

## Apêndice A

# Experimento 1: Ligar e desligar a plataforma de desenvolvimento

Este experimento objetiva fornecer um primeiro contato do usuário com o sistema robótico do manipulador UR3. Ele permitirá que o usuário seja capaz de:

- realizar o processo de inicialização do robô manipulador UR3;
- realizar o processo de inicialização do Linux PC;
- rodar um experimento simples;
- realizar o processo de desligamento do sistema.

### A.1 Setup: Robô manipulador UR3 e Controller

A Figura A.1 mostra a plataforma de desenvolvimento de aplicações para o braço manipulador UR3, presente no LARA, usando o ROS.

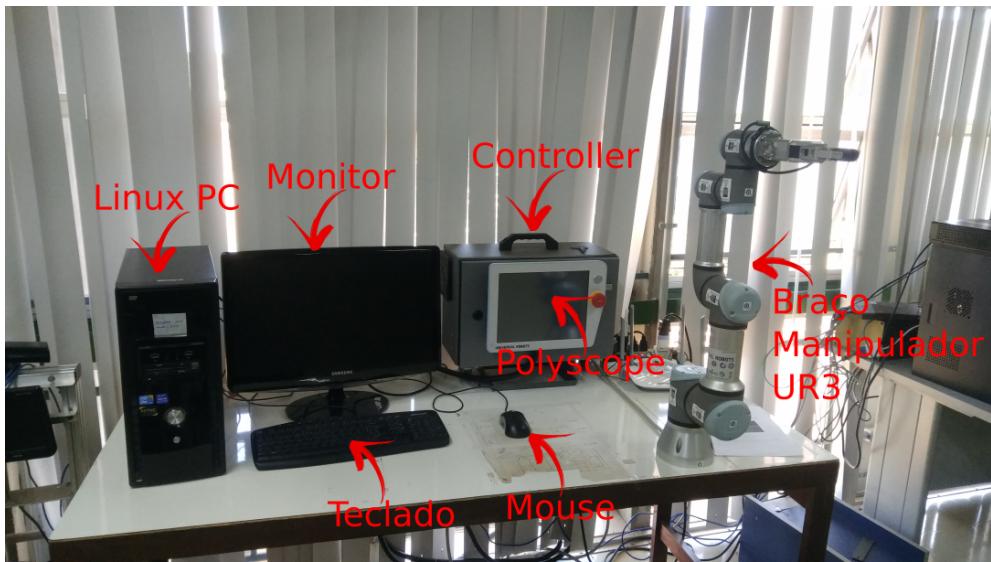


Figura A.1: Sistema completo

O computador do LARA, designado para fechar uma malha de controle com o robô UR3, é apresentado na Figura A.1 e é denominado **Linux PC**. O roteador dedicado ao sistema é chamado **roteador multi-robôs**.

A comunicação entre o roteador multi-robôs e o **Linux PC** é feita usando protocolo TCP/IP usando cabo (veja fotos da seção A.4.4.3).

### A.1.1 Setup: Robô manipulador UR3

Primeiro, ligamos o robô UR3 apertando o botão **POWER** que é mostrado em destaque por um retângulo vermelho na Figura A.2.



Figura A.2: Botão power

Depois de apertar o botão power, aparecerá a tela igual a Figura A.3

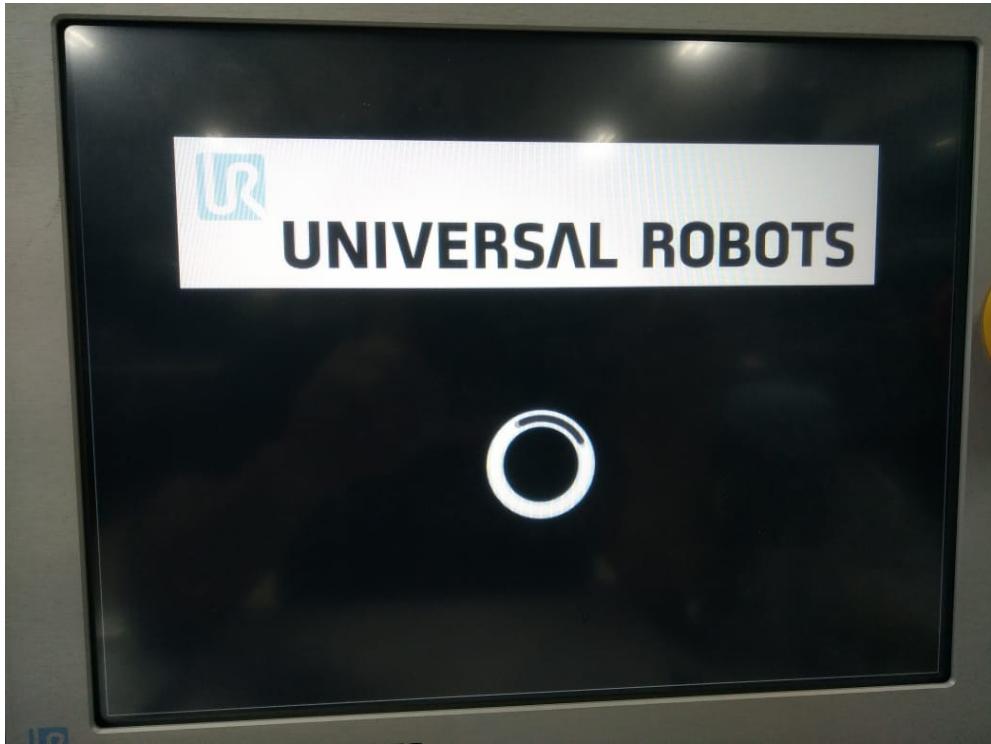


Figura A.3: Tela de carregamento

Depois que o sistema do robô carregar, aparecerá a tela igual a Figura A.4

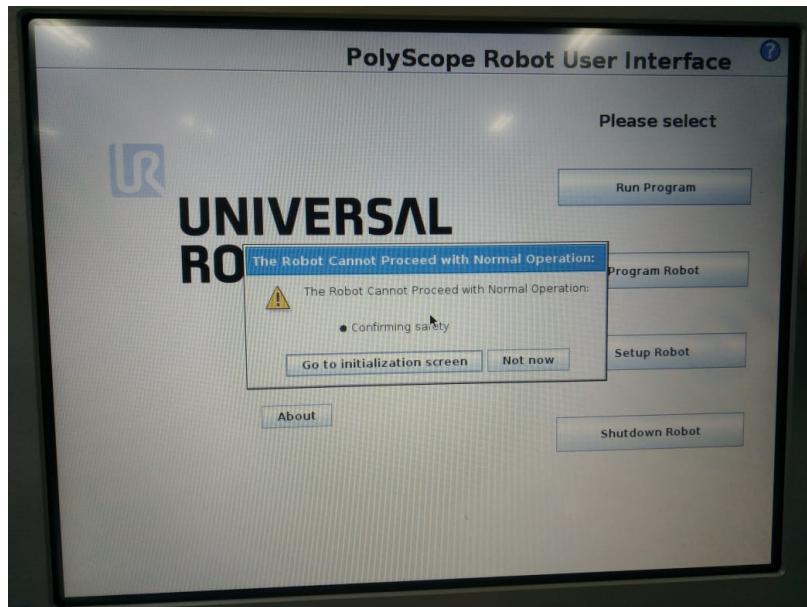


Figura A.4: Tela de apresentação

Agora, para ligar o braço robótico, clique em **Go to initialization screen**, que se encontra no centro da Figura A.4. Neste momento, temos a tela da Figura A.5.

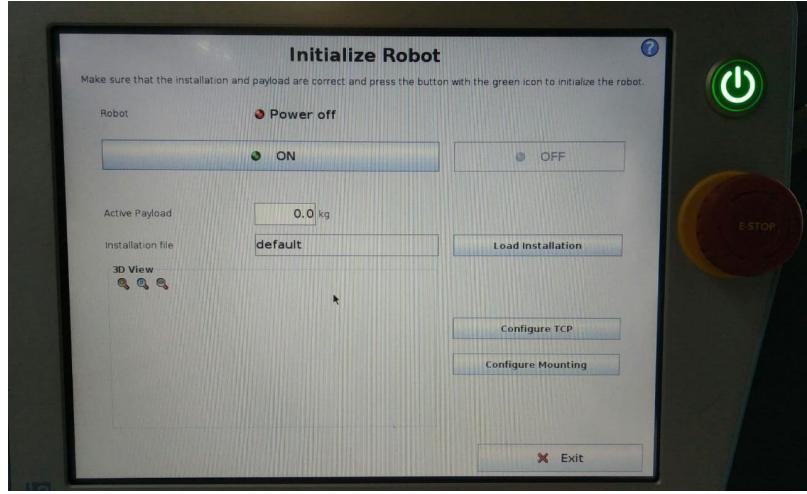


Figura A.5: Tela de inicialização

Para ligar o robô, pressione o botão **ON**. A tela que aparecerá será a da Figura A.6

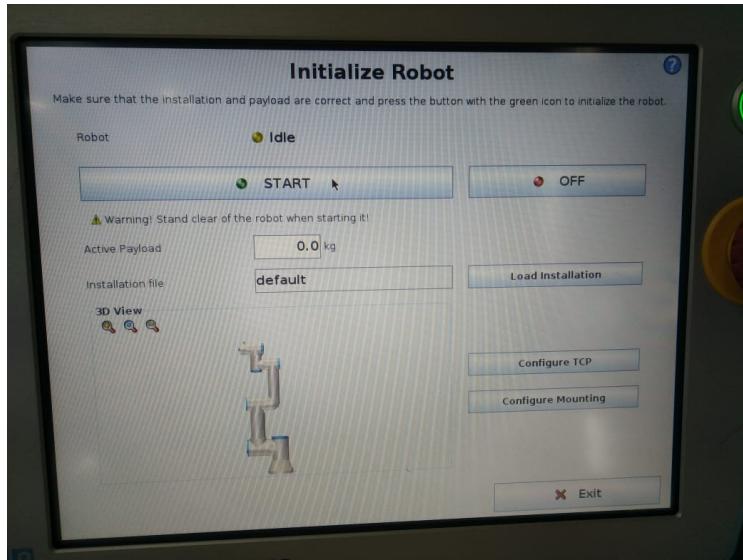


Figura A.6: Manipulador ligado. Destravamento: apertar START

Agora o robô está ligado. Para destravar as juntas aperte o botão **START** na Figura A.6.

**O robô fará um som de destravamento**, que é esperado, e entrará no modo de operação normal que pode ser visto na Figura A.7.



Figura A.7: OK

No canto inferior direito da tela (Figura A.7), aperte **OK**.

Agora, aparecerá uma nova tela (Figura A.8) como algumas opções de funcionalidades do robô. Deve-se clicar em **Program Robot** que está circulado em vermelho.



Figura A.8: Program Robot

Depois de apertar **Program Robot** aparecerá uma tela semelhante a Figura A.9. Aperte **Move**.

Pronto. Finalizamos nosso Setup do robô e a tela final é semelhante a Figura A.10.

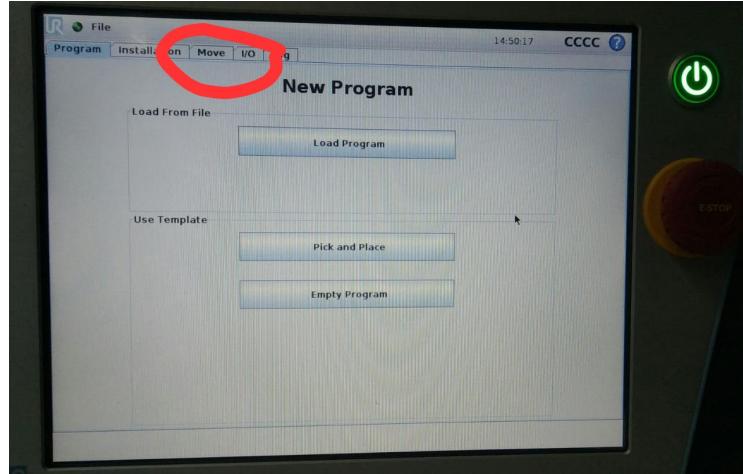


Figura A.9: MOVE

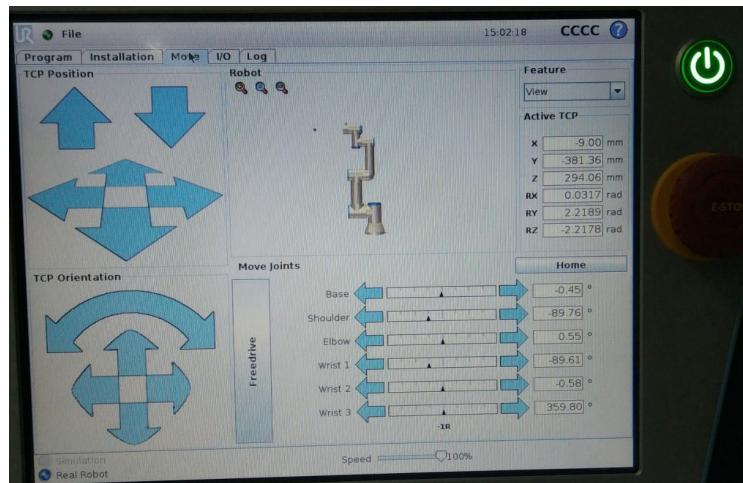


Figura A.10: Tela final para o setup do UR3

### A.1.2 Setup do Computador: Linux PC

Agora vamos ligar o computador Linux PC, que é usado para fechar uma malha de controle com o robô. O usuário do computador é **ur3** e senha **ur3**.

Abra a aplicação chamada **Terminator** circulada em vermelho na barra de tarefa do ubuntu 18.04, que é exatamente como a Figura A.11.

Depois de abrir o **Terminator**, vamos usar o comando **git clone** para baixar para o Linux PC o workspace `catkin_ur3_ws`, localizado no repositório github do LARA denominado lara-unb e localizado em <https://github.com/lara-unb>: digite no **Terminator** o comando<sup>1</sup>

```
git clone https://github.com/lara-unb/catkin_ur3_ws.git
```

---

<sup>1</sup>Em alguns terminais podem surgir na forma \$ git clone https://github.com/lara-unb/catkin\_ur3\_ws.git. Neste caso, retirar o \$

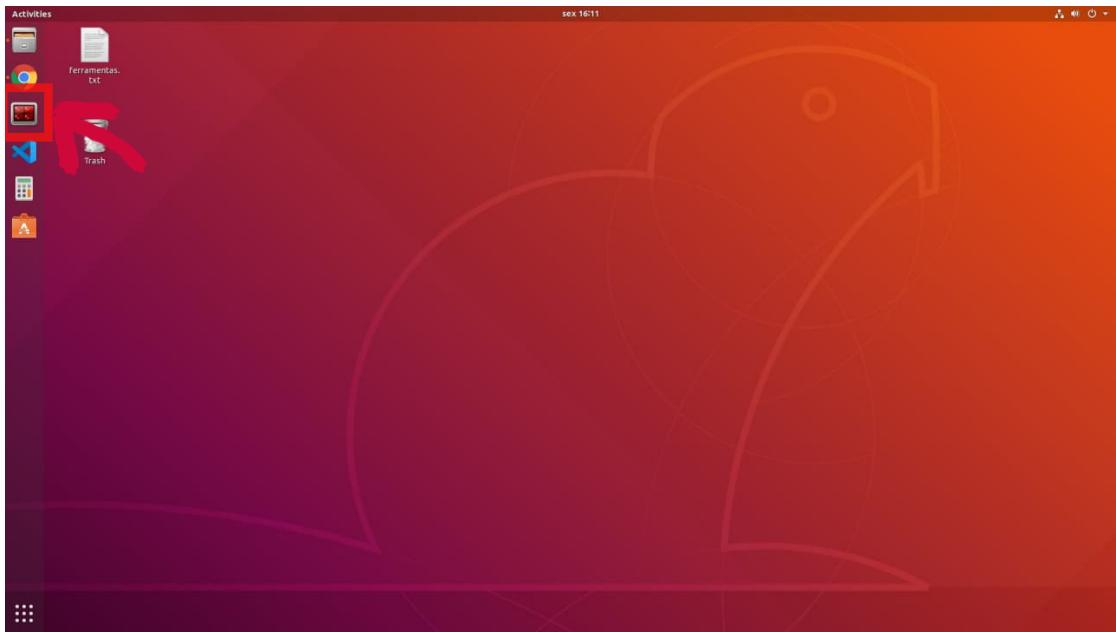


Figura A.11: Terminator

Na situação onde o **workspace cakin\_ur3\_ws** não exista no Linux PC, o workspace será baixado e deve surgir mensagens como mostrado na figura A.12.

```
ur3@ur3:~ ur3@ur3:~ 80x24
ur3@ur3:~ ur3@ur3:~ git clone https://github.com/lara-unb/catkin_ur3_ws.git
Cloning into 'catkin_ur3_ws'...
remote: Enumerating objects: 241, done.
remote: Counting objects: 100% (241/241), done.
remote: Compressing objects: 100% (152/152), done.
remote: Total 241 (delta 106), reused 201 (delta 69), pack-reused 0
Receiving objects: 100% (241/241), 69.46 KiB | 560.00 KiB/s, done.
Resolving deltas: 100% (106/106), done.
ur3@ur3:~ ur3@ur3:~
```

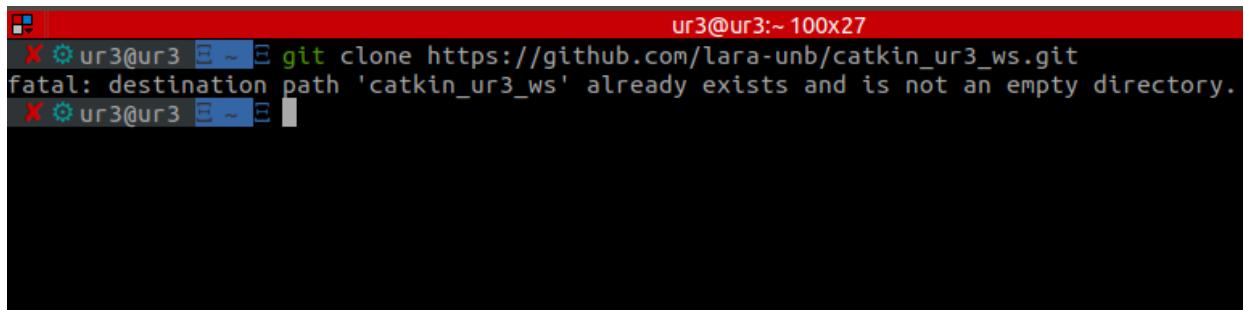
Figura A.12: git clone

Caso o **workspace cakin\_ur3\_ws** já exista no Linux PC, o usuário receberá uma mensagem de que o **workspace** já existe conforme Figura A.13

Com o workspace baixado, o passo seguinte consiste em entrar no diretório workspace usando o seguinte comando

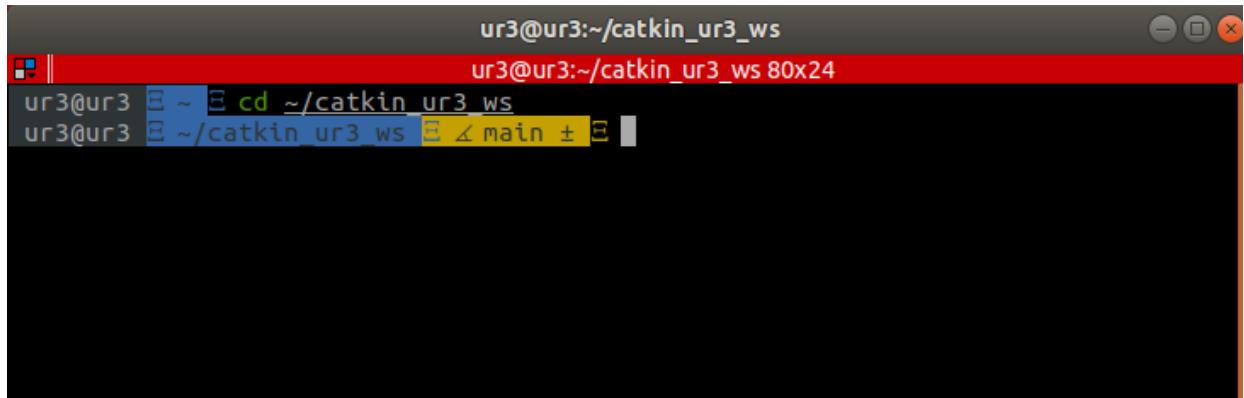
```
cd ~/catkin_ur3_ws
```

Com isso, a tela do Terminator deve ser parecida com a da Figura A.14



```
ur3@ur3:~ 100x27
✗ ⚡ ur3@ur3 ~ ⚡ git clone https://github.com/lara-unb/catkin_ur3_ws.git
fatal: destination path 'catkin_ur3_ws' already exists and is not an empty directory.
✗ ⚡ ur3@ur3 ~ ⚡
```

Figura A.13: Workspace existente



```
ur3@ur3:~/catkin_ur3_ws
ur3@ur3:~/catkin_ur3_ws 80x24
ur3@ur3 ~ ⚡ cd ~/catkin_ur3_ws
ur3@ur3 ~ /catkin_ur3_ws ⚡ ↵ main ± ⚡
```

Figura A.14: catkin\_ws

A seguir – para compilar os códigos da Interface de Comunicação, demos e mensagens e gerar os executáveis – use o comando:

**catkin\_make**

O usuário receberá uma mensagem como a da figura A.15 abaixo.

```

ur3@ur3: ~/UR3_interface/catkin_ur3
ur3@ur3: ~/UR3_interface/catkin_ur3 81x24
[ 21%] Built target geometry_msgs_generate_messages_nodejs
[ 21%] Built target std_msgs_generate_messages_nodejs
[ 21%] Built target control_msgs_generate_messages_nodejs
[ 21%] Built target geometry_msgs_generate_messages_lisp
[ 21%] Built target std_msgs_generate_messages_lisp
[ 21%] Built target control_msgs_generate_messages_lisp
[ 21%] Built target control_msgs_generate_messages_eus
[ 21%] Built target geometry_msgs_generate_messages_eus
[ 21%] Built target std_msgs_generate_messages_eus
[ 21%] Built target std_msgs_generate_messages_cpp
[ 21%] Built target geometry_msgs_generate_messages_cpp
[ 21%] Built target control_msgs_generate_messages_cpp
[ 25%] Building CXX object ur3/CMakeFiles/interface.dir/src/interface.cpp.o
[ 39%] Built target ur3_generate_messages_py
[ 50%] Built target ur3_generate_messages_nodejs
[ 60%] Built target ur3_generate_messages_lisp
[ 75%] Built target ur3_generate_messages_eus
[ 85%] Built target ur3_generate_messages_cpp
Scanning dependencies of target ur3_generate_messages
[ 85%] Built target ur3_generate_messages
[ 89%] Linking CXX executable /home/ur3/UR3_interface/catkin_ur3/devel/lib/ur3/interface
[100%] Built target interface

```

Figura A.15: Compilação

A parte de setup do computador para podermos usar o ROS está pronta. No próximo passo vamos aprender como lançar os pacotes ROS (Robot Operating System).

## A.2 Inicialização do ROS

Os arquivos executáveis já estão prontos (feito no último passo da seção anterior). Nesta seção, vamos apresentar a sequência de comandos necessários para que se lance a Interface de Comunicação.

Como primeiro passo, vamos abrir mais três seções do Terminator. Use Ctrl+shift+O para dividir o Terminator em seção horizontal e Ctrl+shift+E para dividir em seção vertical. Como resultado, teremos 4 (quatro) seções como mostrado na figura A.16 abaixo.

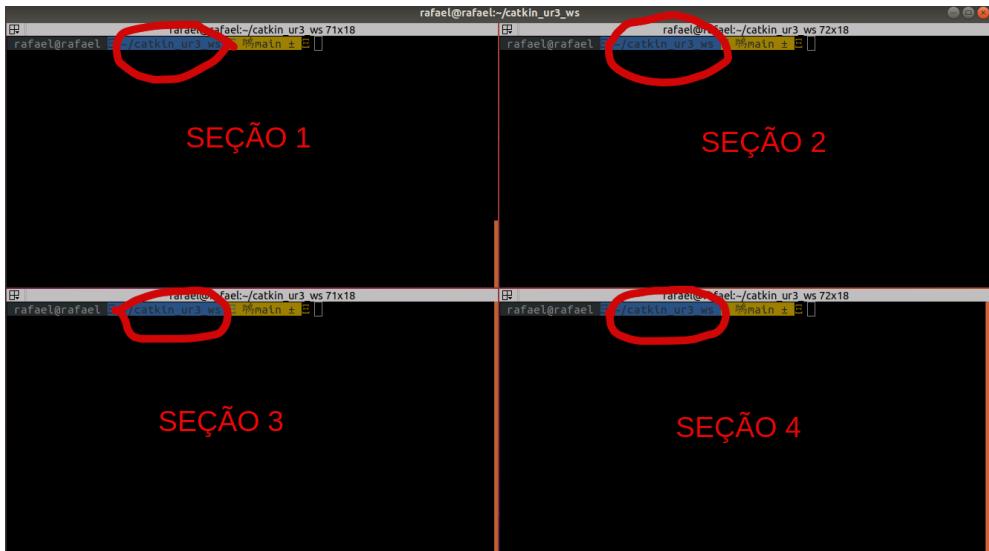


Figura A.16: Seções do Terminator

Cheque se as novas seções estão no mesmo workspace<sup>2</sup> conforme indicado na Figura A.16.

Uma vez que todas as seções estejam trabalhando no mesmo workspace catkin\_ur3\_ws, vamos rodar em cada seção, o comando

```
source devel/setup.zsh
```

para que **todas** as seções abertas sejam diretórios reconhecidos pelo ROS.

Para iniciar a Interface de Comunicação use o comando mostrado abaixo **na Seção 1**:

```
roslaunch ur3 real_ur3.launch
```

Depois de executar o comando acima, o usuário receberá uma resposta do Terminator, com a lista de parâmetros, as etapas de inicialização do UR3 e uma mensagem (em verde) dizendo que o nó **ROS Communication Interface** da Interface de Comunicação está rodando (vide Figura A.17)

---

<sup>2</sup>Caso alguma seção esteja em um workspace diferente dos mostrados na Figura A.16, execute nesta seção o seguinte comando `cd ~/catkin_ur3_ws`

```

ur3@ur3: ~ /catkin_ur3 ws [ ]<main + E roslaunch ur3 real_ur3.launch
... logging to /home/ur3/.ros/log/1bbf59f2-dfe8-11eb-9369-bcaec59c8518/roslaunch-ur3-9883.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://ur3:40909/
SUMMARY
=====
PARAMETERS
  * /rosdistro: melodic
  * /rosversion: 1.14.11
  * /ur3/max_ref_vel/joint_0: 1.5
  * /ur3/max_ref_vel/joint_1: 1.5
  * /ur3/max_ref_vel/joint_2: 1.5
  * /ur3/max_ref_vel/joint_3: 1.5
  * /ur3/max_ref_vel/joint_4: 1.5
  * /ur3/max_ref_vel/joint_5: 1.5
  * /ur3/max_torque/joint_0: 2.0
  * /ur3/max_torque/joint_1: 2.0
  * /ur3/max_torque/joint_2: 2.0
  * /ur3/max_torque/joint_3: 2.0
  * /ur3/max_torque/joint_4: 2.0
  * /ur3/max_torque/joint_5: 2.0
  * /ur3/robot_ip: 192.168.1.56
  * /ur3/robot_port: 30003

NODES
  /
    ur3 (ur3/interface)

auto-starting new master
process[master]: started with pid [9893]
ROS_MASTER_URI=http://localhost:11311

setting /run_id to 1bbf59f2-dfe8-11eb-9369-bcaec59c8518
process[rosout-1]: started with pid [9904]
started core service [/rosout]
process[ur3-2]: started with pid [9908]
[ WARN] [1625747356.650268452]: Init Interface ur3
[ WARN] [1625747356.660381457]: Waiting for ur3 response ...
[ WARN] [1625747356.662069329]: Send script to ur3 ...
[ WARN] [1625747356.686464777]: ur3 is setted in ip 192.168.1.56
[ WARN] [1625747356.686540369]: ur3 is setted in port 30003
[UR3] [1625747362.890395000]: ur3 node is running ;)

```

Figura A.17: Log do Launch ur3

Neste ponto, o nó ROS **Communication Interface**, que permite a intercomunicação entre o Linux PC e o Controller, está rodando. Com isso, a Interface de Comunicação está ativa, ou seja, capturando os dados dos estados das juntas do robô UR3, publicando estes dados no tópico **/ur3/arm** e esperando uma referência de velocidade no tópico **/ur3/ref\_vel**.

### A.3 Demostraçāo do experimento 1

Este experimento consiste em fazer um movimento com 1 minuto de duração da junta M32 (**Elbow Joint**). O movimento é periódico entre as posições 0° e 45° para a junta M32 e é obtido usando-se referência de velocidade.

Para esta pequena demonstração, foi feito, em linguagem **Python**, um nó ROS chamado **exp1** para publicar as referências de velocidade para as juntas do robô<sup>3</sup>.

Para lançar o experimento 1, ou seja, executar o nó ROS **exp1**, vá para a seção 2 no Terminator e execute o seguinte comando

```
roslaunch demos_ur3 exp1.launch
```

Para iniciar a publicação das velocidades de referência para o robô, foi feito um serviço ROS chamado de **/demos\_ur3/start\_exp1** e para chamar esse serviço basta rodar o seguinte co-

---

<sup>3</sup>O arquivo que representa o nó ROS é chamado de **demo\_exp1.py** e está armazenado no pacote **demos\_ur3**.

mando na seção 3 do Terminator com *flag* **data** com o valor **true** (Olhe para robô depois que executar o comando, pois ele começará a fazer o movimento do experimento):

```
rosservice call /demos_ur3/start_exp1 "data: true"
```

Depois disso é esperado que o robô UR3 faça um movimento repetitivo parecido com o movimento do seguinte vídeo que está postado no youtube:

[https://youtu.be/pkN1GS0wX\\_o](https://youtu.be/pkN1GS0wX_o)

Caso queira visualizar os dados em tempo real dos estados da junta, citados em 2.2.3.2, siga os passos do experimento de visualização de dados no Apêndice C.

Caso queira parar o experimento, basta chamar o mesmo serviço com *flag* **data** com o valor **false**, na Seção 3, ou espere o experimento acabar.

```
rosservice call /demos_ur3/start_exp1 "data: false"
```

Caso queira repetir o experimento, execute, na seção 4, o comando abaixo na para coloca o UR3 na posição de descanso.

```
rosservice call /ur3/reset "{}"
```

Depois rode o comando abaixo, na seção 3, para iniciar o experimento.

```
rosservice call /demos_ur3/start_exp1 "data: true"
```

Com a execução do nó **exp1** encerrado, caso desejado, pode-se mover as juntas do UR3 pressionando as setas, demarcados com retângulo vermelho representado na Figura A.18, apresentadas na tela touchscreen da interface **Polyscope**. Porém a conexão da caixa de controle do UR3 com a Interface Comunicação do Linux PC será desfeita e aparecerá uma mensagem como na figura abaixo.

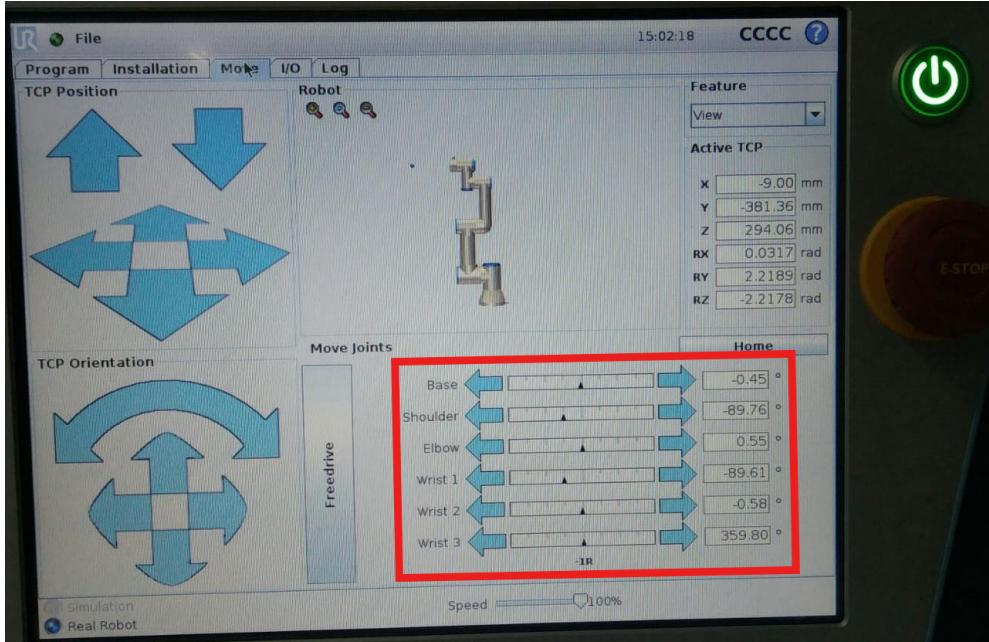


Figura A.18: Setas do Polyscope

```

process[ur3-2]: started with pid [14040]
[ WARN] [1626178241.983887313]: Init Interface ur3
[ WARN] [1626178241.996398715]: Wainting for ur3 response ...
[ WARN] [1626178241.997812734]: Send script to ur3 ...
[ WARN] [1626178242.021061366]: ur3 is setted in ip 192.168.1.56
[ WARN] [1626178242.021148202]: ur3 is setted in port 30003
[ INFO] [1626178247.021900740]: Opening Socket Communication ...
[ UR3 ] [1626178248.217183000]: Communication Interface node is running ;)
[ INFO] [1626178478.285735025]: Closing Socket Communication ...
[ WARN] [1626178479.287904061]: ur3 is setted in ip 192.168.1.56
[ WARN] [1626178479.287987280]: ur3 is setted in port 30003
[ INFO] [1626178484.288815973]: Opening Socket Communication ...
[ INFO] [1626178485.481423549]: UR3 was resetted !
[ UR3 ] [1626178485.481497000]: Communication Interface node is running ;)
[ INFO] [1626178532.917976842]: Closing Socket Communication ...
[ WARN] [1626178533.920110138]: ur3 is setted in ip 192.168.1.56
[ WARN] [1626178533.920184362]: ur3 is setted in port 30003
[ INFO] [1626178538.920799657]: Opening Socket Communication ...
[ INFO] [1626178540.119212121]: UR3 was resetted !
[ UR3 ] [1626178540.119275000]: Communication Interface node is running ;)

[ ERROR] [1626178577.114141147]: The connection with ur3 was broken. Please, reset the Interface Communication
[ INFO] [1626178635.598781723]: ur3 is setted in ip 192.168.1.56
[ WARN] [1626178635.598858587]: ur3 is setted in port 30003
[ INFO] [1626178640.599576383]: Opening Socket Communication ...

```

Figura A.19: Mensagem de erro

Para reconectar a Interface de comunicação no **Linux PC** basta rodar o seguinte comando

`rosservice call /ur3/reset "{}"`

## A.4 Desligamento do sistema

### A.4.1 Certificar se o robô está na posição HOME

Caso o braço manipulador não esteja na posição HOME, como mostra a Figura A.20, use o comando de **reset** em alguma seção disponível do Terminator.

Comando **reset** mostrado logo abaixo para por o UR3 na posição HOME.

```
rosservice call /ur3/reset "{}"
```



Figura A.20: Braço manipulador UR3 na posição HOME

### A.4.2 Desligamento do polyscope

Simplesmente apertar o botão on-off do Polyscope Figura A.21 que é mostrada logo abaixo.

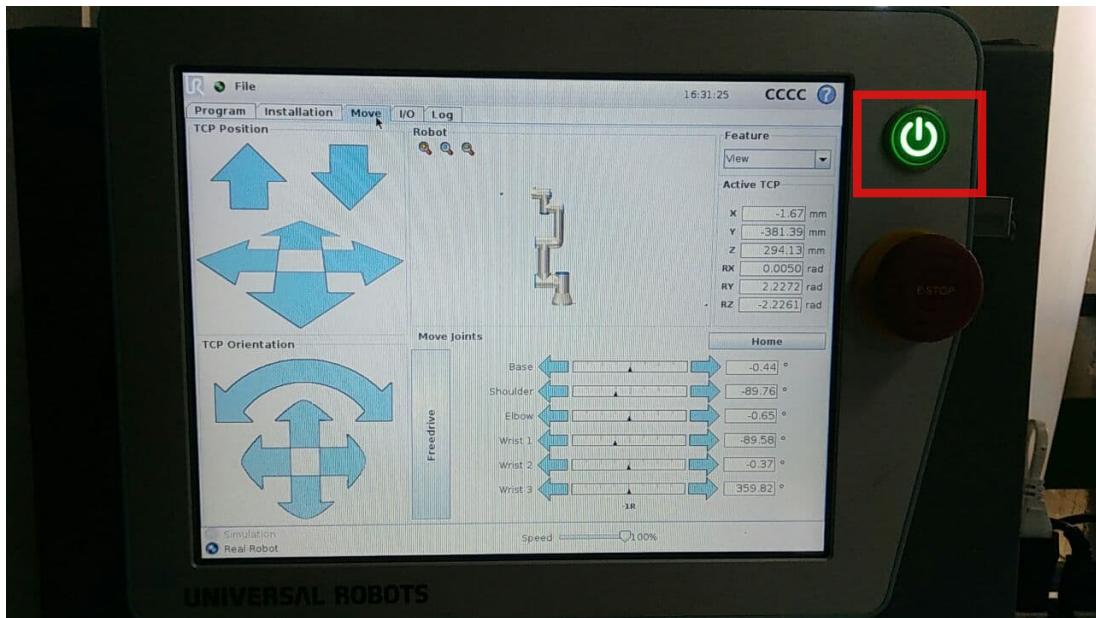


Figura A.21: Botão on-off no desligamento

Aparecerá um POPUP chamado *shutdown* similar ao da Figura A.22.

Pressione o botão com a marcação retangular em vermelho chamado **Power off**.

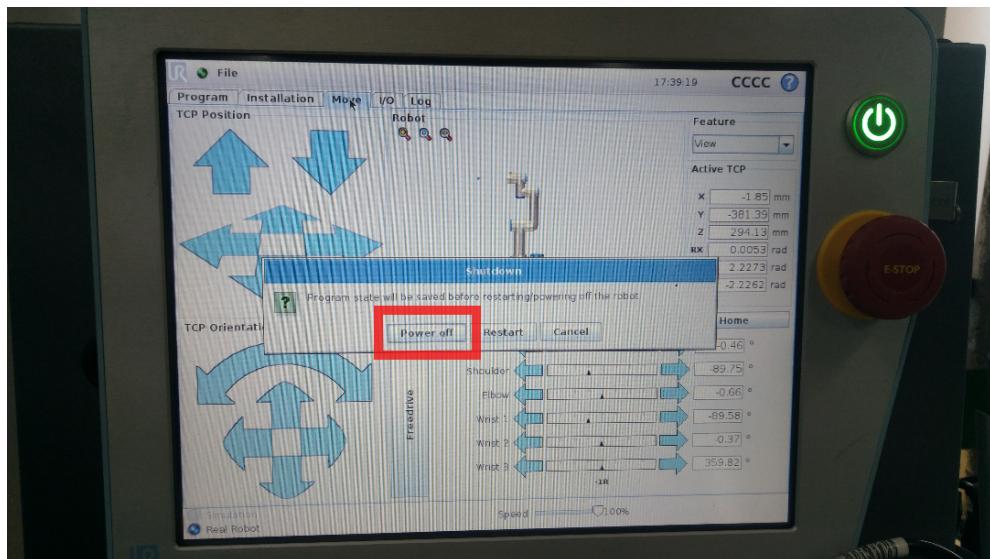


Figura A.22: POPUP shutdown

Feito isso o robô será desligado.

#### A.4.3 Desligamento da Interface de Comunicação Linux PC

A.4.3.1 Na seção 1 (que roda a interface de comunicação) usar o comando Ctrl+C pelo teclado.

A.4.3.2 Fechar o Terminator como qualquer outro aplicativo Lnxux.

#### A.4.4 IMPORTANTE!!

Antes de ir embora, manter o sistema sempre como na foto abaixo.

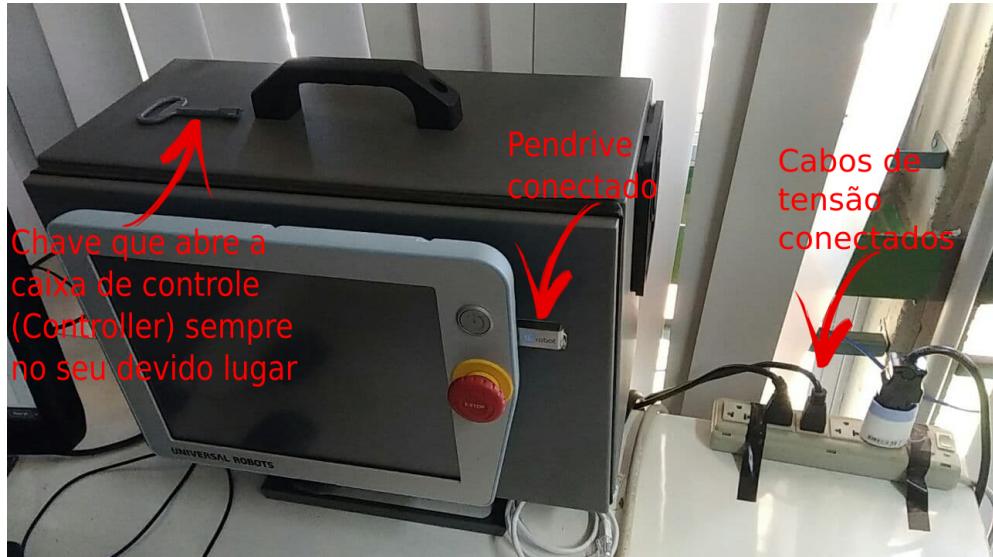


Figura A.23: Cabos de tensão, pendrive e chave

##### A.4.4.1 Não retirar o pendrive

O pendrive fornece os drivers da garra. O fabricante recomenda deixar o pendrive sempre conectado ao polyscope

##### A.4.4.2 Não retire a chave do lugar

A chave é utilizada por usuários avançados para acessar a caixa de controle (Controller). Não abra a caixa de controle a menos que saiba exatamente o que está fazendo.

##### A.4.4.3 Manter os cabos no lugar

Manter os cabos de rede como nas fotos A.24, A.25 e A.26



Figura A.24: Cabo de rede do linux PC



Figura A.25: Cabo de rede do Controller

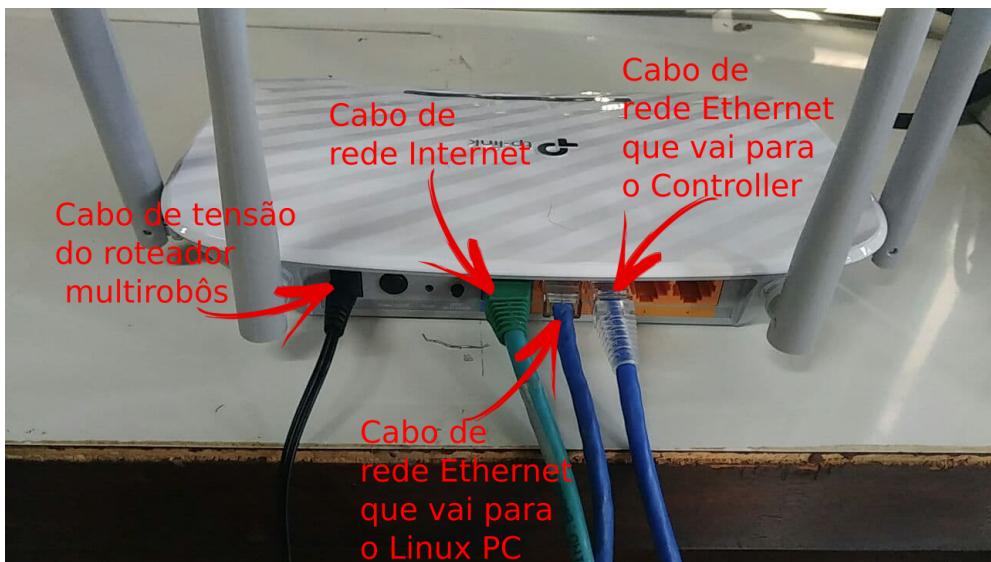


Figura A.26: Cabos de rede do roteador

FIM!