

## GUIDA ALL'INSTALLAZIONE

E' consigliato seguire la procedura di installazione dei vari programmi seguendo l'ordine in cui sono presentate le loro guide all'installazione.

Tutti I test sono stati fatti su "Ubuntu 18.04 Bionic Beaver" come sistema operativo, quindi si raccomanda di usare questa versione.

### INSTALLARE CUDA, CUDNN E BUILDARE OPENCV CON IL LORO SUPPORTO:

Io ho usato le seguenti versioni:

CUDA: 11.3

CUDNN: 8.2.1

OPENCV: 4.5.2

Seguire la guida a questo link:

<https://www.pyimagesearch.com/2020/02/03/how-to-use-opencvs-dnn-module-with-nvidia-gpus-cuda-and-cudnn/>

ATTENZIONE: nello "Step #3: Download OpenCV source code" eseguire i seguenti comandi anzichè quelli riportati sulla guida:

```
cd ~
```

```
wget -O opencv.zip https://github.com/opencv/opencv/archive/4.5.2.zip
```

```
wget -O opencv_contrib.zip https://github.com/opencv/opencv_contrib/archive/4.5.2.zip
```

```
unzip opencv.zip
```

```
unzip opencv_contrib.zip
```

```
mv opencv-4.5.2 opencv
```

```
mv opencv_contrib-4.5.2 opencv_contrib
```

Una volta eseguite queste operazioni, è possibile riprendere a seguire la guida normalmente.

### INSTALLARE UNREAL ENGINE E AIRSIM:

Io ho usato le seguenti versioni:

UNREAL ENGINE: 4.25

AIRSIM: 1.4.0

[https://microsoft.github.io/AirSim/build\\_linux/](https://microsoft.github.io/AirSim/build_linux/)

una volta seguita la guida per Linux e terminata l'installazione:

Home → AirSim → Unreal → Environments

incollare su questo percorso il progetto contenuto nella cartella "AirSim" fornita da me, contiene già tutti gli asset, i codici e i modelli richiesti da Unreal Engine.

## PER APRIRE IL PROGETTO:

- 1) Aprire il terminale;
- 2) Digitare:

```
cd UnrealEngine  
./Engine/Binaries/Linux/UE4Editor
```

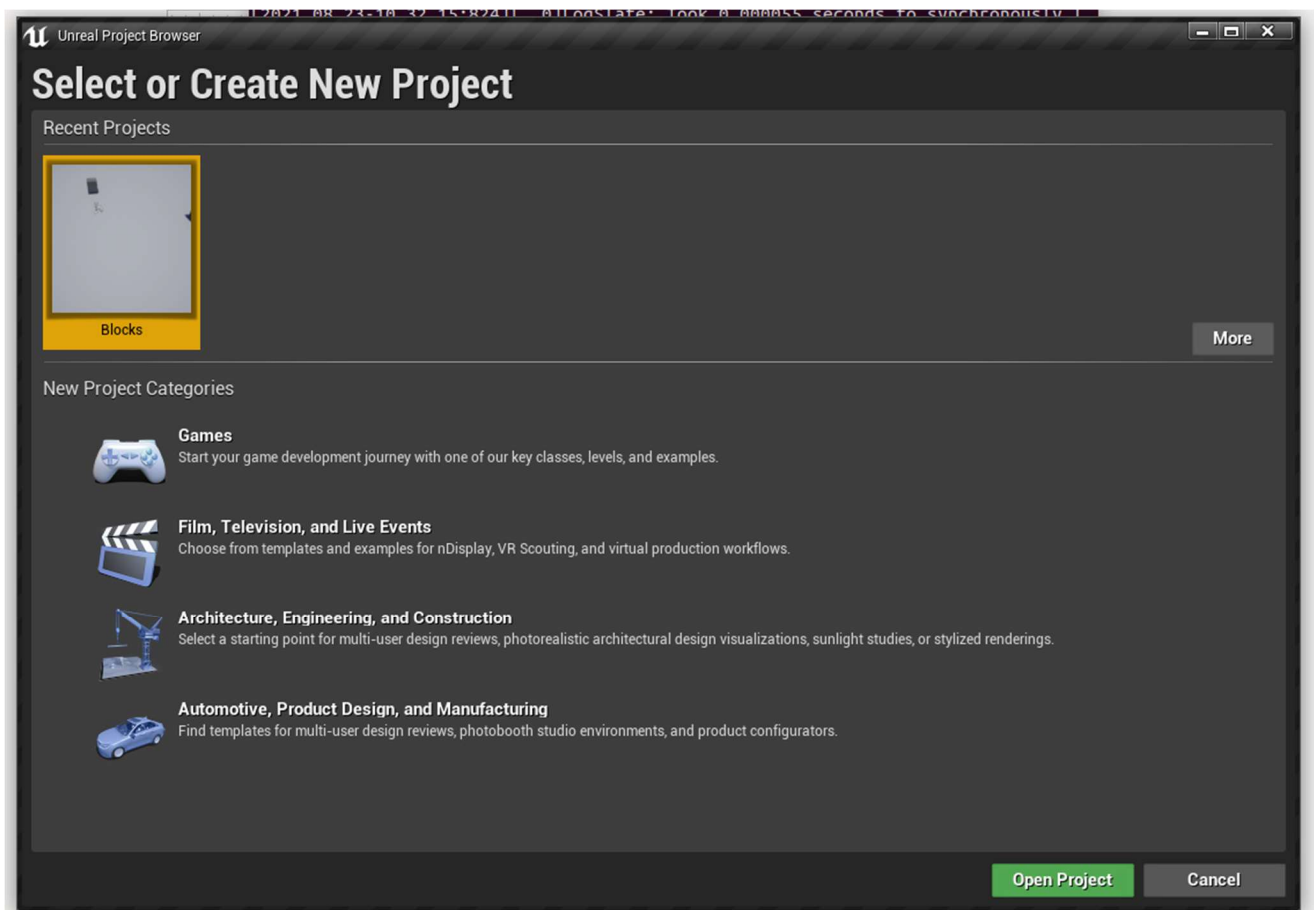
3) Si aprirà una finestra come in figura. Se è il primo avvio, cercare il tasto per aprire un progetto esistente.

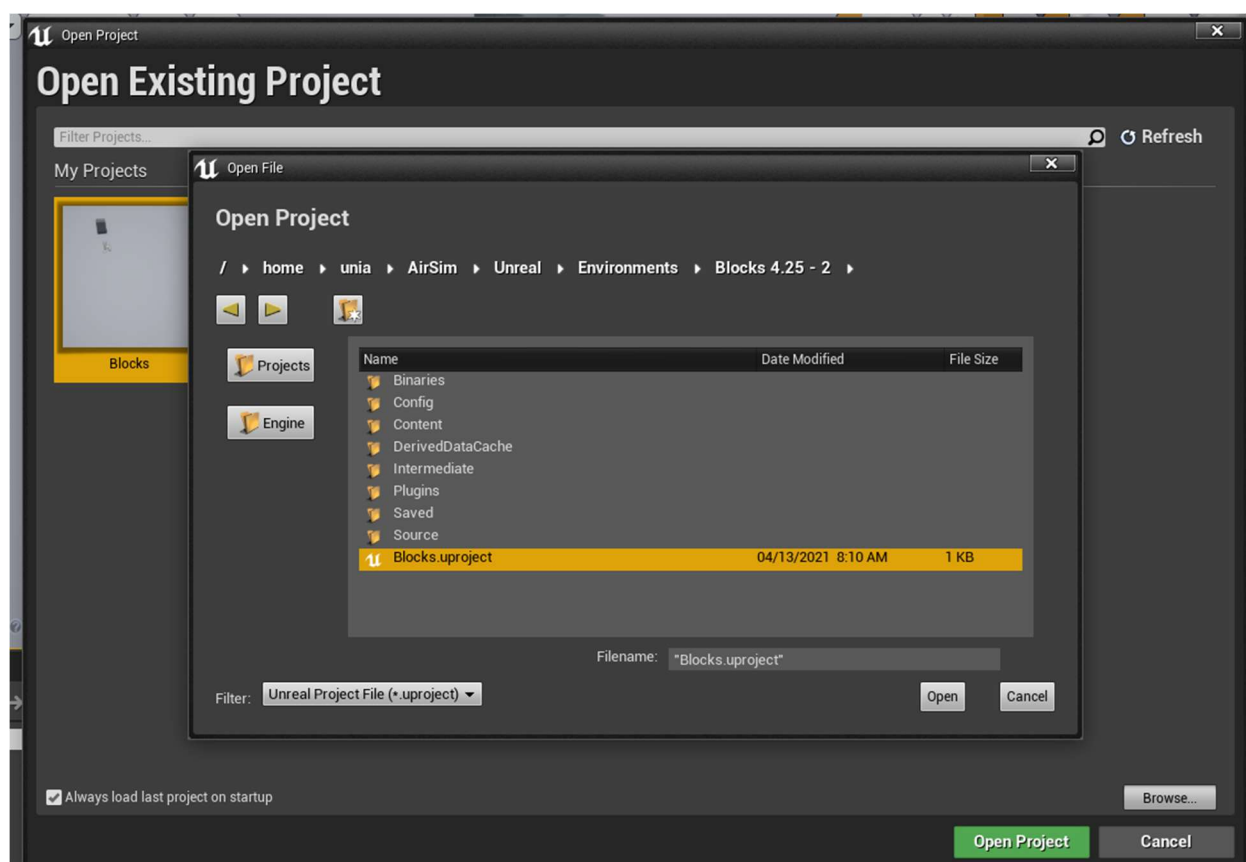
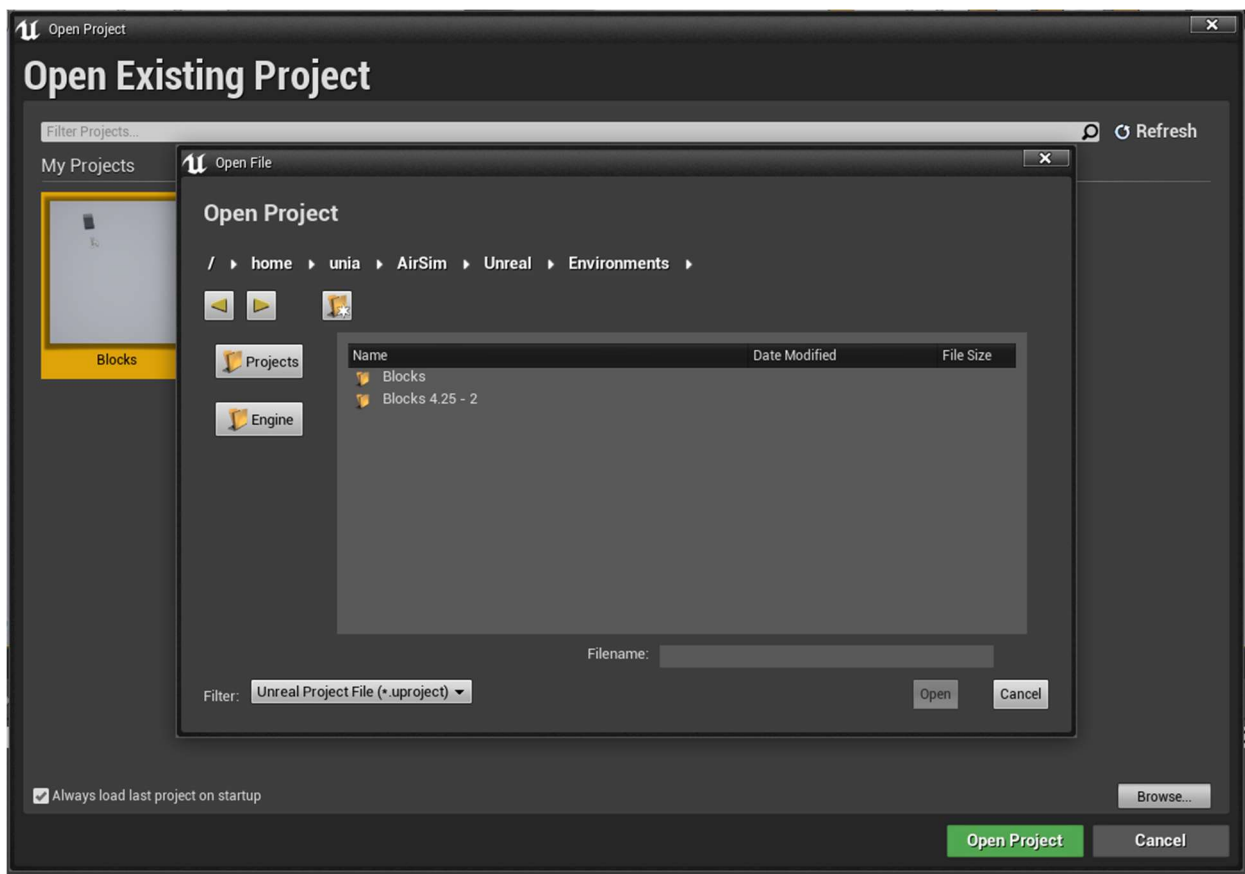
4) Premere “Browse”, seguire il percorso:

AirSim → Unreal → Environments → Blocks 4.25 - 2

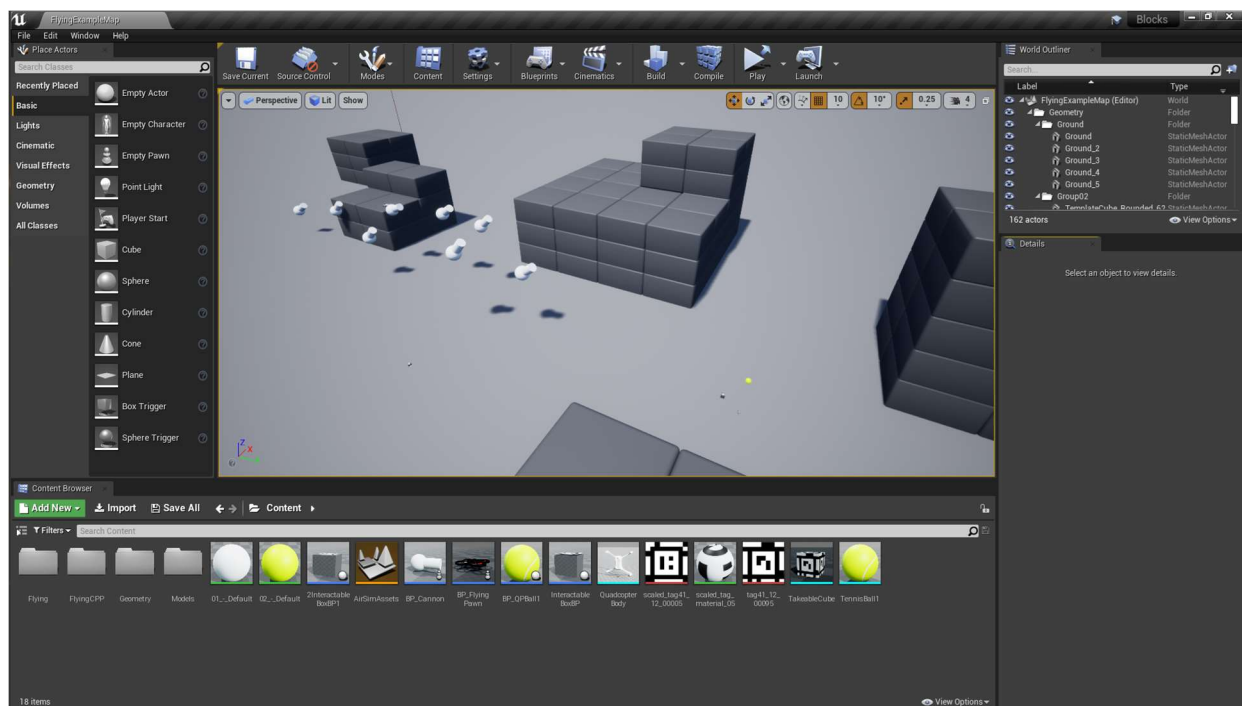
selezionare il file chiamato “Blocks.uproject”

5) Premere “Open”





6) Se tutto è stato eseguito in maniera corretta, si aprirà questa finestra:



7) NON AVVIARLO DIRETTAMENTE!

Andare su

Home → Documents → AirSim

copiare e incollare qui il file “settings.json” che ho creato io (è nella cartella AirSim del progetto che vi ho inviato io)

8) Prima di avviarlo, assicurarsi di premere la freccetta in basso accanto al bottone “Play” e selezionare “Standalone Game”.

9) Attendere finché il riquadro della simulazione non viene aperto.

10) Chiudere la simulazione e passare alle altre procedure di installazione.

## INSTALLARE PX4 E QGROUN CONTROL:

Io ho usato le seguenti versioni:

PX4 AUTOPILOT: 1.12.0 BETA

QGROUN CONTROL: 4.1.2

1) Seguire questo video per PX4:

<https://www.youtube.com/watch?v=OtValQdAdrU&t=75s>

si consiglia di provare ad eseguire l'esempio riportato nel video (quello che avvia la simulazione con JMAVSim) per essere sicuri che tutto funzioni correttamente.

2) Seguire questa guida per QGroundControl:

[https://docs.qgroundcontrol.com/master/en/getting\\_started/download\\_and\\_install.html](https://docs.qgroundcontrol.com/master/en/getting_started/download_and_install.html)

## **INSTALLARE ROS MELODIC:**

Seguire questa guida ufficiale, ma **IMPORTANTE**: nel seguirla, **SALTARE IL PUNTO 1.5 “ENVIRONMENT SETUP”**, il source deve essere fatto a mano ogni volta per evitare problemi.

Link della guida:

<http://wiki.ros.org/melodic/Installation/Ubuntu>

## **INSTALLARE ROS2 FOXY**

Scaricarlo ed installarlo seguendo questa guida. Il sistema raccomandato è Ubuntu 20, ma su Ubuntu 18 funziona lo stesso.

**ATTENZIONE**: arrivati al punto “BUILD THE CODE IN THE WORKSPACE”, anzichè seguire la guida lanciare I seguenti comandi:

```
cd ~/ros2_foxy/  
colcon build --symlink-install --packages-skip ros1_bridge
```

Link della guida:

<https://docs.ros.org/en/foxy/Installation/Ubuntu-Development-Setup.html>

Si consiglia di eseguire l’esempio riportato alla fine della guida nella sezione “Try some examples” per assicurarsi che tutto funzioni.

## **BUILDARE IL ROS1 – ROS2 BRIDGE**

Seguire questa guida:

[https://github.com/ros2/ros1\\_bridge](https://github.com/ros2/ros1_bridge)

Si consiglia di eseguire l’esempio “Example 1: run the bridge and the example talker and listener” per assicurarsi che tutto funzioni.

## **BUILDARE IL ROS WRAPPER PER AIRSIM**

Lanciare I seguenti comandi nel terminale:

1) installare gcc e verificarne la versione:

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

2. installare tf2 sensor e mavros (eseguire nel caso non fossero già stati installati insieme a ROS Melodic), N.B.: E’ UN COMANDO SOLO, QUINDI COPIARE E INCOLLARE TUTTO INSIEME ED ESEGUIRE NEL TERMINALE:

```
sudo apt-get install ros-melodic-tf2-sensor-msgs ros-melodic-tf2-geometry-msgs ros-melodic-mavros*
```

### 3. Installare catkin:

```
sudo apt-get install python-catkin-tools
```

### 4. Buildare il pacchetto:

```
cd ros;  
./opt/ros/melodic/setup.bash  
catkin build;
```

IN CASO DI ERRORI in fase di build specificare la versione di gcc:

```
catkin build -DCMAKE_C_COMPILER=gcc-8 -DCMAKE_CXX_COMPILER=g++-8
```

IMPORTANTE: questa installazione è delicata e non è detto funzionerà al primo colpo, quindi conviene fare dei test. Avviare la simulazione su Unreal Engine e scrivere su un nuovo terminale:

```
source devel/setup.bash;  
roslaunch airsimsim_ros_pkgs airsimsim_node.launch;
```

aprire un terzo terminale e scrivere:

```
source devel/setup.bash;  
roslaunch airsimsim_ros_pkgs rviz.launch;
```

se tutto viene eseguito senza intoppi si aprirà la finestra di rviz e si vedrà il programma funzionare sul terminale, se invece il comando “roslaunch airsimsim\_ros\_pkgs airsimsim\_node.launch;” crasha (si vedranno delle scritte rosse sul terminale tipo messaggi di errore), allora:

1. chiudere la finestra di rviz;
2. sul terminale dove è crashato il wrapper, lanciare il comando:

```
catkin clean
```

3. riprovare a buildare
4. riprovare a rieseguire.

Di solito dopo qualche tentativo si riesce a far funzionare, qui bisogna ripetere la procedura finché non funziona, personalmente non ho trovato altro modo per farlo funzionare se non sperare di avere un pò di fortuna :D

## **BUILDARE IL PX4-ROS2 BRIDGE**

### 1. Installare “Fast DDS”

Seguire questa guida per Ubuntu 18.04:

[https://docs.px4.io/master/en/dev\\_setup/fast-dds-installation.html](https://docs.px4.io/master/en/dev_setup/fast-dds-installation.html)

Al termine della guida, passare al punto 2.

2. Installare (se non sono già stati installati insieme a ROS2), i seguenti tools aprendo il terminale ed eseguendo i seguenti comandi:

```
sudo apt install python3-colcon-common-extensions
```

```
sudo apt install ros-foxy-eigen3-cmake-module
```

```
sudo pip3 install -U empy pyros-genmsg setuptools
```

3. Buildare il Workspace per il bridge. Aprire il terminale e lanciare il seguente comando per creare due nuove cartelle, annidate:

```
mkdir -p ~/px4_ros_com_ros2/src
```

A questo punto, avremo bisogno di scaricare due pacchetti.

Ho riscontrato dei problemi con le build più recenti (non sono riuscita a replicare il processo di installazione), quindi consiglio di utilizzare delle build RISALENTI MASSIMO AD APRILE O MAGGIO perchè sono sicuramente testate e funzionanti.

Per fare ciò,

Andare al link:

[https://github.com/PX4/px4\\_ros\\_com/commits/master](https://github.com/PX4/px4_ros_com/commits/master)

scorrere la pagina e scaricare la versione con codice:

40d8b59844f411e321e32236a2b3e7f9bb4609c2

risalente al 31 maggio.

Copiare la cartella appena scaricata nel percorso:

home/px4\_ros\_com\_ros2/src

una volta terminata la copia, andare al link:

[https://github.com/PX4/px4\\_msgs/commits/master](https://github.com/PX4/px4_msgs/commits/master)

scorrere la pagina scaricare la versione con codice:

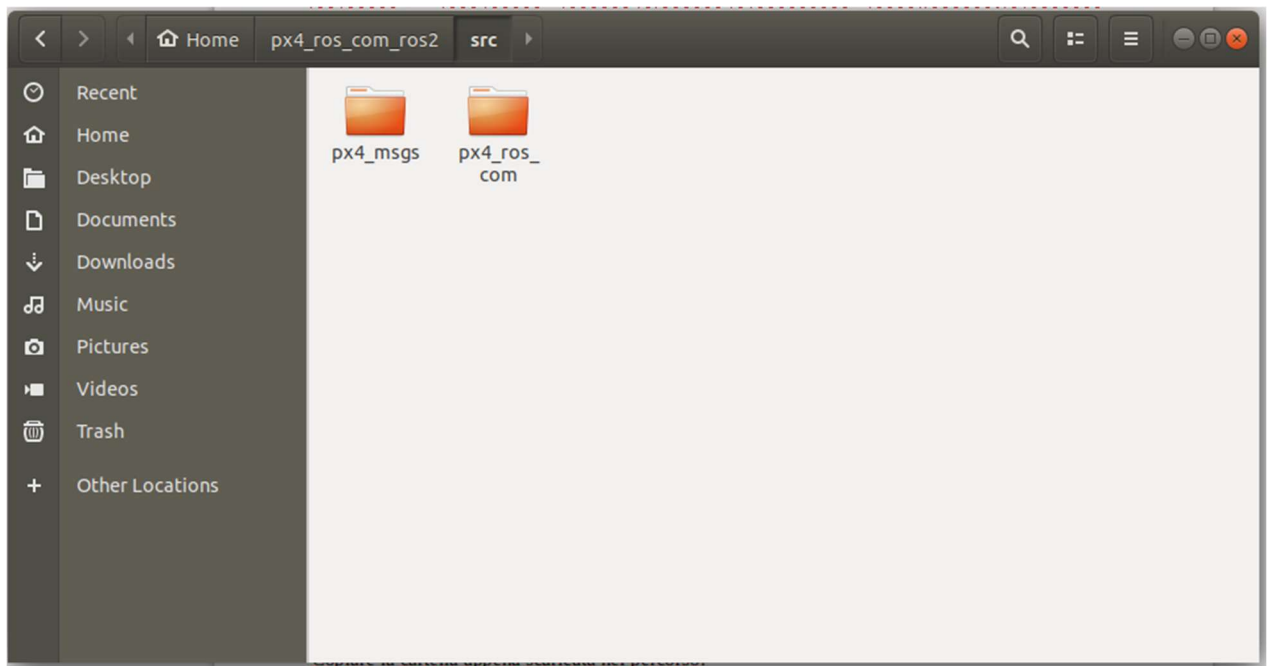
fe3e9ee12cfbd4ec4cb6d0b64e39792d35d2b8e8

risalente anch'essa al 31 maggio.

Copiare la cartella appena scaricata nel percorso:

home/px4\_ros\_com\_ros2/src

alla fine di questa procedura, nella cartella home/px4\_ros\_com\_ros2/src ci dovranno essere altre due cartelle, px4\_msgs e px4\_ros\_com, come indicato in figura:



Andare su:

px4\_ros\_com → scripts

Aprire il seguente file (va bene sia visual studio code che qualsiasi altro editor di testo):

generate\_microRTPS\_bridge.py

Andare alla riga di codice n. 275 e modificarla da: `if not ros2_distro` a `if ros2_distro`

Cancellare o commentare le righe dalla 279 alla 282 (l'istruzione else e il suo contenuto), come indicato in figura.

Il file originale era così:

```
272
273 # get FastRTPS version
274 fastrtps_version = ''
275 if not ros2_distro:
276     # grab the version installed system wise
277     fastrtps_version = subprocess.check_output(
278         "ldconfig -v 2>/dev/null | grep libfastrtps", shell=True).decode("utf-8").strip().split('so.')[1]
279 else:
280     # grab the version of the ros-<ros_distro>-fastrtps package
281     fastrtps_version = re.search(r'Version:\s*([\dd.]+)', subprocess.check_output(
282         "dpkg -s ros-" + ros2_distro + "-fastrtps 2>/dev/null | grep -i version", shell=True).decode("utf-8").strip()).group(1)
283
```

dopo le modifiche, deve essere così:

```
272
273 # get FastRTPS version
274 fastrtps_version = ''
275 if ros2_distro:
276     # grab the version installed system wise
277     fastrtps_version = subprocess.check_output(
278         "ldconfig -v 2>/dev/null | grep libfastrtps", shell=True).decode("utf-8").strip().split('so.')[1]
279
```



salvare le modifiche, aprire il terminale e lanciare I seguenti comandi:

```
cd px4_ros_com_ros2
```

```
./~/ros2_foxy/install/local_setup.bash
```

```
colcon build
```

Se tutto viene buildato in maniera corretta, andare sul percorso:

```
px4_ros_com_ros2 → src → px4_ros_com → src
```

e qui incollare la cartella “major\_tom\_offboard” inclusa nella cartella “Autopilot” del mio progetto. Aprire il terminale, lanciare I comandi per buildare nuovamente:

```
cd px4_ros_com_ros2
```

```
source ~/px4_ros_com_ros2/install/setup.bash
```

```
colcon build --packages-select px4_ros_com
```

MODIFICARE I PARAMETRI DEL DRONE SU QGROUND CONTROL (solo per il primo avvio):

Cliccare due volte sull'icona di Qground Control per aprirlo. Apparirà una GUI.

Aprire due terminali.

Sul terminale 1, digitare:

```
cd Firmware
```

```
make px4_sitl_rtps none_iris
```

Attendere, quando comparirà il messaggio:

```
INFO [simulator] Waiting for simulator to accept connection on TCP port 4560
```

Avviare la simulazione su AirSim come descritto precedentemente, premendo Play come “Standalone Game”.

Sul terminale 2, digitare:

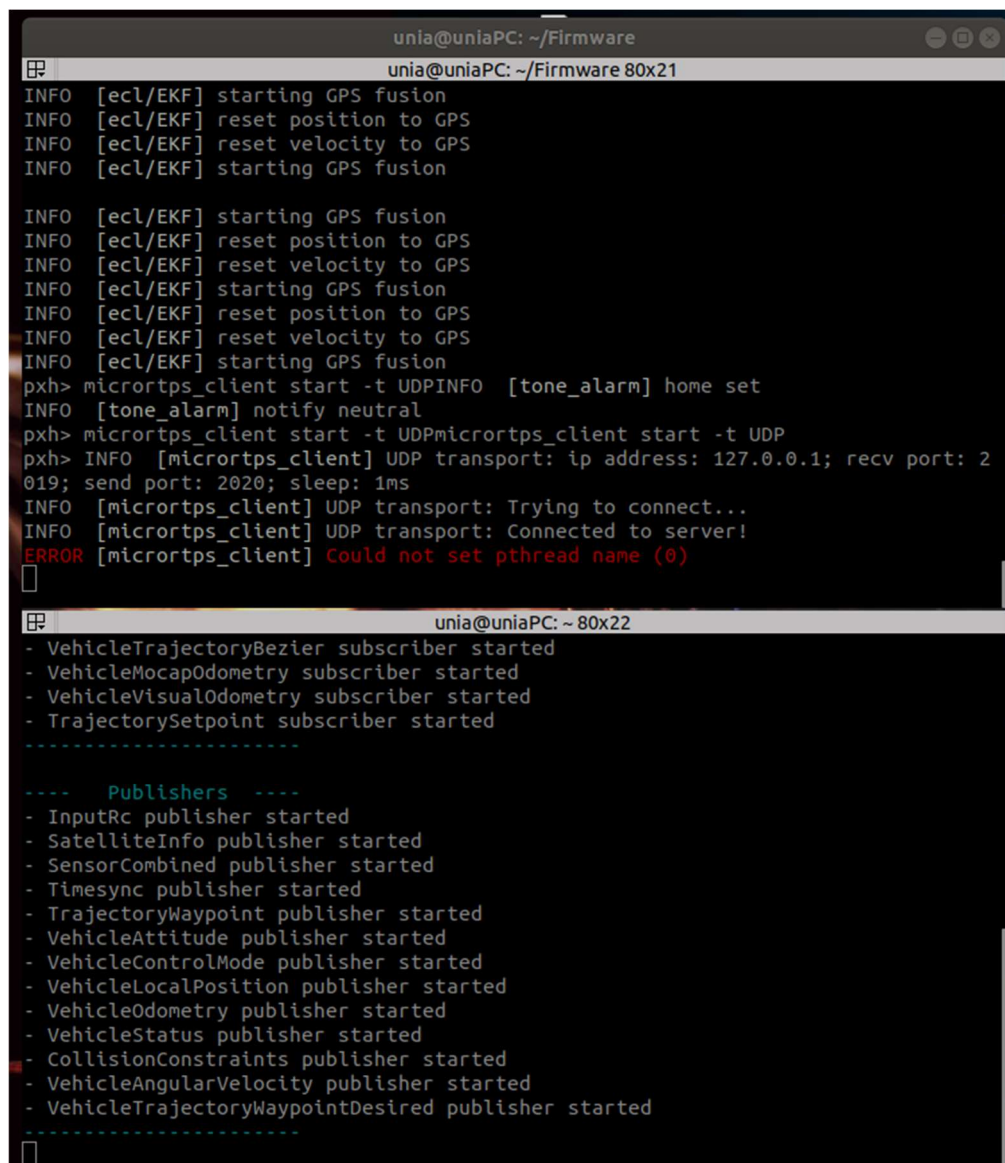
```
source ~/px4_ros_com_ros2/install/setup.bash
```

```
micrortps_agent -t UDP
```

ritornare al terminale 1 e digitare:

```
micrortps_client start -t UDP
```

premere invio ed aspettare che I terminali appaiano come in figura:

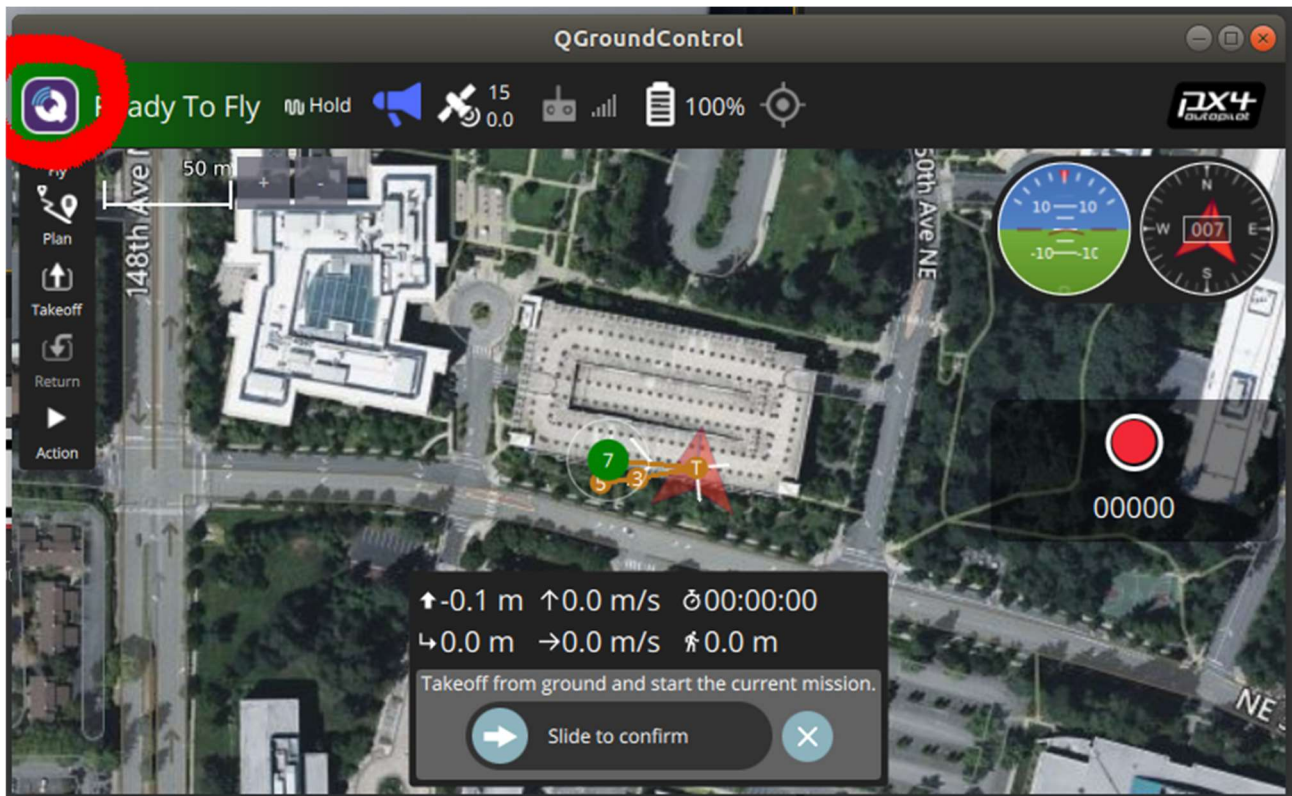


```
unia@uniaPC: ~/Firmware
unia@uniaPC: ~/Firmware 80x21
INFO [ec1/EKF] starting GPS fusion
INFO [ec1/EKF] reset position to GPS
INFO [ec1/EKF] reset velocity to GPS
INFO [ec1/EKF] starting GPS fusion

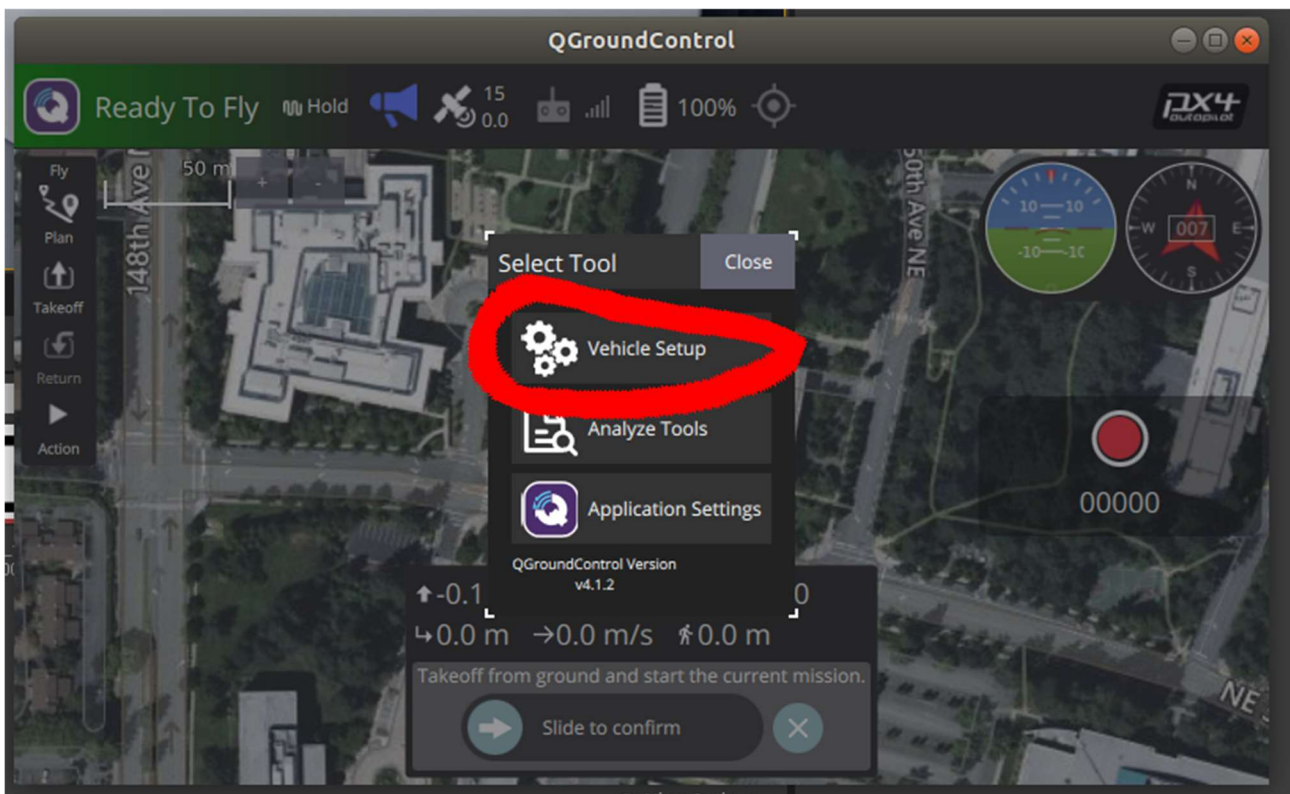
INFO [ec1/EKF] starting GPS fusion
INFO [ec1/EKF] reset position to GPS
INFO [ec1/EKF] reset velocity to GPS
INFO [ec1/EKF] starting GPS fusion
INFO [ec1/EKF] reset position to GPS
INFO [ec1/EKF] reset velocity to GPS
INFO [ec1/EKF] starting GPS fusion
pxh> micrortps_client start -t UDPINFO [tone_alarm] home set
INFO [tone_alarm] notify neutral
pxh> micrortps_client start -t UDPmicrortps_client start -t UDP
pxh> INFO [micrortps_client] UDP transport: ip address: 127.0.0.1; recv port: 2
019; send port: 2020; sleep: 1ms
INFO [micrortps_client] UDP transport: Trying to connect...
INFO [micrortps_client] UDP transport: Connected to server!
ERROR [micrortps_client] Could not set pthread name (0)

unia@uniaPC: ~/80x22
- VehicleTrajectoryBezier subscriber started
- VehicleMocapOdometry subscriber started
- VehicleVisualOdometry subscriber started
- TrajectorySetpoint subscriber started
-----
---- Publishers ----
- InputRc publisher started
- SatelliteInfo publisher started
- SensorCombined publisher started
- Timesync publisher started
- TrajectoryWaypoint publisher started
- VehicleAttitude publisher started
- VehicleControlMode publisher started
- VehicleLocalPosition publisher started
- VehicleOdometry publisher started
- VehicleStatus publisher started
- CollisionConstraints publisher started
- VehicleAngularVelocity publisher started
- VehicleTrajectoryWaypointDesired publisher started
-----
```

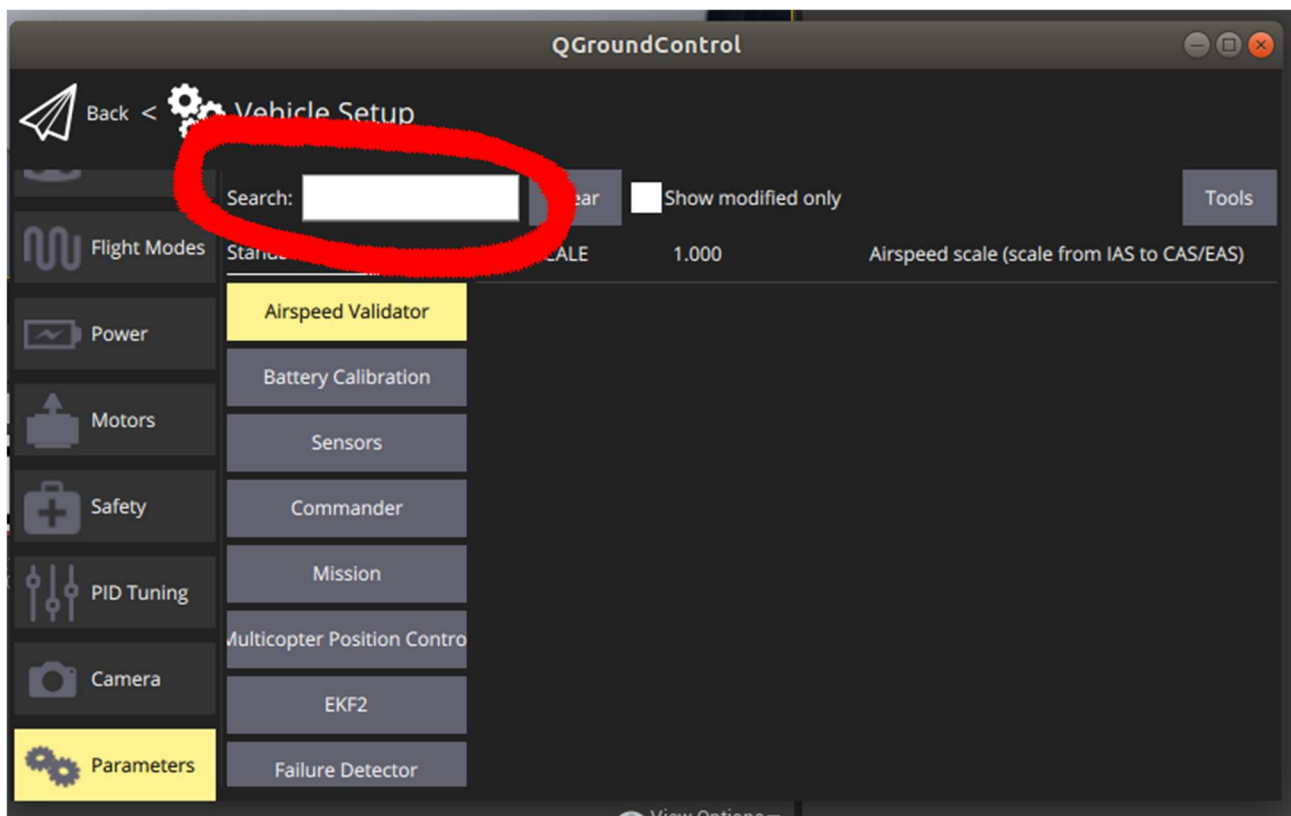
A questo punto, spostarsi sulla GUI di Qground Control che avrà un aspetto come quello della figura seguente. Premere il tasto cerchiato in rosso (cioè la Q grande viola):



Si aprirà questa finestra, premere la voce “Vehicle Setup” (cerchiato in rosso):



Scorrere fino alla voce “Parameters”, come indicato in figura.



Nella barra di ricerca In alto (cerchiata in rosso), cercare I seguenti parametri e modificarli come segue (per effettuare la modifica basta cliccare una volta sul parametro, si apre una finestrella a destra In cui è possibile digitare il valore nuovo, come in figura). Una volta modificato il parametro di interesse, premere sul bottone “Save” e passare al comando successivo sempre cercando nell’apposita barra.

#### LISTA DEI COMANDI DA MODIFICARE:

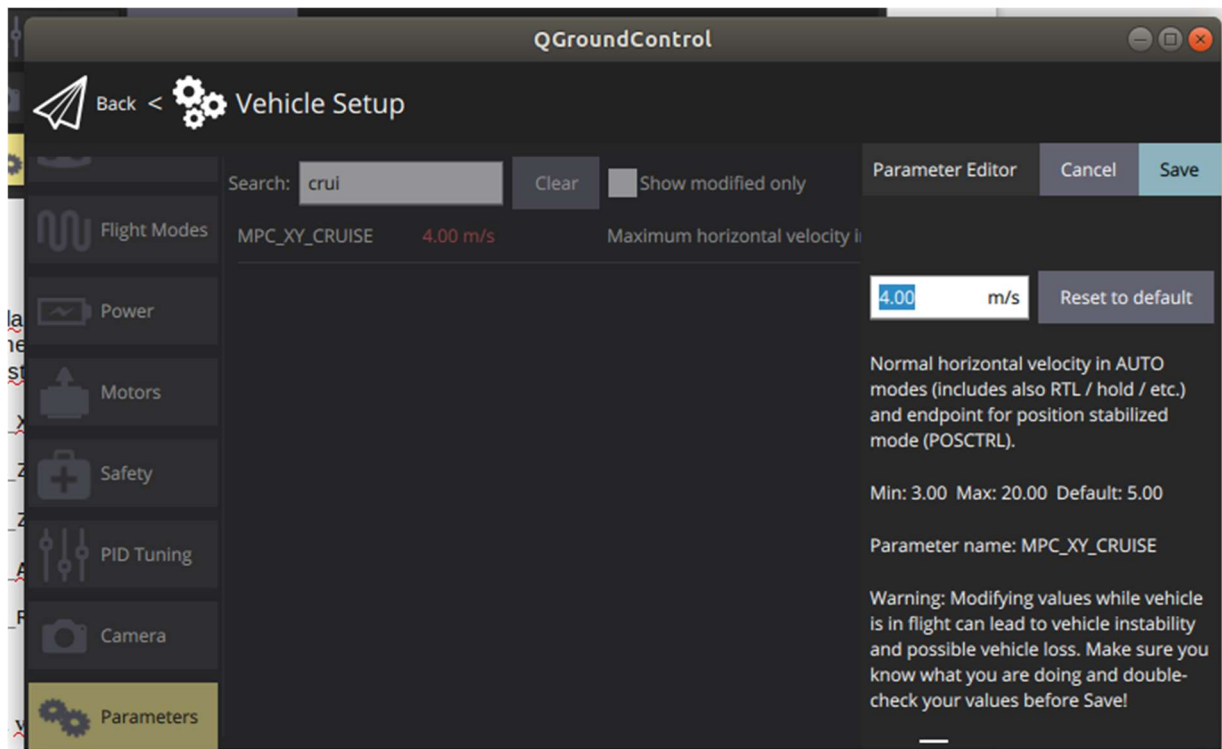
MPC\_XY\_VEL\_MAX: 4.00

MPC\_Z\_VEL\_MAX\_DN: 4.00

MPC\_Z\_VEL\_MAX\_UP: 8.00

MPC\_ACC\_HOR: 2.00

RTL\_RETURN\_ALT: 30.0



Una volta effettuate le modifiche, chiudere tutto seguendo l'ordine:

- 1) chiudere il terminale 2;
- 2) chiudere la simulazione di UnrealEngine;
- 3) chiudere il terminale 1.

## SCARICARE E BUILDARE IL PACCHETTO APRILTAG

aprire il terminale ed eseguire I seguenti comandi:

```
git clone https://github.com/AprilRobotics/apriltag.git
```

```
cmake .
```

```
sudo make install
```

## BUILDARE IL WORKSPACE PER IL SISTEMA DI VISIONE

Aprire il terminale e lanciare I seguenti comandi:

```
mkdir -p ~/dev_ws/src
```

```
cd ~/dev_ws/src
```

```
./~/ros2_foxy/install/local_setup.bash
```

all'interno della cartella src buildare il cv\_bridge e il package del sistema di visione seguendo le guide:

### **cv\_bridge:**

Leggere la documentazione al seguente link per essere sicuri di avere tutte le dipendenze già installate. Al link è presente anche la guida di installazione, che ho comunque riportato sotto.

[https://github.com/ros-perception/vision\\_opencv/tree/ros2/cv\\_bridge](https://github.com/ros-perception/vision_opencv/tree/ros2/cv_bridge)

Aprire il terminale all'interno della cartella /dev\_ws/src ed eseguire i seguenti comandi:

```
sudo apt install python3-numpy
```

```
git clone https://github.com/ros-perception/vision_opencv.git
```

```
cd vision_opencv
```

```
git checkout ros2
```

```
colcon build --symlink-install
```

### **Sistema di visione:**

Incollare nel percorso /dev\_ws/src la cartella "major\_tom\_cpp" inclusa nella cartella "Vision System" del mio progetto.

aprire il terminale e lanciare il comando:

```
colcon build --packages-select major_tom_cpp
```

A questo punto, si può eseguire la simulazione.

## COME ESEGUIRE LA SIMULAZIONE

Aprire QgroundControl facendo il doppio click sulla sua icona, dopodichè aprire 8 terminali e scrivere I comandi seguendo l'ordine riportato sotto:

### TERMINAL 1: Unreal Engine and AirSim:

```
cd UnrealEngine  
./Engine/Binaries/Linux/UE4Editor
```

### TERMINAL 2: PX4:

```
cd Firmware  
make px4_sitl_rtps none_iris
```

Quando PX4 è pronto, apparirà sul terminale il messaggio:

```
INFO [simulator] Waiting for simulator to accept connection on TCP port 4560
```

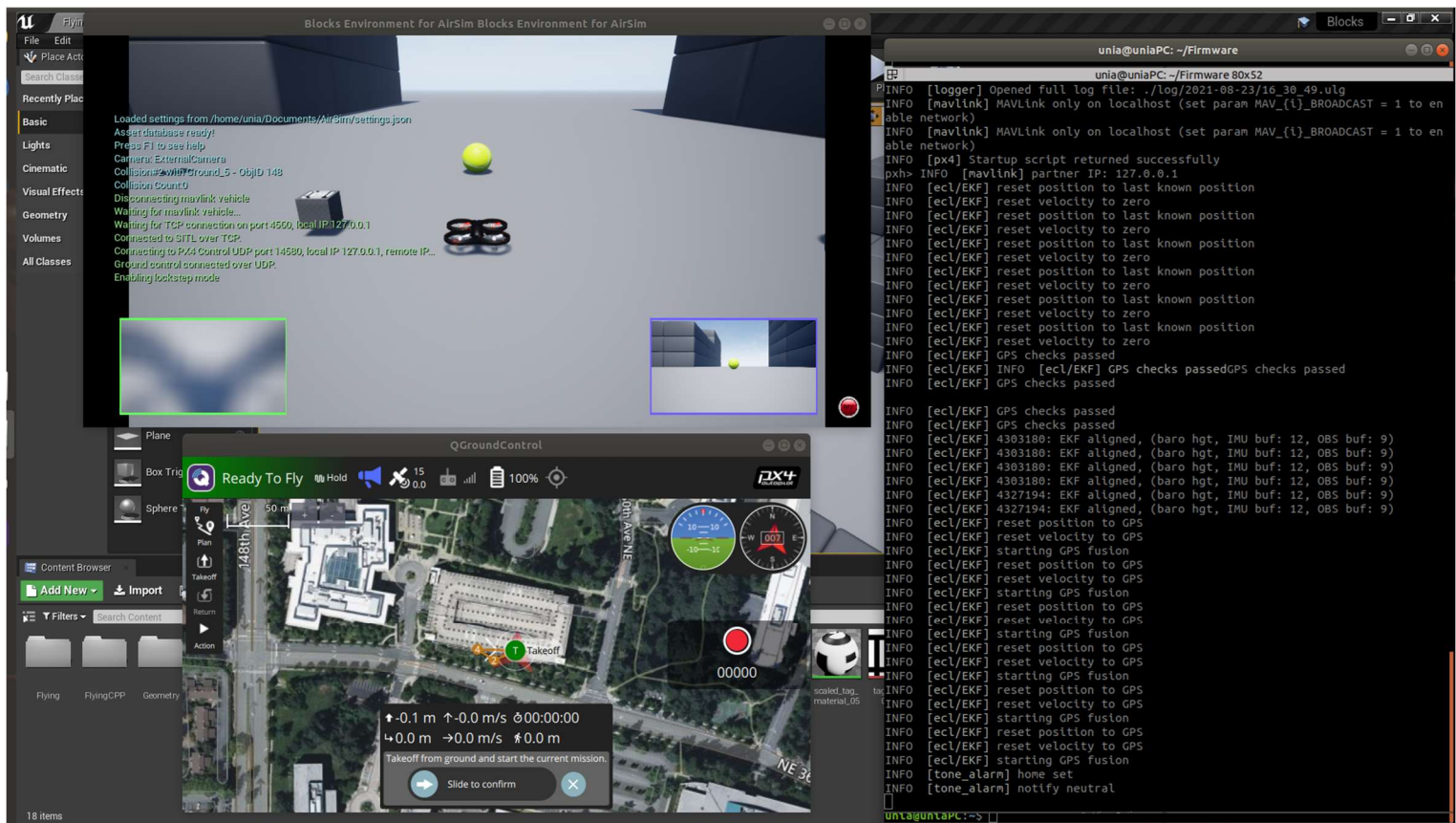
A questo punto, avviare la simulazione su AirSim come descritto precedentemente, premendo Play come "Standalone Game".

Aspettare che la simulazione si avvii come In figura (l'ultima riga del terminale deve dire "notify neutral"):

### POSSIBILI ERRORI DELL'AUTOPILOT:

- 1) se sul terminale appare un messaggio scritto in rosso, "poll timeout ..." aspettare un pò, ad un certo punto sparirà da solo e la simulazione si avvierà normalmente come indicato in figura.
- 2) durante la simulazione potrebbe accadere di veder apparire un messaggio scritto in giallo riportante la scritta "EKF timeout .." in questo caso chiudere tutto seguendo l'ordine riportato alla fine di questa guida e rieseguire. Questo errore indica che per qualche motivo l'Autopilot non sta stimando correttamente lo stato del drone, quindi potrebbe avere comportamenti strani mentre vola che non dipendono dal mio script o dalla configurazione.





Dopo aver avviato la simulazione come nella figura sovrastante, eseguire i comandi sui terminali numero 3-4-5 e 6 e 2.

## How to run the ROS1-ROS2 bridge

### TERMINAL 3:

```
. /opt/ros/melodic/setup.bash
roscore
```

### TERMINAL 4:

```
. /opt/ros/melodic/setup.bash
. ~/ros2_foxy/install/local_setup.bash
export ROS_MASTER_URI=http://localhost:11311
ros2 run ros1_bridge dynamic_bridge
```

### TERMINAL 5:

```
cd AirSim/ros
source devel/setup.bash

roslaunch airmim_ros_pkgs airmim_node.launch;
```



## How to run the PX4 bridge with ROS2

### TERMINAL 6:

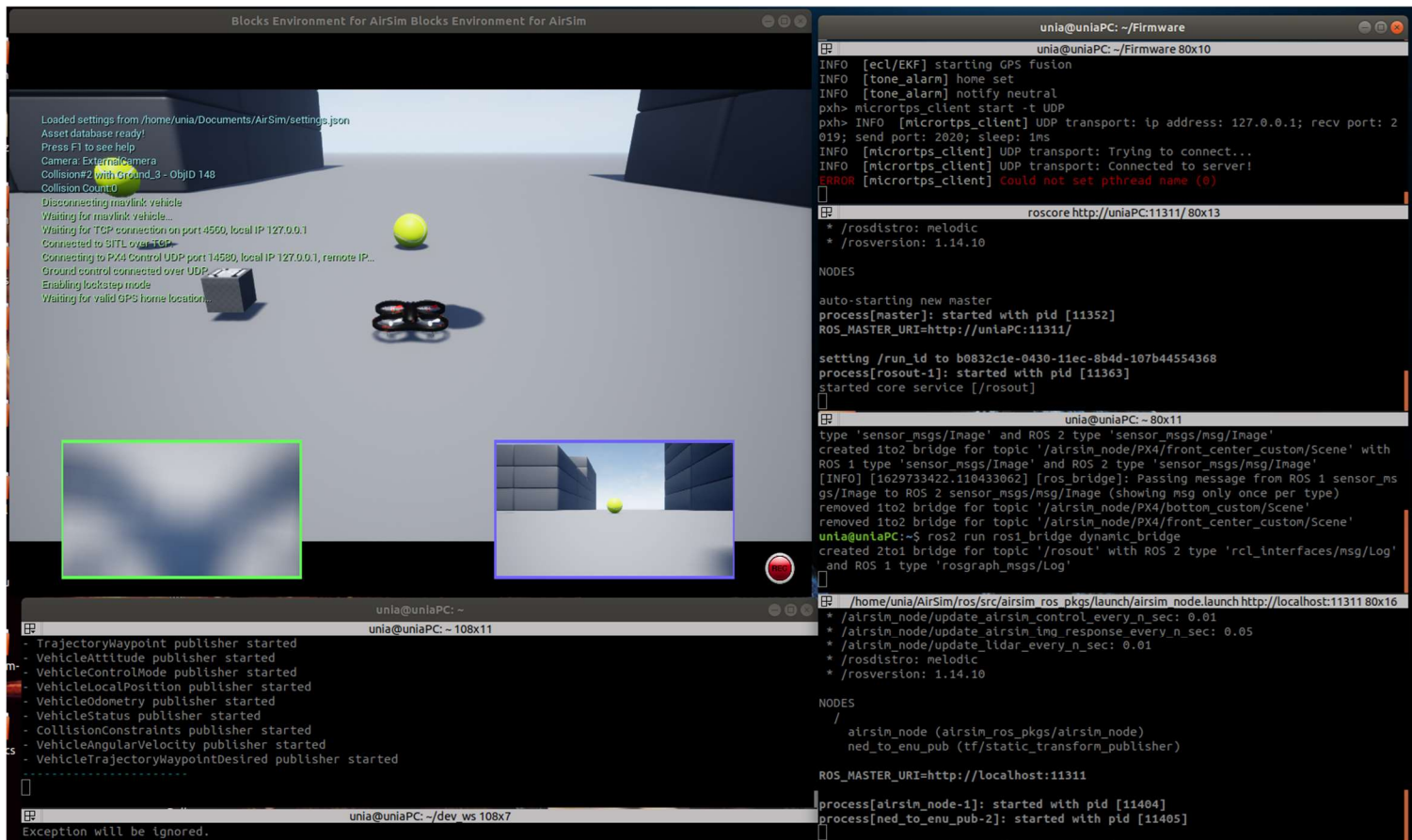
```
source ~/px4_ros_com_ros2/install/setup.bash
```

```
micrortps_agent -t UDP
```

**TERMINAL 2 (DOVE AVEVAMO GIA' AVVIATO PX4), scrivere il seguente comando e premere invio:**

```
micrortps_client start -t UDP
```

I terminali avranno l'aspetto indicato nella figura seguente:



A questo punto, avviare quanto indicato sui terminali numero 7 ed 8.

## Vision system

### TERMINAL 7:

```
cd ~/dev_ws
```

```
. install/setup.bash
```

```
ros2 run major_tom_cpp airsims_img_listener
```

### TERMINAL 8: offboard control script

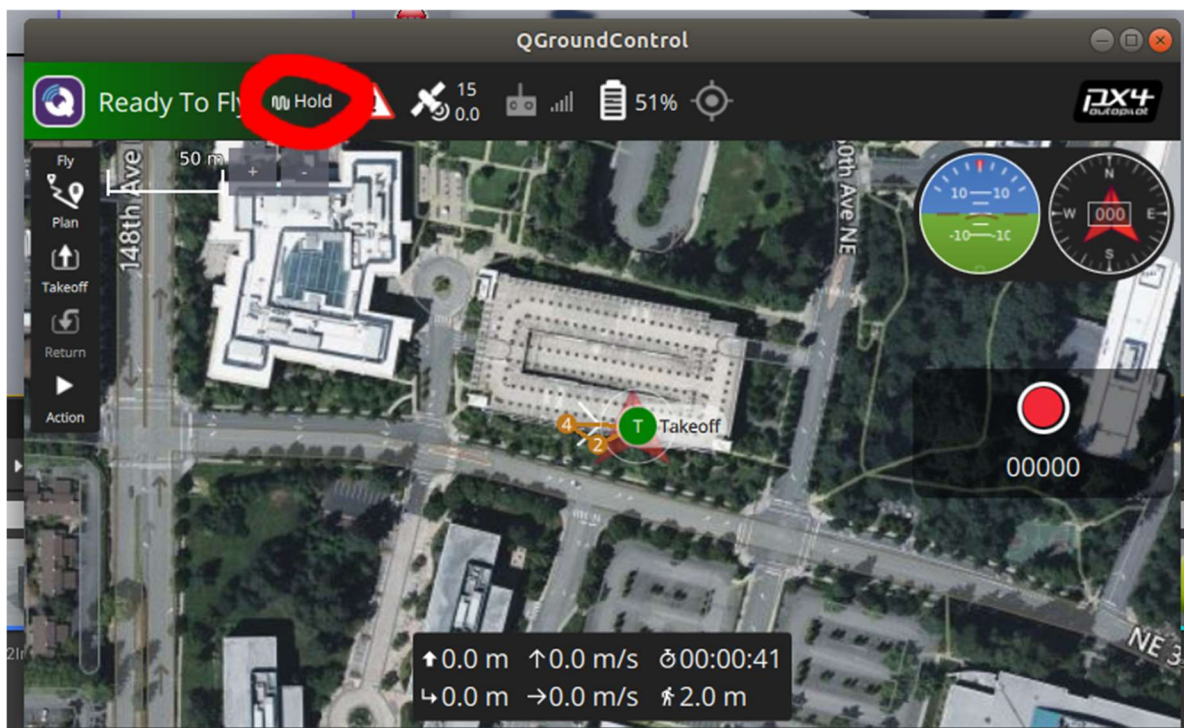
```
cd px4_ros_com_ros2
```

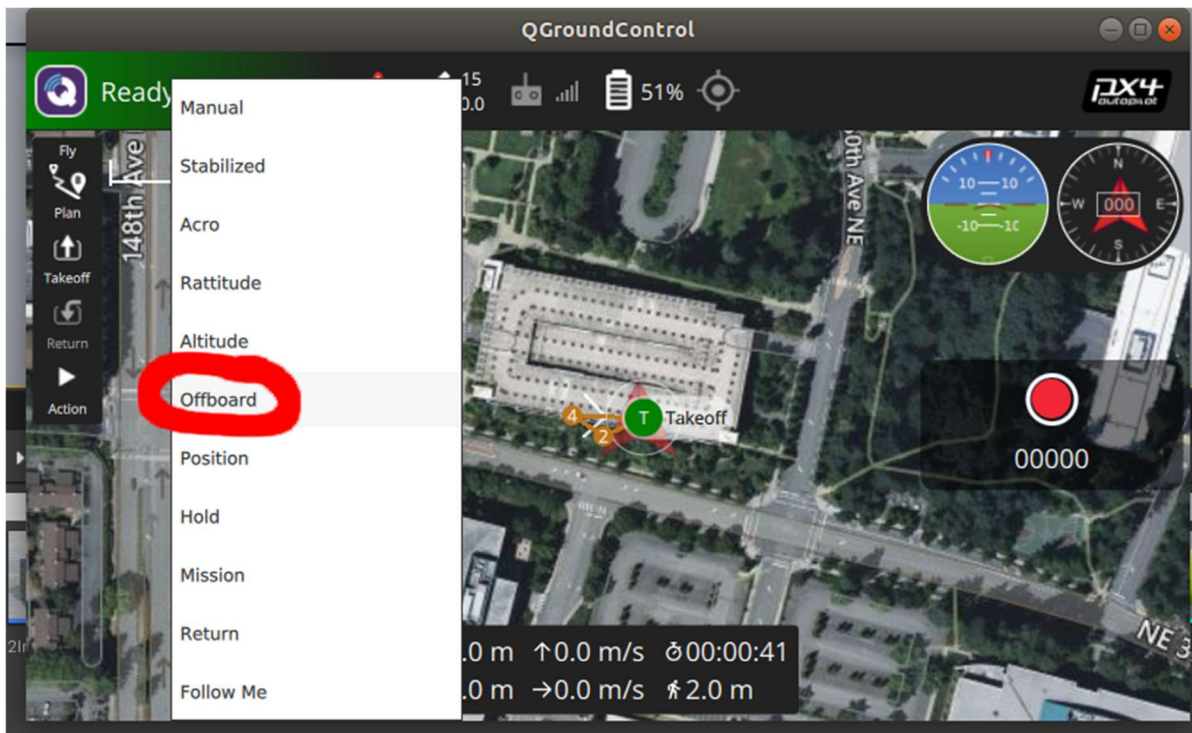
```
source ~/px4_ros_com_ros2/install/setup.bash
```

```
ros2 run px4_ros_com mt_offboard_control
```

Una volta eseguito l'ultimo comando, aspettare qualche secondo e settare la modalità di volo come "Offboard" su QgroundControl come indicato nelle figure successive:

CLICCARE SUI PUNTI CERCHIATI IN ROSSO:





Se tutto è stato eseguito in maniera corretta, il drone inizierà a volare.

### IMPORTANTE! COME CHIUDERE LA SIMULAZIONE

Seguire questo ordine nel chiudere la simulazione, quando ha finito:

- 1) premere la combinazione di tasti CTRL+C (oppure chiudere direttamente) sui terminali dal 3 all'8 (non importa l'ordine);
- 2) chiudere la finestra della simulazione di Unreal Engine;
- 3) premere la combinazione tasti CTRL+C (oppure chiudere direttamente) il terminale 2 (dove era stato avviato PX4).

Se non si segue questo ordine (chiudendo PX4 all'ultimo), per riavviare la simulazione si deve riavviare il PC.