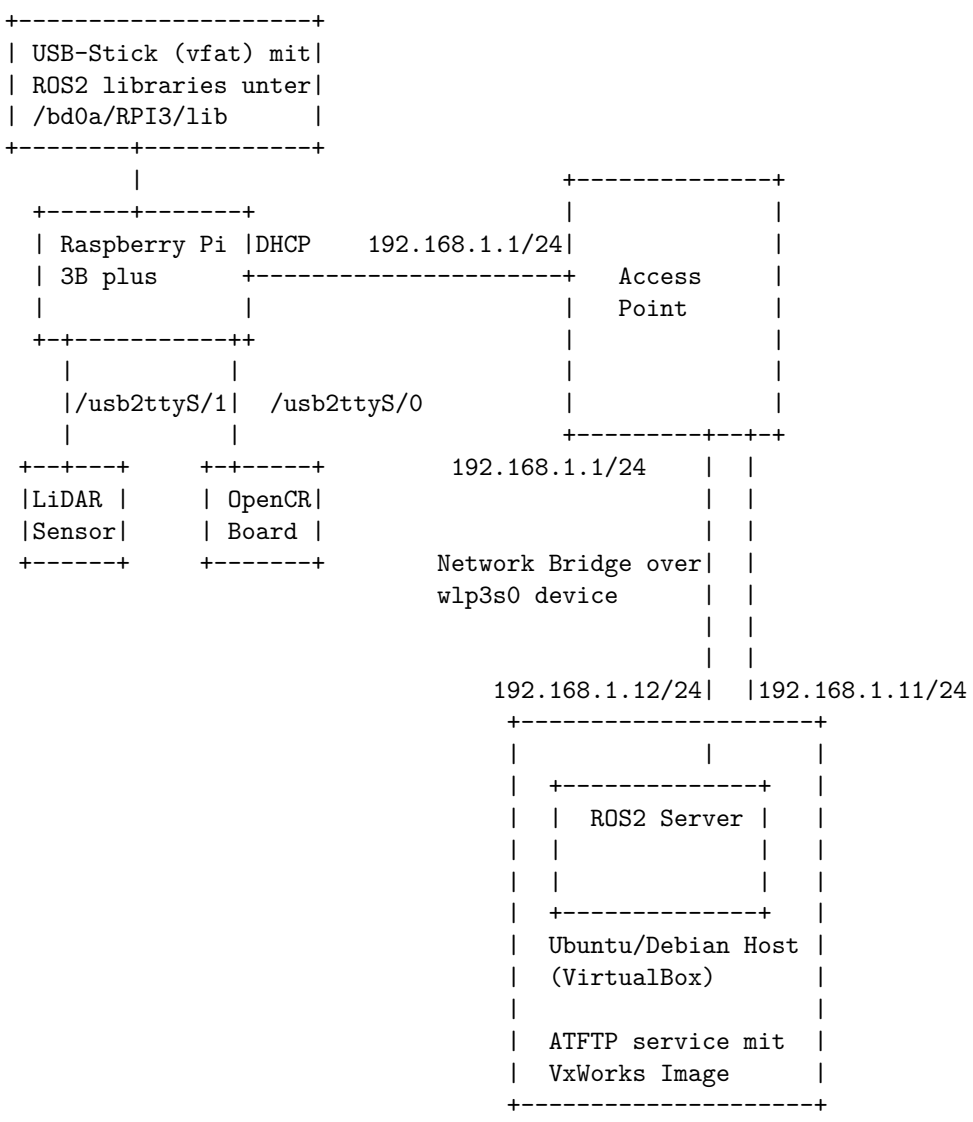
Deployment Guide for VxWorks on TurtleBot3

TurtleBot3 Architecture



Technologies and Prerequisites

Hardware

- integrated SD card reader or adapter
- TurtleBot3 with Raspberry Pi 3B plus model, LiDAR, OpenCR and motor drivers
- Access Point, e. g. TP-Link TL-WR802N

Software

- working Docker installation
- Ubuntu or Debian-based Linux distribution as development host
- VirtualBox as Virtualisation platform for ROS2 control server
- TFTP server to distributed VxWorks image over network
- Network infrastructure and Configuration
- U-Boot environment configuration
- burger.yaml
- Certain version of the VxWorks SDKs available at the SDK download page
- Raspberry Pi 3B SDK (at least version 1.4): https://labs.windriver.com/downloads/wrsdk-vxworks7-raspberrypi3b-1.4.tar.bz2
- ROS2 libraries(clone with HTTPS): https://github.com/Wind-River/vxworks7-ros2-build.git

VxWorks Deployment

To run VxWorks on the TurtleBot3 robot, the SDK provides a pre-build VxWorks image which can be used for a VxWorks network boot. As a basis for the network boot a SD card with a U-Boot environment (RPI3) and a TFTP server (atftp package on Ubuntu dev host) is required.

1) Extract Raspberry Pi 3B SDK

   tar xjvf wrsdk-vxworks7-raspberrypi3b-1.4.tar.bz2

Under `wrsdk-vxworks7-raspberrypi3b/bsps/rpi_3_0_1_1_0/uboot` the image `uVxWorks` and the Raspberry Pi hardware definition `rpi-3b-plus.dtb` are located. After the installation of the aftfp service the image can be uploaded on the server as follows:

atftp

> connect localhost

> put uVxWorks

> put rpi-3b-plus.dtb

> quit

Make sure that the atftp service is running on the correct interface for further steps regarding the VxWorks deployment. In order to prepare an SD card with U-Boot, the following steps are required starting with the creation of suitable SD card partitions. Therefore an official Ubuntu or Raspian image can be used.

```
sudo dd if=2019-09-26-raspbian-buster.img of=/dev/mmcblk0
bs=4M
```

To change the original kernel to the u-boot image the configurations in the `config.txt` file need to be adjusted which at least has to contain the following parameters:

```
dtparam=i2c_arm=on dtparam=spi=on

dtparam=audio=on

[pi4]

dtoverlay=vc4-fkms-v3d

max_framebuffers=2

[all]

enable_uart=1

kernel=u-boot.bin

arm_control=0x200

device_tree_address=0x100

device_tree_end=0x8000

dtparam=i2c_arm=on

dtparam=spi=on

boot_delay=0

disable_splash=1
```

To install the correct U-Boot image on the SD card, a pre-build U-Boot image from the WindRiver Workbench at `WindRiver/vxworks-7/pkgs_v2/unsupported/rpi_3/rpi_3-0.1.1.0/_bootloader` can be used or a suitable image can be build from the U-Boot repository.

```
sudo cp u-boot.bin config.txt /path/to/sd-card
```

For the initial U-Boot configuration interrupt the boot sequence at the very beginning of the RPI3 startup by pressing space on the key board. This can be accomplished by using the RPI3's serial connection on the GPIO block. In case of reaching the `U-BOOT\>` prompt the U-Boot environment can be configured according to the network mentioned above:

setenv ipaddr 192.168.1.10

setenv netmask 255.255.255.0

setenv serverip 192.168.1.11

setenv gatewayip 192.168.1.1

saveenv

setenv bootcmd “tftp 0x08000000 uVxWorks; tftp 0x09000000 rpi-3b-plus.dtb; bootm 0x08000000 - 0x09000000”

saveenv

boot # executes the previous defined bootcmd to proceed booting

Before executing `boot` check wireless connections and the status of the tftp service and every component is empowered well in order to exclude under-voltage issues.

After completing the network boot of the VxWorks image the VxWorks prompt should be also accessible over the serial connection. With `ifconfig` the current IP address of the VxWorks instance can be read out, as the pre-build image uses DHCP as default network configuration. In the described network `192.168.1.0/24` this could be `192.168.1.128`. For further developments it is recommended to use the telnet service available:

telnet 192.168.1.128

To close a telnet session in case of a hard shutdown, press `Ctrl+Alt+9` and type quit to return to bash.

ROS2 Deployment

1) Build ROS2 docker container

   git clone https://github.com/Wind-River/vxworks7-ros2-build.git

   cd vxworks7-ros2-build

   cd Docker/vxbuild

   docker build -t vxbuild:1.0 .

   cd Docker/vxros2build

   docker build -t vxros2build:1.0 .

2) Build ROS2 libraries with RPI 3B SDK > cd vxworks7-ros2-build

   docker run -ti -v :/wrsdk -v $PWD:/work vxros2build:1.0

   wruser@d19165730517:/work source /wrsdk/toolkit/wind_sdk_env.linux

wruser@d19165730517:/work make

Check if build artifacts are present

wruser@d19165730517:/work ls export/root/

include lib llvm

For Rebuild remove existing artifacts

wruser@d19165730517:/work make distclean

wruser@d19165730517:/work make

3) Prepare USB-Stick (/dev/sdc1, could differ from your setup) to store ROS2 libraries. Before the preparation of the USB-Stick the directory structure is initialized.

```
--RPI3
   |
   |--burger.yaml
   |--llvm
   |--lib
   |--include
```

sudo umount /dev/sdc1

sudo mkfs.vfat -F 32 -n 'RPI3' /dev/sdc1

sudo mount /dev/sdc1 /mnt

sudo mkdir /mnt/RPI3

Warning: ROS2 build artifacts contain two symbolic links, but vfat as filesystem does not support symbolic links. Therefore a modified copy-command is required

cp -LR /path/to/libraries /mnt

As the TurtleBot3 configuration is stored in the `burger.yaml` file for the "Burger" model, a copy of such a configuration is required. If you have access to ROS2 package source code of the TurtleBot3 than the configuration file can be found in `~/turtlebot3_ws/src/turtlebot3/turtlebot3_bringup/param/` depending on the model.

cp /path/to/burger.yaml /mnt/RPI3

umount /dev/sdc1

If the `umount` command has finished, the USB-Stick can be plugged into the Raspberry Pi 3B plus which is abstracted as `/bd0a` device.

ROS2 Server Installation

To install a ROS2 Server,a Ubuntu 18.04 instance is required which is hosted as virtual machine using Virtualbox. Besides enough hardware resources (1-2 core, 2048-4096 MB RAM, VirtIO storage driver) it mandatory to activate the mode `network bridge` on the wlan device `wlp3s0`. In addition to that it is recommended to use a `virtio-net` network type adapter.

After the operating system's installation ROS2 can be installed as a package if the ROS apt repository is added. For further details see https://emanual.robotis.com/docs/en/platform/turtlebot3/ros2_setup/.

TurtleBot3 Bringup

As VxWorks and ROS2 deployment are required to bring up a TurtleBot3. In the following it is assumed that the USB-Stick and the serial connections for OpenCR board and LiDAR are correlating with the system architecture from above. However it is possible that serial connections are swapped and therefore the following commands do not work. To check whether all peripherals are connected proceed as follows:

> devs # shows all connected device, look for /bd0a, /usb2ttyS/0, /usb2ttyS/1

> cmd # opens a Shell

> set env LD_LIBRARY_PATH="/bd0a/RPI3/lib" # path to the location of ROS2 libs

> cd /bd0a/RPI3/llvm/bin/ # navigate to pre-build ROS2 binaries

> rtp exec -u 0x20000 timer_lambda.vxe # Start Hello World publisher, interrupt with Ctrl+C

> rtp exec -u 0x80000 turtlebot3_ros.vxe -- -i /usb2ttyS/0 ___params:=/bd0a/RPI3/burger.yaml & # Start TurtleBot3 Bringup routine

> rtp exec -u 0x30000 /bd0a/RPI3/llvm/bin/hlds_laser_publisher.vxe -- ___node:=hlds_laser_publisher & # Starts hlds_sensor publisher