SmartSoft Tutorial Lesson 1: Getting Started with the VMware Installation

M. Sc. Dennis Stampfer
M. Sc. Andreas Steck
Prof. Dr. Christian Schlegel

Servicerobotik Ulm
University of Applied Sciences Ulm

http://smart-robotics.sourceforge.net/
http://www.servicerobotik-ulm.de
http://www.youtube.com/user/roboticsathsulm

Hochschule Ulm

SmartSoft Tutorial Lesson 1: What is this lesson about?

- we guide you through the first steps of setting SmartSoft and its development tools into operation
  - the effort is fairly low since we provide a VMware installation of all required packages
  - due to the VMware image, there is no need to modify the current settings of your computer
  - due to the VMware image, there is no danger to mess up something and to get stuck with a broken configuration since you can always go back to our original VMware image
  - you can use the VMware image on your preferred operating system
- we guide you through first examples running a Pioneer 3DX robot either in simulation or in real-world
  - you get immediate feedback on the power of SmartSoft and its development tools
  - you get all the hints of where to find more information, documentation, reusable components etc.
- Example 1: Simple Laser Obstacle Avoid
  - simple reactive and collision-free motion of a robot with a laser ranger
  - simple example to get hands-on experience with the Toolchain
- Example 2: Navigation Task (Mapping, Path Planning, etc.)
  - complex set of off-the-shelf software components for navigation
  - example to illustrate how you can compose complex systems out of prebuild components

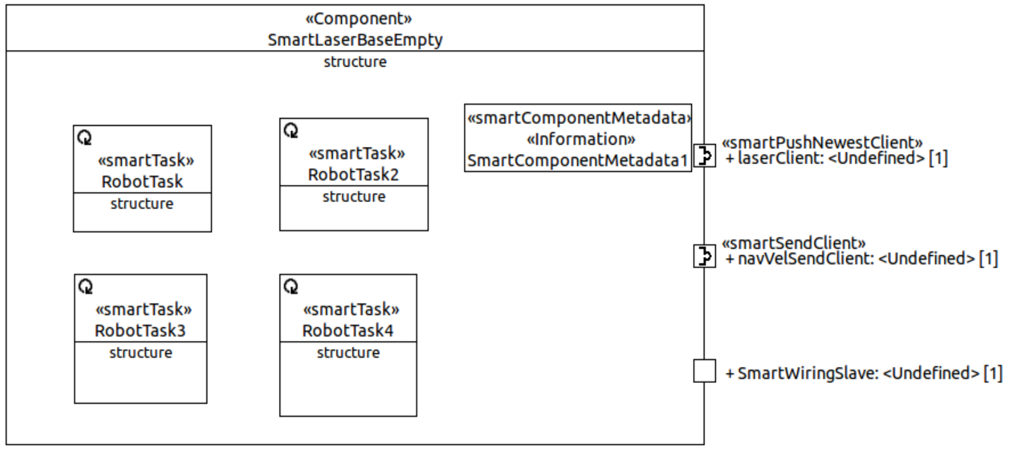
Hochschule Ulm

SmartSoft Tutorial Lesson 1: What skills do you acquire in this lesson?

- Outline of Tutorial Lesson 1
  - Introduction
    - Part 1: The VMware Image
  - Example 1: Smart Laser Obstacle Avoid
    - Part 2: Laser Obstacle Avoid / Simulator
    - Part 3: Laser Obstacle Avoid / Real-World
    - Part 4: Laser Obstacle Avoid / Build Project
  - Example 2: Navigation Task
    - Part 5: Navigation Task / Simulator
    - Part 6: Navigation Task / Real-World
  - Exercises:
    - Part 7: Smart Laser Base Empty
    - Part 8: Exercises

SmartSoft Tutorial Lesson 1: What is SmartSoft?

- SmartSoft is a component based approach for robotics software based on communication patterns as core of a robotics component model.
  - The SmartSoft component model is based on a sound meta-model that provides the ground for model-driven software development.
  - The SmartSoft Toolchain is based on the Eclipse Modeling Project.
- SmartSoft assists the component developer in building software components that adhere to a software component model for robotics.
- SmartSoft assists the application builder in composing complex robotics applications out of off-the-shelf robotics software components.



A Smartsoft component model with two services and four tasks.

Hochschule Ulm

SmartSoft Tutorial Lesson 1:
Part 1: The VMware Image

System Requirements:

- VMware Player >= 6.0
- > 8GB RAM
- > 20GB free disk space

System Information:

- VMware Image based on Ubuntu LTS 12.04. 64bit
- user name / password: smartsoft / smartsoft
- sudo password if needed: smartsoft

How to download and unpack the VMware image:

- download the latest VMware image (VMware Image) and store it in a directory in your home folder:
  - http://sourceforge.net/projects/smart-robotics/files/SmartSoft-VMware
- unpack the file
  - Linux: e.g. by typing ***tar -xjvf <filename>***
  - Windows: e.g. use “7zip”

SmartSoft Tutorial Lesson 1: Part 1: The VMware Image

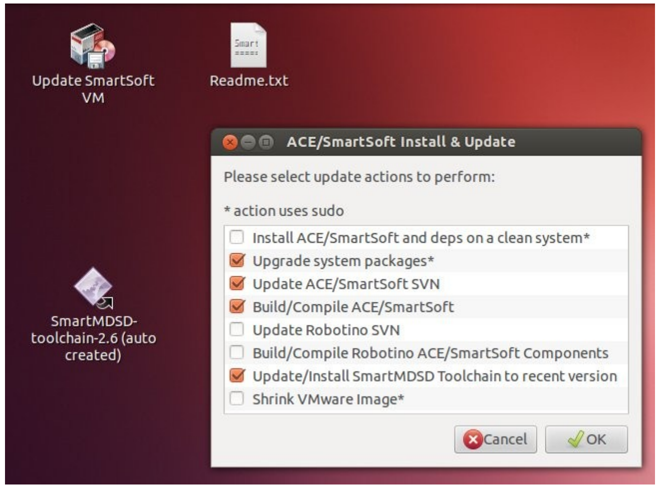
How to start the VMware image:

- start the file `Ubuntu12.04 64bit.vmx` in the VMware Player
  - Linux:
    - shell: `vmplayer Ubuntu12.04 64bit.vmx`
    - GUI:
      - start VMware player
      - “file” / “open a virtual machine” and select the above file
  - Windows: double-click the above file
- now, the VMware player starts the prepared virtual machine
  - the virtual machine appears as a virtual computer inside your computer
  - in our case, this virtual computer is run by Ubuntu
  - we provide a “ready-to-use” installation of the SmartSoft Toolchain in this virtual computer

SmartSoft Tutorial Lesson 1:
Part 1: The VMware Image

How to update and start the SmartSoft Toolchain:

- It is recommended to update your svn repository and the installed SmartMDSD Toolchain. Therefore double-click the “Update SmartSoft VM” icon in the GUI of the Ubuntu Linux in the virtual machine and choose OK.
- To start the Toolchain double-click the “SmartMDSD Toolchain” icon
  - now a dialog to select the workspace shows up
  - please select ***/home/smartsoft/workspace/ws-smartsoft-ace-v2***
- please always properly shutdown or suspend the VM as you do with regular computers!


Update SmartSoft VM
Readme.txt
SmartMDSD-toolchain-2.6 (auto created)
ACE/SmartSoft Install & Update
Cancel
OK

Please select update actions to perform:

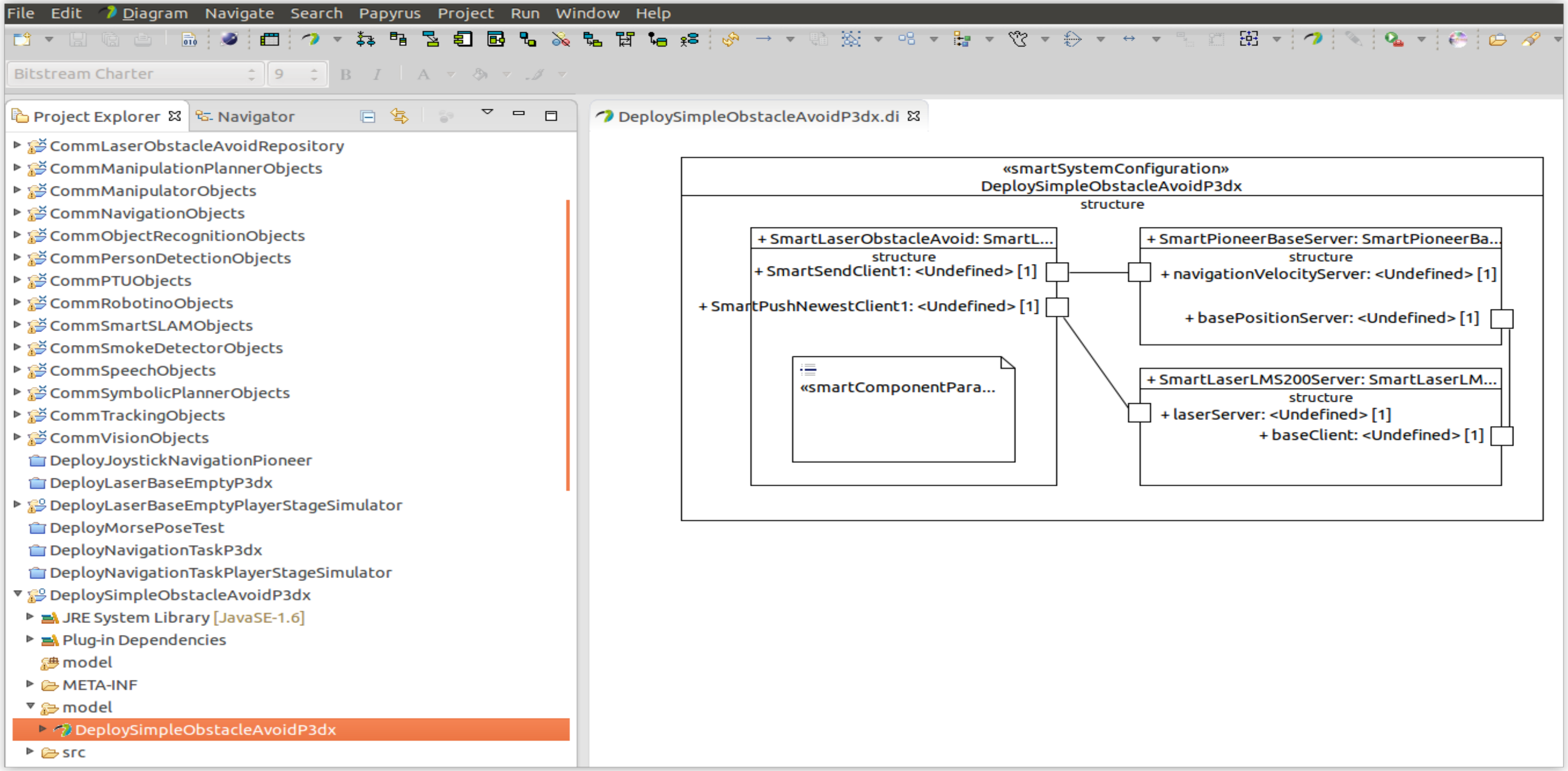
* action uses sudo

- [ ] Install ACE/SmartSoft and deps on a clean system*
- [x] Upgrade system packages*
- [x] Update ACE/SmartSoft SVN
- [x] Build/Compile ACE/SmartSoft
- [ ] Update Robotino SVN
- [ ] Build/Compile Robotino ACE/SmartSoft Components
- [x] Update/Install SmartMDSD Toolchain to recent version
- [ ] Shrink VMware Image*

Hochschule Ulm

SmartSoft Tutorial Lesson 1:
Part 1: The VMware Image

The Ubuntu desktop and SmartMDSD Toolchain:


File Edit Diagram Navigate Search Papyrus Project Run Window Help
Bitstream Charter
9
Project Explorer
Navigator
DeploySimpleObstacleAvoidP3dx.di

CommLaserObstacleAvoidRepository
CommManipulationPlannerObjects
CommManipulatorObjects
CommNavigationObjects
CommObjectRecognitionObjects
CommPersonDetectionObjects
CommPTUObjects
CommRobotinoObjects
CommSmartSLAMObjects
CommSmokeDetectorObjects
CommSpeechObjects
CommSymbolicPlannerObjects
CommTrackingObjects
CommVisionObjects
DeployJoystickNavigationPioneer
DeployLaserBaseEmptyP3dx
DeployLaserBaseEmptyPlayerStageSimulator
DeployMorsePoseTest
DeployNavigationTaskP3dx
DeployNavigationTaskPlayerStageSimulator
DeploySimpleObstacleAvoidP3dx
JRE System Library [JavaSE-1.6]
Plug-in Dependencies
model
META-INF
model
DeploySimpleObstacleAvoidP3dx
src

«smartSystemConfiguration»
DeploySimpleObstacleAvoidP3dx
structure

+ SmartLaserObstacleAvoid: SmartL...
structure
+ SmartSendClient1: <Undefined> [1]
+ SmartPushNewestClient1: <Undefined> [1]
«smartComponentPara...

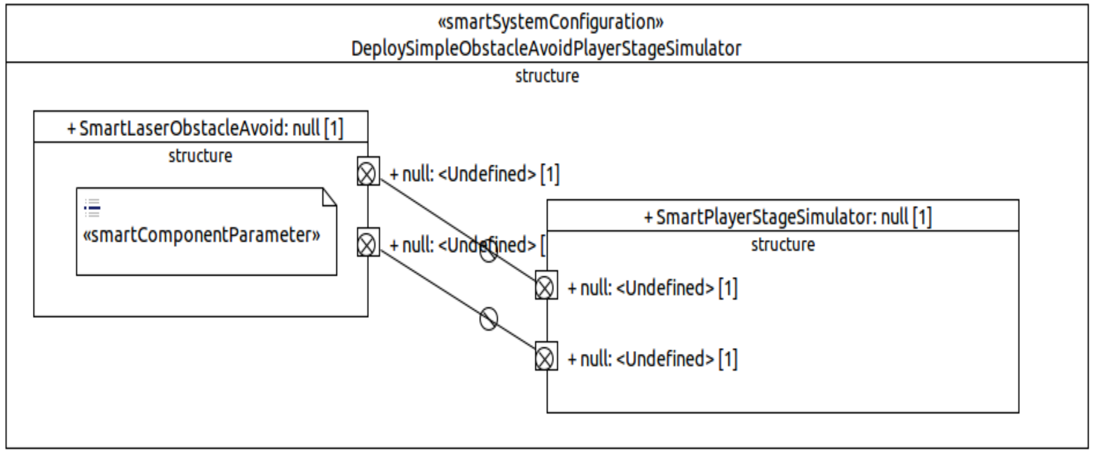
+ SmartPioneerBaseServer: SmartPioneerBa...
structure
+ navigationVelocityServer: <Undefined> [1]
+ basePositionServer: <Undefined> [1]

+ SmartLaserLMS200Server: SmartLaserLM...
structure
+ laserServer: <Undefined> [1]
+ baseClient: <Undefined> [1]

Hochschule Ulm

SmartSoft Tutorial Lesson 1:
Part 2: Laser Obstacle Avoid / Simulator

- Simple Laser Obstacle Avoid / Simulator
  - simple reactive and collision-free motion of a robot with a laser ranger
  - simple example to get hands-on experience with the Toolchain
- You will learn
  - how to start a readily available set of components using the simulator
- Description of the Example:
  - SmartPlayerStageSimulator:
    - software component which provides the Player/Stage 2D simulator
    - simulates laser ranger and Pioneer P3DX platform
  - SmartLaserObstacleAvoid:
    - software component which implements a simple reactive obstacle avoidance
    - periodically receives laser range scans, calculates steering commands and forwards these to the P3DX platform (simulator) for execution


«smartSystemConfiguration»
DeploySimpleObstacleAvoidPlayerStageSimulator
structure
+ SmartLaserObstacleAvoid: null [1]
structure
«smartComponentParameter»
+ null: <Undefined> [1]
+ null: <Undefined> [
+ SmartPlayerStageSimulator: null [1]
structure
+ null: <Undefined> [1]
+ null: <Undefined> [1]

Hochschule Ulm

SmartSoft Tutorial Lesson 1: Part 2: Laser Obstacle Avoid / Simulator

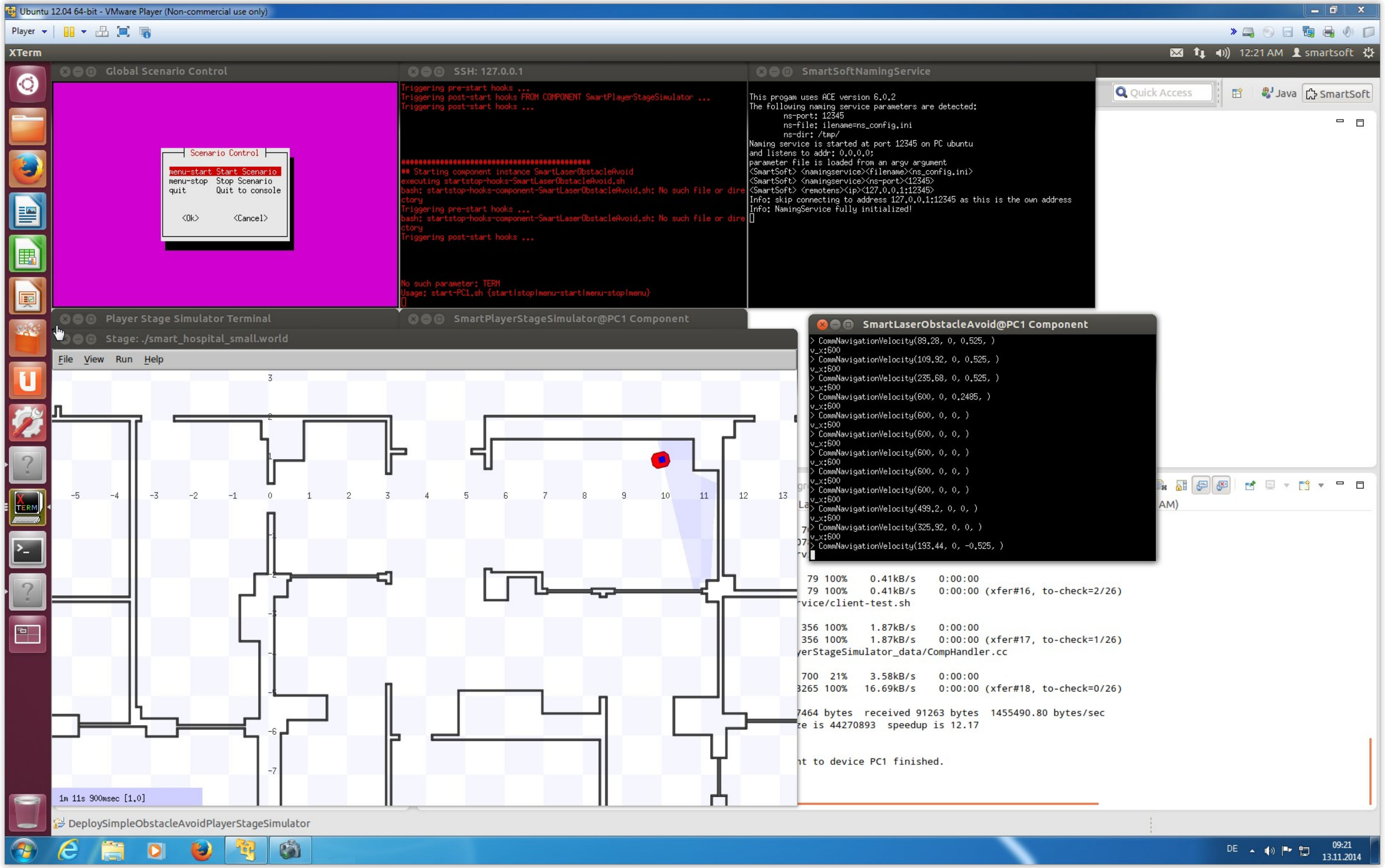
- Simple Laser Obstacle Avoid / Simulator
  - Open the project DeploySimpleObstacleAvoidPlayerStageSimulator and all referenced projects
  - The components are started by right mouse click on the deployment and choose “Deploy + Run SmartMDSD Deployment”. If the Error “FILES ARE MISSING FROM THE DEPLOYMENT” occurs please compile all referenced projects first.
    - (The startup procedure is automatically generated)
    - Terminal (Global Scenario Control) is opened automatically
      - » *Choose menu-start to start scenario*
        two further terminals are opened automatically
      - » *Choose a world in the SSH-terminal and press enter*
  - now, all components and the simulator are being started
  - next, you will see the robot moving and the scan of its laser ranger within a GUI of the simulator as illustrated on the next slide

  - you can move the robot by dragging it (left mouse button)
  - you can rotate the robot by the mouse clicking the right button
  - you can zoom in / zoom with the mouse wheel
  - you can move the map by dragging it (left mouse button)

Hochschule Ulm

SmartSoft Tutorial Lesson 1:
Part 2: Laser Obstacle Avoid / Simulator

- Simple Laser Obstacle Avoid / Simulator
  - The simple Laser Obstacle Avoid in action



Ubuntu 12.04 64-bit - VMware Player (Non-commercial use only)
XTerm
Global Scenario Control
Scenario Control
menu-start Start Scenario
menu-stop Stop Scenario
quit Quit to console
<Ok>
<Cancel>
Player Stage Simulator Terminal
Stage: ./smart_hospital_small.world
File
View
Run
Help
1m 11s 900msec [1.0]
SmartPlayerStageSimulator@PC1 Component
DeploySimpleObstacleAvoidPlayerStageSimulator

SSH: 127.0.0.1

```
Triggering pre-start hooks ...
Triggering post-start hooks FROM COMPONENT SmartPlayerStageSimulator ...
Triggering post-start hooks ...




############################################
## Starting component instance SmartLaserObstacleAvoid
executing startstop-hooks-SmartLaserObstacleAvoid.sh
bash: startstop-hooks-component-SmartLaserObstacleAvoid.sh: No such file or dire
ctory
Triggering pre-start hooks ...
bash: startstop-hooks-component-SmartLaserObstacleAvoid.sh: No such file or dire
ctory
Triggering post-start hooks ...




No such parameter: TERM
Usage: start-PC1.sh {start|stop|menu-start|menu-stop|menu}
```

SmartSoftNamingService

```
This progam uses ACE version 6.0.2
The following naming service parameters are detected:
        ns-port: 12345
        ns-file: ilename=ns_config.ini
        ns-dir: /tmp/
Naming service is started at port 12345 on PC ubuntu
and listens to addr: 0.0.0.0;
parameter file is loaded from an argv argument
<SmartSoft> <namingservice><filename><ns_config.ini>
<SmartSoft> <namingservice><ns-port><12345>
<SmartSoft> <remotens><ip><127.0.0.1:12345>
Info: skip connecting to address 127.0.0.1:12345 as this is the own address
Info: NamingService fully initialized!
```

SmartLaserObstacleAvoid@PC1 Component

```
> CommNavigationVelocity(89.28, 0, 0.525, )
v_x:600
> CommNavigationVelocity(109.92, 0, 0.525, )
v_x:600
> CommNavigationVelocity(235.68, 0, 0.525, )
v_x:600
> CommNavigationVelocity(600, 0, 0.2485, )
v_x:600
> CommNavigationVelocity(600, 0, 0, )
v_x:600
> CommNavigationVelocity(600, 0, 0, )
v_x:600
> CommNavigationVelocity(600, 0, 0, )
v_x:600
> CommNavigationVelocity(600, 0, 0, )
v_x:600
> CommNavigationVelocity(600, 0, 0, )
v_x:600
> CommNavigationVelocity(499.2, 0, 0, )
v_x:600
> CommNavigationVelocity(325.92, 0, 0, )
v_x:600
> CommNavigationVelocity(193.44, 0, -0.525, )
```

```
 79 100%    0.41kB/s    0:00:00
 79 100%    0.41kB/s    0:00:00 (xfer#16, to-check=2/26)
rvice/client-test.sh

356 100%    1.87kB/s    0:00:00
356 100%    1.87kB/s    0:00:00 (xfer#17, to-check=1/26)
yerStageSimulator_data/CompHandler.cc

700  21%    3.58kB/s    0:00:00
3265 100%   16.69kB/s    0:00:00 (xfer#18, to-check=0/26)

7464 bytes  received 91263 bytes  1455490.80 bytes/sec
ze is 44270893  speedup is 12.17


nt to device PC1 finished.
```

Hochschule Ulm

SmartSoft Tutorial Lesson 1:
Part 2: Laser Obstacle Avoid / Simulator

to stop the simulation and all components, use the following command in the terminal from which you started all the components (Global Scenario Control):

- » *Choose menu-stop to stop scenario (all terminals except the Global Scenario Control get closed automatically do not close manual!)*
- » *Choose quit to quit scenario*

SmartSoft Tutorial Lesson 1:
Part 3: Laser Obstacle Avoid / Real-World

Important Hint:

If you do have

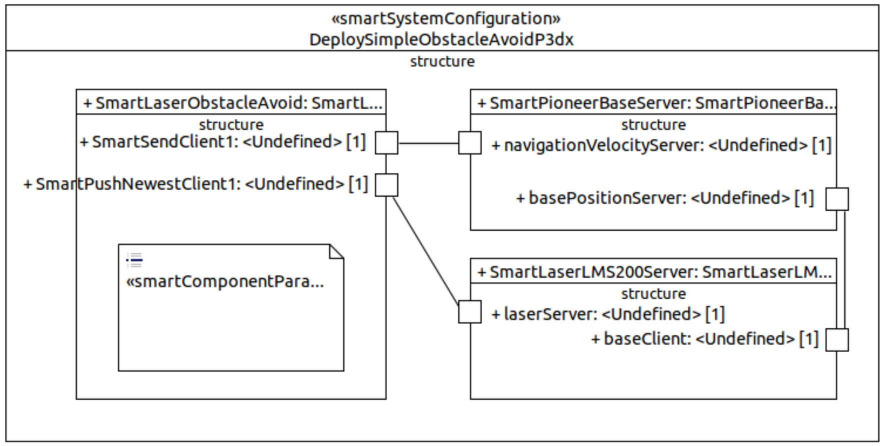
- a Pioneer 3DX platform with a SICK LMS200 laser ranger
- installed Ubuntu LTS 12.04. 64bit on the computer on-board the Pioneer

then you can continue with this part.

Otherwise, please read through this part to gain further insights but continue with the hands-on exercises in Part 4 which again use the simulation.

SmartSoft Tutorial Lesson 1: Part 3: Laser Obstacle Avoid / Real-World

- Simple Laser Obstacle Avoid / Real-World Deployment
  - same as part 2 but now on a real-world platform
- You will learn
  - how to reuse the “laserObstacleAvoid” component without any modifications
  - how to deploy the components on a real robot via the Toolchain
  - how to start the components on a real robot
- Description of the Example:
  - the SmartPlayerStageSimulator is replaced by two components:
    - smartPioneerBaseServer:
      - » provides services that interface to the Pioneer P3DX platform
    - smartLaserLMS200Server
      - » provides services for laser range scans of a SICK LMS200
  - the laserObstacleAvoid-Component (even its binary) is completely unchanged and now simply connects to services not anymore provided by the simulator but by the above two components for a real robot and a real laser ranger.


«smartSystemConfiguration»
DeploySimpleObstacleAvoidP3dx
structure
+ SmartLaserObstacleAvoid: SmartL...
structure
+ SmartSendClient1: <Undefined> [1]
+ SmartPushNewestClient1: <Undefined> [1]
«smartComponentPara...
+ SmartPioneerBaseServer: SmartPioneerBa...
structure
+ navigationVelocityServer: <Undefined> [1]
+ basePositionServer: <Undefined> [1]
+ SmartLaserLMS200Server: SmartLaserLM...
structure
+ laserServer: <Undefined> [1]
+ baseClient: <Undefined> [1]

Hochschule Ulm

SmartSoft Tutorial Lesson 1:
Part 3: Laser Obstacle Avoid / Real-World

- Simple Laser Obstacle Avoid / Real-World
  - First, the components need to be deployed onto the target robot platform
    - run the deployment step from within the Toolchain
      - » right button click on the project “DeploySimpleObstacleAvoidP3dx” (this contains the deployment for the real-robot and not the simulator anymore)
      - » select “Deploy + Run SmartMDSD Deployment”

Hochschule Ulm

SmartSoft Tutorial Lesson 1:
Part 3: Laser Obstacle Avoid / Real-World

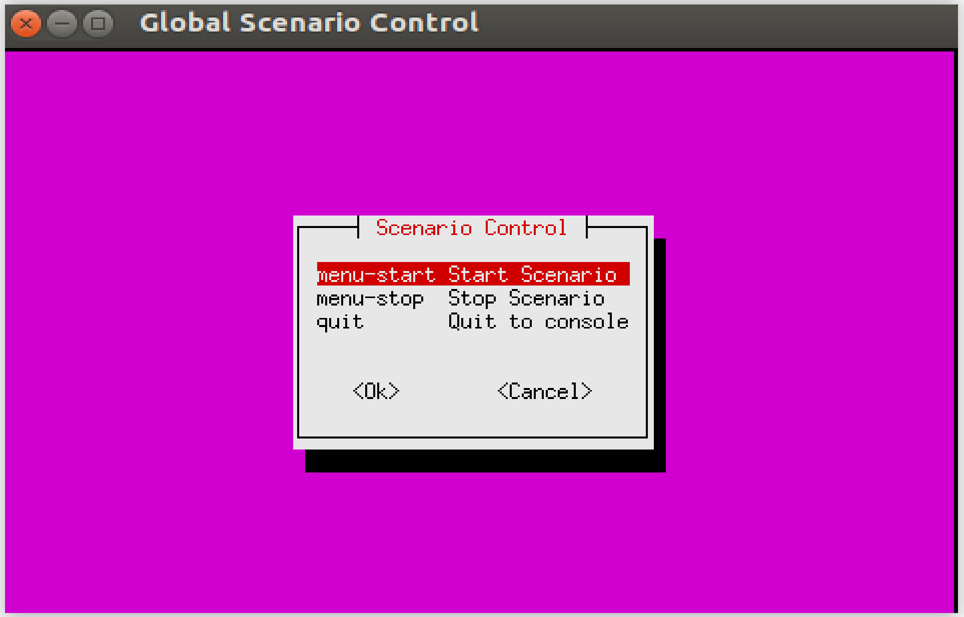
- Simple Laser Obstacle Avoid / start and stop

  – now, the deployment script is being executed which deploys the components on the target robot platform
    » the target platform (IP, username, directory) are parameters within the deployment model (tagged values)
    » these parameters can be modified in the diagram

  - If communication was successful same as on simulation terminal get opened automatically
    » ***Choose menu-start to start scenario***

  – now, all components are being started and the robot starts moving around

  - Third, the components can be stopped by ***Choose menu-stop to stop robot***



```
Scenario Control

menu-start  Start Scenario
menu-stop   Stop Scenario
quit        Quit to console

   <Ok>          <Cancel>
```

Hochschule Ulm

SmartSoft Tutorial Lesson 1:

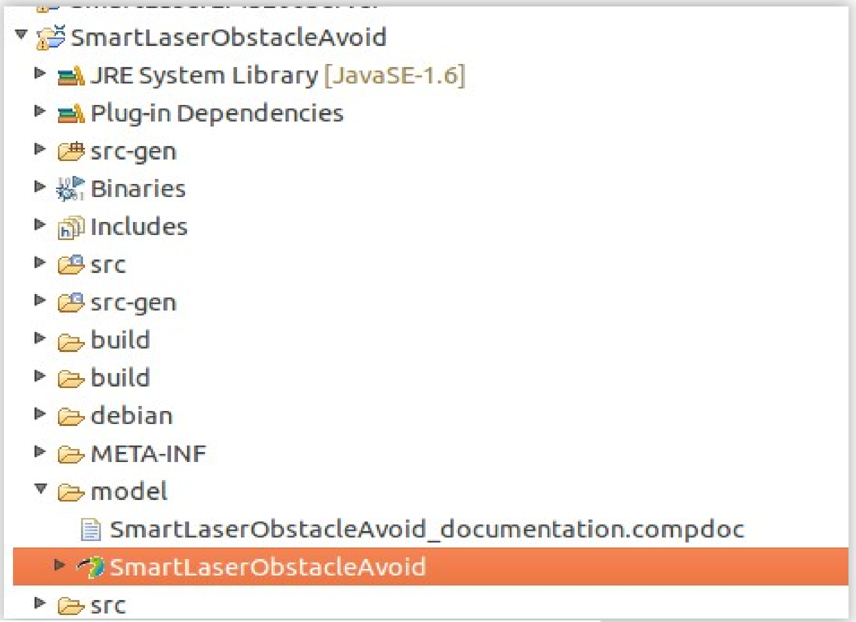
Part 4: Laser Obstacle Avoid / Build Project

- Smart Laser Obstacle Avoid / Build Project
  - we will inspect the laser obstacle avoid component (used in part 2 / part 3) at the source code level and the model level
  - we will run the build process (model transformation, code generation, compilation, linking)
- You will learn
  - how the laser obstacle avoid algorithm works
  - about the file structure of the laser obstacle avoid example component in the Toolchain
  - how to modify a component at the source code level
  - how to rebuild a component after source code level modifications
- Description of the Example:
  - (see part 2)

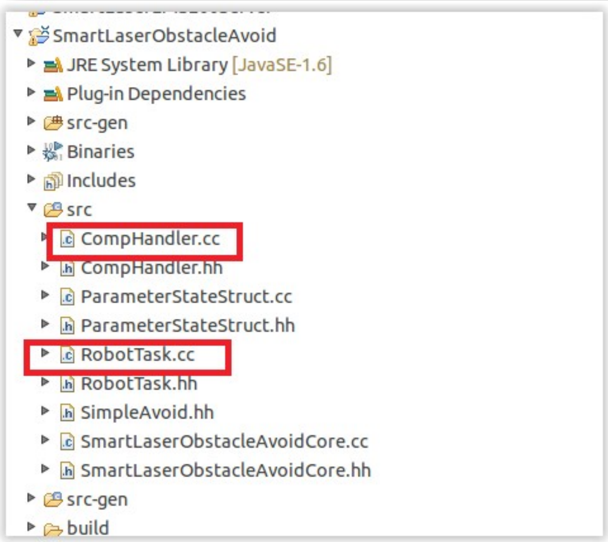
Hochschule Ulm

SmartSoft Tutorial Lesson 1: Part 4: Laser Obstacle Avoid / Build Project

- Smart Laser Obstacle Avoid / Build Project
  - first, open the platform independent component model
    - navigate to the path model/ in project "SmartLaserObstacleAvoid"
    - double click the model file: SmartLaserObstacleAvoid
    - the model of the component with two client data ports (for laser and navigation commands) is shown along with a task that implements the obstacle avoidance algorithm
  - second, open the source code of the RobotTask
    - open the file src/RobotTask.cc by double click
    - the method RobotTask::on execute() comprises the implementation of the algorithm
      - » it first receives laser scans,
      - » then calculates new navigation commands and
      - » finally sends them to robot base
    - you may now modify the source code and enhance the algorithm



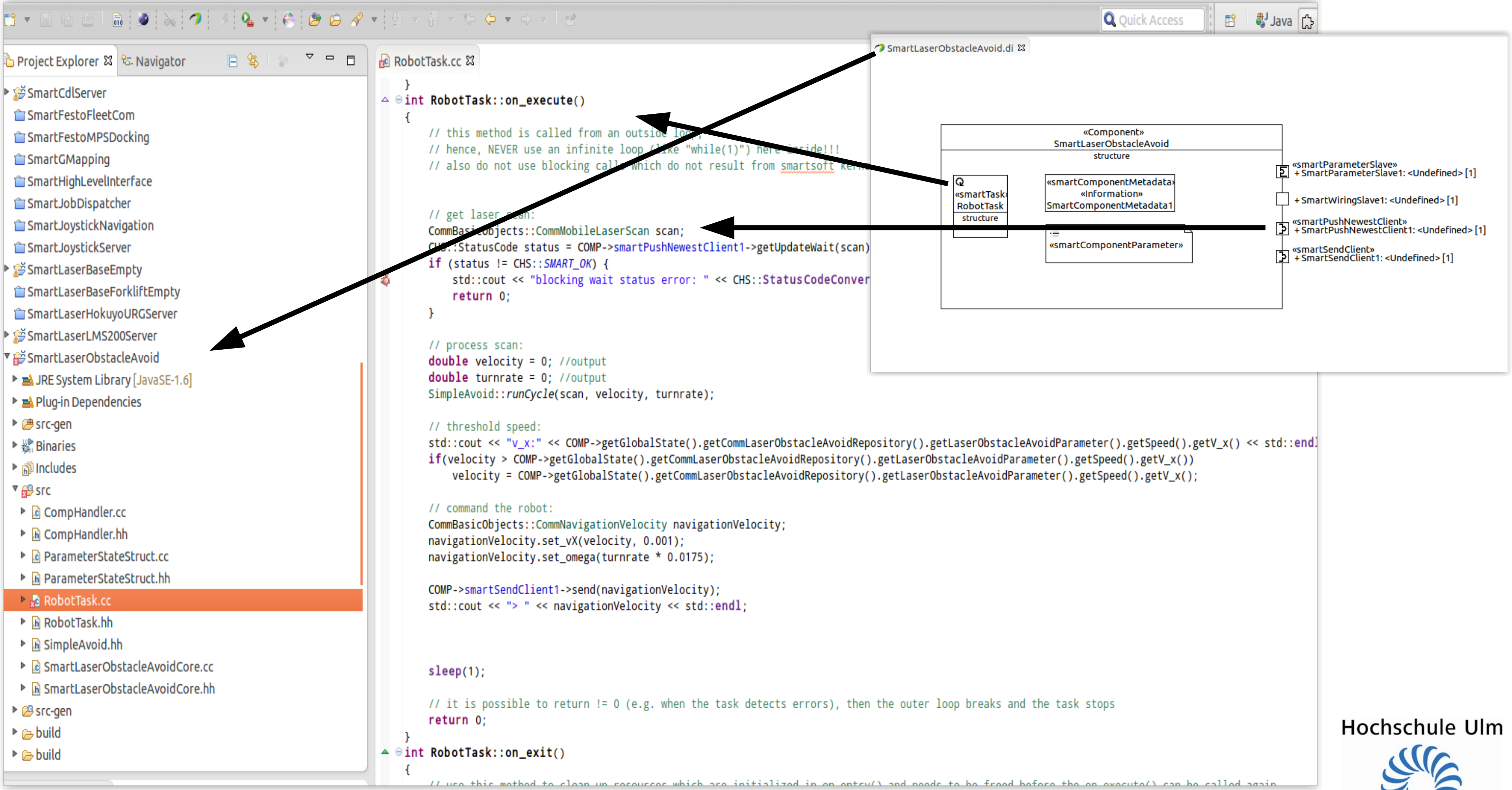
SmartLaserObstacleAvoid
- JRE System Library [JavaSE-1.6]
- Plug-in Dependencies
- src-gen
- Binaries
- Includes
- src
- src-gen
- build
- build
- debian
- META-INF
- model
  - SmartLaserObstacleAvoid_documentation.compdoc
  - SmartLaserObstacleAvoid
- src



SmartLaserObstacleAvoid
- JRE System Library [JavaSE-1.6]
- Plug-in Dependencies
- src-gen
- Binaries
- Includes
- src
  - CompHandler.cc
  - CompHandler.hh
  - ParameterStateStruct.cc
  - ParameterStateStruct.hh
  - RobotTask.cc
  - RobotTask.hh
  - SimpleAvoid.hh
  - SmartLaserObstacleAvoidCore.cc
  - SmartLaserObstacleAvoidCore.hh
- src-gen
- build

Hochschule Ulm

SmartSoft Tutorial Lesson 1:
Part 4: Laser Obstacle Avoid / Build Project

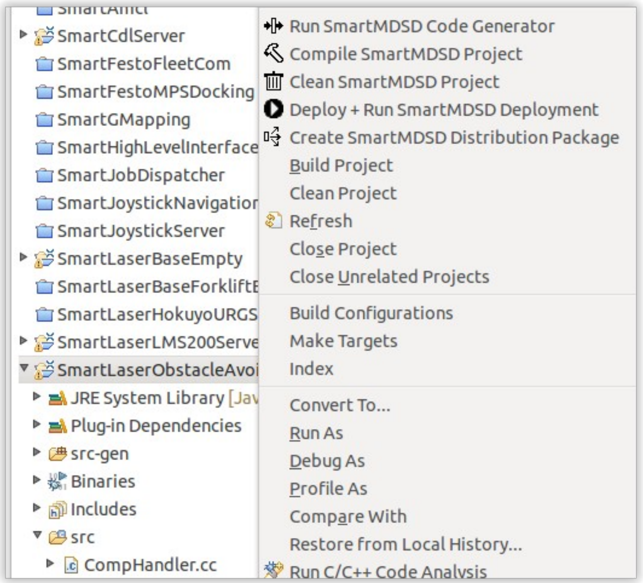


```
}
int RobotTask::on_execute()
{
    // this method is called from an outside loop,
    // hence, NEVER use an infinite loop (like "while(1)") here inside!!!
    // also do not use blocking calls which do not result from smartsoft kern


    // get laser scan:
    CommBasicObjects::CommMobileLaserScan scan;
    CHS::StatusCode status = COMP->smartPushNewestClient1->getUpdateWait(scan)
    if (status != CHS::SMART_OK) {
        std::cout << "blocking wait status error: " << CHS::StatusCodeConver
        return 0;
    }

    // process scan:
    double velocity = 0; //output
    double turnrate = 0; //output
    SimpleAvoid::runCycle(scan, velocity, turnrate);

    // threshold speed:
    std::cout << "v_x:" << COMP->getGlobalState().getCommLaserObstacleAvoidRepository().getLaserObstacleAvoidParameter().getSpeed().getV_x() << std::endl
    if(velocity > COMP->getGlobalState().getCommLaserObstacleAvoidRepository().getLaserObstacleAvoidParameter().getSpeed().getV_x())
        velocity = COMP->getGlobalState().getCommLaserObstacleAvoidRepository().getLaserObstacleAvoidParameter().getSpeed().getV_x();

    // command the robot:
    CommBasicObjects::CommNavigationVelocity navigationVelocity;
    navigationVelocity.set_vX(velocity, 0.001);
    navigationVelocity.set_omega(turnrate * 0.0175);

    COMP->smartSendClient1->send(navigationVelocity);
    std::cout << "> " << navigationVelocity << std::endl;



    sleep(1);

    // it is possible to return != 0 (e.g. when the task detects errors), then the outer loop breaks and the task stops
    return 0;
}
int RobotTask::on_exit()
{
    // use this method to clean up resources which are initialized in on_entry() and needs to be freed before the on_execute() can be called again
```

Hochschule Ulm

SmartSoft Tutorial Lesson 1: Part 4: Laser Obstacle Avoid / Build Project

- Simple Laser Obstacle Avoid / Build Project
  - finally, run the MDSD process to build a new binary
    - Right click on the component project and choose “Run SmartMDSD Code Generator”
      - » this will trigger the MDSD process to generate source code
    - Build (compile) the project via “compile SmartMDSD Project” by right-clicking on the project
  - you can now start the example in the very same way as described in part 2 and 3
    - the deployment / start scripts remain the same but now use your just built binary.

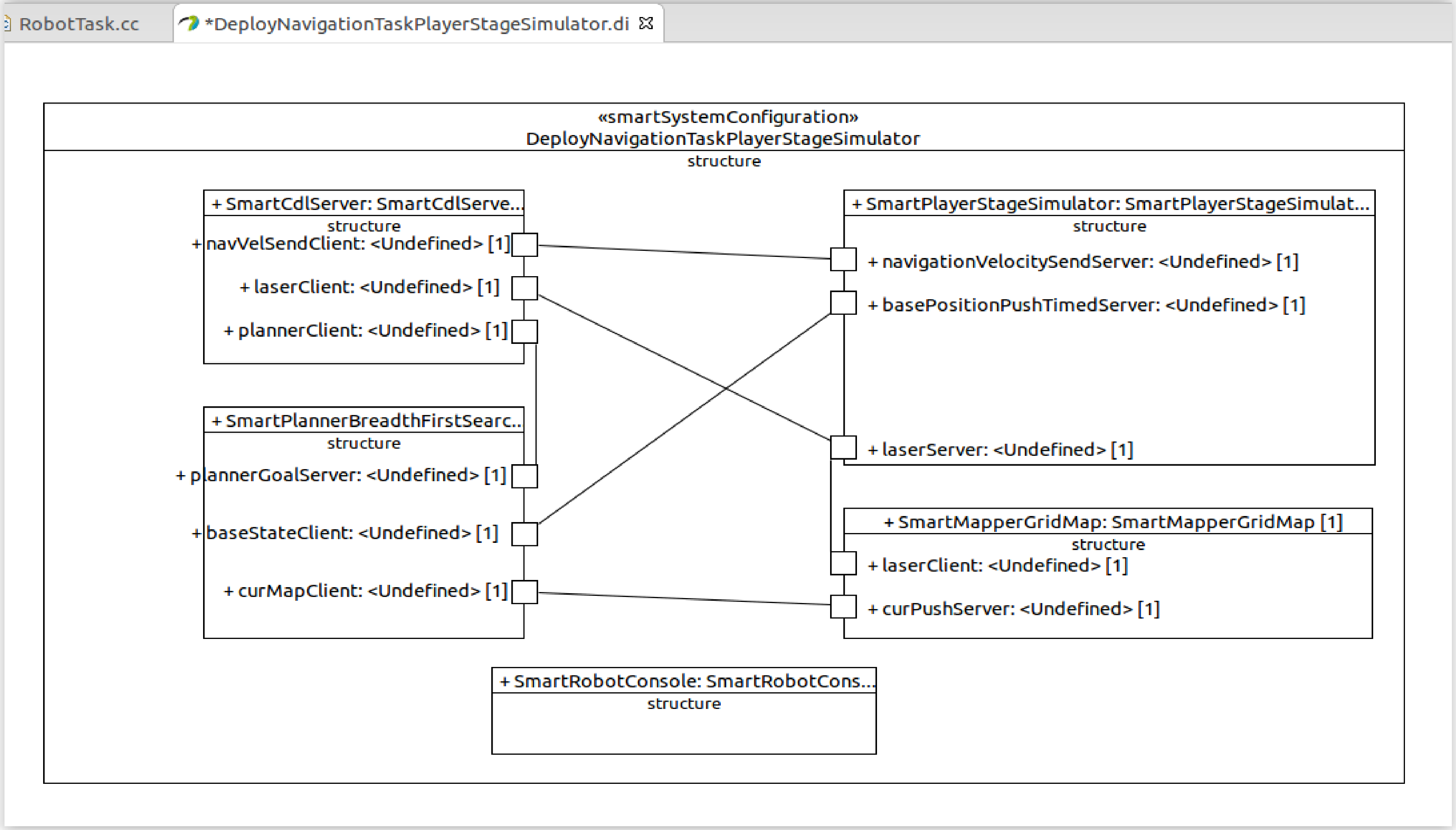


SmartSoft Tutorial Lesson 1:
Part 5: Navigation Task / Simulator

- Navigation Task (Mapping, Path Planning, etc.) / Simulator
  - navigate to x/y coordinates using a path planner (for goal-directed motion), a map building component (to take into account changes in the environment and obstacles on the fly), and a motion control mechanism (takes into account kinematics, dynamics and shape of the robot for high speed motions)
- You will learn
  - how you can use a complex and prebuilt deployment providing a navigation stack
  - how components interact with each other based on services and how these are wired
- Description of the Example:
  - SmartPlayerStageSimulator (see Parts 2 and 3)
  - SmartMapperGridMap
    - » probabilistic 2D grid map (long term map, current map): maps the environment
  - SmartPlannerBreadthFirstSearch
    - » breadth first search algorithm to find path in map towards goal
    - » sends intermediate waypoints to the CDL motion execution component
  - SmartCdlServer
    - » drives towards intermediate waypoints from planner taking into account robot kinematics, dynamics, shape
  - SmartRobotConsole
    - » simple user interface for the above components

SmartSoft Tutorial Lesson 1:
Part 5: Navigation Task / Simulator

- Navigation Task (Mapping, Path Planning, etc.) / Simulator



RobotTask.cc
*DeployNavigationTaskPlayerStageSimulator.di
«smartSystemConfiguration»
DeployNavigationTaskPlayerStageSimulator
structure
+ SmartCdlServer: SmartCdlServe...
structure
+ navVelSendClient: <Undefined> [1]
+ laserClient: <Undefined> [1]
+ plannerClient: <Undefined> [1]
+ SmartPlannerBreadthFirstSearc...
structure
+ plannerGoalServer: <Undefined> [1]
+ baseStateClient: <Undefined> [1]
+ curMapClient: <Undefined> [1]
+ SmartPlayerStageSimulator: SmartPlayerStageSimulat...
structure
+ navigationVelocitySendServer: <Undefined> [1]
+ basePositionPushTimedServer: <Undefined> [1]
+ laserServer: <Undefined> [1]
+ SmartMapperGridMap: SmartMapperGridMap [1]
structure
+ laserClient: <Undefined> [1]
+ curPushServer: <Undefined> [1]
+ SmartRobotConsole: SmartRobotCons...
structure

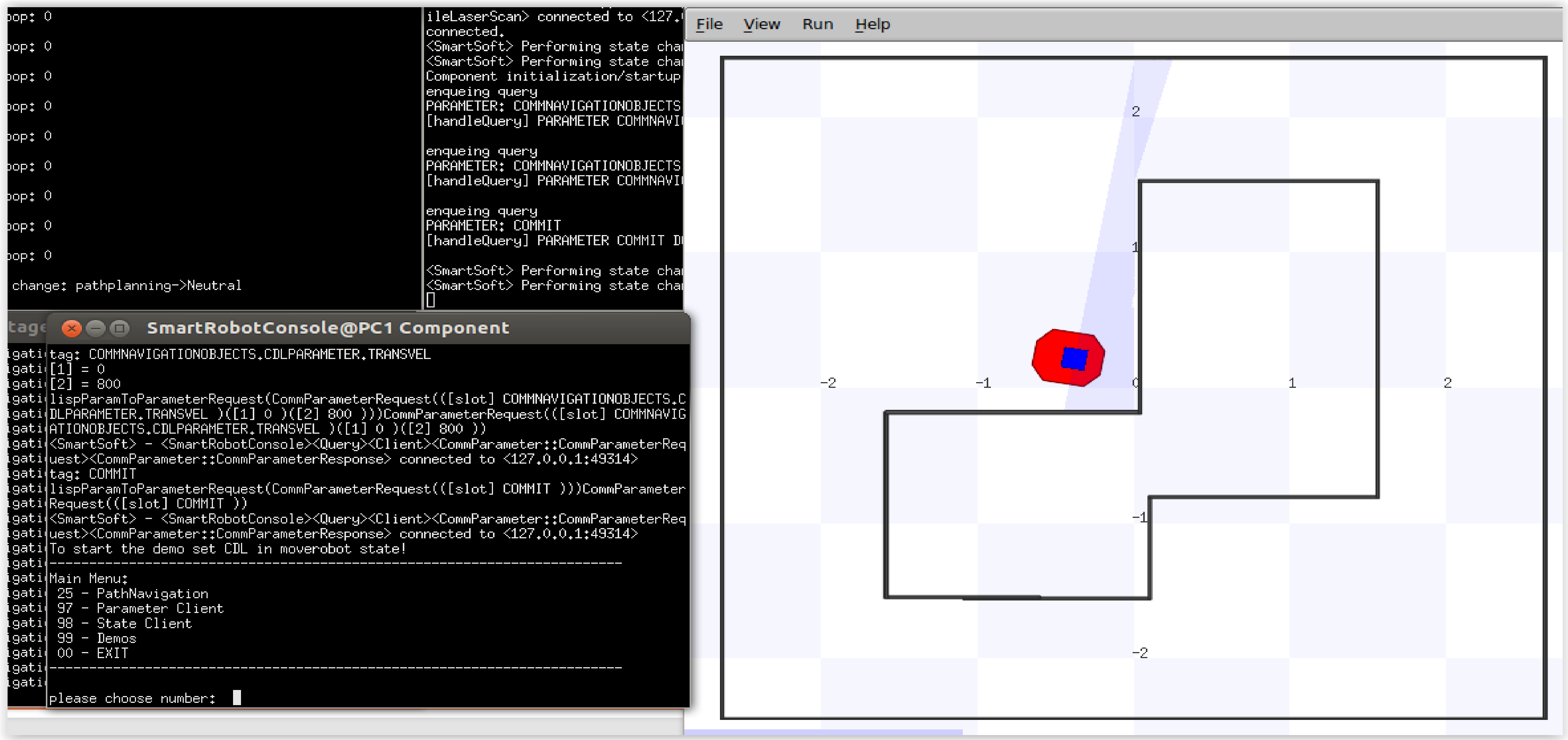
Hochschule Ulm

SmartSoft Tutorial Lesson 1:
Part 5: Navigation Task / Simulator

- Navigation Task (Mapping, Path Planning, etc.) / Simulator
  - starting the navigation task
    - execute the deployment script in the deployment project DeployNavigationTaskPlayerStageSimulator (see part 2, just different project / script name)
    - now, the simulator and all the components are being started
  - operating the navigation task
    - look for the terminal which contains the robot console
    - choose "99 - Demos" (type 99 and hit enter) to get the list of the demos:
    - now choose "Demo 2 Planner-CDL GOTO"
    - now enter the goal coordinates in brackets measured in millimetre from the initial position (x:0 y:0) of the robot: e.g. "(1000)(0)" means a goal 1m ahead of the robot
    - the robot moves in the simulator to the entered position immediately
    - to move the robot to another goal choose "Enter Goal? (y/n): " y and enter the coordinates, e.g. (0)(0) to move to initial position
    - the demo stops by entering "n" if you are asked whether you want to enter an goal.

SmartSoft Tutorial Lesson 1: Part 5: Navigation Task / Simulator

- Navigation Task (Mapping, Path Planning, etc.) / Simulator



```
pop: 0

pop: 0

pop: 0

pop: 0

pop: 0

pop: 0

pop: 0

pop: 0

pop: 0

 change: pathplanning->Neutral
```

```
ileLaserScan> connected to <127.
connected.
<SmartSoft> Performing state cha
<SmartSoft> Performing state cha
Component initialization/startup
enqueing query
PARAMETER: COMMNAVIGATIONOBJECTS
[handleQuery] PARAMETER COMMNAVI

enqueing query
PARAMETER: COMMNAVIGATIONOBJECTS
[handleQuery] PARAMETER COMMNAVI

enqueing query
PARAMETER: COMMIT
[handleQuery] PARAMETER COMMIT D

<SmartSoft> Performing state cha
<SmartSoft> Performing state cha
```

SmartRobotConsole@PC1 Component

```
tag: COMMNAVIGATIONOBJECTS.CDLPARAMETER.TRANSVEL
[1] = 0
[2] = 800
lispParamToParameterRequest(CommParameterRequest(([slot] COMMNAVIGATIONOBJECTS.C
DLPARAMETER.TRANSVEL )([1] 0 )([2] 800 )))CommParameterRequest(([slot] COMMNAVIG
ATIONOBJECTS.CDLPARAMETER.TRANSVEL )([1] 0 )([2] 800 ))
<SmartSoft> - <SmartRobotConsole><Query><Client><CommParameter::CommParameterReq
uest><CommParameter::CommParameterResponse> connected to <127.0.0.1:49314>
tag: COMMIT
lispParamToParameterRequest(CommParameterRequest(([slot] COMMIT )))CommParameter
Request(([slot] COMMIT ))
<SmartSoft> - <SmartRobotConsole><Query><Client><CommParameter::CommParameterReq
uest><CommParameter::CommParameterResponse> connected to <127.0.0.1:49314>
To start the demo set CDL in moverobot state!
------------------------------------------------------------------------
Main Menu:
 25 - PathNavigation
 97 - Parameter Client
 98 - State Client
 99 - Demos
 00 - EXIT
------------------------------------------------------------------------

please choose number:
```

Hochschule Ulm

SmartSoft Tutorial Lesson 1:
Part 6: Navigation Task / Real-World

Important Hint:

If you do have

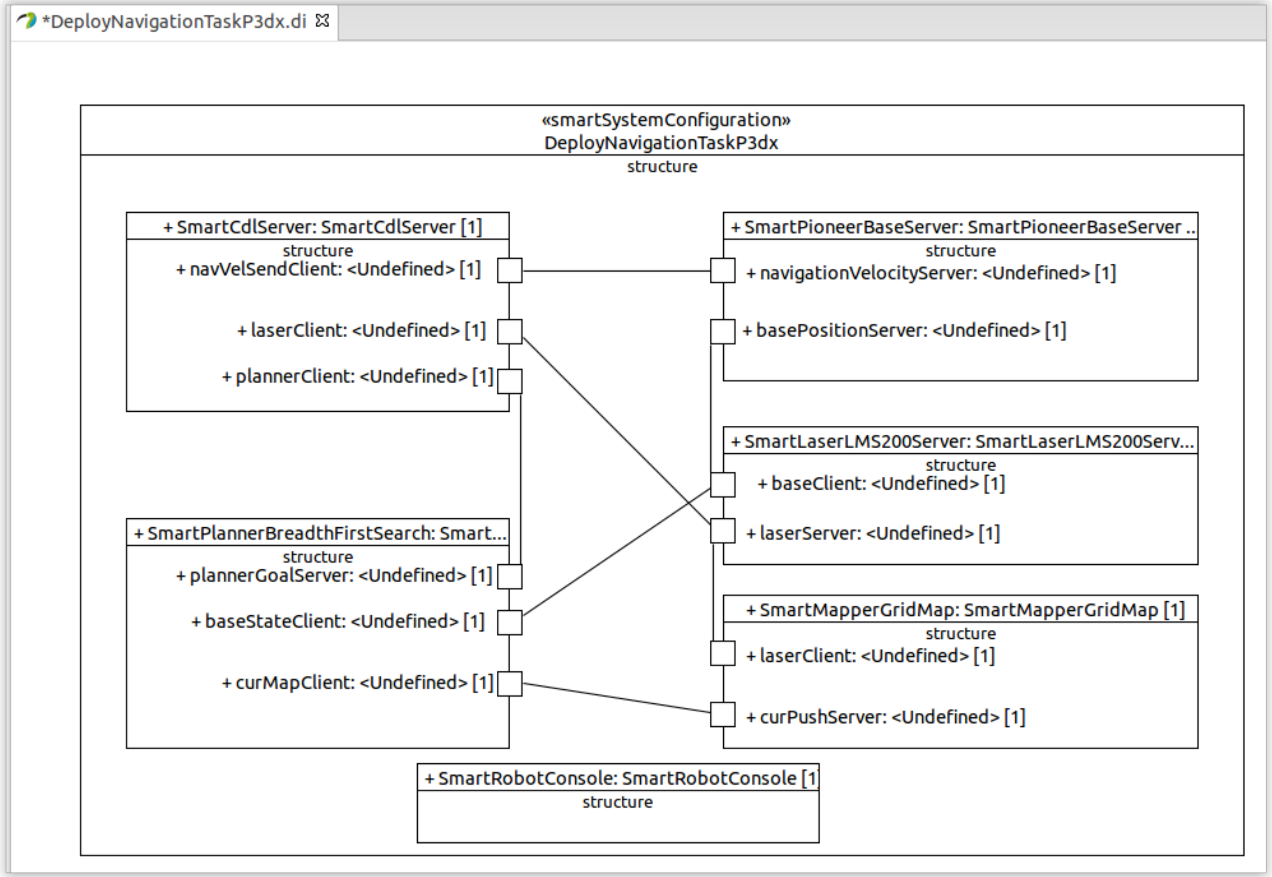
- a Pioneer 3DX platform with a SICK LMS200 laser ranger
- installed Ubuntu LTS 12.04 64 bit on the computer on-board the Pioneer

then you can continue with this part.

Otherwise, please read through this part to gain further insights but continue with the hands-on exercises in Part 7 which again use the simulation.

SmartSoft Tutorial Lesson 1:
Part 6: Navigation Task / Real-World

- Navigation Task / Real-World
  - refer to part 3 (just different project / script name) to run the example on a real robot
  - the prepared deployment project is called DeployNavTask
  - refer to the previous part on instructions how to operate the navigation task


*DeployNavigationTaskP3dx.di
«smartSystemConfiguration»
DeployNavigationTaskP3dx
structure
+ SmartCdlServer: SmartCdlServer [1]
structure
+ navVelSendClient: <Undefined> [1]
+ laserClient: <Undefined> [1]
+ plannerClient: <Undefined> [1]
+ SmartPioneerBaseServer: SmartPioneerBaseServer ..
structure
+ navigationVelocityServer: <Undefined> [1]
+ basePositionServer: <Undefined> [1]
+ SmartLaserLMS200Server: SmartLaserLMS200Serv...
structure
+ baseClient: <Undefined> [1]
+ laserServer: <Undefined> [1]
+ SmartPlannerBreadthFirstSearch: Smart...
structure
+ plannerGoalServer: <Undefined> [1]
+ baseStateClient: <Undefined> [1]
+ curMapClient: <Undefined> [1]
+ SmartMapperGridMap: SmartMapperGridMap [1]
structure
+ laserClient: <Undefined> [1]
+ curPushServer: <Undefined> [1]
+ SmartRobotConsole: SmartRobotConsole [1
structure

SmartSoft Tutorial Lesson 1: Part 6: Navigation Task / Real-World

- Navigation Task (Mapping, Path Planning, etc.) / Real-World Deployment
  - same as part 5 but now on a real-world platform
- You will learn
  - how to make reuse of the navigation stack on a real robot
  - how to deploy the components on a real robot via the Toolchain
  - how to start the components on a real robot
- Description of the Example:
  - the SmartPlayerStageSimulator is replaced by SmartPioneerBaseServer and SmartLaserLMS200Server (see part 3).
  - all other components that are reused from the previous example (Navigation Task Simulation in part 5), even the binaries, are completely unchanged and now simply connect to services not anymore provided by the simulator but by the above two components for a real-robot and a real laser ranger.

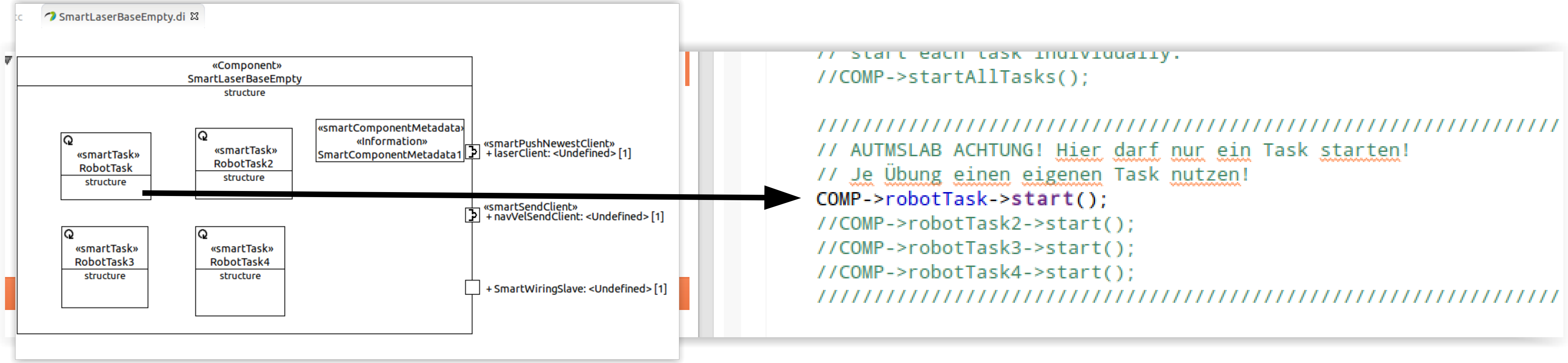
SmartSoft Tutorial Lesson 1:
Part 7: Laser Base Empty

- Laser Base Empty
  - we provide an empty component hull similar to the laser obstacle avoid component (see part 2/3)
  - instead of only one RobotTask, the component hull comprises four empty tasks:
    - these tasks are used within / prepared for the next parts of this tutorial (of course, there is no restriction in the number of tasks in a SmartSoft component)
    - at each time only one task should be active
    - you can use one thread per exercise
- You will learn
  - how to implement your own algorithms within a predefined component hull
  - how to access the services of the component hull by your algorithm
- Description of the Example:
  - SmartLaserBaseEmpty:
    - software component hull where you can implement the exercises of the next parts of this tutorial
  - components of part 2 and 3 are used

Hochschule Ulm

SmartSoft Tutorial Lesson 1: Part 7: Laser Base Empty

- Laser Base Empty: The component hull
  - open the model of the component in the project named SmartLaserBaseEmpty
  - the component comprises four tasks. The file src/RobotTask.cc contains a simple skeleton:
    - » the task receives a laser scan
    - » it commands the robot to drive forward for three seconds
    - » it commands the robot to stop
  - you can start from this skeleton and implement your solution to the following exercises
  - be sure to only start one task. The tasks are started at the end of the file src/CompHandler.cc:

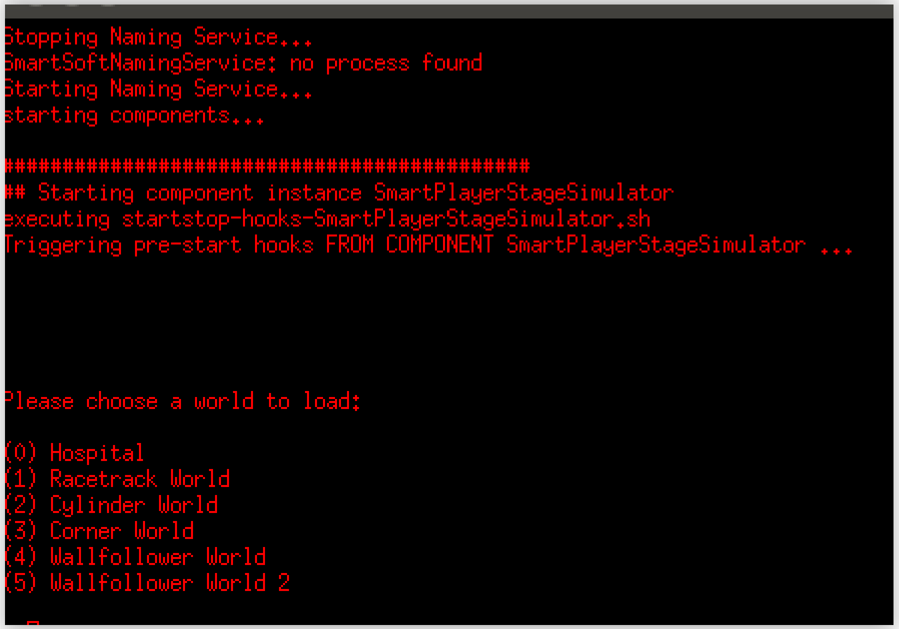


```
// start each task individually:
//COMP->startAllTasks();

/////////////////////////////////////////////////////////////////
// AUTMSLAB ACHTUNG! Hier darf nur ein Task starten!
// Je Übung einen eigenen Task nutzen!
COMP->robotTask->start();
//COMP->robotTask2->start();
//COMP->robotTask3->start();
//COMP->robotTask4->start();
/////////////////////////////////////////////////////////////////
```

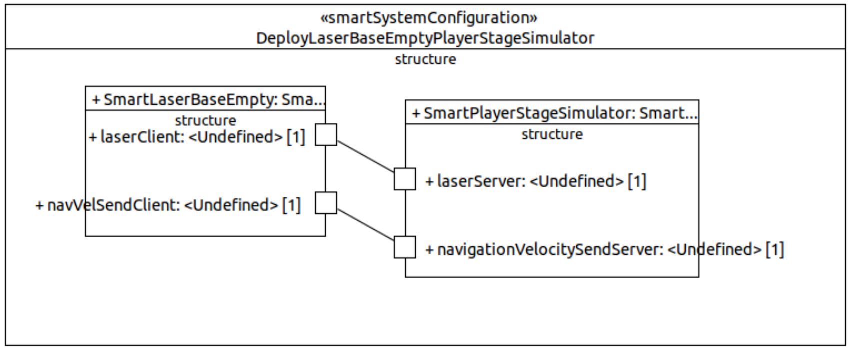
Hochschule Ulm

SmartSoft Tutorial Lesson 1: Part 7: Laser Base Empty

- Laser Base Empty: How to test your implementation
  - first, test your algorithm in the simulator
    - » build your project as described in part 4
    - » to start the simulator, refer to part 2 (replace DeployLaserObstacleAvoid by DeployLaserBaseEmpty)
    - » choose an environment (by typing the corresponding number) when the simulator starts
  - second, if you have a Pioneer P3DX:
    - » refer to part 3 and run your exercises on the real robot
    - » (replace DeployLaserObstacleAvoidRobot by DeployLaserBaseEmptyRobot)



```
Stopping Naming Service...
SmartSoftNamingService: no process found
Starting Naming Service...
starting components...

##############################################
## Starting component instance SmartPlayerStageSimulator
executing startstop-hooks-SmartPlayerStageSimulator.sh
Triggering pre-start hooks FROM COMPONENT SmartPlayerStageSimulator ...



Please choose a world to load:

(0) Hospital
(1) Racetrack World
(2) Cylinder World
(3) Corner World
(4) Wallfollower World
(5) Wallfollower World 2
```

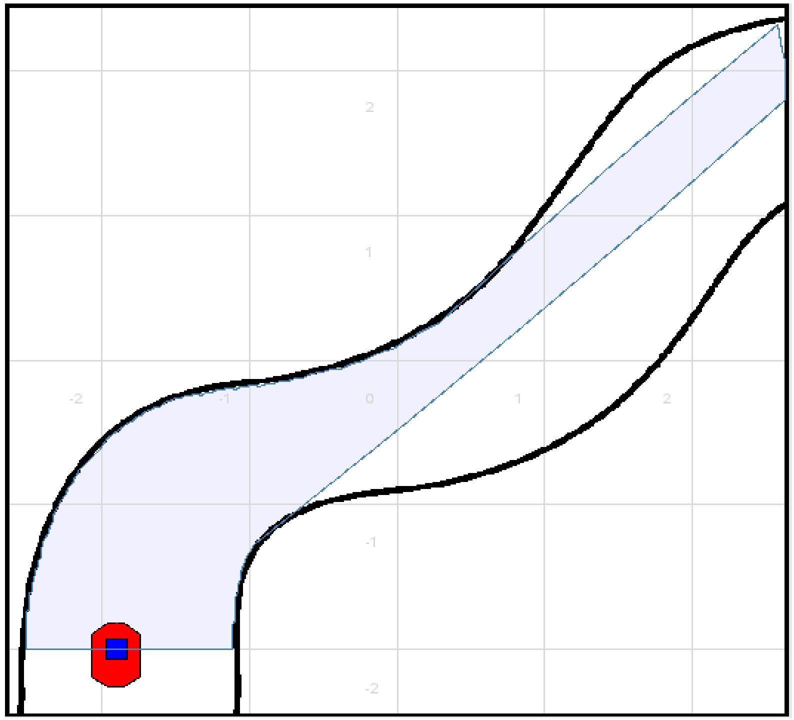

«smartSystemConfiguration»
DeployLaserBaseEmptyPlayerStageSimulator
structure
+ SmartLaserBaseEmpty: Sma...
structure
+ laserClient: <Undefined> [1]
+ navVelSendClient: <Undefined> [1]
+ SmartPlayerStageSimulator: Smart...
structure
+ laserServer: <Undefined> [1]
+ navigationVelocitySendServer: <Undefined> [1]

SmartSoft Tutorial Lesson 1: Part 8: Exercises

- Exercises
  - implement the following exercises in the previously described component hull
- You will learn
  - how to navigate your robot through a simple scenario
  - how to use the laser ranger
  - how to command the robot to move with an v- / omega-interface
  - how to detect and approach a cylinder
- Description of the Example:
  - the following exercises are individual worlds to load in the simulator
  - environment ground plane corresponds to ZAFH laboratory
    - 5.29 x 4.90m
    - grid in the simulator: 1m

SmartSoft Tutorial Lesson 1: Part 8: Exercises

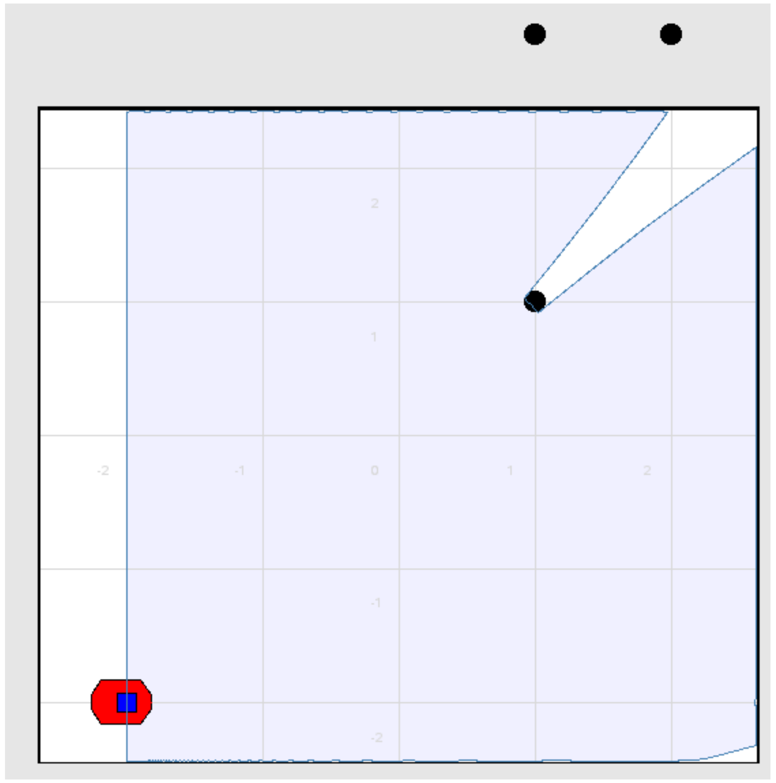
- Exercise 1: Racetrack
  - use the laser ranger to calculate robot movements for save navigation through a tube / racetrack
  - scenario: Racetrack World (1)
  - hint: try to drive in the middle of the tube by keeping the same distance to the left and right wall.



Hochschule Ulm

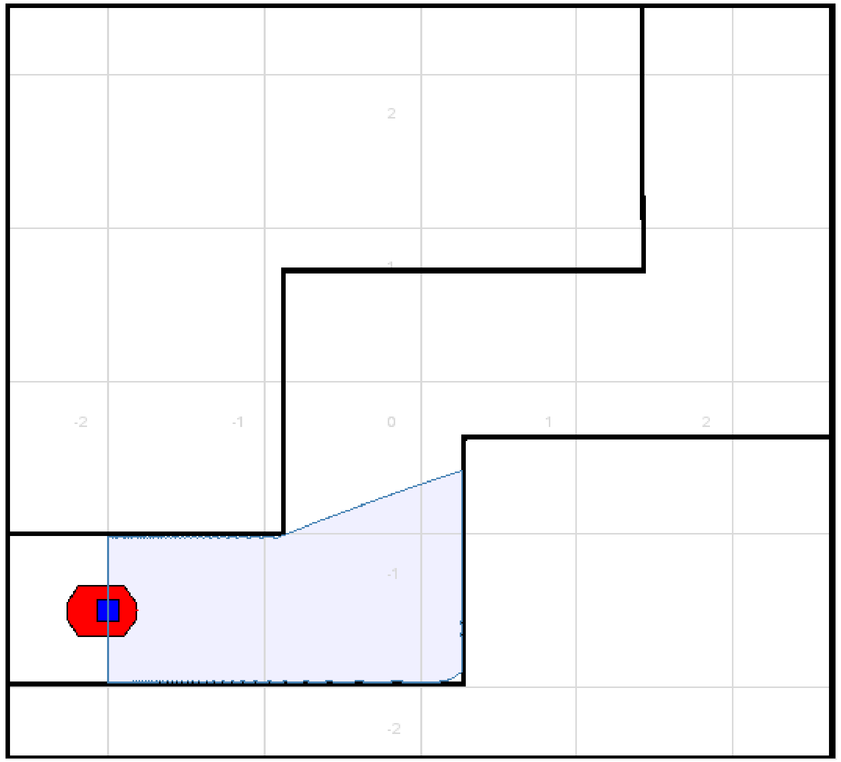
SmartSoft Tutorial Lesson 1: Part 8: Exercises

- Exercise 2: Approach cylinder
  - use the laser ranger to find the cylinder
  - calculate robot movements to navigate to the cylinder and stop in front of it
  - you can move the cylinder by dragging it with the mouse (left button)
    - » you can even move more cylinders inside the scenario
    - » cylinder diameter: 15cm
  - scenario: Cylinder World (2)
  - hint: The laser scan points will make a depth jump at the left and right edge of the cylinder.



SmartSoft Tutorial Lesson 1:
Part 8: Exercises

- Exercise 3: Corner World
  - use the laser ranger to calculate movements to the end of the simple labyrinth and to move back.
  - scenario: Corner World (3)

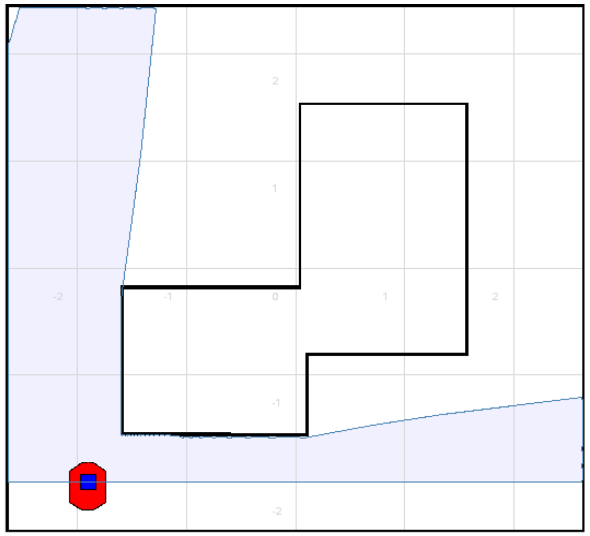


Hochschule Ulm

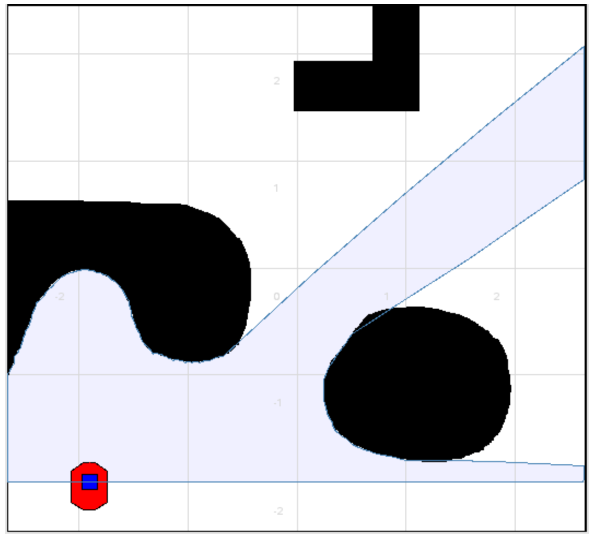
SmartSoft Tutorial Lesson 1:
Part 8: Exercises

- Exercise 4: Wall following
  - use the laser ranger to calculate movements to follow the left / right wall
  - test your algorithm using both worlds
  - scenarios
    - » Wallfollwer World (4)
    - » Wallfollower World 2 (5)

Wallfollower World:



Wallfollower World 2:



Hochschule Ulm

SmartSoft Tutorial Lesson 1: Where to find more information?

- SmartSoft framework and SmartSoft MDSD Toolchain:
  - http://servicerobotik-ulm.de
  - http://servicerobotik-ulm.de/drupal/doxygen/aceSmartSoft

- Description of available components and communication objects:
  - http://servicerobotik-ulm.de/drupal/doxygen/components_commrep
  - in particular: Components
    - SmartPioneerBaseServer
    - SmartLaserLMS200Server
    - SmartLaserObstacleAvoid
  - in particular: Communication Objects
    - CommMobileLaserScan
    - CommNavigationVelocity

Hochschule Ulm