Using maps on the robots

For some parts of your software you will be using maps. In the simulator these are automatically shown. When working with the real robots you wont have a map. This means it is hard to see if localization is working correctly. You also cannot visualize the plan of your robot since this plan is defined in map frame.

Confession

The reason that sim-rviz can show the map is because we are cheating. Usually you can only visualize that which your robot has knowledge of. However in sim-rviz we are simultaneously visualizing what your robot knows and the (simulated) reality. Best practice is to only visualize what the robot knows even when you are simulating.

Rviz settings

You will have to adjust some settings in Rviz to see the map. In the list on the left on the very top you will see the option "fixed frame". It is now set to "odom". i.e. it shows the robots position according to the odometry. Change this option to "map". Your robot will become white which indicates that Rviz no longer knows where your robot is. This is correct. Later in this document we will tell you how to publish the pose of your robot.

Define map

To define the map for visualization go to the robot using `ssh<robot>` and use

```
define-map <map_metadata.yaml>
```

This will allow you to see the map in Rviz. You will still need to import the map in your cpp code separately.

Send Pose Estimate

In order for Rviz to show your robot it must know where your robot is in map frame. To this end we have a function in `emc::io` called `sendPose`. Use it like this:

```
#include <emc/io.h>
#include <emc/rate.h>

int main()
{
    // Create IO object, which will initialize the io layer
    emc::IO io;

    // Create Rate object, which will help keeping the loop at a fixed frequency
    emc::Rate r(10);

    // Loop while we are properly connected
    while(io.ok())
    {
            double x = 0.88;
            double y = 3.42;
            double theta = 1.57;
        // Send a pose estimate for visualization (x, y, theta)
        io.sendPoseEstimate(x, y, theta);

        // Sleep remaining time
        r.sleep();
    }

    return 0;
}
```

This will publish your pose estimate for visualization. It does not affect anything in your own code. You should now see your robot properly in Rviz.

Simulating localization

To more accurately test your system for the final challenge you may want to use sendPose in the simulator as well. In that case don't forget to add the following line to your simulators config file:

```
{
        ...
        "provide_internal_pose": false,
        ...
}
```

This will ensure the simulator is not sending the true pose for visualization. (like we mentioned, we cheat when visualizing the simulator)

Send Path

In the exercises for global navigation you may have seen the function `io.sendPath()`. Similar to `io.sendPose` this function is for visualization only. It publishes the global path of the robot in map frame. You cannot see this path unless you have fixed frame set to "map" in

Rviz.
Use the function like this

```
#include <emc/io.h>
#include <emc/rate.h>

#include <vector>
#include <cmath>

int main()
{
    // Create IO object, which will initialize the io layer
    emc::IO io;

    // Create Rate object, which will help using keeping the loop at a fixed
frequency
    emc::Rate r(10);

    // Loop while we are properly connected
    while(io.ok())
    {
        // create exampe path
        double radius = 5;
        double dtheta = 0.3;
        std::vector<std::vector<double>> path;
        for (int i=0; i<10; i++)
        {
            double theta = i* dtheta;
            std::vector<double> point;
            double x = cos(theta)*radius;
            double y = sin(theta)*radius;
            point.push_back(x);
            point.push_back(y);
            path.push_back(point);
        }
        // Send a the path for visualization(path, color{red, green, blue})
        io.sendPath(path, {0.0, 0.0, 1.0});
        // Sleep remaining time
        r.sleep();
    }

    return 0;
}
```