Raspberry Pi4 (8GB RAM): ROS MelodicでRoombaを動かそう！

demura.net/education/lecture/18594.html

August 15, 2020

Raspberry Pi4 (8GB RAM)、ROS MelodicでRoombaを動かすまでの記事。

更新：ソースコードの変更で間違と不足箇所があったので赤字で訂正しました。ご指摘頂いた玉川大学の岡田先生ありがとうございました(2020-9-6)。

環　境

- Raspberry Pi4 Model B, RAM 8GB
- Xubuntu18.04.4 LTS
  インストール方法はこの記事を参照。なお、**この方法（イメージファイル）を使わない場合は、以下の説明どおり実施してもうまく動かない場合があるかもしれません。**
- ROS Melodic
  インストール方法はこの記事を参照。
- Roomba 600シリーズ
  本記事では3万円弱で購入できる606で実行した。本記事で使用するCreate_autonomyパッケージは800シリーズまで対応しているので、そのシリーズまで大丈夫だと思うが未検証。**なお、900シリーズ以上はRoombaからシリアルポートがなくなり、Create_autonomyパッケージも対応していないため使えないので注意。**
- Roomba通信ケーブル
  Pi4とRoombaを接続するケーブル。以下を参考に自作するか、ここから購入する。本記事では2011年に購入したルンバ研究開発キットに付属していた通信ケーブルを使っている。
  - 自作：ルンバの制御用USBシリアルケーブルを綺麗に作りたいときのメモ、cvl-robogt's diary
  - 購入：USB to DINシリアルケーブルfor Roomba、作成2　(5431円)
- モバイルバッテリ
  10000mAhだとPi4の負荷により変わるが数時間は持つ。今、Pi4でこのブログを書いているが電流は0.8Aほど。
- SONY DualShock4 (DS4)コントローラ

インストール

- 準備
  - $ `sudo apt install python-catkin-tools`
  - `$ sudo apt install -y libqt4-dev`
- ワークスペースの生成
  - $ `mkdir -p ~/catkin_ws/src`
  - $ `cd catkin_ws`
  - $ `catkin init`
- create_autonomy
  - カナダのSimon Fraser University, Autonomy Lab.のJacob Perronさんのフォークしたi Robot社のRoomba, Create2用のROSドライバー create_autonomyを使う。
    RoboticaUtnFrba/create_autonomy
  - ダウンロード
    - `$ cd ~/catkin_ws/src`
    - `$ git clone https://github.com/RoboticaUtnFrba/create_autonomy.git`
    - `$ git clone https://github.com/RoboticaUtnFrba/libcreate.git`
  - 依存関係ファイルのインストール
    - RTIMULib (IMU用のリアルタイムライブラリ)のインストール
      - `$ cd ~/catkin_ws`
      - `$ ./src/create_autonomy/sensors/ca_imu/scripts/install_rtimulib.sh`
    - 他のファイルのインストール
      - `$ cd ~/catkin_ws`
      - `$ sudo apt install -y python3-vcstool`
      - `$ vcs import src < src/create_autonomy/dependencies.repos`
      - `$ rosdep update`
      - `$ rosdep install --from-paths src -yi`
  - ソースコードの変更
    - viso2(Visual Odometryライブラリ）関連のソースコードをそのままではビルドできないので次のように改変するか~/catkin_ws/src/viso2以下のディレクトリを削除する。なお、この改変作業はRaspberryPi4のARM CPU用なので、Intel CPUを搭載したPCにこの作業をしてはいけない。
      - ~/catkin_ws/src/viso2/libviso2/CMakeLists.txtの10行目と
        ~/catkin_ws/src/viso2/viso2_ros/CMakeLists.txt の35行目を次のように変更する。
        - 変更前： `SET(CMAKE_CXX_FLAGS -mfpu=neon)`
        - 変更後： `SET(CMAKE_CXX_FLAGS -march=armv8-a+simd)`
      - ~/catkin_ws/src/viso2/libviso2/libviso2/src/filter.hの25行目とmatcher.hの30行目に#if defined(__ARM_NEON__)がある。その１行上に以下の行を挿入する。
        `#define __ARM_NEON__`
    - create_driver：ルンバを接続すると”[CREATE] Unknown mode detected”と大量のエラーメッセージが出るが動作には特に問題はない。心臓に悪いのこのエラーメッセージを吐かないようにする。なお、createでは問題はないようだ。
      ~/catkin_ws/src/create_autonomy/ca_driver/src/create_driver.cppの678行目を以下のように行の先頭に//をつけてコメントアウトする。
      `// ROS_ERROR("[CREATE] Unknown mode detected")`
  - ワークスペースのビルド
    - `$ cd ~/catkin_ws`
    - 次のコマンドでビルドする。私の環境では約28分かかった。
      `$ catkin build -j $(nproc) -DCMAKE_BUILD_TYPE=Release -DARM_CROSS_COMPILATION=ON`

- セットアップ
  - ~/.bashrcの最後に以下の２行があるか確認する。ない場合は追加する。
    ```
    source /opt/ros/melodic/setup.bash
    source ~/catkin_ws/devel/setup.bash
    ```
  - 以下のコマンドを実行して設定を反映させる。
    ```
    $  source ~/.bashrc
    ```

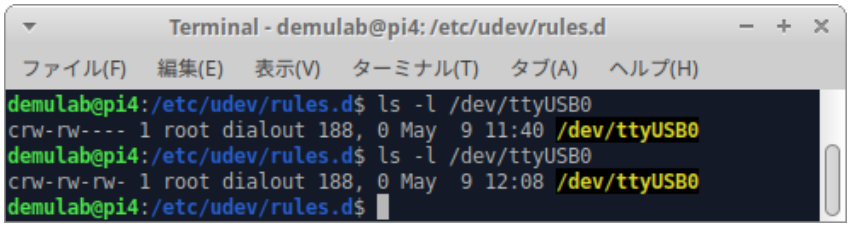
設　定

udev

- Linuxはデバイスをファイルとして扱い、Pi4とRoombaを通信するときは/dev/ttyUSB0。USB-Serial変換ICでttyUSBではなくttyACMの場合もあるり、ttyUSBの後の番号は他の接続機器により0ではない場合もある。一般ユーザはデバイスファイルを扱えないのでパーミッションを変更する必要があるがUSBを接続する度に変更するのは面倒なのでudevを設定する。
- まず、Pi4とroombaを通信ケーブルで接続する。
- 次に、以下のコマンドでデバイスを確認する。本記事で使用しているUSB-Serial変換用ICはFuture Technology Devices Internatinal社のFT232だとわかる。Future Technologyの左の数字0403はベンダーID、6001はプロダクトIDなのでメモしておく。

  $ lsusb

- /etc/udev/rules.d/77-roomba.rulesというファイルをつくり、以下を保存する。ここで、ベンダーIDは上でメモったベンダーID、プロダクトIDも同様な数字を入れる。この例では、ベンダーIDは0403、プロダクトIDは6001になる。
  ```
  KERNEL=="ttyUSB*", ATTRS{idVendor}=="ベンダーID", ATTRS{idProduct}=="プロダクトID", GROUP="dialout", MODE="0666"
  ```
- Room通信ケーブルを抜き差しするとudevの設定が反映される。確認はls -l デバイスファイル名でパーミッション設定を確認する。この例だと、上がケーブルを抜き差しする前で、下が後。すべてのユーザに対してrw（読み書き）可能となっていることがわかる。



- USBの設定

  create_autonomyのサンプルではUSBのデバイス名を/dev/roombaにしているので、以下のコマンドとシェルスクリプトでその設定を行う。
  - `$ sudo usermod -a -G dialout $USER`
  - `$ source ~/catkin_ws/src/create_autonomy/ca_bringup/scripts/udev/create_rules.sh`
- では、次にいよいよRoombaを動かします。Pi4をシャットダウンして電源を外しましょう。

Roombaを動かそう！

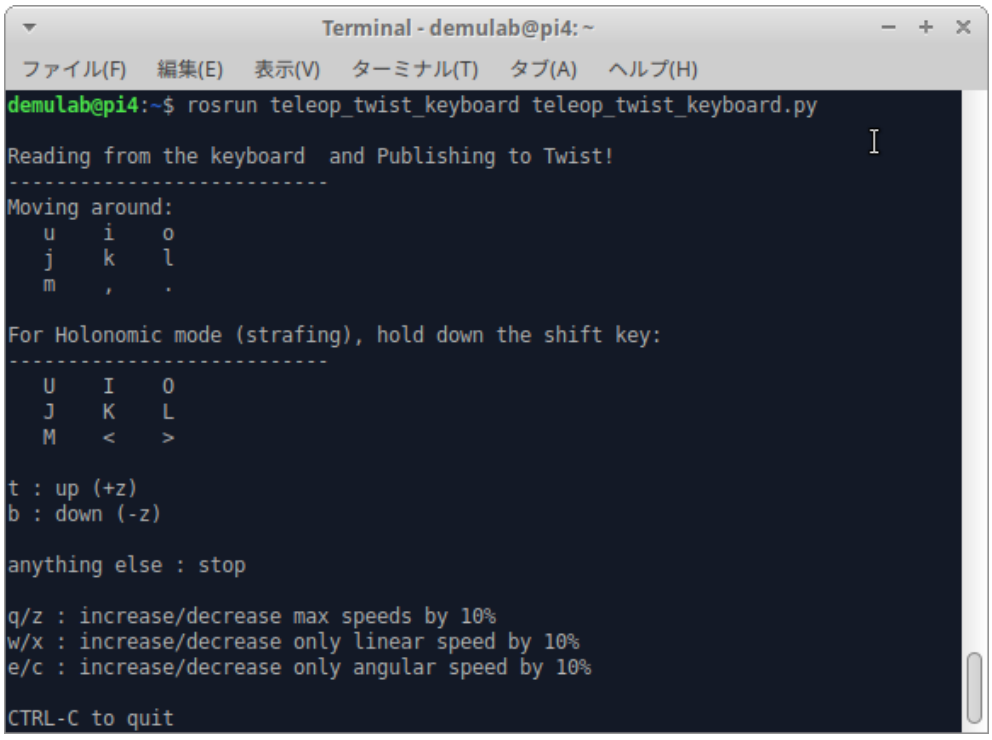
- Roombaと通信ケーブルを接続してから、Roombaの電源を入れる。
- Roombaを動かすために次のlaunchファイルを起動する。
  `$  roslaunch ca_driver create_2.launch`
- 前進

  ROSではトピック/create1/cmd_velに速度指令値を設定して、配信(pub)するだけでロボットは動く。ここで、geometry_msgs/Twistは速度指令値の型、'{linear: {x: 0.2}, angular: {z: 0.0}}'が指令値で、linearは並進速度(x軸方向に0.2m/s)、angularは角速度(z軸まわりに0 rad/s)、オプション-rは配信する周期[Hz]デフォルトは10Hzなので動きがガタつくのでここでは100Hz、10ms周期にしている。

  ```
  $ rostopic pub /create1/cmd_vel geometry_msgs/Twist '{linear: {x: 0.2}, angular: {z: 0.0}}' -r 100
  ```

  Teleopで動かそう
  - いちいちコマンドを打っていてはRoombaをうまく制御できないので、キーボードを使って動かしてみる。
  - キーボード用とコントローラ用の2つteleopパッケージをインストールする。
    - `$ sudo apt install ros-melodic-teleop-twist-keyboard`
    - `$ sudo apt install ros-melodic-teleop-twist-joy`
  - 実行

    次のコマンドを実行する
    - `$ rosrun teleop_twist_keyboard teleop_twist_keyboard.py cmd_vel:=/create1/cmd_vel`
    - 以下の画面になる。キーボードのuiojklm,,.キーで操作できる。止めるときは'q'キーを押す。



DS4コントローラのROSパッケージps4_rosの導入

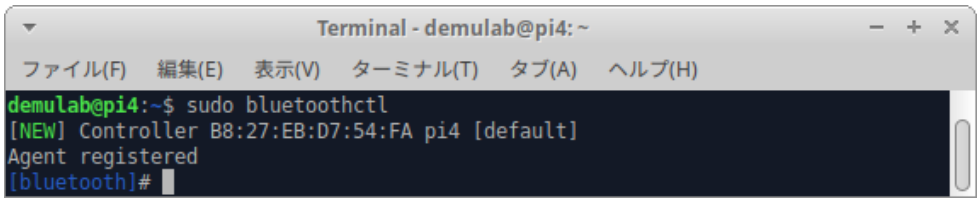
- サイト

  https://github.com/solbach/ps4-ros
- インストール
  - `$ cd ~/catkin_ws/src`
  - `$ git clone https://github.com/solbach/ps4-ros.git`

- launchファイルの変更
  ~/catkin_ws/src/ps4-ros/launch/ps4.launchの下から３~5行目を以下のように変更する。
  Roombaに合うように最大速度とTopic名を変更する。
  <param name="scale_linear" value="0.5"/>
  <param name="scale_angular" value="0.25"/>
  <param name="pub_topic" value="/create1/cmd_vel"/>

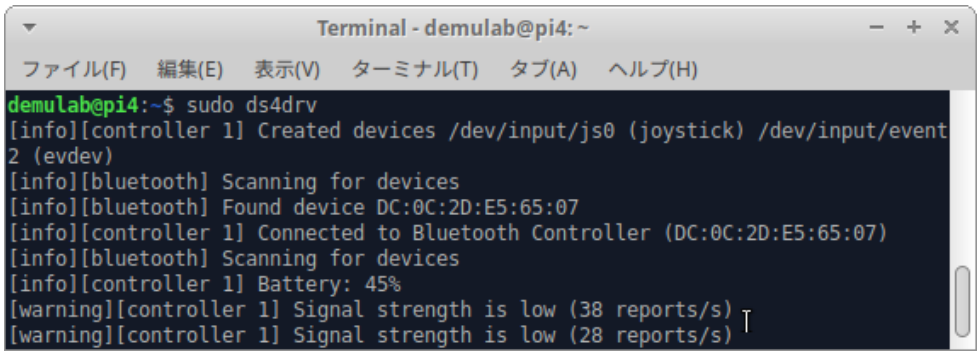
Bluetoothの設定

- Pi4のUbuntu18.04の場合はそのままではBluetoothが使えないので必要なパッケージをインストールする。なお、Futaka TatsuyaさんのQiitaを参考にした。なお、RAM8GBで使用したJames Chamber氏の非公式版Xubuntu desktopのイメージではpi-bluetoothをインストールするとエラになるので省いた。
  ARMのUbuntu Server 18.04でBluetoothを使う(for Raspberry Pi), Futaka Tatsuya
- パッケージのインストール
  - `$ sudo apt install bluetooth bluez`
  - `$ sudo systemctl enable bluetooth`
  - `$ sudo reboot`
  - 再起動後に以下のコマンドを実行してコントローラが認識されていれば成功。
    `$ sudo  bluetoothctl`



SONY DualShock4 (DS4)コントローラの設定(Bluetooth)

- 以下のサイトを参照した。
  ds4drv
- ds4drvドライバのインストール。ds4drvはLinuxのSONY DualShock 4 用ドライバ。
  - python2のパッケージ管理システムpipをインストールする。
    `$ sudo apt install python-pip`
  - DS4のドライバをインストールする。
    `$ sudo pip install ds4drv`
- DS4とPi4のペアリング
  - DS4をPi4の近くに置く。
  - ds4drvの起動
    - $ sudo ds4drv
    - 以下のように表示される。
      [info][controller 1] Created devices /dev/input/js0 (joystick) /dev/input/event8 (evdev)
      [info][bluetooth] Scanning for devices
  - ペアリング
    - DS4左上のSHAREボタンとDSボタンを同時に5秒程度長押しする。LEDが白く点滅したらボタンから手を話す。
    - LEDが青くなりds4drvを起動した端末で以下のように表示されたら成功。



自動ログイン

Pi4をディスプレイなしで起動するので自動ログインの設定にする。

以下の内容の/etc/lightdm/lightdm.conf.d/1-autologin.confを作成してシャットダウンする。ユーザ名に自動ログインしたいユーザ名を入れる。

[SeatDefaults]
autologin-user=ユーザ名
autologin-user-timeout=0

DS4コントローラでRoombaを動かそう！

- これで準備ができたので始めからDS4コントローラを使いロボットを動かすまでの一連の手順を説明する。
- Pi4とRoombaを通信ケーブルで接続する。
- Roombaの電源を入れる。
- Pi4とモバイルバッテリを接続する。Pi4が起動する。

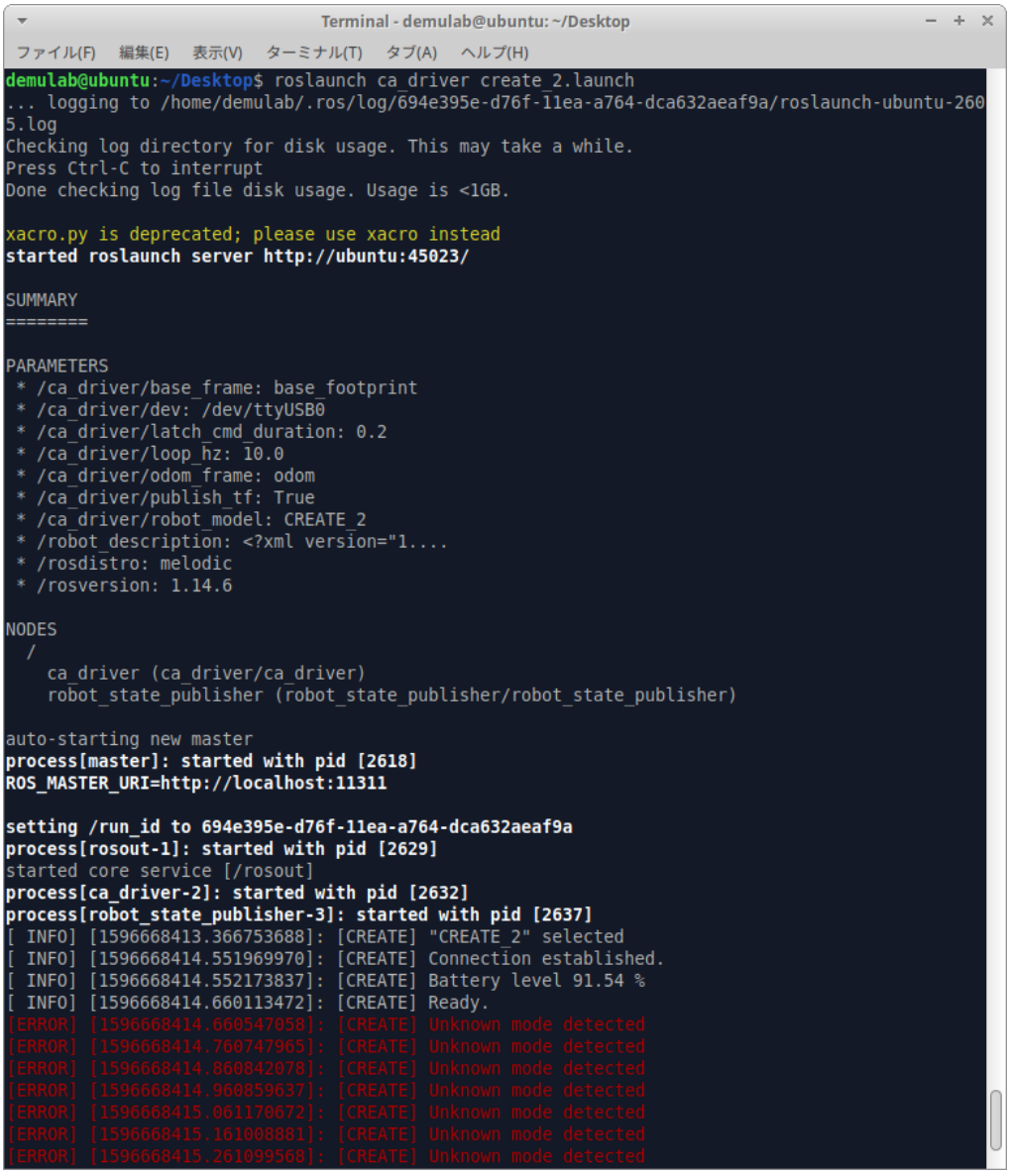
- ノートパソコンを起動して端末(Windowsの場合はsshクライアント。rloginなどのアプリ)を３つ開き、各端末からPi4にssh接続する。上から順番に実行する。
  - １番目の端末
    - 以下のコマンドを実行する。

      ```
      $ sudo ds4drv
      ```

    - DS4コントローラのShareとPSキーを長押しペアリングする。
  - ２番目の端末

    Roombaの制御パッケージを起動する。

    - `$ roslaunch ca_driver create_2.launch`
    - 成功すると「ピッ」と音がなり、以下のように表示される。



- ３番目の端末
  - 以下のコマンドでリモートコントロールのアプリ(パッケージ)を起動する。
    - `$ roslaunch ps4_ros ps4.launch`
    - 以下のエラーが出る場合は次のコマンドを実行する。

```
RLException: ERROR: could not contact master
The traceback for the exception was written to the log file
```

```
$ source /home/ユーザ名/catkin_ws/devel/setup.bash
```

  - 上のコマンドを実行した直後にキャリブレーションが必要となる。L2とR2を同時に押すと以下のように表示される。Release L2 and R2と表示されたら指を話すとキャリブレーションが終了し、利用可能となる。
    なお、[ERROR] [1589086134.806825552]: Couldn't open joystick force feedback! [error][controller 1] Failed to create input device: "/dev/uinput" cannot be opened for writingとエラーが出るが無視する。[WARN] [1562730896.630287264]: Press L2 and R2 to calibrate: 0%
    [WARN] [1562730896.731446432]: Press L2 and R2 to calibrate: 5%
    [WARN] [1562730896.832234176]: Press L2 and R2 to calibrate: 10%
    [WARN] [1562730896.933098464]: Press L2 and R2 to calibrate: 15%
    [WARN] [1562730897.034052928]: Press L2 and R2 to calibrate: 20%
    [WARN] [1562730897.135351008]: Press L2 and R2 to calibrate: 25%
    [WARN] [1562730897.236488864]: Press L2 and R2 to calibrate: 30%
    [WARN] [1562730897.337511264]: Press L2 and R2 to calibrate: 35%
    [WARN] [1562730897.438300640]: Press L2 and R2 to calibrate: 40%
    [WARN] [1562730897.539139328]: Press L2 and R2 to calibrate: 45%
    [WARN] [1562730897.640097056]: Press L2 and R2 to calibrate: 50%
    [WARN] [1562730897.741000544]: Press L2 and R2 to calibrate: 55%
    [WARN] [1562730897.842041696]: Press L2 and R2 to calibrate: 60%
    [WARN] [1562730897.943081120]: Press L2 and R2 to calibrate: 65%
    [WARN] [1562730898.044020896]: Press L2 and R2 to calibrate: 70%
    [WARN] [1562730898.144871008]: Press L2 and R2 to calibrate: 75%
    [WARN] [1562730898.245700928]: Press L2 and R2 to calibrate: 80%
    [WARN] [1562730898.346876544]: Press L2 and R2 to calibrate: 85%
    [WARN] [1562730898.447959104]: Press L2 and R2 to calibrate: 90%
    [WARN] [1562730898.548941536]: Press L2 and R2 to calibrate: 95%
    [WARN] [1562730898.649878528]: Press L2 and R2 to calibrate: 100%
    [WARN] [1562730898.750981568]: Release L2 and R2
    [INFO] [1562730900.952626016]: Calibrated – Ready to use
- DS4コントローラの操作方法は次のとおり。ルンバが動作したら成功。
  - L2ボタン：前進
  - R2ボタン：後進
  - 左ジョイステック：回転

終了方法

Pi4でUbuntuが動いているので電源を急に切るとファイルシステムが壊れて起動しなくなったり、ルンバもリセットをかけないと動作しなくなる恐れがあるので次の手順で終わりましょう。

- ルンバを動かすために起動した端末（Windowsのsshクライアントアプリ含む)上で、Ctrl + C (コントロールキーとCキーを同時に押す）して実行中のプログラムを終了させる。
- どれか１つの端末で以下のコマンドを実行してPi4をシャットダウンする。

```
$ sudo shutdown -h now
```

- Pi4に接続していたモバイルバッテリーは忘れずに外し、ルンバの電源も切りましょう。これで作業終了！

お疲れ様。