MEIC-A
Sistemas de Automação
2011 / 2012

MobotArch

Projecto de
Arquitectura Móvel de Robôs

Manual de Utilizador

Grupo de Trabalho:

- Gil Lacerda Nr.º 64761
- Rui Lageira Nr.º 64860

I. Cenários de Utilização

O **MobotArch** consta numa implementação, em linguagem de programação Java, de uma arquitectura inteligente que suporta os seguintes cenários de mobilidade de um ou mais Rasteirinhos (controlador MD25):

1. Seguimento de pista: o Rasteirinho é conduzido ao longo de uma linha de pista com a qual se tenta manter centrado.

Material necessário:

- Rasteirinho MD25
- Câmera *Web*
- eBox3350MX ou *Netbook*

Para realizar este cenário:

- seguir os passos descritos na secção **Instação da Arquitectura**.
- executar o programa *WebCamMaster.jar* (ver secção **Execução**):

exemplo 1 (desaconselhado no eBox) - para utilizar a câmera *web* associada ao dispositivo `/dev/video0`[1], ver no terminal informações do estado do programa e mostrar janelas com a imagem original da câmera, bem como a imagem a preto e branco com o *centroid* desenhado, executar:

```
> java -jar WebCamMaster.jar -vwi 0
```

exemplo 2 (recomendado no eBox) - para utilizar a câmera *web* associada ao primeiro dispositivo que for encontrado e visualizar o *frame-rate* no terminal:

- Correr em ambiente de consola para melhorar desempenho, premindo Ctrl+Alt+F1. (para voltar ao ambiente gráfico premir Ctrl+Alt+F7)

- Executar o programa:

```
> java -jar WebCamMaster.jar -fi 0
```

2. Seguimento de pista + "Follow the Leader" : um dos Rasteirinhos efectua o seguimento de pista e os restantes imitam o seu movimento, numa perspectiva de "Follow the Leader", segundo um modelo de comunicação de difusão. O Rasteirinho que segue pista constitui o "Líder" que difunde as ordens aos restantes Rasteirinhos - seguidores do "Líder", segundo um modelo Mestre → um ou mais Escravos.

Material necessário:

- Dois ou mais Rasteirinhos MD25
- Câmera *Web*
- *Router wireless*
- Computador(es) eBox3350MX + placa(s) *wireless* USB

e/ou

- *Netbook*(s)

Para realizar este cenário:

- seguir os passos descritos na secção **Instação da Arquitectura** para cada computador interveniente.

1 executar `ls /dev/video*` para listar câmeras disponíveis.

No(s) computadores a controlar (Seguidor(es) do Líder):

- ligar à rede local do *router*
- executar o programa *Slave.jar* (ver secção **Execução**)

exemplo – para usar o porto de comunicação UDP pré-definido (64860) executar:

```
> java –jar Slave.jar
```

No computador que realiza o seguimento de pista ("Líder"):

- criar ficheiro *Slaves.txt* inserindo <porto>@<endereço IP local> de cada seguidor, por linha.[2]
- executar o programa *WebCamMaster.jar* (ver secção **Execução**)

exemplo 1 (desaconselhado no eBox) - para utilizar a câmera *web* associada ao dispositivo `/dev/video1`[1], ver no terminal informações do estado do programa e mostrar janelas com a imagem original da câmera, bem como a imagem a preto e branco com o *centroid* desenhado e controlar todos os seguidores intervenientes, executar:

```
> java –jar WebCamMaster.jar –vwib 1 BROADCAST
```

exemplo 2 (recomendado no eBox) - para utilizar a câmera *web* associada ao primeiro dispositivo que for encontrado e controlar todos os seguidores intervenientes:

- Correr em ambiente de consola para melhorar desempenho, premindo Ctrl+Alt+F1. (para voltar ao ambiente gráfico premir Ctrl+Alt+F7)

- Executar o programa:

```
> java –jar WebCamMaster.jar –b BROADCAST
```

3. Controlo por teclado

I. Controlo local: um Rasteirinho é controlado localmente pelo computador que o conduz.

Material necessário:

- Computador eBox3350MX
- Rasteirinho MD25
- Teclado

Para realizar este cenário:

- seguir os passos descritos na secção **Instação da Arquitectura**.
- executar o programa *KeyboardMaster.jar* (ver secção **Execução**)

exemplo:

```
> java –jar KeyboardMaster.jar
```

II. Controlo local + "Follow the Leader": um Rasteirinho é controlado localmente pelo computador que o conduz e os restantes são controlados por este, segundo um modelo Mestre → um ou mais Escravos, via *wireless*.

Material necessário:

- Computador(es) eBox3350MX + placa(s) *wireless* USB
e/ou
- *Netbook*(s)
- Rasteirinhos MD25
- Teclado
- *Router wireless*

2 executar `ifconfig` em cada eBox e procurar endereço IP local (192.168.X.Y) na rede sem fios.

Para realizar este cenário:

- seguir os passos descritos na secção **Instação da Arquitectura** em cada computador interveniente.
- ligar cada computador interveniente à rede local do *Router*.

No(s) computador(es) a controlar:

- executar o programa *Slave.jar* (ver secção **Execução**)

exemplo – para usar o porto de comunicação UDP pré-definido (64860) executar:

```
> java –jar Slave.jar
```

No computador de controlo:

- criar ficheiro *Slaves.txt* inserindo <porto>@<endereço IP local> de cada eBox, por linha.[2]
- executar o programa *KeyboardMaster.jar* (ver secção **Execução**)

exemplo – para enviar os comandos de controlo para todos os intervenientes, executar:

```
> java –jar KeyboardMaster.jar –lb Slaves.txt BROADCAST
```

exemplo – para enviar comandos de controlo para um interveniente específico com IP exemplo 192.168.1.193, executar:

```
> java –jar KeyboardMaster.jar –lb Slaves.txt 192.168.1.193
```

III. Controlo remoto: um computador remoto (dentro de uma rede local *wireless*) conduz um ou mais Rasteirinhos, segundo um modelo Mestre → um ou mais Escravos, via wireless.

Este cenário é semelhante ao anterior, com as seguintes diferenças:

- o computador mestre que controla os restantes não está associado a nenhum Rasteirinho.

- nos exemplos de execução do programa do computador Mestre, no argumento das opções deve ser incluída a opção `r` (indica que apenas os módulos remotos dos escravos são controlados e não o módulo do próprio).

II. Instalação da Arquitectura

A arquitectura foi testada no sistema operativo Crunchbang 10, pelo que se recomenda a execução neste ambiente, no entanto existe compatibilidade com qualquer sistema operativo Linux. Em distribuições baseadas em Debian, a instalação da arquitectura deverá ser directa, sendo válidas as instruções a partir do ponto **Instalação de Java Runtime Environment (JRE)**.

Instruções para instalar o sistema operativo recomendado (eBox3350MX ou outro)

- Instalar o Crunchbang 10 i386 numa *pendrive,* cartão SD ou disco rígido (se existir), a partir de uma outra *pendrive* arrancável com a imagem *iso* (seguir este tutorial). A instalação deve ser efectuada utilizando o próprio computador onde se pretende correr o sistema operativo.

- Obter ligação à internet (via *ethernet* se não configurou nenhum dispositivo wireless). Executar os seguintes comandos:

```
~$ sudo apt-get update
~$ sudo apt-get dist-upgrade
```

Instalação de Java Runtime Environment (JRE)

```
~$ sudo apt-get install openjdk-6-jre
```

Instalação de Bibliotecas Partilhadas

No comandos seguintes `PROJ_DIR` indica o caminho absoluto para a pasta que resulta de descomprimir o ficheiro comprimido do projecto (*MobotArch.tar.gz*).

```
~$ sudo apt-get install libavcodec53
~$ sudo apt-get install libavformat53
~$ sudo apt-get install libswscale2
~$ mkdir <dir_bibliotecas_partilhadas>
~$ cp PROJ_DIR/native_libraries/* <dir_bibliotecas_partilhadas>
```

- Adicionar a directoria <dir_bibliotecas_partilhadas> à variável de ambiente LD_LIBRARY_PATH, adicionando no ficheiro `~/.bashrc` as seguintes linhas:

```
LD_LIBRARY=$LD_LIBRARY_PATH:<dir_bibliotecas_partilhadas>
export LD_LIBRARY_PATH
```

→ Neste ponto os programas da arquitectura deverão estar aptos a serem executados.

III. Execução

Os programas executáveis encontram-se na pasta *runnable* incluída no conteúdo do ficheiro comprimido *MobotArch.tar.gz*. Para extrair o conteúdo, pode ser utilizado o comando:

```
tar -zxvf MobotArch.tar.gz <dir_destino>
```

A sintaxe de execução dos programas da arquitectura é a seguinte:

```
java -jar {NomeDoPrograma}.jar [-options] [camera_id] [threshold] [max_value]
     [max_area] [masters_file] [slaves_file] [server_port] [client_port] [slave]
```

Os argumentos dos programas são opcionais e são descritos na tabela em baixo. As últimas três colunas da tabela indicam quais os argumentos que são suportados em cada programa.

		{NomeDoPrograma}		
Parâmetro	Descrição	WebCam Master	Keyboard Master	Slave
-options	Opções do programa e *flags* dos argumentos. Inclúi os caracteres indicados em baixo. Se for especificada mais do que uma opção os caracteres correspondentes devem ser concatenados. A especificação dos argumentos relativos às suas *flags* deve ser feita pela ordem em que as *flags* foram concatenadas.			
	`i` – *flag*: o argumento `camera_id` é especificado	X		
	`T` – *flag*: o argumento `threshold` é especificado	X		
	`V` – *flag*: o argumento `max_value` é especificado	X		
	`A` – *flag*: o argumento `max_area` é especificado	X		
	`m` – *flag*: o argumento `masters_file` é especificado			
	`l` – *flag*: o argumento `slaves_file` é especificado	X	X	
	`s` – *flag*: o argumento `server_port` é especificado	X	X	
	`c` – *flag*: o argumento `client_port` é especificado			X
	`b` – *flag*: o argumento `slave` é especificado	X	X	
	`v` – imprimir informações de estado do programa	X	X	X
	`d` – imprimir informações de *debug* do programa	X	X	X
	`r` – difundir comandos apenas para módulos remotos e não para os próprios		X	
	`w` – mostrar janelas de imagem original da câmera *web* e imagem de orientação	X		
	`a` – mostrar na janela de orientação todos os *centroids* calculados	X		
	`f` – imprimir no terminal o *frame-rate* da câmera em tempo de execução	X		
`camera_id`	Inteiro que especifica o *id* da câmera *web* associada ao dispositivo lógico do sistema operativo: `/dev/video`{*id*}[1]. Atribuir o valor -1 origina a captura da primeira câmera que for encontrada. Se não for especificado, o valor por omissão é `-1`	X		
`threshold`	Valor entre 0-255 que especifica a barreira entre o preto e o branco no cálculo da imagem p&b calculada. Se não for especificado, o valor por omissão é `55`	X		

`max_value`	Valor a ser atribuído ao pixel se valor(pixel) < threshold. Se não for especificado, o valor por omissão é `255`	X		
`max_area`	Valor máximo de área permitido para cálculo do seu centro (*centroid*). Se não for especificado, o valor por omissão: `18000`	X		
`masters_file`	(Não usado) Ficheiro de texto que contém por cada linha a identificação de cada interveniente controlador, de acordo com a notação <porta>@<endereço IP local>. Se não for especificado, o valor por omissão é `Masters.txt`			
`slaves_file`	Ficheiro de texto que contém por cada linha a identificação de cada interveniente remoto a controlar, de acordo com a notação <porta>@<endereço IP local> (ver ficheiro de exemplo *Slaves.txt*). Se não for especificado, o valor por omissão é `Slaves.txt`	X	X	
`server_port`	Porto de comunicação UDP do interveniente que controla os intervenientes remotos. Se não for especificado, o valor por omissão é `64761`	X	X	
`client_port`	Porto de comunicação UDP do interveniente que é controlado. Se não for especificado, o valor por omissão é `64860`			X
`slave`	IP local do interveniente particular a ser controlado, ou `BROADCAST` para controlar todos os intervenientes.	X	X	