Technische Universiteit
Eindhoven
University of Technology

Installation Guide

Installing Matlab Pixhawk PSP

Master:	Systems & Control
Department:	Mechanical Engineering
Research group:	Control Systems Technology
Date:	November 14, 2018

Eindhoven, November 14, 2018

Table of Contents

Chapter 1

Installation

Installation is based on a clean Linux installed PC. Furthermore, Matlab R2017B is used during this tutorial. Other versions can be used on your own risk.

1.1 Connect to TU/e WiFi

From a clean Ubuntu 18.04 LTS, connect to the TU/e network [1].

Connect to TU/e WiFi Right click on the wireless symbol in your top-panel and select:
'tue-wpa2'

Move to tab "WiFi Security" and use these settings:

Security:	WPA & WPA2 Enterprise
Protected:	Protected EAP (PEAP)
Anonymous identity:	sXXXXXX
CA Certificate:	'No CA certificate is required'
PEAP version:	Automatic
Inner authentication:	MSCHAPv2
Username:	sXXXXXX
Password:	XXXXXXX

Update dependencies. First start a Terminal (Ctrl + Alt + t)

```
sudo apt-get update
```

Activate colors in terminal

```
gedit home/<username>/.bashrc
```

Uncomment the line by removing #:

```
# force_color_prompt=yes
```

Save and restart the terminal.

1.2 Install Terminator

Installing terminator is based on [2].

```
sudo apt-get install terminator
```

after installation

1. Open "Terminator".
2. Make the set-up to be saved: some split ups
   (a) Right Click.
   (b) Choose "split horizontally" or "split vertically"

3. Continue the previous step, until you are happy.
4. Right Click.
5. Choose ”preferences”.
6. Pick tab ”layout”.
7. Click ”add”.
8. Enter a name (= <your chosen name>).
9. Press ”enter”.
10. Click ”close”.

Configuring Terminator as default terminal
Open a terminal (Ctrl + Alt + t)

```
cp /usr/share/applications/terminator.desktop home/<username>/.local/share/applications
```

Open launcher file

```
gedit home/<username>/.local/share/applications/terminator.desktop
```

Find the first occurrence of Exec=terminator and add -ml ¡your chosen name¿ (So, it should now be

```
Exec=terminator —ml <your chosen name>)
```

Save and close the file
Log-out and log back in.
Press the windows key on your Keyboard, and search for terminator.
Drag the terminator icon to the launcher (the menu with icons on the left). You can now always start the terminator with your layout by clicking the icon.

1.3 Installing Metacity

Open a terminal (Ctrl + Alt + t)

```
sudo apt—get update
sudo apt—get install gnome—session—flashback
```

After installation, reboot.
In the login prompt, choose GNOME Flashback (Metacity).

1.4 Installing Matlab

Detailed information can be found [3].
First, install cifs-utils. Open a terminal (Ctrl + Alt + t)

```
sudo apt—get install cifs—utils
```

Make a directory to mount the matlab network location:

```
sudo mkdir /mnt/matlab
```

Mount software distribution tree:

```
mount -t cifs //campusmp.campus.tue.nl/software/mworks/linux /mnt/matlab -o ...
    username=<yourusernamehere>
```

Go to the specific Matlab folder:

```
cd /mnt/matlab/R2017b
```

Start installing Matlab:

```
sudo ./install
```

1.5 Matlab Pixhawk PSP Installation

Download the Matlab Pixhawk PSP (PX4PSP_v3_0_4_351_R2017b.mltbx) from [4].
Open Matlab by opening a terminal and running:

```
matlab
```

Create a folder in Documents:

```
mkdir home/<username>/Documents/MATLAB/PX4_Project
```

Copy the PSP to the project folder:

```
cp home/<username>/Downloads/PX4PSP_v3_0_4_351_R2017b.mltbx ...
    home/<username>/Documents/MATLAB/PX4_Project
```

In Matlab, navigate to the PSP location:

```
cd home/<username>/Documents/MATLAB/PX4_Project
```

Double click the .mltbx and click install. Give Matlab user permissions

```
sudo chmod -R 777 /usr/local/MATLAB/R2017b/toolbox/local
```

And create a directory px4:

```
mkdir home/<username>/Documents/MATLAB/PX4_Project/px4
cd home/<username>/Documents/MATLAB/PX4_Project/px4
```

Run the following command in Matlab:

```
PixhawkPSP('/home/<username>/Documents/MATLAB/PX4_Project/px4')
```

If this command crashes, go to [5].
Or install the canberra package

```
sudo apt-get install libcanberra-gtk-module
```

Install Git

```
sudo apt-get install git
```

If the GUI starts, press Download Firmware.
Although the download might result in errors, this will be fixed later on. If asked for username and

password, just press enter. Basically all the other necesary files are downloaded and the px4 firmware will be downloaded later see section 1.7. Probably NxWidgets.git and tools.git are not found, they will be downloaded manually. After it is finished, save and exit the GUI.
Open a terminal (Ctrl + Alt + t) and go to the project folder in Linux

```
cd home/<username>/Documents/MATLAB/PX4_Project/px4/Linux_setup
```

Provide the bash script of executing rights and run it:

```
chmod +x ubuntu_sim_common_deps.bash
sudo ./ubuntu_sim_common_deps.bash
```

1.6 Installing additional compiler

Matlab utilizes gcc-arm-none-eabi-5_4-2017q2 compiler, which is not supported by the PX4 code. Therefore, the old compiler needs to be remover and a new one needs to be installed. The following commands and instructions can be found at [6].
First run:

```
sudo apt-get install python-serial openocd \
flex bison libncurses5-dev autoconf texinfo \
libftdi-dev libtool zlib1g-dev -y
```

Remove any old versions of the arm-none-eabi toolchain.

```
sudo apt-get remove gcc-arm-none-eabi gdb-arm-none-eabi binutils-arm-none-eabi ...
    gcc-arm-embedded
sudo add-apt-repository --remove ppa:team-gcc-arm-embedded/ppa
```

Go to the home directory

```
cd home/<username>
```

By running, the following command, an overview of folders is shown.

```
ls
```

Search for old gcc-arm-none-eabi directories and if present, remove them. This includes directories and .tar.bz2 files.

```
rm -rf gcc-arm-none-...
```

Edit the bashrc file by running:

```
gedit .bashrc
```

If, at the end of the file, a line exist in the form of:

```
export PATH=/home/<username>/gcc-arm-none-eabi-.../bin:$PATH
```

Remove it entirely. The new GCC cross0compiler can be installed by running:

```
pushd .
cd home/<username>
wget ...
    https://armkeil.blob.core.windows.net/developer/Files/downloads/gnu-rm/7-2017q4/gcc-arm-none-eabi-7-20
tar -jxf gcc-arm-none-eabi-7-2017-q4-major-linux.tar.bz2
```

```
exportline="export PATH=$HOME/gcc-arm-none-eabi-7-2017-q4-major/bin:\$PATH"
if grep -Fxq "$exportline" home/<username>/.profile; then echo nothing to do ; else ...
    echo $exportline >> home/<username>/.profile; fi
popd
```

After popd appears in the terminal, press enter. Once the command is executed and finished, restart the machine. After the reboot, run in a terminal the following command:

```
arm-none-eabi-gcc --version
```

If everything is properly installed, the result should look something like:

```
arm-none-eabi-gcc (GNU Tools for Arm Embedded Processors 7-2017-q4-major) 7.2.1 ...
    20170904 (release) [ARM/embedded-7-branch revision 255204]
Copyright (C) 2017 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

1.7 Downloading Software manually

Open a terminal and go to the project folder:

```
cd home/<username>/Documents/MATLAB/PX4_Project/px4/
```

Start downloading the firmware manually by running:

```
git clone https://github.com/mathworks/PX4-Firmware.git
```

Copy the px4_simulink_app module directory to the new firmware directory.

```
cd home/<username>/Documents/MATLAB/PX4_Project/px4
cp -a Firmware/src/modules/px4_simulink_app/ PX4-Firmware/src/modules
```

Rename the old Firmware folder and copy the new firmware folder:

```
cd home/<username>/Documents/MATLAB/PX4_Project/px4
mv Firmware Firmware_old
cp -a PX4-Firmware Firmware
```

Add the Simulink software to the make file, by editing the make file:

```
cd home/<username>/Documents/MATLAB/PX4_Project/px4
gedit Firmware/cmake/configs/nuttx_px4fmu-v2_default.cmake
```

Add the Simulink module to the build code:

```
modules/px4_simulink_app
```

Return to the Matlab GUI:

```
PixhawkPSP('/home/<username>/Documents/MATLAB/PX4_Project/px4')
```

Press validate firmware and check for errors. Save settings and Exit. In Matlab, open an example, such as "px4demo_rgbled.slx" in the example folder in px4, and build the code. If the build does not work, try changing permissions of the Firmware directory by typing in a terminal:

```
sudo chmod -R 777 /home/<username>/Documents/MATLAB/PX4_Project/px4/Firmware
```

1.8 Uploading code to the Pixhawk

After building is completed, the code can be uploaded by opening a terminal, and running the following commands:

```
cd home/<username>/Documents/MATLAB/PX4_Project/px4/Firmware
make px4fmu-v2_default upload
```

If the upload results in errors, add user to the dialout group, [7]:

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

Logout and login again. Uploading code should now be possible, unless the boot process alternation has not been configured on the Pixhawk.

1.9 Boot process alternation

Under the assumption that the Pixhawk PX4 firmware is uploaded through QGroundcontrol, a boot process alternation can be made. An overview of all possible boot process changes can be found here [8]. In order to do so, plug the micro-sd card in the PC, create a folder etc on it and create a file called rc.txt.

```
usleep 1000
uorb start
usleep 1000
nshterm /dev/ttyACM0 &
usleep 1000
px4io start
usleep 1000
#commander start
#usleep 1000
# if this line is active, Simulink will not upload code to the Pixhawk via usb
#mavlink start -d /dev/ttyS1 -b 115200
#usleep 5000
#dataman start
#usleep 1000
#navigator start
#usleep 1000
sh /etc/init.d/rc.sensors
usleep 1000
#sh /etc/init.d/rc.logging
#usleep 1000
#gps start
attitude_estimator_q start
position_estimator_inav start
usleep 1000
mtd start
usleep 1000
param load /fs/mtd_params
usleep 1000
rgbled start
usleep 1000
gps start
usleep 1000
px4_simulink_app start
usleep 1000
exit
```

During boot, this will start the essential modules, such as the uorb messages bus, the nsh terminal and the px4_simulink_app.

1.10 Putty

By using putty, command messages can be send and received to the PX4 system console. To install putty, run in a terminal:

```
sudo apt—get install —y putty
```

After installation, startup putty by running

```
putty
```

For a first-time setup, apply the following configuration:

Connection Type:	Serial
Serial line:	/dev/ttyACM0
Speed	57600
Saved Sessions	PX4 Session

After setting the configuration, press Save. Every time putty is started, the PX4 Session will be available and can be loaded to start communicating with the PX4.
Information about the PX4 System Console can be found at [9]. The most usefull commands are:

```
ls
```

which displays all the directories currently present on the Pixhawk.

```
free
```

which displays the memory allocation on the Pixhawk.

```
help
```

which displays all commands available and the builtin apps present on the Pixhawk.

Chapter 2

What did we learn?

2.1 Finished:

- If we want to arm the motors, the RC NEEDS to be connected, otherwise the motors will stay disarmed. This feature is inherited from the PX4 flight stack. In nsh run px4io status to see the arming status and the servo outputs (motor pwm values).

2.2 Temporary Fix:

- CSC Parameters provided by the PSP from Matlab results in an error during building, which is due to the fact that it tries to include a file which has a path which the cmake procedure can not handle. Main guess would be that it includes a space somewhere. **Temporary Solution:** In the software, all Pixhawk_CSC instances are removed in file InitFcn_R2017b.m and it is saved as InitFcn_R2017b_no_csc.m. In the Simulink file PX4_Run_Mode_R2017b.slx the initialization file is also changed to the latter. Furthermore, if a solution is found for the CSC problem, the InitFcn_R2017b.m can be used again.

2.3 Issues:

- Uploading issues. Every once and a while, the upload procedure aborts resulting in a failed code upload to the Pixhawk.
- CSC Parameters provided by the PSP from Matlab results in an error during building, which is due to the fact that it tries to include a file which has a path which the cmake procedure can not handle. Main guess would be that it includes a space somewhere.

Bibliography

[1] T. United, "How to connect to the tu/e wireless network." http://www.techunited.nl/wiki/index.php?title=How_to_connect_to_the_TU/e_wireless_network.

[2] TU/e, "Embedded motion control/tutorials/customizing ubuntu." http://cstwiki.wtb.tue.nl/index.php?title=Embedded_Motion_Control/Tutorials/Customizing_Ubuntu.

[3] TU/e, "Tu/e matlab installation instructions on linux." https://intranet.tue.nl/universiteit/diensten/ict-services/hulp-en-ondersteuning/software-tue-werkplek/matlab-voor-linux/.

[4] Mathworks, "Matlab pixhawk psp." https://nl.mathworks.com/hardware-support/forms/pixhawk-downloads-conf.html?elqsid=1539948744240&potential_use=Education.

[5] Mathworks, "Psp window load error." https://nl.mathworks.com/matlabcentral/answers/364551-why-is-matlab-unable-to-run-the-matlabwindow-application-on-linux.

[6] px4, "Gcc compiler." https://dev.px4.io/en/setup/dev_env_linux_ubuntu.html#nuttx-based-hardware.

[7] px4, "Adding user to dialout." https://dev.px4.io/en/setup/dev_env_linux.html#Development-Toolchain.

[8] px4, "System startup." https://dev.px4.io/en/concept/system_startup.html.

[9] px4, "Px4 system console." https://dev.px4.io/en/debug/system_console.html.