Robot engineering with modular simulation

Autonomous robots, TME290

Ola Benderius
ola.benderius@chalmers.se

Applied artificial intelligence
Vehicle engineering and autonomous systems
Mechanics and maritime sciences
Chalmers

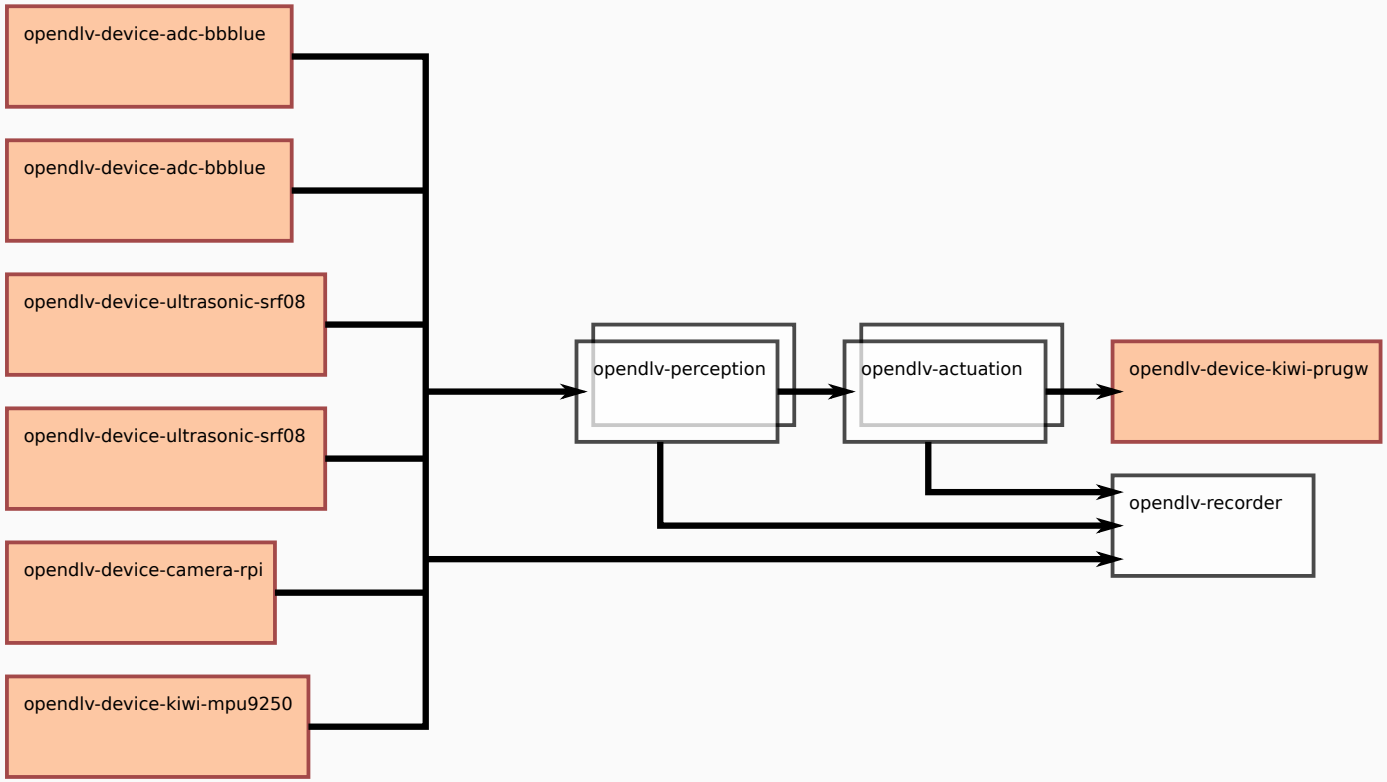
Recording data

```
docker run --rm -ti --init --ipc=host -v ${PWD}/recordings:/recordings -v /tmp:/tmp \
  registry.opendlv.org/community/opendlv-data-recorder-video-h264:1.0 --cid=140 --name=img.i420 \
  --rec=/recordings/video.rec --width=640 --height=480 --verbose
```

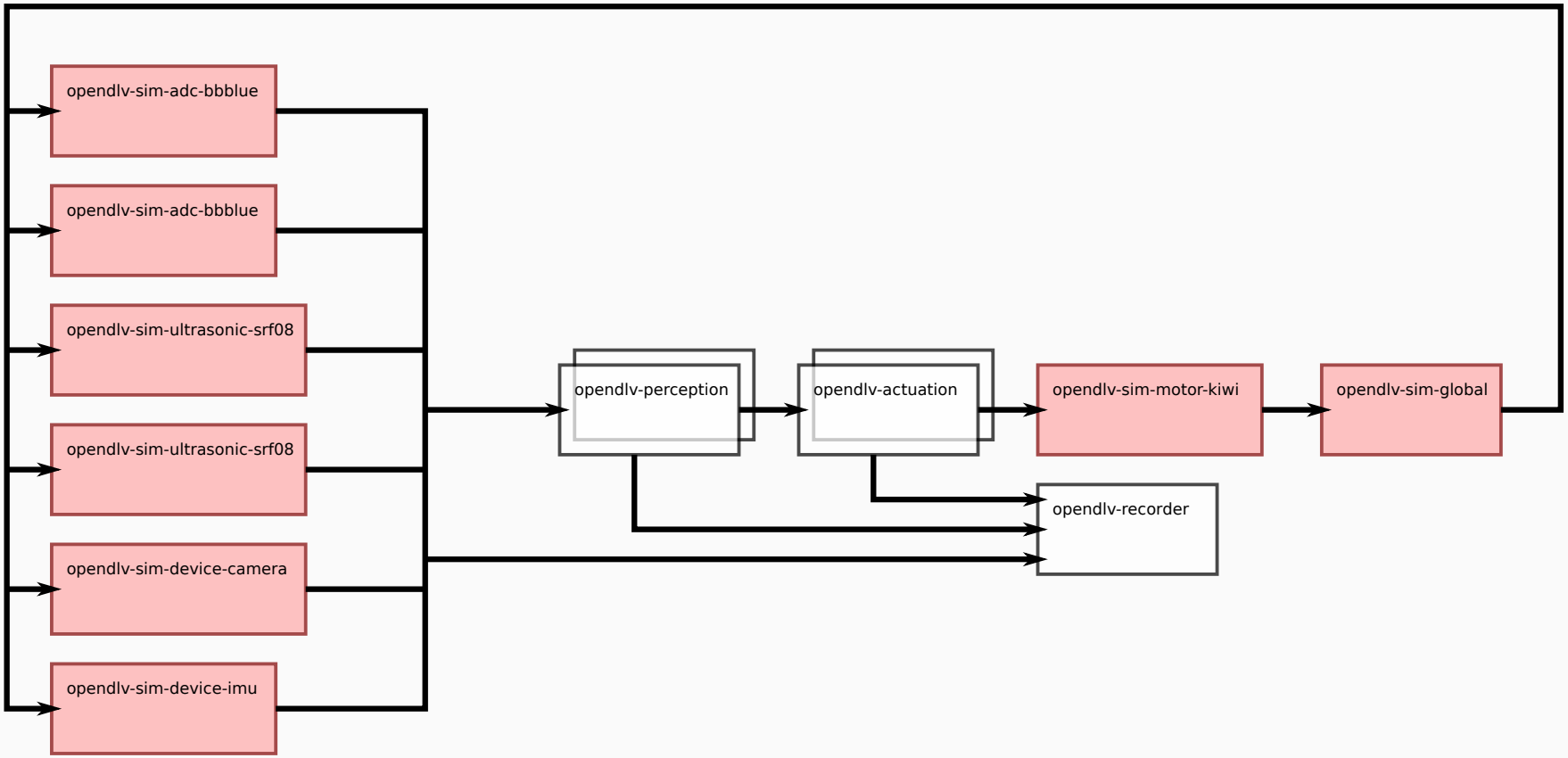
- UPDATE: If the CID is present, then all evenlopes are also captured into the `.rec` file. See all options HERE
- Also, there was a bug in the camera service: You need to update to the next version:
  `opendlv-device-camera-rpicamv2:v0.1.2`
  (we can help you).

Kiwi simulation

The Kiwi microservices



The Kiwi microservices, for simulation

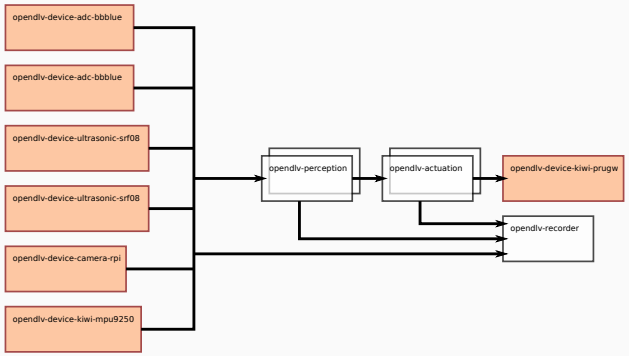


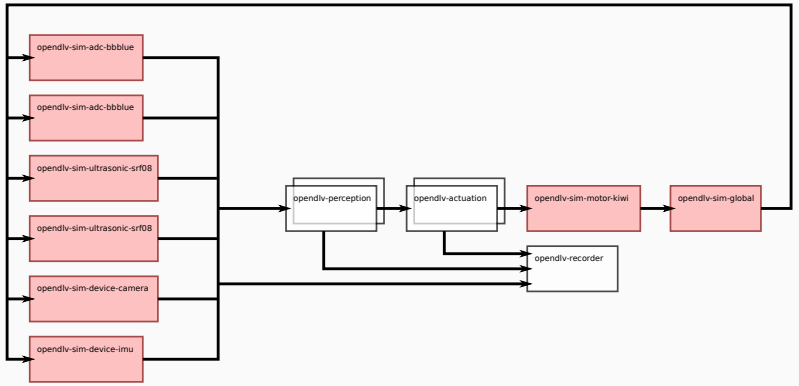
Simulation principle

Since a microservice does not know where the data comes from, it only care about type and content, then we can easily change microservices into simulated components (same principle as for replay).

- Even parts of the system can be simulated, alongside with physical parts

Messages, IR sensor (left and right)





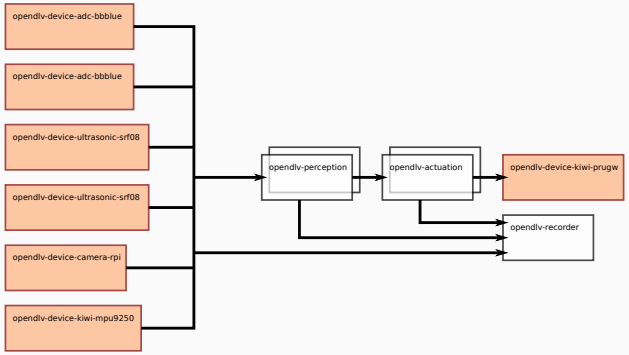
`opendlv-device-adc-bbblue`

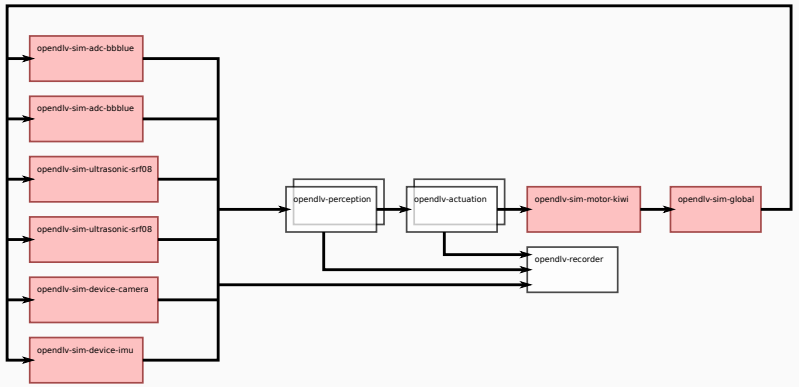
- Output: VoltageReading, DistanceReading
  (the message namespaces are omitted for readability)

`opendlv-sim-adc-bbblue`

- Input: Frame
- Output: VoltageReading, DistanceReading

Messages, ultransonic sensor (front and rear)





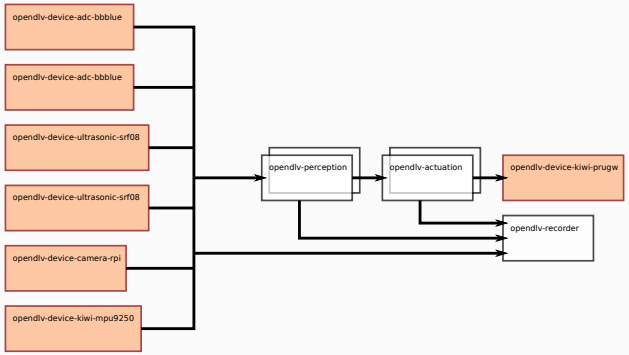
`opendlv-device-ultrasonic-srf08`

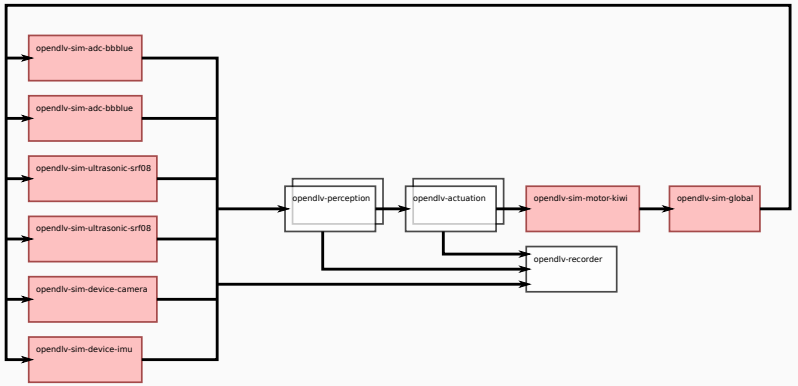
- Output: DistanceReading

`opendlv-sim-ultrasonic-srf08`

- Input: Frame
- Output: DistanceReading

Messages, camera





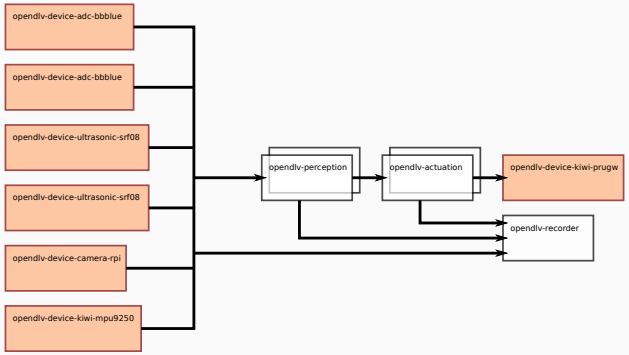
`opendlv-device-camera-rpi`

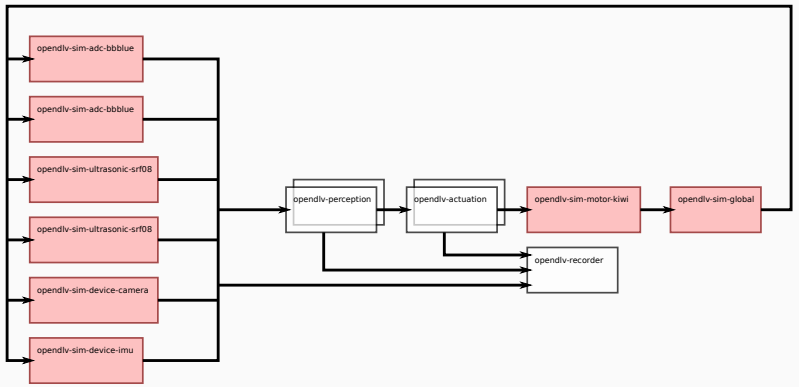
- Output (shared memory): RGB and i420 image data

`opendlv-sim-device-camera`

- Input: Frame
- Output (shared memory): ARGB and i420 image data (not yet updated)

Messages, IMU





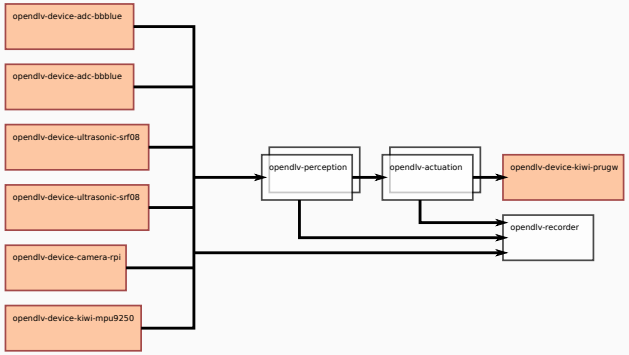
`opendlv-device-kiwi-mpu9250`

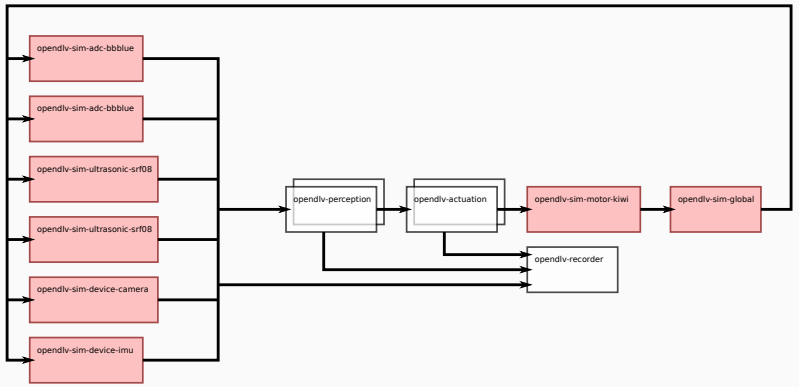
- Output: AccelerationReading, AngularVelocityReading, MagneticFieldReading, Orientation, GeodeticHeadingReading

`opendlv-sim-device-imu`

- Input: Frame, KinematicState
- Output: AccelerationReading, AngularVelocityReading, MagneticFieldReading, Orientation, GeodeticHeadingReading

Messages, motor controller





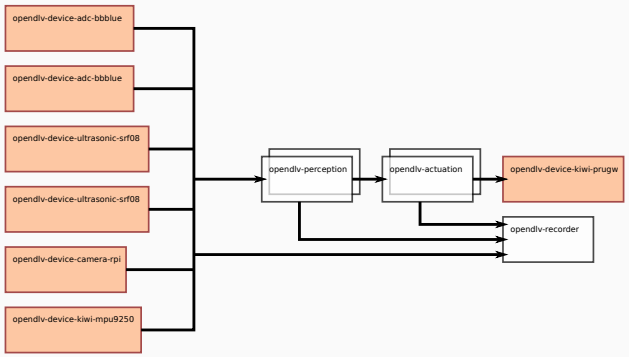
`opendlv-device-kiwi-prugw`

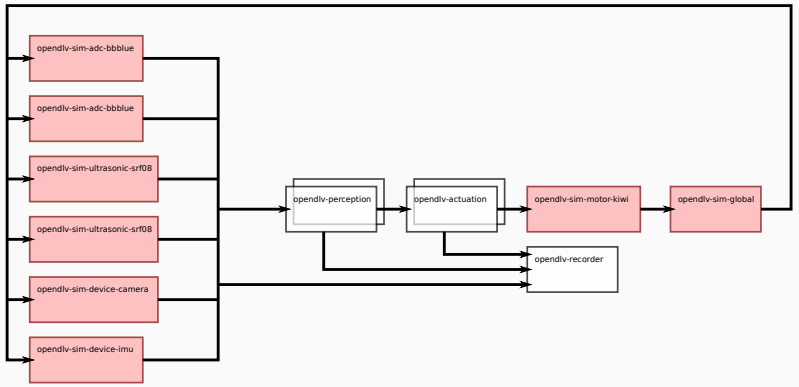
- Input: GroundSteeringRequest, PedalPositionRequest

`opendlv-sim-motor-kiwi`

- Input: GroundSteeringRequest, PedalPositionRequest
- Output: KinematicState

Messages, integrator (global coordinates)





`opendlv-sim-global`

- Input: KinematicState
- Output: Frame

How to use the simulation

On the Canvas page, we have uploaded the `simulate-kiwi.yml` and the `simulation-map.txt` files.

How to use the simulation

On the Canvas page, we have uploaded the `simulate-kiwi.yml` and the `simulation-map.txt` files.

An example without camera

This is simulation without the camera. This is relevant for the home assignments. For the project we would like to add camera, and that will be covered in the end of the lecture.

The `simulation-map.txt` is used for sensor simulation.

The `simulation-map.txt` is used for sensor simulation.

Let's have a look inside:

```
-2.0,-2.0,-2.0,2.0;
-2.0,2.0,2.0,2.0;
2.0,2.0,2.0,-2.0;
2.0,-2.0,-2.0,-2.0;
```

Let's go through the `simulate-kiwi.yml` file (for docker-compose), part by part.

```
    virtual-space:
        image: registry.opendlv.org/community/opendlv-virtual-space:1.0
        network_mode: "host"
        command: "/usr/bin/opendlv-virtual-space␣--cid=111␣--freq=50␣--frame-id=0␣--x=0.0␣\
␣␣␣␣␣␣␣␣␣␣--yaw=0.2␣--timemod=1.0"
```

```
    virtual-motor-kiwi:
        image: registry.opendlv.org/community/opendlv-virtual-motor-kiwi:1.0
        network_mode: "host"
        command: "/usr/bin/opendlv-sim-motor-kiwi␣--cid=111␣--freq=200␣--frame-id=0␣--timemod=1.0"
```

```
    virtual-rangefinder-ultrasonic-front:
        image: registry.opendlv.org/community/opendlv-virtual-rangefinder-ultrasonic-srf08:1.0
        network_mode: "host"
        volumes:
          - ./simulation-map.txt:/opt/simulation-map.txt
        command: "/usr/bin/opendlv-virtual-rangefinder-ultrasonic-srf08 \
          --map-file=/opt/simulation-map.txt --x=0.2 --y=0.0 --yaw=0.0 \
          --cid=111 --freq=10 --frame-id=0 --id=0"

    virtual-rangefinder-ultrasonic-rear:
        image: registry.opendlv.org/community/opendlv-virtual-rangefinder-ultrasonic-srf08:1.0
        network_mode: "host"
        volumes:
          - ./simulation-map.txt:/opt/simulation-map.txt
        command: "/usr/bin/opendlv-virtual-rangefinder-ultrasonic-srf08 \
          --map-file=/opt/simulation-map.txt --x=-0.2 --y=0.0 --yaw=3.14 \
          --cid=111 --freq=10 --frame-id=0 --id=1"
```

```
    virtual-adc-bbblue-left:
        image: registry.opendlv.org/community/opendlv-virtual-adc-bbblue:1.0
        network_mode: "host"
        volumes:
          - ./simulation-map.txt:/opt/simulation-map.txt
        command: "/usr/bin/opendlv-virtual-adc-bbblue␣--map-file=/opt/simulation-map.txt␣\
␣␣␣␣␣␣␣␣␣␣--x=0.0␣--y=0.1␣--yaw=1.57␣--cid=111␣--freq=10␣--frame-id=0␣--id=2"

    virtual-adc-bbblue-right:
        image: registry.opendlv.org/community/opendlv-virtual-adc-bbblue:1.0
        network_mode: "host"
        volumes:
          - ./simulation-map.txt:/opt/simulation-map.txt
        command: "/usr/bin/opendlv-virtual-adc-bbblue␣--map-file=/opt/simulation-map.txt␣\
␣␣␣␣␣␣␣␣␣␣--x=0.0␣--y=-0.1␣--yaw=-1.57␣--cid=111␣--freq=10␣--frame-id=0␣--id=3"
```

```
    logic-test-kiwi:
        image: registry.opendlv.org/community/opendlv-logic-test-kiwi:1.0
        network_mode: "host"
        command: "/usr/bin/opendlv-logic-test-kiwi␣--cid=111␣--freq=10"
```

In the second home assignment you should use two of these microservices as a starting point.

- opendlv-logic-test-kiwi (LINK)
- opendlv-sim-motor-kiwi (LINK)

Let's take a look at the logic (LINK)

Let's take a look at the logic (LINK)

- Study this on your own later, it is expected that you understand most parts of the code.

Now it's time to test the simulation.

Now it's time to test the simulation.

- Simply start the use case with '`docker-compose -f simulate-kiwi.yml up`'
- Make sure that the `simulation-map.txt` is in the same folder

Recording the results

Most often it is desired to record the results of the simulation. We have therefore prepared a second docker-compose file named `interface-kiwi.yml` with a user interface.

Recording the results

Most often it is desired to record the results of the simulation. We have therefore prepared a second docker-compose file named `interface-kiwi.yml` with a user interface.

- Simply start the use case with '`docker-compose -f interface-kiwi.yml up`'
- To access the interface, point your browser to http://localhost:8081
- To record: Start the interface, start recording, in a second terminal start the simulation
- To analyse the data: Export as CSV (for example the opendlv::sim::Frame for position)

To modify a microservice, do the following:

- Download the source code, for example from HERE
- Make changes
- Recompile with, for example:
  `docker build -t registry.opendlv.org/community/opendlv-logic-test-kiwi:1.0 .`
- Restart the docker-compose. Your local version will replace the online one.

How about camera simulation?

It can be added to the simulation as a separe microservice: HERE

Questions

Please post all questions on the Canvas discussion pages, in that way we can all benefit from the answers, and I can highlight important outcomes.