vision & aerial robotics

CAR
CENTRO DE AUTOMÁTICA Y ROBÓTICA

CSIC

POLITÉCNICA

Aerostack2

The ROS 2 aerial robotics framework

Miguel Fernandez-Cortizas
20/10/2023 ROSCON'23 New Orleans

What is Aerostack2?

Aerostack2 is an open-source software framework that helps developers **design and build multirobot aerial robotic systems**. It is designed with ROS 2 and its part of their ecosystem.

It is an evolution of the former **Aerostack** [1] framework developed and used succesfully in the CVAR Group since 2016.

[1] J.L. Sanchez-Lopez, M. Molina, H. Bavle, C. Sampedro, R. A. Suarez-Fernandez, P. Campoy. (2017).
A Multi-Layered Component-Based Approach for the Development of Aerial Robotic Systems: The Aerostack Framework.
Journal of Intelligent & Robotic Systems, 88, 683–709.

What is our mission?

Ease the development of **autonomous** aerial robot systems

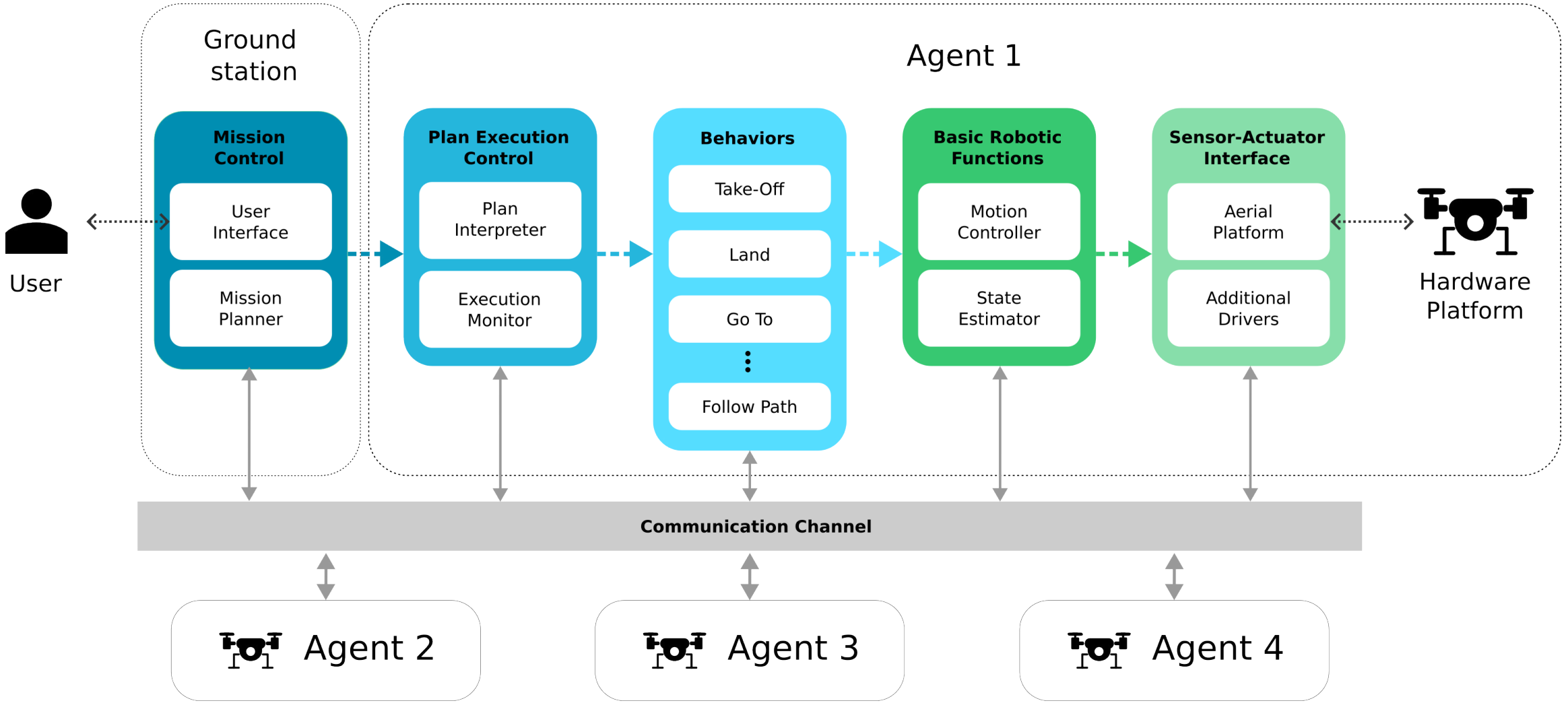
Develop a suitable framework for **academic** and **industrial** applications

Create ROS 2 community that enhances the development of a **common framework**

Which were our requirements?

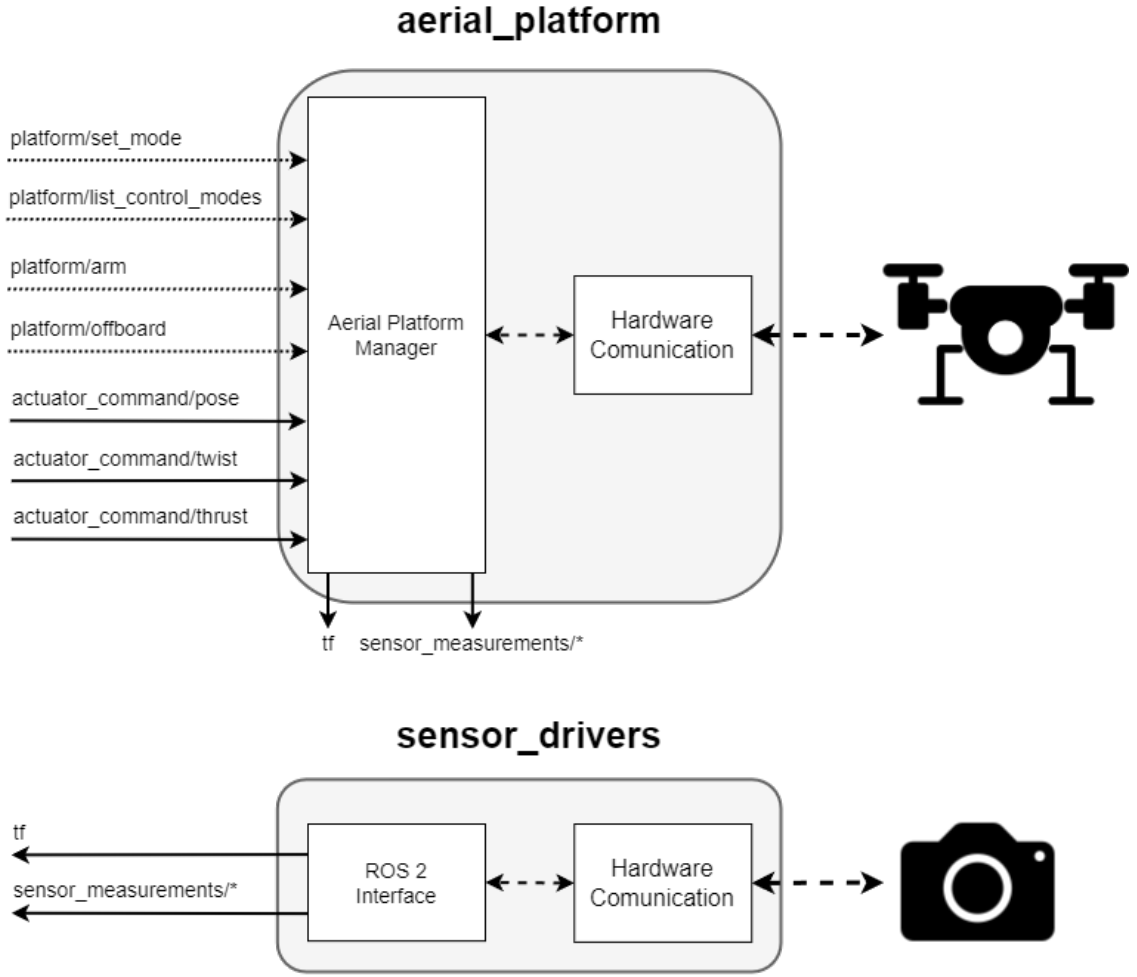
1. Modularity and flexibility
2. Support a variety of platforms
3. Handle multiple drones
4. Easy to create new missions
5. Allows a safe system development
6. Support indoor & outdoor flights operations

Aerostack2 architecture



Sensor actuator interface

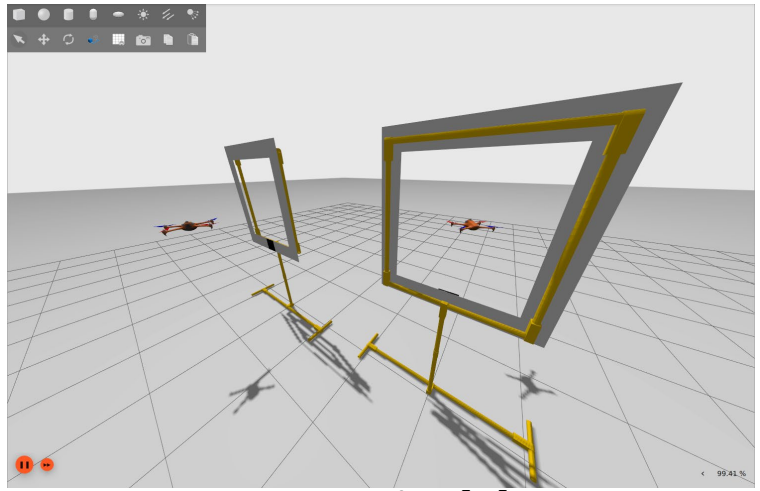
- Platform agnostic
- Easy Sim 2 Real deployment
- Heterogeneous swarms
- Standardized sensor Communication in ROS 2



Supported platforms

Bitcraze Crazyflie 2.X

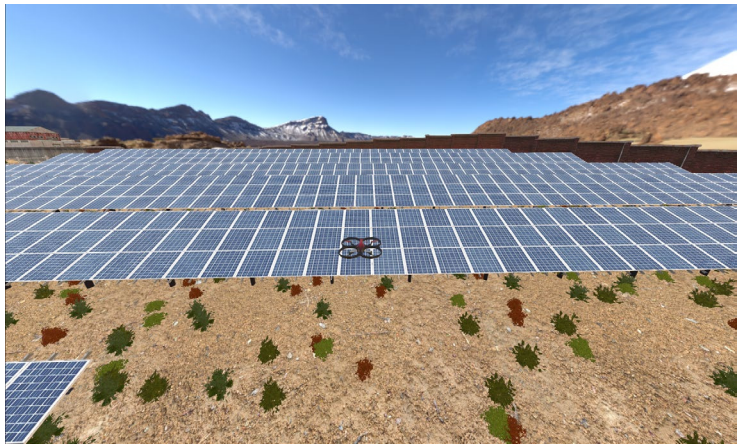
Ryze Tello



Gazebo [1]

DJI Matrice 200/300 series

PX4 based platforms



Flightmare [2]

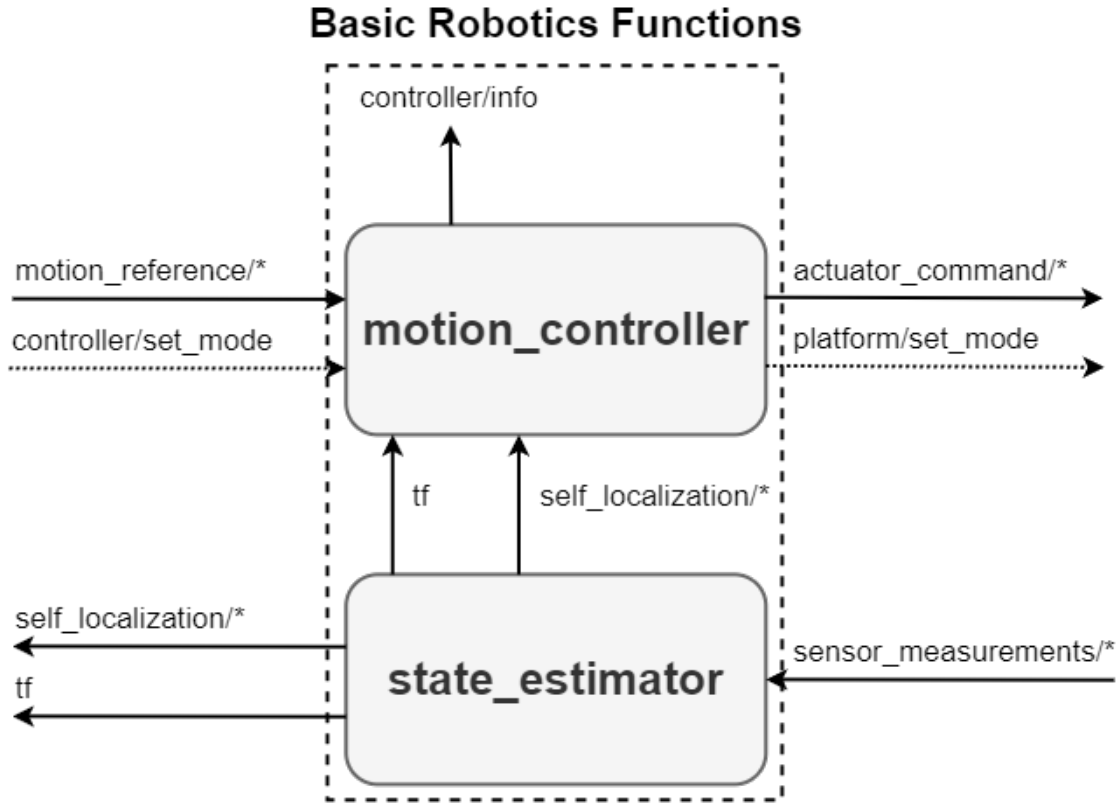
[1] https://github.com/gazebosim [2] https://github.com/uzh-rpg/flightmare

Basic robotic functions

A collection of software components that perform **fundamental robotic functions** in aerial robotics:

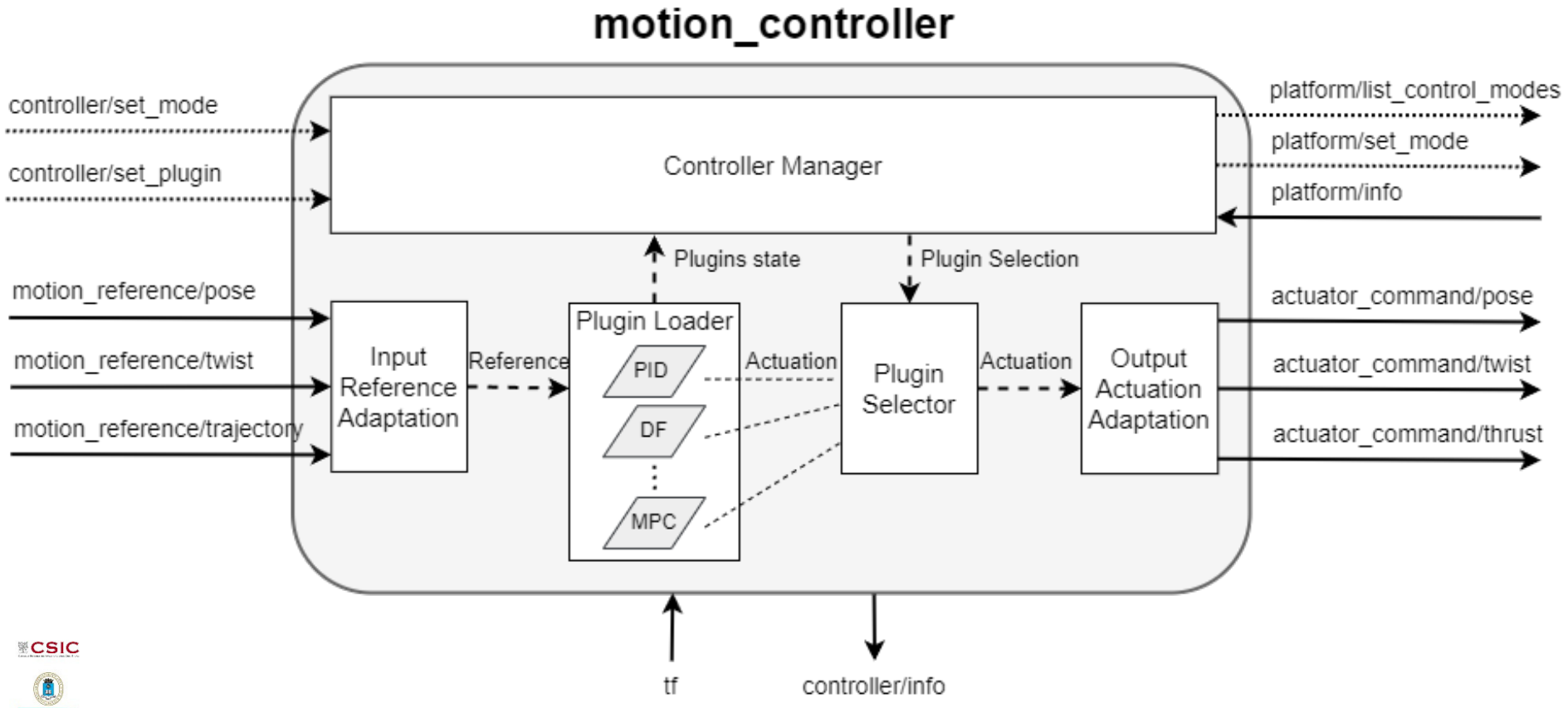
1. Motion controller
2. State estimator

They are designed to be general and reusable with alternative algorithms (**Plugins**).



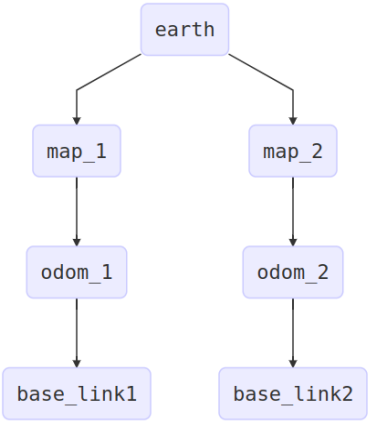
Motion controller

- Load multiples plugins and select the output of one
- Satisfy the motion command requirements constrained by the platform available control modes
- Adapt frames of references and output commands

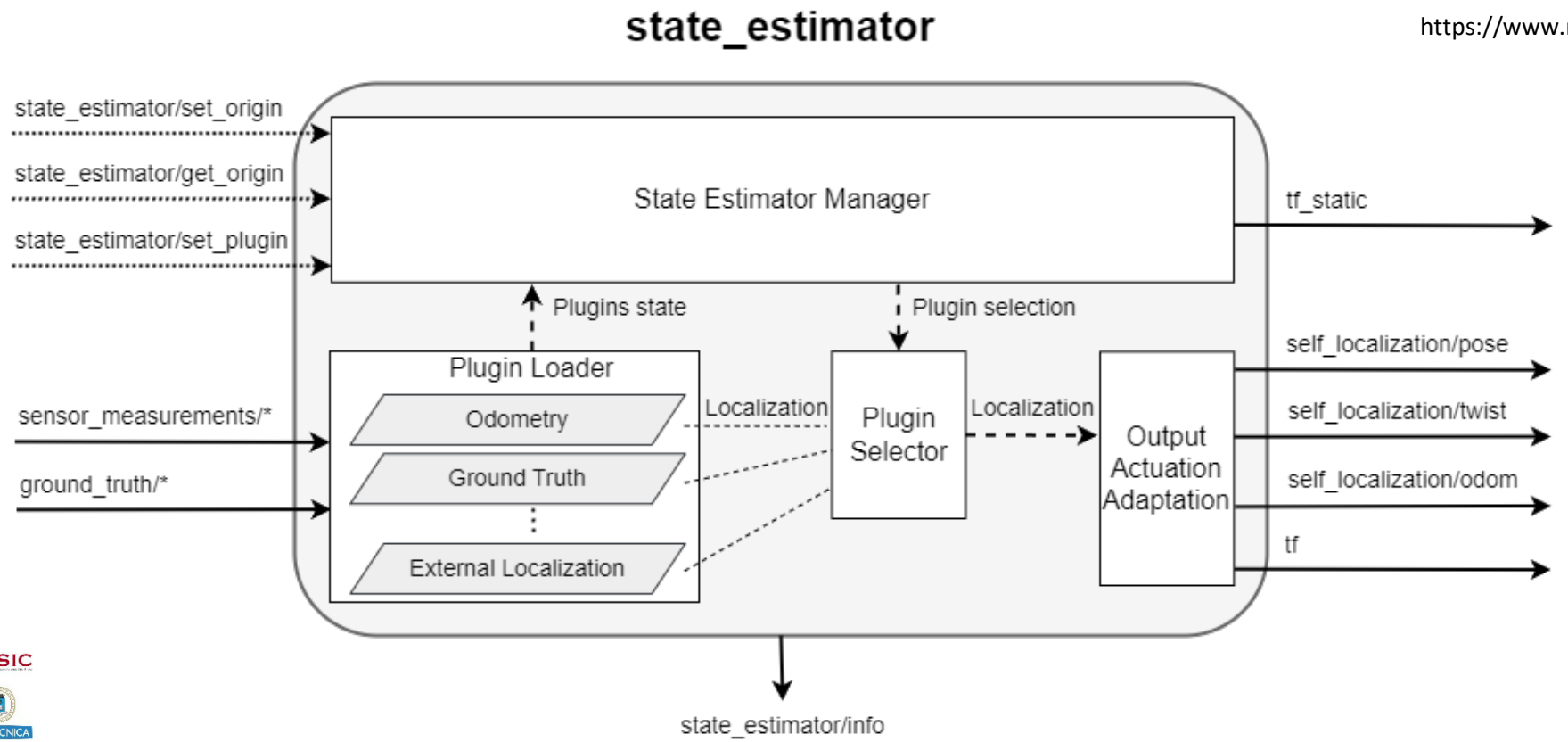


State estimator

- Select the plugin using the sensor input data available to generate "map" and "odom" localization
- Use ground truth data or GPS origin to generate "earth" localization
- Manage a common origin with others agents

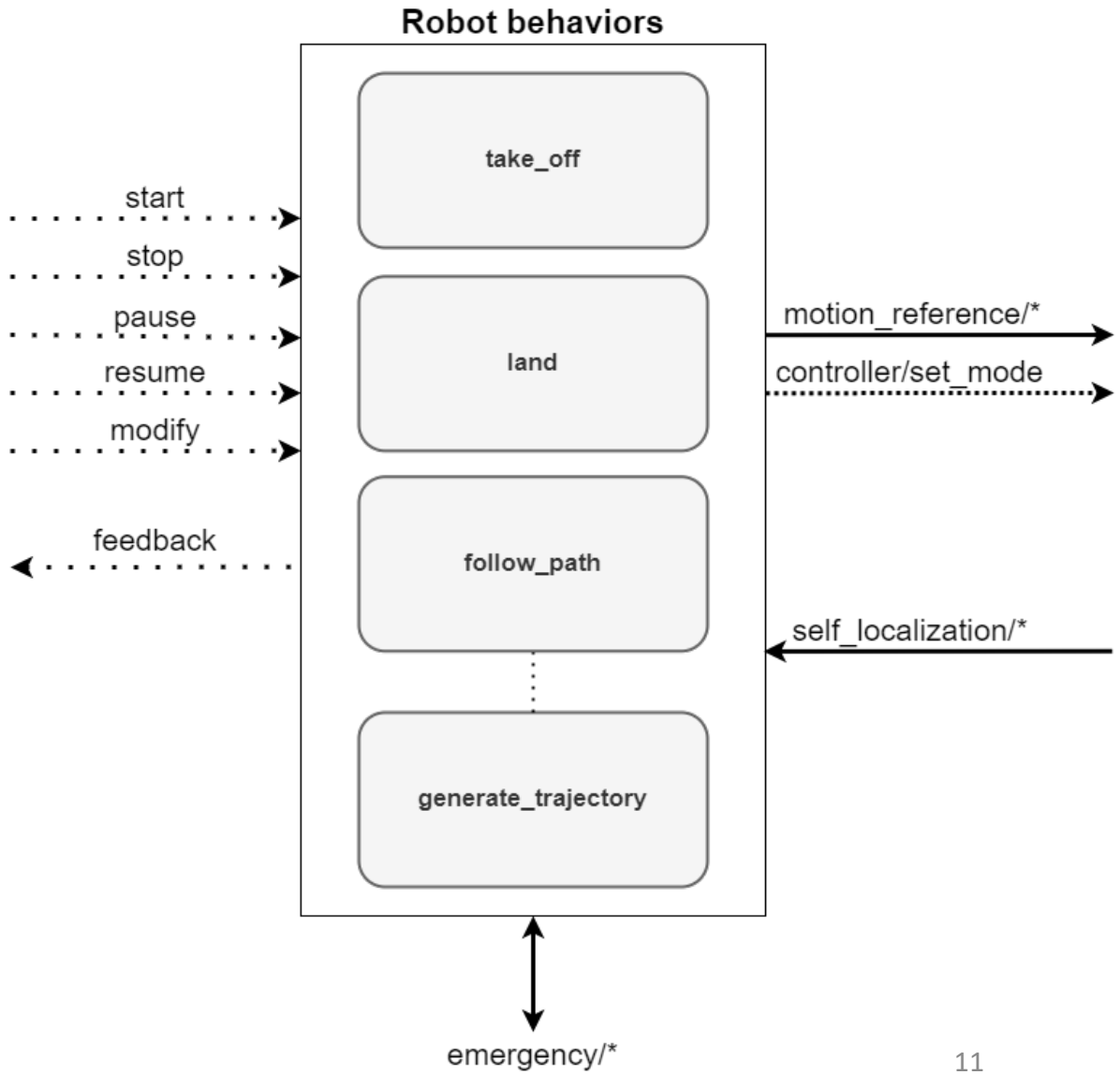


https://www.ros.org/reps/rep-0105.html



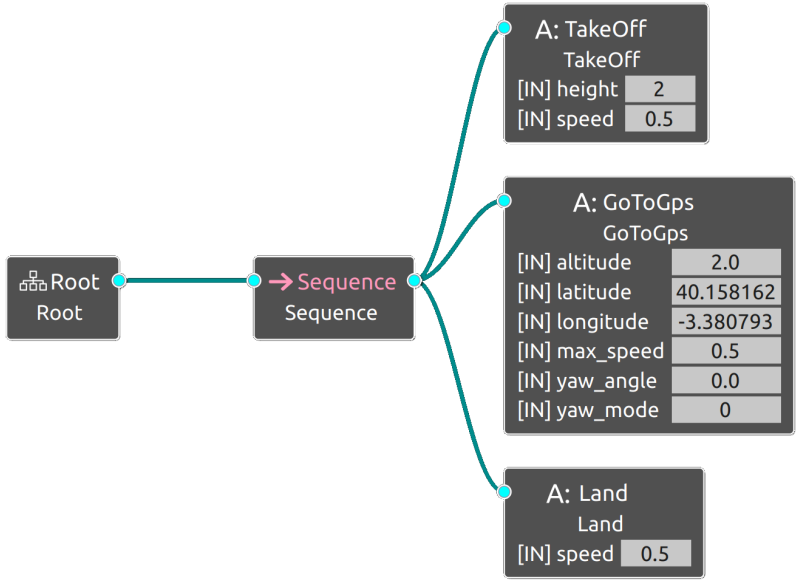
Behaviors

- Each behavior corresponds to a specific robot skill related
- Each behavior encapsulates the control and execution monitoring of its task
- They are implemented extending ROS 2 actions to provide additional capabilities



Plan execution control

- Orchestrates and monitorizes the activation and deactivation of the different behaviors
- Different ways:
  - AS2 Python API
  - Behavior Trees [3]
  - AS2 Mission plan interpreter
- Depending on the application a Local Planner can be designed for modifying the plan



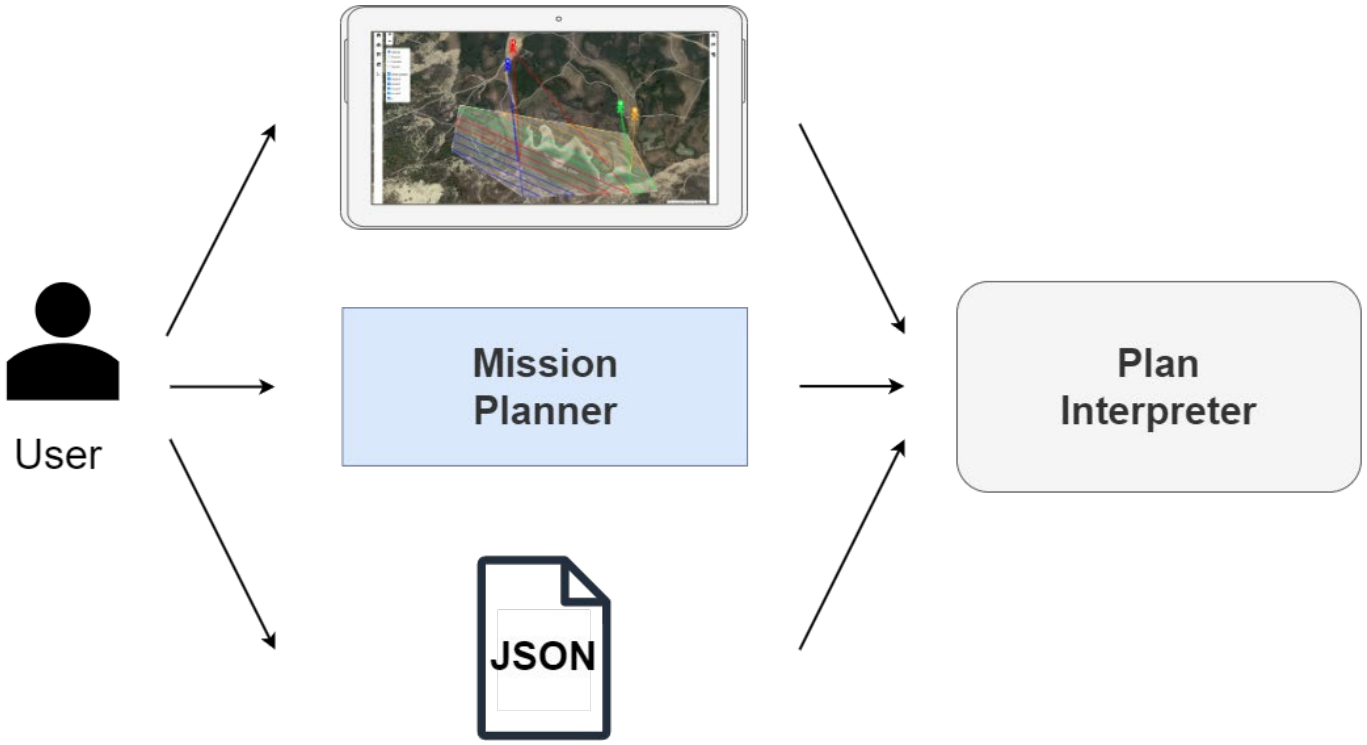

```python
import rclpy
from as2_python_api.drone_interface import DroneInterface

rclpy.init()
drone = DroneInterface(drone_id="drone0")

drone.takeoff(height=1.0, speed=0.5)
drone.go_to.go_to_point([1.0, 1.0, 2.0], speed=0.5, frame_id="earth")
drone.land(speed=0.5)

drone.shutdown()
rclpy.shutdown()
```

[3] https://github.com/BehaviorTree/BehaviorTree.CPP

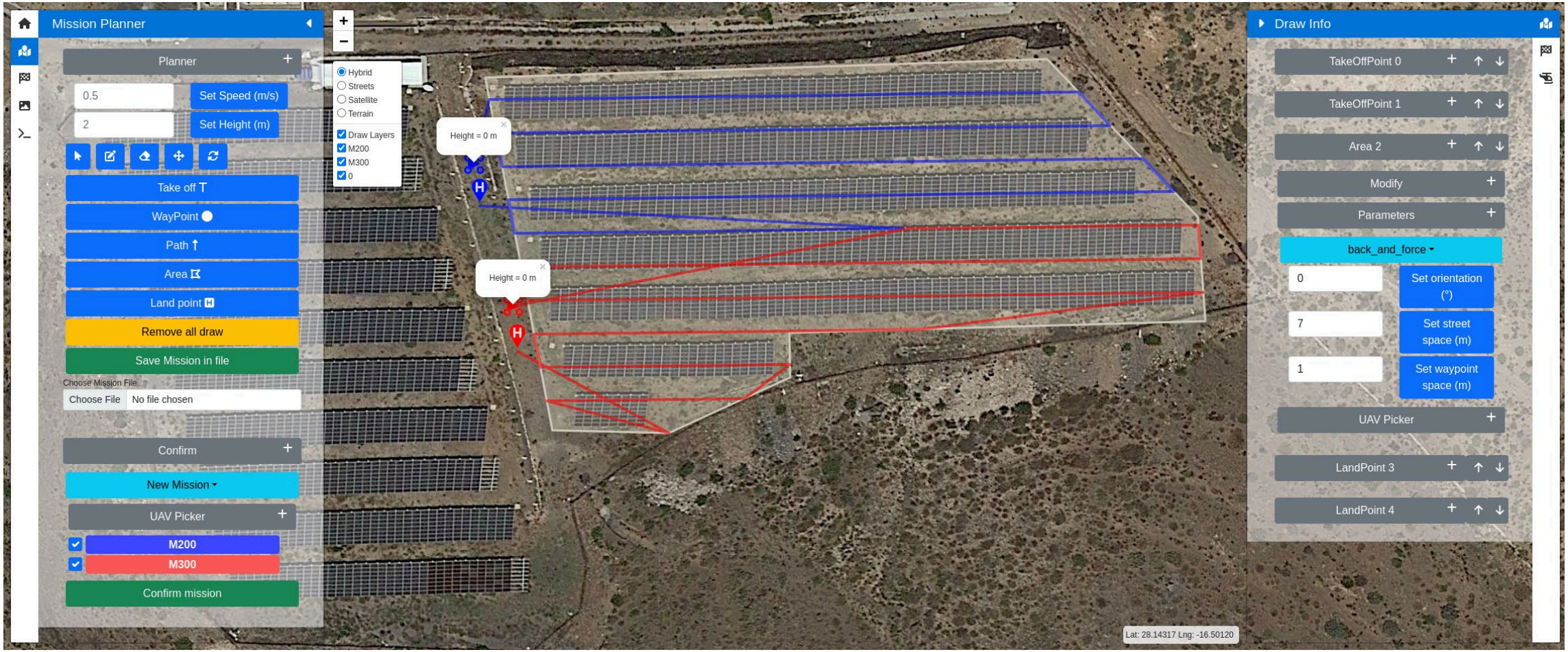
Mission specification

Different ways for generating a plan:

- Use of Web GUI (for outdoor missions)
- Design of an specific mission planner
- Defined by the user manually

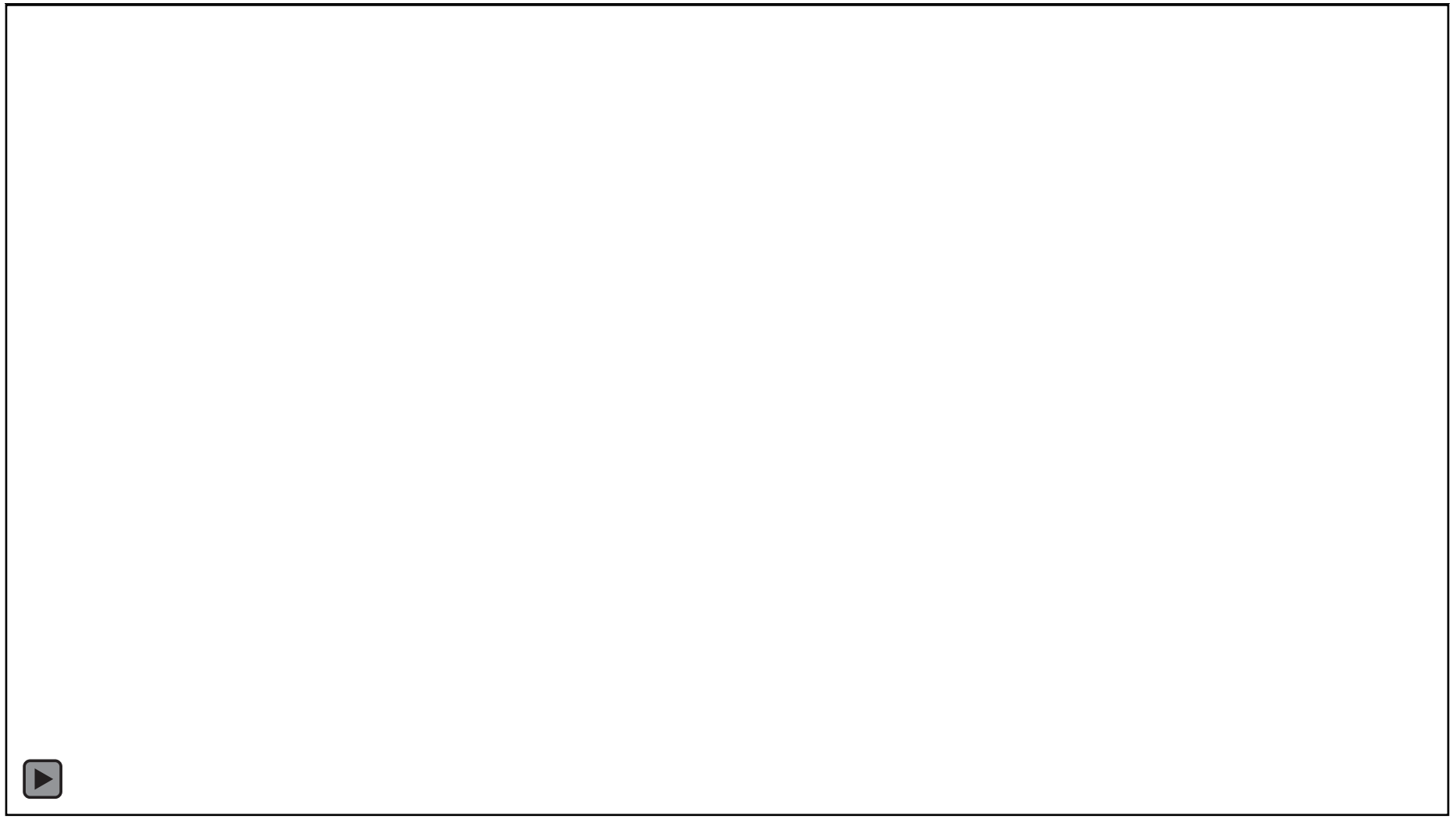


Web GUI

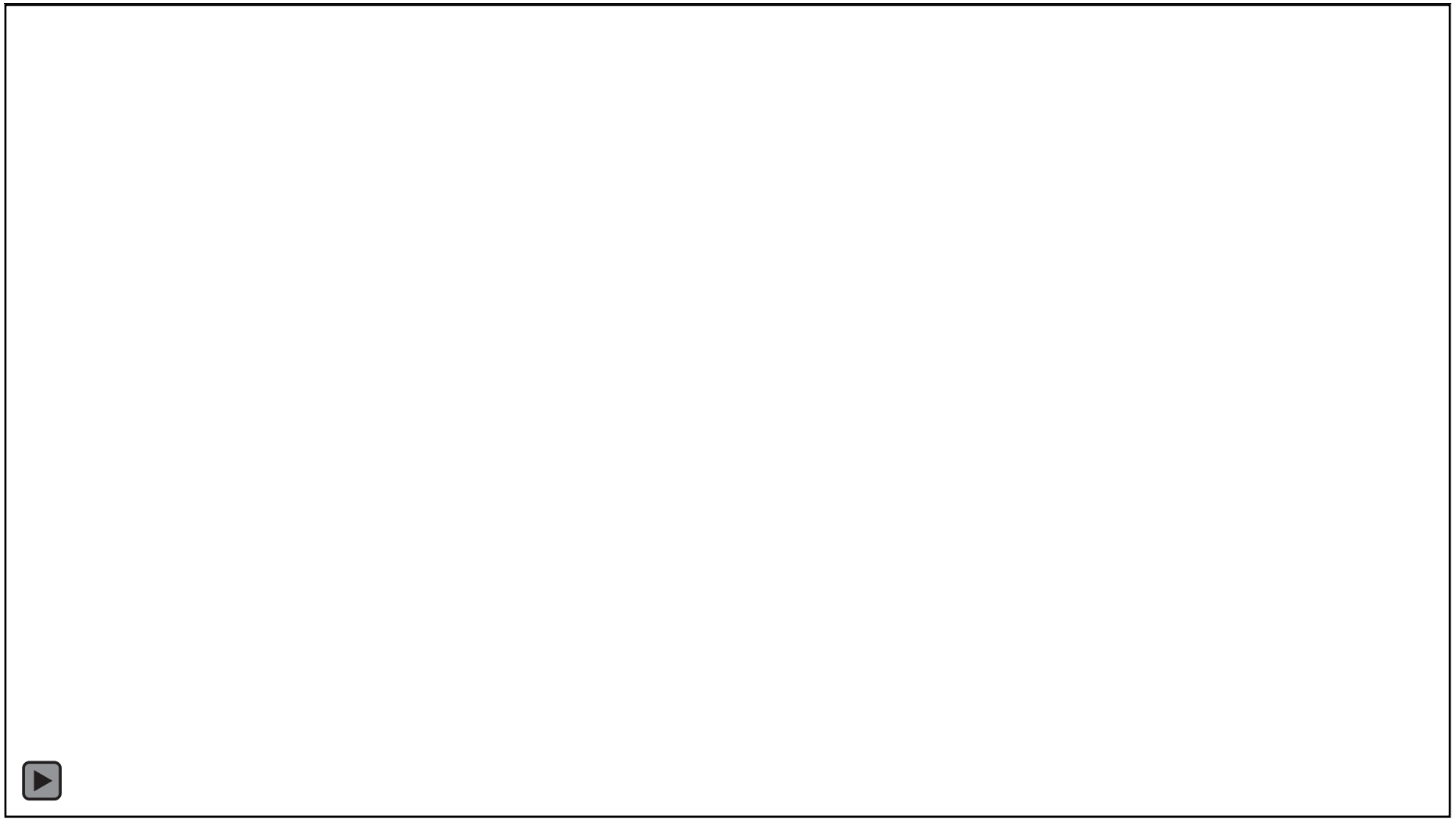


Mission Planning | **UAVs Monitoring** | **Mission Parameters**

Multi-robot PV plant inspection



Multi-robot PV plant inspection



Mission planning example

Mission Specification

```
{
  "blade_length": 15.0,
  "inspection_distance": 10.0,
  "vertical_overlap": 0.2,
  "inspection_angles": [0],
  "security_height": 10.0,
  "security_distance": 10.0
}
```

Mission Planner

Mission Plan

Target Approach

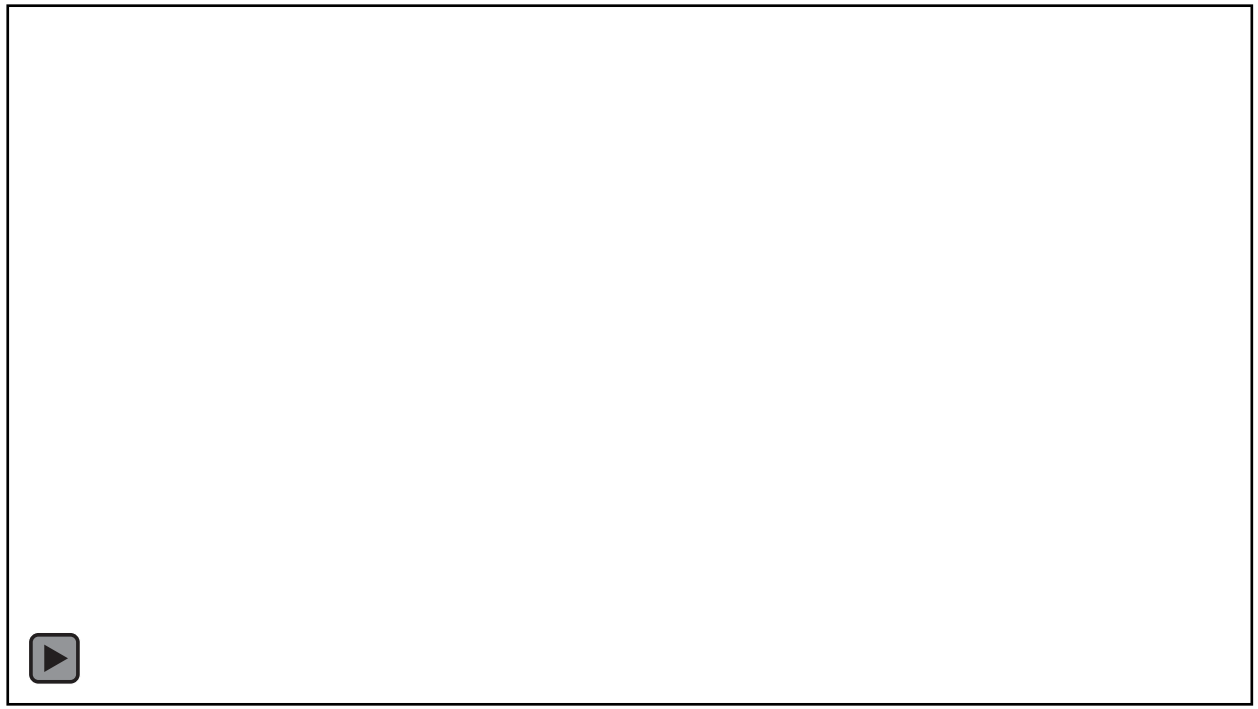
Blades Inspection

Return Home

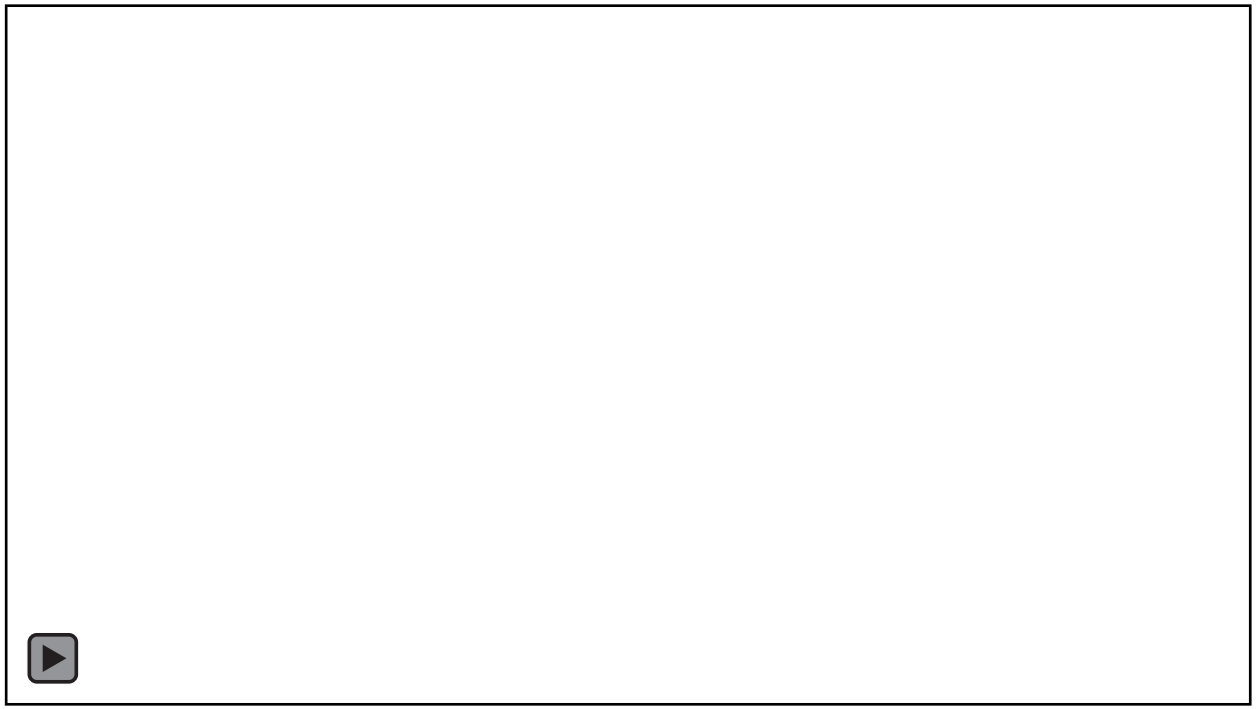
```
{
"target": "drone0",
"plan": [
{
   "behavior": "takeoff",
   "args": { "height": 1.0, "speed": 0.5}},
{
  "behavior": "go_to",
  "args": { "_x": 0.0, "_y": 0.0, "_z": 20.0, "frame_id": "earth" }},
{
  "behavior": "go_to",
  "args": { "_x": 10.0, "_y": 0.0, "_z": 0.0, "frame_id": "wind_turbine" }},
{
  "behavior": "follow_reference",
  "args": { "_x": 10.0, "_y": 0.0, "_z": 0.0, "frame_id": "wind_turbine" }},
{
  "behavior": "follow_reference",
  "args": { "_x": 10.0, "_y": 0.0, "_z": 2.26, "frame_id": "wind_turbine" }},
{
  ...
{
   "behavior": "follow_reference",
   "args": { "_x": 10.0, "_y": 0.0, "_z": 13.55, "frame_id": "wind_turbine" }},
{
   "behavior": "go_to",
   "args": { "_x": 0.0, "_y": 0.0, "_z": 20.0, "frame_id": "earth" }},
{
  "behavior": "land",
  "args": { "speed": 0.5 }}]
}
```

Windmill inspection

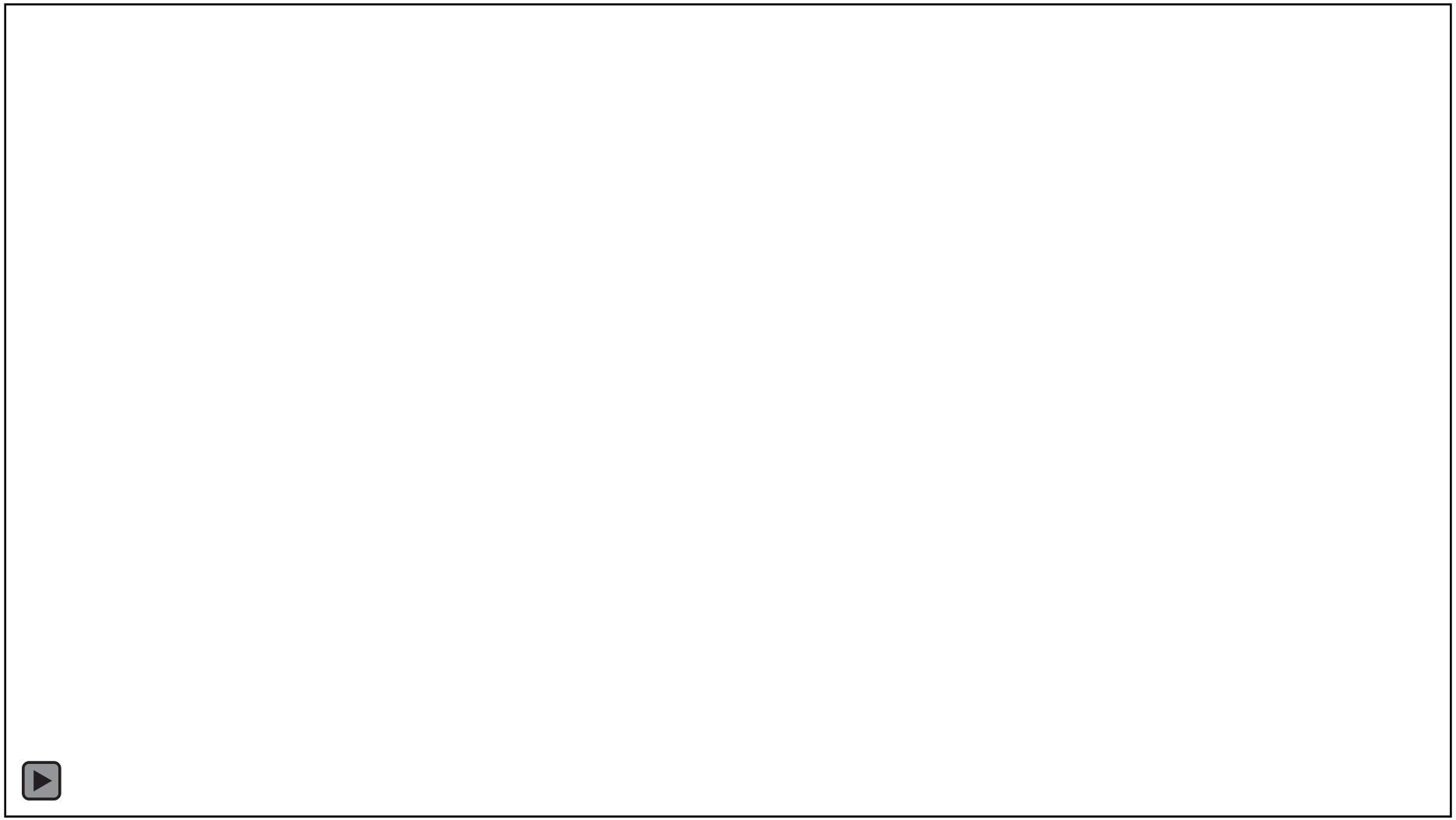
Simulation inspection



Augmented reality inspection

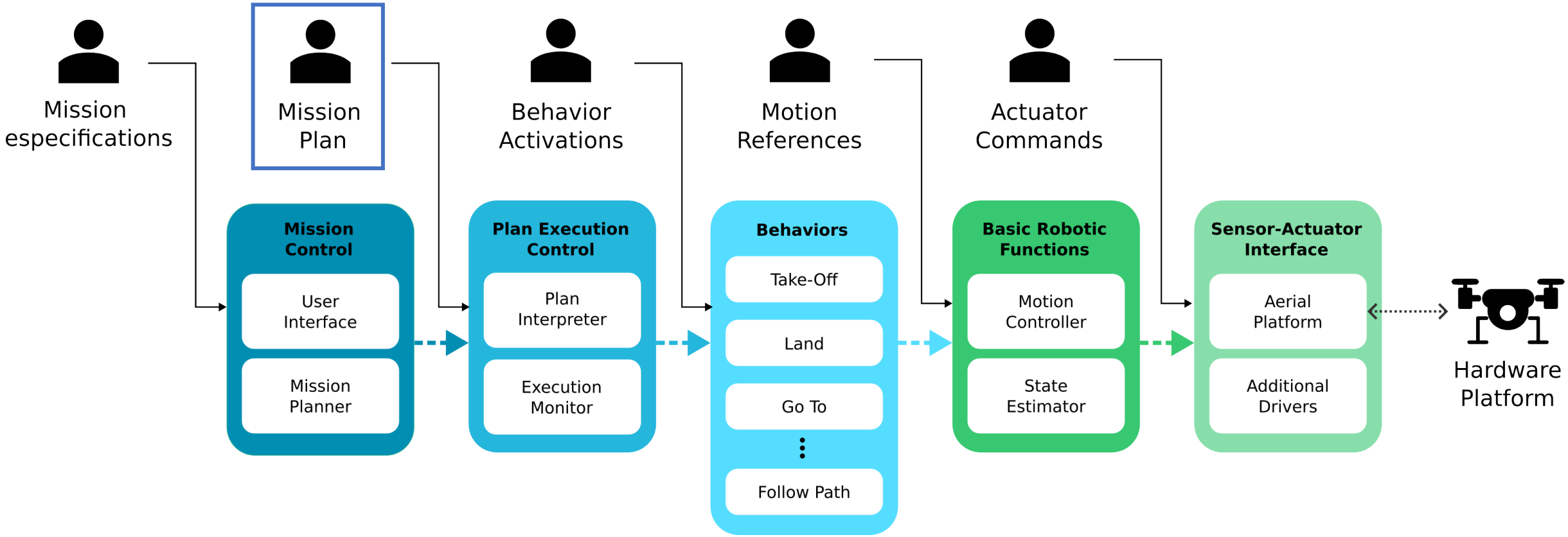


Multi-robot robust area coverage



How to get started?

- Aerostack2 can be used at different levels depending on the user needs



Launching AS2

```
$ ros2 launch as2_platform_ign_gazebo ign_gazebo_launch.py
    namespace:=cf1
    use_sim_time:=true
$ ros2 launch as2_state_estimator state_estimator_launch.py
    namespace:=cf1
    use_sim_time:=true
    plugin_name:=ground_truth
$ ros2 launch as2_motion_controller controller_launch.py
    namespace:=cf1
    use_sim_time:=true
    plugin_name:=pid_speed_controller
$ ros2 launch as2_behaviors_motion motion_behaviors_launch.py
    namespace:=cf1
    use_sim_time:=true
    follow_path_plugin_name:=follow_path_plugin_position
    go_to_plugin_name:=go_to_plugin_position
    takeoff_plugin_name:=takeoff_plugin_position
    land_plugin_name:=land_plugin_speed
$ python3 mission.py
```

Aerial Platform

State Estimator

Motion Controller

Behaviors

Mission

Going real

```
$ ros2 launch as2_platform_crazyflie crazyflie_swarm_launch.py
    swarm_config_file:=swarm_config_file.yaml
$ ros2 launch as2_state_estimator state_estimator_launch.py
    namespace:=cf1
    use_sim_time:=false
    plugin_name:=mocap_pose
$ ros2 launch as2_motion_controller controller_launch.py
    namespace:=cf1
    use_sim_time:=false
    plugin_name:=pid_speed_controller
$ ros2 launch as2_behaviors_motion motion_behaviors_launch.py
    namespace:=cf1
    use_sim_time:=false
    follow_path_plugin_name:=follow_path_plugin_position
    go_to_plugin_name:=go_to_plugin_position
    takeoff_plugin_name:=takeoff_plugin_position
    land_plugin_name:=land_plugin_speed
$ python3 mission.py
```

Changes:

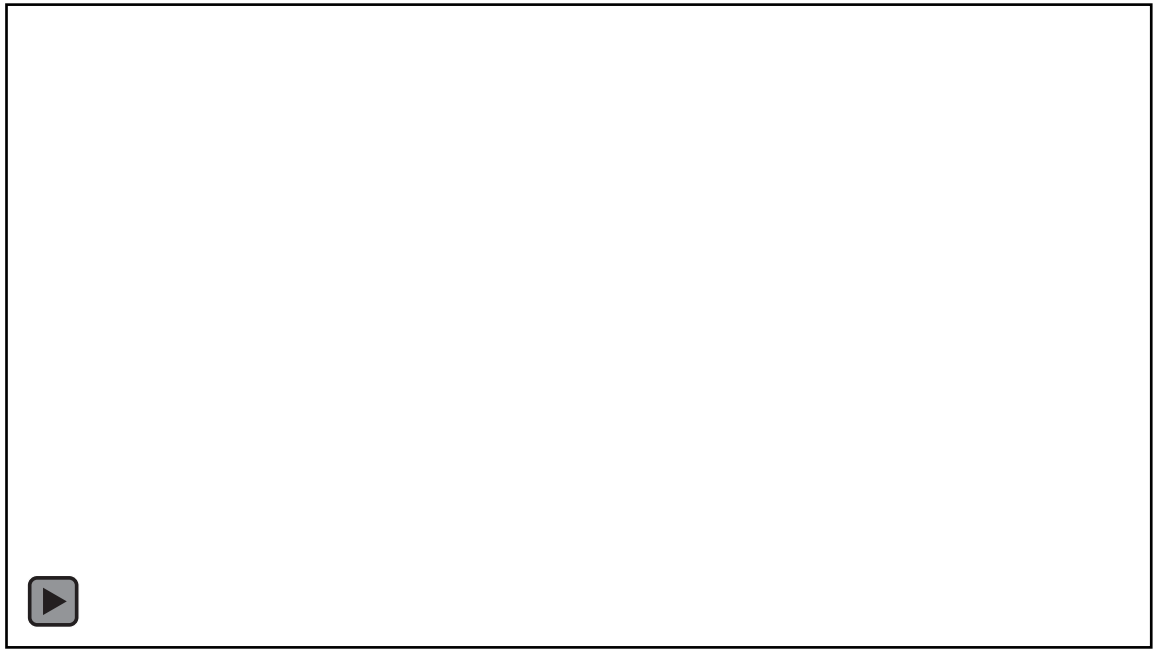
1. Select Aerial Platform node

```
# swarm_config_file.yaml
--
cf1:
    uri: radio://0/80/2M/E7E7E7E701
```

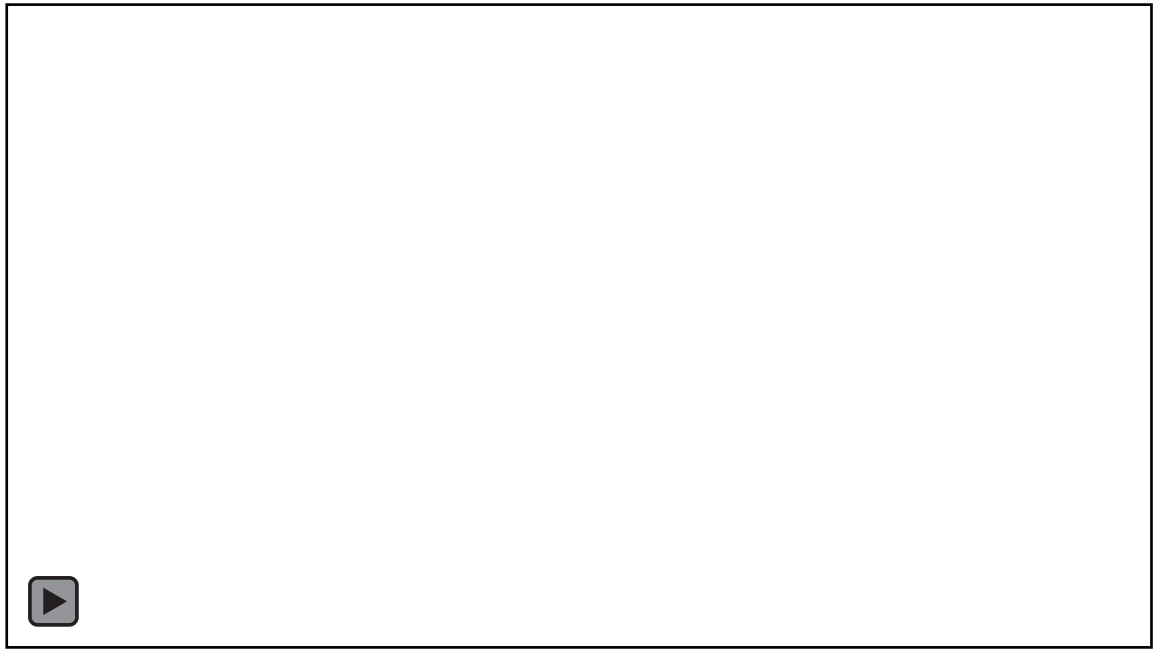
2. Disable use_sim_time
3. Change estimation plugin

Running the experiments

Simulation

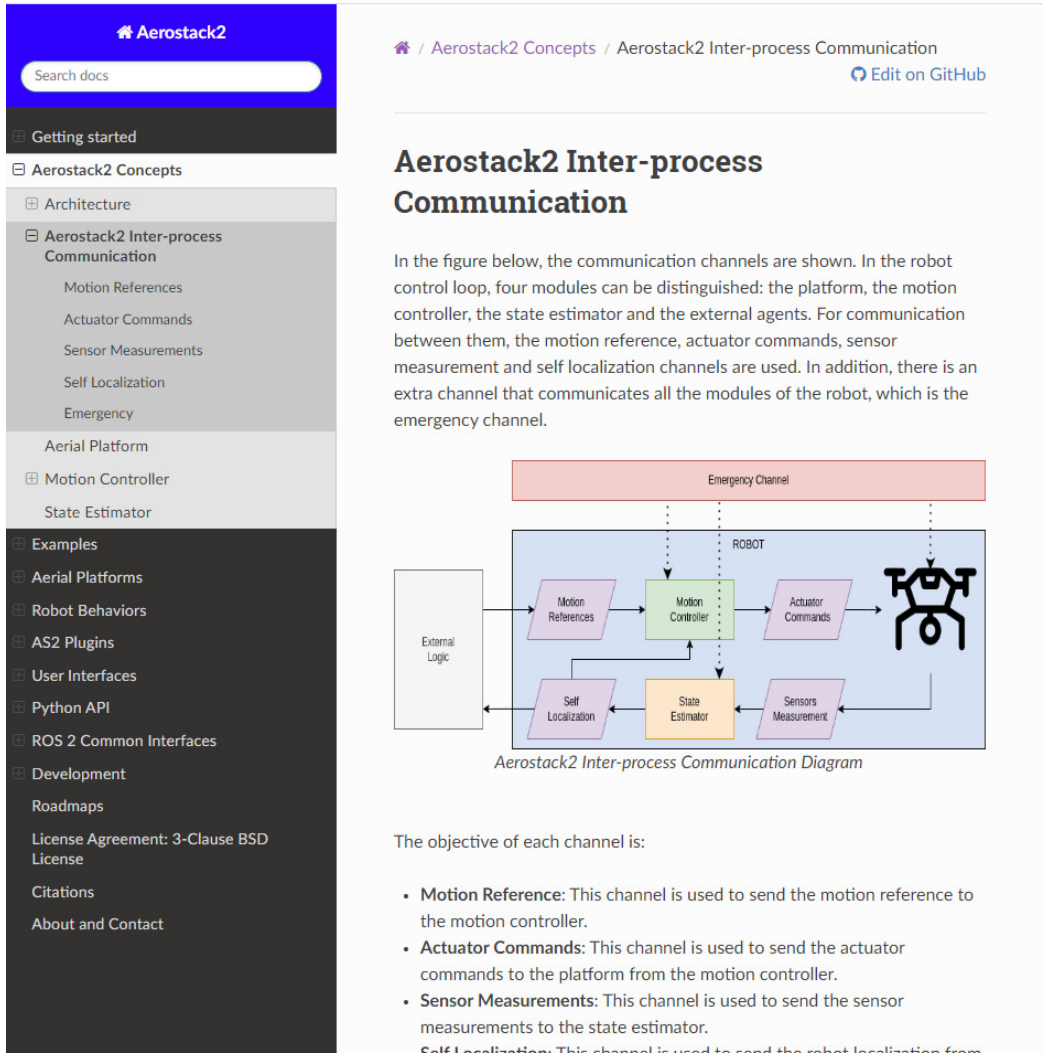


Real



Further resources

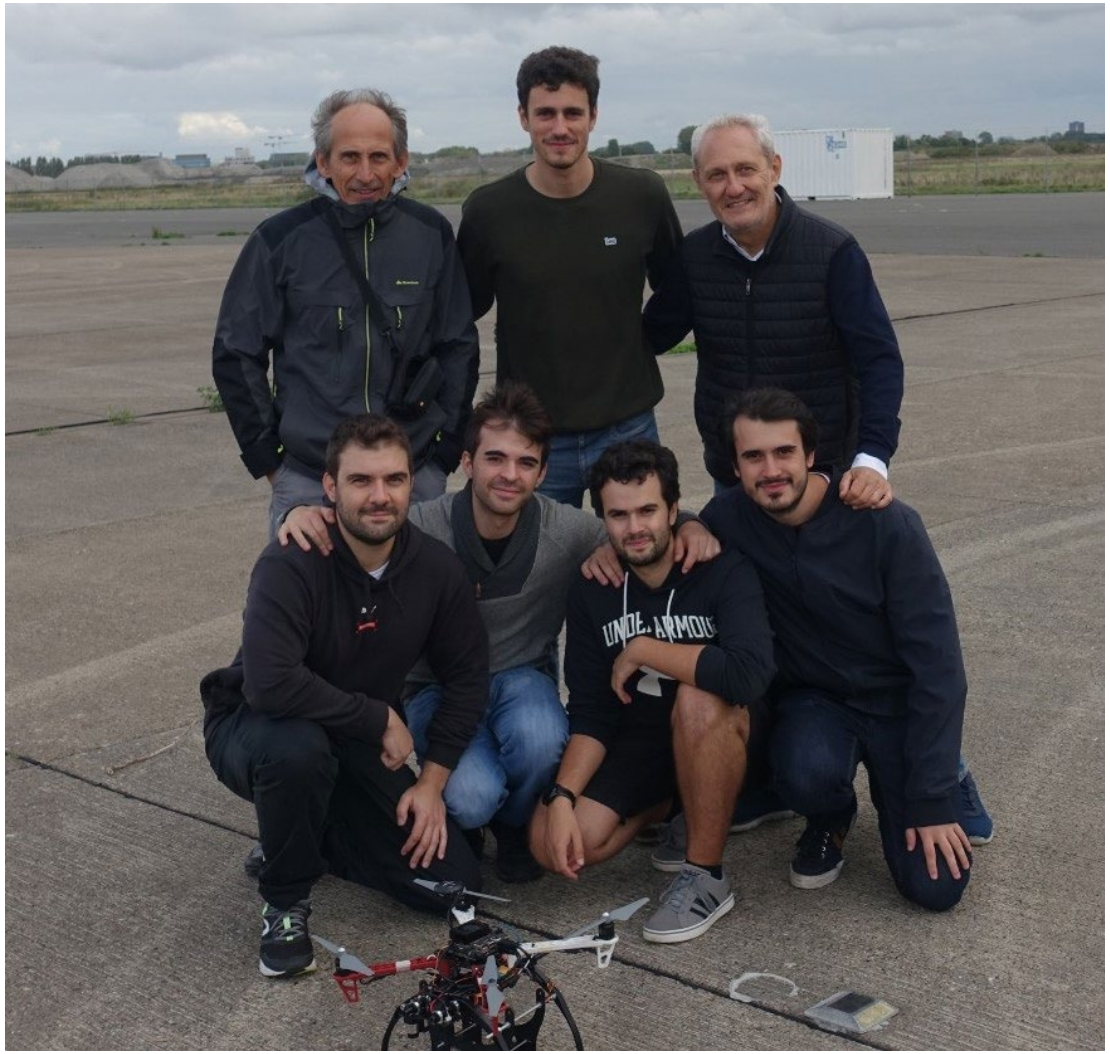
- Documentation: https://aerostack2.github.io/
- Github: https://github.com/aerostack2/aerostack2/
- Paper: (Preprint) M. Fernandez-Cortizas, M. Molina, P. Arias-Perez, R. Perez-Segui, D. Perez-Saura, and P. Campoy, 2023, "Aerostack2: A software framework for developing multi-robot aerial systems", ArXiv DOI 2303.18237



Our team

Main contributors:

- Miguel Fernandez-Cortizas
- Martin Molina
- Pedro Arias-Perez
- Rafael Perez-Segui
- David Perez-Saura
- Javier Melero-Deza
- Pascual Campoy



Funding:

MINISTERIO DE CIENCIA E INNOVACIÓN

Plan de Recuperación, Transformación y Resiliencia

Financiado por la Unión Europea NextGenerationEU

AGENCIA ESTATAL DE INVESTIGACIÓN

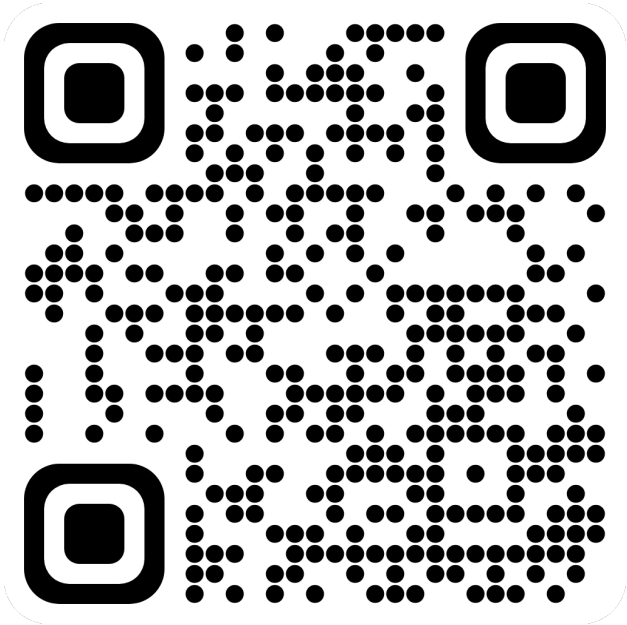
vision & aerial robotics

CAR CENTRO DE AUTOMÁTICA Y ROBÓTICA

CSIC

POLITÉCNICA

Thanks for your attention!!!



Aerostack2

vision & aerial robotics

CAR CENTRO DE AUTOMÁTICA Y ROBÓTICA

CSIC

POLITÉCNICA