Project 2. Control and communication

In project 2, you will be able to choose one of the two different tasks, depending on the topics and technologies you would like to learn more about. In *Task 1*, you will be able to work in Jupyter with mathematical modeling of a close-loop dynamical system and tuning of controller parameters. In *Task 2*, you will be able to develop two Python programs: one will simulate sensor reading that are published using ZeroMQ, and the other will subscribe to these readings and do simple processing.

See the tasks on the pages below.

Task 1: Tuning of a closed-loop controller

Your task is to design a closed-loop controller for a system with known mathematical model, namely the model of a vehicle velocity used in the seminars:

$$\frac{dv(t)}{dt} = \frac{F_p u(t) - \frac{1}{2}\rho A C_d v(t)^2}{m} - g sin(\theta(t))$$

where:

- $v(t)$ is the vehicle's velocity, m/s (process variable);
- $u(t)$ is the percentage of a combined gas/brake pedal, between -50% and 100% (control variable);
- $\theta(t)$ is the slope of the road, radians (disturbance);
- $m = 700\ kg$ is the mass of the vehicle;
- $A = 5\ m^2$ is cross-sectional area of the vehicle;
- $\rho = 1.255\ {}^{kg}/_{m^2}$ is air density;
- $g =\ 9.80665\ {}^{m}/_{s^2}$ is the gravitational acceleration constant;
- $F_p = 30\ N/(\%\ pedal)$ is the thrust parameter.

Perform the four steps described below and prepare a report inside the Jupyter notebook.

Step 1. Simulate the open-loop system with step input using the `scipy.integrate.odeint` or `scipy.integrate.solve_ivp` function.

Step 2. Define a model for a closed loop system with either P-only control or PI-control.

Step 3. Simulate the closed-loop system together with tuning the parameters of the controller. You may choose either the *setpoint tracking* scenario (changing velocity setpoint $v_{SP}(t)$ in time) or disturbance rejection scenario (changing slope $\theta(t)$ in time).

Step 4. Experiment with the developed model by analyzing how it reacts to different controller parameters, different values of A and m, and different scenario of setpoint function $v_{SP}(t)$ or disturbance function $\theta(t)$.

Task 2. Sensor simulator with publish/subscribe communication

Your task is to develop a simulator of a component that acts as a sensor of road slope (publishes periodic updates using ZeroMQ), together with a ZeroMQ subscriber that receives sensor readings and prints whether the road is straight, uphill or downhill.

Perform the steps described below and write a report on your work. Submit the report, together with two Python files: (1) the sensor simulator (publisher), and (2) the subscriber.

Step 1. Define a scenario of how road slope changes with time.

Step 2. Decide on sensor update rate (e.g. 500 ms or 1 s).

Step 3. Implement a sensor simulator component that follows the defined scenario and performs publishing using ZeroMQ PUB socket at the defined rate.

Step 4. Implement a subscriber component that utilizes ZeroMQ SUB socket and, upon receiving of each slope value, prints to the terminal whether the road is straight, uphill or downhill.

Step 5. Launch the two components together and assure that everything works as expected.