AlphaLink Engineering GmbH
Bismarckstraße 10-12
10625 Berlin

Hardware-in-the-Loop Simulator

Turn your Flying Lab into a Full Simulator

Manual

Version: 2.0
Date: October 31, 2023

Terms and Conditions

The Hardware-in-the-Loop (HiL) for the AlphaLink Flying Lab consists of software and hardware for experimental purposes. The software is provided on a USB stick. The MATLAB/ Simulink model may not be copied or redistributed in whole or part. One copy per set is allowed for internal use only. Publications that refer to the supplied MATLAB/Simulink model must always be made with reference to this manual.

Note: The current version of the HiL is designed for computers with MS Windows as operational system and is tested with MATLAB R2017b. Downward compatibility is not guaranteed; upward compatibility is not given.

Delivery Scope

Included in the delivery of an AlphaLink HiL are:[1]

1. Original Vector Informatik equipment and software:
    - Flying Lab (Talon Model), incl. battery
    - VN1640A CAN/LIN Network Interface, incl. 2x CANpiggy 1057Gcap (Version a) or VN1630A CAN/LIN Network Interface (Version b) with CANterm 120,
    - CANoe PRO (24-months license),
2. CAN cable,
3. Remote Control (FrSky Taranis Q X7 RC Transmitter 2.4 GHz with 16 channel, incl. battery) and RC Receiver (FrSky R-XSR EU LBT), and
4. USB Stick with all relevant software and this manual (see the directory in Appendix A).

[1] *Note: International shipments may not include batteries.*

Contents

1 Introduction

This manual guides the reader through the installation of a Hardware-in-the-Loop (HiL) simulator for the PX4 environment. It is based on the AlphaLink Unmanned Aircraft Experimental System (UAXS), the *Flying Lab* for research and teaching purposes. After a short introduction to HiL simulation, a quick start for initiating the HiL simulation is presented.

Note: All path and file references refer to the directory as provided on the USB stick. For a complete overview, see Appendix A. It is recommend to copy the relevant files from the USB stick to your hard drive, while maintaining the structure of the directory.

1.1 Why Hardware-in-the-Loop Simulation?

Flight tests are expensive and time consuming. HiL simulations can increase success rates of real flight tests and therefore ensure that the flight test works well and the invested effort is worthwhile. This renders testing in advance highly desirable. For autonomous or remotely piloted aircraft, testing should be as close as possible to real-world conditions and this can only be achieved with a HiL simulator. In such HiL testing, the movement of the aircraft, the environment, and the behavior of sensors and actuators are replaced by a mathematical model to perform a realistic simulation. All the remaining physical components are tested with a physical aircraft in the real world.

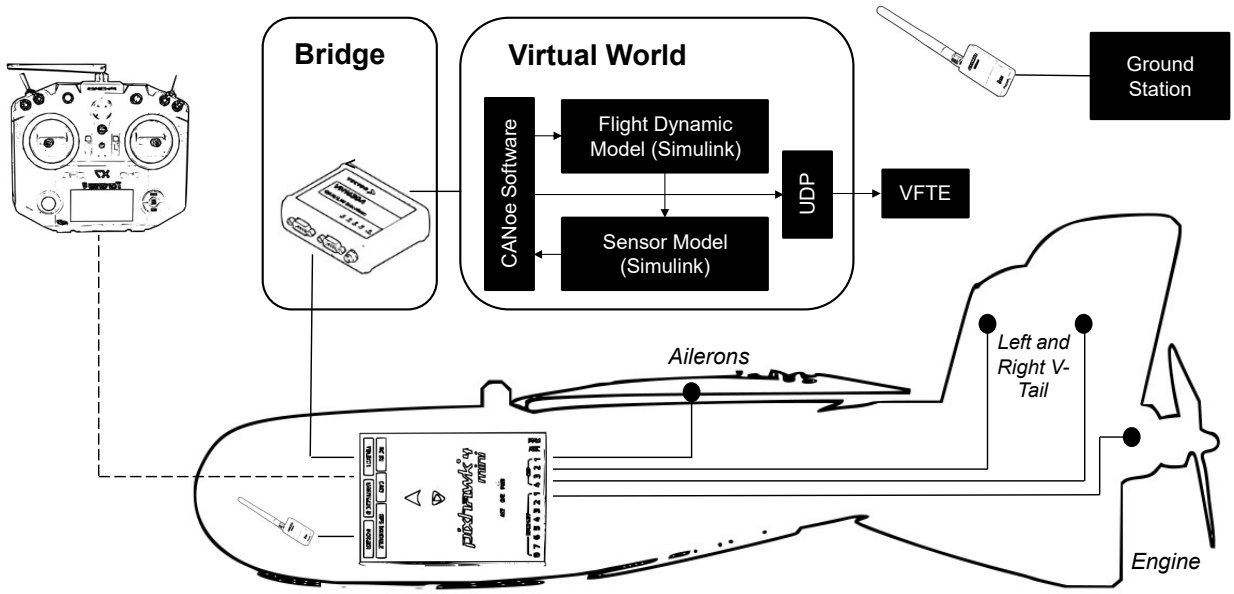


Figure 1: HIL Components.

The provided HiL set consists of four main components illustrated in Fig. 1:

- the aircraft,
- the interface,
- the simulation, and
- the visual.

The first component is the aircraft system provided by the AlphaLink UAXS. This includes the Nano Talon as aircraft and the Pixhawk 4 Mini (Pixhawk) as flight controller inside. The Pixhawk will be the desired target hardware to run the controller structure that shall be tested. The aircraft system is exactly the same as for ordinary flight testing. Hence, it takes the RC command as inputs and sends the flight data as outputs to the ground station, in this case the software QGroundControl. The difference between the HiL simulation and the basic Flying Lab is the modified flight stack, i.e. the Pixhawk is interacting with the simulation environment instead of the real world. This is achieved through the interface component.

The HiL Simulator is using the CAN bus for the communication between the hardware and the simulation loop. This CAN bus is controlled using the original hardware and software from *Vector Informatik*. The interface provides the hardware with the required sensor data, while transmitting the actuator commands from the hardware to the simulation model.

The simulation model is the third component and is running on the simulation host connected with the software. The simulation is done using *MATLAB/Simulink*. It takes the actuator commands and the initial start conditions for the desired testing situation as inputs. With this, the model computes all dynamics ranging from the flight dynamics to the sensor and the actual actuator behavior. The objective is providing the sensor data of the aircraft to the flight controller under nearly real-world conditions.

To visualize the simulated flight of the aircraft, the simulation model is connected to the AlphaLink Virtual Flight Test Environment (VFTE) as the fourth component. The advantage of this simulator is the opportunity to test the controller structure directly on the target hardware without the risk of flight accidents. The testing proceeds in a safe environment, while all remaining aspects from flight testing like RC control and ground monitoring through the virtual ground station are given. In addition, the movement of the control surface can be seen in real time at the physical aircraft model, while the resulting dynamic behavior is observed in the VFTE. Also critical situations like sensor failure can be tested without the consequences of any physical damage. In the end, the HiL creates completely new opportunities and helps saving a lot of time and money.

AlphaLink provides all necessary components for turning the Flying Lab into a full simulator: the aircraft system (the original Flying Lab), the VN1630A/VN1640A CAN Interface, a RC remote control, and a custom CAN cable. Additionally, a USB stick with all the required software is provided. This stick is named HIL MAIN and contains eight folders, of which five are central to the HiL:

- `CANoe`: this subfolder contains all data relating to CANoe such as the required configuration and `DBC` file, which is necessary for the simulation interface;
- `VM HIL`: the virtual machine that contains the development environment for the Pixhawk is in this subfolder;
- `Simulation Simulink`: this subfolder contains the simulink model for the simulation and all the required MATLAB scripts;
- `Simulator Webserver`: the source code for the implementations concerning the VFTE and the interface to the VFTE are in this subfolder;
- `Documentation`: this subfolder includes all the necessary documentations, e.g. this manual and also the manual for the basic Flying Lab.

In the following sections, all aspects of the supplied hardware and software will be explained. In the quick-start guide, only the build up and the starting procedure are described; the succeeding chapters will go in detail through all the supplied software.

1.2 Quick-Start Guide

This quick start gives the basic set of instruction to safely start the simulation. It is therefore assumed that all the required software is already installed:

- ✓ CANoe (PRO license, version 14.0 or newer),
- ✓ VMware (Workstation Pro 15 Player).
- ✓ MATLAB/Simulink (R2017b), including `Vector CANoe` library,
- ✓ Node.js, and
- ✓ QGroundControl.

An explanation regarding the required installation process and use of each software is given in the subsequent Ch. 2 to 6.

The quick-start instructions are given in chronological order beginning with the structure and the build up of the simulator. Afterwards, the preparation of the components and the final initialization follow.

1.2.1 Building up the Simulator

The HiL setup consists of five main hardware components connected with three different types of cables. The hardware components are:

- Simulation host,
- VN1630A/VN1640A Vector CAN Interface with CANterm,
- AlphaLink Flying Lab, incl. a Pixhawk 4 Mini flight controller,
- (*optional*) RC remote control, and
- Telemetry transceiver and battery pack (optional).

Those are connected with

- 2x micro USB cables,
- 1x USB cable type B, and
- 1x custom CAN cable.

Note: You should always treat all the physical components with due care when plugging or unplugging cables to connect them during the build-up to avoid interrupted connections or damages.

Building up the HiL, requires four steps.

1. Use the USB type B cable to connect the VN1630A/VN1640A with the host computer (see Fig. 2). If the communication is established, the control LED of the VN1630A/ VN1640A should be flashing green.

Figure 2: VN1630A Connection to Host Computer. (*VN1640A similar.*)

2. Use a micro USB cable to connect the Pixhawk to the host computer. The micro USB interface is found in the head of the aircraft (see Fig. 3).

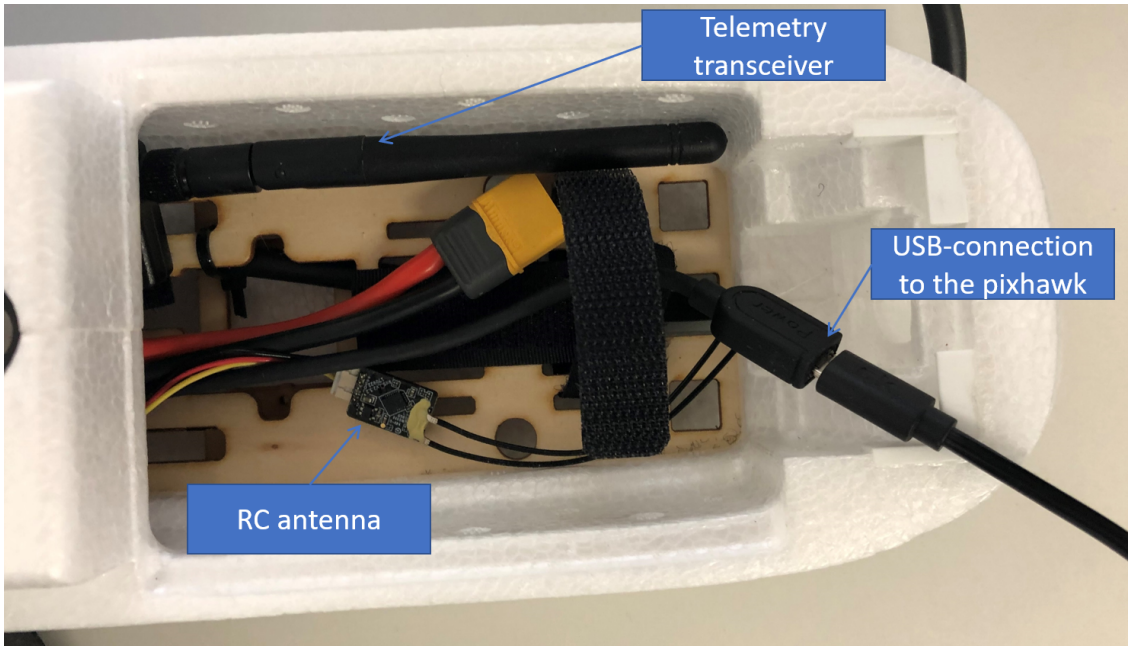


Figure 3: Pixhawk Connection to Host computer.

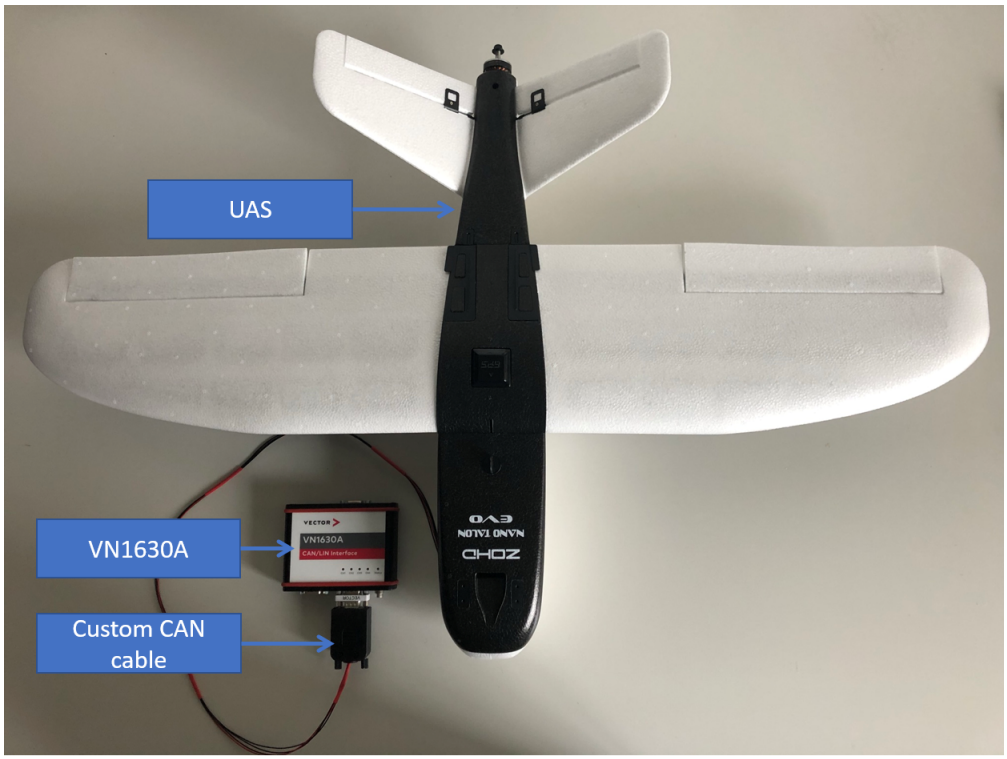


Figure 4: Setup for the CAN Communication.

3. Connect the aircraft and the Channel 1 port of VN1630A/VN1640A box through the custom CAN cable (see Fig. 4). To prevent the risk of disconnection, the cable is mounted inside the aircraft and already plugged in the corresponding Pixhawk interface (see Fig. 5).[2] Verify, that the termination resistance is between the VN1630A/VN1640A box and the custom CAN cable and that the cable is plugged in channel 1 (see Fig. 6).

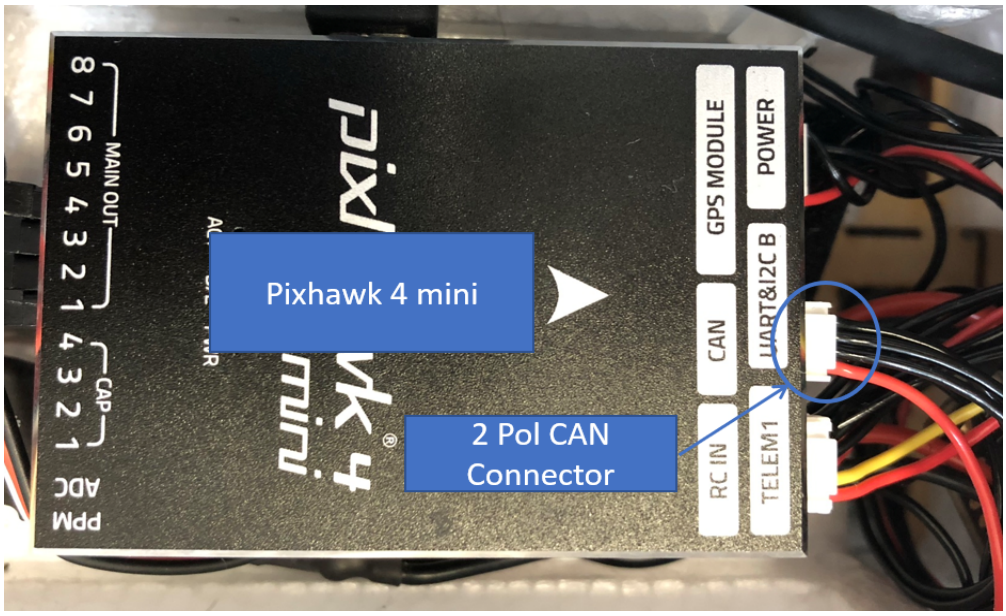


Figure 5: Pixhawk inside the UAXS.

[2] If you ordered the HiL as an upgrade to your existing Flaying Lab, you have to connect the CAN cable to the Pixhawk on your own.

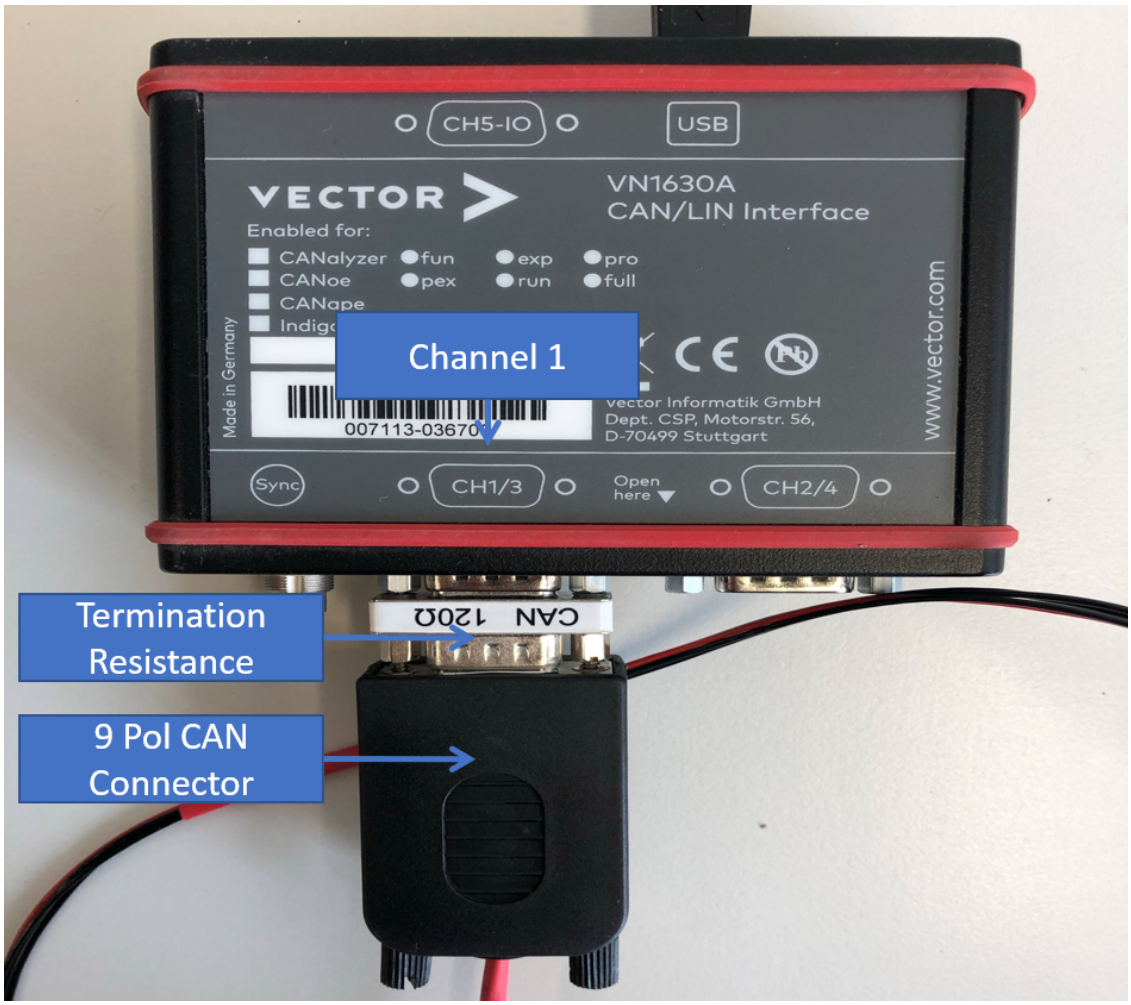


Figure 6: CAN Cable Connection to VN1630A. (*VN1640A similar.*)

4. Establish the connection between remote control and the Pixhawk. To do so, turn on the remote control by pressing the Power button until the start screen appears. If a warning appears, press the right button to continue. Now, select the pre-configured `UAXS HiL model` as the operating model. This is done through pressing the left central button and selecting the desired model with the right wheel. Figure 7 shows the selection menu as well as the remote control. After the selection, the model name has to be marked. Now, press the right button and choose `Select model` in the opening menu. Then press the `Exit` button two times. Now, the model name (*UAXS HiL*) and the battery status should be displayed on the screen. To check the connection on the Pixhawk side, verify that the LED of the RC antenna lights up. This LED lays inside the head of the aircraft and is shown also in Fig. 3. The remote control is now interacting with the Pixhawk.

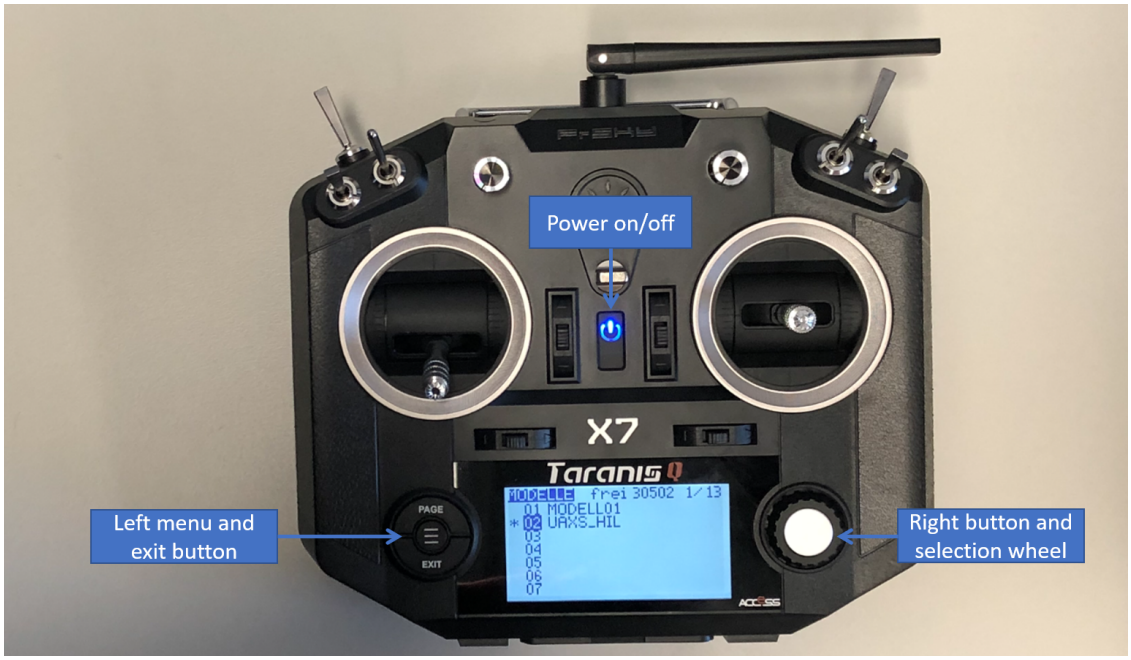


Figure 7: Remote Control.

1.2.2 Control Law Implementation

The direct law is already implemented as flight controller. If you want to use it for the HiL, you can skip to the next section.

If you want to implement your own flight controller, you have to flash the desired code for testing on the Pixhawk.[3] The following steps are required:

1. Start VMware Workstation Player and open the supplied ubuntu virtual machine (VM). It is located inside the `VM HIL` folder. Unlock the ubuntu user with the following log in data:

   ```
   USER: HIL_user
   PW:   HIL_flyinglab
   ```

2. Open the workspace in ubuntu and go to the determined folder for the flight controller. The path is given as follows:
   `Talon/Firmware_Ctr/src/modules/flight_control`

   After opening the folder `flight_control`, replace the old source and header files with the new desired ones (automatically generated by MATLAB). You may have to adjust the names of the files to use for compilation in `CMakeLists.txt`.

3. Compile the code. To do so, open the terminal and type these commands:

   ```
   cd ~/Talon/Firmware_Ctr
   sudo make px4_fmu-v5_hil upload
   ```

After the compilation of the code the command stops and waits for the bootloader of the Pixhawk. To continue the flash process, the micro USB connection from the Pixhawk has to be unplugged and replugged again. The expected console output is shown in Fig. 8.[4]

[3] The code generation procedure itself is explained in detail in the manual of the UAXS.

[4] Note that the exact command in this figure differs from the description.

```
px4@ubuntu:~/Talon/Firmware$ cd ~/Talon/Firmware
px4@ubuntu:~/Talon/Firmware$ make px4_fmu-v5_fixedwing upload
[0/1] Re-running CMake...
-- PX4 config file: /home/px4/Talon/Firmware/boards/px4/fmu-v5/fixedwing.cmake
-- PX4 config: px4_fmu-v5_fixedwing
-- PX4 platform: nuttx
-- PX4 lockstep: disabled
-- PX4 version: v1.9.2
-- cmake build type: MinSizeRel
-- PX4 ECL: Very lightweight Estimation & Control Library v1.9.0-rc1
-- Building and including px4_io-v2_default
-- ROMFS: px4fmu_common
-- ROMFS: Adding rc.board_defaults
-- ROMFS: Adding rc.board_sensors
-- Configuring done
-- Generating done
-- Build files have been written to: /home/px4/Talon/Firmware/build/px4_fmu-v5_fixedwing
[682/683] uploading px4
Loaded firmware for board id: 50,0 size: 1030216 bytes (49.90%), waiting for the bootloader...
```

Figure 8: Compilation of Source Code.

If the bootloader still can't be found, most probably the Pixhawk is connected to MS Windows and not to the VM. In this case, you have to change the connection status in VMware. The explicit procedure for this is described in Sec. 3.2. The whole PX4 source code with the controller on top is now flashed on the Pixhawk. If it was successful, the console will print out the text seen in Fig. 9.

```
Found board id: 50,0 bootloader version: 5 on /dev/serial/by-id/usb-3D_Robotics_PX4_BL_FMU_v5.x_0-if00
sn: 002500223438510a39303535
chip: 10016451
family: STM32F7[6|7]x
revision: Z
flash: 2064384 bytes
Windowed mode: False

Erase  : [====================] 100.0%
Program: [====================] 100.0%
Verify : [====================] 100.0%
Rebooting. Elapsed Time 23.008
```

Figure 9: Uploading the Source Code.

After successful completion, the VM can be minimized or closed.

1.2.3 QGroundControl

QGroundControl (QGC) is a virtual ground station for monitoring the flight mission of the aircraft. It needs to be installed and opened in MS Windows.[5] There are two different options to connect the aircraft to QGC. The first option is via USB. To do so, change the connection settings of the aircraft in the VM from `VM` to `Host` (detailed information are provided in Sec. 3.2). The second option is to connect the telemetry transceiver via USB cable to the simulation host. After opening QGC in MS Windows, the aircraft should be connected to QGC. If this is not the case, restart QGC or unplug and replug the micro USB cable that connects the aircraft; alternatively, unplug and replug the telemetry transceiver. The top right corner in QGC shows the connection status (see Fig. 10).

[5]The installation procedure of QGC is explained in detail in the manual of the UAXS.

Figure 10: Connection Status in QGroundControl.

When the aircraft is connected, check if the pre-initialized sensors setting is set to the option `rotation_none`

Also, check

a) if the RC remote control is connected to the aircraft and

b) that the RC channels are displayed in the MAVLink Inspector (see Fig. 11). This procedure is typically only necessary if QGC was connected to another Pixhawk before, as this might affect the settings. For more information on these settings, review Sec. 6.1.

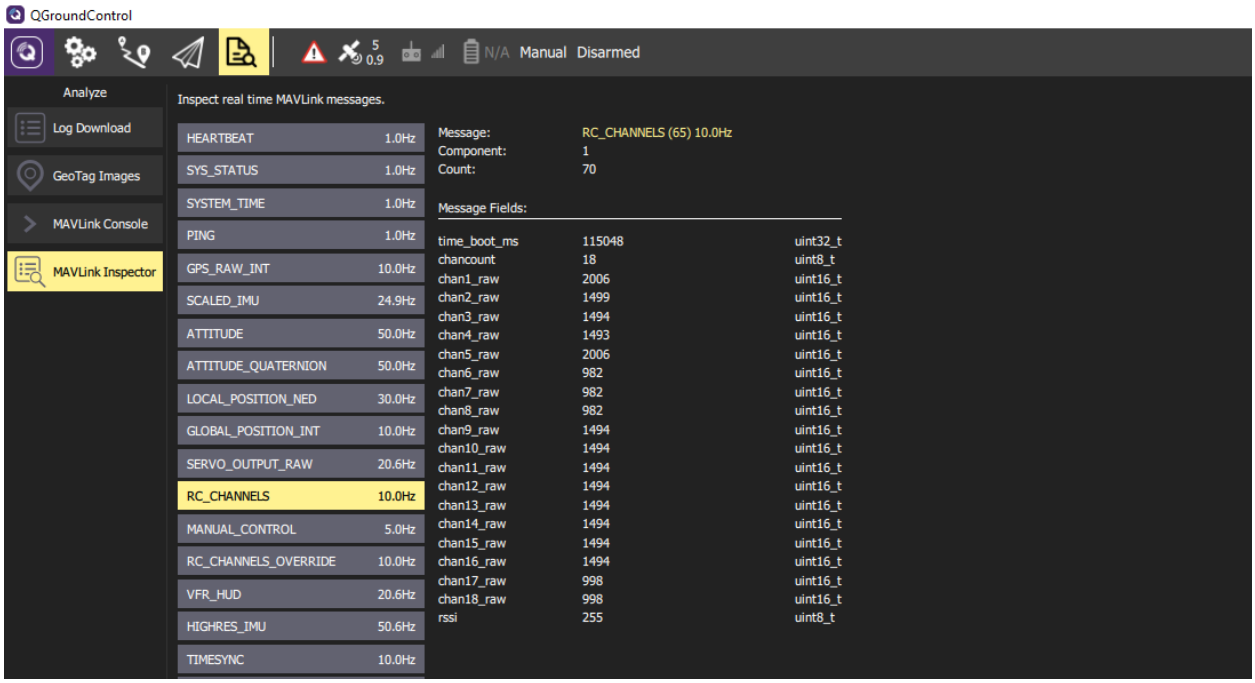


Figure 11: MAVLink Inspector.

1.2.4 Interface

The simulation is performed using the CAN bus controlled by the VN1630A/VN1640A. It is operated through the CANoe software or through Simulink directly.

CANoe Interface

After the software is installed, two steps are required to prepare the simulation:[6]

[6]The installation guide for CANoe is provided by Vector Informatik as part of the HiL.

1. Start CANoe in MS Windows.

2. The Graphical User Interface of CANoe will be shown. Upon first start, the CANoe Configuration has to be opened using the `File` tab and then selecting the `Open` option from the pop up menu. Select the desired configuration file from the `CANoe` folder (see Fig. 12). Then, the desired interface setup is ready; the fully configured setup for a running simulation is shown in Fig. 13.

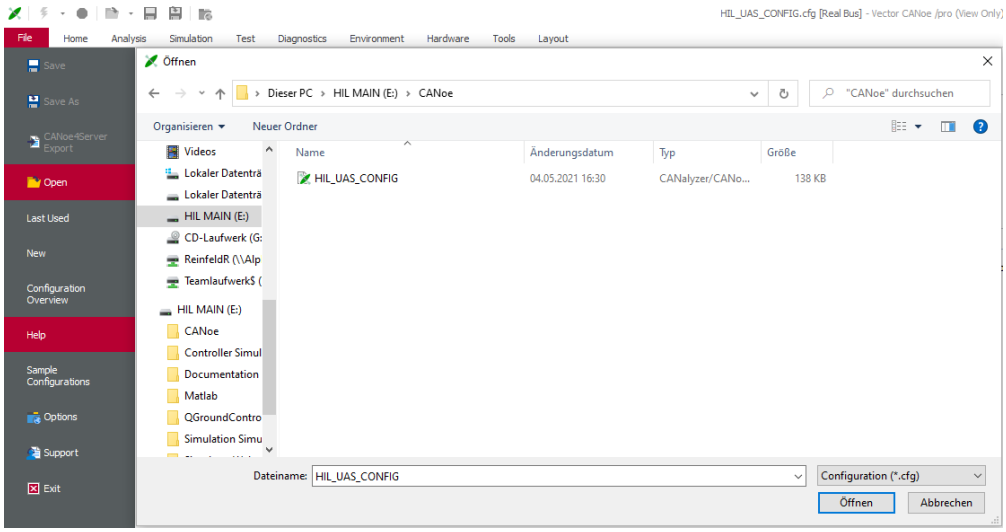


Figure 12: Procedure for Setting Up CANoe Configuration.

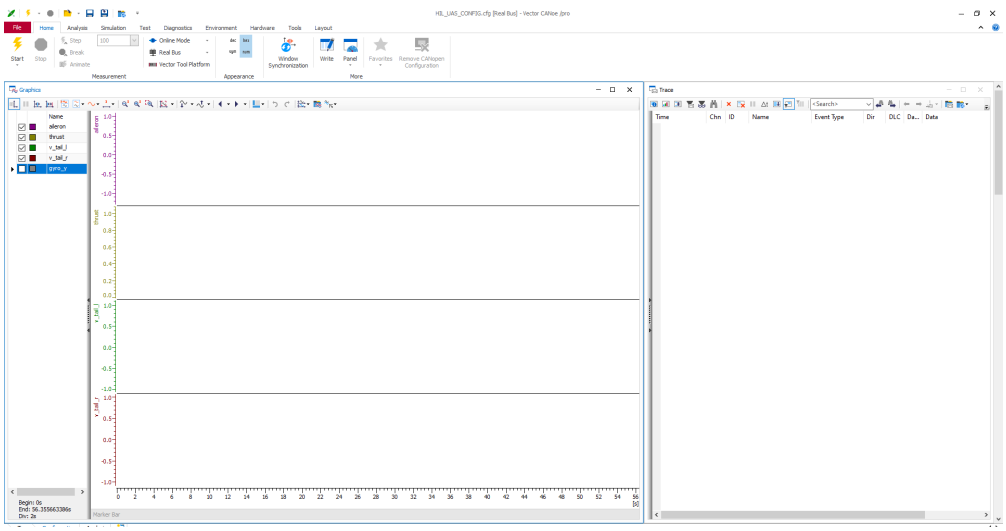


Figure 13: Desired CANoe Configuration.

The next time, CANoe can be opened without any additional setup procedure, because the configuration will be automatically selected.

Simulink CAN Interface

From 2023 on, the HiL is delivered with a 2021a Simulink model that directly uses the CAN interface of MATLAB. Therefore, the installation of CANoe is not required anymore. CANoe can still be used to analyze communication on the CAN bus.

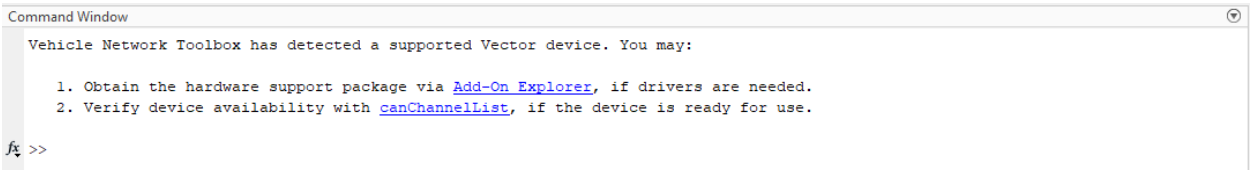


Figure 14: MATLAB Status Message Indicating a Connected CAN Interface.

Before opening the Simulation Model, ensure that the VN 1630/1640 CAN interface is connected to the computer and is detected by MATLAB (see Fig. 14). If it is not detected by MATLAB, ensure that all drivers have been installed correctly.
If you have accidentally opened and saved the Simulink model before the CAN interface was connected, you have to run the `CAN_Adjust.p` MATLAB script to reinitialize the CAN interface.

1.2.5 Simulation Model

The simulation of all the dynamics is done using a Simulink model. To start this model, the following steps are required:

1. Open MATLAB 2017 and change the working directory to the `Simulation Simulink` subfolder originally provided on the USB stick.

2. To initialize the simulation model, open and run the `init_sim.m` MATLAB script. Then open the `simmodel.slx` Simulink model, and wait until the window with the model opens. Simulink is now set up and prepared. Until now, do **NOT** press the Start button in Simulink; before you can start the whole simulation, a final step is required.

1.2.6 Virtual Flight Test Environment

The AlphaLink Virtual Flight Test Environment (VFTE) is a web browser-based simulation environment. A proprietary version exists for the HiL. To start it, follow these steps:

1. Open the Windows Command Prompt and change the working directory to the subfolder `Simulator Webserver` originally provided on the USB stick.

2. To start the VFTE, type

   ```
   node index.js
   ```

   After an initialization, the command prompt should print out the following confirmation of successful UDP and server start: `UDP and HTTP Server started.`
   Note: You should execute the commands with administration rights to avoid permission conflicts.

3. Open the browser and start the local host through typing the following address:

   ```
   localhost:8000
   ```

The VFTE should appear now awaiting the simulation to start. Go back to Simulink and press the `Start` button to launch the whole simulation. The VFTE shows the start in the perspective of the drone pilot: If you want to switch to a follow-me mode, press `p`. With the successful

start of the simulation, you can observe the movement of the aircraft in the VFTE and in QGC. Furthermore, you can observe the data traffic from the interface in the CANoe window; in the default settings, the incoming actuator commands are plotted and the trafficking messages are registered in the trace.

2 CANoe Details

In this chapter, the details for the usage of CANoe will be explained. CANoe is a software from Vector Informatik. Together with the VN1630A box, it provides the interface for the data link layer of the CAN bus to communicate with the simulation host. The software also provides some useful tools to manage and control the CAN bus. The following sections will guide you through the basic configuration of CANoe afterwards.

Note: Here, it is assumed that you have already successfully installed CANoe and activated your license. If not, the CANoe Installation Quick Start is recommended for setting up the software.

2.1 Connection to Simulink

The simulation data are exchanged between CANoe and Simulink via the `CANoe Simulink Interface`. This interface has to be installed using a file provided by Vector Informatik. Further Requirements are MATLAB 2017b and the Mingw-w64 Compiler in the MATLAB compiler settings.[7] For the installation, the following procedure is recommended:

1. Navigate to the folder, where you installed CANoe (e.g. `/Program Files/Vector CANoe 14.0`) and open the install file `Vector_AddOn_Matlab_Interface_V632` under `Installer Additional Components/Matlab`.
   Note: You should execute the command with administration rights to avoid permission conflicts.

2. Follow the instructions from the installer. Choose MATLAB 2017b as the installation environment because the installer does currently not recognize any newer version. After successful installation, the program will close itself.

3. Restart your computer. Then open Simulink in MATLAB 2017b. If the installation was successful, a new category in the Simulink library will appear called *CANoe*. In this case, the CANoe input and output blocks of the provided simulation model will be recognized by Simulink.

2.2 Configuration

First, open the configuration of CANoe. To do so, select `File` tab in the top left corner in CANoe and choose the option `Open`. A file navigation will open. Select the `HIL_UAS_CONFIG.cfg` file from the `CANoe` subfolder. The configuration file defines all parameters and settings concerning the CAN bus, which acts as the simulation interface. The software is handled through a Graphical User Interface (GUI).

2.2.1 GUI Setup

The default layout for the regular use is shown in Fig. 15. The `Graphics` window has to be closed to reveal the basic GUI setup. The GUI of the default configuration consists of three main windows (see Fig. 15).

[7] Installation of the compiler has to be done inside MATLAB.

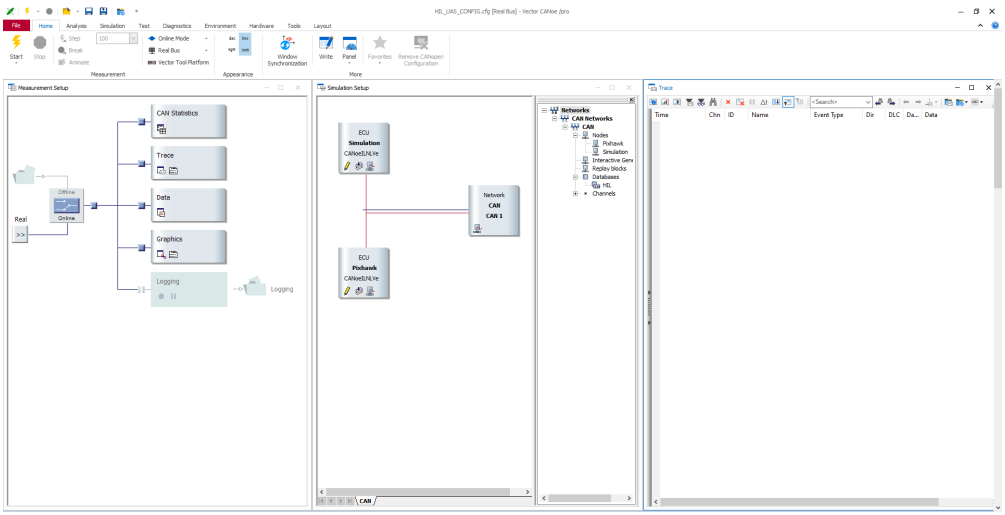


Figure 15: CANoe GUI.

1. **Measurement Setup:** This setup gives you the opportunity to access tools to see the statistics of the measurement or the involved data (see a detailed view in Fig. 16). It also offers the possibility to plot them in a live view. These tools will be further examined in the next subsection. Using the switch in the center, you can choose between taking the real data from the bus into account or doing a completely virtual simulation with logged data from a previous measurement. The latter one is useful to analyze and repeat certain situations.

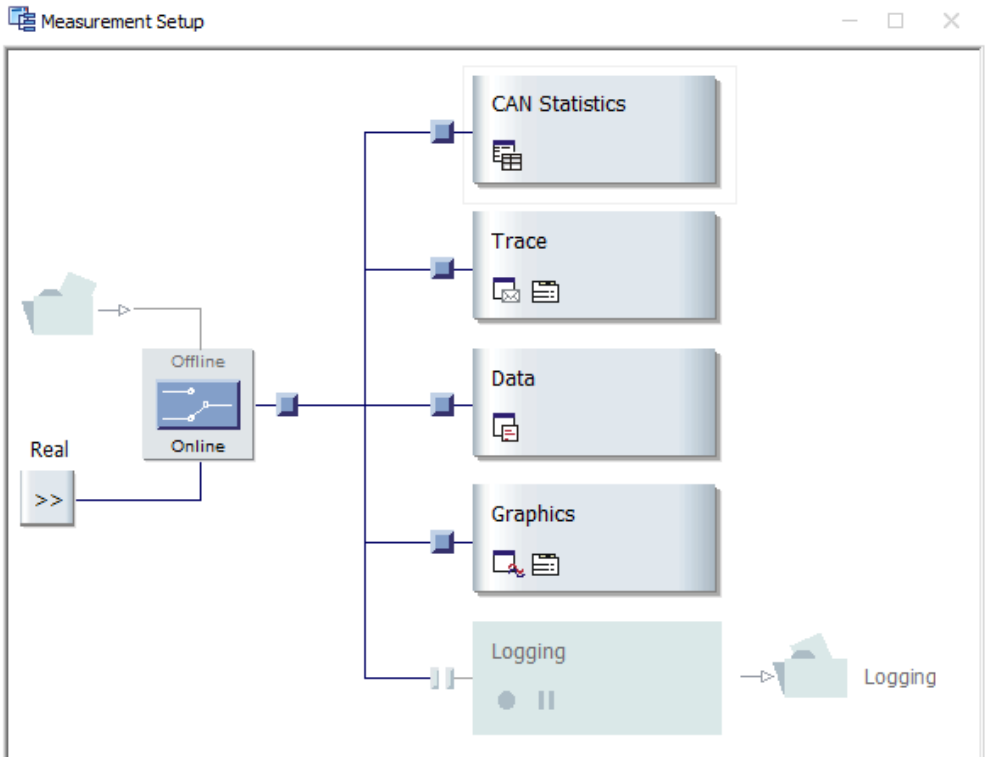


Figure 16: CANoe Measurement Setup Window.

2. **Simulation Setup:** You can adjust the parameters of the CAN bus and also the participants on the bus in this setup (see a detailed view in Fig. 17). The participants are represented as nodes, with the `Simulation` node being the simulation host and the `Pixhawk` node being the aircraft. To change the bus setting, double click the box with the `Network/CAN/CAN 1` node in the setup. The `Network Hardware Configuration` Window will open (see Fig. 18).

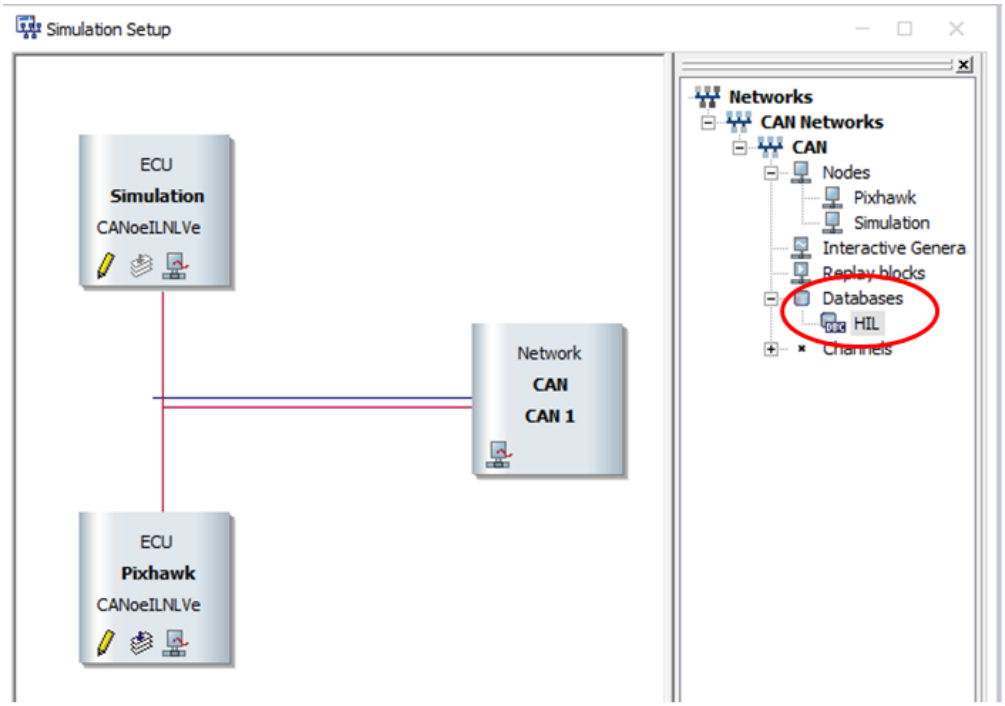


Figure 17: CANoe Simulation Setup Window.

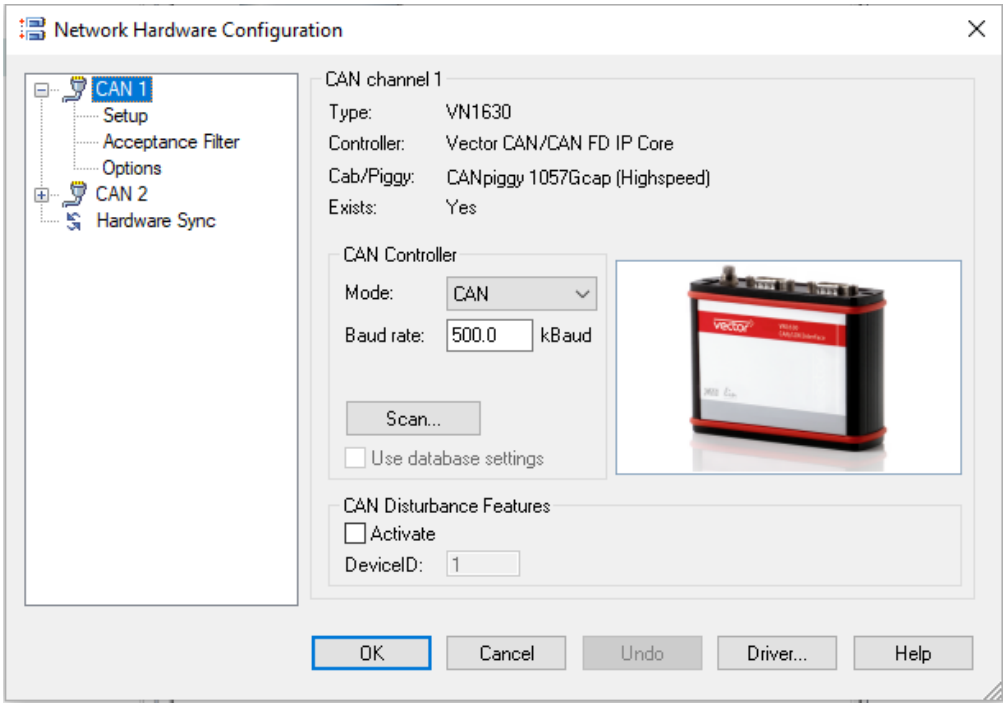


Figure 18: CANoe Network Hardware Configuration Window.

This window includes the menu options for the bus speed and sampling point. For the HiL, the baud rate must be set to $500\,\text{kBaud}$ and the sampling point to $62\,\%$. It also contains the so-called self acknowledgment (*TX Self-ACK*) option.[8] This means that a message does not require having a receiver to acknowledge it. In turn, the messages do not get stuck if the bus is waiting for a receiver. This option is useful if the CAN message is aimed to be seen through a digital analyzer.

3. **Trace:** This window lists all the incoming and outgoing messages with their corresponding timestamp (see Fig. 19). The depicted messages are defined through a database that is added in the simulation setup. Vector is using a proprietary database system called DBC.

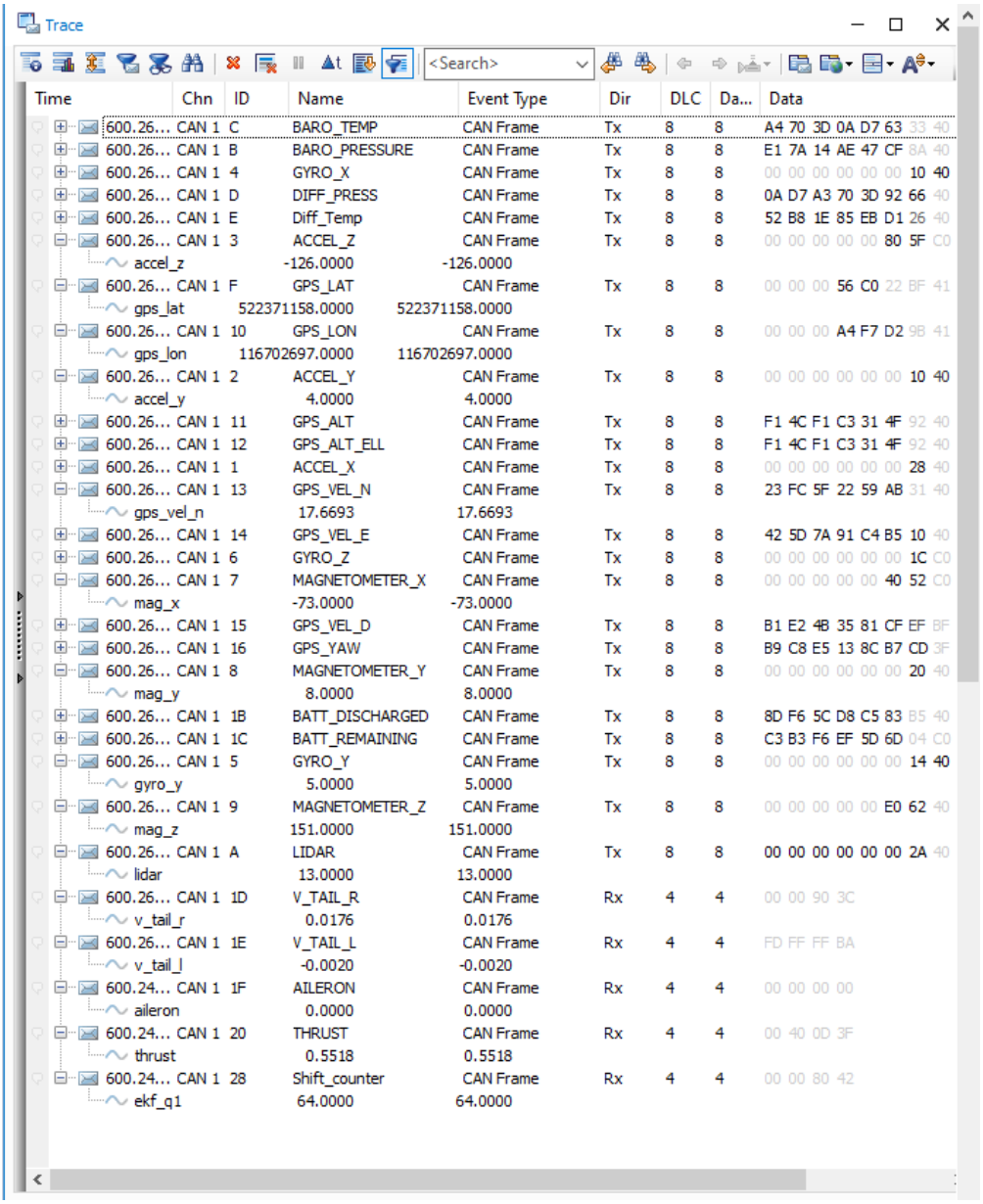


Figure 19: CANoe Trace Window.

[8]Sampling point and self acknowledgment option can be accessed though selecting the `Setup` submenu in the `Network Hardware Configuration` window (see Fig. 18).

2.2.2 DBC File

The configuration comes with a linked DBC file labeled `HIL`. This DBC defines all parameters of the messages that are exchanged between the Pixhawk and the simulation host. You can open it through double clicking its icon in the `Simulation Setup` window (see Fig. 17). The CANdb++ Editor is required; it is included in the successful installation of CANoe. The opened database can be seen in Fig. 20.

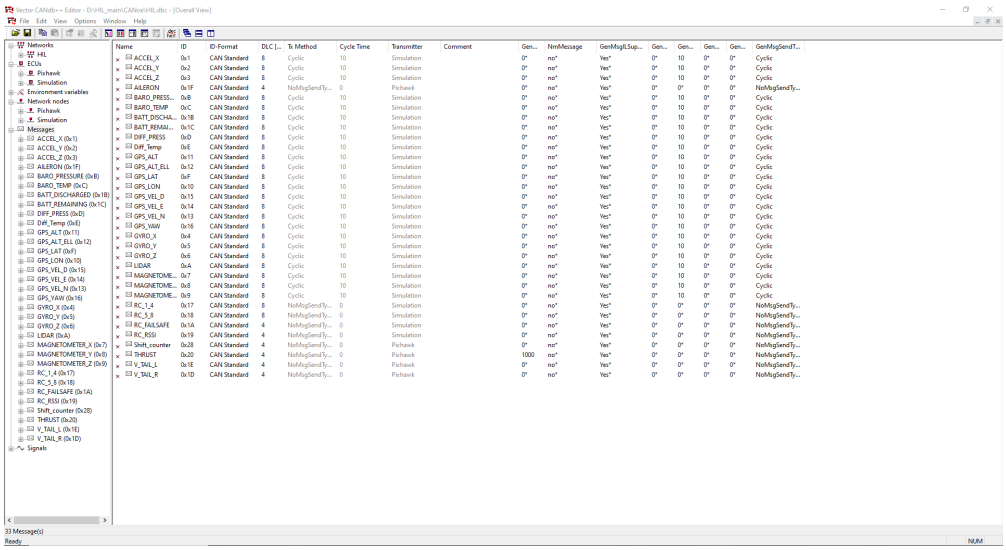


Figure 20: DBC HIL.

The starting point of the definition is the network that is called *HIL*. This Network has two so-called electronic control units (ECU): the simulation host called `Simulation` and the Pixhawk. Both are represented by one node in this network (see Fig. 20). Their corresponding messages are assigned to these nodes. This means the simulation host (the `Simulation` node) transmits the sensor values while the `Pixhawk` node transmits the actuator commands.

A message is defined by one CAN frame with a payload of $64\,\mathrm{bit}$. Because of that, one frame can carry multiple values called *signals*, if it is in the range of these values (e.g. eight `uint8_t`). Except for the actuator commands (that are floats), all values are given as a double. Hence, only one frame can carry one signal. This is why signals and corresponding message names only differ in the capitalization. In the signal part, the data type, bit order, and start bit of the value will be defined. In the message definitions, the important settings such as the DLC and the MSGID (message ID) are given. Further, you can set a transfer method and the repetition rate of the signal. This is used for the simulation to transmit the data coming from Simulink.[9]

2.3 Tools

CANoe provides various tools for a rich analysis of the measured data. One of the very useful ones is the `Graphics` tool. It can be accessed through the `Measurement Setup`. With this tool, you can plot data in real time (see Fig. 21).

[9]To add a new message or alter the configuration, please see the CANoe instructions provided by Vector Informatik.

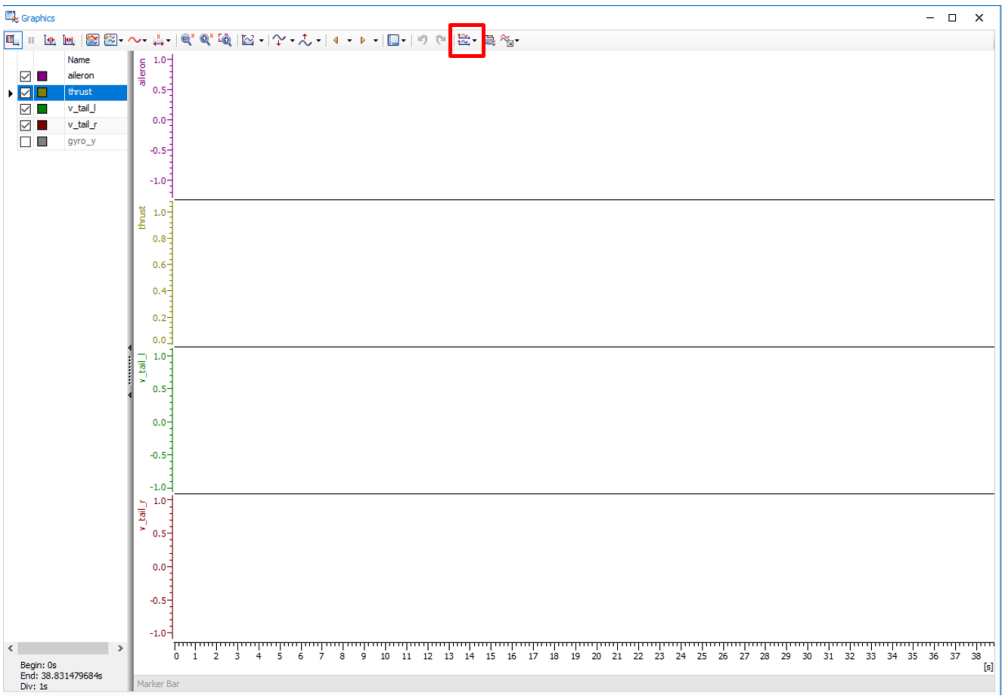
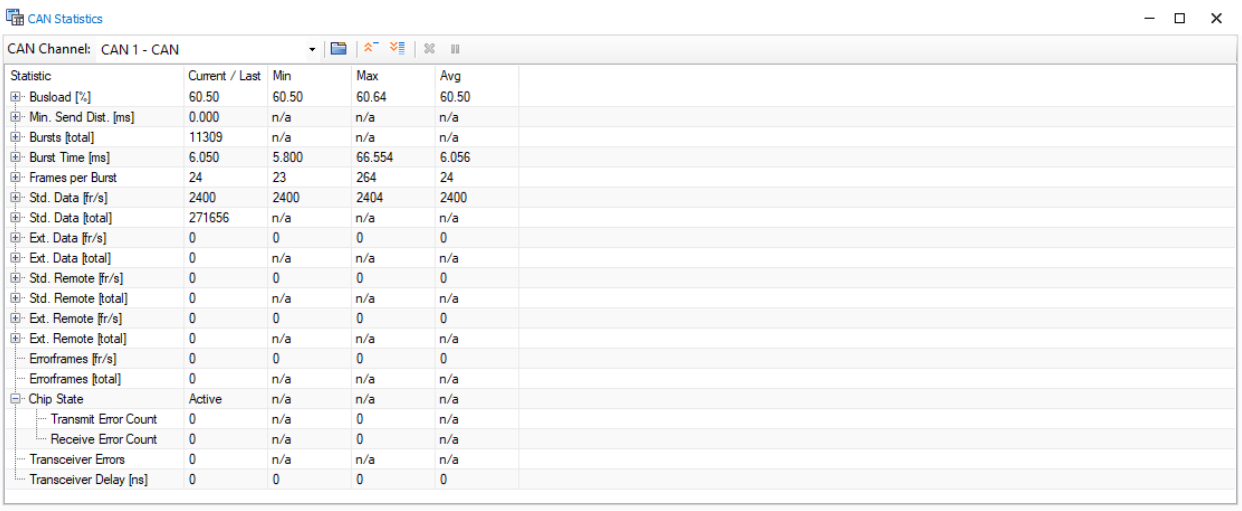


Figure 21: CANoe Graphics Window.

To add a signal or delete it from the plot, click the right mouse button and chose from the window. For adding the signal, a menu with the DBC appears, where you can select a signal from a message. The selected signal is then added to the left sidebar; to deactivate it, uncheck the box.

Another useful feature in the `Graphics` tool is `Plot separately`, marked in red in Fig. 21. Through this setting, all signals get plotted separately. A further useful tool are the `CAN Statistics` that are also accessed through the `Measurement Setup`. The statistic shows all relevant parameters for monitoring the bus and also provide insights if transmission errors happen (see Fig. 22 for a sample measurement).



Statistic	Current / Last	Min	Max	Avg
Busload [%]	60.50	60.50	60.64	60.50
Min. Send Dist. [ms]	0.000	n/a	n/a	n/a
Bursts [total]	11309	n/a	n/a	n/a
Burst Time [ms]	6.050	5.800	66.554	6.056
Frames per Burst	24	23	264	24
Std. Data [fr/s]	2400	2400	2404	2400
Std. Data [total]	271656	n/a	n/a	n/a
Ext. Data [fr/s]	0	0	0	0
Ext. Data [total]	0	n/a	n/a	n/a
Std. Remote [fr/s]	0	0	0	0
Std. Remote [total]	0	n/a	n/a	n/a
Ext. Remote [fr/s]	0	0	0	0
Ext. Remote [total]	0	n/a	n/a	n/a
Errorframes [fr/s]	0	0	0	0
Errorframes [total]	0	n/a	n/a	n/a
Chip State	Active	n/a	n/a	n/a
Transmit Error Count	0	n/a	0	n/a
Receive Error Count	0	n/a	0	n/a
Transceiver Errors	0	n/a	n/a	n/a
Transceiver Delay [ns]	0	0	0	0

Figure 22: CANoe CAN Statistics Window.

3 Virtual Machine Details

In contrast to the stand-alone Flying Lab, the coding and compiling environment for the HiL is located in the supplied Virtual Machine (VM) using Ubuntu. This is due to the bigger repository of tools in Linux that are needed to access the functionalities of the CAN bus communication. The following sections will therefore cover all information regarding the VM that are required for the handling of the HiL.

3.1 Installation

A VM is a simulation of a computer system inside a computer system. This means that for example that you can run a Linux Distribution on the same computer parallel to the host operating system, e.g. MS Windows. To run the supplied VM, the VM Workstation Player 15 of VMware is required. You can download it for free for non-commercial use.[10] After the download, install the player following the instructions provided by VMware. Then restart the computer and open the player (see Fig. 23).

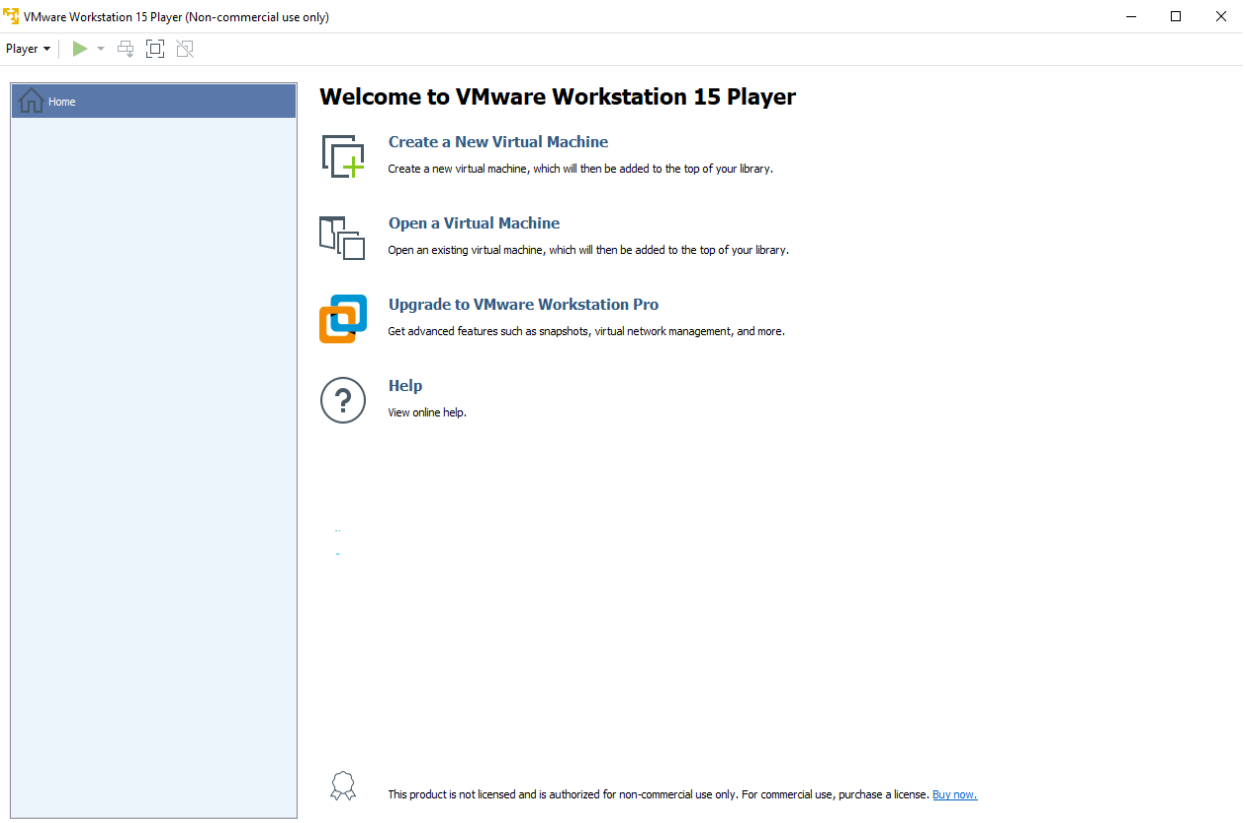


Figure 23: Launch Window VMware Workstation 15 Player.

Click `Open a Virtual Machine`. Navigate to the `VM HIL` folder and select the `HiL_px4_ubuntu` file (see Fig. 24).
Note: Remember that it is recommended to copy the relevant files from the USB stick to your hard drive first.)

[10] `https://my.vmware.com/de/web/vmware/downloads/info/slug/desktop_end_user_computing/vmware_workstation_player/15_0`.

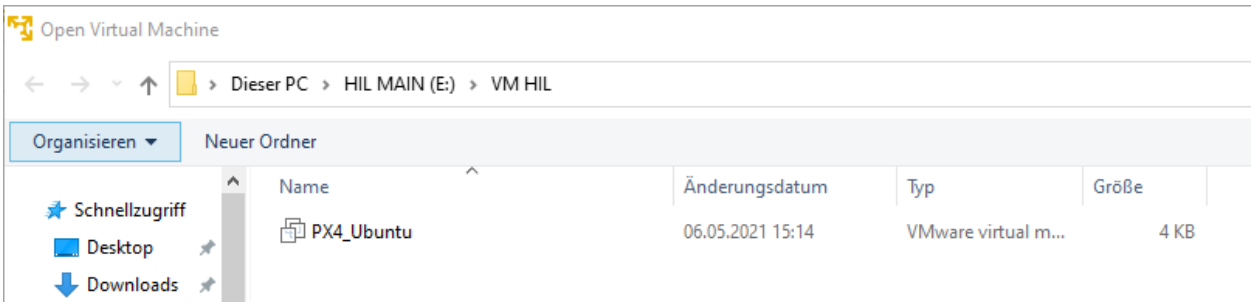


Figure 24: File Navigator for VM.

The selected VM is then shown in the VMware Workstation Player. Press `Play virtual machine` to start the VM.
The notification seen in Fig. 25 gives the information that the VM is in a suspended state. This means, the ubuntu system will start right at the point of the last usage, i.e. the loading time is minimized and it gives huge flexibility because all opened projects and files stay restored.

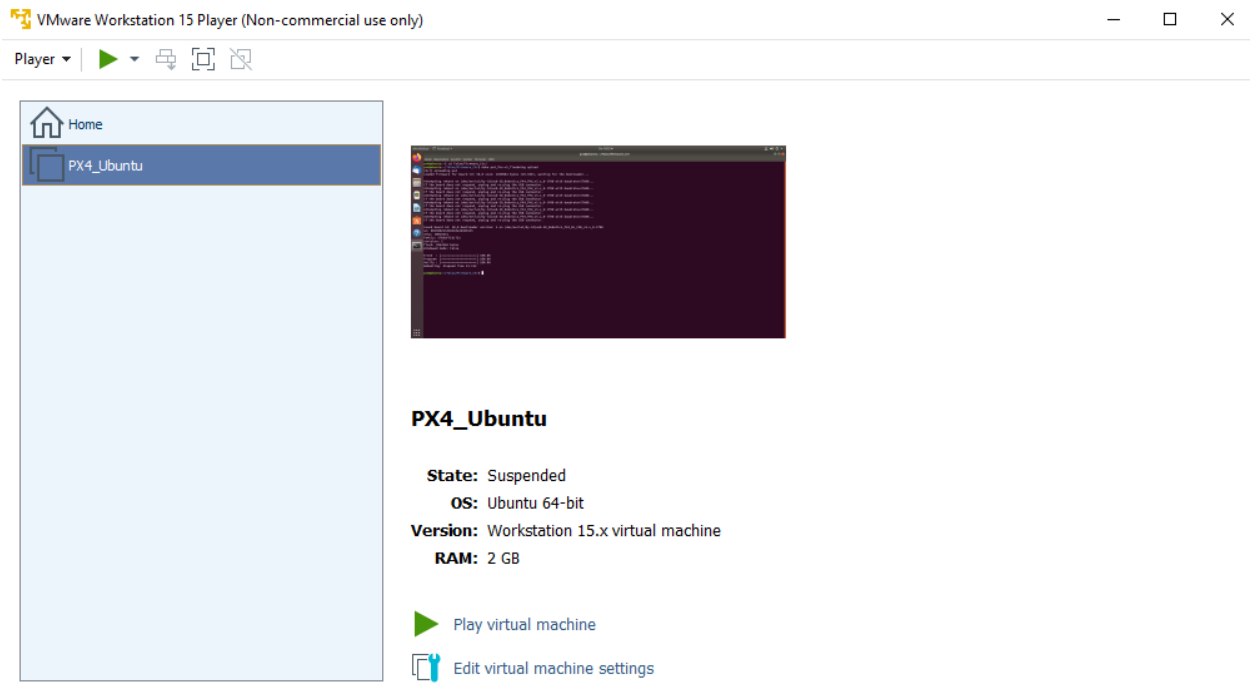


Figure 25: Launch Window VMware Workstation Player.

After restarting, the system requests the user to log in. The log-in information for this VM are given as follows:

```
USER: HIL_user
PW:   HIL_flyinglab
```

This password is also requested after executing a command with `sudo` (e.g. used in the next section).

3.2 Working with the Virtual Machine

Inside the VM, you find the coding environment for the PX4 flight controller. To implement the controller and compile and flash the source code, you can use the terminal and the file man-

ager. The starting point is the controller structure that was implemented using the AlphaLink controller template. The MATLAB `Embedded Coder` generates the necessary C++ source and header files. This code now has to be integrated in the PX4 source code by pasting it in the corresponding `flight_control` folder with all the necessary source code for the controller (see Fig. 26).[11]

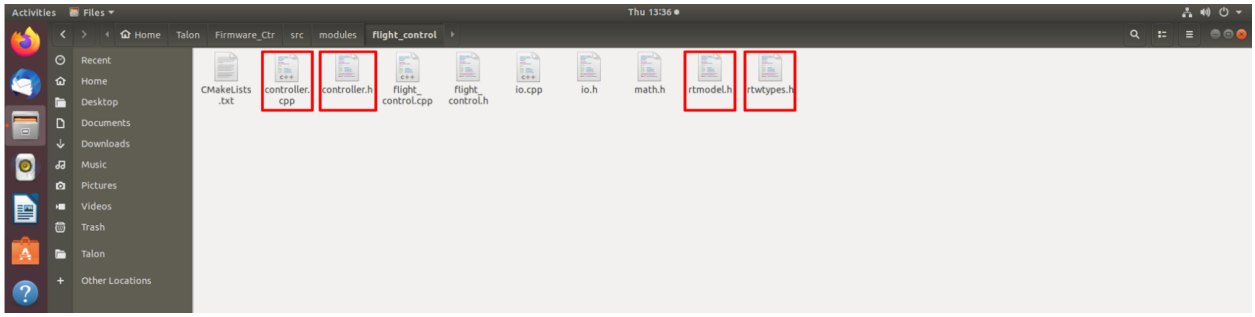


Figure 26: Path to the Relevant Flight Controller Folder.

All the generated C++ and header files will replace their old counterparts. Check if all the names match with the names of the files marked in red in Fig. 26.

To compile and upload the code, the terminal is needed. You can open it through clicking the black icon with the white arrow in it. In the terminal, three basic commands are important for the process:

- `cd` - change the directory,
- `ls` - list all the files and folders from the current directory, and
- `sudo make` - run a makefile as superuser.

To compile the code, navigate to the development directory. This is done through the following command:

```
cd ~/Talon/Firmware_Ctr
```

Now, the command position is located in the correct directory. To compile and flash the code, the following three commands are required:

```
sudo make px4_fmu-v5_hil clean
sudo make px4_fmu-v5_hil
sudo make px4_fmu-v5_hil upload
```

With the `make [file name] clean` command, the previous code is erased to avoid potential errors. The second command compiles the code (compare Fig. 8 in the quick-start guide). The last command finally flashes the code on the Pixhawk flight controller. If the bootloader cannot find any device, then the Pixhawk may be connected to the host operating system (MS Windows). To change that setting, click right on the small icon in the upper right corner of the VMware player. Then choose `Disconnect from host` (see Fig. 27). The icon will now light up in green and appear in a brighter shape (see Fig. 28), indicating that it is connected to the VM.

[11] Complete path: `Talon/Firmware_Ctr/src/modules/flight_control`.

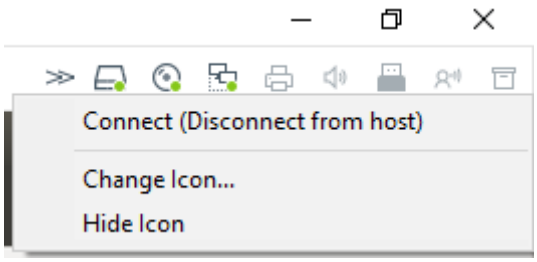


Figure 27: Disconnecting the Pixhawk from MS Windows Computer.

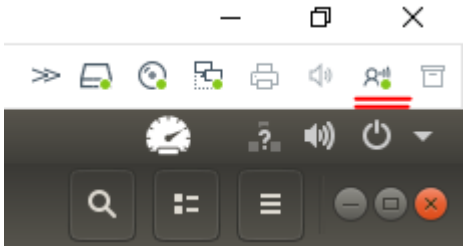


Figure 28: Connecting the Pixhawk to the Virtual Machine.

3.3 Additional Information

The provided VM includes further pre-installed software. Among those, `Stacer` is an additional, convenient monitoring tool to keep the VM clean. Over time, a bulk of temporary data can bloat up all the virtual storage. This in turn can lead to a broken VM that then cannot be accessed anymore. `Stacer` offers a solution through cleaning all temporary data. You access it through clicking on the Brush icon in the application list (this list appears when you click the menu in the bottom left corner).

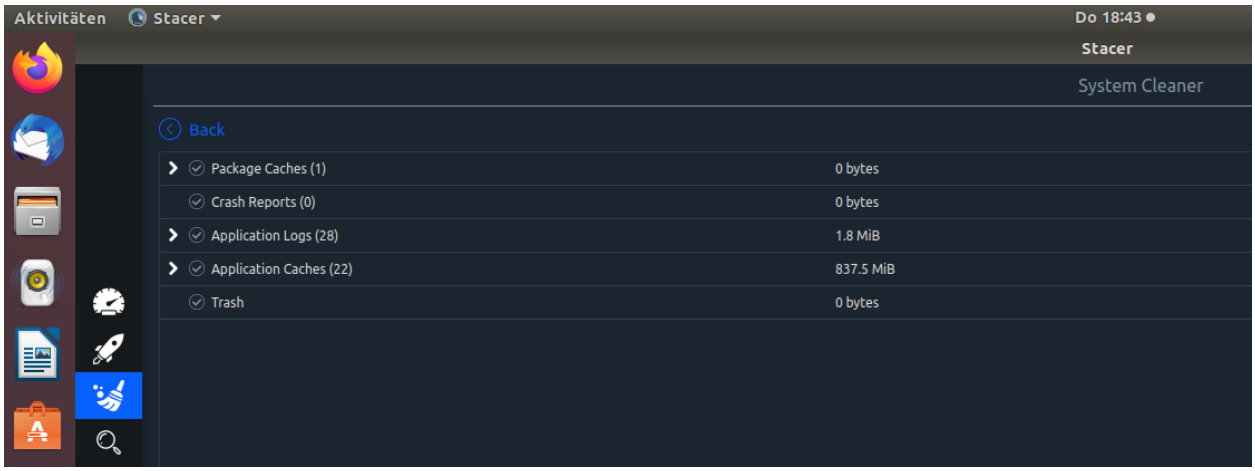


Figure 29: Cleaning with Stacer.

Figure 29 shows the files that can be deleted. To clean the storage, click the Brush button located at the bottom. Note that cleaning the applications data (`Applications Logs`, `Applications Caches`) may lead to performance decrease or undesired behavior of the applications. Hence, these files should only be cleaned with caution.

4 Simulink Model

This chapter provides an overview of the simulation setup and shows you how to manipulate sensor data.

4.1 Overview

The flight simulation is done using a Simulink model called `simmodel.slx`.To initialize this model with the desired parameters, the `init_sim.m` script is automatically executed as an initialization function when the simulation is started. You can change the `init_sim.m` script in MATLAB or specify a different initialization function in Simulink (`Property Inspector` > `Callbacks` > `InitFcn`). Furthermore, the `trim_routine.m` script can be used to calculate trim values for the control surfaces, thrust, and angle of attack for any flight condition. For this, an airspeed and flight path angle have to be specified in the `trim_routine.m` script. A `.mat` file with the new trim values will be created automatically. This file can be implemented in the `init_sim.m` script and, hence, a new flight state is used within the nonlinear simulation. [12]

The initialization script, the trim routine, and also the simulation model are all located in the folder `Simulation Simulink` on the provided USB stick. The model can be basically separated in six components (see Fig. 30).

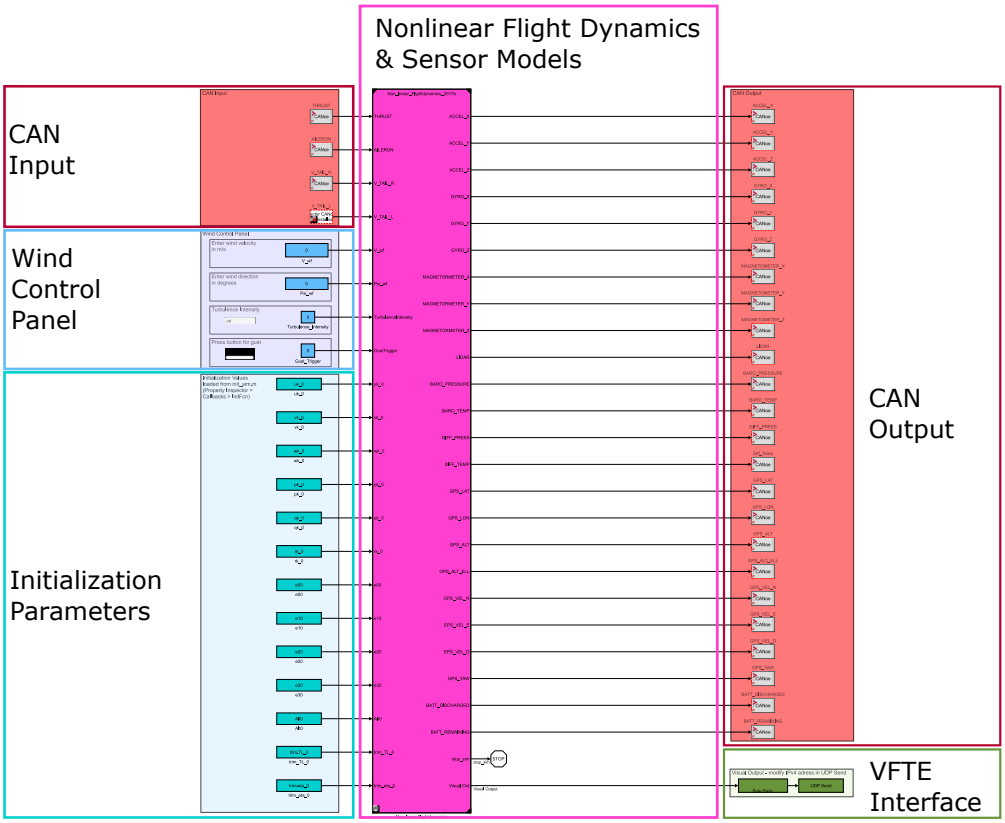


Figure 30: Overview Simulink Model.

[12] To do this, the new `.mat` file must first be copied to the `Simulation_Simulink` folder. Then the name of the new `.mat` file can be specified in line 30 of the `init_sim.m` script and the code can be activated by removing the comment function (deleting the % symbol).

1. **CAN Input:** The first component (left, top) is the system input. This component receives the command signals from the aircraft and forwards them to the nonlinear simulation model.

2. **Wind Control Panel:** The second component (left, center) allows to modify atmospheric parameters. Stationary wind speed and direction as well as gusts and turbulence can be set.

3. **Initialization Parameters:** The third component (left, bottom) passes the initialization parameters for the simulation, which are loaded by default from the `init_sim.m` script, to the nonlinear simulation model.

4. **Nonlinear Flight Dynamics & Sensor Models:** The center component computes the overall flight dynamics and sensor models. This component relies on several modules. It calculates the associated behavior of the control surfaces and the thrust. The external forces and moments are calculated. Using these calculated forces and moments as input, the equations of motion for the aircraft are solveed. Finally, the flight mechanical parameters like the Euler angles, the airspeed, the altitude etc. are converted to those values that the sensors of the aircraft would measure under the simulated flight conditions. This is done through adding the identified sensor dynamics and the noise.

   Table 1 lists the sensors inside of the aircraft that are simulated.

   Table 1: Overview of Simulated Sensors inside the Aircraft.

Sensor	Simulated Values
Accelerometer	accel_x, accel_y, accel_z
Gyro	gyro_x, gyro_y, gyro_z
Magnetometer	mag_x, mag_y, mag_z
LiDAR	lidar
Barometer	baro_press, baro_temp
Differential Pressure Sensor	diff_press, diff_temp
GPS	gps_lat, gps_lon, gps_alt, gps_alt_ell, gps_vel_n, gps_vel_e, gps_vel_d, gps_yaw

5. **CAN Output:** In this component (right, top), the simulated sensor data are transferred through the Simulink CANoe interface to CANoe. Over the CAN bus, they are then finally transferred to the aircraft.

6. **VFTE Interface** The last component (right, bottom) refers to the visualization and provides an interface to the VFTE. In this interface, all the required data for the visualization in the VFTE will be sent trough a UDP connection. Therefore, the signals are mixed over a `Mux` block and are converted to single bytes. Then, they are sent over UDP to a server, which hosts the VFTE in the local web browser. For more Information, please review Ch. 5.

4.2 Manipulation of Data

A significant advantage in HiL testing is the simulation of failure conditions in a safe environment. Failure conditions can be generated through sending deliberately wrong sensor values. The provided simulation model offers you an opportunity for this sensor data manipulation through adding a predefined subsystem to the output of the simulation model. Figure 31 shows how this subsystem is integrated.

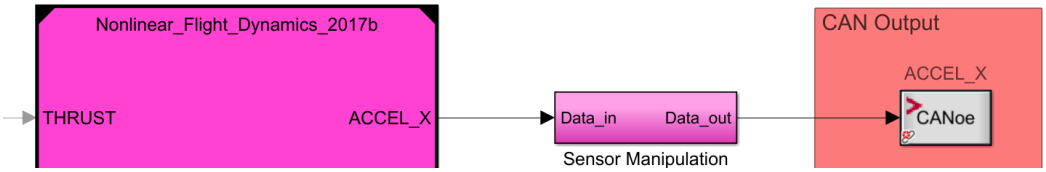


Figure 31: Integration of the Subsystem.

You can find this subsystem in the `Simulation Simulink` subfolder as `data_manip.slx` model; it is also included in the simulation model as a comment, so it is ignored. Inside this subsystem, there are two different ways of manipulating the incoming data (see Fig. 32):

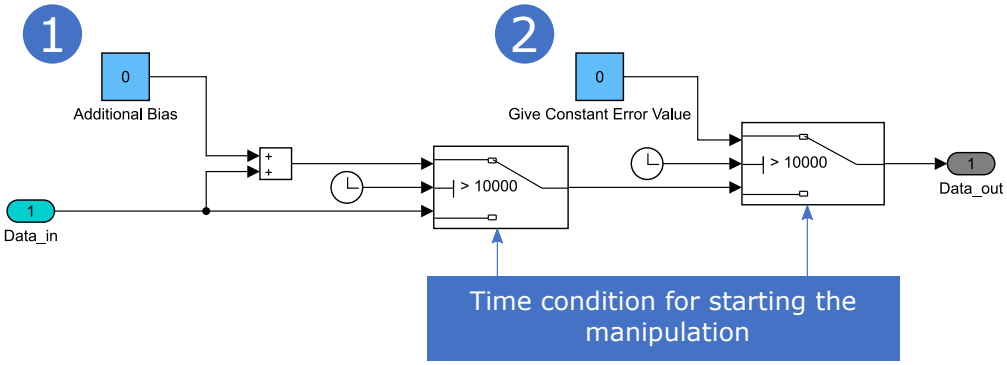


Figure 32: Subsystem of the Data Manipulation.

1. The first way is adding an additional bias to the signal. That way, the desired bias can be defined inside the given constant block.

2. The second way is sending a wrong constant value instead of the calculated value. To start this manipulation, two switches are integrated. In the first switch, a component is added to the signal (1 and 2) while in the second switch the signal is replaced by a constant value. Both have a switching condition that is linked to the simulation time. This gives you the opportunity to start the manipulation at a desired point in time. To set that time, you have to define a threshold inside this block.

   Let us look at one example: as seen in Fig. 32, the threshold is defined as 10,000. This means the switch would change from the unmanipulated lower input to the manipulated upper input at a simulation time of 10,000 seconds. The same applies to the second switch. If only the first type of manipulation is desired, i.e. where the error is added to the signal, the threshold of the second switch should be set to `inf`. This would result in only the first manipulation becoming active over time.

   Instead of using a constant value, you can use arbitrary inputs like a step for manipulation.

5 Node.js

This chapter provides an overview about the working principles of Node.js in combination with the Virtual Flight Test Environment (VFTE). It also discusses the configuration procedure and the start of the Node.js program in more detail.

5.1 Overview

The AlphaLink VFTE is used for the visualization of the simulated flight dynamics. It is a web-based visual, where the six degress of freedom are displayed through computer graphics. This requires a communication between the simulation model in Simulink and the VFTE to exchange the required parameters for the correct representation of the aircraft. This is done using Node.js. To install Node.js on MS Windows, the installer `node-v14.16.1-x64.msi` in the directory `Simulator Webserver` can be used with its default settings. [13]
The provided Node.js program creates a UDP server and a web server. The UDP server receives the simulation data from the Simulink model and transfers it to the web server, where the VFTE is hosted.

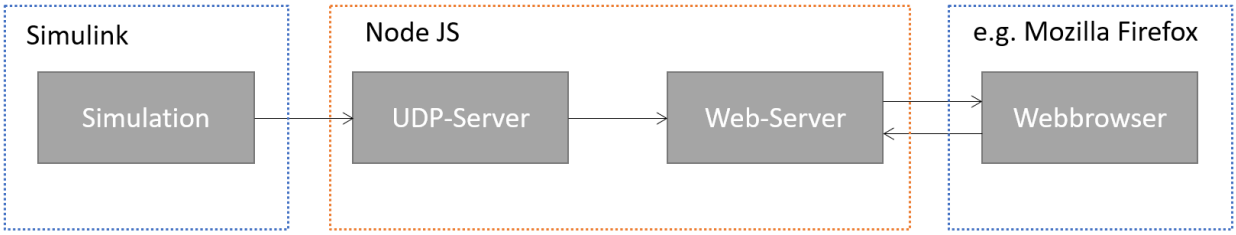


Figure 33: Scheme Node.js.

The web browser now accesses the hosted web server through the local host and processes the visualization so that the movement of the aircraft can be followed. Figure 33 illustrates the basic principle.

5.2 Configuration and Start

The UDP communication between Simulink and Node.js uses the port 7788 while the communication with the web server uses the port 8000. Node.js creates those ports using the local IP address of the simulation host. Depending on the local network of the computer, this address may vary. Hence, the given Simulink model broadcasts its UDP message to each possible IP address with an open 7788 port.[14] If the local address is known, the actual IP address indicated in the block transmitted through UDP can replace the broadcast settings. This is done inside the VFTE interface module in the Simulink model as shown in Fig. 34.

To start this interface, execute the provided Node.js application `index.js` in the command prompt (see the steps from the quick start described in Sec. 1.2.6). To access the web server from the browser, the local host with the port 8000 has to be selected. To do so, type

```
localhost:8000
```

[13] The latest version of Node.js can alternatively be downloaded from `https://nodejs.org/en/download/` for various operating systems.

[14] If a connection is not possible, the MS Windows Defender may block the UDP port.

in the address bar and the VFTE will be displayed (see. Fig. 35).

The ports used can be changed in the `index.js` file in the `Simulator Webserver` directory. The port of the UDP server is set in line 55 the; the port of the HTTP server is set in line 58. Subsequently, use the new UDP port in the UDP Block of the Simulink model; use the HTTP port when calling the VFTE in the web browser.

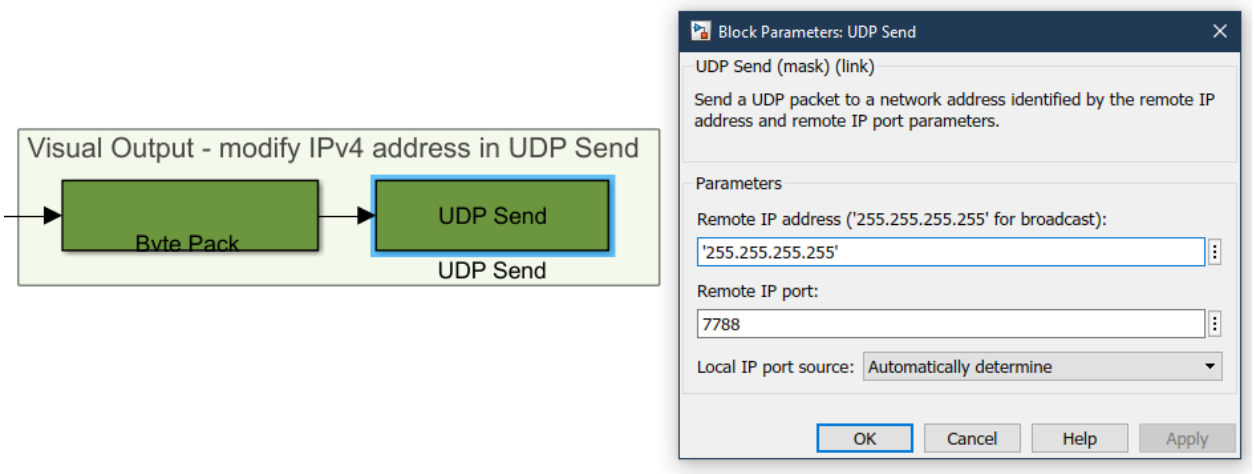


Figure 34: Settings of the Transmitted UDP Block.

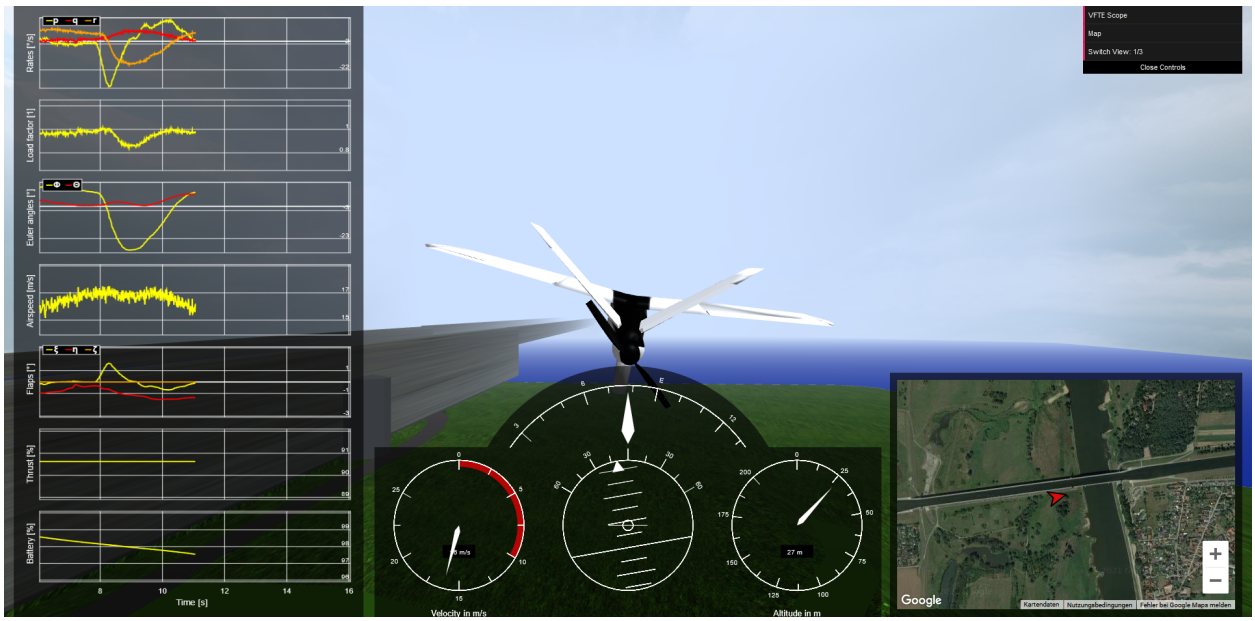


Figure 35: AlphaLink Virtual Flight Test Environment - Follow-Me Mode with Scope and Map.

6 QGroundControl

QGroundControl (QGC) is the default monitoring and ground control application for the aircraft within the PX4 flight controller environment. For the basic setup, install QGC with the given installer in the folder `QGroundControl` from the USB stick.[15]

A telemetry transceiver is integrated in the aircraft, because it uses telemetry in real flight tests and in the HiL simulation. The transceiver must be connected to the computer. In contrast to real flight tests, the aircraft can also communicate using a USB cable in HiL simulations. Hence, this cable also must be connected to the computer.
Note: If the VM is running, check that the flight controller is connected to the host. For more detailed information, please see Ch. 3.

There are two important interfaces in QGC:

1. The overall GUI, where the received flight parameters can be monitored and the binary log file of the mission can be downloaded; and
2. An inline of the shell of the operating system. The flight controller runs on that operating system.

6.1 Graphical User Interface

In the Graphical User Interface (GUI), you can monitor the mission, configure the aircraft and download the mission log from the flight controller.

6.1.1 Monitoring

There are two ways of monitoring the flight mission. Figure 36 shows the first way. You can access it through clicking on the icon representing a paper plane.

Figure 36: Standard Mission Monitor.

[15]The detailed installation procedure is explained in detail in the manual of the UAXS.

In this window, the flight path is plotted on a map and the position is continuously updated. In addition, a Primary Flight Display (PFD) is presented in the upper right corner. There, you can observe the bank angle and the pitch angle. The box below allows to list more parameters that will be displayed during the mission, e.g. the airspeed. A red arrow marks the position on the map and indicates the heading.

The second way to monitor the mission is the MAVLink Inspector. You can access it through clicking on the icon representing a sheet with lens and then choosing the `Inspector` (see Fig. 37).

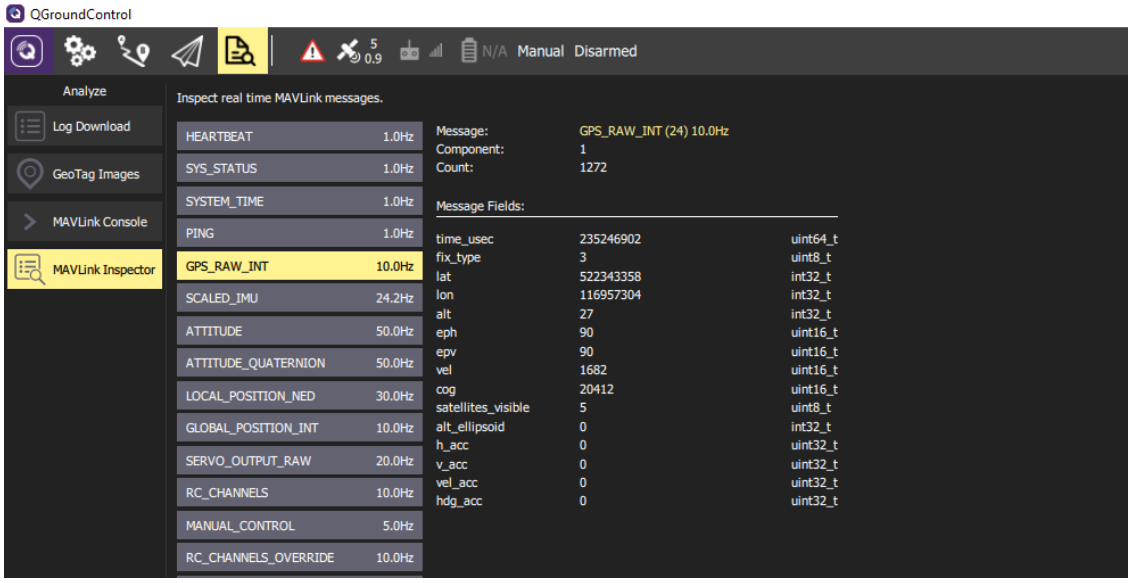


Figure 37: MAVLink Inspector.

There is a list of data transfers called *topics* (see Fig. 37). In these topics, all important sensor data and estimated data can be inspected. The topics are named after the content; also the update rate is given. For example, the topic labeled 'high resolution imu' gives insights into the acceleration and angular speed of the aircraft.

6.1.2 Configuration

The configuration and calibration is done in the setup menu. You reach the menu through clicking the `Settings` icon (see. Fig. 38).

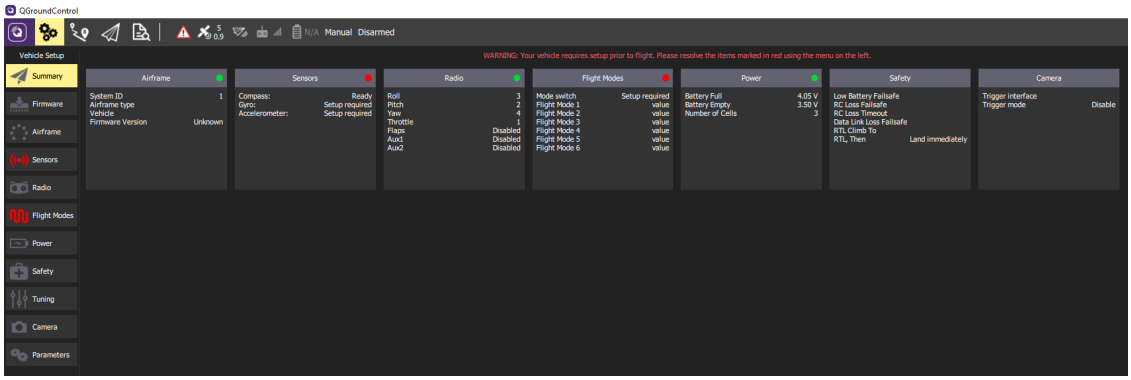


Figure 38: QGroundControl Configuration.

For the simulation, it is only important to check that there is no preconfigured rotation in the sensors setup. To do so, select the sensors setup and check whether `rotation_none` is selected for 'Compass', 'Gyro' and 'Accel' in the `Calibration` (see Fig. 39).

Figure 39: Sensor Setup Menu.

6.1.3 Log File

To download the log file, go to the `Log` window. You can find it in the same category as the `Inspector` (see Fig. 40).

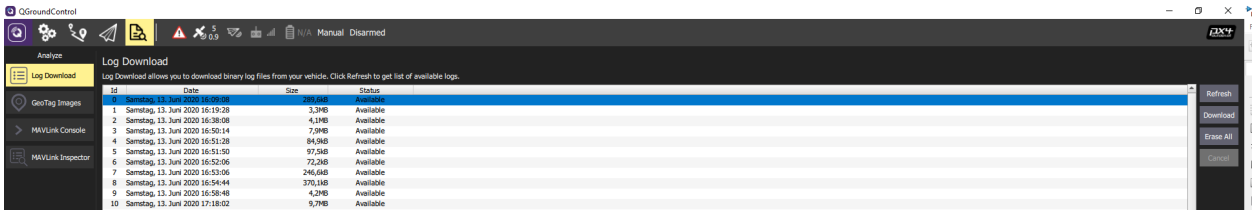


Figure 40: Log Download Menu.

The recorded logs are listed with a timestamp. Select the newest log and press the `Download` button. It may happen that the displayed time stamp of the log differs from the actual time. To identify the correct log, you may observe which file size is growing over time – this will be the desired file. The log file is provided as a `.ulg` binary file. To convert it to a `.mat` file, AlphaLink provides a dedicated program (see the UAXS Manual for a detailed description).

6.2 Nutshell

The flight controller runs on a real-time operating system (RTOS) called *NuttX* with a shell for interaction (*Nutshell*). This shell provides a console that you can access through the MAVLink console. You can find it in the same side menu like the `Log` file or the `Inspector` (see Fig. 41). For the actual monitoring, two commands are important. The first command is:

```
uorb top
```

This command lists all internal messages of the flight controller together with the update rate. The flight controller communicates internally through a so-called Object Request Broker (uORB). It can bee seen as an internal data bus that sends all the different data like sensor driver and estimators. Each message can be identified on the bus through its own topic (see Fig. 41).

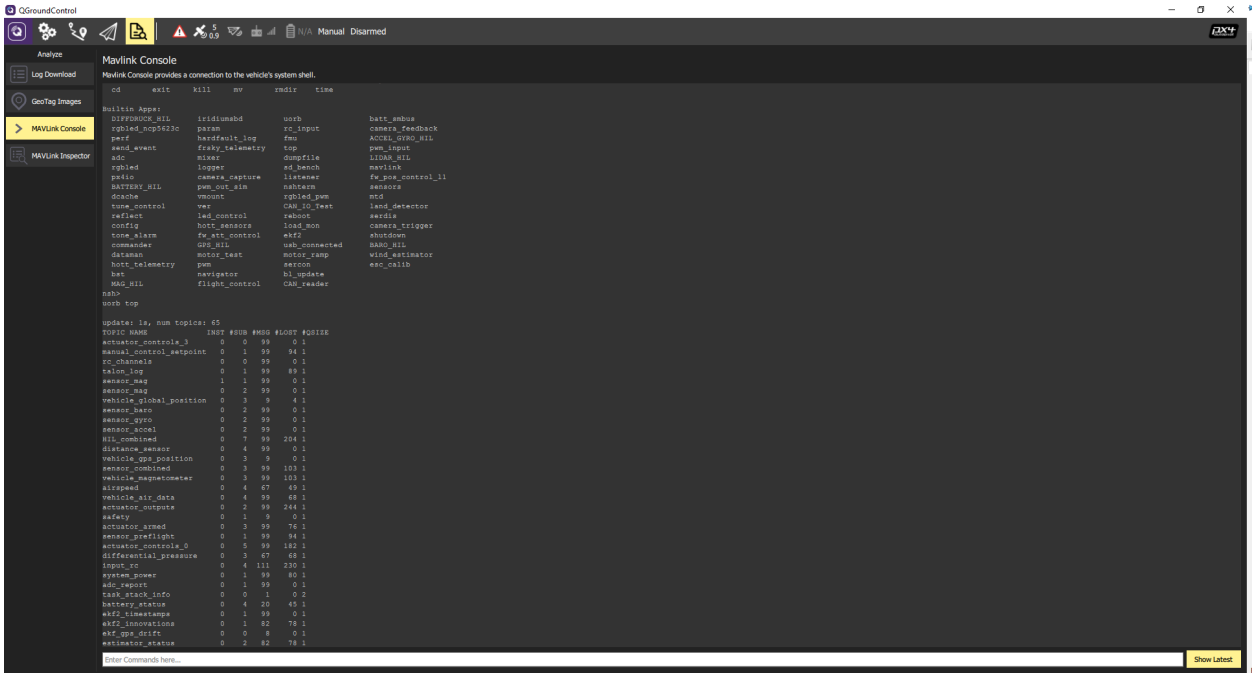


Figure 41: MAVLink Console.

You can see the content of the message through the command listener using the second command:

```
listener [topic name]
```

The argument for this command is the desired topic name. Replace `[topic name]` with the correct name of the topic as listed in the MAVLink Console. To inspect the barometric data such as static pressure or the temperature, for example, type:

```
listener vehicle_air_data
```

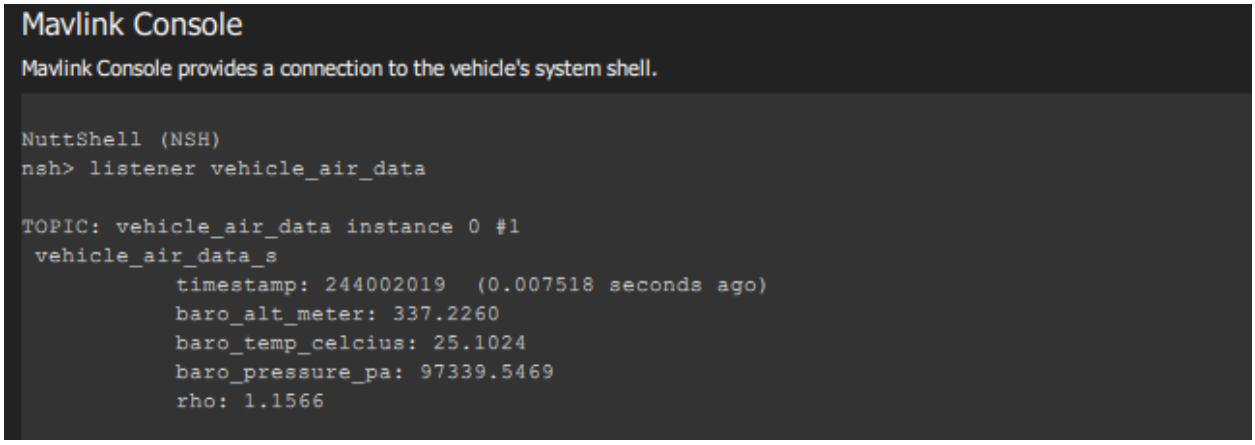


Figure 42: Inspect Sensor Message Content.

Figure 42 shows how this command would look in the console. For further information regarding uORB, please see the documentation from PX4.[16]

[16] `https://docs.px4.io/master/en/middleware/uorb.html`.

6.3 Resetting the Parameter Configuration

If the estimation of the HiL Simulation is represented wrongly, e.g. so that the Talon seems to fly backwards or the heading is inaccurate, the parameter settings in GQC may be changed. This may lead, for example, to a rotation of the measured acceleration and hence, a wrong estimation with a rotated state. You can change to the default parameters by following these steps:

1. Connect the Pixhawk to the user's computer and open QGC.
2. Go to the `Vehicle Setup` view, select the `Parameters` option, and click the `Tools` button on the right side (see Fig. 38).
3. Choose "Load from file" and navigate to the USB Stick to choose the `Param_HIL.params` file from the folder `QGroundControl`.
4. Finally, restart the Pixhawk by unplugging the power connection and reconnect.

A USB Stick Directory