**PX4<>AirSim<>Gazebo Installationsanleitung:**

Die Installationnsanleitung soll als Ergänzung zur Anleitung auf <https://microsoft.github.io/AirSim/build_linux/> und <https://microsoft.github.io/AirSim/gazebo_drone/> dienen. Mit der Installation soll es möglich sein Gazebo und PX4 auf Linux via AirSim und Unreal Environment auf einem Windows Host zu verknüpfen. Als Voraussetzung muss bereits AirSim auf Windows in ein UE Environment eingebettet sein und erfolgreich getestet sein.

* Zu erst muss AirSim in Linux installiert werden. Wenn AirSim in Linux (Ubuntu 18.04) installiert wird, werden dabei die folgenden Befehle benötigt. Zu finden sind diese unter der AirSim Github Page.

git clone https://github.com/Microsoft/AirSim.git

cd AirSim  
./setup.sh  
./build.sh

Falls der ./build.sh Befehl einen Fehler auswirft, muss versucht werden mit folgendem Befehl nötige Dependencies/Packages zu installieren. Dies sollte eigentlich bereits durch ./setup.sh geschehen, passierte da aber nicht. Der zusätzlich auszuführende Befehl ist der folgende:

sudo apt-get install libstdc++-8-dev

* Danach muss AirSim für GazeboDrone installiert werden, dazu müssen gemäss Github Page die folgenden Befehle ausgeführt werden:

sudo apt-get install libgazebo9-dev  
./**clean**.sh  
./**setup**.sh

./**build**.sh --**gcc**cd GazeboDrone  
mkdir build && cd build  
cmake -DCMAKE\_C\_COMPILER=gcc-8 -DCMAKE\_CXX\_COMPILER=g++-8 ..

make

Wenn der ./build.sh Befehl aufgrund des fehlenden Compilers abgebrochen wird mit dem nachfolgenden Befehl den nötigen Compiler installieren:

sudo apt-get install gcc-8 g++-8

* Im Anschluss PX4 auf Ubuntu klonen und die folgenden Befehle nutzen

mkdir -p PX4

cd PX4

git clone https://github.com/PX4/PX4-Autopilot.git --recursive

* PX4 mit folgendem Befehl builden und System neustarten:

bash ./PX4-Autopilot/Tools/setup/ubuntu.sh

* Falls der Fehler *“Failed building wheel for pillow”* auftritt die folgenden Befehle ausführen um die fehlenden Packages manuell zu installieren:

sudo apt-get install libjpeg-dev zlib1g-dev

pip3 install Pillow

Ebenfalls kann der nachfolgende Code versucht werden, falls es immer noch nicht funktioniert.

sudo pip install -U setuptools

* Mit dem nachfolgenden Befehl PX4 und Gazebo starten:

make px4\_sitl gazebo

Falls ein Fehler ausgegeben wird, dass der Compiler beziehungsweise der Pfad zum Compiler nicht erkannt wird, dann die folgenden Befehle ausführen:

sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-8 20

sudo update-alternatives --install /usr/bin/g++ g++ /usr/bin/g++-8 20

* Falls nach einem erfolgreichen make Befehl beim Warten auf den TCP Port nicht mit Gazebo verbunden wird und die Simulation nicht startet, diese beenden und folgenden Befehl ausführen.   
  sudo apt upgrade libignition-math2

Danach mit den folgen zwei Befehlen PX4 nochmals neu ausführen.

make clean

make px4\_sitl gazebo

* PX4 sollte sich nun mit Gazebo verbinden und sobald Gazebo geöffnet ist, kann im PX4 Terminal mit commander takeoff und commander land die Funktionalität von beiden Systemen getestet werden.
* Wenn der Test bis jetzt erfolgreich war, kann die Verbindung zur AirSim Library hergestellt werden. Dafür das Verzeichnis /AirSim/GazeboDrone/src öffnen und die Datei main.cpp mit einem Text Editor öffnen. Linie 42 mit der IP Addresse des Host Computers ergänzen:

Linie 42/ msr::airlib::MultirotorRpcLibClient client("Host IP Address");

* Um die Host IP Addresse aufzurufen eine Powershell öffnen und ipconfig ausführen. Die unten aufgeführte Ausgabe enthält die IP Addresse in gelb.

*Windows-IP-Konfiguration*

*Ethernet-Adapter Ethernet:*

*Medienstatus. . . . . . . . . . . : Medium getrennt*

*Verbindungsspezifisches DNS-Suffix: home*

*Ethernet-Adapter VirtualBox Host-Only Network #2:*

*Verbindungsspezifisches DNS-Suffix:*

*Verbindungslokale IPv6-Adresse . : fe80::549f:f8df:9ed0:a14c%19*

*IPv4-Adresse . . . . . . . . . . : 192.168.208.1*

*Subnetzmaske . . . . . . . . . . : 255.255.255.0*

*Standardgateway . . . . . . . . . :*

* AirSim muss nun erneut gebaut werden dazu die folgenden Befehle ausführen:

./**clean**.sh  
./**setup**.sh

./**build**.sh **–gcc**

cd GazeboDrone/build  
cmake -DCMAKE\_C\_COMPILER=gcc-8 -DCMAKE\_CXX\_COMPILER=g++-8 ..

make

* Um die Interne Physik von AirSim (FastPhysics) durch die von Gazebo zu ersetzen muss das settings.json File wie folgt angepasst werden:

"SettingsVersion": 1.2,

"SimMode": "Multirotor",

"ClockType": "SteppableClock",

"PhysicsEngineName": "ExternalPhysicsEngine",

"Vehicles": {

"PX4": {

"VehicleType": "PX4Multirotor",

……

Der Rest des settings.json File kann belassen werden wie es ist.

* Damit nun PX4 mit Windows Host und der darin installierten GCS kommuniziert muss Mavlink Broadcast aktiviert werden. Dazu im PX4-Autopilot Ordner die Datei px4-rc.mavlink öffnen (zu finden unter: PX4/PX4-Autopilot/ROMFS/px4fmu\_common/init.d-posix

Die Datei enthält die folgenden Befehle. Die gelb hinterlegten Befehle von -m auf -p ändern.

*#!/bin/sh*

*# shellcheck disable=SC2154*

*udp\_offboard\_port\_local=$((14580+px4\_instance))*

*udp\_offboard\_port\_remote=$((14540+px4\_instance))*

*[ $px4\_instance -gt 9 ] && udp\_offboard\_port\_remote=14549 # use the same ports for more than 10 instances to avoid port overlaps*

*udp\_onboard\_payload\_port\_local=$((14280+px4\_instance))*

*udp\_onboard\_payload\_port\_remote=$((14030+px4\_instance))*

*udp\_onboard\_gimbal\_port\_local=$((13030+px4\_instance))*

*udp\_onboard\_gimbal\_port\_remote=$((13280+px4\_instance))*

*udp\_gcs\_port\_local=$((18570+px4\_instance))*

*# GCS link*

*mavlink start -x -u $udp\_gcs\_port\_local -r 4000000 -f -p*

*mavlink stream -r 50 -s POSITION\_TARGET\_LOCAL\_NED -u $udp\_gcs\_port\_local*

*mavlink stream -r 50 -s LOCAL\_POSITION\_NED -u $udp\_gcs\_port\_local*

*mavlink stream -r 50 -s GLOBAL\_POSITION\_INT -u $udp\_gcs\_port\_local*

*mavlink stream -r 50 -s ATTITUDE -u $udp\_gcs\_port\_local*

*mavlink stream -r 50 -s ATTITUDE\_QUATERNION -u $udp\_gcs\_port\_local*

*mavlink stream -r 50 -s ATTITUDE\_TARGET -u $udp\_gcs\_port\_local*

*mavlink stream -r 50 -s SERVO\_OUTPUT\_RAW\_0 -u $udp\_gcs\_port\_local*

*mavlink stream -r 20 -s RC\_CHANNELS -u $udp\_gcs\_port\_local*

*mavlink stream -r 10 -s OPTICAL\_FLOW\_RAD -u $udp\_gcs\_port\_local*

*# API/Offboard link*

*mavlink start -x -u $udp\_offboard\_port\_local -r 4000000 -f -p onboard -o $udp\_offboard\_port\_remote*

*# Onboard link to camera*

*mavlink start -x -u $udp\_onboard\_payload\_port\_local -r 4000 -f -p onboard -o $udp\_onboard\_payload\_port\_remote*

*# Onboard link to gimbal*

*mavlink start -x -u $udp\_onboard\_gimbal\_port\_local -r 400000 -p gimbal -o $udp\_onboard\_gimbal\_port\_remote*

Mit den gemachten Änderungen wird mavlink vom PX4 Kontroller nun die entsprechenden UDP Ports gebroadcasted. Der Windows Host kann nun auf den entsprechenden UDP Ports die Signale abhören und für Befehle empfangen.

* Die Simulation kann nun gestarten werden im dem der Befehl

make px4\_sitl gazebo

ausgeführt wird und im Ordner /AirSim/GazeboDrone/build das Skript mit

./GazeboDrone

gestartet wird. Auf dem Windows Host die Unreal Simulation starten und QGroundControl öffnen. Das ./GazeboDrone Skript sollte sich nun mit AirSim auf dem Windows Host verbinden und die berechneten Positionen vom Gazebo Simulator weiterleiten. QGroundControl sollte sich nun automatisch über UDP Broadcast von PX4 verbinden. Über die AirSim Pyhton API können weiterhin Sensordaten von der Unreal Simulation abgerufen werden. Die AirSim Pyhton API funktioniert aber nicht mehr um die Drohne zu steuern. Dies kann nun durch MAVSDK geschehen. Dabei MAVSDK auf dem Windows Host über den UDP Port 14540 verbinden. Mehr Infos dazu siehe Anleitung zur Installation von MAVSDK. Im PX4 Kontroller muss keine zusätzliche MAVLink Verbindung für die MAVSDK API erstellt werden, da die AirSim Pyhton API nicht mehr auf den Flight Controller zugreift.

**WICHTIG**

* Weiter nützliche Informationen und Einstellung sind unter <https://docs.px4.io/v1.12/en/simulation/gazebo.html#headless> zu finden.
* Um Parameter standardmässig beim Start von PX4 definieren können diese im File «px4-rc.params unter *PX4/PX4-Autopilot/ROMFS/px4fmu\_common/init.d-posix* angehängt werden. Für Offboard Controll werden die folgenden Parameter empfohlen

*#!/bin/sh*

*# shellcheck disable=SC2154*

*#param set-default MAV\_SYS\_ID $((px4\_instance+1))*

*#param set-default IMU\_INTEG\_RATE 250*

*# Set Parameter for Offboard Control without Remote Controller*

*param set-default NAV\_DLL\_ACT 0*

*param set-default NAV\_RCL\_ACT 0*

*param set-default COM\_OF\_LOSS\_T 30*

*param set-default COM\_OBL\_ACT 1*

*param set-default COM\_RCL\_EXCEPT 4*

Für mehr Info und eine Erklärung zu den definierten Parameter kann hier gefunden werden <https://docs.px4.io/v1.12/en/advanced_config/parameter_reference.html>

* Um die GPS Koordinaten des Startpunktes zu setzen den folgenden Ordner öffnen *~/PX4/PX4-Autopilot/Tools/sitl\_gazebo/worlds* . Darin enthalten sind alle world Files von Gazebo. Standardmässig wird das *empty.world* File verwendet. Um darin die GPS Koordinaten festzulegen das File mit den Koordinaten gemäss gelb hinterlegtem Beispiel anpassen:

<?xml version="1.0" ?>

<sdf version="1.5">

<world name="default">

<!-- A global light source -->

<include>

<uri>model://sun</uri>

</include>

<!-- A ground plane -->

<include>

<uri>model://ground\_plane</uri>

</include>

<include>

<uri>model://asphalt\_plane</uri>

</include>

<spherical\_coordinates>

<surface\_model>EARTH\_WGS84</surface\_model>

<latitude\_deg>47.130633650952</latitude\_deg>

<longitude\_deg>7.24060672327239</longitude\_deg>

<elevation>458.0</elevation>

</spherical\_coordinates>

<physics name='default\_physics' default='0' type='ode'>

<gravity>0 0 -9.8066</gravity>

<ode>

<solver>

<type>quick</type>

<iters>10</iters>

<sor>1.3</sor>

<use\_dynamic\_moi\_rescaling>0</use\_dynamic\_moi\_rescaling>

</solver>

<constraints>

<cfm>0</cfm>

<erp>0.2</erp>

<contact\_max\_correcting\_vel>100</contact\_max\_correcting\_vel>

<contact\_surface\_layer>0.001</contact\_surface\_layer>

</constraints>

</ode>

<max\_step\_size>0.004</max\_step\_size>

<real\_time\_factor>1</real\_time\_factor>

<real\_time\_update\_rate>250</real\_time\_update\_rate>

<magnetic\_field>6.0e-6 2.3e-5 -4.2e-5</magnetic\_field>

</physics>

</world>

</sdf>

Anschliessenden sollte der neu gewählte Startpunkt via Ground Controll Station sichtbar sein.