MEAM 620 Project 1 Phase 4

Report due: TBA

1 Introduction

This document details instructions for using the hardware and software as well as the assignment and submission procedures.

For this assignment, you will work as a team to implement the controller and trajectory generator you developed in the previous phases on a Crazyflie 2.0.

You will need to sign up for lab time slots online, the instructions for which would be posted on Piazza. There will be two groups in the lab at the same time. Each group has its own Crazyflie and is provided with a desktop computer from which all code will be run.

You are encouraged to bring your own laptop so all group members can fully participate. You may also want to bring USB sticks to transfer files more easily onto the desktop.

2 Safety

Due to its small size, the only serious risk is facial injuries. Here are the safety rules.

- Only one (1) student total at any one time is allowed in the VICON area.
- Students in the VICON area must wear safety goggles.
- In case a quad is out of control, protect your face, duh.

3 Assignment

Your task for this phase is to implement a controller and trajectory generation of your choosing and get it to work with a real CrazyFlie quad rotor. As a group, you will need to complete the tasks below and demonstrate to a TA that you were able to fly the trajectories. The Matlab environment you will be using is located in the `/studentcode` folder on the desktop provided. You are welcome to create additional files as needed to organize your scripts - just keep all of your work in your folder.

3.1 What to bring

You must bring the following scripts from the previous phases:

- `controller.m`
  Note: will almost certainly require gains tuning.
- `trajectory_generator.m`
  Be prepared: you will likely have to adjust the velocity profiles of your trajectory generator on the fly!

Bringing a laptop may also be a good idea.

3.2 What to expect

Your lab session will be following these steps:

- safety instructions, familiarize yourself with hardware and the matlab environment
- tune your controller until the robot hovers stable. Contact the TA so he can confirm you passed this step.
- fly simple test waypoints that you make up yourself, until you are confident flying the test waypoints.
- fly each of the 3 test trajectories while the TA is watching to confirm completion.
- after each run, save the desired and actual trajectories to a file to be included in your report.

Ideally you would use the same controller for all trajectories, but if you have to tune the controller inbetween that is ok as well.

3.3 What to submit

There are three deliverables for this assignment:

1. **Group Report**: Each group should submit a report approximately five pages in length (including plots - you don't need five pages of pure text) containing the following information:
   (a) The names of the group members
   (b) A description of any controller that you used, including equations, and the actual gains you used.
   (c) A description of your trajectory generation for waypoint following, including equations if relevant.
   (d) Plots of the actual vs desired position of the quadrotor for all three tasks. Make sure to label your axes and provide multiple views in order to translate the 3D information to the 2D paper.

   The report should be a pdf and should be sent to meam620@gmail.com.
2. **Code**: A .zip file of all code that your team used for this phase. Send this with the report. Note, you only need to submit one code base for the entire team.
3. **Individual "Report"**: Each person should send a one paragraph email to meam620@gmail.com with their name and group number and a short description of what they personally did for the group, and a short description of the contributions of their teammates.

4 Startup Procedure

1. Make sure that the Vicon system is on and running, and tracking the quad you intend to fly.
2. Check that the battery indicator of the CrazyFlie is blinking (not solid) red, and that the VICON tracks the quad.
3. Place the quad near the middle of the VICON area (Safety goggles!)
4. From your matlab environment, create a new quad object and takeoff!

5 Information about the CrazyFlie 2.0

(Disclaimer: the following section is partially cut-and-paste from Bitcraze's website)

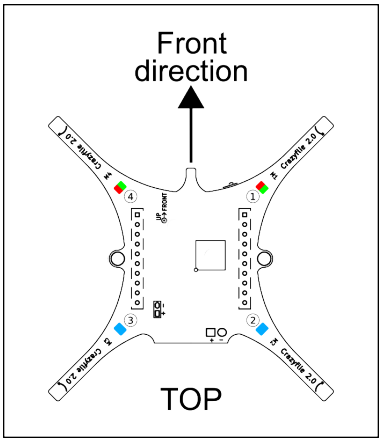

Figure 1: The meaning of the lights:

- power on and all is good: The blue LEDs (2 and 3) are fully lit and the front right LED (1) is blinking red twice every second.
- power on and all is good but sensors are not yet calibrated: The blue LEDs (2 and 3) are fully lit and the front right LED (1) is blinking red with 2 seconds interval. Put the Crazyflie 2.0 on a level surface and keep it absolutely still to calibrate.
- radio connected: The front left LED (4) is flickering in read and/or green.
- battery low: The front right LED (1) is fully lit in red. It's time to land and re-charge the battery.
- charging: The back left blue LED (3) is blinking while the right back blue LED (4) is lit.
- self test fail: The right front LED (1) is repeatedly blinking five short red pulses with a longer pause between groups.

First things first: On the bottom of each CrazyFlie, you can find a marker that should read "m01" through "m07". Make sure you are flying the robot you intend to, not the one of the other team!

The *back* of the CrazyFlie is where the battery cables are hanging, the *front* is where the little nose protrudes from the board. The CrazyFlie is charged through a micro-usb cable, and (in our situation) controlled via 2.4GHz RF signal.

To switch the CrazyFlie on, push the microscopic button on the *front* of the board, somewhat hidden by the battery. This will initiate the startup sequence:

- run self tests - the Crazyflie 2.0 checks that the hardware is OK
- calibrate sensors - the Crazyflie 2.0 reads its sensors to get base values. It must be absolutely still to do this, so it is best to put it on a level surface for a second.
- ready to fly!

The lights of the CrazyFlie are labeled M1-M4, see Fig. 1 for their meaning. The most important one is the battery indicator: blinking red is good, SOLID RED IS BAD. If the battery is empty you will be given a different quad to fly.

6 After a Crash

- make sure the battery is centered and the quad balances
- all props are straight, level, and spin freely?
- none of the motor holders are broken?
- if it no longer flies right contact the TA for repairs or a new quad.

7 Using the VICON

There is a separate PC dedicated to the VICON. Unless it is already running, start the "Vicon Tracker" software, and click to the "Objects" tab (we already defined tracker objects for you).

Scroll down until you find the crazyflie you want to track and mark it. Make sure it is e.g. "crazym04", the letter "m" must be there! If you now put the quad into the VICON area you should see it show up on the Vicon Tracker software, with a little coordinate system attached. If you don't see the coordinate system, the tracking is not working.

Any coordinates you will be given refer to the VICON's reference frame. The [0,0,0] spot is marked on the ground in the middle of the area, with the x-axis pointing away from you (towards the elevators), the y-axis to the left (towards the hallway), the z-axis up.

8 Software

The overall setup is shown in Fig 2. Students will be working on the machines meam620-a and meam620-b that run the Matlab MAV environment.

The backend, which runs on the host "demo-nuc", not only relates ROS messages from the VICON and the CrazyRadio, but also implements simple controllers to hover or land the CrazyFlies. From the Matlab MAV environmenty ou can switch over to those controllers in case your own software malfunctions.

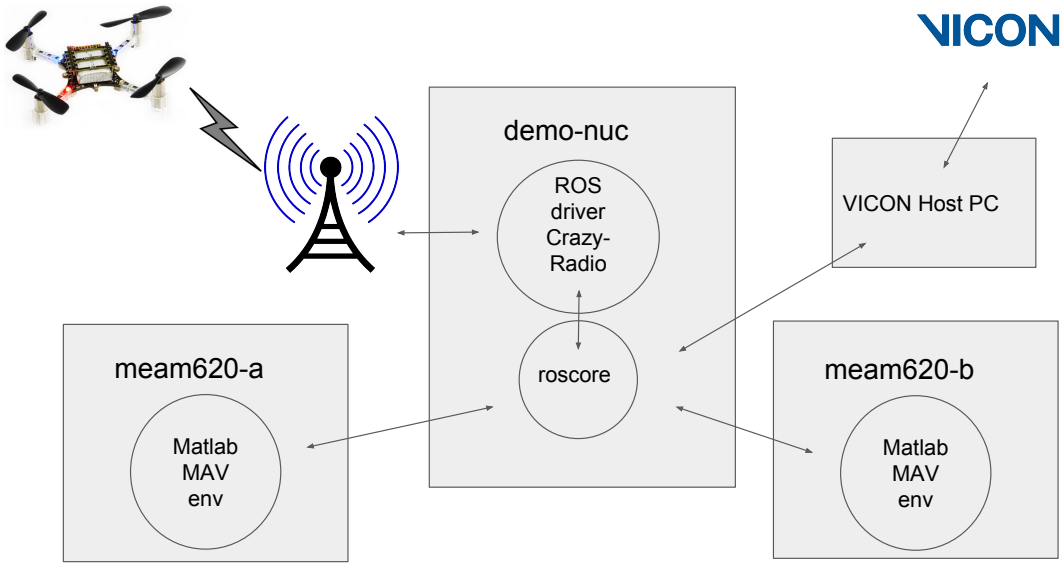

Figure 2: Lab section setup for using the Crazyflies. The student desktops meam620-a and meam620-b are connected to the machine "demo-nuc", which acts as the ROS master and relays message to the crazyflies via a driver. It also connects to the VICON motion capture system.

9 Using the Matlab MAV Environment

The development environment is completely written in Matlab and will accept the code you developed in the earlier phases as drop-in. When you arrive, you will be given a complete copy of the sources which you can modify as you see fit. Here is a rough outline of the functionality provided.

```
%
% Instantiate a new quad object. This also starts a new ros node, and
% callbacks triggered by the incoming messages from the Vicon will
% start to occur. If you want to stop the callbacks (a good idea if
% you decide to edit the code!), do this:

>> rosshutdown

%
%
>> quad = QuadrotorROS('crazym05');
Shutting down existing ros nodes...
Initializing global node /matlab_global_node_39791 with NodeURI
http://192.168.129.233:41271/

%
% Now command the quad to take off and hover. This
% is all done by the backend, your software is not involved yet.
%
>> quad.takeoff;
Motors 1
Taking off

%
% Here is how to land it. Again, all this is done by the backend.
```

```
%
>> quad.land;
Landing
Motors 0
```

```
%
% To test your controller, first take off, then switch to
% ``shover'' mode (student hover)
%
>> quad.takeoff;
Motors 1
Taking off
>> quad.shover;   % now your code is being exercised!!!
>> quad.land;     % the backend takes over and lands it safely
Landing
Motors 0
```

```
%
% This is how to  invoke the trajectory generator and fly the first
% course of test waypoints.
%
>> twp = make_test_waypoints;
>> quad = QuadrotorROS('crazym04');
>> quad.takeoff;
>> quad.sgo_to_waypoint(twp{1});
>> quad.land;
```

```
%
% There are some skeletons provided in the source code, like
%
% quad.dryrun()       runs the trajectory generator which you can then
%                     plot with
% quad.plotPosition() plots the desired and actual position
%
% Feel free to expand on those
%
```

10 Differences Between Simulator and Real Environment

- The Crazyflies accept attitude commands, i.e. you can directly command the angle, and the on-board controller will close the loop for you. Any moments you command will be ignored.
- You will have to re-tune your gains because the mapping between commanded thrust and actually observed force is not perfect, and in general the attitude controller is much less responsive than what you developed in the simulator.
- The good news is: you will only have to tune the position gains.
- You may have to adjust the mass slightly up or down to make the quad rotor hold altitude at exactly the desired position.