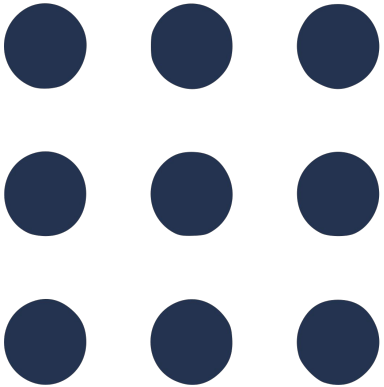


Multi-vehicle simulation with ROS 2 and Gazebo

Alejandro Hernández
Cordero

June 2023

Index

- **vehicle_gateway**
- **Simulation**
- **Testing**
- **Multirobot**
- **Demo world**
- **zenoh**
- **Conclusions**

Index

- **vehicle_gateway**
- Simulation
- Testing
- Multirobot
- Demo world
- zenoh
- Conclusions

vehicle_gateway

- The goal of this project is to create a pluginlib-based C++ library that can interface with several vehicle SDK's.


ONE API
TO RULE THEM ALL
imgflip.com

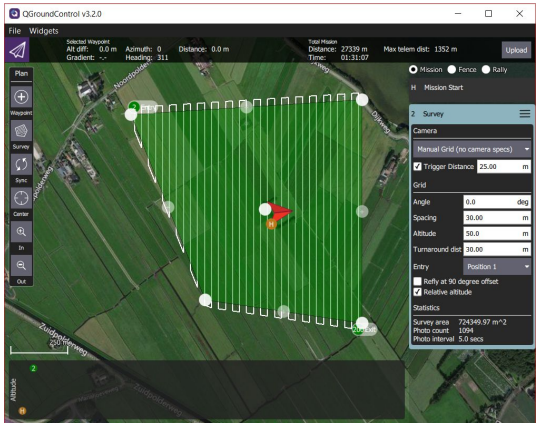
vehicle_gateway

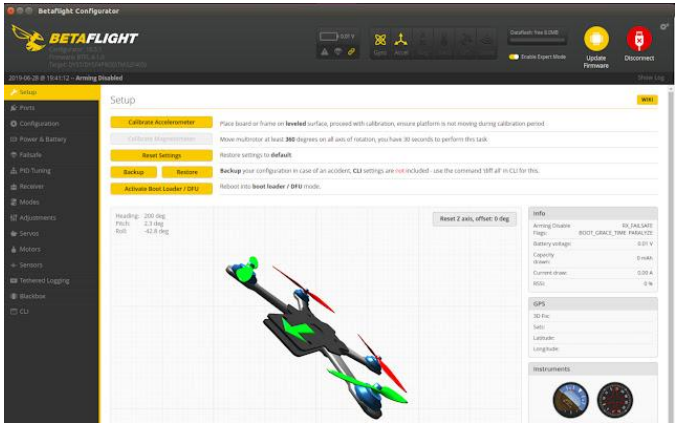
- The goal of this project is to create a pluginlib-based C++ library that can interface with several vehicle SDK's.
- Download and install the required target to run Software In The Loop (SITL)
  - Betaflight (Experimental)
  - PX4

```
$> px4
$> px4-commander
$> betaflight_SITL.elf
```

vehicle_gateway

- The goal of this project is to create a pluginlib-based C++ library that can interface with several vehicle SDK's.
- Download and install the required target to run SITL
  - Betaflight (Experimental)
  - PX4
- Download and install ground stations
  - QGroundControl
  - Betaflight configurator





vehicle_gateway

- The goal of this project is to create a pluginlib-based C++ library that can interface with several vehicle SDK's.
- Download and install the required target to run SITL
  - Betaflight (Experimental)
  - PX4
- Download and install ground stations
  - QGroundControl
  - Betaflight configurator
- vehicle_gateway: A pluginlib-based system for interfacing to vehicle SDK's.
  - vehicle_gateway_betaflight: Betaflight plugin for the Vehicle Gateway.
  - vehicle_gateway_px4: PX4 plugin for the Vehicle Gateway.

vehicle_gateway

- vehicle_gateway is pure CPP without dependencies
  - ROS 2 Humble
- vehicle_gateway_betaflight
  - pluginlib
  - rclcpp
  - sensor_msgs
  - std_msgs
- vehicle_gateway_px4
  - pluginlib
  - rclcpp
  - px4_msgs
  - tf2
  - zenohc

vehicle_gateway

```
pluginlib::ClassLoader<vehicle_gateway::VehicleGateway> loader;
std::shared_ptr<vehicle_gateway::VehicleGateway> gateway;

gateway_ = this->loader_.createSharedInstance(
    "vehicle_gateway_px4::VehicleGatewayPX4");
gateway_->init(0, nullptr);
gateway->arm()
gateway->takeoff()
// do stuff
gateway->land()
gateway->destroy()
```

vehicle_gateway

```
/// Arm vehicle
virtual void arm() = 0;

/// Arm vehicle (blocking method)
virtual void arm_sync() = 0;

/// Disarm vehicle
virtual void disarm() = 0;

/// Disarm vehicle (blocking method)
virtual void disarm_sync() = 0;
```

vehicle_gateway

```
/// Get flight mode
/// \return Flight mode
virtual FLIGHT_MODE get_flight_mode() = 0;

/// Get Vehicle type
/// \return Vehicle type
virtual VEHICLE_TYPE get_vehicle_type() = 0;

/// Get VTOL state
/// \return VTOL state
virtual VTOL_STATE get_vtol_state() = 0;

/// Takeoff the robot
virtual void takeoff() = 0;

/// Land the robot
virtual void land() = 0;
```

vehicle_gateway

```
/// VTOL
/// Transition to fixed wings
virtual void transition_to_fw() = 0;

/// Transition to fixed wings (blocking method)
virtual void transition_to_fw_sync() = 0;

/// Transition to multicopter
virtual void transition_to_mc() = 0;

/// Transition to multicopter (blocking method)
virtual void transition_to_mc_sync() = 0;
```

vehicle_gateway

```
/// Get ground speed
/// \return Get ground speed
virtual float get_ground_speed() = 0;

/// Get ground speed
/// \return Get ground speed
virtual float get_airspeed() = 0;

/// Get altitude
/// \return Get altitude
virtual float get_altitude() = 0;

/// Get 0: latitude, 1: longitude, and 2: altitude
virtual std::vector<double> get_latlon() = 0;

virtual void get_local_position(float & x, float & y, float & z) = 0;
```

vehicle_gateway

```
  virtual void go_to_latlon(double lat, double lon, float alt_amsl) = 0;

  virtual void go_to_latlon_sync(
    double lat, double lon, double alt,
    double latlon_threshold = 0.5, double alt_threshold = 0.5) = 0;

  virtual void set_local_position_setpoint(float x, float y, float z, float yaw) = 0;

  virtual void offboard_mode_go_to_local_setpoint_sync(
    double x, double y, double alt, double yaw, double airspeeed = 15.0,
    double distance_threshold = 10.0,
    vehicle_gateway::CONTROLLER_TYPE controller_type =
vehicle_gateway::CONTROLLER_TYPE::POSITION) =  0;
```

Python API

- Requirements
  - pybind11

Python API

- Requirements
  - pybind11
- Examples - vehicle_gateway_python
  - mc_to_fw_to_mc.py
  - mc_to_fw_to_offboard.py
  - position_control.py
  - test_takeoff_land.py
  - velocity_control.py
  - vtol_body_rates.py
  - vtol_position_control.py

Python API

- Requirements
  - pybind11

```
import vehicle_gateway

px4_gateway = vehicle_gateway.init(args=sys.argv, plugin_type='px4', vehicle_id=0)
px4_gateway.arm_sync()
px4_gateway.takeoff()
x, y, z = px4_gateway.get_local_position()
px4_gateway.land()
px4_gateway.disarm_sync()
px4_gateway.destroy()
```

vehicle_gateway

- The goal of this project is to create a pluginlib-based C++ library that can interface with several vehicle SDK's.
- Download and install the required target to run SITL
  - Betaflight (Experimental)
  - PX4
- Download and install ground stations
  - QGroundControl
  - Betaflight configurator
- vehicle_gateway: A pluginlib-based system for interfacing to vehicle SDK's.
  - vehicle_gateway_betaflight: Betaflight plugin for the Vehicle Gateway.
  - vehicle_gateway_px4: PX4 plugin for the Vehicle Gateway.
- Python wrappers
  - pybind11
- end-to-end testing
- Simulation performance test

Index

- vehicle_gateway
- **Simulation**
- Testing
- Multirobot
- Demo world
- zenoh
- Conclusions

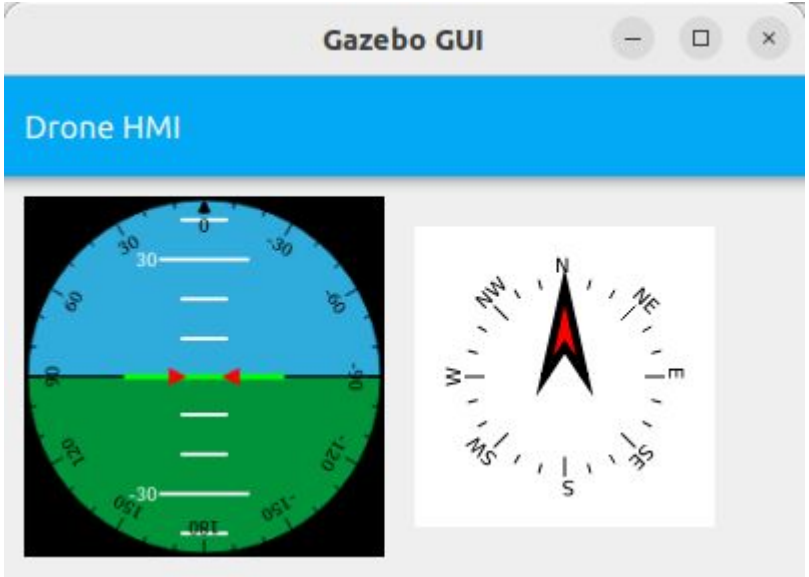
Simulation

Gazebo

- Physics
  - Accurate simulation with DART.
  - Plugin system for physics backend.
- Rendering
  - Ogre-Next 2.3 rendering engine (PBR).
  - Plugin system for rendering backend.
- Sensors
  - Rich camera system.
  - Extend with sensor plugins.
- Simulation Engine
  - Extend with system plugins.
- GUI
  - Custom GUI widgets as plugins.
- ROS / ROS 2 integration (ros_gz)

Aerial widgets in Gazebo

- Compass
- Attitude
- Generic plugins
- Improvements:
  - airspeed indicator?
  - takeoff/land/mode buttons?



Simulation

- Requirements
  - ROS 2 Humble
  - Gazebo Garden
  - ros_gz
- PX4
  - px4_sim
    - Launch single vehicle
    - Launch multi vehicle
- Betaflight
  - betaflight_sim
    - Launch single vehicle
    - Launch single vehicle with joystick
  - betaflight_gazebo (Gazebo plugin)
    - UDP connection with Betaflight - send RC commands

Launch files

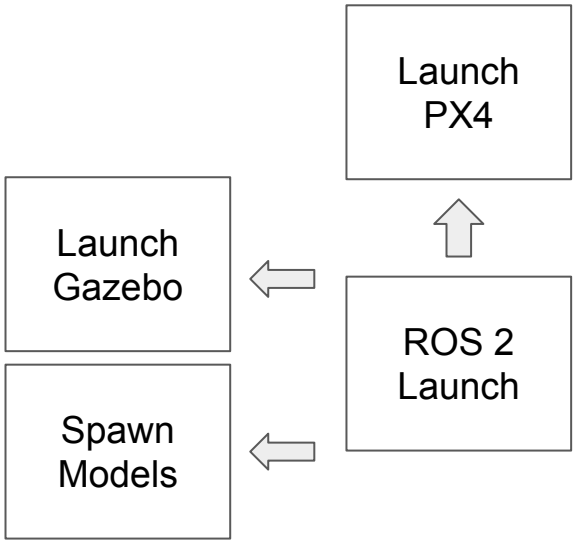
- **drone_id:** set the vehicle ID, default=0
- **drone_type:** Sim Models (x500, standard_vtol, rc_cessna)
- **dds_domain:** Set DDS_DOMAIN_ID
- **sensor_config:** Sensor configuration from configs_px4 directory
- **frame_name:** Frame name included in the SDF world file
- **groundcontrol:** Start ground control station.
- **world_name:** World name (without .sdf)
- **model_pose:** Model pose (x, y, z, roll, pitch, yaw)

```
ros2 launch px4_sim px4_sim.launch.py drone_type:='x500' world_name:=null_island
model_pose:="-9.7948, -8.31, 2, 0, 0, 0"
```

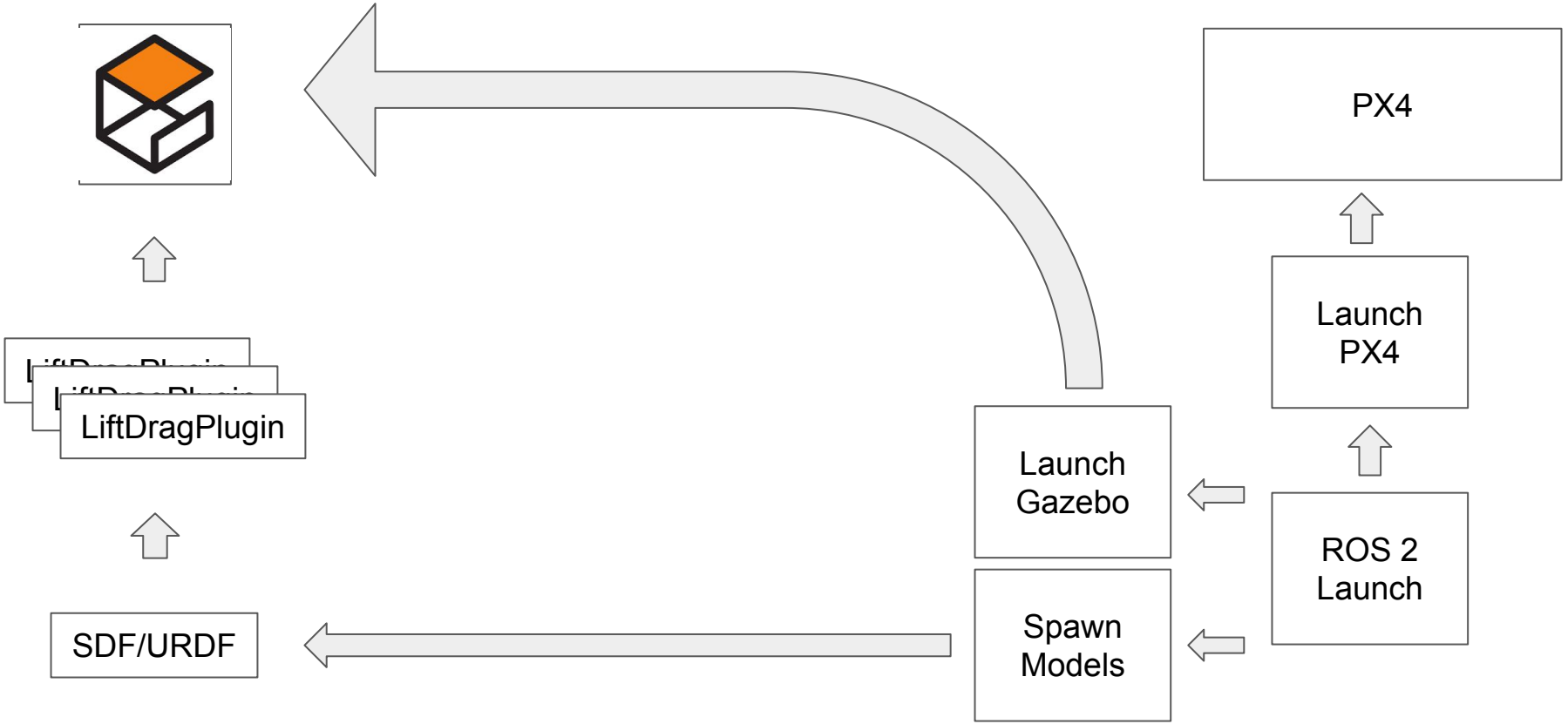
vehicle_gateway_models

- Launch files include sensor_config parameter
- Extend models
  - X500
    - stock
    - camera
  - Standard VTOL
    - stock
    - camera

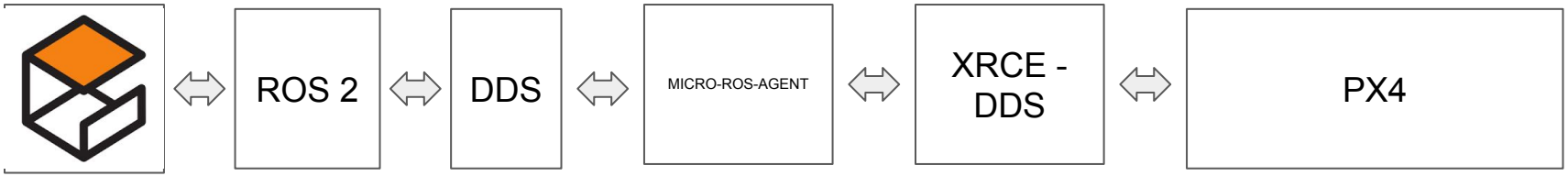
px4_sim



px4_sim

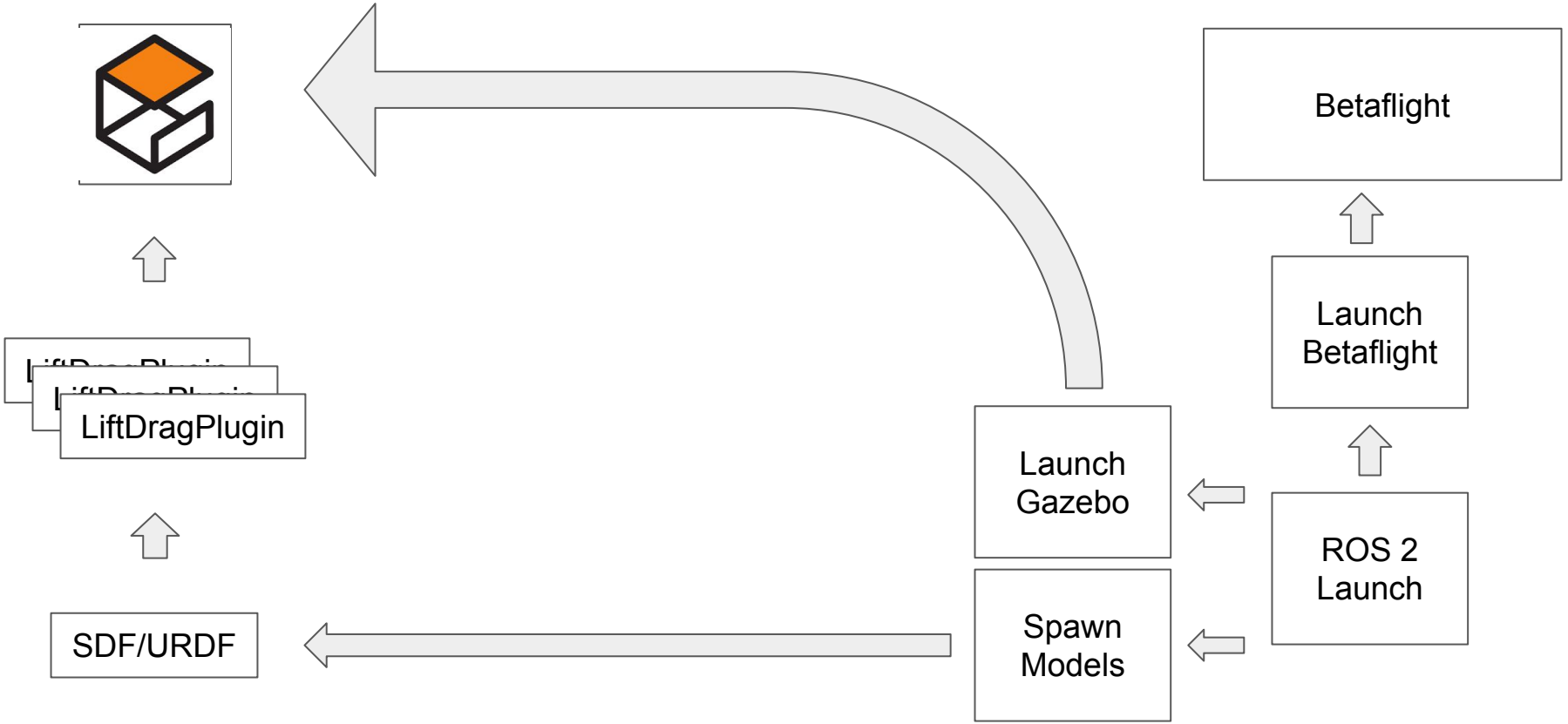

PX4
Launch PX4
ROS 2 Launch
Launch Gazebo
Spawn Models
SDF/URDF
LiftDragPlugin

px4_sim

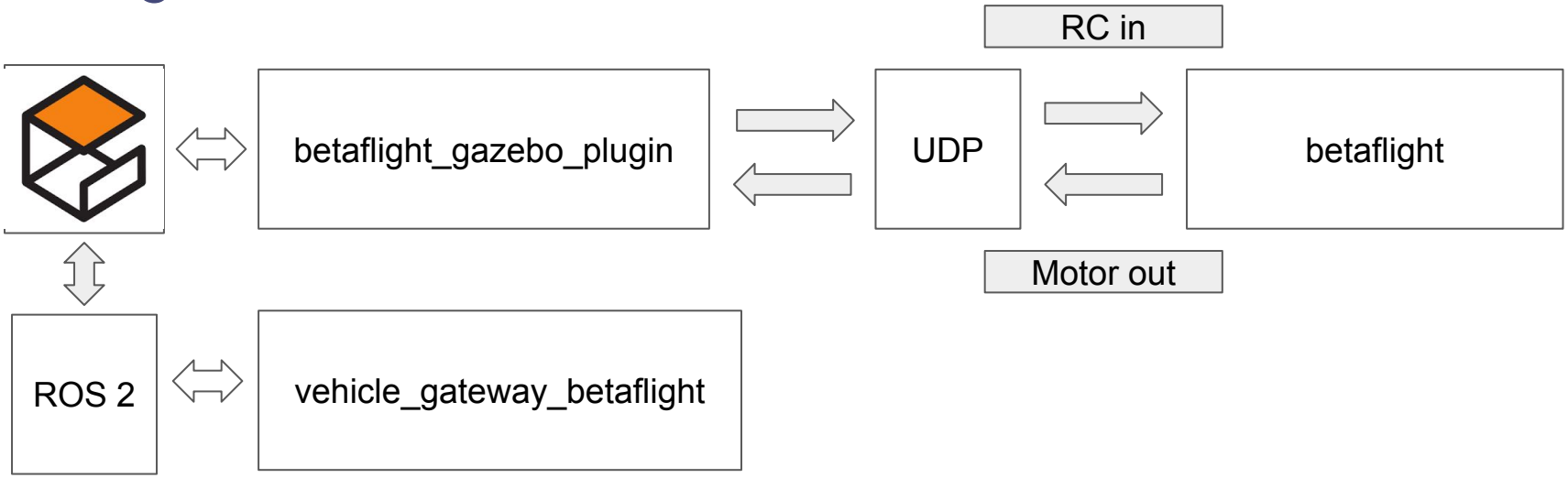


/fmu/in/onboard_computer_status
/fmu/in/sensor_optical_flow
/fmu/in/vehicle_command
…
/fmu/out/timesync_status
/fmu/out/vehicle_attitude
/fmu/out/vehicle_control_mode
/fmu/out/vehicle_gps_position
…

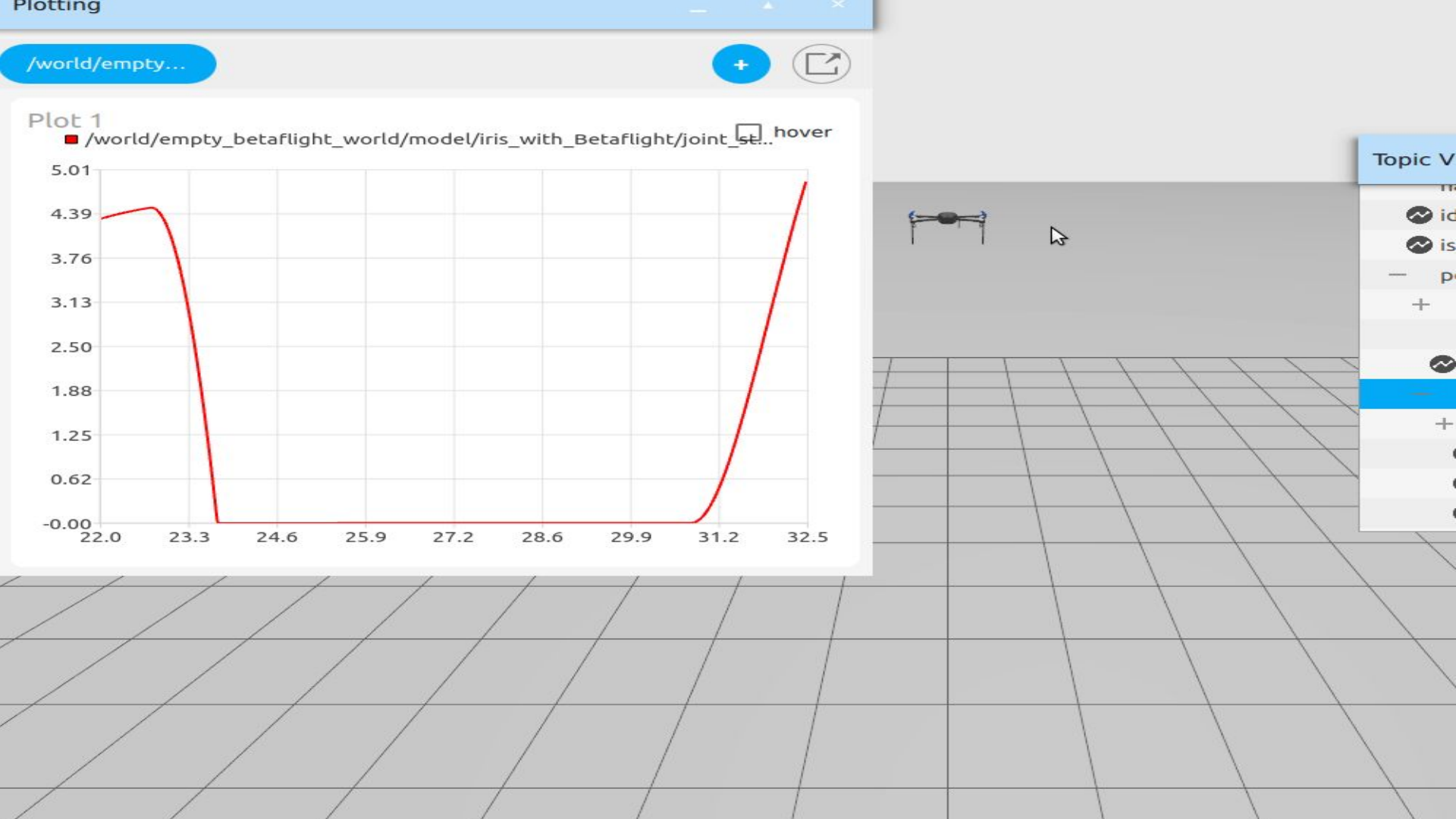
betaflight_sim



betaflight_sim

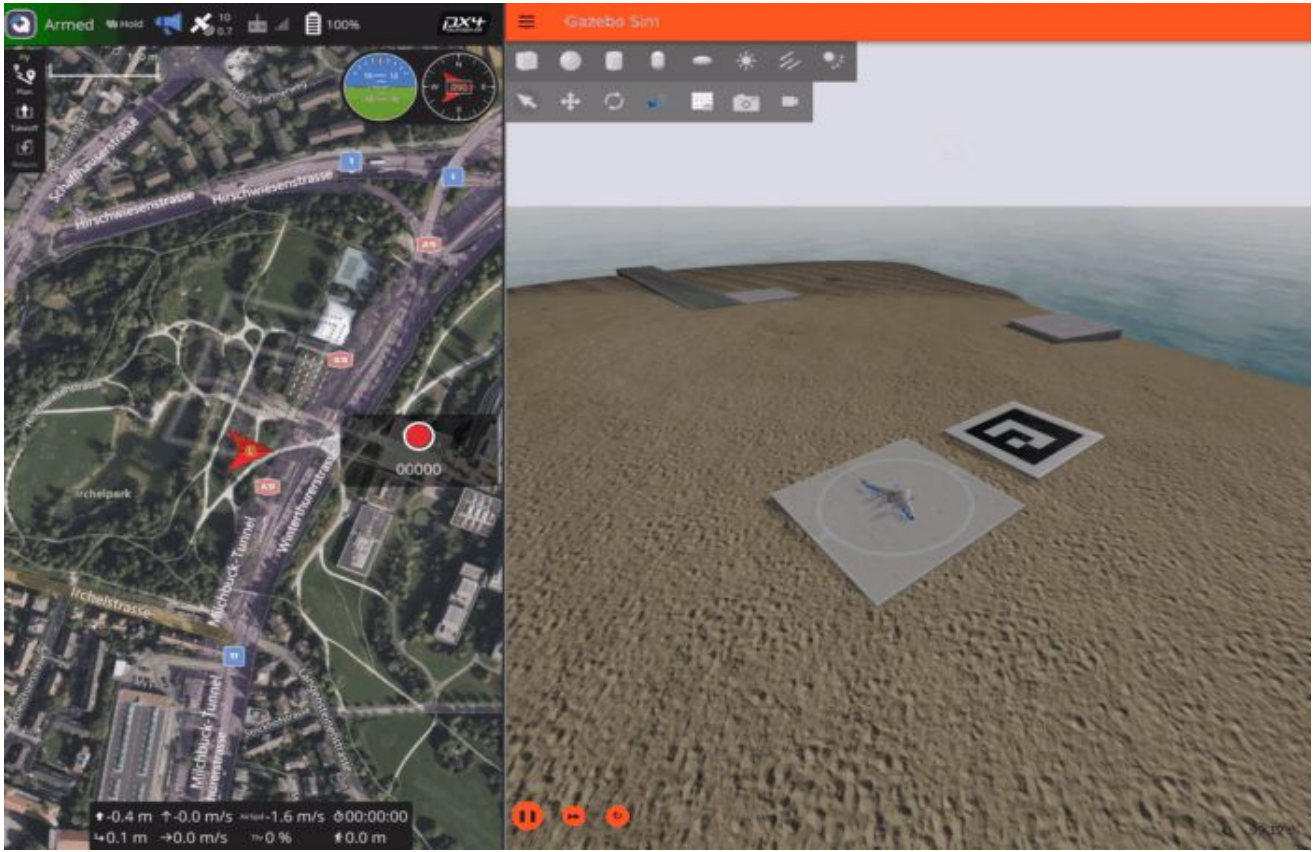


/global_position/rel_alt
/imu/data_raw
/imu/mag
/joy
/joy/set_feedback
/motors_out


Plotting
/world/empty...
Plot 1
/world/empty_betaflight_world/model/iris_with_Betaflight/joint_st...
hover
5.01
4.39
3.76
3.13
2.50
1.88
1.25
0.62
-0.00
22.0
23.3
24.6
25.9
27.2
28.6
29.9
31.2
32.5
Topic V

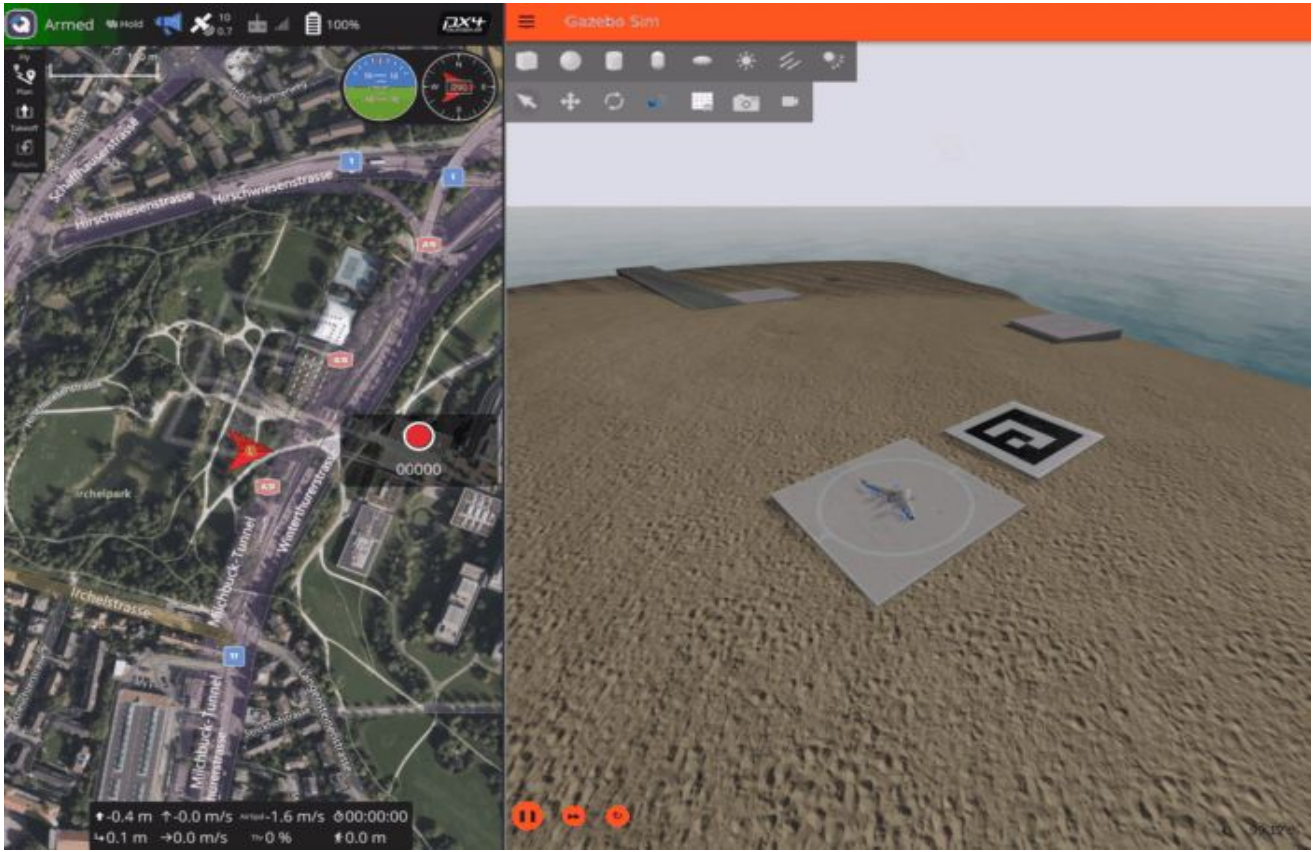
Offboard position control

- MC takeoff
- MC hover over pad
- FW transition
- FW loiter
- FW offboard **position**
  - fly to north waypoint
  - change speed
  - fly to south waypoint
  - fly thome
- FW loiter
- MC transition
- MC hover over pad
- MC land



Offboard body-rate and thrust control

- MC takeoff
- MC hover over pad
- FW transition
- FW loiter
- FW offboard **CTBR**
  - fly to north waypoint
  - change speed
  - fly to south waypoint
  - fly thome
- FW loiter
- MC transition
- MC hover over pad
- MC land



Index

- vehicle_gateway
- Simulation
- **Testing**
- Multirobot
- Demo world
- zenoh
- Conclusions

Testing - vehicle_gateway_integration_test

- End to end testing
- Gazebo headless mode

```
if 'SHOW_GZ_GUI' in os.environ and os.environ['SHOW_GZ_GUI']:
    gz_gui_args = ''
else:
    gz_gui_args = '--headless-rendering -s'
gz_args = f'{gz_gui_args} -r -v 4 empty_px4_world.sdf'

included_launch = IncludeLaunchDescription(
    PythonLaunchDescriptionSource(
        [os.path.join(get_package_share_directory('ros_gz_sim'),
                      'launch', 'gz_sim.launch.py')]),
    launch_arguments=[('gz_args', [gz_args])]
)
```

Testing - vehicle_gateway_integration_test

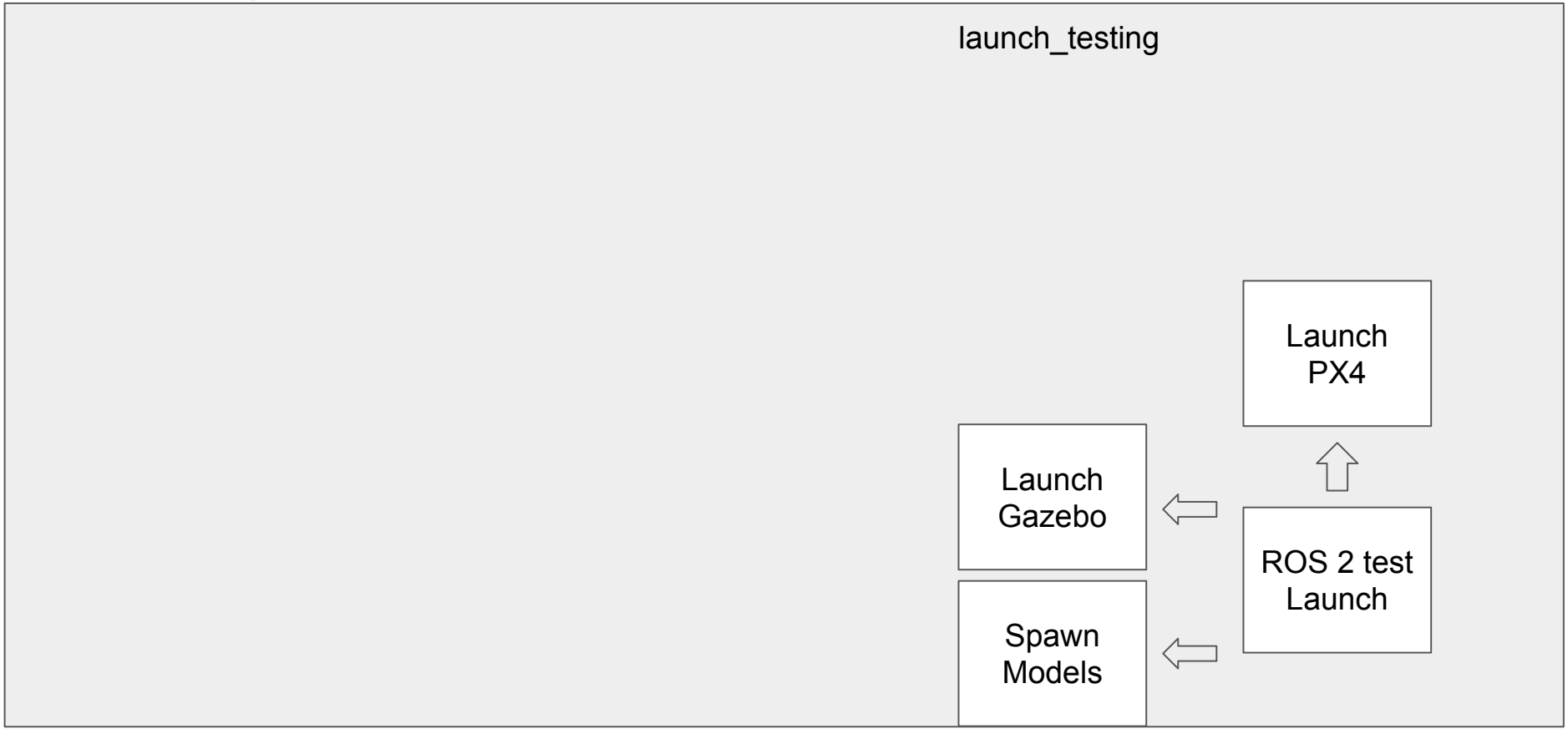
- End to end testing
- Gazebo headless mode
- Python script with the test
  - arm/disarm
  - takeoff/land
- Check outputs

```
proc_output.assertWaitFor('Ready for takeoff!', process=run_px4,
                                                timeout=100, stream='stdout')
```

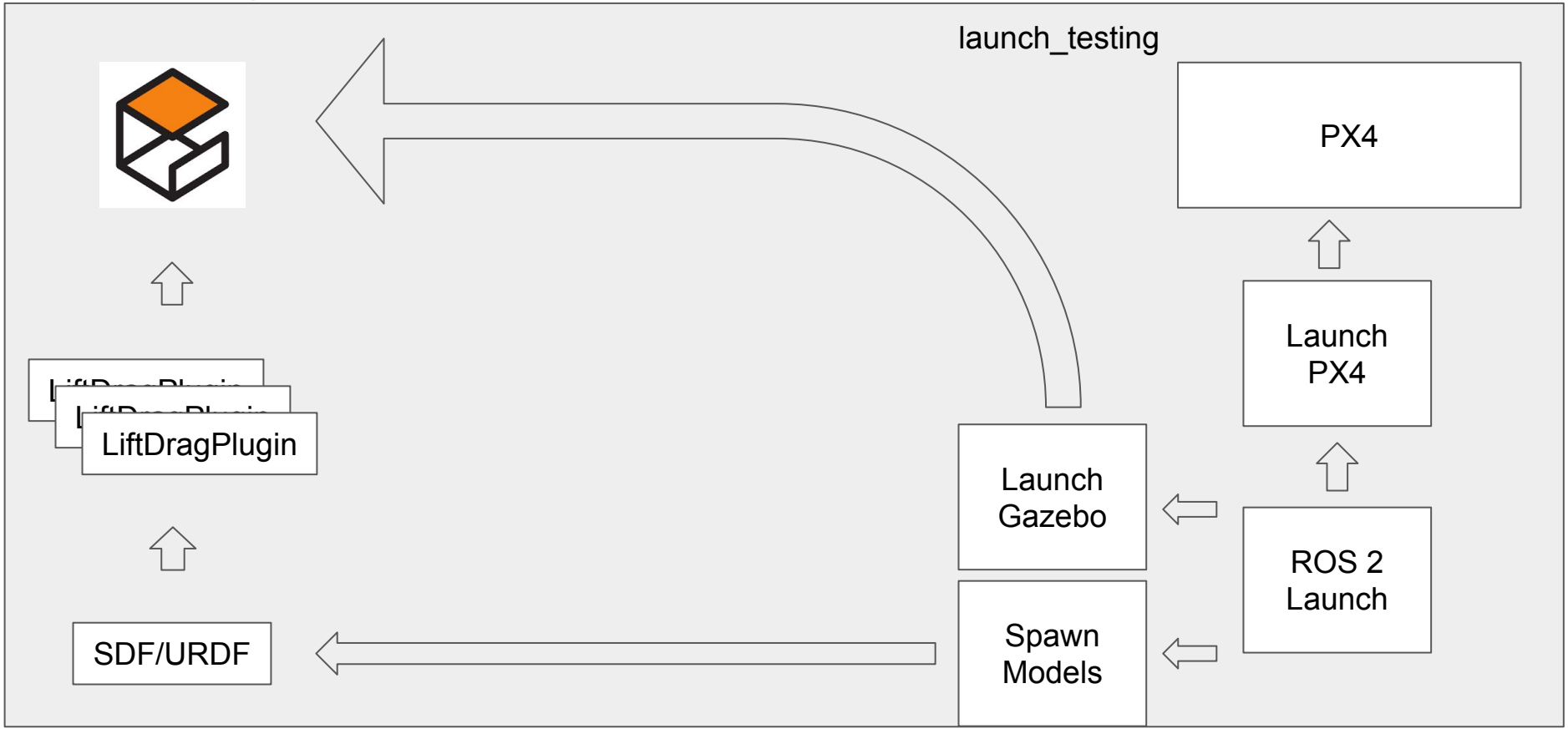
- Check exit codes

```
launch_testing.asserts.assertExitCodes(proc_info, process=run_px4,
                                                    allowable_exit_codes=[0])
```

Testing


launch_testing
Launch PX4
Launch Gazebo
ROS 2 test Launch
Spawn Models

Testing


launch_testing
PX4
Launch PX4
Launch Gazebo
ROS 2 Launch
Spawn Models
SDF/URDF
LiftDragPlugin

Testing

launch_testing

ROS 2 ⇔ DDS ⇔ MICRO-ROS-AGENT ⇔ XRCE - DDS ⇔ PX4

Python Script test

- Wait for outputs
- Wait for exist codes
- Kill all processes

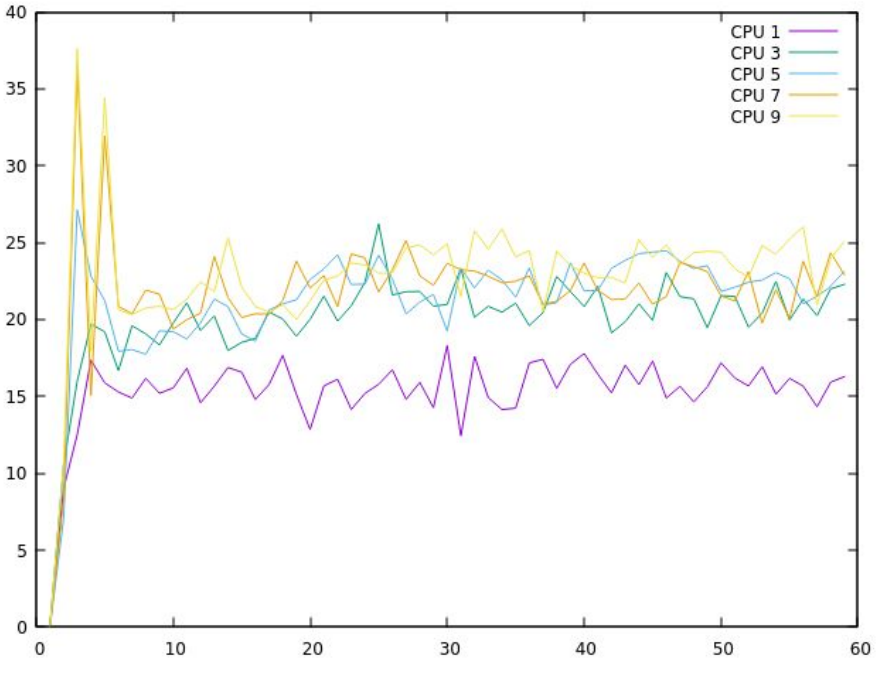
ROS 2 test Launch

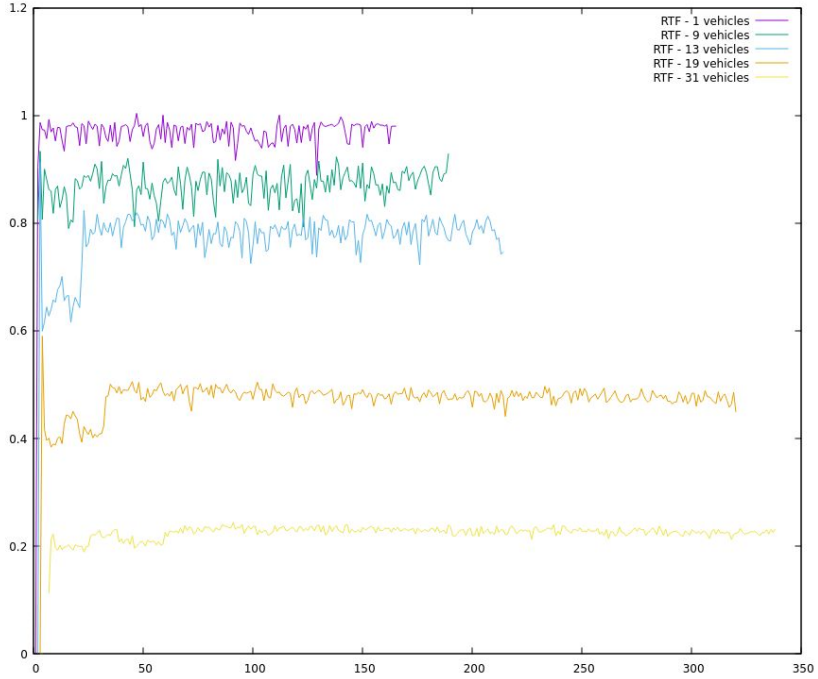
Simulation - performance tests

- Collect CPU and memory system usage
- Collect Gazebo real time factor (RTF)
- Run multidrone simulation
  - Same DDS domain
  - Different DDS domain
- Script to plot data

Simulation - performance tests

- CPU versus # vehicles and RTF vs # vehicles. Physics thread is 100%
- Note this is physics-bound; RTF decreases with more vehicles





Index

- vehicle_gateway
- Simulation
- Testing
- **Multirobot**
- Demo world
- zenoh
- Conclusions

multirobot

- PX4 only:
  - Quads
  - Fixed wing planes
  - VTOLs
- Spawn robot and generate namespaced topics/services
- YAML file to configure your simulation

```yaml
- vehicle_id: 1
  frame_name: pad_1
  vehicle_type: x500
  sensor_config: stock
  model_pose: ""
  dds_domain_id: 1
- vehicle_id: 2
  frame_name: pad_2
  vehicle_type: standard_vtol
  sensor_config: stock
  model_pose: ""
  dds_domain_id: 2
```

multirobot

- PX4 only:
  - Quads
  - Fixed wing planes
  - VTOLs
- Spawn robot and generate namespaced topics/services
- YAML file to configure your simulation
- Launch file

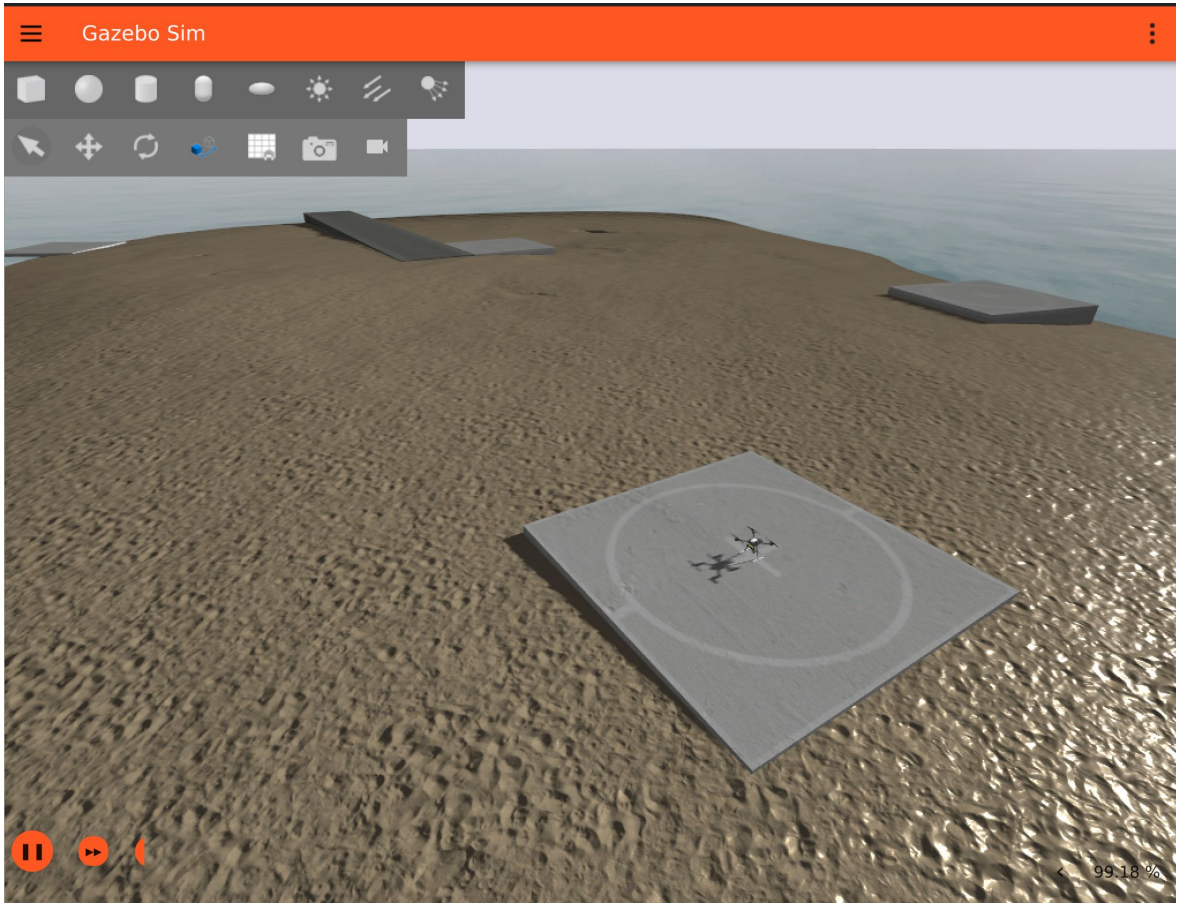
```
export MULTIROBOT_CONFIG=<path_to_config_file>/multirobot_config.yaml
ros2 launch px4_sim px4_sim_multi.launch.py
```

Index

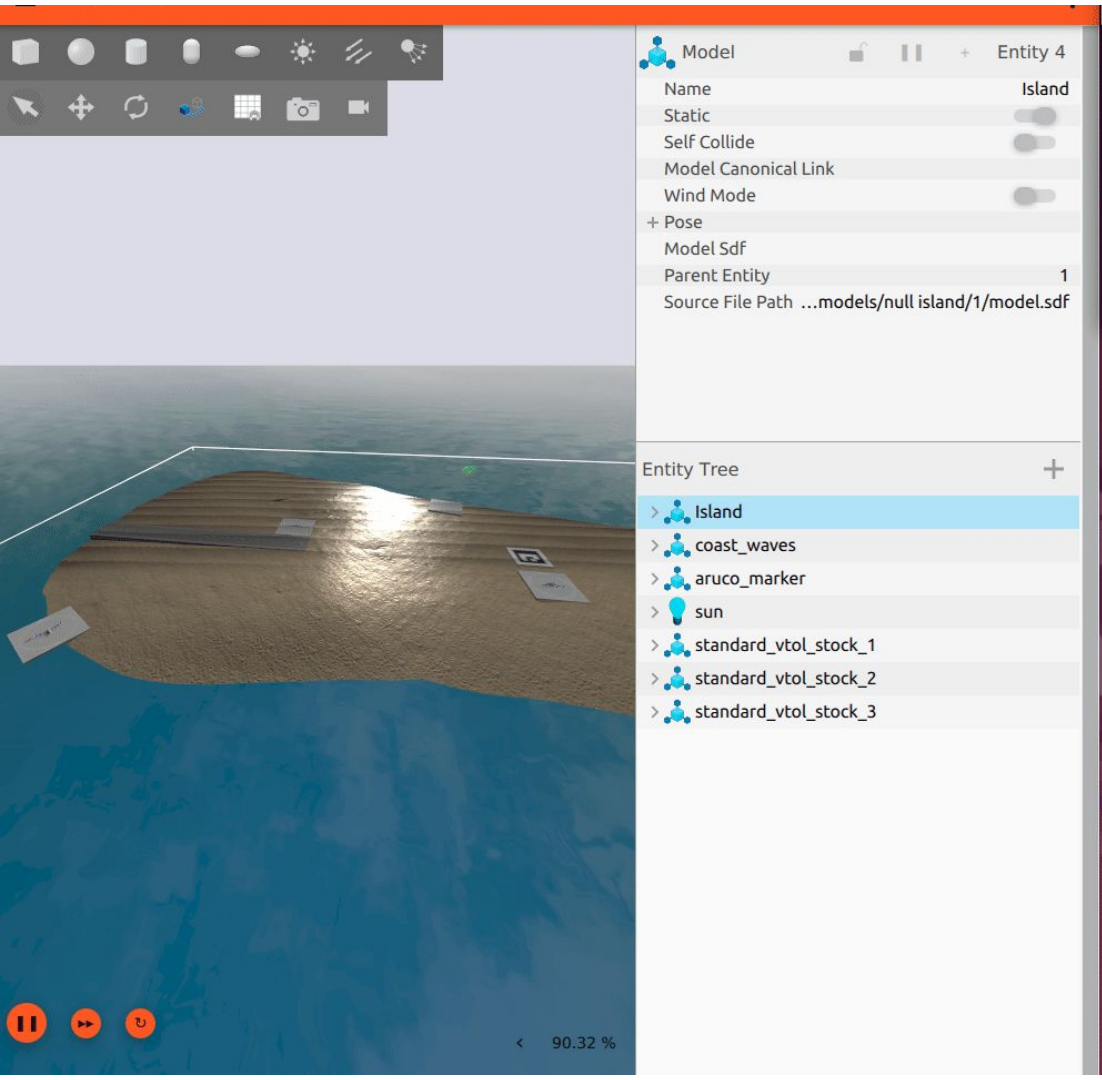
- vehicle_gateway
- Simulation
- Testing
- Multirobot
- **Demo world**
- zenoh
- Conclusions

World

- several helipads
  - future formation flight, etc
- one runway
- (0, 0) lat/lon
  - avoid confidential data
- sand material 🏝️
- wave-effect shader ✨
- What's next?
  - Boats ?
  - Rovers ?



ros2 launch px4_sim px4_sim.launch.py drone_type:=x500 world_name:=null_island frame_name:=pad_2




```
ahcorde@karsa:~/tii_ws$ python3 /home/ahcorde/tii_ws/src/vehicle_gateway/vehicle
_gateway_python/examples/test_takeoff_land.py 1
Arming...
Takeoff!
Current position: (x: -0.55, y: 0.39, z: -4.97)
Landing...
Disarming...
calling std::thread::join()
done with std::thread::join()
Takeoff and land demo complete.
ahcorde@karsa:~/tii_ws$ python3 /home/ahcorde/tii_ws/src/vehicle_gateway/vehicle
_gateway_python/examples/test_takeoff_land.py 1
```

ahcorde@karsa: ~/tii_ws

```
ahcorde@karsa:~/tii_ws$ python3 /home/ahcorde/tii_ws/src/vehicle_gateway/vehicle
_gateway_python/examples/test_takeoff_land.py 2
Arming...
Takeoff!
Current position: (x: -1.33, y: 0.17, z: -5.13)
Landing...
Disarming...
calling std::thread::join()
done with std::thread::join()
Takeoff and land demo complete.
ahcorde@karsa:~/tii_ws$
```

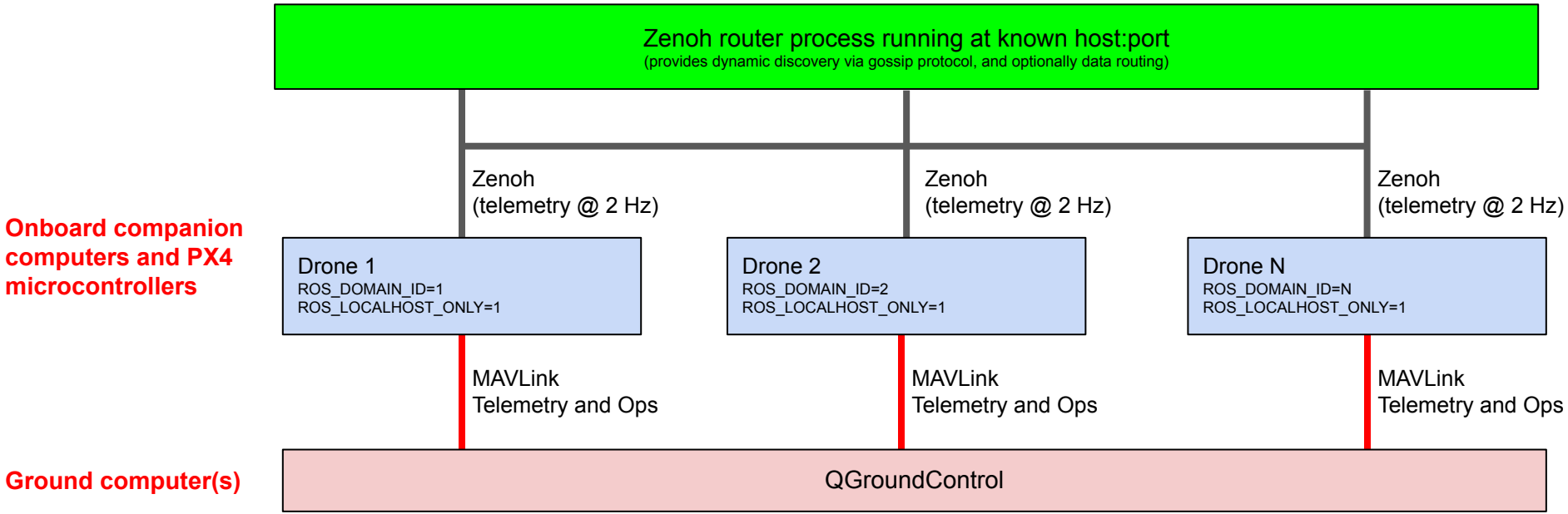
ahcorde@karsa: ~/tii_ws

```
ahcorde@karsa:~/tii_ws$ source install/setup.bash
ahcorde@karsa:~/tii_ws$ python3 /home/ahcorde/tii_ws/src/vehicle_gateway/vehicle
_gateway_python/examples/test_takeoff_land.py 3
Arming...
Takeoff!
Current position: (x: -0.14, y: -0.13, z: -4.43)
Landing...
Disarming...
calling std::thread::join()
done with std::thread::join()
Takeoff and land demo complete.
ahcorde@karsa:~/tii_ws$
```

Index

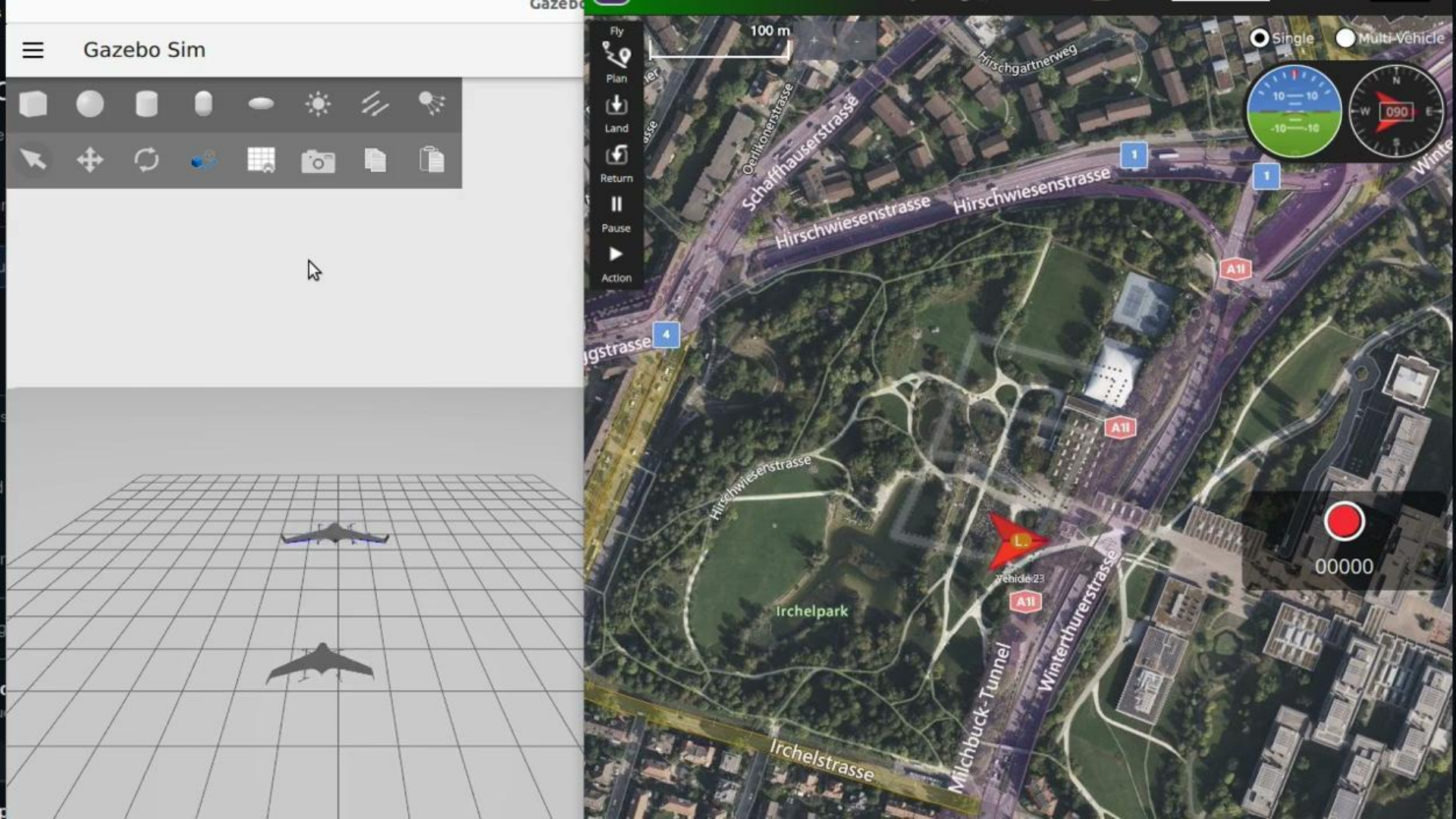
- vehicle_gateway
- Simulation
- Testing
- Multirobot
- Demo world
- **zenoh**
- Conclusions

Using Zenoh as a scalable backplane



Zenoh demo

- Sending 2Hz
  - Data (encoded as JSON for ease of future expansion)
    - Vehicle id
    - position
    - airspeed
    - heading
  - Zenoh key: /vehicle_gateway/<vehicle_id>/state
- More details in the package README.md


Gazebo Sim
Fly
Plan
Land
Return
Pause
Action
100 m
Single
Multi-Vehicle
Hirschgartnerweg
Oerlikonerstrasse
Schaffhauserstrasse
Hirschwiesenstrasse
Hirschwiesenstrasse
A11
A11
A11
Irchelpark
Irchelstrasse
Milchbuck-Tunnel
Winterthurerstrasse
Vehicle 23
00000

Index

Conclusions

Status

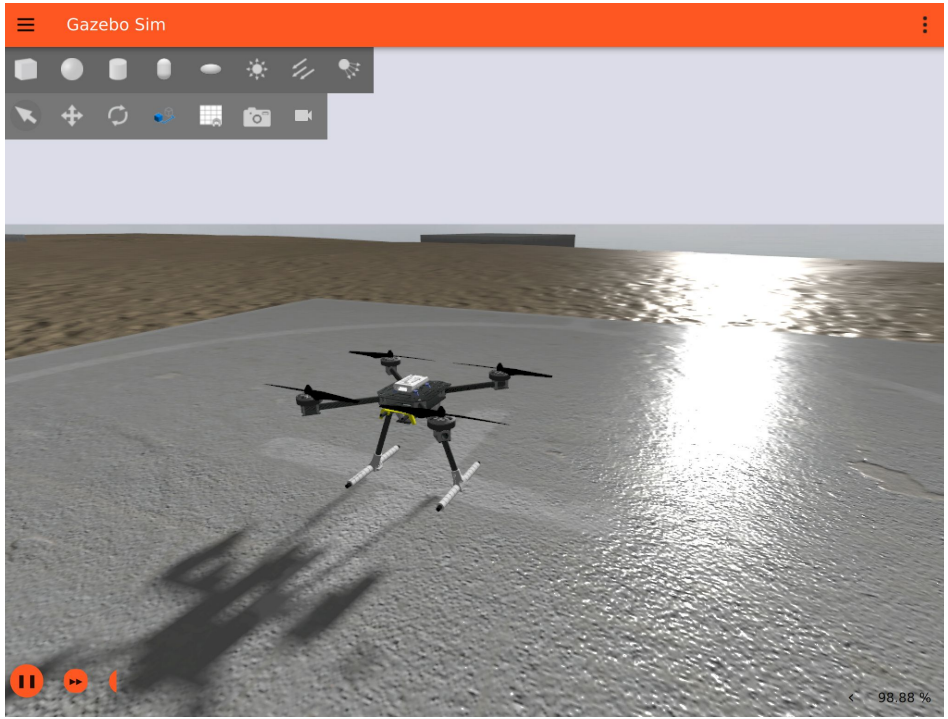
- Support quads, fixed wing planes and VTOLs
- Multirobot
- End to end testing in CI

Future work:

- Add support for others autopilots
  - Ardupilot, cleanflight, etc

Feedback:

- We invite everyone to try it and test it! We are happy to receive your feedback and contributions



Thank you!