Summit XL
Installation Guide for ROS Kinetic
V 1.0

0- Ubuntu 16.04 LTS Installation

Please visit https://help.ubuntu.com/lts/installation-guide/index.html for further information.

1- ROS Kinetic Installation

Add sources to sources.list:
```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
```

Add the keys:
```
sudo apt-key adv --keyserver hkp://pool.sks-keyservers.net --recv-key 0xB01FA116
```

Install Kinetic:
```
sudo apt-get update && sudo apt-get install ros-kinetic-desktop-full
```

Initialize rosdep:
```
sudo rosdep init && rosdep update
```

Environment Setup:
```
echo "export ROS_MASTER_URI='http://$HOSTNAME:11311'" >> ~/.bashrc
```

```
echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc && source ~/.bashrc
```

```
echo "source ~/robot_params.env" >> ~/.bashrc && source ~/.bashrc
```

Getting rosinstall:
```
sudo apt-get install python-rosinstall
```

2- Packages

2.1- ROS packages

Install following ROS package (to resolve future dependencies):

```
sudo apt-get install -y ros-kinetic-transmission-interface ros-kinetic-effort-controllers ros-kinetic-joint-state-controller ros-kinetic-navigation ros-kinetic-ros-control ros-kinetic-ros-controllers ros-kinetic-velocity-controllers ros-kinetic-control-toolbox ros-kinetic-cmake-modules ros-kinetic-serial ros-kinetic-joystick-drivers ros-kinetic-rosbridge-server ros-kinetic-robot-localization ros-kinetic-twist-mux ros-kinetic-imu-tools ros-kinetic-mavros-* ros-kinetic-teb-local-planner ros-kinetic-slam-gmapping ros-kinetic-ackermann* ros-kinetic-rosbridge-suite
```

```
ros-kinetic-urg-node ros-kinetic-ar-track-alvar* screen ros-kinetic-libpcan rcopy
ros-kinetic-astra-* ros-kinetic-move-base ros-kinetic-amcl ros-kinetic-robotnik-msgs
```

2.2- Other packages

```
sudo apt-get install -y openssh-server vim subversion git apache2 mingetty libpopt-dev
sysstat ifstat ntpdate
```

2.3- Catkin workspace and sources folder creation

```
cd && mkdir -p catkin_ws/src && cd catkin_ws && catkin_make
```

```
source devel/setup.bash
```

```
cd && mkdir sources
```

Add the workspace source to .bashrc through this command:

```
echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc && source ~/.bashrc
```

3- PEAK-CAN library installation

There is an incompatibility between last kernel versions and peak can driver, it's necessary to install correct kernel version:

```
sudo apt install linux-image-generic-hwe-16.04* linux-headers-generic-hwe-16.04*
```

```
sudo gedit /etc/default/grub
```

Modify GRUB_DEFAULT (NOT NECESSARY but recommended):

GRUB_DEFAULT='Advanced options for Ubuntu>Ubuntu, with Linux 4.15.0-43-generic'

```
sudo update-grub
```

(**Last tested drivers: 8.6.0**)

Setup "libpopt.dev" dependency by install libpopt-dev library into the system:

```
sudo apt-get install libpopt-dev
```

```
sudo reboot
```

Download and install last drivers from:
http://www.peak-system.com/fileadmin/media/linux/index.htm **or execute:**

```
cd && cd Downloads && wget
http://www.peak-system.com/fileadmin/media/linux/files/peak-linux-driver-8.6.0.tar.gz &&
sudo tar -xzf peak-linux-driver-8.6.0.tar.gz && cd peak-linux-driver-8.6.0
```

Make Install:

```
sudo make clean && sudo make NET=NO_NETDEV_SUPPORT && sudo make install
```

Rename the Peak Can device

```
sudo gedit /etc/udev/rules.d/46-pcan.rules
```

Add the line:
KERNEL=="pcanusb*", SYMLINK+="pcan_base", MODE="0666"

Reload udev rules or reboot to take effect:

```
sudo service udev reload
sudo service udev restart
sudo udevadm trigger
sudo rmmod pcan
sudo modprobe pcan
```

Check if Peak Can is working

Connect Peak USB-CAN converter and check that the OS loads the module (maybe you have to reboot):

```
lsmod | grep pcan
ls /dev/pcanusb*
```

Check that the peakcan module is compiled to be used with NO NETDEV SUPPORT:

```
cat /proc/pcan
```

You should see: **-NA- under ndev**

```
*------------ PEAK-System CAN interfaces (www.peak-system.com) ------------
*------------------------ Release_20120319_n (7.5.0) ----------------------
*---------------- [mod] [isa] [pci] [dng] [par] [usb] [pcc] -----------------
*--------------------- 1 interfaces @ major 250 found -----------------------
*n -type- ndev --base-- irq --btr- --read-- --write- --irqs-- -errors- status
32    usb -NA- ffffffff 255 0x0014 00000005 00000034 00000168 00000075 0x000c
```

Install geographiclib_datasets

```
cd && wget
https://raw.githubusercontent.com/mavlink/mavros/master/mavros/scripts/install_geographiclib_datasets.sh

chmod +x install_geographiclib_datasets.sh

sudo ./install_geographiclib_datasets.sh
```

4 - Summit-XL workspace

Download the workspace file through this link:
https://drive.google.com/open?id=1um9HCZMGnHEwFcj3fThcJh6zI_8Ud5_z
And put the downloaded file in ~/catkin_ws directory.

Now run this commands:

```
cd && cd catkin_ws

tar -xzf rb1_sources<versión descargada>.tar.gz && rm -r rb1_sources.tar.gz

cd src && rosdep install --from-paths . --ignore-src -r -y

cd libraries && ./install_debs.sh
```

Installation of the library robotnik_base_hw

Run this commands to install the library:

```
cd ~/catkin_ws/src/robotnik_base_hw/lib/

sudo dpkg -i ros-kinetic-robotnik-base-hw-lib_0.9.5-0xenial_amd64.deb
```

In case of missing dependencies with other ROS packages, you will have to install them via apt-get install ros-kinetic-****.

Building all the packages

Run this commands to compile the all the packages of the workspace:

```
cd ~/catkin_ws

catkin_make
```

5 - PAD configuration

This section will guide you in the installation of joystick controllers for the manual teleoperation.

5.1 PS4

Requirements

- **Bluetooth device:** now not all dongles are compatibles. It has been tested with the *Konig Dongle Buletooth* (4.0). Check Bluetooth dongle compatibility for a list of tested dongles.

Driver

In ubuntu 16.04 the DS4 is paired through as standard bluetooth device using the System Settings -> Bluetooth -> Pairing Wizard. After the wizard is launched and is searching for devices, press the **Share + PS button** on the pad.

Once is paired, only the PS button must be pressed to connect the PAD to the computer.

This way, when the PAD is connected to the computer, a /dev/input/jsX device will be created. Only joystick and buttons can be used using this method, but accelerometers do not work.
Thus information about pad state is only published when there is an update. We use the accelerometer high frequency publishing as a way to ensure that the connection has not been lost.

To also receive the accelerometers the DS4DRV must be used:

```
sudo apt-get install python-pip
sudo pip install ds4drv
```

Startup configuration

1) copy https://raw.githubusercontent.com/chrippa/ds4drv/master/ds4drv.conf to /etc/ds4drv.conf:

```
cd /etc && sudo wget
https://raw.githubusercontent.com/chrippa/ds4drv/master/ds4drv.conf
```

2) edit this file and change (sudo gedit ds4drv.conf):

```
[ds4drv]
# Run ds4drv in background as a daemon
daemon = true
# Location of the log file in daemon mode
daemon-log = /tmp/ds4drv.log
# Location of the PID file in daemon mode
daemon-pid = /tmp/ds4drv.pid
# Enable hidraw mode

hidraw = true
```

4) Add a udev.rule

Using the PS4 in Hidraw mode (the mode used in 16.04), creates two /dev/input/jsX devices. One created by the DS4DRV and one created by the OS when the pad is paired. If DS4DRV is not running, then data will be published through the device created by the SO (only buttons, not accelerometers), and the other will not be created. If DS4DRV is running, then data will be published through the device created by DS4DRV (with accelerometers included) and the other device will be created also, but no data will be published.

There can be a race condition between the name of the devices, we don't know which will be /dev/input/js[0-9].

To solve it, add this line to the file */etc/udev/rules.d/55-ds4drv.rules (sudo gedit /etc/udev/rules.d/55-ds4drv.rules)*:

```
KERNEL=="js[0-9]*", SUBSYSTEM=="input",
SYMLINK+="input/js_base"
```

In order to check which name is, run this command:

```
udevadm info -a /dev/input/js0
```

This line matches the device created by DS4DRV (using the name attribute, the name of the device created by the SO is only "Wireless Controller", as recognizes the pad as a generic pad).

6- IMU configuration

6.1 Pixhawk - IMU

The device comes configured from Robotnik.

It is necessary to set up the udev rules to link the device correctly:

First, identify the serial number:

```
udevadm info -a /dev/ttyUSB0 | grep serial
         SUBSYSTEMS=="usb-serial"
         ATTRS{serial}=="FT91CQPY"
```

```
        ATTRS{serial}=="0000:00:14.0"
```

Second, create or modify the rules located in */etc/udev/rules.d/50-pixhawk.rules*, setting the value in bold, got from the previous command.

```
KERNEL=="ttyUSB[0-9]*", OWNER="summit", GROUP="dialout",
MODE="0666"
KERNEL=="ttyUSB[0-9]*", ATTRS{idProduct}=="6001",
ATTRS{serial}=="FT91CQPY", NAME="%k", SYMLINK="ttyUSB_PX4",
GROUP="dialout", MODE="0666"
```

Finally, reload and restart the rules:

```
sudo service udev reload
sudo service udev restart
sudo udevadm trigger
```

7- System configuration

7.1- User System permissions

```
sudo usermod -a -G dialout $USER
sudo usermod -a -G root $USER
```

7.2 - Screenrc

In order to prepare the program who manages the console:

```
sudo nano ~/.screenrc
```

And paste this text:

```
termcapinfo xterm* ti@:te@
shell -$SHELL
setenv LD_LIBRARY_PATH
/home/rb1/catkin_ws/devel/lib:/opt/ros/kinetic/lib:/opt/ros/kinetic/lib/x86_64-linux-gnu
zombie kr
verbose on
```

7.3- Boot/startup scripts

In order to to autoboot robot controllers you need to configure the following:

7.3.1 Auto-login

```
sudo systemctl edit getty@ttyX
```

** Where **X** is [1,2,3,4]*
Add:

```
[Service]
ExecStart=
ExecStart=-/sbin/agetty --autologin username --noclear %I 38400
linux
```

Change ***username*** to ***root*** in x =1 and ***summit*** in x = [2, 3, 4]

Save, exit, then:

```
systemctl enable getty@ttyX.service
```

7.3.2 ROS processes autorun

Configuration by script

There is a script located in summit_bringup/scripts to add the environment parameters into the system.
It will add the default environment parameters as well as the the autorun configuration.

- Go to summit_base_bringup/scripts
- run the script
    - `./configure_autorun.sh`

***NOTE**: This scripts edits the ~/.bashrc and it is intended to be run just once, otherwise you need to remove the added content in the end of the file.*

Manual configuration

Edit .bashrc of summit user:

```
cd && gedit .bashrc
```

Add the following lines at the end of file:

```
# SUMMIT XL
# AUTOBOOT
echo "ROBOTNIK SUMMIT XL"
Terminal=`tty`
case $Terminal in
    "/dev/tty2") sleep 10; roscore;;
    "/dev/tty3") sleep 20;

    screen -S bringup -d -m roslaunch summit_xl_bringup
summit_xl_complete.launch;;
esac
```

7.3.3 Gamepad autorun

Edit .bashrc of root user:

```
sudo su
cd && sudo gedit .bashrc
```

Or add the following lines at the end of file to **PS4 configuration**:

```
# ROOT
# AUTOBOOT
echo "ROBOTNIK SUMMIT XL"
    Terminal=`tty`
    case $Terminal in
        "/dev/tty1") sleep 5;
        ds4drv;;
    esac
```

7.4 - CPU power button behavior

This configures power button behaviour. Edit powerbtn file:

```
sudo gedit /etc/acpi/events/powerbtn
```

Add # to comment line: **#action=/etc/acpi/powerbtn.sh**
Add a new line:
`action=/sbin/poweroff`

We need to exit from the sudo user, so write on the terminal:

```
exit
```

```
sudo acpid restart
```

7.5 - GRUB modifications

Open grub with an editor:
`sudo gedit /etc/default/grub`

Add a line with this assignment: GRUB_RECORDFAIL_TIMEOUT=N (In this case N = 0)
Set N to the desired timeout in case of a previously failed boot
`GRUB_RECORDFAIL_TIMEOUT=0`

Update Grub: sudo update-grub
`sudo update-grub`

8- Robot configuration

8.1 Environment variables

There is a script located in summit_xl_bringup/scripts to add the environment parameters into the system.

It will add the default environment parameters as well as the the autorun configuration.

- Go to summit_xl_bringup/scripts

- run the script
    - `./configure_autorun.sh`

***NOTE**: This scripts edits the ~/.bashrc and it is intended to be run just once, otherwise you need to remove the added content in the end of the file. Don't run it again if you did it in 7.2.2*

It will create the default config file robot_params.env and will be copied to the home folder

You can use this params:

```
export ROBOT_ID=summit_xl
# summit_xl.urdf.xacro
# true, false
export ROBOT_HAS_FRONT_LASER=false
# sick_tim561, hokuyo_ug01, hokuyo_ust
export ROBOT_FRONT_LASER_MODEL=hokuyo_ust
#export ROBOT_FRONT_LASER_PORT=/dev/ttyACM0
#export ROBOT_FRONT_LASER_IP=192.168.0.10
# true, false
export ROBOT_HAS_REAR_LASER=false
# sick_tim561, hokuyo_ug01, hokuyo_ust
export ROBOT_REAR_LASER_MODEL=hokuyo_ust
#export ROBOT_FRONT_LASER_PORT=/dev/ttyACM1
#export ROBOT_FRONT_LASER_IP=192.168.0.11
# true, false
export ROBOT_HAS_FRONT_PTZ_CAMERA=false
export ROBOT_FRONT_PTZ_CAMERA_IP=192.168.0.185
export ROBOT_FRONT_CAMERA_MODEL=axis_m5013
# true, false
export ROBOT_HAS_REAR_PTZ_CAMERA=false
export ROBOT_REAR_PTZ_CAMERA_IP=192.168.0.186
export ROBOT_REAR_CAMERA_MODEL=axis_m5013
# 24V, 48V
export ROBOT_BASE_HW_BATTERY_VOLTAGE=24
# disabled, automatic_hw, automatic_sw, manual_sw
export ROBOT_BASE_HW_DOCKER_MODE=automatic_hw
# true, false
export ROBOT_HAS_FRONT_RGBD_CAMERA=false
# usb bus
#export ROBOT_FRONT_RGBD_CAMERA_ID=#1
# true, false
export ROBOT_HAS_REAR_RGBD_CAMERA=false
# usb bus
#export ROBOT_REAR_RGBD_CAMERA_ID=#1
# true, false
export ROBOT_HAS_GPS=false
# ps3, ps4 (default)
export ROBOT_PAD_MODEL=ps4
# 24V motors: 12.52, 48V motors: 9.56
export ROBOT_GEARBOX=9.56
# true, false
export ROBOT_HAS_ENCODER=true
# skid, omni, steel_skid, steel_omni
export ROBOT_KINEMATICS=steel_omni
```

```
# true, false
#export ROBOT_HAS_ARM=false
# double
export ROBOT_BASE_HW_BATTERY_OFFSET=
```

Description of each environment variable:

- **ROBOT_ID** indicates the name of the robot. This is the name of the namespace under all the nodes will be working. This is also used as the prefix of all the subcomponents.*(*summit_xl*)*
- **ROBOT_XACRO** indicates the path where the xacro file is. (inside the robot folder in robot_description)*(*summit_xl.urdf.xacro*)*
- **ROBOT_FRONT_LASER_MODEL** indicates the model of the laser that the robot is using. The model is the name of the launch file.*(*sick_tim561/hokuyo_ug01/hokuyo_ust*)*
- **ROBOT_REAR_LASER_MODEL** indicates the model of the laser that the robot is using. The model is the name of the launch file.*(*sick_tim561/hokuyo_ug01/hokuyo_ust*)*
- **ROBOT_HAS_FRONT_LASER** indicates if the robot has a laser in front. *(*true/false*)*
- **ROBOT_HAS_REAR_LASER** indicates if the robot has a laser in rear. *(*true/false*)*
- **ROBOT_HAS_FRONT_PTZ_CAMERA** indicates if the robot has the ptz camera in front. *(*true/false*)*
- **ROBOT_HAS_REAR_PTZ_CAMERA** indicates if the robot has the ptz camera in front. *(*true/false*)*
- **ROBOT_HAS_GPS** indicates if the robot has gps. *(*true/false*)*
- **ROBOT_HAS_FRONT_RGBD_CAMERA** indicates if the robot has a front rgbd camera. *(*true/false*)*
- **ROBOT_FRONT_RGBD_CAMERA_ID** camera id to identify in the bus
- **ROBOT_HAS_REAR_RGBD_CAMERA** indicates if the robot has a front rgbd camera. *(*true/false*)*
- **ROBOT_REAR_RGBD_CAMERA_ID** camera id to identify in the bus
- **ROBOT_PAD_MODEL** pad model used. *(*ps4/ps3/logitechf710/xbox360*)*
- **ROBOT_GEARBOX** establishes the motor gearbox value. *(*24V: 12.52 | 48V: 9.56*)*
- **ROBOT_HAS_ENCODER** indicates if the robot has encoders. *(*true/false*)*
- **ROBOT_KINEMATICS** kinematic configuration of the robot. *(*skid/omni/steel_skid/steel_omni*)*
- `**ROBOT_HAS_ARM** indicates if the robot has an arm *(*true/false*)*

***NOTE**: Please check the README in summit_xl_bringup package in order to get the most updated list of variables*