SETTING UP AN EXPERIMENT

You have to use the MRS UAV System and implement your experiment in ROS (ROS 1, not ROS 2). The repository for the MRS UAV System contains an installation guide and additional documentation is on the wiki. The most important documents are Mandatory readings for newcomers and How to start the simulation.

You have to prepare and demonstrate your experiment in the Gazebo simulator, which is integrated with the MRS UAV System. If your experiment does not work in the simulator, it cannot be executed in the real world.

These sensors are available for your experiments:

Basics:
- Pixhawk onboard sensors - IMU, Barometer, Magnetometer
- Onboard computer - Intel NUCi7 10th generation, 16GB RAM, core i7 x86 CPU

GNSS:
- M8N GPS receiver
- Emlid reach M2 RTK

Cameras:
- Bluefox MLC200w RGB or Grayscale + various lenses
- Basler dart daA1600-60uc RGB + various lenses
- Basler dart daA1920-160um Grayscale + various lenses
- Realsense D435i - RGBD + IMU
- Realsense D455 - RGBD
- PicoFlexx PMD 1 - ToF
- Realsense L515 - ToF
- Realsense T265

Lidars:
- Garmin Lidar Lite V3 (altitude measurement)
- RPLidar A3 (2D lidar)
- Ouster OS-1-16 (3D lidar)
- Ouster OS-0-128 (3D lidar)

Thermal Cameras:
- TeraRanger Evo Thermal 33
- FLIR Lepton
- FLIR Boson

Special:
- UVDAR relative localization system
- Custom VIO camera - Bluefox MLC200wG + synchronised IMU

Note that some sensors are only available in limited quantities.

You can use multiple drones for your experiment, but there is a difference between the simulator and real world. In the simulator, the code is running centrally on one computer, while in the real world, each drone is running its code onboard. This means that if you want to share any data between the drones, it has to happen over Wi-Fi using the nimbro_network package. You can communicate with ROS topics and services using nimbro_network, but expect a low bandwidth (for example streaming images between multiple drones is not really possible) and unreliability.

When preparing for the real-world experiment, you should consider these guidelines:

Do not hardcode things.
People often start by hardcoding things like the UAV_NAME, or parameters of their programs. This is very impractical during a real-world experiment. The UAV_NAME is stored in the $UAV_NAME environmental variable, and your code should load it and use it. Configurable parameters of your code should be loaded from a config file, which can be changed without the need to recompile your code. Use our example ros node as inspiration.

Do not launch new nodes during flight.
Your node should be running even before takeoff, in a “deactivated” state (e.g. not sending “goto” commands). You should have an activation service, which will be called (manually or automatically) after the takeoff is finished. This way, if your node crashes for some reason, we can see it even before takeoff, which saves time and batteries. Also, starting a new ros node can be hard on the CPU load, which is not what we want during flight.

The real world is different from simulations.
You should expect that in the real world, the initial position of the UAV will not be 0,0 and the heading will not be the same every time. The safety area is constrained by the real-world environment, and therefore it can have an arbitrary shape and orientation. If you are using a GPS/RTK localization, the X-axis will be aligned with the East direction and Y will be aligned with the North direction.

Your code has to have human-readable debug/status outputs
Your node should print out its status in a human-readable form. The output should, for example, contain the current state of your node, information about what data the node is waiting for, what it is publishing, etc.

You have to prepare your own real-world tmux script.
You should start with our template tmux script, copy it into your repository and modify it according to your needs. This tmux script should run all your nodes, and it should contain panes for calling your services (if necessary). You should not run anything outside of this tmux script.