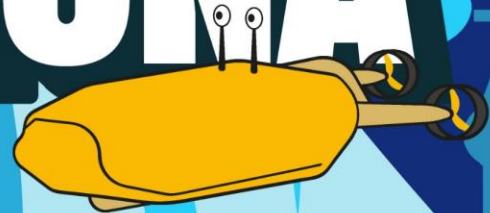
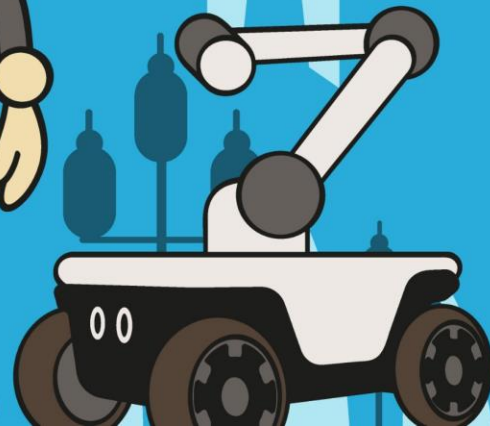
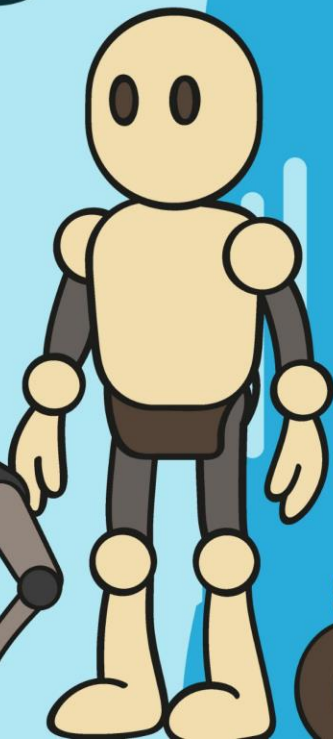
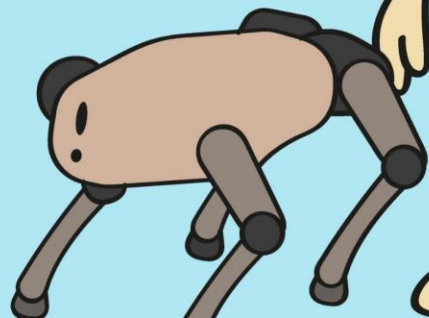
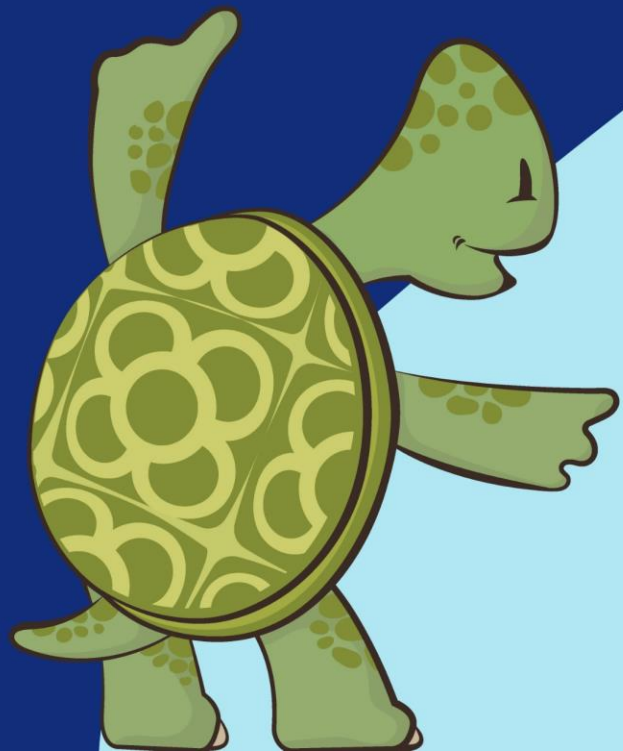


ROSCon ES '25 BARCELONA



Aerostack2, desarrolla
tu enjambre de drones
desde simulación a real



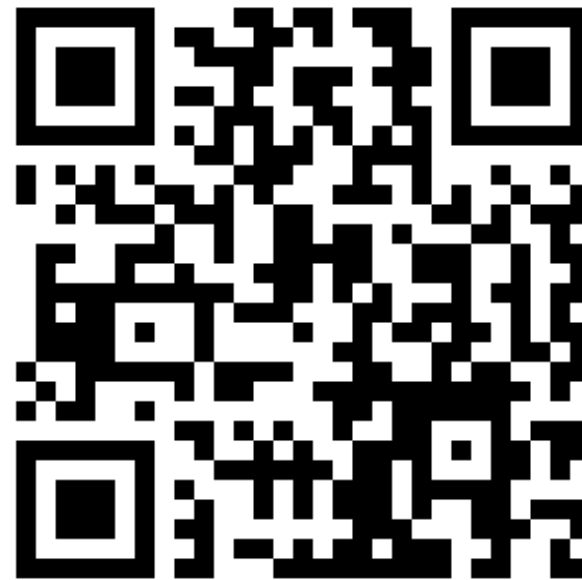
Pedro Arias y Guillermo García

Researcher

CVAR, Universidad Politécnica de Madrid



Aerostack2, desarrolla
tu enjambre de drones
desde simulación a real



[https://github.com/aerostack2/demo
_ROSConES25](https://github.com/aerostack2/demo_ROSConES25)



Agenda



- ¿Qué es Aerostack2?
- Arquitectura
- Ejemplos prácticos:
 - Control por referencias al controlador.
 - Control de ejecución de plan y estructura de un proyecto.
 - Paso a real, uso de Crazyflies.
 - Misión multi-agente compleja: Drone-convoy.
- Otros casos de uso

¿Qué es Aerostack2?

Aerostack2 es un framework de código abierto destinado al diseño y creación de sistemas aéreos multiagentes, basado en ROS 2.

Es una evolución de su predecesor Aerostack, usado por nuestro grupo Computer Vision & Aerial Robotics (CVAR) Group desde 2016.





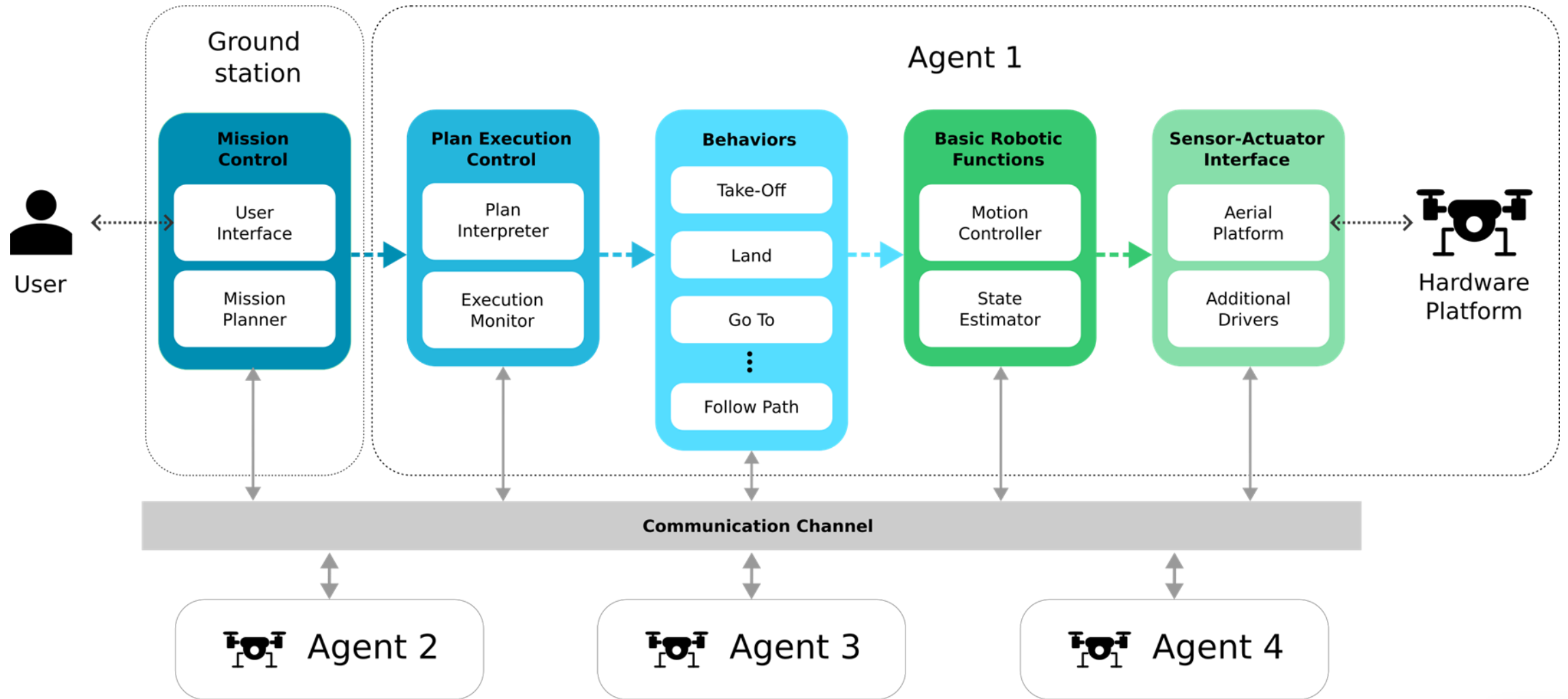
Especificaciones



1. Modular y flexible
2. Soporte de distintas plataformas
3. Manejo de enjambres
4. Facilidad en la creación de misiones
5. Desarrollo seguro de sistemas
6. Soporte de vuelos en interior y exterior

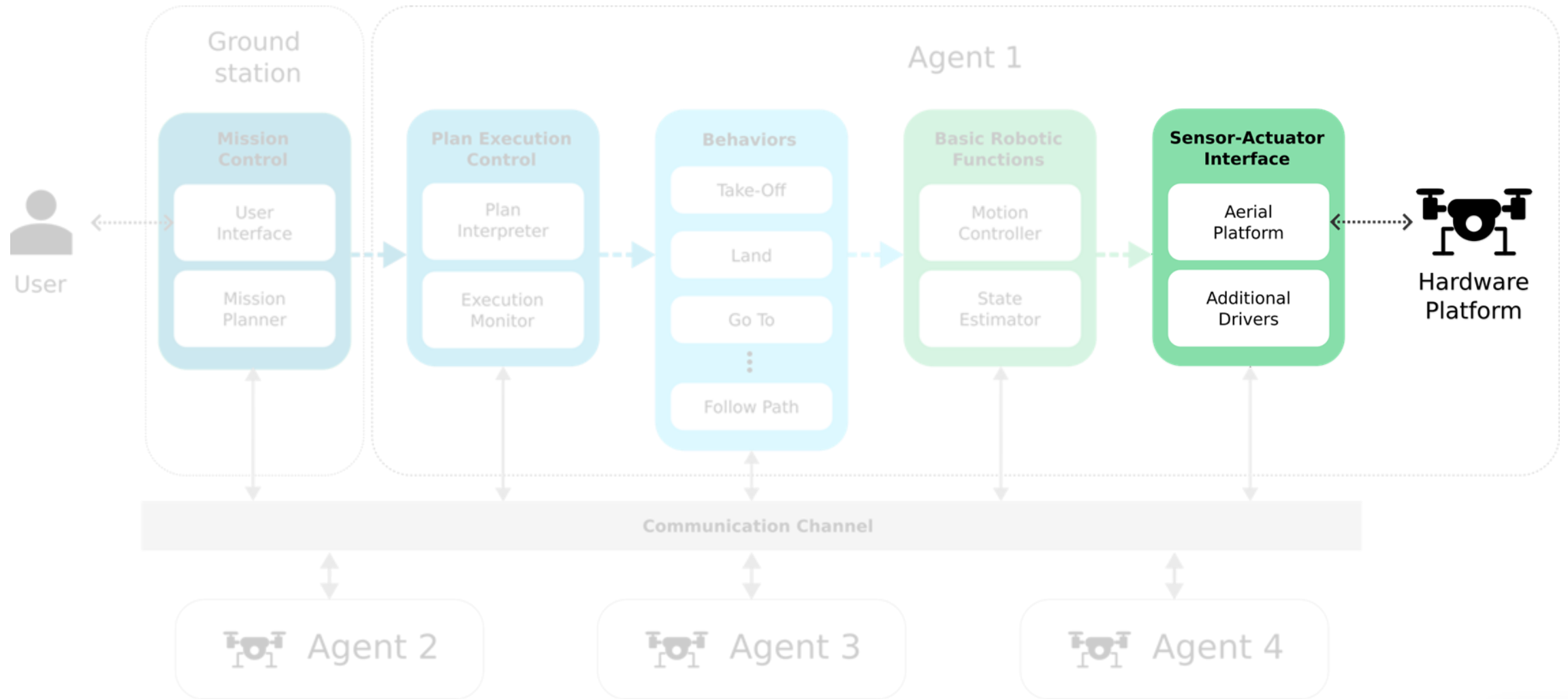


Arquitectura





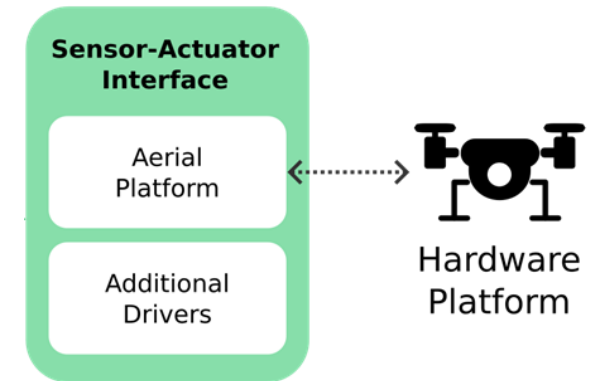
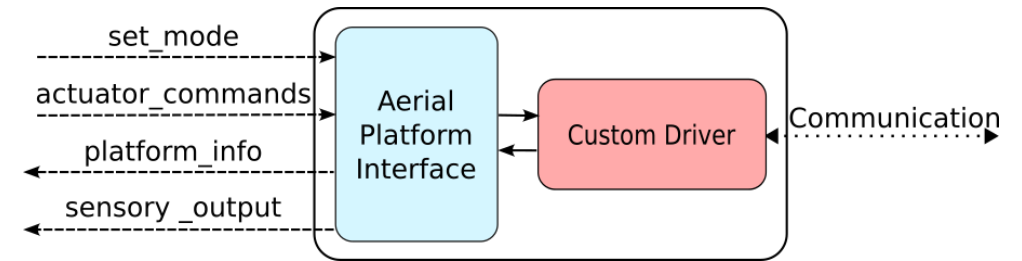
Arquitectura



Interfaz con los actuadores y sensores

Interfaz entre las distintas plataformas aéreas y sus sensores con Aerostack2.

- Agnóstica a la plataforma.
- Facilita el paso de simulación a real.
- Permite trabajar con enjambres heterogéneos.
- Emplea los estándares de comunicación de ROS 2.





Plataformas soportadas



Bitcraze
Crazyflie 2.X



DJI Ryze Tello



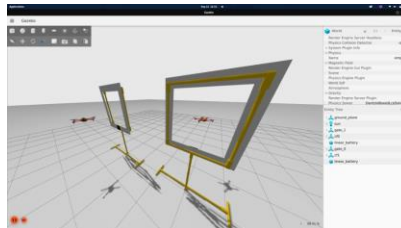
PX4 / MAVlink



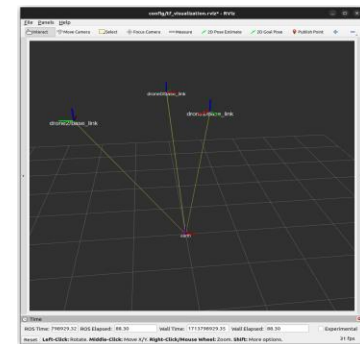
Betaflight



DJI Matrice



Gazebo



AS2 Multirotor
Simulator



Gazebo Platform



1. Definir la simulación (world.yaml)
2. Lanzar el simulador de Gazebo

```
ros2 launch as2_gazebo_assets launch_simulation.py  
simulation_config_file:=<%= simulation_config_file %>
```

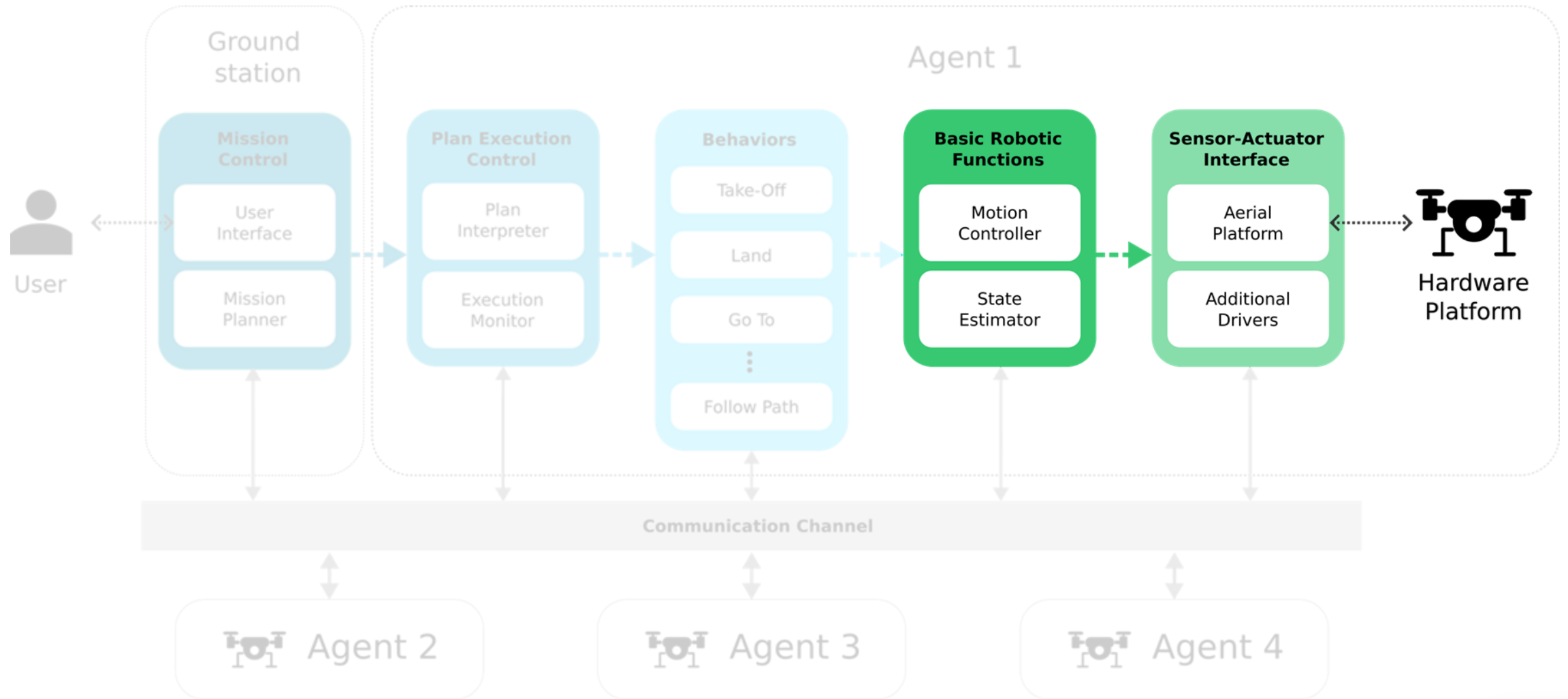
3. Lanzar un nodo plataforma por dron (namespace = model_name)

```
ros2 launch as2_platform_gazebo  
platform_gazebo_launch.py  
namespace:=<%= drone_namespace %>  
platform_config_file:=config/config.yaml  
simulation_config_file:=<%= simulation_config_file %>
```

```
world_name: "empty"  
drones:  
- model_type: "crazyflie"  
  model_name: "drone0"  
  xyz:  
    - 0.0  
    - 0.0  
    - 0.3  
- model_type: "crazyflie"  
  model_name: "drone1"  
  xyz:  
    - -0.5  
    - 0.0  
    - 0.3  
- model_type: "crazyflie"  
  model_name: "drone2"  
  xyz:  
    - -1.0  
    - 0.0  
    - 0.3
```



Arquitectura

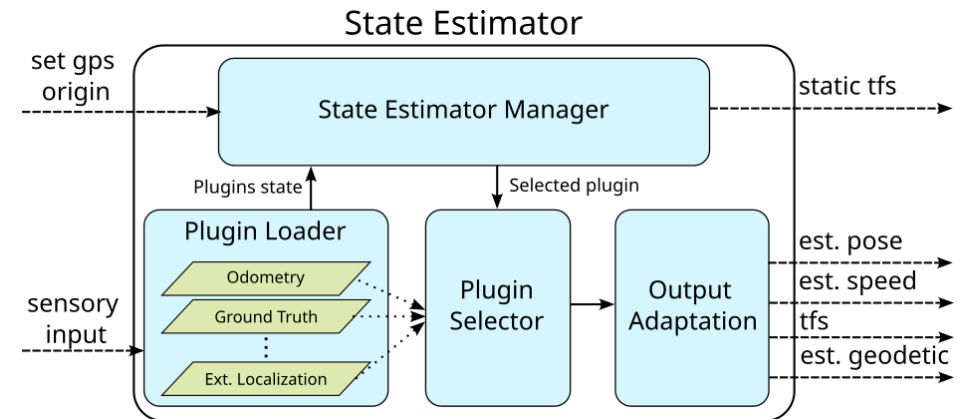




State Estimator

Encargado de proporcionar el estado del dron.

- Cargar y seleccionar plugins.
- Generar el árbol de TFs.
- Conversión entre geodésicas y cartesianas.
- Adaptar las entradas y salidas.



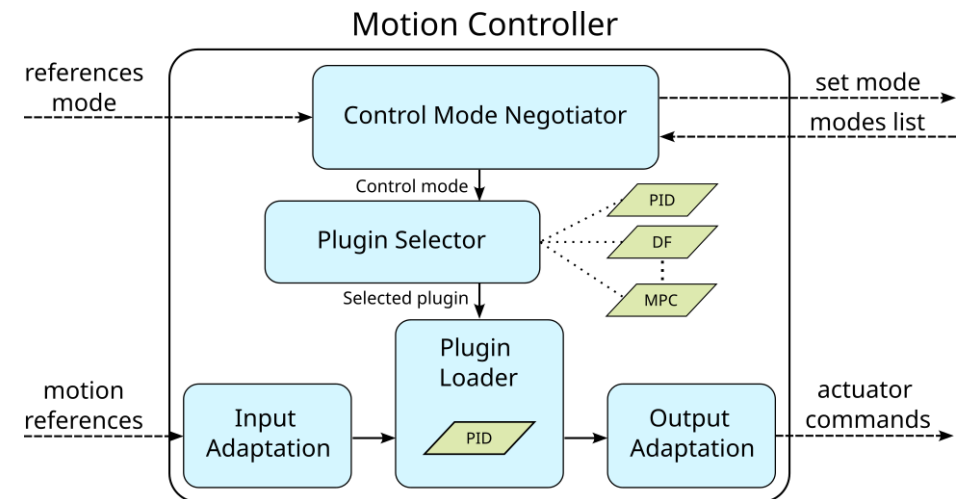
```
ros2 launch as2_state_estimator state_estimator_launch.py
namespace:=<%= drone_namespace %>
config_file:=config/config.yaml
```



State Estimator

Encargado del movimiento del dron.

- Cargar y seleccionar plugins.
- Seleccionar el modo de control.
- Adaptar las entradas y salidas.



```
ros2 launch as2_motion_controller controller_launch.py
  namespace:=<%= drone_namespace %>
  config_file:=config/config.yaml
  plugin_name:=pid_speed_controller
  plugin_config_file:=config/pid_speed_controller.yaml
```



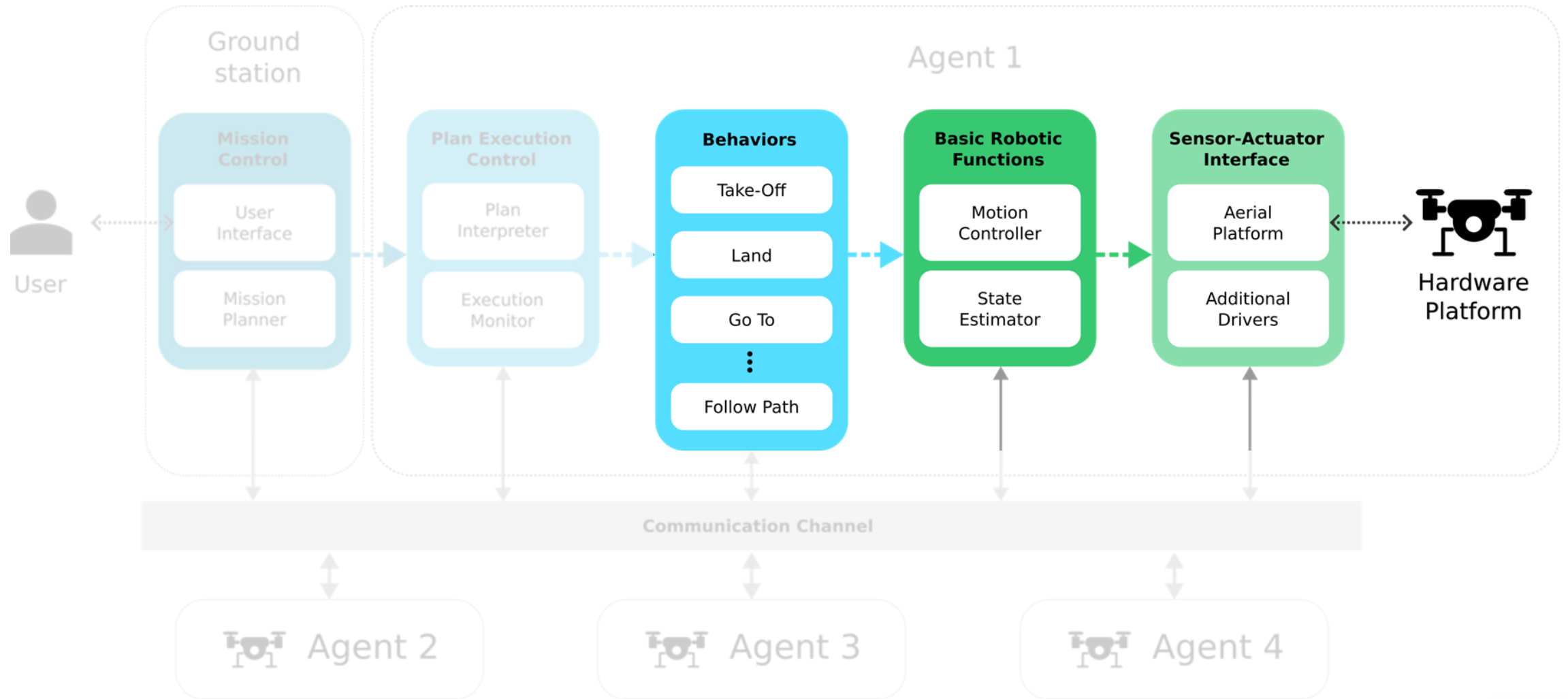
Ejemplos prácticos



1. Control por referencias al controlador.
2. Control de ejecución de plan y estructura de un proyecto.
3. Paso a real, uso de Crazyflies.
4. Misión multi-agente compleja: Drone-convoy.



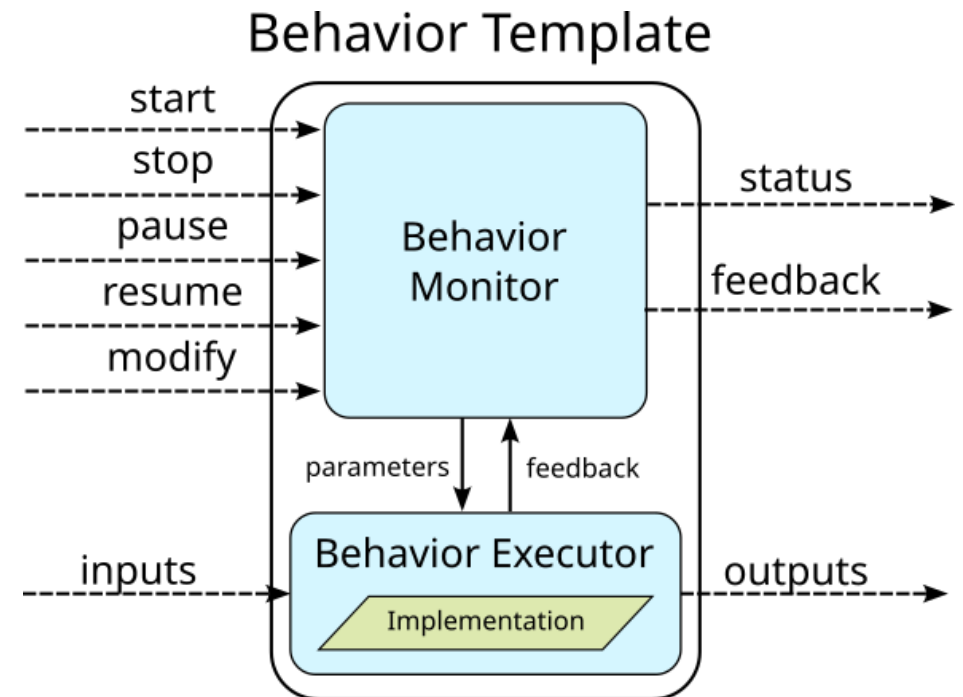
Arquitectura





Behaviors

- Proporcionar una capa lógica para formular planes de misión de manera uniforme y más simplificada.
- Cada comportamiento corresponde a una habilidad específica del robot relacionada.
- Cada comportamiento encapsula el control y la supervisión de la ejecución de su tarea.
- Se implementan ampliando las acciones de ROS 2 para proporcionar capacidades adicionales.





Behaviors



Motion Behaviors

- `ros2 launch as2_behaviors_motion motion_behaviors_launch.py`
 `namespace:=<%= drone_namespace %>`
 `config_file:=config/config.yaml`

Trajectory Generation Behavior

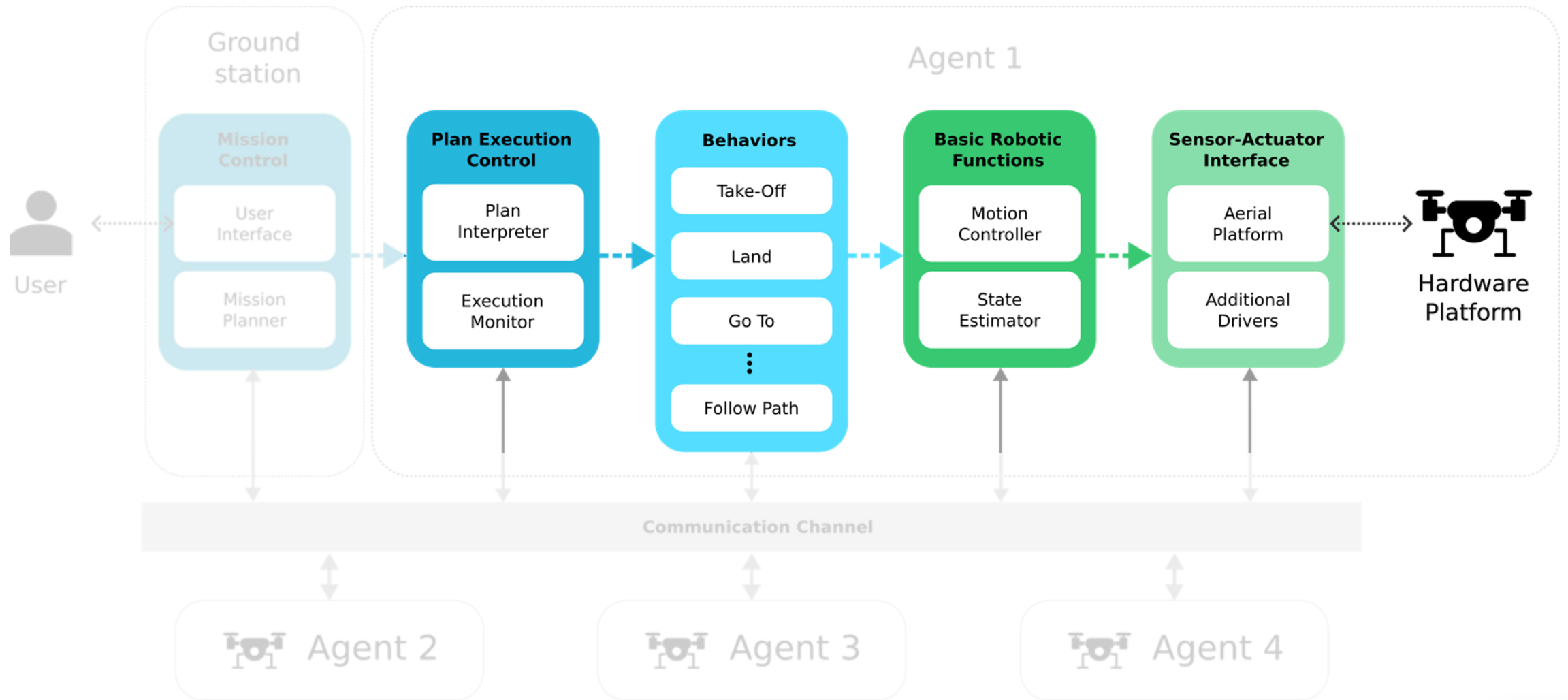
- `ros2 launch as2_behaviors_trajectory_generation generate_polynomial_trajectory_behavior_launch.py`
 `namespace:=<%= drone_namespace %>`
 `config_file:=config/config.yaml`

Point Gimbal Behavior

- `ros2 launch as2_behaviors_perception point_gimbal_behavior.launch.py`
 `namespace:=<%= drone_namespace %>`
 `config_file:=config/config.yaml`



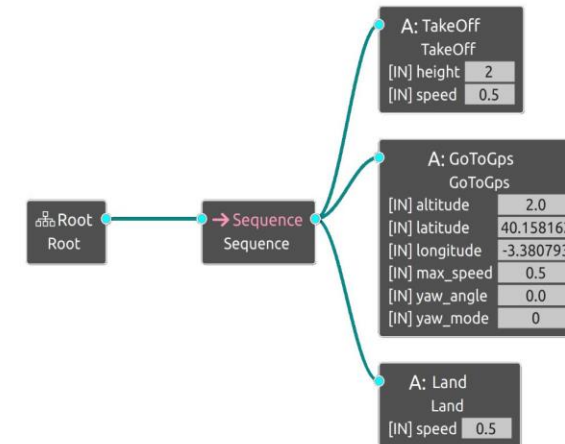
Arquitectura





Plan Execution Control

Se encarga de la especificación del plan de misión y la supervisión de su ejecución.



Behavior Trees

```
"target": "drone0",
"plan": [
{
  "behavior": "takeoff",
  "args": {"height": 1.0, "speed": 0.5}},
{
  "behavior": "go_to",
  "method": "go_to_point",
  "args": {"point": [1.0, 1.0, 2.0], "speed": 0.5, "frame_id": "earth"}},
{
  "behavior": "land",
  "args": {"speed": 0.5 }}
]
```

AS2 Mission Interpreter

```
import rclpy
from as2_python_api.drone_interface import DroneInterface

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

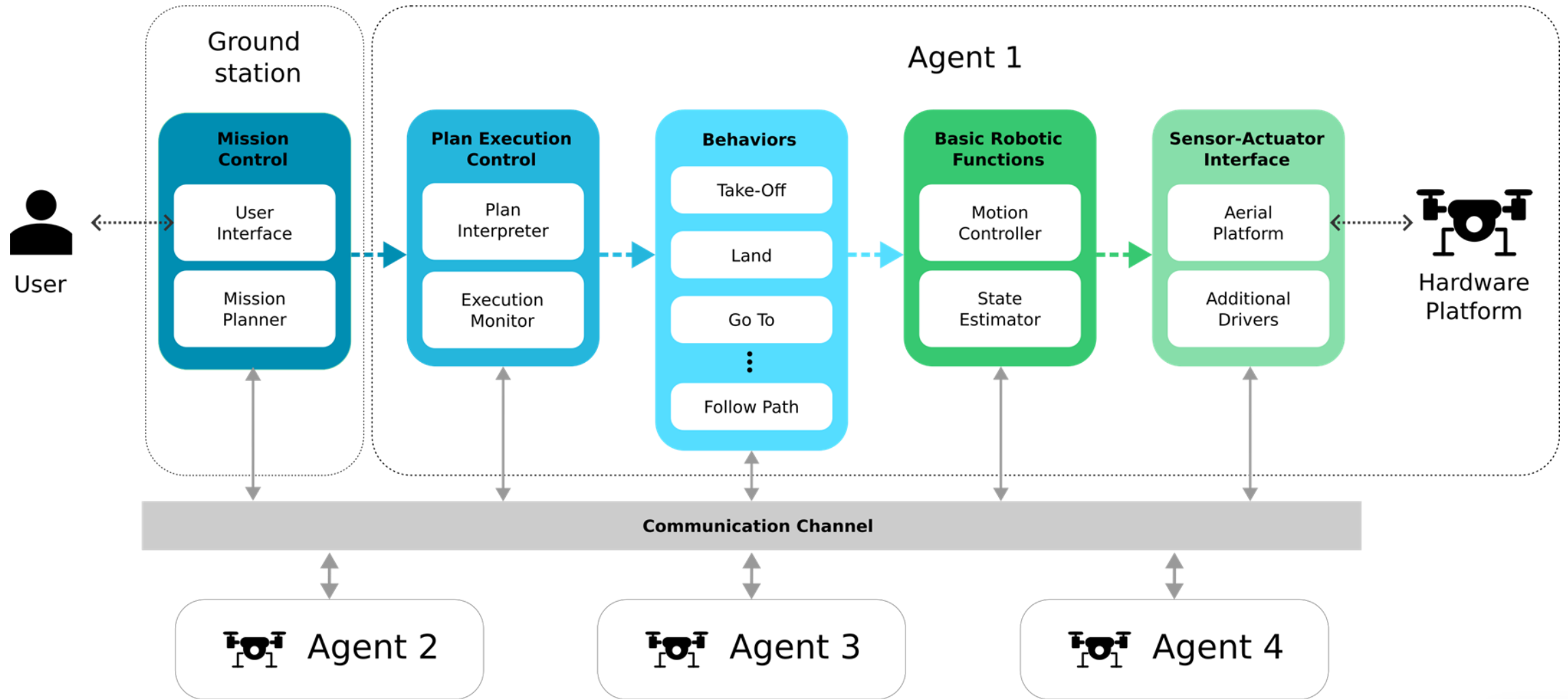
drone.takeoff(height=1.0, speed=0.5)
drone.go_to.go_to_point([1.0, 1.0, 2.0], speed=0.5, frame_id="earth")
drone.land(speed=0.5)

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

AS2 Python API



Arquitectura



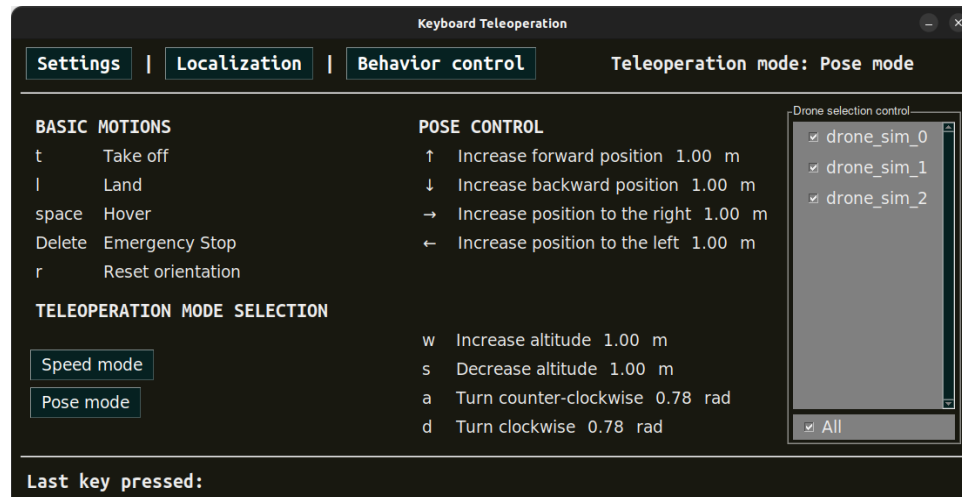


Mission Control

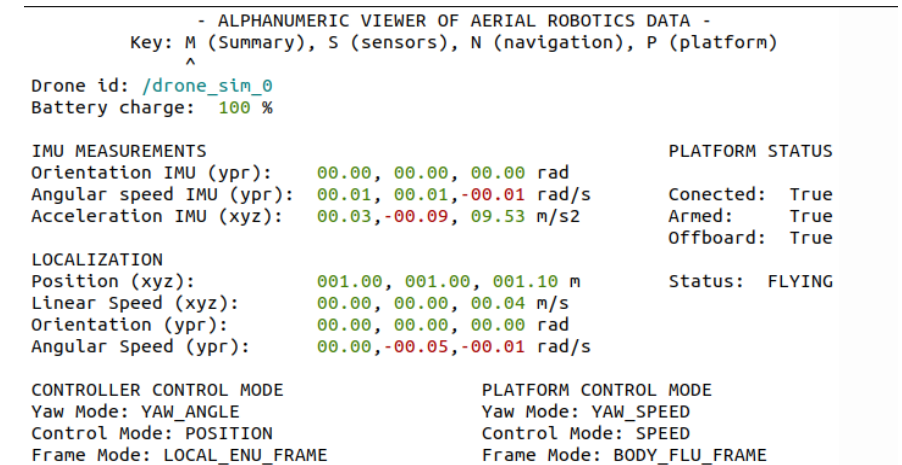
Destinado a facilitar la definición y supervisión de las misiones



AS2 GUI



AS2 Keyboard Teleoperation



AS2 Alphanumeric Viewer



Ejemplos prácticos



1. Control por referencias al controlador.
2. Control de ejecución de plan y estructura de un proyecto.
3. Paso a real, uso de Crazyflies.
4. Misión multi-agente compleja: Drone-convoy.



Ejemplos prácticos



1. Control por referencias al controlador.
2. Control de ejecución de plan y estructura de un proyecto.
3. Paso a real, uso de Crazyflies.
4. Misión multi-agente compleja: Drone-convoy.



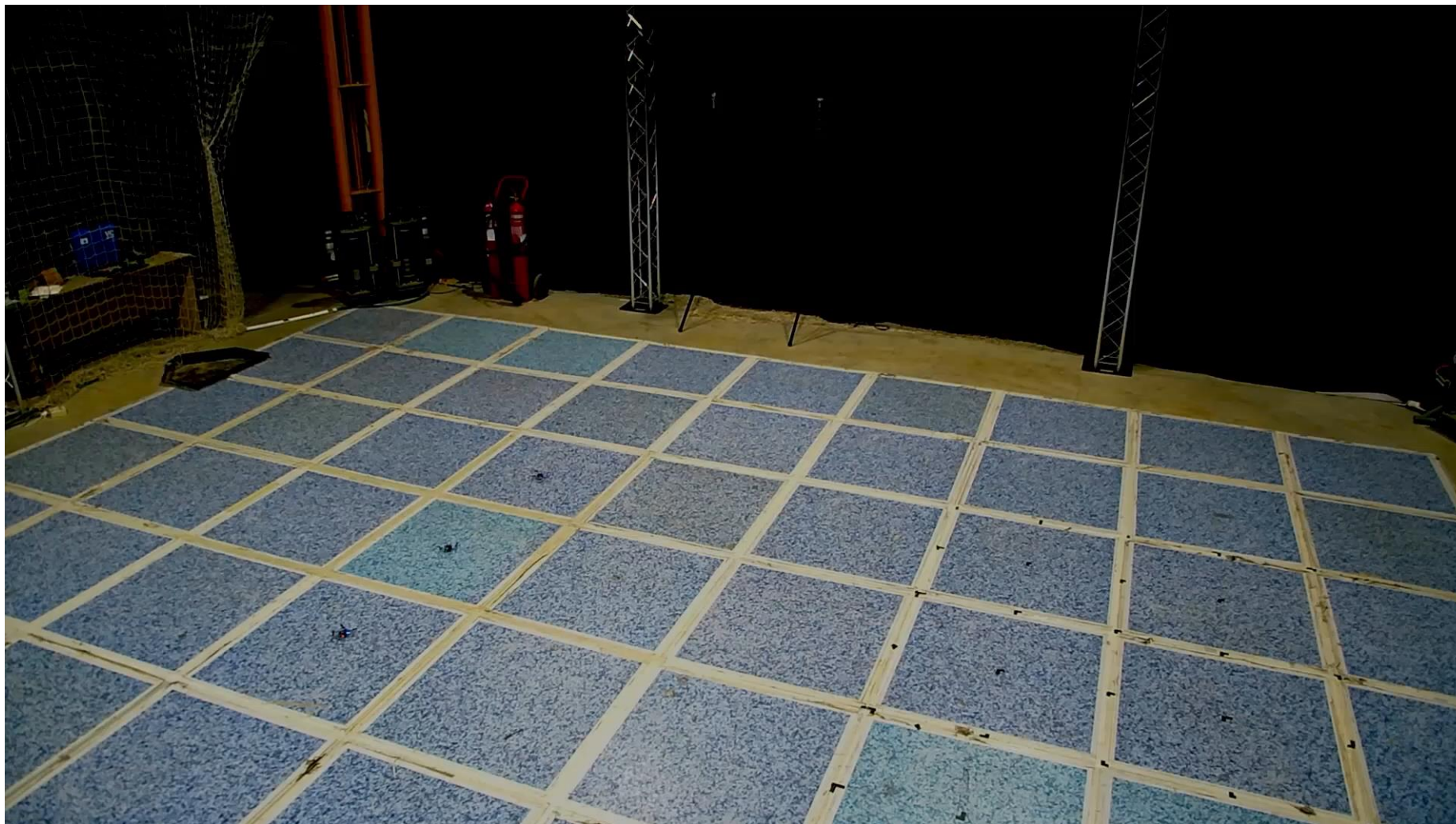
Ejemplos prácticos



1. Control por referencias al controlador.
2. Control de ejecución de plan y estructura de un proyecto.
3. Paso a real, uso de Crazyflies.
4. Misión multi-agente compleja: Drone-convoy.

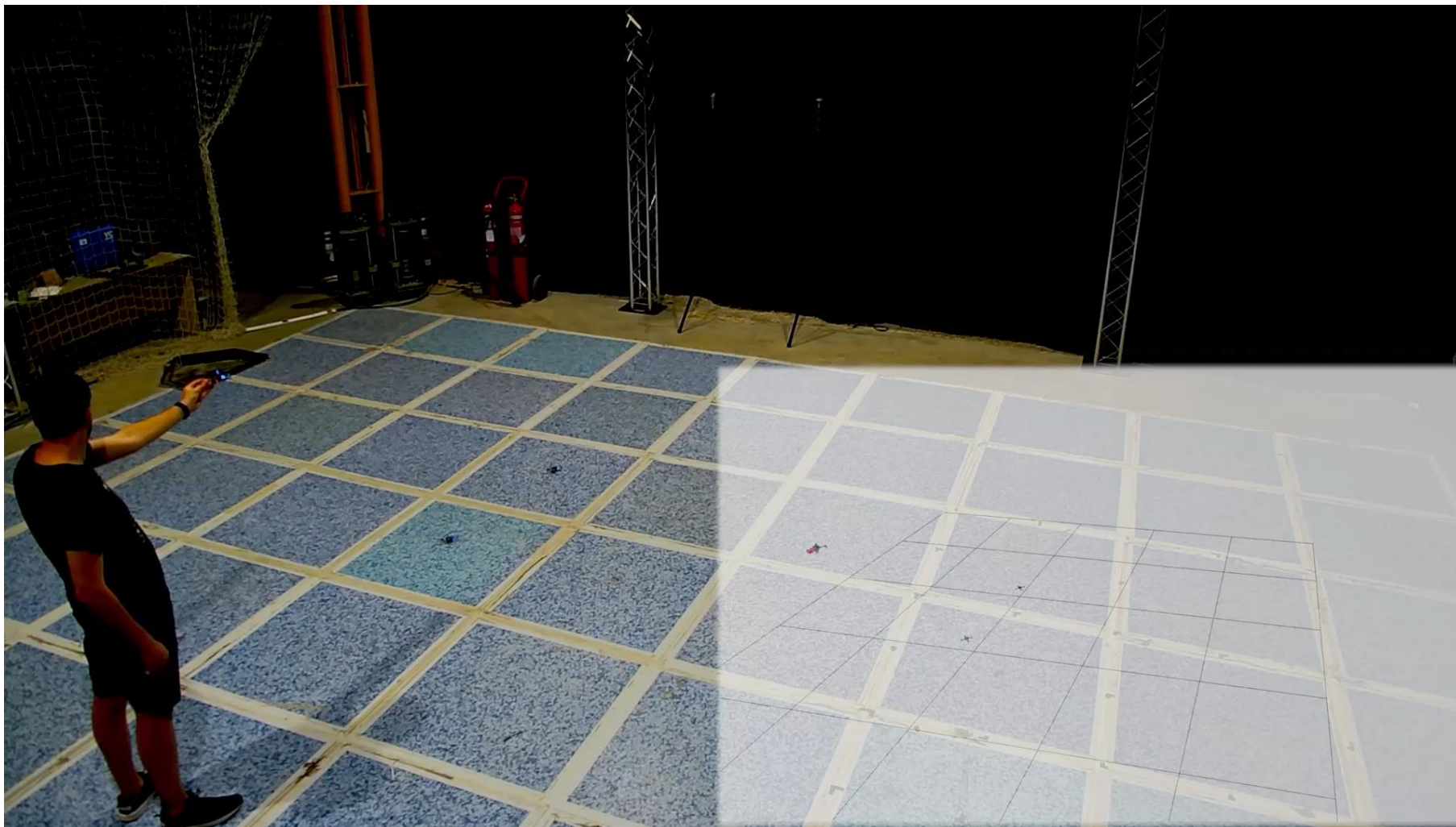


Experimento Real





Experimento Real





Casos de Uso





Casos de Uso



Exploring Unstructured Environments using Minimal Sensing on Cooperative Nano-Drones

Pedro Arias-Perez¹, Alvi Ka Gautam², Miguel Fernandez-Cortizas¹,
David Perez-Saura¹, Srikanth Saripalli² and Pascual Campoy²

¹CYATH - Universidad Politécnica de Madrid, ²USL - Texas A&M University





Casos de Uso



Universidad Politécnica de Madrid
Escuela Técnica Superior de Ingenieros Industriales

Photovoltaic plant inspection with area
coverage path planning using Aerostack2





Casos de Uso





Agradecimientos



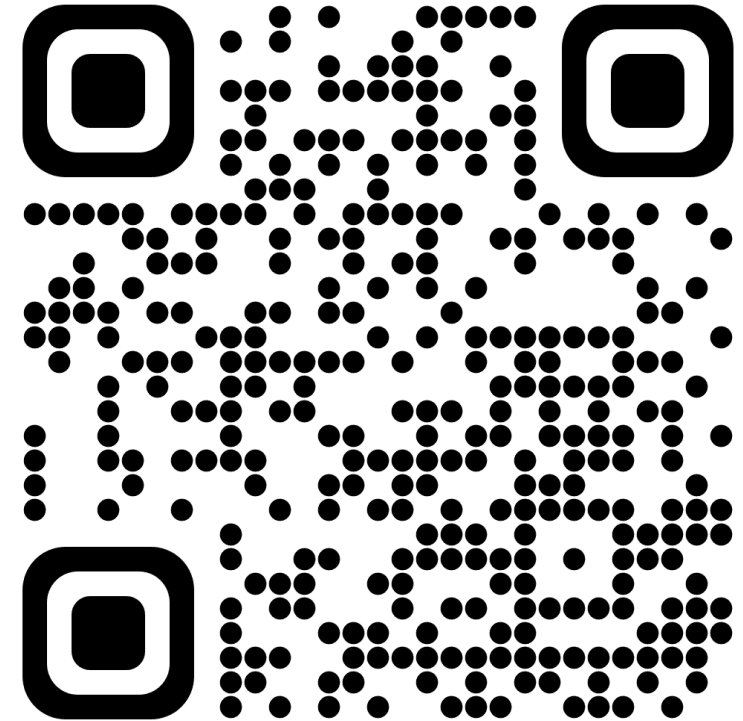
Principales autores:

- Miguel Fernandez-Cortizas
- **Pedro Arias-Perez**
- Rafael Perez-Segui
- Javier Melero-Deza
- David Perez-Saura
- **Guillermo García Patiño Lenza**
- Rodrigo Da Silva Gómez
- Martin Molina
- Pascual Campoy
- Carmen De Rojas Pita Romero
- Francisco José Anguita Chamorro





Please consider to star and contribute
to our project on GitHub!



Gracias por vuestra atención