Commander un robot UR avec le driver ROS2

Installation d'Ubuntu avec des capacités temps-réel

Pour un usage basique, un Ubuntu (ou Linux Mint) classique permet de piloter le robot :

- Installer Ubuntu LTS 22.04
-

Pour éviter des instabilités il est conseillé d'installer un noyau Linux avec des capacités temps-réel (PREEMPT_RT kernel). En particulier avec un robot de la Série-E, la fréquence de contrôle plus élevée peut entraîner des trajectoires non fluides sans noyau temps-réel.

- Vérifier qu'il reste au moins 25Go d'espace disque
- On se place dans un dossier pour la compilation

```
mkdir -p ~/rt_kernel_build
cd ~/rt_kernel_build
```

```
tar xf linux-4.14.139.tar
cd linux-4.14.139
xzcat ../patch-4.14.139-rt66.patch.xz | patch -p1
make oldconfig
```

Récupérer les sources du noyau temps-réel

- Regarder les versions LTS du noyau Linux : https://www.kernel.org/category/releases.html

- Regarder la version actuelle et les versions proposées par Ubuntu : Update Manager -> view -> Linux Kernel
- Regarder les versions maintenues activement du noyau PREEMPT_RT -> https://wiki.linuxfoundation.org/realtime/preempt_rt_versions
- Choisir la version du noyau Linux PREEMPT_RT maintenue activement correspondant à une version LTS et proposée par Ubuntu
- Exemple pour un Lenovo Thinkpad T480
  - Actuellement installé : 5.19 (non-LTS)
  - Proposé : 5.19 et 5.15 (LTS)
  - On choisit 5.15 car LTS et activement maintenu en PREEMPT_RT
- Récupérer les sources pour `5.15.107-rt62`

```
wget https://cdn.kernel.org/pub/linux/kernel/projects/rt/5.15/patch-5.15.107-rt62.patch.xz
wget https://cdn.kernel.org/pub/linux/kernel/projects/rt/5.15/patch-5.15.107-rt62.patch.sign
wget https://www.kernel.org/pub/linux/kernel/v5.x/linux-5.15.107.tar.xz
wget https://www.kernel.org/pub/linux/kernel/v5.x/linux-5.15.107.tar.sign
```

- Décompresser les fichiers téléchargés

```
xz -dk patch-5.15.107-rt62.patch.xz
xz -d linux-5.15.107.tar.xz
```

- Importer les clés publiques des développeurs du noyau et du patch (5.15-rt Joseph Salisbury 2026-10)

```
gpg2 --locate-keys torvalds@kernel.org gregkh@kernel.org
gpg2 --keyserver hkp://keys.gnupg.net --search-keys salisbury
```

- Vérifier l'intégrité des fichiers source

```
gpg2 --verify linux-5.15.107.tar.sign
```

> gpg: Good signature from "Greg Kroah-Hartman <gregkh@kernel.org>" [unknown]
> gpg: WARNING: This key is not certified with a trusted signature!
> gpg: There is no indication that the signature belongs to the owner.

```
gpg2 --verify patch-5.15.107-rt62.patch.sign
```

```
gpg: Good signature from "Tom Zanussi <tom.zanussi@linux.intel.com>"
[unknown]
gpg:                 aka "Tom Zanussi <zanussi@kernel.org>" [unknown]
gpg:                 aka "Tom Zanussi <tzanussi@gmail.com>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
```

Compiler le noyau temps réel

- Extraire l'archive tar, appliquer la patch et configurer le noyau temps-réel

```
tar xf linux-5.15.107.tar
cd linux-5.15.107
xzcat ../patch-5.15.107-rt62.patch.xz | patch -p1
make oldconfig
```

- Choisir l'option `4. Fully Preemptible Kernel (RT) (PREEMPT_RT_FULL) (NEW)` pour l'option `Preemption Model`

```
Preemption Model
  1. No Forced Preemption (Server) (PREEMPT_NONE)
> 2. Voluntary Kernel Preemption (Desktop) (PREEMPT_VOLUNTARY)
  3. Preemptible Kernel (Low-Latency Desktop) (PREEMPT)
  4. Fully Preemptible Kernel (RT) (PREEMPT_RT_FULL) (NEW)
choice[1-4]: 4
```

- Compiler le noyau

```
make -j `getconf _NPROCESSORS_ONLN` deb-pkg
```

- Après la compilation, installer les paquets `linux-headers` et `linux-image` dans le dossier parent (pas les paquets `-dbg`)

```
sudo apt install ../linux-headers-5.15.107-rt62_*.deb ../linux-image-5.15.107-rt62_*.deb
```

Définir les permissions utilisateur pour exécuter des tâches temps-réel

- Le Driver ROS2 va planifier des threads avec les permissions de votre utilisateur. Il faut donc autoriser votre utilisateur à utiliser lancer des threads avec une priorité temps-réel. On créé le groupe des utilisateurs temps-réel et on y ajoute l'utilisateur :

```
sudo groupadd realtime
sudo usermod -aG realtime $(whoami)
```

- S'assurer que `/etc/security/limits.conf` contient :

```
@realtime soft rtprio 99
@realtime soft priority 99
@realtime soft memlock 102400
@realtime hard rtprio 99
@realtime hard priority 99
@realtime hard memlock 102400
```

Note: Pour que ces changements soient pris en compte il faut se déconnecter et se reconncter. On redémarrera plus tard.

https://github.com/HowardWhile/Ubunutu22.04-RT-Kernel

Configurer GRUB pour toujours booter le noyau temps-réel

- Lister les noyaux disponibles

```
awk -F\' '/menuentry|submenu / {print $1 $2}' /boot/grub/grub.cfg
```

> menuentry Ubuntu
> submenu Advanced options for Ubuntu
>
> menuentry Ubuntu, with Linux 5.15.107-rt62
> menuentry Ubuntu, with Linux 5.15.107-rt62 (recovery mode)

- Définir le noyau `5.15.107-rt62` par défaut avec un pattern `"submenu_name>entry_name"`

```
sudo sed -i 's/^GRUB_DEFAULT=.*/GRUB_DEFAULT="Advanced options for Ubuntu>Ubuntu, with Linux
5.15.107-rt62"/' /etc/default/grub
$ sudo update-grub
```

Vérification de la capacité de préemption temps-réel

```
uname -v |  cut -d" " -f1-4
```

```
#1 SMP PREEMPT RT
```

Optionnel : Désactiver le CPU speed scaling

Les threads planifiés en temps-réel s'exécutent sans problème. Cependant, des composants externes tels que les systèmes de visual servoing, non planifiés pour le temps réel peuvent être interrompus par les fonctionnalités d'économie d'énergie des processeurs récents qui changent leur fréquence d'horloge de manière dynamique.

- Pour vérifier et modifier les modes d'économie d'énergie :

```
sudo apt install cpufrequtils
cpufreq-info
```

```
current CPU frequency is XXX MhZ
```

- Désactiver le changement de fréquence automatique :

```
sudo systemctl disable ondemand
sudo systemctl enable cpufrequtils
sudo sh -c 'echo "GOVERNOR=performance" > /etc/default/cpufrequtils'
sudo systemctl daemon-reload && sudo systemctl restart cpufrequtils
```

Configuration du robot UR

https://docs.ros.org/en/ros2_packages/rolling/api/ur_robot_driver/installation/robot_setup.html

Récupération de la calibration usine

Les robots sont calibrés en usine. Pour que les calculs cinématiques effectués dans ROS soient exacts, il faut récupérer les données de calibration. Sinon la précision des trajectoires envoyées depuis ROS et exécutées sur le robot risquent d'être de l'ordre du centimètre (au lieu du dixième de millimètre en temps normal).

https://docs.ros.org/en/ros2_packages/rolling/api/ur_robot_driver/installation/robot_setup.html#extract-calibration-information

---

Auteur: Gauthier Hentz, sur le wiki de l'innovation de l'IUT de Haguenau

Revision #5
Created 25 April 2023 14:18:27 by admin_idf
Updated 31 January 2024 13:10:57 by admin_idf