If you are looking for information about the lab, consult the documentation.

Exercise 1. (*Computer setup*)

a) Follow these instructions to clone and install the CPM Lab software for simulation. You will need MATLAB and Simulink with the DDS add-on, the Control System Toolbox, the Optimization Toolbox, and the System Identification Toolbox. If you need another toolbox later, they are easiest to install using the MATLAB installer. We recommend setting up a real Ubuntu system, but you can also use our ready-to-use virtual machine.

b) Go through the tutorials. You can skip parts related to using the physical lab and high-level controllers (HLCs) in C++.

Exercise 2. (*Setting up your development repository*)

Set up the folder structure for your development repository. You should not need to change any code outside of your repository.

a) Join your team's repository at GitLab to which you have been invited per email.

b) For smooth execution of your code in the lab, clone your repo such that you end up with the following folder structure (example for group a, team 1):

```
$HOME/dev
|-- software
    |-- cpm_lib
    |-- high_level_controller
    |-- ...
|-- lu2021
    |-- a1
        |-- ...
```

The path to your repository will be called `TEAMREPO` in the following. Other paths are given from `HOME/dev/software`.

c) You were also invited to a template repo, where some skeleton code and several functions are provided. Initialize your repo with this template.

Exercise 3. (*Communication with and control of vehicles*)

a) How is the communication between components in the lab realized?

b) How can you read a vehicle's state in MATLAB?

c) The file `cpm_lib/dds_idl/VehicleCommandDirect.idl` defines the message format of messages sent to control the vehicle in direct control mode. What are the elements of the interface definition language-file?

d) Modify the example `TEAMREPO/13_vehicle_basics/main_direct.m` to have the vehicle move in a sinus-shape.

e) Repeat Task c) and Task d) for the path tracking mode.

f) Repeat Task c) and Task d) for the trajectory following mode.

g) How is the experimental concept for synchronization and determinism presented in the lecture realized in the lab?

Always ensure that the middleware's period in the lab control center "Parameters" tab is set according to your expectations for the control loop period in the HLC.

Exercise 4. (*Path tracking basics*)

a) Understand how the Stanley controller [1] works.

b) The file `mid_level_controller/src/Controller.cxx` implements the speed controller and calls the path tracking controller. How is the speed controlled on the lab vehicles? What are the benefits of having the speed control on the vehicle instead in the HLC?

c) If you want to define a closed path, what is the requirement on the first and last point so that the interpolation along the path's distance works correctly from start to end?

Exercise 5. (*Position control of path tracking vehicle*)

Have a vehicle follow a reference position on a path. Take a look at `TEAMREPO/common/get_path_points.m`. Understand how the path `'outer_lane'` is defined and use it from now on.

a) Write a function to determine the distance traveled along your path from the measured vehicle state. You can use the function `TEAMREPO/common/compute_distance_on_path.m` as a starting point. Remember that traveling along the path in the lab aims to emulate traveling along an infinite path while always progressing. Therefore, extend the function such that the result is always increasing as the vehicles are progressing.

b) Identify the input/output dynamics of the velocity. A second order delay system should provide reasonable results. What parts of the actual system are responsible for these system dynamics?

c) Create a state-space model of one vehicle that uses your input and the distance traveled as an output. Provide the model in `TEAMREPO/common/get_longitudinal_model.m`.

d) Design a PID controller to track a reference position s_{ref} given as

$$s_{\text{ref}}(t) = 1.1 \cdot t + 0.5 \cdot \sin t. \tag{1}$$

e) Design a model predictive control (MPC) to follow the reference position s_{ref} as defined in Equation 1. The following constraints on the input should be considered

$$\begin{aligned} v_{\text{min}} &= 0\,\text{m/s}, & a_{\text{min}} &= -1\,\text{m/s}^2, \\ v_{\text{max}} &= 1.5\,\text{m/s}, & a_{\text{max}} &= 0.5\,\text{m/s}^2. \end{aligned} \tag{2}$$

Convert the constraint on the acceleration to a constraint on your input change. Use the future reference trajectory as an input to your MPC. You can use the MPC implementation from the template repo. Feel free to understand the computations and use it as is or adjust it to your needs.

References

[1] Gabriel M. Hoffmann, Claire J. Tomlin, Michael Montemerlo and Sebastian Thrun. *Autonomous automobile trajectory tracking for off-road driving: Controller design, experimental validation and racing*. IEEE American Control Conference, 2007.