

CENTRO UNIVERSITÁRIO FEI

DOCUMENTAÇÃO ROBOFEI HUMANOID TEAM: Atom

São Bernardo do Campo

2024

SUMÁRIO

1	INTRODUÇÃO	8
2	PROGRAMAÇÃO	9
2.1	PROCEDIMENTOS	9
2.1.1	Ligar os motores	9
2.1.2	USB Rules	9
2.1.3	Comandos personalizados no terminal	11
2.1.4	Servidor UDP	11
2.1.5	Conectar Wi-Fi	12

LISTA DE ILUSTRAÇÕES

Figura 1	–	Exemplo de arquivo .rules	10
Figura 2	–	Exemplo utilizando a interface <i>video</i>	10
Figura 3	–	Informações da interface <i>video</i>	11
Figura 4	–	Reiniciar as regras de interface <i>video</i>	11

LISTA DE TABELAS

LISTA DE ABREVIATURAS

This document is incomplete. The external file associated with the glossary ‘acronym’ (which should be called `documentacao.acr`) hasn’t been created.

This has probably happened because there are no entries defined in this glossary. Did you forget to use `type=acronym` when you defined your entries? If you tried to load entries into this glossary with `\loadglsentries` did you remember to use `[acronym]` as the optional argument? If you did, check that the definitions in the file you loaded all had the type set to `\glsdefaulttype`.

This message will be removed once the problem has been fixed.

LISTA DE SÍMBOLOS

This document is incomplete. The external file associated with the glossary ‘symbols’ (which should be called `documentacao.sls`) hasn’t been created.

This has probably happened because there are no entries defined in this glossary. Did you forget to use `type=symbols` when you defined your entries? If you tried to load entries into this glossary with `\loadglsentries` did you remember to use `[symbols]` as the optional argument? If you did, check that the definitions in the file you loaded all had the type set to `\glsdefaulttype`.

This message will be removed once the problem has been fixed.

LISTA DE ALGORITMOS

1 INTRODUÇÃO

Esse documento tem como objetivo registrar o desenvolvimento do projeto Atom (documentação mecaninca, elétrica e de programação), a ideia inicial é anotar o máximo de informações possíveis sobre cada área.

2 PROGRAMAÇÃO

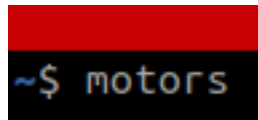
2.1 PROCEDIMENTOS

Esta seção tem como objetivo relatar os procedimentos gerais da programação, por exemplo como ligar o robô, quais as funcionalidades que temos e como podemos modifica-las entre outros procedimentos. Lembrando sempre que tudo que relatarmos aqui não significam que sejam os modos mais eficientes ou da melhor forma, sempre questionem os métodos e pensem melhores maneiras para resolver os problemas, e quando encontrarem novas formas melhores, pois vão encontrar, alterem esse documentos.

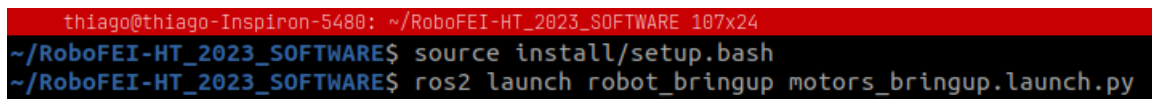
2.1.1 Ligar os motores

Primeiramente devemos verificar se os motores estão com energia, para isso basta apenas conectar a bateria e colocar o botão de emergência na posição ON. Caso o motor esteja alimentado de forma correta, os LED's deles deverão acender e em seguida desligar. Caso isso não ocorra solicitar ajuda para a equipe de Elétrica.

Com os motores alimentados (certifique-se de que o código está compilado) podemos então executar o nó dos motores de duas formas, sendo a primeira executando o comando *motors* no terminal:



Ou indo até o diretório do RoboFEI e executando o comando *source install/setup.bash* e na sequência o launch:



2.1.2 USB Rules

Como utilizamos mais de uma porta USB, uma para os motores e outra para a IMU foi necessário encontrarmos uma maneira de separar os USB's. Outro problema também que encontramos é que quando ligamos o USB o sistema do Linux não nos concede automaticamente permissão o suficiente para fazermos certas coisas, como por exemplo quando vamos executar o

código dos motores, caso não seja dada permissão antes o código não consegue comunicar com os motores.

Para tal problema podemos utilizar "Regras" para as USB, dessa forma podendo atribuir novos nomes e definir novas permissões, esse arquivo está contido no diretório do RoboFEI dentro de `robot_commands`, e sua extensão é `.rules`.

Figura 1 – Exemplo de arquivo `.rules`

```
# IMU Rules
KERNEL=="ttyUSB*", ATTRS{serial}=="A6022L8Q", SYMLINK+="imu", MODE="0777"

# Motors Rules
KERNEL=="ttyUSB*", ATTRS{serial}=="AI05A3SZ", SYMLINK+="motors", MODE="0777"

# Camera Rules
KERNEL=="video*", ATTRS{product}=="HD Pro Webcam C920",ATTR{index}=="0", SYMLINK+="camera"
```

Explicando um pouco mais do arquivo, inicialmente podemos ver que ambas as definições iniciam com o *KERNEL* (é o responsável por controlar os periféricos do computador), e nele vamos definir qual interface queremos modificar as regras. Para indicar o USB utilizamos o `ttyUSB` (em alguns sistemas se utilizam `ttyACM`), o símbolo de asterisco nos diz que iremos inicialmente selecionar todos os componentes que estão conectados ao USB, e idem para o vídeo, mas que posteriormente serão filtrados de alguma maneira.

citar
<https://www.si>

Primeiramente analisando as linhas da IMU e dos motores, podemos ver na sequência da especificação do *KERNEL*, uma outra filtragem é utilizando o valor serial do USB, no nosso caso cada conversor que utilizamos possui um número serial (podemos utilizar outros métodos para filtrar qual USB queremos encontrar), em seguida utilizamos o *SYMLINK* para linkar a porta USB com um novo nome, e utilizamos *MODE* para definir quais as permissões que precisamos para o USB.

Utilizamos a mesma linha de raciocínio para a câmera, porém utilizamos o nome do produto e o *index* para definirmos onde qual a interface que queremos modificar.

Os procedimentos que devemos seguir caso precisemos alterar as regras é inicialmente saber qual interface o dispositivo está conectado, eles ficam todos situados no diretório `/dev` (podendo ser por exemplo `ttyUSB`, `video`, entre outros). sabendo qual a interface podemos usar o comando:

Figura 2 – Exemplo utilizando a interface `video`

```
thiago@thiago-Inspiron-5480:~ 80x24
~ ▶ ls -la /dev/video*
crw-rw----+ 1 root video 81, 0 Oct  4 21:46 /dev/video0
crw-rw----+ 1 root video 81, 1 Oct  4 21:46 /dev/video1
```

Na sequencia podemos ver as informações dessa interface como mostra a imagem abaixo.

Figura 3 – Informações da interface *video*



```

~ ▶ udevadm info -a -n /dev/video0

Udevadm info starts with the device specified by the devpath and then
walks up the chain of parent devices. It prints for every device
found, all possible attributes in the udev rules key format.
A rule to match, can be composed by the attributes of the device
and the attributes from one single parent device.

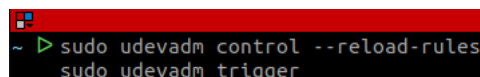
looking at device '/devices/pci0000:00/0000:00:14.0/usb1/1-6/1-6:1.0/video4linux/video0':
KERNEL=="video0"
SUBSYSTEM=="video4linux"
DRIVER=="
ATTR{dev_debug}=="0"
ATTR{index}=="0"
ATTR{name}=="Integrated_Webcam_HD: Integrate"
ATTR{power/async}=="disabled"
ATTR{power/control}=="auto"
ATTR{power/runtime_active_kids}=="0"
ATTR{power/runtime_active_time}=="0"
ATTR{power/runtime_enabled}=="disabled"
ATTR{power/runtime_status}=="unsupported"
ATTR{power/runtime_suspended_time}=="0"
ATTR{power/runtime_usage}=="0"

```

Como podemos ver, cada interface tem sua combinação de informações e utilizando elas podemos separar a interface que queremos utilizar.

Após criar o arquivo *.rules* com as suas necessidades, basta copiar o arquivo para o diretório */etc/udev/rules.d* e em seguida realizar a sequencia de comandos:

Figura 4 – Reiniciar as regras de interface *video*



```

~ ▶ sudo udevadm control --reload-rules
sudo udevadm trigger

```

Caso tenha ocorrido tudo certo, suas regras já estarão funcionando, e você pode verificar elas usando o comando *ls -la /dev/{novo nome interface}*.

Caso fique alguma dúvida pode-se utilizar o link <https://dev.to/enbis/how-udev-rules-can-help-us-to-recognize-a-usb-to-serial-device-over-dev-tty-interface-pbk>, onde possui uma melhor explicação do assunto e uma outra abordagem para realizar esse processo.

2.1.3 Comandos personalizados no terminal

2.1.4 Servidor UDP

FAZER

2.1.5 Conectar Wi-Fi

FAZER