anteriormente referido neste documento, de um modo semelhante à gravação de pontos, é possível ao utilizador gravar os seus programas construídos em RAPID, sendo que para isso o resultado é o ilustrado na Figura 6.4. O texto contido neste ficheiro diz respeito a um exemplo de uma trajetória genérica como a recriada na Figura 6.5. Para este tipo de programação são evidentes as vantagens no controlo de uma trajetória, isto porque requer menos pontos de referência e pode-se contudo obter resultado final mais perfeito.

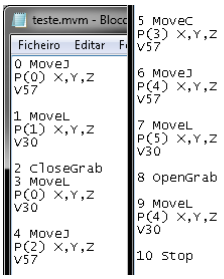


Figura 6.4: Ficheiro texto com instruções do código principal em RAPID

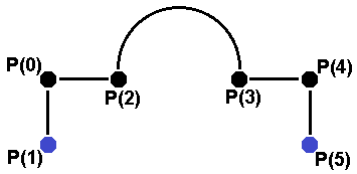


Figura 6.5: Exemplo de trajectória

Finalmente, é importante referir que o modo de programação em *RAPID* fornecido pela aplicação não foi terminado nesta versão da aplicação, sendo que, independentemente do tipo de movimento selecionado, os movimentos executados são sempre do tipo J, o que signica que são trajectórias ponto a ponto.

Embora todas as bases de este tipo de programação já se encontrem presentes, tal como se pode constatar no exemplo fornecido. Como sugestão de trabalho futuro deve-se desenvolver corretamente um modelo de cinemática inversa para se adaptar na aplicação já preparada para essa função. Adicionalmente seria interessante adicionar-se um modo de visão artificial ou acrescento de vários sensores de modo a que a aplicação possa absorver mais informação do ambiente que rodeia o robô manipulador.

6.2 Discussão de resultados

Outros testes experimentais revelaram que após uma breve análise à robustez física do manipulador, verificou-se que existe uma considerável folga na própria estrutura dos servomotores *Dynamixel*, esta folga é mais acentuada em pontos mais junto à base porque suportam mais peso. Devido ao facto deste trabalho ter como objectivo apenas a utilização de componentes *Bioloid*, mantendo a estrutura já utilizada em projetos anteriores, esse problema não foi resolvido. Na posteridade essa lacuna poderá ser facilmente colmatada com o fabrico de peças de estrutura mais robustas e desenhadas com a finalidade de construírem um robô manipulador. Várias são as entidades independentes que com o recurso a uma simples impressora 3D, fabricam essas mesmas peças estruturais. Uma outra maneira mais simples de tentar resolver o mensionado problema seria adicionar umas argolas (anilhas) nos locais mais críticos. Caso estas permitam continuar a boa liberdade de movimetos do manipulador seria a forma mais rápida e barata de conceber a robustez pretendida no robô.

Após a montagem completa do manipulador e já devidamente conectado foram feitos alguns testes. Testes estes que revelaram que a velocidade de actuação dos *Dynamixels* é excessiva para a aplicação em causa o que acentua terrivelmente o mau amortecimento do sistem em resposta às instruções dadas pelo controlador. Na Figura 6.6 *(a)* pode-se ter noção de qual é o tipo de amortecimento que os servomotores *Dynamixel AX-12* proporcionam.

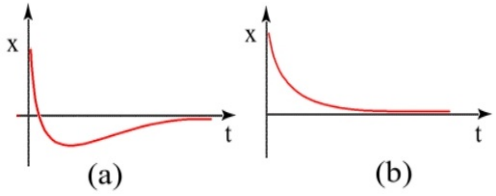


Figura 6.6: Resposta do sistema: (a)Amortecimento crítico; (b) Superamortecido; X - posição; t - tempo

Como este trabalho foi apenas direcionado para o controlo de posição, a resolução mais plausível imposta foi a limitação da velocidade dos servomotores, que fez com que o amortecimento do sistema fosse muito mais aceitável. No entanto a resposta ideal para o amortecimento do sistema seria algo do género ao exposto na Figura 6.6 *(b)*.

Outro resultado, inerente à fase de testes, mostou que a temperatura dos servomotores aumenta muito rapidamente com o uso do manipulador. Este facto deve-se à sobrecarga a que alguns dos *Dynamixels* se encontram sujeitos,

devido à fraca estrutura que suportam. Este problema não pode ser suprimido neste trabalho, mas foi tido em conta, sendo que os valores de temperatura são constantemente monitorizados e o robô é inactivo caso estes valores ultrapassem valores considerados perigosos.

O modelo tridimensional contido na aplicação não é de todo o mais pormenorizado em termos de detalhes gráficos, no entanto, representa o robô manipulador com um grau consideravelmente satisfatório. A inclusão deste modelo tridimensional foi uma grande valia para o funcionamento da aplicação, principalmente porque permite ao utilizador usufruir de um modo de programação desconectado do robô manipulador.

Por fim resta referir que esta aplicação juntamente com o respetivo robô manipulador já foi apresentada ao público, nomeadamente na *Qualific@* que é uma 'Feira de Educação, Formação, Juventude e Emprego' e que teve lugar na *EXPONOR* (Porto). Mais tarde também foi apresentado na *Noite Europeia dos Investigadores* sobre a temática 'Robótica Descodificada' que decorreu no *Centro Ciência Viva* de Bragança. Em ambos os locais foram feitas demonstrações das funcionalidades da aplicação em que cativou especial interesse a interface correspondente ao simulador 3D.

Capítulo 7

Conclusões

O projeto realizado no âmbito deste estágio profissional teve como principal objetivo o desenvolvimento de um robô manipulador constituído por componentes da plataforma didática *Bioloid*, que possibilitasse sustentar uma significativa evolução na formação dos tópicos de robótica a baixo custo. Para este efeito foi construído um robô manipulador com quatro graus de liberdade que utiliza seis servomotores *Dynamixel AX-12* do *Bioloid Comprehensive Robot Kit*. A estrutura do robô manipulador utilizada neste trabalho foi uma estrutura reaproveitada de outro trabalho semelhante, sendo que todo o *software* contolador foi desenvolvido de raíz.

O controlo deste robô manipulador foi implementado através de uma aplicação desenvolvida na plataforma *Lazarus* que utiliza a linguagem de programação *Pascal*.

A aplicação desenvolvida permite para além de controlar o robô manipulador, construir programas que contenham os movimentos adquiridos pelo robô na execução de trajetórias. Estes programas podem ser guardados em ficheiros de texto e podem ser acedidos pela aplicação quando for pretendido. Os ficheiros que armazenam os programas contêm informação relativa à posição dos servomotores e a velocidade da correspondente trajetória a executar, sendo este um modo simples e eficaz de programar por aprendizagem de pontos.

De forma a melhorar a interface com o utilizador, foi desenvolvido um simulador tridimensional do robô manipulador, que permite replicar fielmente os movimentos do robô manipulador e proporciona a vantagem ao utilizador de construir os seus programas sem ter necessariamente o próprio robô conectado à unidade de controlo.

Foram também acrescentadas várias opções adicionais e um modo de programação alternativo que se assemelha à linguagem de programação *RAPID*, mesmo que ainda em desenvolvimento, este modo permite ao utilizador aproximar a sua experiência de programação nesta aplicação à programação de um robô industrial comercial. Este modo de programação permite guardar a informação relativa ao seu programa principal, sendo que o ficheiro em causa irá conter informação relativa à linha em que se encontra a instrução, o tipo de movimento que deve ser efetuado, a posição objectivo e a velocidade da trajetória.

Após a exposição do funcionamento da aplicação ao público ficou claro que a componente do simulador 3D despertou elevado interesse, sendo considerada um sucesso e uma bom tema para ser amplamente aprofundado.

Da experiência adquirida ao longo da realização deste trabalho foram identificados alguns tópicos que podem ser desenvolvidos no futuro de forma a melhorar as funcionalidades da aplicação. Nomeadamente, é possível referir as seguintes:

- Implementar algoritmos para o sistema de controlo que permitam o controlo de trajectórias e respetivas velocidades.
- Adicionar módulos de sensores ou utilizar *Dynamixels* que já os possuam embutidos, para um melhor controlo do espaço envolvente do robô manipulador.
- Adicionar no ambiente do modelo 3D objectos interactivos e que respeitem as leis da física e as propriedades dos materiais; deve ser ainda possível a interacção do robô através do modelo 3D.
- Desenvolver mais aprofundadamente o menu *Help* que permite a ajuda ao utilizador e acrescentar pacotes de línguas.

Por fim, um ponto crucial no desenvolvimento futuro desta aplicação diz respeito à necessidade de solidificar a robustez do manipulador, deve-se por isso projectar e construir uma nova estrutura ou apenas construir peças de apoio à estrutura existente. Esta tarefa pode ser simplificada com recurso às modernas e acessíveis impressoras 3D.

Bibliografia

[1] Anildio Barbosa. Desenvolvimento de robô manipulador didáctico baseado em componentes bioloid. Master's thesis, IPB, 2010.

[2] Antonio Barrientos. *Fundamentos de Robótica*. McGraw-Hill, 1997.

[3] CrustCrawler. Bioloid comprehensive kit, October 2012 2011. URL `http://www.crustcrawler.com/products/bioloid/index.php`.

[4] Eric Grange. Glscene open solution for delphi, October 2012 2011. URL `http://glscene.sourceforge.net/wikka/HomePage`.

[5] Paulo Leitão. *Notas de Apoio a Sistemas de Automação - Tecnologias de Sistemas de Automação Industrial*. 2010.

[6] Free Pascal. Glscene, October 2012 2012. URL `http://wiki.freepascal.org/GLScene`.

[7] Trossen Robotics. Bioloid comprehensive robot kit, October 2012 . URL `http://www.trossenrobotics.com/bioloid-comprehensive-robot-kit.aspx`.

[8] ROBOTIS. *Bioloid QuickStart - Comprehensive Kit Robot Series*. .

[9] ROBOTIS. *USB2Dynamixel - User's Manual*. .

[10] ROBOTIS. *Dynamixel AX-12*. 2006.

[11] Vítor M. F. Santos. *Robótica Industrial*. Universidade de Aveiro, 2003.

Anexos

Apéndice A

Kit Bioloid

O *Bioloid Comprehensive Robot Kit* da *Robotis* é um *kit* de robótico de caracter educacional baseado em torno de "controlo inteligente de servomotores em série", que não só são capazes de alternar entre posicionamento e rotação contínua completa, mas também fornece acesso ao feedback sensorial, como velocidade, posição, binário, temperatura, corrente e tensão de cada servo. Os *kits Bioloid* são concebidos de forma a que permitam ao utilizador construir vários tipos de robôs. Além da qualidade do hardware, o *Kit Bioloid* vem com algum *software* baseado em *GUI* bastante poderoso. O *software* é gratuito, por isso no caso de o utilizador usar este *kit* para fins educacionais, para pesquisa, ou mera recreação pessoal, é possível descarregar os programas e instalar estes mesmos em todos computadores que necessite. Os *kits* de robôs *Bioloid* são perfeitos para a Educação, passatempos, pesquisa e competições.

Na Figura A.1 encontra-se o *Bioloid Comprehensiv Kit* que foi utilizado para a execução deste projeto. Este *kit* é comercializado por vários fornecedores de equipamento robótico, electrónico, lúdico ou educacional, podendo ser adquirido directamente nos seus postos de venda ou efectuando uma encomenda on-line nos sítios dos estabelecimentos de comércio ou no próprio sítio oficial da *ROBOTIS*.

Figura A.1: *Bioloid Comprehensive Kit*

O fornecedor deste *kit* (*ROBOTIS*), providencia as instruções para a montagem de vários tipos de robôs diferentes, os mais populares exemplos desses robôs são os ilustrados na Figura A.2. De modo a facilitar a programação destes robôs,

são incluídos modelos 3D destas mesmas montagens no *software* fornecido. Existe uma vasta comunidade de entidades independentes que partilham livremente aplicações desenvolvidas nos programas fornecidos e em *software* desenvolvido pelos próprios utilizadores.

Figura A.2: Exemplos opções de montagem do *kit Bioloid* fornecidas pelo fabricante

Tal como se pode comprovar analisando novamente a Figura A.2, todos os robôs possuem embutido um elemento que lhes é comum, que é a consola de programação *CM-5*. Esta unidade central de processamento armazena o programa pré-elaborado pelo utilizador e com recurso a baterias nele contidas torna este módulo independente, permitindo executar o mencionado programa a qualquer altura. Esta consola é comparada em vários aspectos a um autómato industrial.

Na Figura A.3 pode-se ver o modo de como é estabelecida a comunicação, com a finalidade de programação do módulo *CM-5*, entre o PC e a consola principal. Esta consola é então programada através de um cabo série que liga ao computador e um Jack 3.5 na outra extremidade do cabo, que liga no módulo *CM-5*. [3]

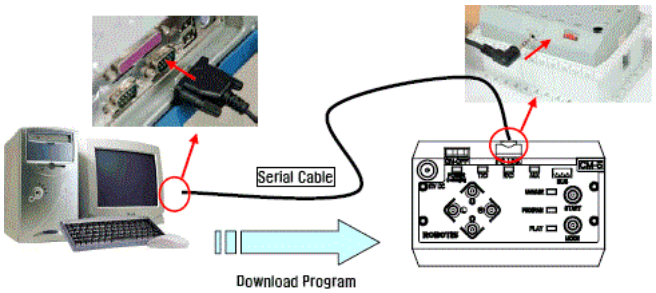


Figura A.3: Modo de estabelecer comunicação entre PC e a Consola Principal (*CM-5*)

A *ROBOTIS* fornece aos clientes destes *kits* duas aplicações de extrema utilidade, essas aplicações são *Motion Editor* e *Behavior Control Programmer*, e é possível ver o seu aspecto pela observação da Figura A.4. A aplicação *Motion Editor* oferece um modo de programação através de uma interface gráfica dos robôs contidos nos exemplos do próprio programa. Esta é uma interface gráfica que permite criar e guardar posições e movimentos de vários módulos Dynamixel em simultâneo, para serem usadas posteriormente pelo *Behavior Control*

Program. O utilizador pode criar comportamentos movendo as diferentes juntas manualmente e guardar essas posições. É normalmente utilizado quando existe a necessidade de executar movimentos complexos. [7]

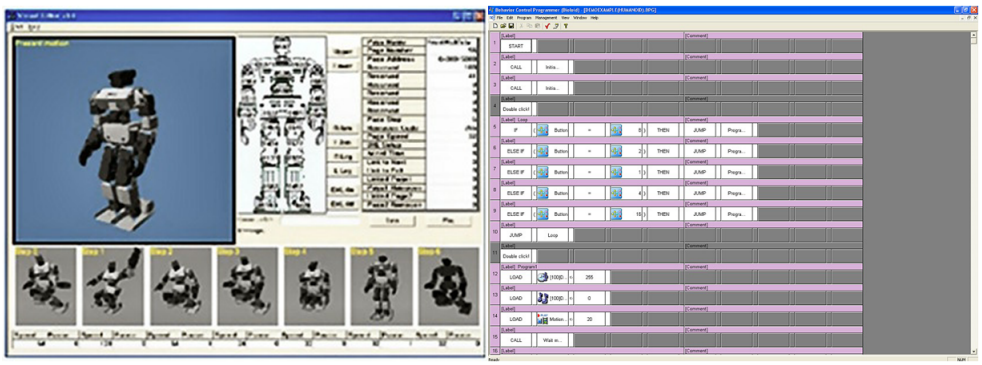


Figura A.4: *Motion Editor* (à esquerda); *Behavior Control Programmer* (à direita)

O *Behavior Control Program* é um programa que permite criar uma série de regras de comportamento e ações que o robô deve adoptar frente às circunstâncias propostas. Este é o *software* principal para criar os programas para os robôs *Bioloid*, sendo que consiste em múltiplas linhas nas quais são inseridos os blocos de instruções. Para programar neste ambiente é necessário conhecer a lista de instruções válidas.

Hoje em dia existe uma constante atualização no fabrico de novos *kits Bioloid*. A versão mais actualizada é o *Bioloid Premium Kit*, que está retratada na Figura A.5, esta é a montagem de um robô humanóide que utiliza a melhor tenologia que este novo *kit* fornece, tendo ao seu dispor um avançado leque de novos sensores.

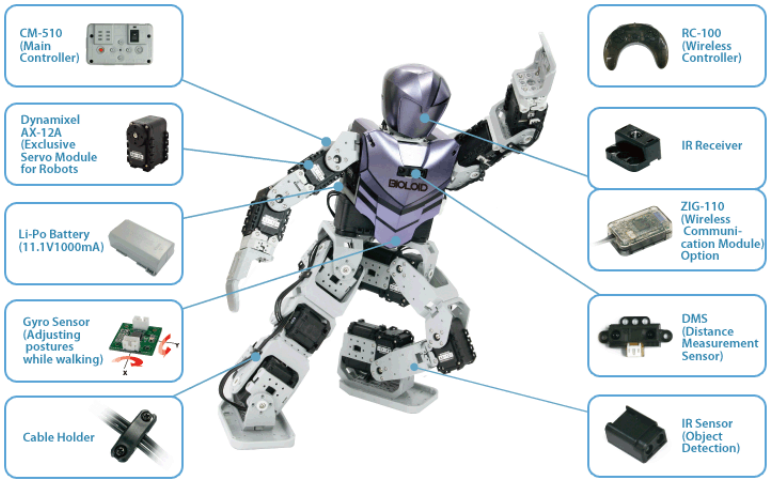


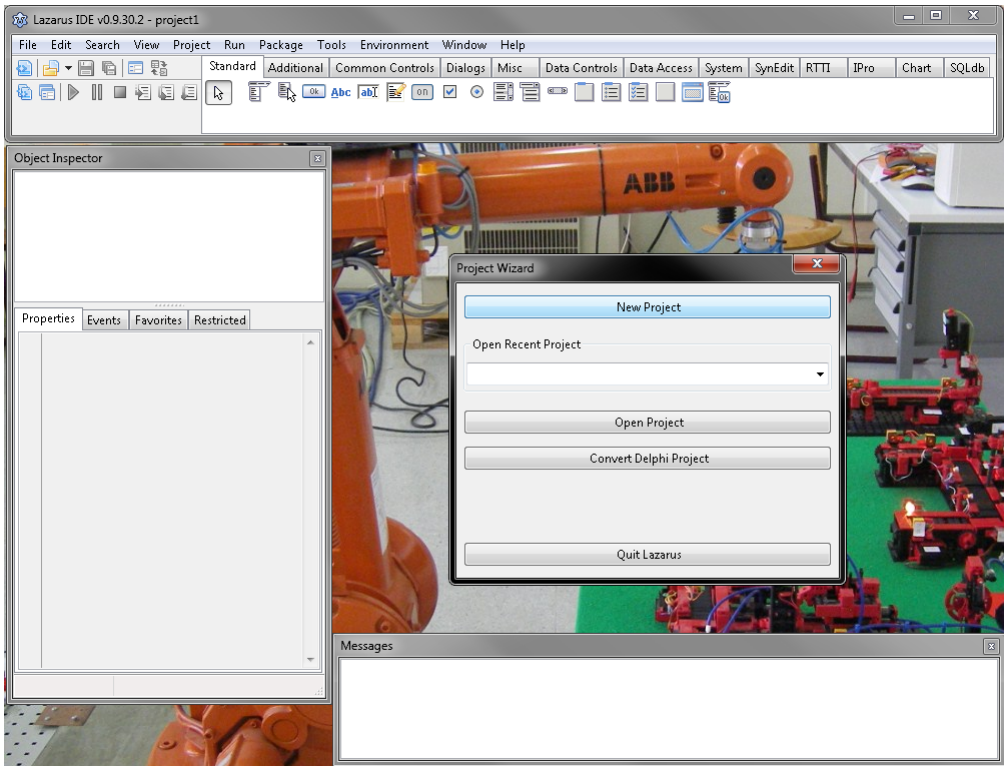
Figura A.5: *Bioloid Premium Kit*

O facto de estes *kits* serem de fácil programação e de preços razoáveis, permitiu à *ROBOTIS* difundir exemplarmente os seus produtos. Esta expansão leva a que novos produtos sejam constantemente submetidos a novos desafios e testes, aproveitando em paralelo o desenvolvimento de novas tecnologias, dando posterior origem a componentes de melhor qualidade, mais avançados e com mais características de parametrização.

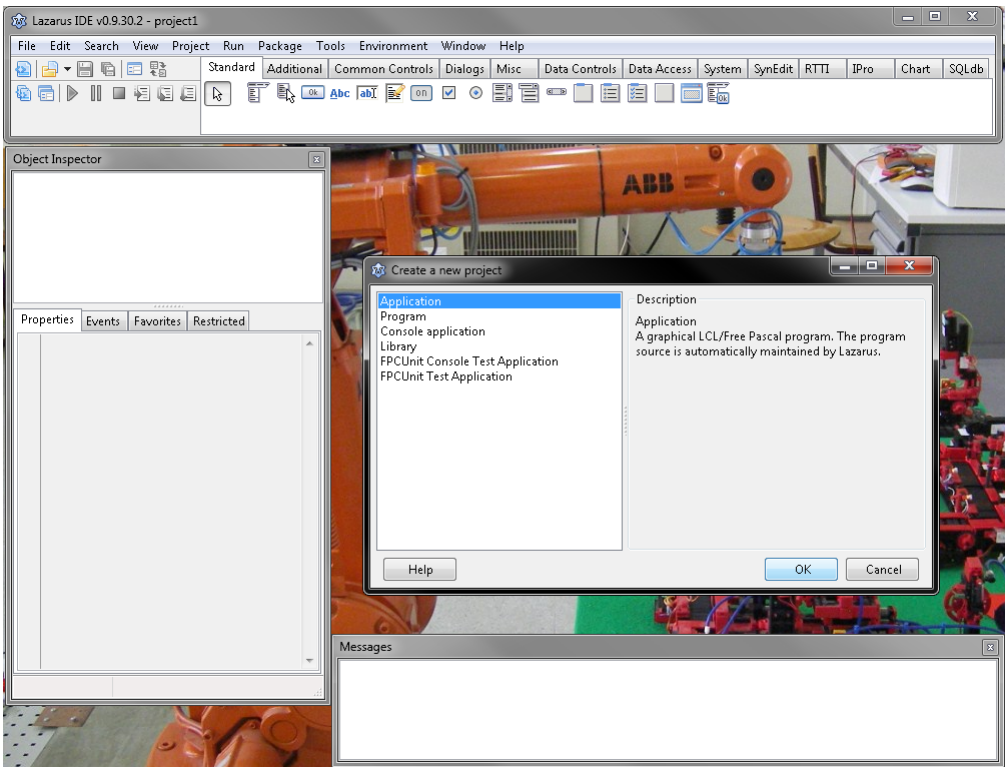
Apêndice B

Ambiente *Lazarus*

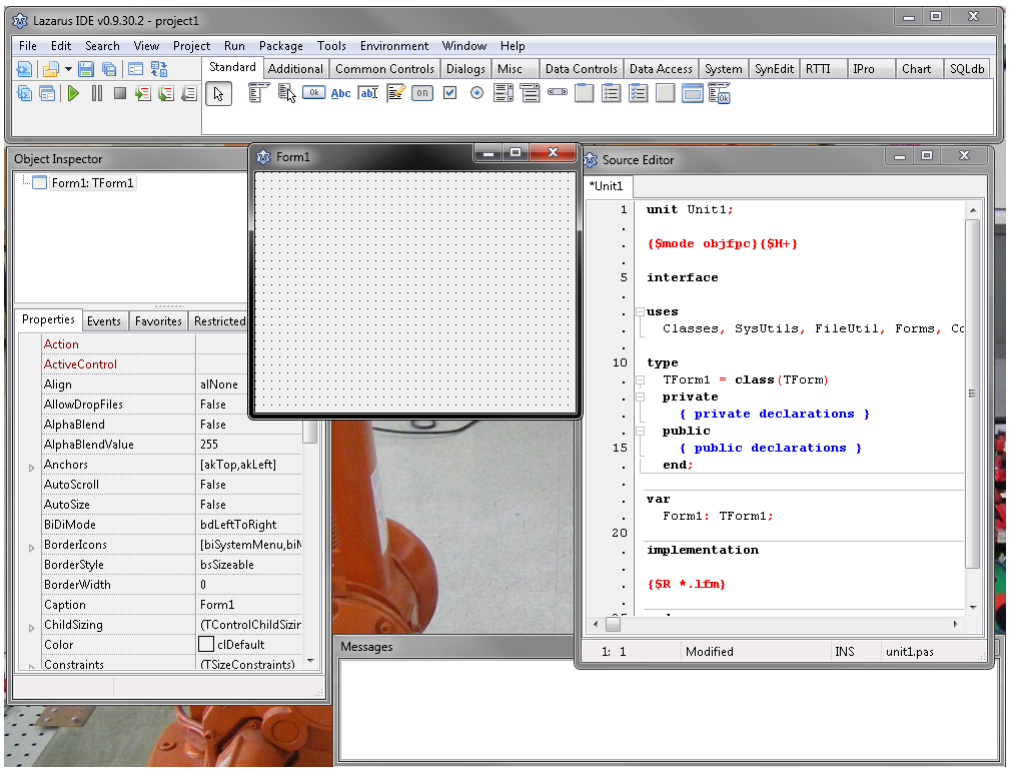
B.1 Criar uma nova aplicação



Project Wizard - Criar um novo projeto (*New Project*)



Selecionar o tipo de projeto a criar



Ambiente *Lazarus* de uma nova aplicação

B.2 Descrição do ambiente de trabalho

Lazarus é uma multi-plataforma livre *IDE* (*Integrated Development Environment* - Ambiente de Desenvolvimento Integrado), que proporciona uma experiência de desenvolvimento *Delphi* para programadores em *Pascal*. Isto permite criar programas de natureza gráfica (*GUI- Graphical User Interface* - interface gráfica para o utilizador). É desenvolvido para, e suportado por o compilador *Free Pascal* para gerar um ficheiro executável. Desde o início de 2008, *Lazarus* está disponível para *Microsoft Windows*, várias distribuições *Linux*, *FreeBSD*, e *Mac OS X*.

Esta poderosa ferramenta foi o programa utilizado para desenvolver a aplicação *Educational Manipulator Studio V1.0*. Embora a referida aplicação desenvolvida tem sido evoluída de uma aplicação já existente (*ESTiG IPB Dynamixel Servos Controller V0.0*).

Conforme o exibido na Figura B.1 assim que se cria uma nova aplicação, surgem no ambiente de trabalho cinco janelas distintas. A janela principal é a que se encontra no topo superior da imagem. É aqui onde se encontram todos os elementos que foram usados no *Educational Manipulator Studio*. Esses elementos são de vários tipos, podendo ser elementos visíveis como por exemplo: botões, caixas de texto, tabelas, barras de progresso, etc. Ou elementos de funcionamento da aplicação, como por exemplo: relógios, diálogos de abrir e guardar ficheiros, organizadores de barras de menus, entre outros.

Todos os elementos (objectos) retirados da janela principal têm de ser colocados, conforme o pretendido, na janela intutulada de *Form* (*Form1* para o caso da Figura B.1). Uma vez aqui colocados estes elementos podem ser dispostos livremente e é também possível aceder ao seu conteúdo e funcionalidades. Para isso clica-se no elemento que se pretende editar e aparecem todas as suas opções na janela *Object Inspector*. É exactamente nesta janela que se atribui grande parte das características dos objectos. De modo a complementar o bom funcionamento da aplicação temos a janela *Source Editor*, que produz automaticamente algumas partes de código (em *Pascal*), quando é explorado o *Object Inspector*. As linhas de código que podem ser geradas automaticamente são apenas a construção da estrutura de procedimentos, cabe portanto ao utilizador preencher esses procedimentos com os comandos desejados. Por fim existe a janela *Messages*, que tal como o próprio nome indica apenas exibe mensagens. Estas mensagens têm especial importância a quando da compilação das aplicações, isto porque são identificados possíveis erros e sua a sua causa, isto torna mais fácil o acesso a esses erros e seguidamente facilita a sua correcção.

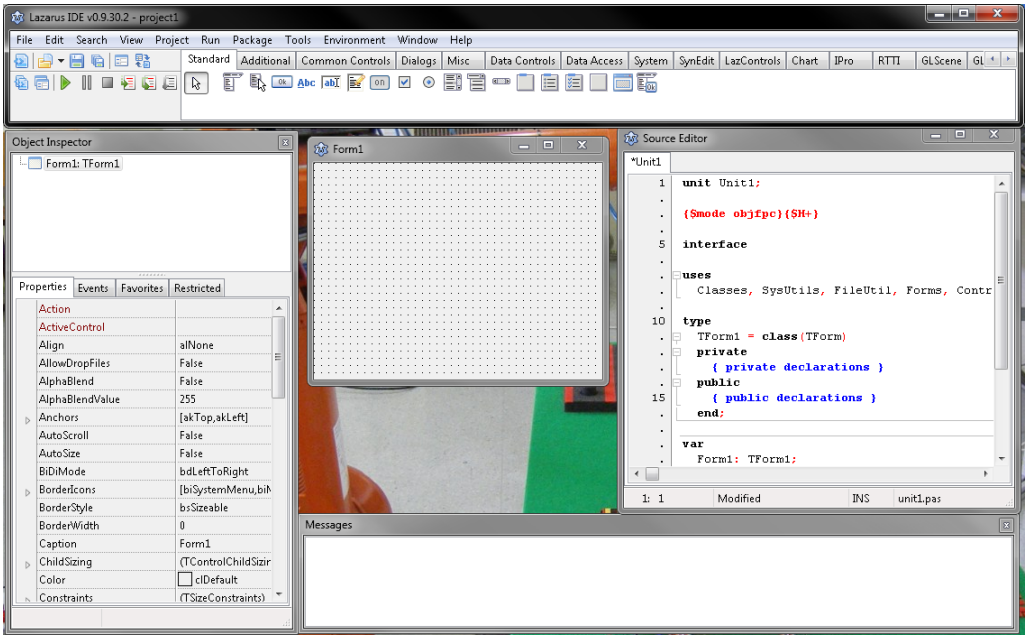
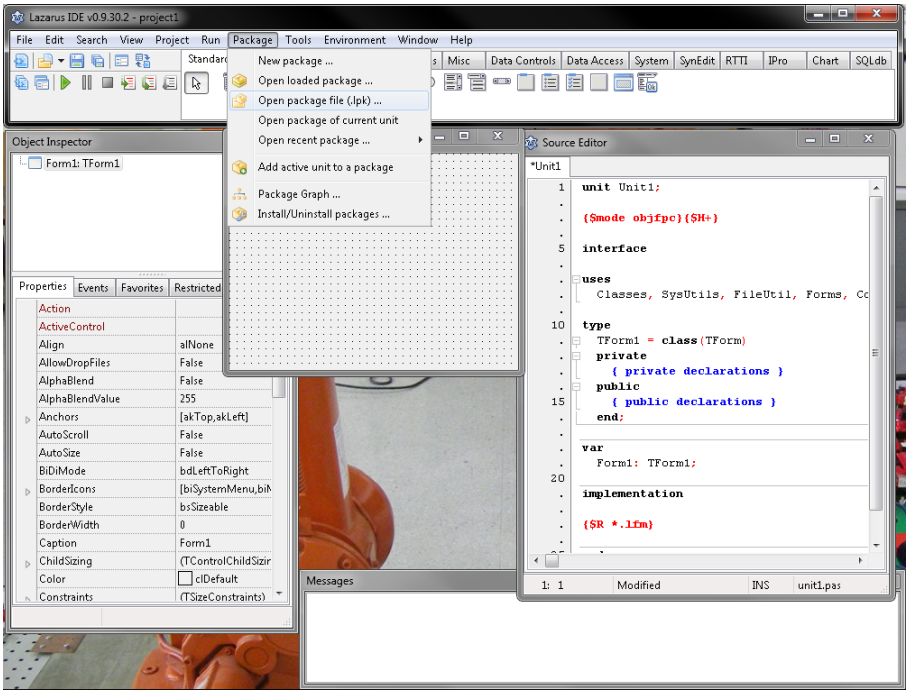


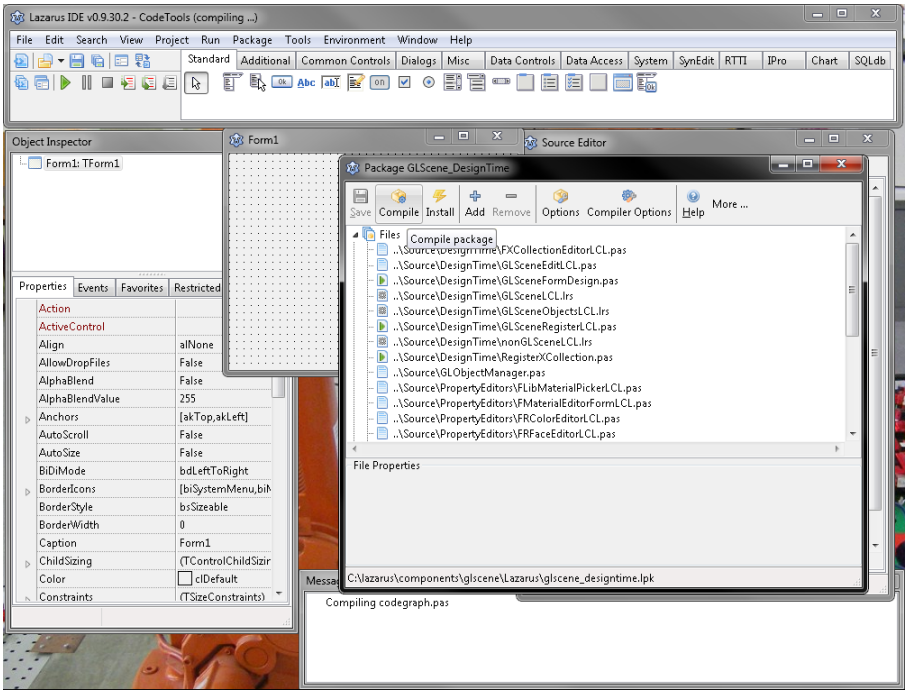
Figura B.1: Ambiente *Lazarus* de uma nova aplicação

Apêndice C

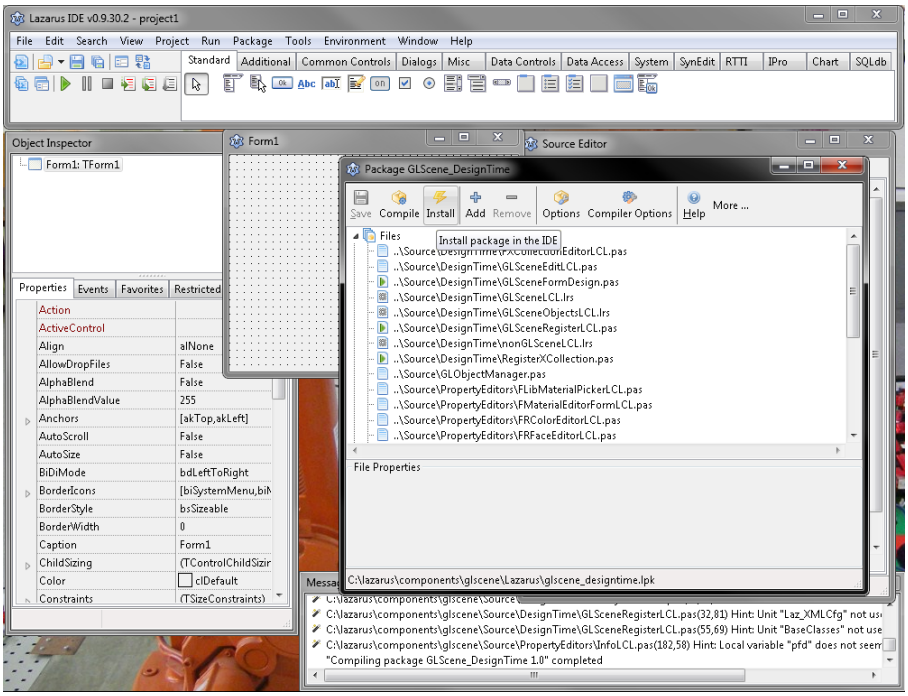
Instalação da componente de *software GLScene*



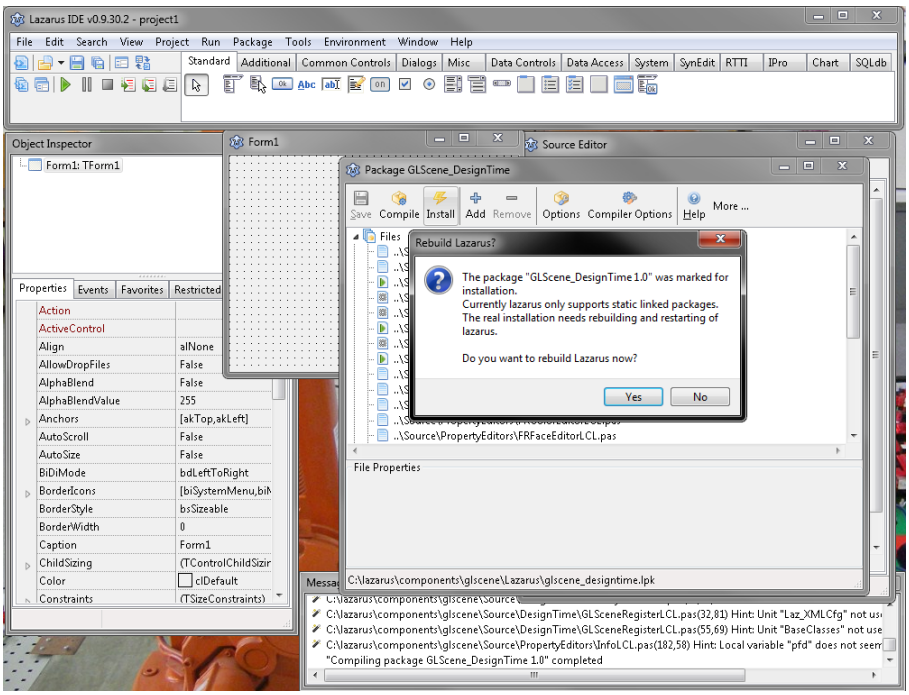
Package -> Open packege file (.lpk)...



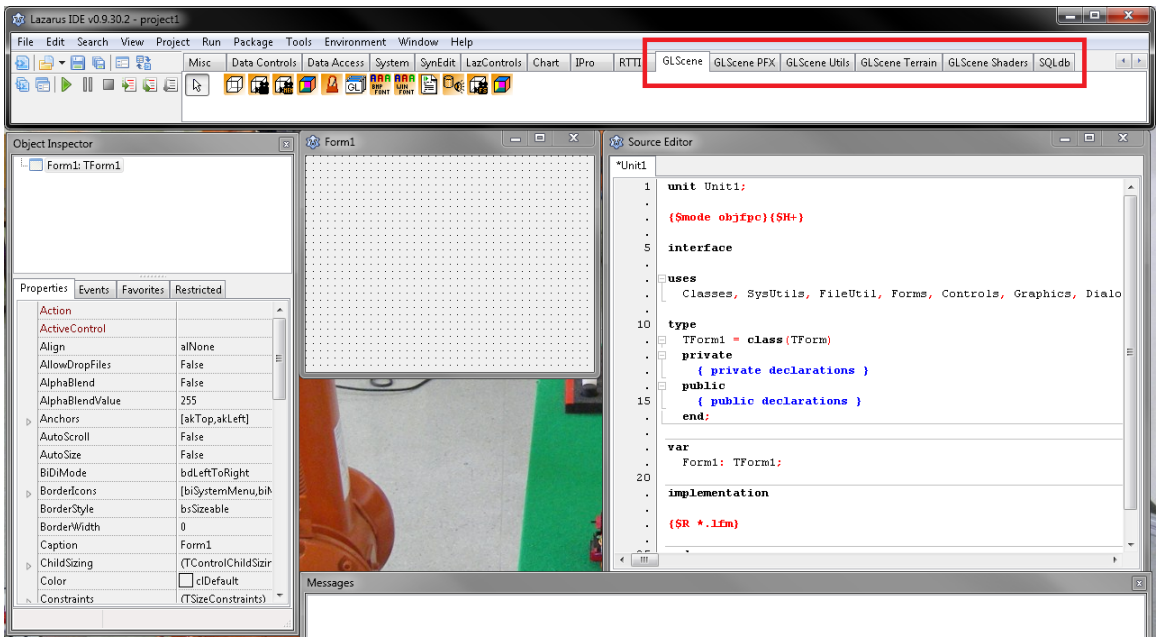
Compile package -> Compile



Install package in the IDE -> Install



Rebuild Lazarus? -> Yes



Novos separadores *GLScene*