1 Lab Scope and Aim

Flying robots or Micro Aerial Vehicles (UAVs) are becoming increasingly common and span a huge range of sizes and shapes and they are widely used in different applications such as power-line monitoring, bridge inspection, urban structure coverage, forest fire inspection, etc. The aim of this lab is to introduce the students to the concepts of unmanned aircraft modeling, control, as well as into the basics of motion planning. In particular, the robot that you will use in this lab is the crazyflie 2.0 and you will be developing your control and motion algorithms within Simulink framework. This lab assignment involves (a) an intro to the CrazyFlie 2.0 (b) template code to use on the robot.

Prerequisites: The students are suggested to have a good background in the following areas:
- *Automatic Control Systems*
- *Advanced Mathematics*
- *Complete Lab1 and Lab2 Tasks*
- *Complete knowledge of coordinate frames*
- *Simulink and Programming in General*

Lab Supplementary material: This lab is accompanied with the supplementary material to support the assignments. The material is included in the lab3.zip file.

Note1: You should always fly inside the flying area.
Note2: Never fly when the battery level is low.
Note3: Never fly above 2m altitude.
Note4: Before staring your experiments, ask TA's to help you with enabling the tracking in Motion Capture System.
Note5: You should terminate the experiment before hitting the ceiling or any other object, including the nets in the flying arena.
Note6: You should read the description of crazyflie using `https://wiki.bitcraze.io/projects:crazyflie2:index` **and** `https://wiki.bitcraze.io/projects:crazyflie2:userguide:index`
Note7: You should remove your shoes before going inside of flying arena
Note8: Provide plots demonstrating reference and actual trajectories.
Note10: Pose and velocity measurements are provided in global ENU frame through VICON Motion Capture System.
Note11: In the simulink file you receive, you should implement an emergency landing block. This block will take immediate action when you want to land your crazyflie.

2 Simulink

The lab implementation is structured around Simulink. Simulink, developed by MathWorks, is a graphical programming environment for modeling, simulating and analyzing dynamical systems. Its primary interface is a graphical block diagramming tool and a customizable set of block libraries. More information regarding simulink can be found at `https://se.mathworks.com/products/simulink/getting-started.html`

3 CrazyFlie

The Crazyflie Nano Quadcopter is a miniature quadcopter as depicted in Figure 1. It weights about 27 gr and is 9 cm motor-to-motor. The flight time is up to 7 min with standard 170 mAh Li-Po battery. It has IMU with 3 axis gyro, 3 axis accelerometer, 3 axis magnetometer, and high precision pressure sensor. The framework is already installed and you should not upgrade the on-board framework. More information regarding crazyflie can be find in `https://www.bitcraze.io/crazyflie-2/`.

Figure 1: Crazyflie 2.0.

3.1 CrazyFlie bring-up

This section provides instructions on how to connect to Crazyflie using the lab's laptop. For logging to the computer use the following information:
Username: xxx (to be provided at the lab)
Password: xxx (to be provided at the lab)

Connecting to Crazyflie For connecting to the Crazyflie You need to connect a crazyradio dongle (Figure 2) to the USB port of the laptop. For guaranteeing low latency in connection, the CrazyRadio should always look towards the platform.
Open the terminal in Ubuntu, the icon shape is shown in Figure 3. You can press the icon or use following shortcut: "Ctrl + Alt + T"
The terminator has different windows as presented in Figure 4, in each of them you can run different commands. To be more specific, it is equivalent when you open multiple command windows in

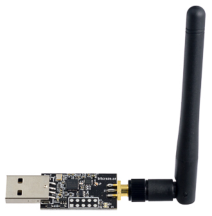


Figure 2: Crazyradio dongle.

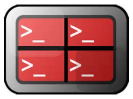
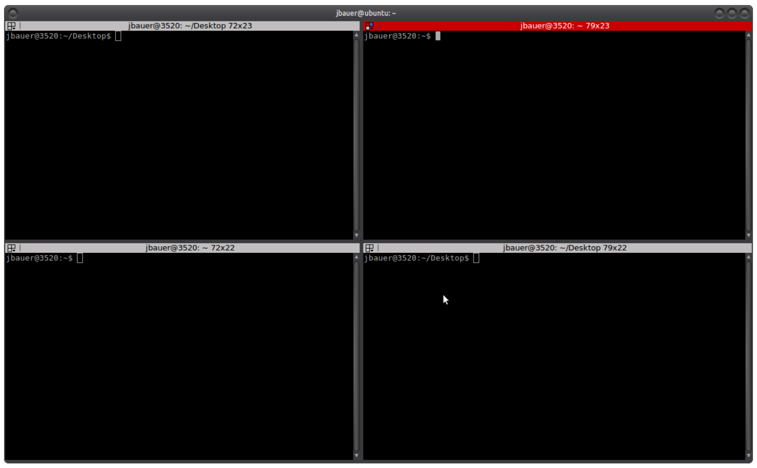


Figure 3: Terminator icon.

Microsoft Windows.



```
jbauer@3520:~/Desktop$
```

```
jbauer@3520:~$
```

```
jbauer@3520:~$
```

```
jbauer@3520:~/Desktop$
```

Figure 4: Terminator windows.

For connecting to Crazyflie you must run the following command in one of the terminator windows:

1. *run* - start crazyflie

In another command window type “matlab”, to open MATLAB. Open the simulink files accompanied with the lab. The data flow to Crazyflie is similar to the turtlebot from Lab2. Now you can proceed to the assignment section.

4 Assignments

Task 1: Hovering Create a controller for the UAV to hover in a stable manner at 1 m height. Additionally, the controller should be able to handle external position or heading disturbances and always recover the hovering state.

Task 2: Pose A to Pose B Create a controller of your choice, which can track arbitrary position and yaw references within the flying volume of the FROST lab. The controller should be stable and converge to the position and yaw references smoothly without overshoot or oscillations.

Task 3: Following a trajectory In this task you will define different trajectories and validate your controller with each of them. The trajectory followed by the crazyflie and reference path should be plotted for evaluation.

- Create a square trajectory at constant altitude z and the UAV should follow the reference trajectory while looking at the next waypoint.
- Create a circular trajectory with constant altitude and the UAV heading should be towards the center of the circle while it follows the reference trajectory.
- Modify the circular trajectory by tilting it 45 degrees in z as depicted in Figure 5.

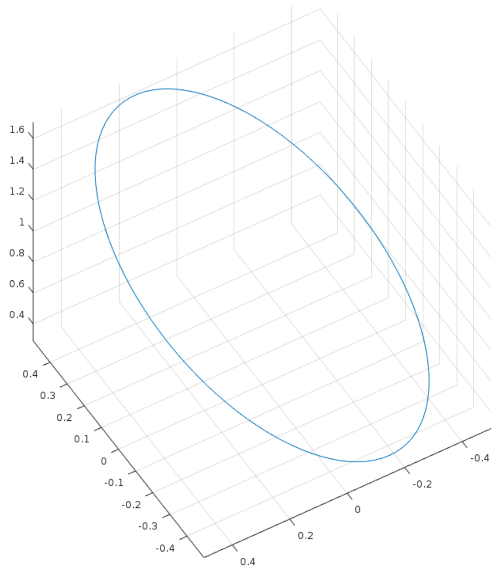


Figure 5: A circular path tilted 45 degrees in space with a diameter of 1 meter.

Task 4: Leader-Follower (optional and for higher grade) In this task you will create a "follower" - your crazyflie should track the crazyflie of another team with a constant off-set while it is following its reference trajectory. (Hint: add a new odometry subscription block in your simulink file)

Task 5: Collision Avoidance for multi-agent scenarios (optional and for higher grade)
In this task you will create a collision avoidance system such that your crazyflie will avoid the crazyflie of another team. Using artificial potential field concepts, a simple way to do so could be to shift your reference p_{ref} proportional to the distance to the other crazyflie by some Δp_{ref}. Define a radius of influence r^i. If the other crazyflie is closer than r^i, shift your position reference by

$$\Delta p_{ref} = L(1 - \frac{|| \hat{p} - \hat{p}_a ||}{r^i}) \frac{\hat{p} - \hat{p}_a}{|| \hat{p} - \hat{p}_a ||} \tag{1}$$

where $\hat{p}$ is the estimated position of your crazyflie, $\hat{p}_a$ is the estimated position of the other crazyflie, and L is a gain.