Working document of the development of the SML lab for Quads' autonomous deployment. It contains an hardware description and the students advances.

Quads' Set-Up in SML

- Yaw control
- Hover with integral action (altitude hold)
- Trajectory tracking: circle
- Automatic landing when QUALISYS fails (using landing from quad)
- Velocity estimation (is there a salt and pepper filter?)
- trajectory generation: leader following
- Problems with more than one quad: ROS code needs to be adapted (messages names need to be different for each quad)
- What can we log from quad

1 SetUp

ROS controller for IRIS+ ==========

A small guide to setting up and running this project.

Install ROS

Follow the [official instructions](http://wiki.ros.org/indigo/Installation/Ubuntu) for ROS Indigo. After configuring the Ubuntu repositories and keys, this includes installing the full desktop install via sudo apt-get install ros-indigo-desktop-full (This might download more than 1GB of packages)

Then initialize rosdep sudo rosdep init rosdep update

Make sure the ROS environment variables are set when bash-terminals are launched. To do this, add source /opt/ros/indigo/setup.bash . You can do this, by executing echo "source /opt/ros/indigo/setup.bash" » /.bashrc source /.bashrc in a terminal window.

The installation of rosinstall (package python-rosinstall) is not required.

Setup Workspace

(for more detailed instructions see the [official ROS tutorial](http://wiki.ros.org/ROS/Tutorials/Installingan Create a workspace folder and make sure it contains a folder named src . In this src folder run catkin_init_workspace and make sure this directory is in the ROS_PACKAGE_PATH variable. You can check this with echo $ROS_PACKAGE_PATH . If the directory is missing (which is likely), add the line export ROS_PACKAGE_PATH= /[your ros workspace]/src:$ROS_PACKAGE_PATH to the file /.bashrc .

Copy the relevant folders (currently guad_control , mavros and gui) into the src folder. To make sure that you mavros can access your USB port, you might need to add your user to the group dialout by executing sudo usermod -a -G dialout $USER .

Install the ROS control toolbox sudo apt-get install ros-indigo-control-toolbox

Do NOT do this: Make sure mavros is installed (sudo apt-get install ros-indigo-mavros) and run

Build Project

catkin_make

Run Project

Without Mavros

In two terminal windows (both in the workspace root), run source ./devel/setup.bash roslaunch quad_control iris1.launch and source ./devel/setup.bash rqt –standalone tabbedGUI –args Iris1/

With Mavros

If you want to connect to an actual quadcopter, run the following three blocks, each in his own terminal window in the workspace root: source ./devel/setup.bash roslaunch quad_control iris1_mavros.launch , source ./devel/setup.bash roslaunch mavros apm2.launch and source ./devel/setup.bash rqt –standalone tabbedGUI –args Iris1/

Uninstall ROS

Only if needed:

To uninstall ROS and remove all configuration files, execute the following commands sudo apt-get purge ros-indigo* sudo apt-get purge python-rosdep python-rospkg python-rosinstall sudo apt-get autoremove and remove all lines in /.bashrc concerning ROS.

2 ACRO Mode

Link

"The stick input is a number between -4500 *to* $+4500$*, to represent sort of* $\pm 45°$*. We take that number, and multiply it by Acro_RP_P, to get the the angular rate in centi-degrees. Default RP_P is* 4.5*. So* $4500 * 4.5$ *is* 20200*, or* 202 *deg/second. RP* 10 *will give you* 450 *deg/sec, this is what I flew in my video.*
Angular acceleration rate is the actual rate that the copter can accelerate in roll or pitch. It can't jump from 0, to 450 *deg/sec instantaneously. It could take* 0.5 *seconds. This would mean the angular acceleration rate is actually 900 deg/sec/sec. This will be determined, physically, by the power of the motors, the weight of the frame, responsiveness of the ESC's, etc. We currently don't consider this at all, in the code.*"

3 Qualisys

Instructions for turning Qualisys ON

1. On dock bar of Windows, open **Project** icon
2. Select **/Inet/Labhybrid2**
3. Click on **create new** icon
4. **CTRL+D**
5. **Right click** (on mouse) and select **6DOF**
6. Identified rigid bodies should be there

Instructions for when Qualisys goes down (not always necessary)
(If, when turning on mocap – **rosrun mocap ros_mocap.py**, terminal complains about connection to Qualisys – likely an issue regarding the IP address of the computer running Qualisys)

1. Go to the Qualisys computer and open a Terminal (**cmd.exe**)
2. Write **ipconfig**
3. On *Ethernet Adapter Local Area Connection* search for *IPv4 Address* (copy address, e.g. 130.237.50.84)
4. Search for *mocap_source.py* (on Ubuntu) and change **host** with the copied address (it might be the same though)

Change Body Id of IRISi

- IRISi where $\mathbf{i} = \{1, 2, 3, \cdots\}$
- Go to mocap.launch in the mocap directory and iris**i**.launch in the scenarios directory
- Change id to e.g. **13**
- On terminal, *rosrun mocap ros_mocap.py* followed by *rostopic echo /body_data/id_**13*** should now yield the msg related to IRIS**i**

2015-09-06 Qualisys now has a fixed IP (by Rui Oliveira)

- **sml-qualisys.ddns.net**
- This IP will probably be the same for one year, so you don't have to be checking the IP daily from now on.

To add to manual

- Go to website http://www.pyqtgraph.org/ and open Debian/Ubuntu package python-pyqtgraph_0.9.10-1_all.deb with the software center (it will be installed): necessary for gui where plots are being drawn
- sudo apt-get install ros-<distro>-rqt
- sudo apt-get install ros-<distro>-rqt-common-plugins
- Rotors: do not forget to install additional ROS packages as mentioned in github https://github.com/ethz-asl/rotors_simulator

References