IPG Politécnico da Guarda
Polytechnic of Guarda

RELATÓRIO DE PROJETO

Licenciatura em Engenharia Informática

Pedro Figueiredo Meneses Rocha

Dezembro | 2013

Escola Superior de Tecnologia e Gestão

Instituto Politécnico da Guarda

Robonova@IPG 2.0 **- Robô Humanoide Autónomo Interativo**

Projeto de Informática do curso de Engenharia Informática

Orientador: Professor Carlos Carreto

Pedro Figueiredo Meneses Rocha

Dezembro 2013

Agradecimentos

Em primeiro lugar queria agradecer ao Professor Carlos Carreto, por ter aceite ser o meu orientador para este projecto. Em seguida agradecer ao Instituto Politécnico da Guarda por ter disponibilizado todo o equipamento necessário para o desenvolvimento deste projecto.

Num contexto mais pessoal gostaria de agradecer a minha família e amigos por proporcionarem algum apoio para o desenvolvimento deste projecto e para uma última fase académica da minha aprendizagem.

Resumo

Este relatório descreve o trabalho desenvolvido no âmbito de um projecto de informática que consistiu na programação de um robô humanóide interactivo, capaz de detetar e interagir com pessoas, tirando-lhes fotografias para posteriormente as colocar numa página do Facebook.

Este projecto vem dar continuidade ao projecto Robonova@IPG que já se tinha iniciado anteriormente [1].

O trabalho foi desenvolvido nas instalações do Instituto Politécnico da Guarda, no âmbito da disciplina de Projeto de Informática, do curso de Engenharia Informática do Instituto Politécnico da Guarda.

Palavras chave: Robô Humanoide, Robonova, Facebook, Interação

Abstract

This report describes the work developed under a project that consisted in programming an interactive humanoid robot, able to detect and interact with people, by taking them pictures that are published in a Facebook page.

This project continues the project Robonova@IPG already started earlier [1].

The work was conducted at the Polytechnic Institute of Guarda, as a project of the Informatics Engeering course.

Keywords: Humanoid Robot, Robonova, Facebook, Interaction

Índice

Índice de Figuras

Índice de códigos

1. Introdução

Este projeto vem dar continuidade ao projeto Robonova@IPG que já se tinha iniciado anteriormente. Este robô será capaz de detetar e interagir com pessoas, tirando-lhes fotografias para posteriormente as colocar numa página do Facebook.

Neste foram acrescentadas algumas ideias novas para assim poder melhorar o projecto anterior. Um dos novos comportamentos foi a criação de uma sequência de movimentos para o robô. Estes movimentos são executados autonomamente para chamar a atenção de alguém que esteja por perto mas só são executados enquanto a câmara do robô não detetar alguma pessoa por perto. Caso contrário, este detetar uma pessoa, o robô irá executar uma outra funcionalidade anteriormente referida, o movimento do robô consoante a posição da pessoa. Se esta estiver muito longe o robô irá aproximar-se, se isto não acontecer o robô irá distanciar-se. Visto isto falta mencionar que o robô executa este movimento se a pessoa estiver na área de captação da câmara do robô. Esta área foi implementada com objetivo de conseguir que a pessoa fique de frente para o robô. Após isto falto salientar, se a pessoa for detetada e não estiver na área de captação, é iniciado um comportamento de comunicação entre o robô e a mesma. Neste caso o robô envia um comando de voz à pessoa, dizendo para esta se movimentar para a esquerda ou para a direita, para assim se conseguir que esta esteja dentro na área de captação da câmara.

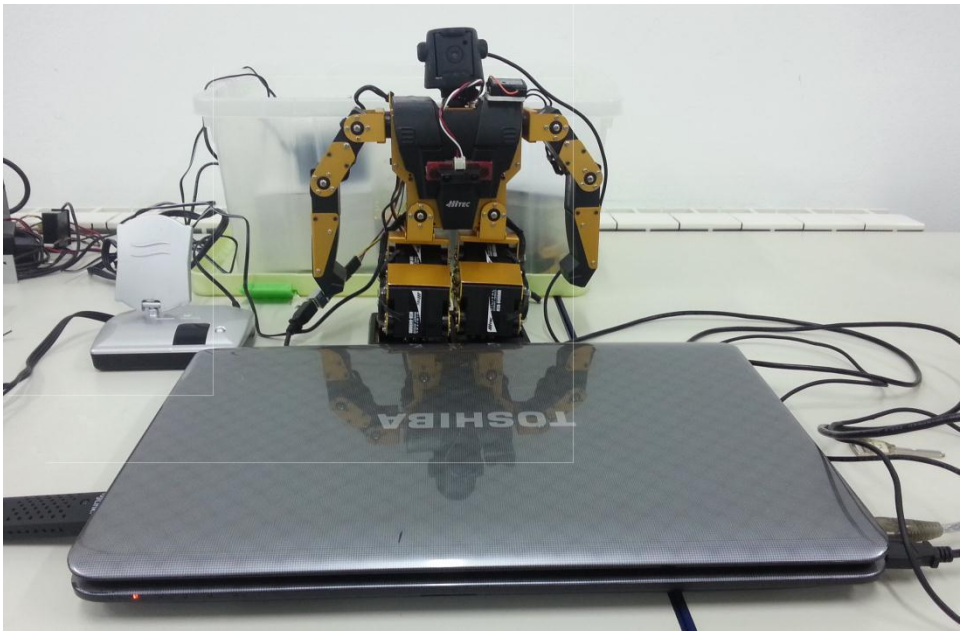


Figura 1 - Sistema robô + computador.

A forma encontrada para resolver esta situação, foi através da elaboração de uma aplicação desktop em C#, que tem como objetivo controlar a deteção de faces, carregamento de imagens para uma rede social e o controlo autónomo do robô. Para isto foram usadas algumas bibliotecas já existentes nesta área, estas serão descritas na secção 3.

A metodologia usada para desenvolver esta aplicação consistiu exclusivamente do Desenvolvimento Ágil. Este tem como objetivo que se avalie o desenvolvimento do software com alguma frequência, para proporcionar um bom funcionamento do mesmo.

1.1. Motivação

Por um lado existiu a motivação do desafio técnico que tem como objetivo desenvolver um robô humanóide interativo de baixo custo, capaz de detetar e interagir com pessoas, tirando-lhes fotografias para posteriormente as colocar no Facebook. Este é um projeto muito motivador devido ao fato de ter várias componentes informáticas que permitiram aplicar na prática as diferentes competências adquiridas ao longo do curso.

Ainda assim existiu a motivação pessoal, o interesse em trabalhar com robôs humanóides e com outras tecnologias implementadas no projeto.

1.2. Objetivos

Como referido anteriormente, o projeto realizado consistiu na continuação do desenvolvimento do projeto Robonova@IPG [1], de modo a incorporar novas funcionalidades e melhorar o funcionamento do mesmo. Para tal foram definidos os seguintes objetivos:

- Organizar o software de controlo já existente, numa arquitetura baseada em comportamentos, implementada com base em máquinas de estados, para que seja mais simples e flexível o desenvolvimento de software de controlo e as funcionalidades de interação do robô.
- Desenvolver e integrar no software de controlo do robô, novos comportamentos interativos, nomeadamente um comportamento de espera no qual o robô executa movimentos e falas pré-programados para chamar a atenção das pessoas, até que detecte as mesmas com os seus sensores e um comportamento para o robô recuperar de quedas que possam ocorrer.
- Melhorar o funcionamento geral do sistema, otimização sempre que possível, os componentes de software e de hardware.

1.3. Estrutura do documento

Para além do primeiro capítulo de introdução, o relatório está organizado da seguinte forma.No segundo capítulo são mencionados os robôs humanóides interativos, descrevemos alguns kits de robôs humanóides existentes no mercado. No terceiro capítulo é descrito o desenvolvimento do robô humanóide interactivo. No quarto capítulo, é onde se fala sobre a conclusão do projeto. Finalmente, no último capítulo, o quinto, são mostradas alguns conceitos que poderão ser implementados para um trabalho futuro.

1.4. Descrição das tarefas

As principais tarefas de desenvolvimento da aplicação foram:

Tarefa 1 – Análise dos requisitos.

Tarefa 2 – Pesquisa e obtenção da documentação.

Tarefa 3 – Implementação do relatório.

Tarefa 4 – Elaboração do relatório

Figura 2 - Mapa de Gantt das tarefas realizadas.

2. Robôs Humanoides Interativos

2.1. Robô Bioloid

Nos dias de hoje existe uma enorme diversidade de robôs interativos humanoides, nesta secção irei descrever alguns deles. Na figura seguinte é apresentado o robô Bioloid.

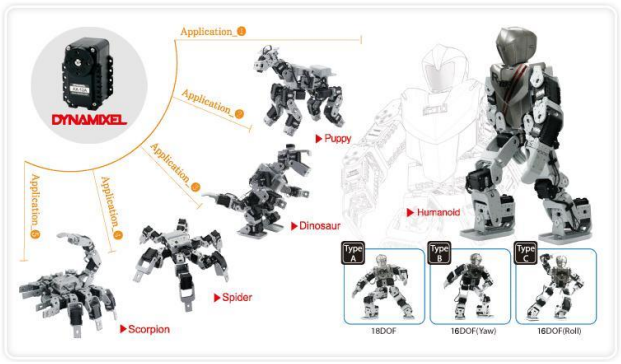


Figura 3 - Vários tipos de montagem do robô Bioloid [2].

Este robô pode ser montado em diferentes formas, podendo ser assim um robô em forma de aranha, de um escorpião, um dinossauro ou até mesmo um cão. São vários tipos de animais que este robô é capaz de representar. Na seguinte figura serão apresentados os vários sensores deste robô.

Figura 4 - Sensores do robô Bioloid (adaptado de [2]).

2.2. Robô Nao

Este robô tem incorporadas várias características, como por exemplo: falas em 8 linguas diferentes, reconhecimento facial, localização de sons, deteção de queda, entre outros. Na seguinte imagem estão representados os sensores deste robô.

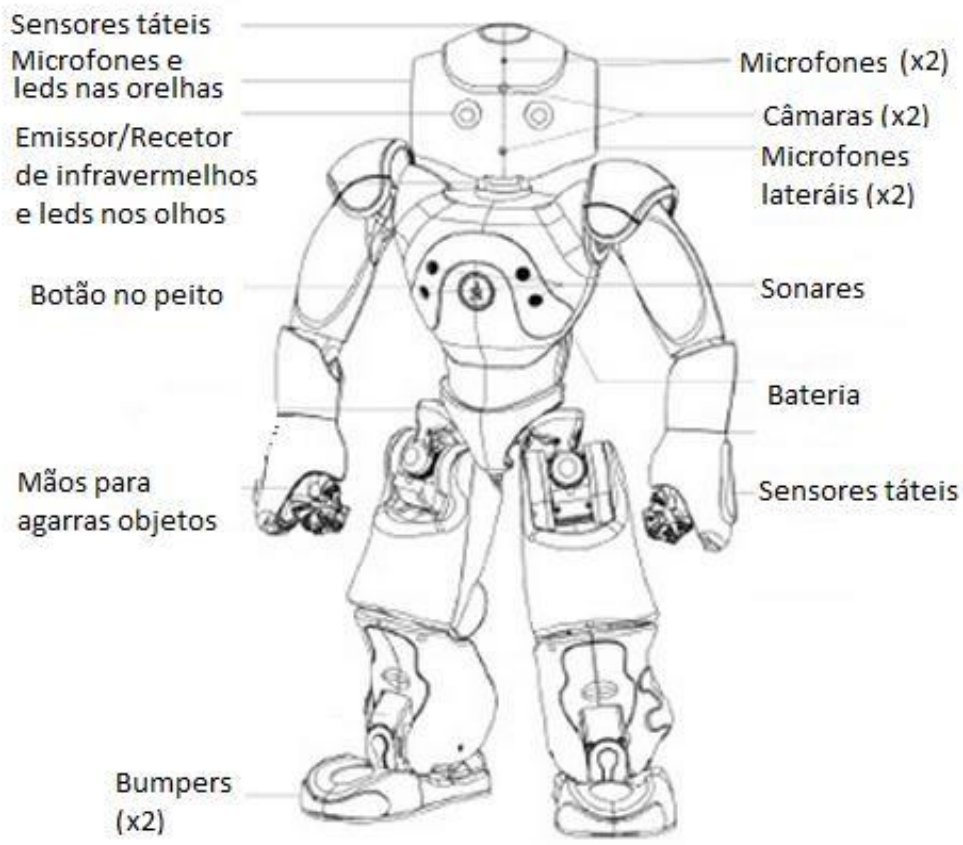


Figura 5 - Sensores do roobô Nao (adaptado de [3]).

2.3. Robô Robonova

O robô Robonova utilizado no projeto. Este robô inicialmente não tinha nenhum sensor incorporado, posteriormente foram colocados segundo a necessidade apresentada. A explicação destes sensores inicia-se na seção 3, lá estão descritos pormenorizadamente cada um deles, e a sua função para este projeto.

A ferramenta que foi utilizada para o desenvolvimento deste projeto foi o RoboBASIC v2.5, da parte de movimentos implementados pelo robô. Para poder elaborar esta parte de código e fazer as ligações corretamente dos sensores ao robô foi necessário recorrer a alguns manuais para facilitar a programação [4] e [5]. Para além disto também existiu a programação em C#, que trata do reconhecimento facial, e do envio de comandos para robô executar. O código completo de cada programa encontra-se na seção anexos. Através das ferramentas que vão ser apresentadas consegue-se elaborar uma criação de movimentos para o robô poder executar Na seguinte imagem encontra-se uma das ferramentas descritas anteriormente.

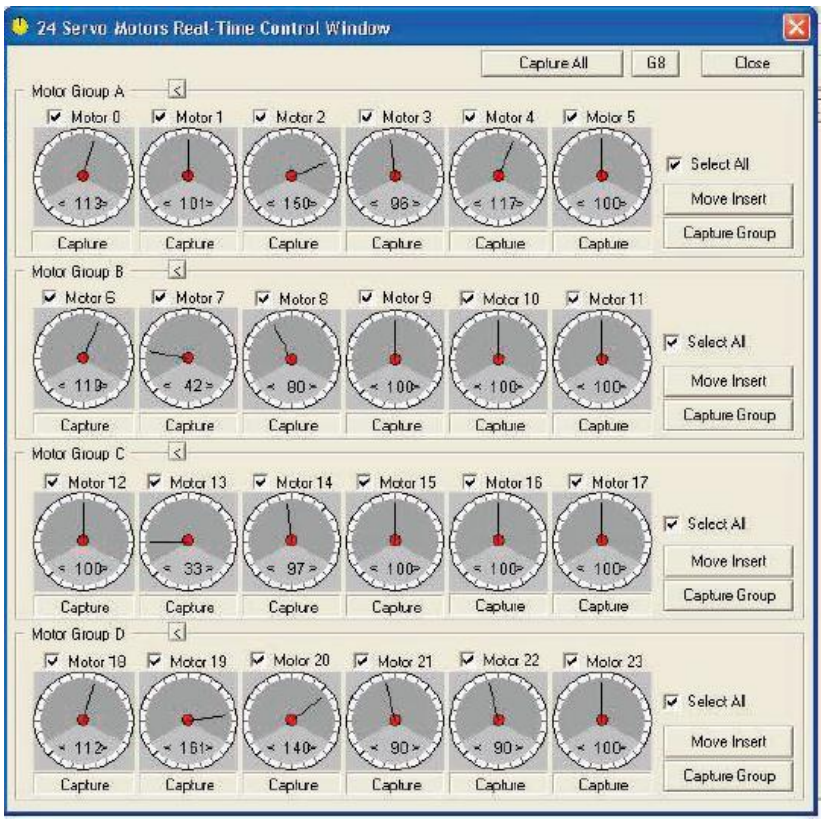


Figura 6 - Ferramenta de controlo dos servomotores

Esta ferramenta permite alterar a rotação de cada motor, após isto poderá colocar-se a alteração no código. Isto é feito através do botão "Move Insert", como está demonstrado na imagem anterior. A próxima ferramenta é um pouco mais intuitiva, basta mover as bolas vermelhas presentes em cada barra para escolher o novo ângulo do motor. O botão "Insert Move" serve para colocar o ângulo escolhido no código do projeto.

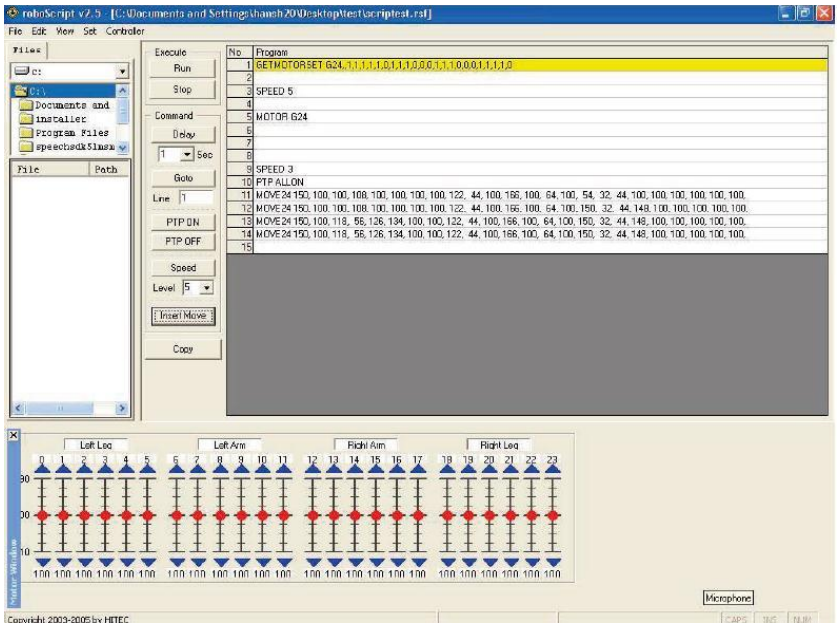


Figura 7 - Ferramenta Roboscript v2.5.

Na figura 8 está apresentado o robô Robonova usado para este projeto.

Figura 8 - Robô Hitec Robonova-I.

O robô Robonova possui 16 servomotores, 3 para cada membro superior e 5 para cada membro inferior. São servomotores HSR-8498 e são produzidos pela empresa Hitec. Na figura 9 podemos observar o exemplo de um dos motores presentes no robô.

Figura 9 - Servomotor HSR-8498.

Especificações do servomotor:

- Interface: Protocolo HMI
- Voltagem de operação: 4,8V até 6,0V
- Angulo de operação: 180º
- Peso: 55g
- Tamanho: 40 x 20 x 37 mm
- Torque (6.0V) : 10kg/cm
- Velocidade (6.0V) : 0,20 seg/60º

Foi utilizada uma placa MR-C3024, é aqui onde existe guardada toda a programação feita na linguagem roboBasic. Esta placa tem uma variedade de portas de entrada e saída, desde 24 portas de servomotores e 16 portas de entrada para sensores, como por exemplo, o giroscópio ou o sensor de queda.

Figura 10 - Placa de controlo MR-C3024.

Especificações da placa:

- CPU: Atmel ATMEGA128 8bit RISC
- Portas IO: 24
- Portas servo: 24
- Portas analógicas: 8
- Memória para programas: 32KBytes
- Controlo de LCD
- Comunicação em série RS-232

Nesta tabela vão ser comparados os kits de cada um dos robôs referidos anteriormente.

	Bioloid	Robonova	Nao	Robonova(projecto)
Várias formas de montagem	X	-	-	-
Comunicação wireless	X	-	X	-
Sensor giroscópio	X	-	-	X
Detetar queda	-	-	X	X
Sensor de distância	X	-	X	X
Interação com pessoa	X	-	X	X

Legenda: X (Existe no robô) - (Não existe no robô)

Figura 11 - Tabela de comparação de kits de Robôs.

No geral os robôs não vêm com sensores incorporados, tornando assim a interacção com a pessoa inexistente. Como um dos grandes objectivos deste projeto é criar interacção com a pessoa, foi necessário implementar alguns sensores para tornar isso possível.

3. Desenvolvimento do robô humanoide interativo

Para este projeto foi implementada uma máquina de estados, para assim se obter um melhor controlo do robô. Para além disto esta arquitetura permite uma maior simplicidade e flexibilidade na adição ou alteração de comportamentos do robô.

A máquina de estados implementada nos programas do robô e do computador, são formadas por vários comportamentos, estes são alterados consoante o valor que é enviado da porta série do computador para o robô.

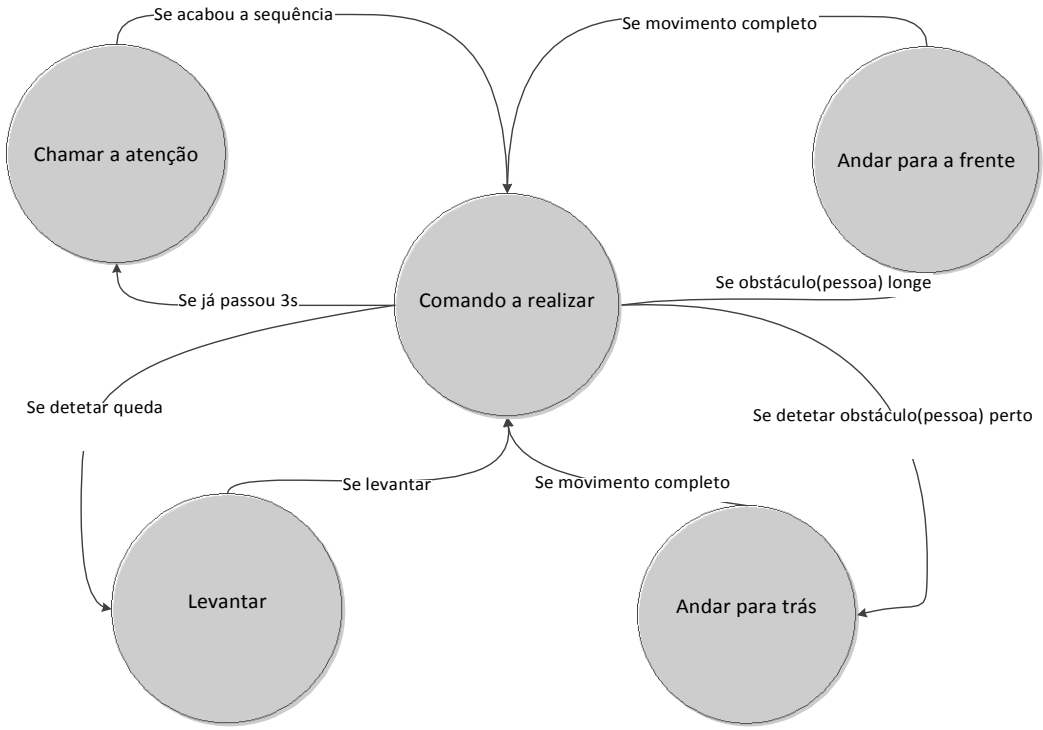


Figura 12 - Máquina de estados do robô.

O estado principal desta máquina de estados será o "Comando a realizar", este é o estado onde o robô recebe dados do computador e verifica se existe algum comportamento a executar. Caso contrário, o robô não detete alguma face em 3segundos, este irá mudar de estado para o "Chamar a atenção". Neste é feito uma sequência de movimentos para assim conseguir a atenção de uma pessoa. No estado "Andar para a frente" o robô irá dirigir-se à pessoa caso esta esteja um pouco longe do robô, se acontecer o contrário o robô terá um novo estado, "Andar para trás", afastando-se da pessoa. Em qualquer um dos estados anteriormente referidos, se existir uma queda este mudará para o estado "Levantar", onde essa situação é corrigida.

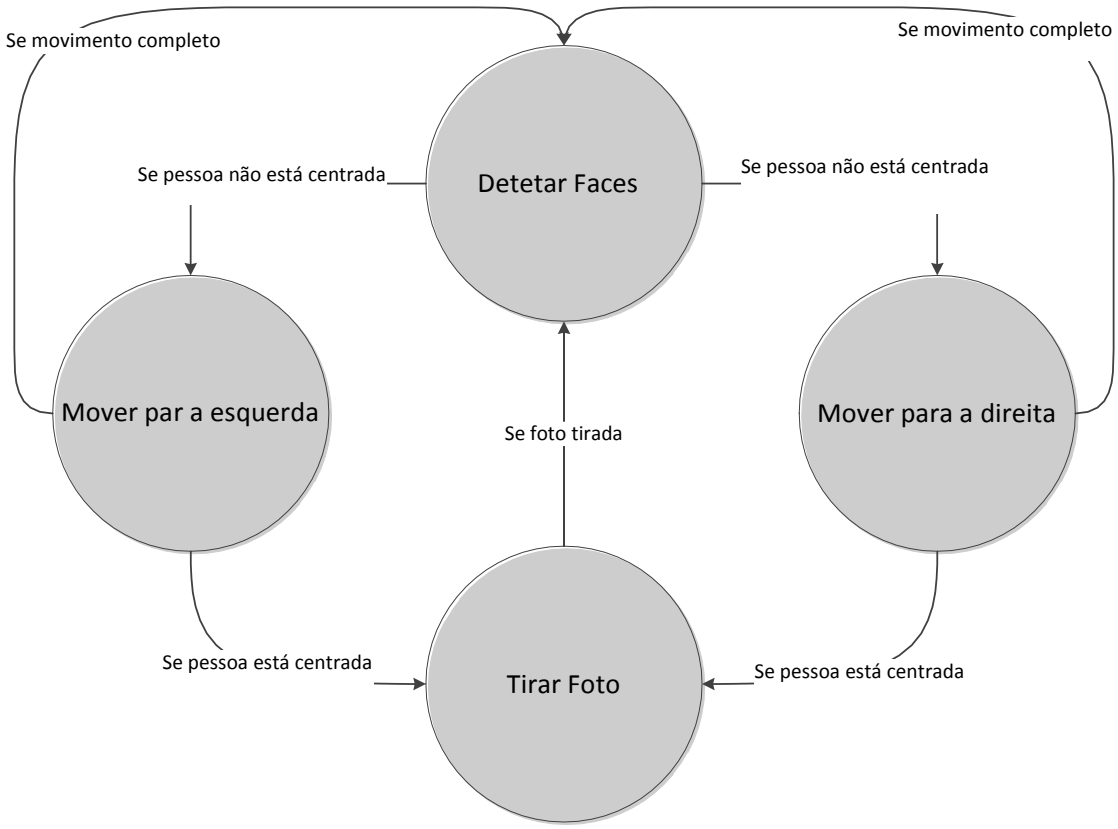


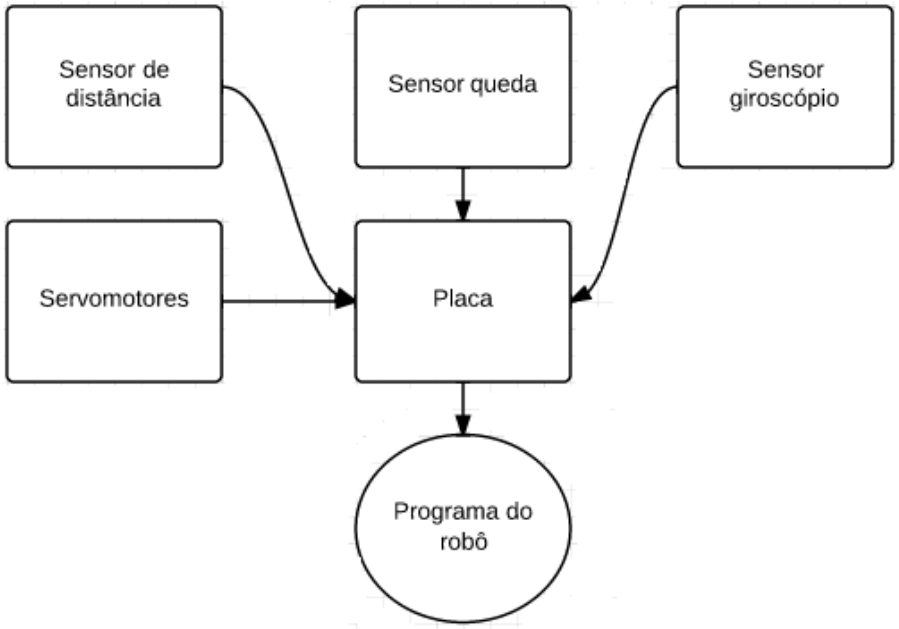
Figura 13 - Máquina de estados do computador.

Esta máquina de estados tem como início o estado "Detetar faces". Este será o estado principal, porque após cada movimento do robô, o estado seguinte será o estado "Mover para a esquerda" ou "Mover para a direita". Caso a pessoa não esteja devidamente centrada com a câmara, o robô enviará uma mensagem sonora, para esta se colocar na devida posição. Se a pessoa estiver ligeiramente desviada para o lado esquerdo da imagem, o robô irá enviar a mensagem para esta dar um passo para a direita (estado "Mover para a direita"). Caso aconteça o contrário, a pessoa esteja ligeiramente no lado direito da imagem, será enviada uma mensagem para esta dar um passo para a esquerda (estado "Mover para a esquerda").

O último será o estado "Tirar foto", este é um estado onde o robô está centrado com a câmara, podendo então tirar a foto.

Na figura seguinte encontra-se o esquema de software e hardware relativemente ao programado robô e do programa do computador.

a)



⟶ **- Sinal analógico**

b)

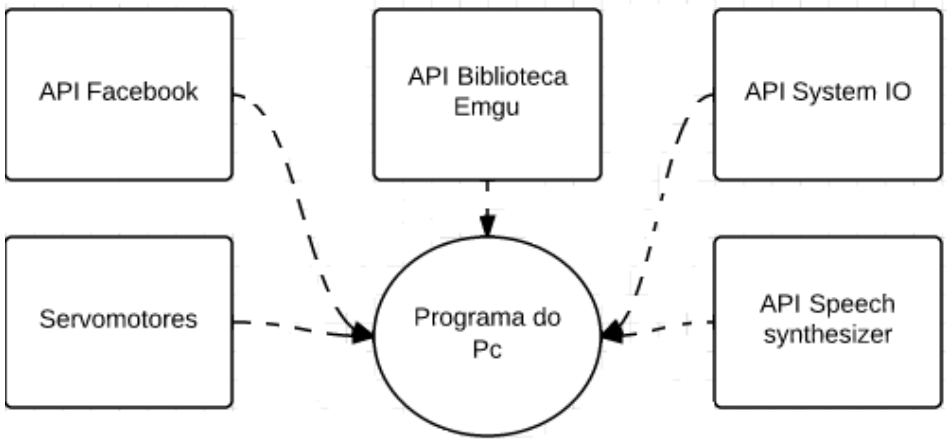


Figura 14 - a) Sistema de hardware do robô b) Sistema de software do computador.

Na parte do programa do robô, estão ligados vários sensores à placa, como por exemplo: sensor de distância, sensor queda e o giroscópio. Estes são sensores que foram implementados para permitir atingir os objectivos apresentados inicialmente para o projeto.

O sensor de distância permitiu saber se o robô deteta algum objecto dentro da sua área de alcance. Na secção 3.1.4 estará explicado com mais detalhe.

O sensor quedo foi implementado no robô, para se conseguir detetar uma possível queda. Na secção 3.1.3 podemos observar como este sensor funciona de uma maneira mais pormenor.

O sensor giroscópio tem como função principal manter o equilíbrio quando o robô se movimenta, na secção 3.1.2 podemos ver com mais detalhe o seu funcionamento.

Agora na parte do programa do computador. A câmara wireless permitiu o

reconhecimento facial de uma pessoa, podendo também tirar uma foto à pessoa para posteriormente ser colocada no Facebook. Na secção 3.1.1 está explicado com mais detalhe o funcionamento da câmara. Após isto seria necessário colocar a foto em algum lugar, surgiu a solução da rede social, o Facebook. Esta está descrita com mais detalhe na secção 3.6. Para se obter imagem da câmara no programa do computador, foi necessário instalar o EMGU, que é um NET wrapper da biblioteca OpenCV. Na secção 3.5 é explicado a implementação desta biblioteca.

Finalmente foi adicionado ao programa do computador a biblioteca System.IO, que permitiu o uso de uma porta série, para assim conseguir estabelecer comunicação entre as duas plataformas, o robô e o computador.

Para desenvolver o programa do computador e o do robô foi necessário utilizar duas ferramentas, tais como o VisualStudio2012 na linguagem C# para o computador e o RoboBasic v2.5 na linguagem Robobasic para o robô.

3.1. Sensores implementados no robô

3.1.1. Câmara de vídeo

Foi utilizada uma câmara de vídeo wireless, com o objetivo de captar imagem do meio exterior, para assim ser feita a detação facial da pessoa. A câmara foi montada numa nova cabeça, esta foi criada de propósito por uma impressora 3D. Na figura 15 podemos observar o novo modelo.

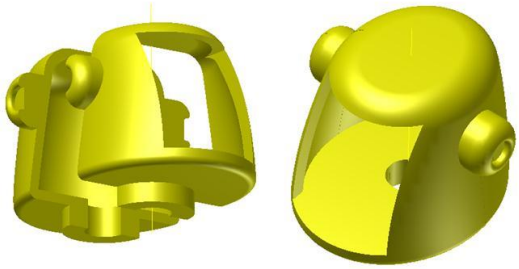


Figura 15 - Modelo da nova cabeça.

Com este modelo novo já seria possível instala-la no robô, na figura 16 podemos observar a câmara implementada.

Figura 16 - Câmara implementada no robô.

Especificações da câmara:

- Iluminação: 1 Lux
- Frequência: 1.2MHz
- Voltagem: 8V
- Alcance do sinal: 100 metros
- Resolução: 640x480 pixeis
- Peso: 18g
- Tamanho: 27 x 32 x 21m

Figura 17 - Câmara de video wireless.

É necessário instalar um driver para se conseguir obter imagem da câmara. O kit LogiLink USB2.0 Video Grabber é acompanhado por um CD de instalação. Quando este foi instalado, foi necessário ter algum cuidado, pois quando se liga ao computador este poderá instalar logo um driver que poderá não funcionar. Por isso quando se liga o aparelho ao computador, é necessário cancelar a actualização instantânea deste driver. Este driver é simples de instalar e assim a câmara irá funcionar correctamente. Na figura 16 está mostrado todo o

material necessário para obter imagem da câmara.

Figura 18 - Kit USB2.0 Video Grabber.

Este kit vem acompanhado com dois transformadores. Um deles serve para alimentar a câmara, e o outro para o recetor de imagem. Este é ligado ao computador por uma porta USB.

Especificações Kit LogiLink:

- Captura de vídeo e áudio pela interface USB2.0
- Tamanho compactado, fácil para transportar
- Captura de áudio sem ter placa de som
- Incluí um software de editor de vídeo profissional: Unlead Video Studio 10.0 SE DVD
- Suporta windows2000/XP/Vista/7

Foi utilizada uma câmara de baixo custo e de "marca branca". Esta muda os estados do robô consoante a distância da pessoa ao mesmo. Os estados envolvidos são: "Andar para a frente", "Andar para trás", "Mover para a esquerda" e "Mover para a direita". Estes estados estão referidos e explicados na seção 3.

3.1.2.Giroscópio

O Robonova pode ser programado para usar automaticamente os dados do giroscópio para tentar manter o equilíbrio quando se movimenta.

Foi usado um giroscópio Piezo Gyro PK3, que foi montado num dos ombros do robô. Este sensor foi ligado à porta analógica nº1 e à porta nº4. Na figura 19, está apresentado a ligação deste sensor.

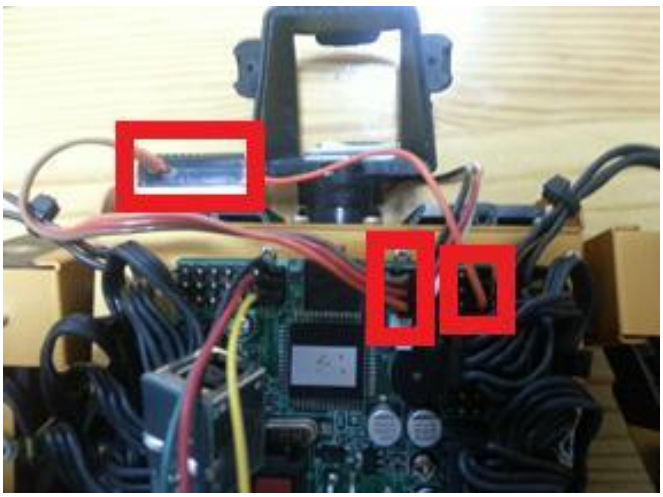


Figura 19 - Ligação do sensor giroscópio.

Na figura 20 podemos observar o sensor giroscópio.

Figura 20 - Giroscópio Piezo Gyro PK3.

Especificações:

- Voltagem de operação: 4.8V até 6.0V
- Corrente (a 4.8V): 10mA
- Temperatura: -5ºC até 60ºC
- Peso: 7g
- Tamanho: 26 x 27 x 11m

Neste código é onde se alteram os valores do sensor, para assim permitir que o robô execute os movimentos da melhor forma, por vezes o sensor irá corrigir alguns dos valores dos servomotores para assim criar um sistema estável.

Código 1- Configuração do sensor giroscópio.

```
'Configuração do sensor giroscópio'
DIR G6A,1,0,0,1,0,0
DIR  G6B,1,1,1,1,1,1
DIR  G6C,0,0,0,0,0,0
DIR  G6D,0,1,1,0,1,0
GETMOTORSET
G24,1,1,1,1,1,0,1,1,1,0,0,0,1,1,1,0,0,0,1,1,1,1,1,0
SPEED 5
```

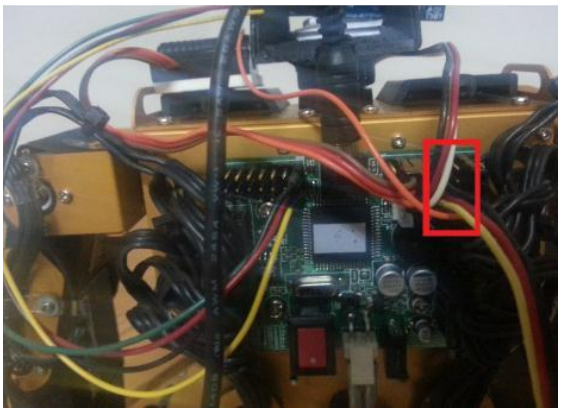
```
MOTOR G24
GYROSET G6A, 0, 1, 1, 0, 0, 0
GYROSET G6D, 0, 1, 1, 0, 0, 0
GYROSET G6B, 1, 0, 0, 0, 0, 0
GYROSET G6C, 1, 0, 0, 0, 0, 0
GYRODIR G6A,0,0,1,0,0,0
GYRODIR G6D,0,0,1,0,0,0
GYRODIR G6B,0,0,0,0,0,0
GYRODIR G6C,0,0,0,0,0,0
GYROSENSE G6A, 0,240,240, 0, 0, 0
GYROSENSE G6D, 0,240,240, 0, 0, 0
GYROSENSE G6B, 60,0,0, 0, 0, 0
GYROSENSE G6C, 60,0,0, 0, 0, 0
```

3.1.3. Sensor de queda

Este sensor foi implementado com o objectivo de poder detetar uma possível queda do robô. Este é um sensor caseiro, criado exclusivamente para o projeto, que basicamente consiste num interruptor electrónico.

O sensor quedo está ligado a porta analógica nº3. Na figura 21 está apresentado o sensor queda e a sua ligação ao robô.

a)



b)

Figura 21 - a) Sensor queda b) Implementação do sensor queda.

Especificações:

- Temperatura: -30ºC até 60ºC
- Resistência: pull-up de 20kΩ

Este sensor consegue determinar duas situações completamente diferentes. Uma em que o robô se encontra na posição vertical e outra na horizontal. Quando se encontra na posição horizontal, quer dizer que este sofreu uma queda.

Foi implementado o seguinte código para corrigir isso de uma forma simples e autónoma.

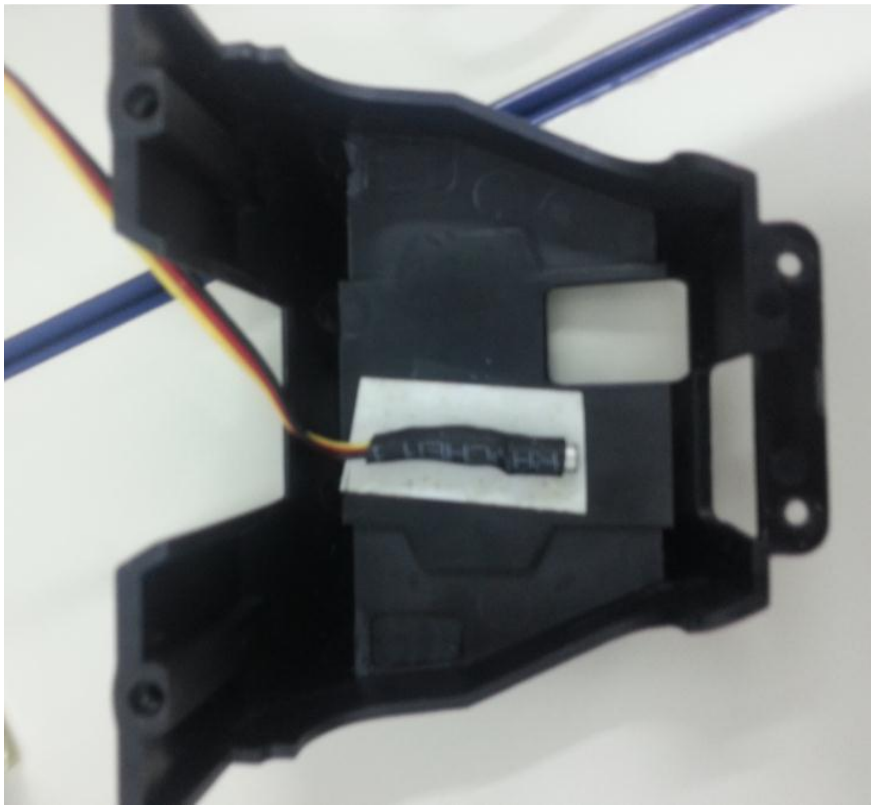


Figura 22 - Montagem do sensor queda às costas do RoboNova.

Código 2 - Função queda do robô.

```
Ligação feita à porta analógica(3) caso o valor do sensor
seja 1 (robo na vertical) volta para a pose inicial, caso
contrario,0, executa o movimento forward_standup, que faz
com que o robô se levante sozinho.

levantar:
B = AD (3)
IF B =0 THEN
GOSUB horizontal
RETURN
horizontal:
GOTO forward_standup
RETURN
```

Aqui é onde se corrige a queda do robô. Se o robô se encontrar na posição horizontal, o valor do sensor será 0, e assim se determina se irá executar o movimento para se levantar de uma forma autónoma.

3.1.4. Sensor de distância

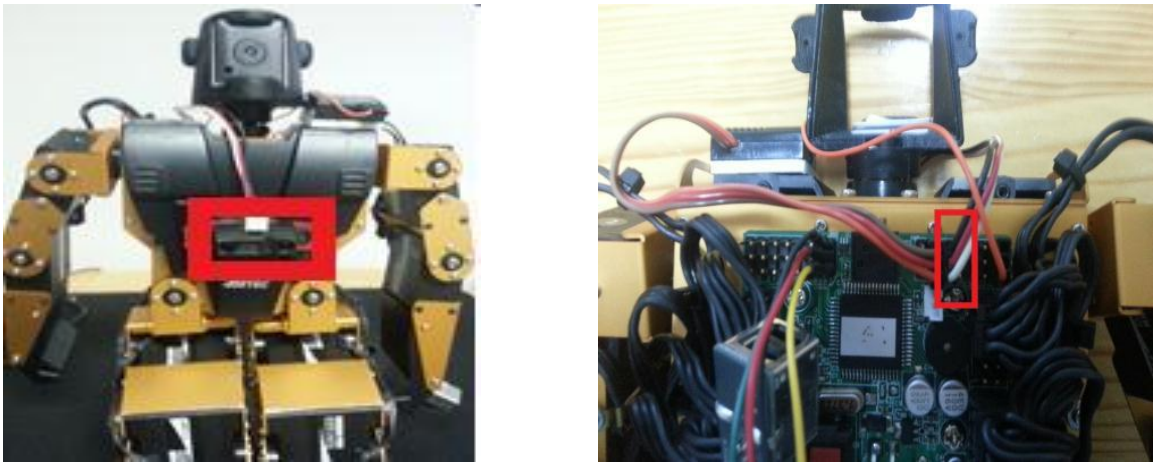
A implementação deste sensor tem como objetivo detetar se existe algum obstáculo (pessoa) perto do robô, para assim este poder mudar de estado. Como o robô tem um estado que engloba uma sequência de movimentos, foi necessário implementar este sensor para detetar se algum obstáculo (pessoa) se aproximar do robô durante a realização desta sequência. Esta compilação de movimentos do robô estará mais detalhada na seção 3.2.

Foi utilizado um sensor GP2D12 produzido pela empresa Sharp. Este sensor consegue detetar objetos até 80 centímetros de distância. Na figura 23 está um exemplo de um sensor de distância. Este sensor está ligado a porta analógica nº2.

Figura 23 - Sensor de distância

Especificações:

- Voltagem de operação: 4.5V até 5.5V
- Corrente (a 5V): 50mA
- Temperatura: -10ºC até 60ºC
- Distância: de 10cm a 80cm
- Peso: 12g
- Tamanho: 44 x 13 x 14 mm



a) b)

Figura 24 - a) Implementação de sensor de distância no robô b) Ligação do sensor de distância.

Na figura 24 está apresentado a ligação do sensor de distância ao robô e a sua

localização. Este foi montado na parte da frente, foi afixado ao robô.

Para converter a distância em cm é necessário colocar o valor lido pelo sensor numa fórmula de conversão. Após isto foram implementados alguns limites de distância, permitindo assim que exista um maior controle para a deteção de um objeto (pessoa), que se encontrar à frente do robô.

Código 3 -Função para medir a distância do objeto ao robô.

```
readsensor:

'Porta analógica(2) nesta função o valor lido pelo sensor,
passa por uma formula para assim conseguirmos converter o
valor'em cm. Se o valor for > 70 ou < 20 então está fora do
alcance caso contrario esta dentro.

B = AD (1)
IF B < 4 THEN B = 4
B = B - 3
B = 6787 / B
B = B - 4
IF B > 70 OR B < 20 THEN
GOSUB fora
ELSE
GOSUB dentro
ENDIF

RETURN
```

Nesta parte de código, é onde se calcula o valor em cm do sensor de distância. Inicialmente é lido o valor para uma variável que depois é transformado por uma fórmula para o valor em cm.

3.2. Os movimentos pré-programados do robô

Inicialmente necessitava-se de alguns movimentos interessantes do robô, para este poder captar a atenção das pessoas. Após alguma pesquisa, foram encontrados alguns movimentos interessantes. Com isto foi elaborada uma sequência de movimentos para o robô executar de uma só vez. Caso este detete algum objeto (pessoa) pessoa por perto, a sequência de movimentos deve parar e voltar à parte inicial do programa, aplicando-se aqui a mudança de estado anteriormente falada no seção 3.

Nesta sequência existem vários movimentos, um deles faz com que o robô represente um movimento da ginástica artística, o avião. Seguidamente temos outro em que o robô elabora um conjunto de movimentos dentro da área das artes marciais. Por último existe um movimento que faz com que o robô se deite no chão e seguidamente se levante, uma possível simulação de uma queda [8].

3.3. A comunicação entre o robô e o computador externo

Foi utilizado uma porta série para conseguir comunicar entre o computador e o robô. Para existir uma comunicação entre estas duas plataformas foi necessário utilizar um circuito S22-USB-SERIAL-INT-CONN, [9]. Este circuito converte os sinais da porta série do computador para sinais TTL usados pelo controlador do Robonova. Para além disto, foi instalado um driver [10], necessário para o funcionamento da comunicação.

Por aqui são enviados comandos para o robô, para este os executar. A biblioteca que permitiu esta ligação, é a System.IO, esta garante acesso às portas séries do computador, para uma possível programação.

Na figura 25, está apresentado a ligação do robô ao computador pela porta série. Esta é ligada à porta RX do robô. Esta porta é usada para enviar dados, neste caso do computador para o robô. Este por sua vez está ligado ao computador pela porta usb.

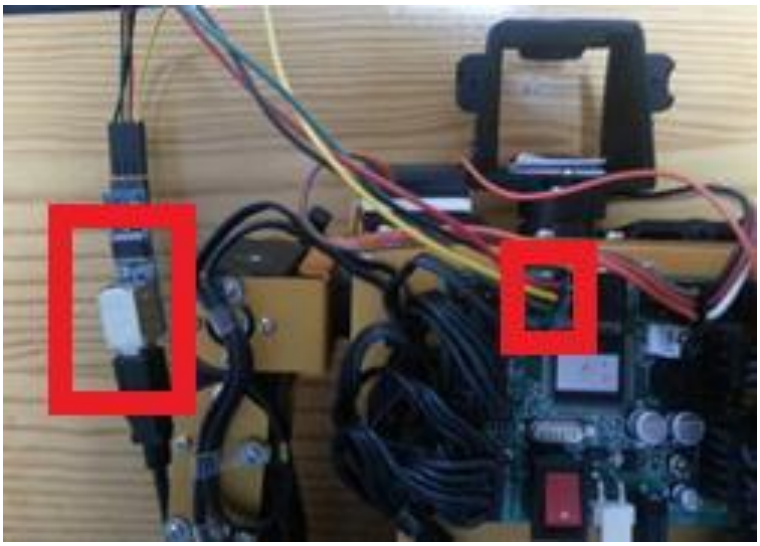


Figura 25 - Ligação da porta série

Código 4 - Criar a ligação da porta série

Criação da Porta:

```
SerialPort sp;
sp = new SerialPort("COM7");
sp.BaudRate = 57600;
sp.Parity = Parity.None;
sp.StopBits = StopBits.One;
sp.DataBits = 8;
sp.Handshake = Handshake.None;
try
   {
     //Abrir porta série
     sp.Open();
     //valor de leitura para 500 ms
     sp.ReadTimeout = 500;
```

```
    }
catch (System.Exception ex){ }
```

Posteriormente é necessário escrever na porta série, usando a linha de código (exemplo):

```
sp.write("A");
```

Nesta parte é criada uma ligação do computador ao robô, por a porta série. Inicialmente escolhemos qual a porta que queríamos ex: (COM7), após isto foi necessário definir um BaudRate e mais algumas propriedades da mesma.

3.4. A voz do robô

Foi criada uma voz artificial para uma possível comunicação entre robô e utilizador. Para isto necessitou-se de uma classe do framework.NET, e tem como nome Speech Synthesizer. Esta classe tem como objectivo fazer a sintetização da voz artificial, para assim criar uma interatividade mais eficaz. A implementação desta classe foi relativamente fácil, basta importa-la para projeto e assim já se pode trabalhar com ela.

Código 5 - Criação de voz artificial.

```
SpeechSynthesizer speaker = new SpeechSynthesizer();
Exemplo de reprudução
speaker.SpeakAsync("You are too far. Please come closer."
```

3.5. A deteção de faces e a captura da fotografia

Para a detetar as pessoas que se encontravam perto do robô, foi usado uma câmara wireless. As imagens capturadas pela câmara são enviadas para um recetor de ondas radio, que por sua vez envia para o programa principal através da porta USB do computador.

Neste caso utilizou-se a biblioteca EMGU do OpenCv.Esta biblioteca é um .NET wrapper para a biblioteca OpenCV e neste projeto foi utilizado para obter acesso às imagens transmitidas pela câmara e também para proceder à deteção facial em tempo real. Esta biblioteca pode ser obtida no site do OpenCV [6]. Neste existe toda a informação necessária para um bom funcionamento da biblioteca na aplicação.

É necessário ter cuidado com o sistema operativo do computador, porque existiram alguns problemas em receber imagem da câmara. O problema foi resolvido usando um sistema operativo de 32 bits.

Código 6 - Obter imagem da câmara.

```
Iniciar a variável
Capture capture = null;
Capturar imagem
capture = new Capture(0); Id da camara, este começa no
número 0 e avança consoante o número de aparelhos que
estejam ligados ao mesmo computador, exemplo (0,1,2,..).
```

Para que o robô possa tirar uma fotografia à pessoa, é necessário fazer a detecção facial. Assim consegue-se obter isoladamente a cara de cada uma das pessoas. Segue-se o código para efectuar o reconhecimento facial.

Código 7 - Detetar faces.

```
private void ProcessFrame(object sender, EventArgs arg)
        {

Image<Bgr, Byte> ImageFrame = capture.QueryFrame();
//Verifica a necessidade de efectuar uma rotação há imagem.
//Com uma webcam normal, parece não ser necessário  //double
i = 270;
// ImageFrame = ImageFrame.Rotate(i, new Bgr());
  if (ImageFrame != null)
     {
     Image<Gray, byte> grayframe = ImageFrame.Convert<Gray,
byte>();
     var faces = grayframe.DetectHaarCascade(haar, 1.4, 4,
HAAR_DETECTION_TYPE.DO_CANNY_PRUNING, new Size(25, 25))[0];
foreach (var face in faces)

               {
     rect = face.rect;
```

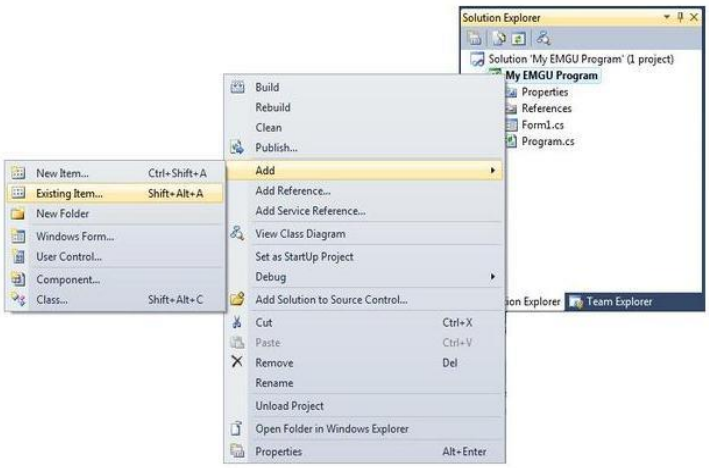
```
        ImageFrame.Draw(face.rect, new Bgr(Color.Green), 3);
        break;
                }
            }
```

Relativamente ao código anteriormente apresentado, é onde o robô capta imagem através da câmara implementada nele. Esta é transformada em tons de cinza para assim poder existir a detecção facial.

1º Passo - Incluir as DLLs no projeto

- Emgu.CV.dll
- Emgu.CV.UI.dll
- Emgu.Util.dll

a)



b)

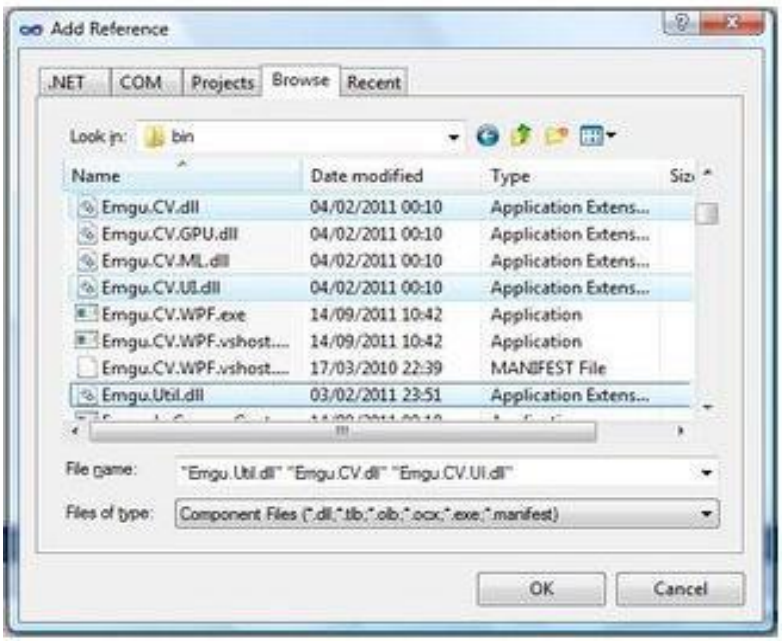


Figura 26- a) Adicionar item ao projeto b) Adicionar DLLs ao projeto.

2º Passo - Incluir ficheiro Haar.xml como anteriormente foram adicionados as DLLs.

3ºPasso - Definir o objeto imagebox, para assim mostrar imagens da câmara. Este objeto faz parte da DLL Emgu.CV.UI.dll. Para o podermos utilizar, deveremos adicioná-lo.

Este passo é necessário porque quando se testa o programa num computador diferente é necessário que alguns ficheiros sejam copiados para um novo diretório.

Figura 27 - Propriedade necessária para a implementação desta biblioteca

3.6. A publicação da fotografia no Facebook

O Facebook é uma rede social que permite a partilha de informações, fotos, etc.. Este projeto tem como finalidade utilizar esta rede para partilhar as fotos que vai tirando às pessoas que interagem com o robô.

Inicialmente quando foi implementado o código do projeto recorrente a esta parte, existiu um problema. A sessão já tinha expirado, sendo assim necessário criar outra conta e fazer a ligação desta à aplicação desenvolvida em C#. Todos os passos estarão detalhados num tutorial que se encontra na Internet[7]. O login para esta conta será:

Utilizador: RoboNovaGuarda
Password: robonovaipg1

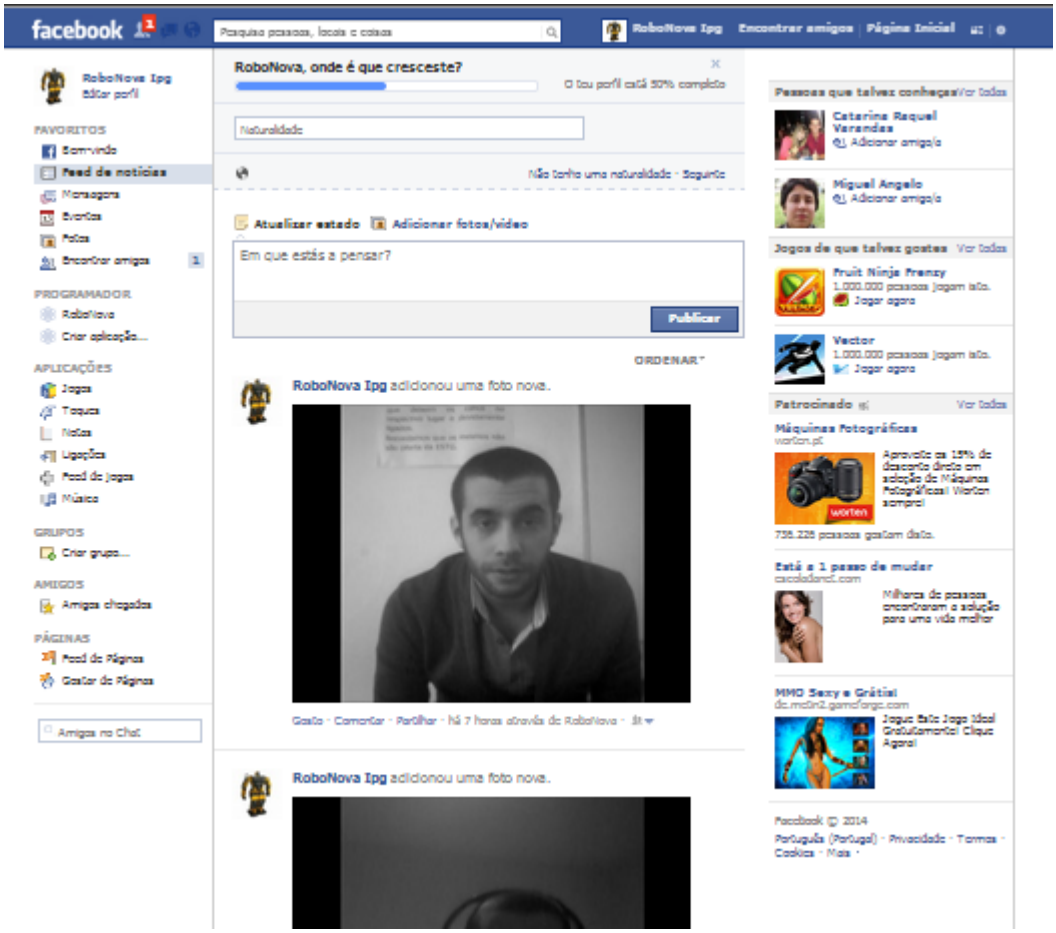


Figura 28 - Página do Facebook.

4. Conclusão

Após a finalização do projecto, foi verificado que consegui cumprir todos os objetivos apresentados tirando um, a cada à retaguarda.

A implementação do sensor queda tinha como objetivo inicial permitir ao robô resolver qualquer tipo de queda. Mas se existir uma queda à retaguarda o robô não poderá detetá-la, fazendo assim com que o robô não consigo voltar a uma posição vertical.

Durante o desenvolvimento do projeto existiram alguns inconvenientes. Foi necessário ter algum cuidado com os componentes do projeto. Pois alguns deles só funcionam com um driver específico ou é necessário ter uma plataforma que seja compatível a todos eles. É também preciso ter em conta o estado dos sensores, pois estes poderão estar danificados, comprometendo assim a sua utilidade para o desenvolvimento do projeto.

Neste momento dentro do projeto existe uma maior interação entre o robô e a pessoa. Existe um maior número de funcionalidades implementadas, permitindo assim elaborar um projeto mais interativo.

Gostei de trabalhar neste projeto, um robô diferente, com movimentos muito interessantes, este é capaz de chamar a atenção para outros possíveis projetos ou a continuação do mesmo.

4.1. Trabalho futuro

Existiram algumas componentes ao longo do desenvolvimento do projeto que poderiam ser melhoradas.

O robô neste momento interage com a pessoa se elas estiverem dentro da área de captação da câmara. Uma possível alteração será fazer com que o robô consiga detetar pessoas à sua volta. Para isto será necessário implementar um novo estado. Este estado consiste exclusivamente na rotação do robô. Para isto ser possível será necessário implementar um código que faça com que o robô rode sobre si mesmo, para assim conseguir abranger uma maior área de captação.

A queda à retaguarda não está implementada, o robô só corrige a queda frontal devido ao fato de este sensor não possibilitar tal efeito, mas posteriormente poderá arranjar-se um novo sensor que proporcione a resolução deste problema.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Almeida, Marco Vaz. *RobonovaIPG - Robô Interativo Autónomo*. Relatório de Projeto de Informática, Instituto Politécnico da Guarda, 2012.

[2] Robot Bioloid. [Online] http://www.robotis.com/xe/bioloid_en.

[3] Robot Nao. [Online] http://www.aldebaran-robotics.com/en/.

[4] Buckley, David. *Robonova Manual* [Online] http://www.davidbuckley.net/DB/RoboNova/RoboNova_files/RoboNova%20Manual-Eng-V1.50.pdf.

[5] RoboBASIC English Command Instruction Manual. [Online] 11 18, 2005. http://davidbuckley.net/DB/RoboNova/RoboNova_files/roboBASIC%20English%20Co mmand%20Instruction%20Manual(Version%202.10%2020051115).pdf.

[6] OpenCv. OpenCv. [Online] http://opencv.org/.

[7] Hamouda, Eslam. [Online] http://www.youtube.com/watch?v=Kc_3vLBtv3U.

[8] Program for roboScript. [Online] http://www.ing.unibs.it/~arl/docs/documentation/Robonova/Template%20Program%20f or%20roboScript/.

[9] Acroname. [Online] http://www.acroname.com/.

[10] FTDI Chip. [Online] http://www.ftdichip.com/FTDrivers.htm.

ANEXO A – PROGRAMA DO PC

```
//RobonovaIPG  versão  final
//Pedro Rocha 10 - 12 – 2013
//Este programa captura a imagem que está ligada ao
//computador, fazendo com que o robô mude de estado
//consoante a posição da pessoa à câmara. Posteriormente
//será colocada uma fotografia na página facebook com a
//cara da pessoa que interagiu com o robô.

using System;
using  System.Collections.Generic;
using       System.ComponentModel;
using System.Data;
using  System.Drawing;
using     System.Linq;
using System.Text;
using  System.Threading.Tasks;
using    System.Windows.Forms;
using Emgu.CV;
using Emgu.Util;
using  Emgu.CV.Structure;
using Emgu.CV.UI;
using Emgu.CV.CvEnum;
using  System.Speech.Synthesis;
using System.IO;
using  System.Diagnostics;
using        System.Media;
using     System.IO.Ports;
using AForge.Video;
using  AForge.Video.DirectShow;
using Facebook;
using System.Web;


namespace  RoboNovaIPG
{
public partial class AppFacebook : Form
{
//declaração de variáveis

Capture  capture  =  null;
private  HaarCascade  haar;
private Rectangle rect;
private Stopwatch sw = new Stopwatch();
private   bool   fotografar   =   false;
private int framessemface = 0;
private int framescomface = 0;




public static bool IsLogin = false;
public static bool IsBackFromLogin = false;
public static string estado;
public static int zonameio = 0;
```

```
//-- Speech --
SpeechSynthesizer speaker = new SpeechSynthesizer();

//-- Facebook --
// FacebookClient fb;

//-- Serial Port --
SerialPort sp;

public    AppFacebook()
{
InitializeComponent();

//-- Facebook --

Directory.CreateDirectory("fotos");

//-- OpenCV --

capture = new Capture(2); //id do aparelho ligado ao computador
neste caso é o 2.


//-- Serial Port --

//A porta série ligou-se à com7 que foi definida quando se
instalou o driver desta. O baudrate não pode ser alterado tem
que ser o que esta definido porque de outra maneira não existe
comunicação

sp = new SerialPort("COM7");
sp.BaudRate     =     57600;
sp.Parity   =    Parity.None;
sp.StopBits  =  StopBits.One;
sp.DataBits = 8;
sp.Handshake = Handshake.None;

try
{
//open serial port
sp.Open();
//set read time out to 500 ms
sp.ReadTimeout = 500;
}
catch (System.Exception ex)
{

}




////efetua login na conta facebbok
//  Frmlogin frm = new Frmlogin();
//   frm.ShowDialog();
```

```
//                  string                  imgURL                  =
string.Format("http://graph.facebook.com/{0}/picture",
FBClass.GetUserId(RoboNovaIPG.Properties.Settings.Default.Access
Token));
//  imgAccount.ImageLocation = imgURL;



Application.Idle += ProcessFrame;

}


private void Form1_Load(object sender, EventArgs e)
{


//incluir o ficheiro XML no projecto como se fez com as dlls.
haar = new HaarCascade("haarcascade_frontalface_default.xml");
sw.Start();
}


private void ProcessFrame(object sender, EventArgs arg)
{

//nesta função faz-se o tratamento da imagm convertendo-a para
//tons de cinzento para assim se poder fazer a detecção de
faces.
```

```
Image<Bgr, Byte> ImageFrame = capture.QueryFrame();

//Verificat a necessidade de efectuar uma rotação há imagem.
//Com uma webcam normal, parece não ser necessário
// double i = 270;
// ImageFrame = ImageFrame.Rotate(i, new Bgr());
if (ImageFrame != null)
{

Image<Gray, byte> grayframe = ImageFrame.Convert<Gray, byte>();
var faces = grayframe.DetectHaarCascade(haar, 1.4, 4,
HAAR_DETECTION_TYPE.DO_CANNY_PRUNING, new Size(25, 25))[0];
foreach (var face in faces)

{
rect = face.rect;
ImageFrame.Draw(face.rect, new Bgr(Color.Green), 3);
break;




}
}

//O imageBox1 é um componente da dll Emgu.CV.UI.dll que deve ser
adicionada à ToolBoxx como descrito em
//http://www.emgu.com/wiki/index.php/Add_ImageBox_Control
imageBox1.Image = ImageFrame;


//Esta condição faz com que se detete a posição da cara da
//pessoa segundo a imagem da camara. Assim é permitido saber se
//a pessoa deve colocar-se mais à esquerda ou à direita da
//camara para assim estrar centrada.

if (rect.Height == 0 || rect.Width == 0)
{
if (sw.ElapsedMilliseconds > 3000)
{
framessemface++;
framescomface = 0;

estado = "CHAMARATENCAO";
}
}
else
{


if (sw.ElapsedMilliseconds > 5000)
{
if (rect.X > 200 && rect.X < 322)
{
zonameio = 1;
}
```

```
if (rect.X > 322 && rect.X < 520)
{
estado = "PASSODIREITA";
sw.Restart();
}
if(rect.X > 0 && rect.X < 200)
{
estado = "PASSOESQUERDA";
sw.Restart();
}

if (rect.Height < 150 || rect.Width < 150)
{
if (zonameio == 1)
{
fotografar = false;
speaker.SpeakAsync("You are too far. Wait a moment.");




sw.Restart();
estado = "ANDARFRENTE";
zonameio = 0;
}


}
else if (rect.Height > 250 || rect.Width > 250)
{
if (zonameio == 1)
{
fotografar = false;
speaker.SpeakAsync("You are too close. Wait a moment");
sw.Restart();
estado = "ANDARTRAS";
zonameio = 0;
}
}
else
{

//Esta função faz com que se de ordem para tirar fotografia
(caso a condição se confirme) à pessoa.

if (fotografar)
{
fotografar = false;
CarregarFoto();
FBClass.Post(RoboNovaIPG.Properties.Settings.Default.AccessToken
, "");
}
else
```

```
{

if (zonameio == 1)
{
speaker.SpeakAsync("Perfect. Dont move please!");
fotografar = true;
estado = "TIRARFOTO";
sw.Restart();

zonameio=0;
}

}
}
}
}
rect = new Rectangle(0, 0, 0, 0);

// Aqui é apresentado à maquina de estados do programa. O robô
//irá executar todos estes estados dependendo da posição da
//pessoa.
```

```
switch  (estado)
{

case "PASSODIREITA":

speaker.SpeakAsync("Picture time !!  one step to right please");

estado = "";

break;

case "PASSOESQUERDA":

speaker.SpeakAsync("Picture time!! one step to left please");

estado = "";

break;

case "ANDARFRENTE":

sp.Write("E");.
estado = "";

break;

case "ANDARTRAS":

sp.Write("B");
estado = "";

break;

case "CHAMARATENCAO":

sp.Write("C");
estado = "";

break;

case "TIRARFOTO":

sp.Write("D");
estado = "";

break;
}

}

// Esta função permite o carregamento da foto para o facebook
```

```
private void CarregarFoto()
{

DateTime dt = DateTime.Now;
string imgname = "rnipg" + dt.ToString("ddMMyyyyHHmmss") +
".jpg";
capture.QueryFrame().Rotate(270, new Bgr()).Save(@"fotos\\" +
imgname);
string attachementPath = @"fotos\\" + imgname;

FBClass.PostImage(RoboNovaIPG.Properties.Settings.Default.Access
Token, attachementPath);

FBClass.Post(RoboNovaIPG.Properties.Settings.Default.AccessToken
, "");
}

private void AppFacebook_Activated(object sender, EventArgs e)
{

if (IsBackFromLogin == true)
{

btnLogin_Click(null, null);
IsBackFromLogin = false;

}

}
private void btnLogin_Click(object sender, EventArgs e) { }

}
}
```

ANEXO B – PROGRAMA DE CONTROLO DO ROBÔ

```
        ProjetoFinal versão final
        'Pedro Rocha 10 - 12 - 2013
        '
        'Declaração das variaveis a usar no programa

        DIM val AS BYTE
        '******* incluir algum código de inicialização se
necessário   *******
         GOTO AUTO
        FILL 255,10000

        DIM  RR  AS  BYTE
        DIM   X   AS   BYTE
        DIM  A16  AS  BYTE
        DIM  A26  AS  BYTE
        DIM QR AS  INTEGER
        DIM  B  AS  INTEGER
        DIM  C  AS  INTEGER
        DIM A AS INTEGER
        DIM  NDET  AS  INTEGER
        DIM  DET  AS  INTEGER
        DIM PS AS BYTE
        DIM  REMO  AS  BYTE
        DIM   J   AS   BYTE
        DIM   U   AS   BYTE
        DIM T AS BYTE

        PTP SETON
        PTP ALLON
```

```
'Criar a direção do motor

DIR G6A,1,0,0,1,0,0
DIR G6B,1,1,1,1,1,1
DIR G6C,0,0,0,0,0,0
DIR G6D,0,1,1,0,1,0

'Iniciar a posição do motor
TEMPO 230

GETMOTORSET
G24,1,1,1,1,1,0,1,1,1,0,0,0,1,1,1,0,0,0,1,1,1,1,1,0
'== motor power on  ============================
SPEED    5
MOTOR G24

'função com os movimentos para a posição inicial do
robô

GOSUB standard_pose

'Configuração do sensor giroscópio
GYROSET G6A, 0, 1, 1, 0, 0, 0
GYROSET G6D, 0, 1, 1, 0, 0, 0
GYROSET G6B, 1, 0, 0, 0, 0, 0
GYROSET G6C, 1, 0, 0, 0, 0, 0
GYRODIR        G6A,0,0,1,0,0,0
GYRODIR        G6D,0,0,1,0,0,0
GYRODIR        G6B,0,0,0,0,0,0
GYRODIR G6C,0,0,0,0,0,0
GYROSENSE G6A, 0,240,240, 0, 0, 0
GYROSENSE G6D, 0,240,240, 0, 0, 0
GYROSENSE  G6B,  60,0,0,  0,  0,  0
GYROSENSE G6C, 60,0,0, 0, 0, 0

'Programa principal
```

```
MAIN:

GOSUB levantar

' Comparar valor ao valor da porta para assim alterar o
estado do robô

 'IF val = 65 THEN  GOSUB recebeu_A
 'IF val = 66 THEN  GOSUB recebeu_B
 'IF val = 67 THEN  GOSUB recebeu_C
 'IF val = 68 THEN  GOSUB recebeu_D
 'IF val = 69 THEN  GOSUB recebeu_E

' voltar para o Main

GOTO MAIN


'== o comando ERX fica em ciclo enquanto não receber
algo pela porta série ==

'ler_porta:
 ERX 57600,val,ler_porta
       '******* mandar tocar aqui uma musica indicando
que foi recebido algo pela porta série *******

'Função para detetar a queda do robô

levantar:

'Porta analógica(3) caso o valor do sensor seja 0 (robo
na vertical) volta para a pose inicial, caso contrario
'executa o movimento forward_standup, que faz com que o
robo se levante sozinho

B = AD (3)
```

```
        IF B =0 THEN
        GOTO  standard_pose
        ELSE
        GOTO  forward_standup
        ENDIF

        RETURN

        readsensor:

        'Porta analógica(2) nesta função o valor lido pelo
sensor passa por uma formula para assim conseguirmos converter o
valor
        'em cm. Se o valor estiver for >70 ou <20 entao está
fora do alcance caso contrario esta dentro,

        B = AD (2)
        IF B < 4 THEN B = 4
        B = B - 3
        B = 6787 / B
        B = B - 4
        IF B > 70 OR B < 20 THEN
        GOSUB fora
        ELSE
        GOSUB  dentro
        ENDIF
        RETURN

        dentro:
        GOTO MAIN
        RETURN

        fora:
        RETURN

        '================================================
```

```
recebeu_A:
'Nesta função o robo ira andar em frente
 GOSUB fast_walk
 GOSUB levantar

RETURN
'================================================
recebeu_B:
 'Nesta função o robo ira andar para trás
 GOSUB backward_walk
 GOSUB levantar

RETURN


recebeu_C:
'Nesta função o robo irá executar uma sequencia de
movimentos

 GOSUB martial_arts_pose
 GOSUB levantar
 GOSUB martial_arts_pose2
 GOSUB levantar
 GOSUB  martial_arts_pose3
 GOSUB levantar
 GOSUB  martial_arts_pose4
 GOSUB levantar
 GOSUB  martial_arts_pose5
 GOSUB levantar
 GOSUB   fall_forward
 GOSUB  back_stand_up
 GOSUB levantar
RETURN

recebeu_D:
```

```
        'Nesta função o robo irá levantar os braços como sinal
de que o robô está preste a tirar uma fotografia

        GOSUB hands_up
        GOSUB levantar
        GOSUB standard_pose

        RETURN

        recebeu_E:

        'Nesta função o robo volta para a sua posição inicial
        GOSUB standard_pose

        RETURN

        hands_up:
          SPEED 5
          MOVE G6A, 100,  76, 145,  93, 100
          MOVE G6D, 100,  76, 145,  93, 100
          MOVE G6B, 100, 168, 150
          MOVE G6C, 100, 168, 150
          WAIT
        RETURN

        '================================================
        martial_arts_pose:
          HIGHSPEED   SETON
          SPEED 5
          'Martial Arts Pose (MAP) right
          MOVE G24, 92, 110, 85, 122, 109, , 100, 177, 163, , ,
, 100, 88, 132, , , , 60, 61, 162, 94, 130,
          HIGHSPEED  SETOFF
          WAIT
          DELAY 500
           'MAP transition (slow)
```

```
          SPEED 3
          MOVE G24, 74, 65, 142, 107, 140, , 100, 177, 163, , ,
, 102, 88, 132, , , , 86, 70, 147, 105, 95,
          WAIT
           'MAP      combo
          HIGHSPEED  SETON
          SPEED 5
          MOVE G24, 74, 66, 142, 108, 140, , 189, 92, 97, , , ,
158, 23, 50, , , , 89, 69, 144, 106, 91,
          WAIT
           'rn_4:
          MOVE G24, 100, 75, 135, 115, 102, , 145, 115, 71, , ,
, 160, 102, 60, , , , 99, 68, 142, 117, 97,
          WAIT
          DELAY 400
           'rn_5:
          MOVE G24, 103, 64, 113, 156, 98, , 156, 44, 33, , , ,
153, 34, 50, , , , 83, 112, 111, 115, 115,
          WAIT
          HIGHSPEED  SETOFF
          WAIT
          DELAY 800
           'MAP transitionraise arms (slow)
          SPEED 5
          MOVE G24, 94, 83, 98, 137, 99, , 68, 144, 125, , , ,
86, 127, 127, , , , 81, 116, 97, 110, 122,
          WAIT
           'rn_7:
          HIGHSPEED  SETON
          SPEED 5
          MOVE G24, 109, 126, 47, 146, 91, , 189, 96, 101, , ,
, 29, 180, 190, , , , 69, 147, 100, 77, 127,
          WAIT
          HIGHSPEED  SETOFF
          WAIT
          DELAY 1000
```

```
            'MAP tranition casual movements
            SPEED 4
            MOVE G24, 82, 42, 141, 131, 119, , 189, 96, 101, , ,
, 31, 180, 190, , , , 101, 113, 93, 113, 95,
            SPEED 5
            MOVE G24, 84, 57, 138, 130, 117, , 160, 24, 58, , , ,
157, 38, 75, , , , 99, 63, 126, 138, 95,
            WAIT
            'MAP transition casual movements
            SPEED 4
            MOVE G24, 86, 82, 125, 108, 101, , 76, 157, 180, , ,
, 85, 75, 139, , , , 97, 113, 94, 111, 113,
            WAIT
            DELAY 300
            'MAP left defend pose
            GOSUB    standard_pose
            RETURN

            martial_arts_pose2:
            HIGHSPEED SETON
            SPEED 5
            MOVE G24, 80, 72, 155, 83, 132, , 176, 103, 92, , , ,
36, 170, 190, , , , 98, 66, 164, 82, 86,
            DELAY 600
            WAIT
            HIGHSPEED  SETOFF
            WAIT
            'MAP arms up
            SPEED 5
            MOVE G24, 87, 61, 165, 90, 118, , 109, 159, 131, , ,
, 100, 141, 122, , , , 106, 53, 188, 75, 87,
            WAIT
            MUSIC "E" 'as code indicator durring troublshooting
            DELAY 200
            RETURN
```

```
martial_arts_pose3:
 'MAP kneel down attack
HIGHSPEED SETON
SPEED 5
MOVE G24, 103, 122, 59, 134, 88, , 177, 34, 45, , , ,
187, 74, 73, , , , 60, 120, 103, 93, 146,
WAIT
DELAY 300
 'rn_13:
MOVE G24, 116, 111, 61, 156, 87, , 185, 96, 93, , , ,
11, 183, 179, , , , 64, 121, 103, 111, 130,
WAIT
DELAY 300
HIGHSPEED  SETOFF
WAIT
RETURN

martial_arts_pose4:
 '(A) MAP casual defencive pose
SPEED 5
MOVE G24, 92, 85, 104, 125, 93, , 189, 10, 15, , , ,
187, 91, 97, , , , 80, 108, 92, 118, 131,
WAIT
 '(B) MAP back and fouth ballance movements (linked
to previous)
SPEED 3
MOVE G24, 83, 77, 123, 110, 99, , 189, 14, 15, , , ,
182, 92, 98, , , , 91, 116, 82, 116, 123,
WAIT
 '(C) MAP back and fouth ballance movements (linked
to previous)
SPEED 3
MOVE G24, 88, 73, 118, 122, 98, , 181, 14, 15, , , ,
180, 87, 98, , , , 87, 101, 90, 126, 124,
WAIT
```

```
          '(D) MAP back and fouth ballance movements (linked
to previous)
          MOVE G24, 93, 89, 86, 138, 101, , 181, 14, 15, , , ,
170, 88, 95, , , , 79, 104, 83, 129, 123,
          WAIT
          '(E) MAP back and fouth ballance movements (linked
to previous)
          MOVE G24, 81, 72, 116, 128, 106, , 190, 10, 15, , , ,
187, 97, 101, , , , 92, 105, 76, 138, 115,
          WAIT
          '(A) MAP back and fouth ballance movements (linked to
previous)
          MOVE G24, 92, 85, 104, 125, 93, , 189, 10, 15, , , ,
187, 91, 97, , , , 80, 108, 92, 118, 131,
          WAIT '
          'MAP transition casual
          SPEED 5
          MOVE G24, 108, 86, 103, 129, 90, , 104, 163, 157, , ,
, 102, 109, 145, , , , 71, 72, 139, 111, 127,
          WAIT
          RETURN


          martial_arts_pose5:
          GOSUB map_right_attack 'As found in the default
Overall Template Program
          DELAY 100
          'AP clap-like sequense
          SPEED 5
          MOVE G24, 101, 65, 148, 100, 99, , 190, 10, 44, , , ,
190, 10, 42, , , , 98, 63, 153, 99, 99,
          WAIT
          SPEED 5
          MOVE G24, 101, 65, 148, 100, 99, , 190, 66, 12, , , ,
190, 66, 14, , , , 98, 63, 153, 99, 99,
          WAIT
```

```
  GOSUB  standard_pose
  WAIT

RETURN

map_sit_pose:
  MOVE G6A, 99, 164, 23, 114, 99
  MOVE G6D, 99, 166, 21, 113, 98
  MOVE G6B, 170, 55, 45
  MOVE G6C, 117, 55, 68
  WAIT
  GOSUB standard_pose

RETURN

'================================================
'================================================

map_stand_pose:
  MOVE G6A, 98, 78, 111, 131, 100
  MOVE G6D, 93, 69, 118, 137, 105
  MOVE G6B, 189, 23, 45
  MOVE G6C, 174, 26, 71
  GOSUB standard_pose
RETURN
'================================================
'================================================

map_right_attack:
  SPEED 7
  GOSUB map_right_attack1

  HIGHSPEED SETON
  SPEED 8
  MOVE G6A, 58, 115, 77, 125, 134
```

```
  MOVE G6D, 93, 157, 20, 134, 110
  MOVE  G6B,  125,  79,  99
  MOVE  G6C,  107,  135,  108
  WAIT
  DELAY        1000
  HIGHSPEED  SETOFF
  SPEED 8
  GOSUB   map_sit_pose
  GOSUB  standard_pose
  RETURN
'================================================
map_right_attack1:
  MOVE  G6D,  85,  71,  152,  91,  107,  60
  MOVE  G6A,  108,  76,  145,  93,  100,  60
  WAIT

RETURN


forward_standup:

  SPEED 8

  MOVE G6A,100,  130,  120,   80,  110,  100
  MOVE G6D,100,  130,  120,   80,  110,  100
  MOVE G6B,150,  160,   10,  100,  100,  100
  MOVE G6C,150,  160,   10,  100,  100,  100
  WAIT

  MOVE G6A,  80,  155,   85,  150,  150,  100
  MOVE G6D,  80,  155,   85,  150,  150,  100
  MOVE G6B,185,   40,  60,   100,  100,  100
  MOVE G6C,185,   40,  60,   100,  100,  100
  WAIT

  MOVE G6A,  75,  165,   55,  165,  155,  100
```

```
  MOVE G6D, 75, 165,  55, 165, 155, 100
  MOVE G6B,185,  10, 100, 100, 100, 100
  MOVE G6C,185,  10, 100, 100, 100, 100
  WAIT

  MOVE G6A, 60, 165,  30, 165, 155, 100
  MOVE G6D, 60, 165,  30, 165, 155, 100
  MOVE G6B,170,  10, 100, 100, 100, 100
  MOVE G6C,170,  10, 100, 100, 100, 100
  WAIT

  MOVE G6A, 60, 165,  25, 160, 145, 100
  MOVE G6D, 60, 165,  25, 160, 145, 100
  MOVE G6B,150,  60,  90, 100, 100, 100
  MOVE G6C,150,  60,  90, 100, 100, 100
  WAIT

  MOVE G6A,100, 155,  25, 140, 100, 100
  MOVE G6D,100, 155,  25, 140, 100, 100
  MOVE G6B,130,  50,  85, 100, 100, 100
  MOVE G6C,130,  50,  85, 100, 100, 100
  WAIT

  GOTO standard_pose

  RETURN


backward_standup:
SPEED 8


  MOVE G6A,100,  10, 100, 115, 100, 100
  MOVE G6D,100,  10, 100, 115, 100, 100
  MOVE G6B,100, 130,  10, 100, 100, 100
  MOVE G6C,100, 130,  10, 100, 100, 100
```

```
  WAIT

  MOVE G6A,100,  10,  83, 140, 100, 100
  MOVE G6D,100,  10,  83, 140, 100, 100
  MOVE G6B, 20, 130,  10, 100, 100, 100
  MOVE G6C, 20, 130,  10, 100, 100, 100
  WAIT

  MOVE G6A,100, 126,  60,  50, 100, 100
  MOVE G6D,100, 126,  60,  50, 100, 100
  MOVE G6B, 20,  30,  90, 100, 100, 100
  MOVE G6C, 20,  30,  90, 100, 100, 100
  WAIT

  MOVE G6A,100, 165,  70,  15, 100, 100
  MOVE G6D,100, 165,  70,  15, 100, 100
  MOVE G6B, 30,  20,  95, 100, 100, 100
  MOVE G6C, 30,  20,  95, 100, 100, 100
  WAIT

  MOVE G6A,100, 165,  40, 100, 100, 100
  MOVE G6D,100, 165,  40, 100, 100, 100
  MOVE G6B,110,  70,  50, 100, 100, 100
  MOVE G6C,110,  70,  50, 100, 100, 100
  WAIT

  GOSUB  standard_pose
  RETURN

fast_walk:
DIM A10 AS BYTE
  SPEED 10
  MOVE G6B,100,  30,  90, 100, 100, 100
  MOVE G6C,100,  30,  90, 100, 100, 100
  WAIT
  SPEED 10
```

```
fast_run01:
  MOVE G6A, 90,  72, 148,  93, 110,  70
  MOVE G6D,108,  75, 145,  93,  95,  70
  WAIT
  SPEED  10
fast_run02:
  MOVE G6A, 90,  95, 105, 115, 110,  70
  MOVE G6D,112,  75, 145,  93,  95,  70
  MOVE G6B, 90,  30,  90, 100, 100, 100
  MOVE G6C,110,  30,  90, 100, 100, 100
  WAIT
  SPEED 10
'---------------------------  2 times
  FOR A10 = 1 TO 4

fast_run20:
  MOVE G6A,100,  80, 119, 118, 106, 100
  MOVE G6D,105,  75, 145,  93, 100, 100
  MOVE G6B, 80,  30,  90, 100, 100, 100
  MOVE G6C,120,  30,  90, 100, 100, 100
fast_run21:
  MOVE G6A,105,  74, 140, 106, 100, 100
  MOVE G6D, 95, 105, 124,  93, 106, 100
  MOVE G6B,100,  30,  90, 100, 100, 100
  MOVE G6C,100,  30,  90, 100, 100, 100
fast_run22:
  MOVE G6D,100,  80, 119, 118, 106, 100
  MOVE G6A,105,  75, 145,  93, 100, 100
  MOVE G6C, 80,  30,  90, 100, 100, 100
  MOVE G6B,120,  30,  90, 100, 100, 100
fast_run23:
  MOVE G6D,105,  74, 140, 106, 100, 100
  MOVE G6A, 95, 105, 124,  93, 106, 100
  MOVE G6C,100,  30,  90, 100, 100, 100
  MOVE G6B,100,  30,  90, 100, 100, 100
```

```
  NEXT A10
'------------------------------
  SPEED 8
  MOVE G6A, 85,  80, 130,  95, 106, 100
  MOVE G6D,108,  73, 145,  93, 100, 100
  MOVE G6B, 80,  30,  90, 100, 100, 100
  MOVE G6C,120,  30,  90, 100, 100, 100
  WAIT
fast_run03:
  MOVE G6A, 90,  72, 148,  93, 110,  70
  MOVE G6D,108,  75, 145,  93,  93,  70
  WAIT
  SPEED 5
  GOSUB standard_pose

  RETURN

backward_walk:

  SPEED 5
  GOSUB backward_walk1

  SPEED 13
  GOSUB backward_walk2

  SPEED 7
  GOSUB backward_walk3
  GOSUB backward_walk4
  GOSUB backward_walk5

  SPEED 13
  GOSUB backward_walk6

  SPEED 7
  GOSUB backward_walk7
  GOSUB backward_walk8
```

```
  GOSUB backward_walk9

  SPEED 13
  GOSUB backward_walk2

  SPEED 5
  GOSUB backward_walk1


  RETURN

backward_walk1:
  MOVE G6A, 85,  71, 152,  91, 112,  60
  MOVE G6D,112,  76, 145,  93,  92,  60
  MOVE G6B,100,  40,  80, , , ,
  MOVE G6C,100,  40,  80, , , ,
  WAIT
  RETURN

backward_walk2:
  MOVE G6A, 90, 107, 105, 105, 114,  60
  MOVE G6D,113,  78, 145,  93,  90,  60
  MOVE G6B, 90,  40,  80, , , ,
  MOVE G6C,100,  40,  80, , , ,
  WAIT
  RETURN

backward_walk9:
  MOVE G6A, 90,  56, 143, 122, 114,  60
  MOVE G6D,113,  80, 145,  90,  90,  60
  MOVE G6B, 80,  40,  80, , , ,
  MOVE G6C,105,  40,  80, , , ,
  WAIT
  RETURN

backward_walk8:
```

```
  MOVE G6A,100,  62, 146, 108, 100, 100
  MOVE G6D,100,  88, 140,  86, 100, 100
  MOVE G6B, 90,  40,  80, , , ,
  MOVE G6C,100,  40,  80, , , ,
  WAIT
  RETURN

backward_walk7:
  MOVE G6A,113,  76, 142, 105,  90,  60
  MOVE G6D, 90,  96, 136,  85, 114,  60
  MOVE G6B,100,  40,  80, , , ,
  MOVE G6C,100,  40,  80, , , ,
  WAIT
  RETURN

backward_walk6:
  MOVE G6D, 90, 107, 105, 105, 114,  60
  MOVE G6A,113,  78, 145,  93,  90,  60
  MOVE G6C,90,  40,  80, , , ,
  MOVE G6B,100,  40,  80, , , ,
  WAIT
  RETURN

backward_walk5:
  MOVE G6D, 90,  56, 143, 122, 114,  60
  MOVE G6A,113,  80, 145,  90,  90,  60
  MOVE G6C,80,  40,  80, , , ,
  MOVE G6B,105,  40,  80, , , ,
  WAIT
  RETURN

backward_walk4:
  MOVE G6D,100,  62, 146, 108, 100, 100
  MOVE G6A,100,  88, 140,  86, 100, 100
  MOVE G6C,90,  40,  80, , ,,
  MOVE G6B,100,  40,  80, , , ,
```

```
  WAIT
  RETURN

backward_walk3:
  MOVE G6D,113,  76, 142, 105,  90,  60
  MOVE G6A, 90,  96, 136,  85, 114,  60
  MOVE G6C,100,  40,  80, , , ,
  MOVE G6B,100,  40,  80, , , ,
  WAIT
  RETURN

standard_pose:

  MOVE G6A, 100,  76, 145,  93, 100, 100
  MOVE G6D, 100,  76, 145,  93, 100, 100
  MOVE G6B, 100,  30,  80, 100, 100, 100
  MOVE G6C, 100,  30,  80, 100, 100, 100
  WAIT

  DELAY 1000

  RETURN

wing_move:
  DIM i AS BYTE
  SPEED 5

  MOVE G6A, 85,  71, 152,  91, 112,  60
  MOVE G6D,112,  76, 145,  93,  92,  60
  MOVE G6B,100,  40,  80, , , ,
  MOVE G6C,100,  40,  80, , , ,
  WAIT

  MOVE G6A, 90,  98, 105, 115, 115,  60
  MOVE G6D,116,  74, 145,  98,  93,  60
  MOVE G6B,100, 150, 150, 100, 100, 100
```

```
MOVE G6C,100, 150, 150, 100, 100, 100
WAIT

MOVE G6A, 90, 121,  36, 105, 115,  60
MOVE G6D,116,  60, 146, 138,  93,  60
MOVE G6B,100, 150, 150, 100, 100, 100
MOVE G6C,100, 150, 150, 100, 100, 100
WAIT

MOVE G6A, 90,  98, 105,  64, 115,  60
MOVE G6D,116,  50, 160, 160,  93,  60
MOVE G6B,145, 110, 110, 100, 100, 100
MOVE G6C,145, 110, 110, 100, 100, 100
WAIT

FOR i = 10 TO 15
      SPEED i
      MOVE G6B,145,  80,  80, 100, 100, 100
      MOVE G6C,145,  80,  80, 100, 100, 100
      WAIT

      MOVE G6B,145, 120, 120, 100, 100, 100
      MOVE G6C,145, 120, 120, 100, 100, 100
      WAIT
NEXT i

DELAY  1000
SPEED 6

MOVE G6A, 90,  98, 105,  64, 115,  60
MOVE G6D,116,  50, 160, 160,  93,  60
MOVE G6B,100, 160, 180, 100, 100, 100
MOVE G6C,100, 160, 180, 100, 100, 100
WAIT

MOVE G6A, 90, 121,  36, 105, 115,  60
```

```
  MOVE G6D,116,  60, 146, 138,  93,  60
  MOVE G6B,100, 150, 150, 100, 100, 100
  MOVE G6C,100, 150, 150, 100, 100, 100
  WAIT
  SPEED 4

  MOVE G6A, 90,  98, 105, 115, 115,  60
  MOVE G6D,116,  74, 145,  98,  93,  60
  WAIT

  MOVE G6A, 85,  71, 152,  91, 112,  60
  MOVE G6D,112,  76, 145,  93,  92,  60
  MOVE G6B,100,  40,  80, , , ,
  MOVE G6C,100,  40,  80, , , ,
  WAIT

  GOSUB standard_pose

RETURN


forward_punch:

  DIM Y AS BYTE

  FOR Y = 0 TO 1

  SPEED 15
  MOVE G6A, 92, 100, 110, 100, 107, 100
  MOVE G6D, 92, 100, 110, 100, 107, 100
  MOVE G6B,190, 150,  10, 100, 100, 100
  MOVE G6C,190, 150,  10, 100, 100, 100
  WAIT
  SPEED 15
  HIGHSPEED SETON
```

```
  MOVE G6B,190,  10,  75, 100, 100, 100
  MOVE G6C,190, 140,  10, 100, 100, 100
  WAIT
  DELAY 500
  MOVE G6B,190, 140,  10, 100, 100, 100
  MOVE G6C,190,  10,  75, 100, 100, 100
  WAIT
  DELAY 500

  MOVE G6A, 92, 100, 113, 100, 107, 100
  MOVE G6D, 92, 100, 113, 100, 107, 100
  MOVE G6B,190, 150,  10, 100, 100, 100
  MOVE G6C,190, 150,  10, 100, 100, 100
  WAIT

  HIGHSPEED SETOFF
  MOVE G6A,100, 115,  90, 110, 100, 100
  MOVE G6D,100, 115,  90, 110, 100, 100
  MOVE G6B,100,  80,  60, 100, 100, 100
  MOVE G6C,100,  80,  60, 100, 100, 100
  WAIT

  NEXT Y

  GOSUB standard_pose

RETURN

fall_forward:
  SPEED 10
  MOVE G6A, 100, 155,  25, 140, 100, 100
  MOVE G6D, 100, 155,  25, 140, 100, 100
  MOVE G6B, 130,  50,  85, 100, 100, 100
  MOVE G6C, 130,  50,  85, 100, 100, 100
  WAIT
  MOVE G6A,  60, 165,  25, 160, 145, 100
```

```
  MOVE G6D,  60, 165,  25, 160, 145, 100
  MOVE G6B, 150,  60,  90, 100, 100, 100
  MOVE G6C, 150,  60,  90, 100, 100, 100
  WAIT
  MOVE G6A,  60, 165,  30, 165, 155, 100
  MOVE G6D,  60, 165,  30, 165, 155, 100
  MOVE G6B, 170,  10, 100, 100, 100, 100
  MOVE G6C, 170,  10, 100, 100, 100, 100
  WAIT
  SPEED 3
  MOVE G6A,  75, 165,  55, 165, 155, 100
  MOVE G6D,  75, 165,  55, 165, 155, 100
  MOVE G6B, 185,  10, 100, 100, 100, 100
  MOVE G6C, 185,  10, 100, 100, 100, 100
  WAIT
  SPEED 10
  MOVE G6A,  80, 155,  85, 150, 150, 100
  MOVE G6D,  80, 155,  85, 150, 150, 100
  MOVE G6B, 185,  40, 60,  100, 100, 100
  MOVE G6C, 185,  40, 60,  100, 100, 100
  WAIT
  MOVE G6A, 100, 130, 120,  80, 110, 100
  MOVE G6D, 100, 130, 120,  80, 110, 100
  MOVE G6B, 125, 160,  10, 100, 100, 100
  MOVE G6C, 125, 160,  10, 100, 100, 100
  WAIT
  RETURN

back_stand_up:
  SPEED 10
  MOVE G6A, 100, 130, 120,  80, 110, 100
  MOVE G6D, 100, 130, 120,  80, 110, 100
  MOVE G6B, 150, 160,  10, 100, 100, 100
  MOVE G6C, 150, 160,  10, 100, 100, 100
  WAIT
  MOVE G6A,  80, 155,  85, 150, 150, 100
```

```
MOVE G6D,  80, 155,  85, 150, 150, 100
MOVE G6B, 185,  40, 60,  100, 100, 100
MOVE G6C, 185,  40, 60,  100, 100, 100
WAIT
MOVE G6A,  75, 165,  55, 165, 155, 100
MOVE G6D,  75, 165,  55, 165, 155, 100
MOVE G6B, 185,  10, 100, 100, 100, 100
MOVE G6C, 185,  10, 100, 100, 100, 100
WAIT
MOVE G6A,  60, 165,  30, 165, 155, 100
MOVE G6D,  60, 165,  30, 165, 155, 100
MOVE G6B, 170,  10, 100, 100, 100, 100
MOVE G6C, 170,  10, 100, 100, 100, 100
WAIT
MOVE G6A,  60, 165,  25, 160, 145, 100
MOVE G6D,  60, 165,  25, 160, 145, 100
MOVE G6B, 150,  60,  90, 100, 100, 100
MOVE G6C, 150,  60,  90, 100, 100, 100
WAIT
MOVE G6A, 100, 155,  25, 140, 100, 100
MOVE G6D, 100, 155,  25, 140, 100, 100
MOVE G6B, 130,  50,  85, 100, 100, 100
MOVE G6C, 130,  50,  85, 100, 100, 100
WAIT

GOSUB standard_pose

RETURN
```