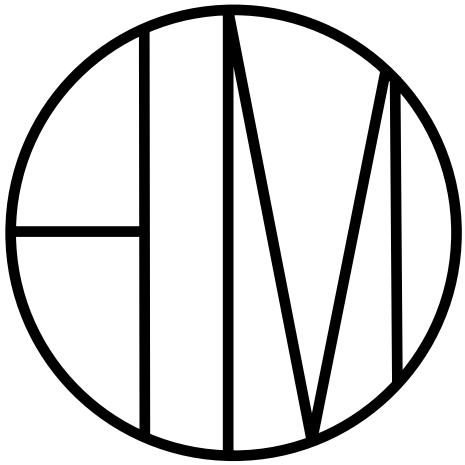
User Manual
LiU Racetrack 2018

LiU Racetrack 2018
Author: Oskar Karlsson
2018-12-17
Version 1.0



Status

Reviewed	Oskar Karlsson	2018-12-13
Approved	Oskar Ljungqvist	2018-12-17

Project identity

2018 HT, LiU Racetrack 2018
Department of Electrical Engineering - Linköping University

Name	**Role**	**Email**
Tomas Busk (TB)	Responsible for LEGO Truck	tombu079@student.liu.se
Mergim Dushku (MD)	Responsible for Migration to ROS	merdu044@student.liu.se
David Forsberg (DF)	Responsible for Migration to Visionen	davfo471@student.liu.se
Marcus Hall (MH)	Responsible for Simulation Enviroment	marha585@student.liu.se
Oskar Karlsson (OK)	Document Manager	oskka906@student.liu.se
Kim Larsson (KL)	Software Architect	kimla207@student.liu.se
Jacob Mourad (JM)	Project Leader	jacmo228@student.liu.se
Sara Nilsson (SN)	Test Manager	sarni339@student.liu.se

Orderer: Oskar Ljungqvist, Linköping University, +46 70 577 18 68, oskar.ljungqvist@liu.se
Course responsible: Daniel Axehill, Linköping University, +46 13 28 40 42, daniel.axehill@liu.se
Advisor: Olov Holmer, Linköping University, +46 13 28 16 17, olov.holmer@liu.se

Contents

Document history

Version	Date	Changes	Performed by	Reviewed by
1.0	2018-12-13	Approved by orderer	Project group	OK

1 Introduction

This document aims to guide users to start and maintain the system components. The user should be able to use this document in order to set up and start the system's software from scratch as this document contains instructions for installation and setup of needed software and initialisation of it.

This user guide is for transfering the old system to ROS and developing a planner and controller for a LEGO truck. To run the old system see the user manual from Racetrack 2017.

Initially you need a computer with the following installed:

- Ubuntu 16.0.4
- ROS Kinetic (**Note:** Newer version than kinetic is **NOT** supported)
- catkin
- Git

Optional: for more information about catkin and ROS read the tutorials in these links respectively

`http://wiki.ros.org/ROS/Tutorials#Beginner_Level`

`http://wiki.ros.org/catkin#Installing_catkin`

2 Software

2.1 Git

Git is easiest installed via the terminal using the command:

```
$ sudo apt-get install git
```

Then the project can be downloaded (pulled) from the git-repository using the commands:

```
$ cd ~/
$ git clone https://gitlab.ida.liu.se/tsrt10_2018/racetrack.git
```

Now you have access to the whole project!

Note: that you must have been given access to the repository by the orderer, or the person in charge of the git repository.

2.2 ROS

This section goes through the steps needed in order to install ROS Kinetic for Ubuntu 16.0.4. For more information, please refer to `http://wiki.ros.org/kinetic/Installation/Ubuntu`. Remember to only run one line of code in the terminal at a time.

To setup your sources.list, enter the following line in the terminal:

```
$ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
```

To set up a key, enter the following line:

```
$ sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 --recv-key 421C365BD9FF1F717815A3895523BAEEB01FA116
```

Now, make sure that your Debian package index is up-to-date by entering the following line:

```
$ sudo apt-get update
```

To install ROS, enter the following line. It is strongly recommend to use a full install of ROS:

```
$ sudo apt-get install ros-kinetic-desktop-full
```

Before ROS can be used, its dependencies need to be set up. Enter the following lines:

```
$ sudo rosdep init
$ rosdep update
```

It is convenient if the ROS environment variables are automatically added to your bash session.

```
$ echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc
$ source ~/.bashrc
```

Now we need a catkin workspace for our project. Catkin is, for most ROS-distributions, included when installing ROS. But to be sure one may run the command:

```
$ sudo apt-get install ros-kinetic-catkin
```

Now we may continue with setting up the catkin workspace.

```
$ source /opt/ros/kinetic/setup.bash
$ mkdir -p ~/catkin_ws/src
$ cd ~/catkin_ws/
$ catkin_make
$ echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc
```

Now, catkin should be installed and the workspace should be built.

2.3 Setup of the Racetrack software

After ROS has been installed and the catkin workspace has been set up it is time to build the ROS projects. The initial steps are to copy the ROS projects from the Git repository into the catkin workspace and build them.

```
$ cp -r ~/racetrack/ros/* ~/catkin_ws/src
$ cd ~/catkin_ws
$ catkin_make
```

The `catkin_make` command may take a few minutes to finish but should not give any errors in the end. The projects should be correctly built and file paths should be correctly set.

3 Usage

During development, the project group used their own laptops in order to implement and execute the different parts of the project. This was because the movement of the stationary Ubuntu-workstation from the project room to the Visionen arena never occurred. These steps have thus not been tested on that Ubuntu-workstation, but they should still work.

3.1 Initially

The first thing to do is to setup the IP-addresses for ROS. The two necessary variables are *ROS_MASTER_URI* which contains the IP-address of the computer running the *roscore*, and *ROS_IP* which contains the IP-address of your own computer, if other than the one running *roscore*. By running the command:

```
$ ifconfig
```

one can find the IP-address in the field *inet addr* (not the 127.0.0.1 address) which should be put as the *ROS_IP*. The address is usually something like 192.168.XX.XX. Do the same for the one computer running *roscore* and enter this as the *ROS_MASTER_URI*. When the IP-addresses are found, open the file ~/.bashrc and add the following lines in the end of the file:

```
export ROS_MASTER_URI=http://(IP-address for roscore-computer):11311
export ROS_IP=(IP-address for own computer)
```

Now run the command:

```
$ source ~/.bashrc
```

And finally, the roscore-computer runs the command:

```
$ roscore
```

This should initialise the ROS-system's so called "core" that keeps track of all the ROS-nodes and topics, with their respective subscribers and publishers. If the ROS-system is to be run with multiple computers running different nodes, each computer needs to set up their respective IP-addresses.

3.2 ROS Packages

Every implemented ROS-node is contained within a package located in the projects git repository under `racetrack/ros`. All that needs to be done for them to work is to follow the steps in section 2.3 then navigate to the package that is to be started using the terminal and finally type:

`$ ./start` This executes a script that launches all ROS-nodes associated with the package and other software needed for them to function properly. Some packages are run using bashscripts which are run by typing the following command:

```
$ .  start_ROSNODENAME
```

where ROSNODENAME is the name of the ROS-node contained in the package you wish to run.

3.3 The GUIs

There are several GUIs implemented during this year that are used for different parts of the project. One of the GUIs is used for debugging and contain plugins for running ROS-nodes as well as monitoring the communication between the nodes that are executed. Another GUI visualises the planned trajectory of the LEGO truck and the trajectory that has been travelled so far. To launch a GUI use one of the following commands:

This launches the GUI for the LEGO truck:

```
$ cd ~/catkin_ws/src/lego_truck_visual
$ ./start
```

This launches the debugging GUI:

```
$ cd ~/catkin_ws/src/debug_visual
$ rqt --perspective-file gui_node_displays.perspective
```

The GUI for the LEGO truck is seen in Figure 1 where the planned trajectory, coloured green, and the travelled trajectory, coloured red, is displayed. The debug-GUI is seen in Figure 2 where a graph that visualises the running ROS-nodes and their communication channels are shown.

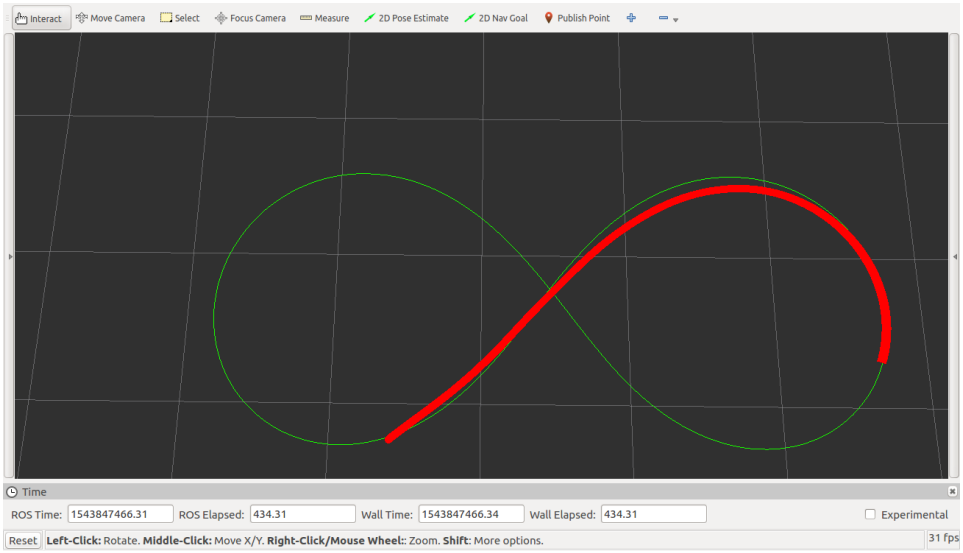


Figure 1: The GUI for the LEGO truck visualising the planned trajectory (green) and the travelled trajectory (red).

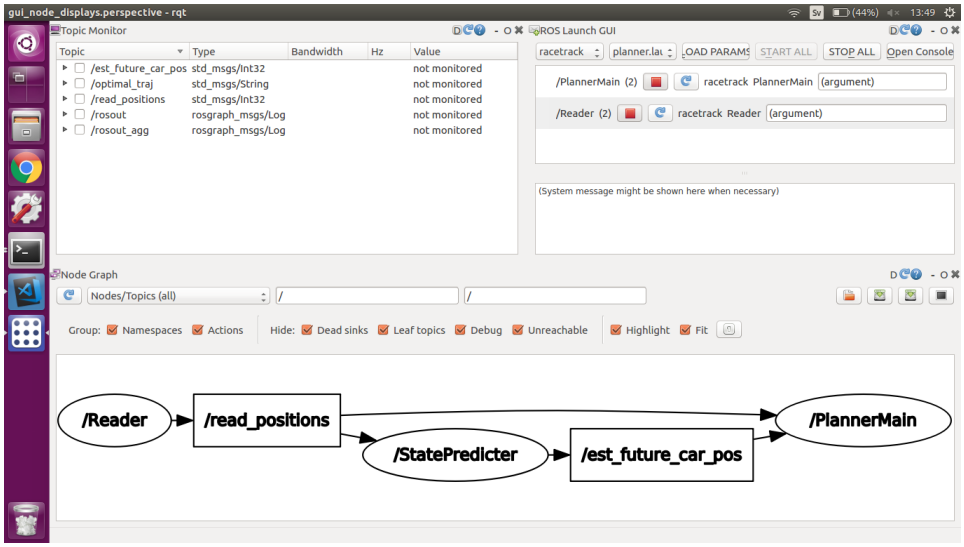


Figure 2: The debug GUI containing the plugins; Topic monitor, ROS launch, Node graph

4 EV3 disk image

To be able to run new code on the LEGO truck using the EV3 a new executable need to be made. To make a new executable the following steps need to be taken:

- Install Docker for your operating system. This is used to simulate the EV3 hardware for faster compilation time. If the compilation is done on the EV3 this would run out of memory before it is complete. Docker may take some time to get acquainted with.
- Download the necessary files from gitlab.
- Compile and make sure no error occur.
- Connect to the EV3 using Bluetooth or wifi.
- Transfer the necessary files to the EV3 and you should be good to go.

4.0.1 Note

Some open-source github libraries have addresses for the input and output ports that don't work. What was found to work is using addresses "in1" and "outA" instead of "ev3-ports:in1" and "ev3-ports:outA" which are used in some libraries. One way of figuring out which addresses are used is to do the following:

```
large_motor m;
std::cout << m.address();
```

This will use the first port it detects a motor at and by looking at the result of the call to address() you can see what the argument should be when using the following code:

```
large_motor m(output_address);
```

where output_address is the result of the call to address().

4.1 Creating a new EV3-disk image

If a new or updated EV3-disk image is needed, the current version can be downloaded from the docker repo and edited before made into an ISO-file using Brickman (https://github.com/ev3dev/brickman) and burnt onto the microSD-card.

5 Running the LEGO truck

To be able to run the LEGO truck in Stora Visionen the steps explained below in Sections 5.1, 5.2, 5.3 need to be made to set up the system.

5.1 Setting up the EV3-unit

Do following steps to set up the LEGO truck:

1. Make sure the EV3 is equipped with fully charged batteries.
2. Make sure the EV3 is placed correctly with the connections *A*-*D* facing backwards.
3. Make sure the driving motor is connected to port C.
4. Make sure the steering motor is connected to port B.

5. Make sure the sensor measuring β_2 between the truck and dolly is connected to port 1.
6. Make sure the sensor measuring β_3 between the dolly and rear trailer is connected to port 2.
7. Press the middle button to start the EV3. Wait until it glows green and shows an IP-adress in the top left corner.

5.2 Setting up Stora Visionen

A few things are needed to setup the positioning system used in Stora Visionen.

1. Make sure QTM is installed on the Windows computer.
2. Download the *Qualisys* project folder from the git repository.
3. Start QTM and select Qualisys as your project folder.
4. If the system hasn't be calibrated earlier in the day perform the calibration setup described under 5.2.1.
5. Start a measurement by pressing the red button, you will be asked to press a physical button to start the measurement.

5.2.1 Calibration of Stora Visionen

Described below is a step by step guide to calibrate the positioning system in Stora Visionen.

1. Place the L-shaped object where the origin is desired to be, preferably in the middle of the room.
2. Go to system setup and select wand calibration.
3. Select Capture/Calibrate and set the calibration time to 200 seconds.
4. Put together the wand and start the calibration. Make sure the calibration has started by looking at the green bar at the bottom of the screen.
5. Calibrate by moving the wand around in smooth movements in the room.
6. If the calibration results in calibration passed you are good to go.

5.3 Setting up the Linux computer

To set up the Linux computer complete the following steps:

1. Make sure the computer is running Ubuntu 16.0.4.
2. Make sure ROS and catkin is installed on the computer.
3. Make sure you have a repository in your root with the folder name *catkin_ws* that is a functioning catkin workspace with a folder named *src* inside of it.
4. Clone the git repository and place all folders under *racetrack/LEGOtruck* inside of your folder *~/catkin_ws/src*.
5. Open up a new terminal and enter the following lines:

   ```
   $ cd ~/catkin_ws
   $ catkin_make
   ```

6. Now connect your computer to the ”Visionen” network.

5.4 Starting the nodes

1. Now place the LEGO truck where you want the LEGO truck to start. Make sure the steering wheels point forward as straight as possible by turning the large black cogwheels.
2. Now connect any computer to the ”Visionen” network. It could be the same computer running roscore. Run the following command:

   ```
   $ ssh robot@ev3IPaddress
   ```

   and replace ev3IPaddress with the IP-address displaced in the top left corner on the EV3. The password is set to: ”racetrack”.
3. When you are connected to the EV3 run the following command:

   ```
   $ sudo nano rosinit.bash
   ```

   and make sure the IP-address after ROS_IP is the IP-address displayed on the EV3. Also make sure the IP-address after *http://* after ROS_MASTER_URI is the IP-address of the computer running roscore.
4. Now exit the editor and run the following commands to initiate ROS on the EV3:

```
$ cd ~/
$ .  rosinit.bash
```

5. When this is completed you can run the following commands to start the controller node:

   ```
   $ cd ~/
   $ .  starttruck.bash
   ```

6. Now open a new terminal on the Linux computer and go to the file *Qualisys-driver.cpp* in the folder ~ *catkin_ws/src/mocap_qualisys/src*. Make sure the IP-address on line 35 is the IP-address of the computer running QTM when connected to the Visionen network.

7. Now run the following commands to compile the code:

   ```
   $ cd ~/catkin_ws
   $ catkin_make
   ```

8. Now run the following command to start the roscore:

   ```
   $ roscore
   ```

   you only need one roscore so if you have one already running you can skip this step.

9. Now open a new terminal and run the following commands to start the node connected to the computer running QTM:

   ```
   $ cd ~/catkin_ws/src/mocap_qualisys
   $ .  start_mocap_qualisys_node.bash
   ```

10. Now open a new terminal and run the following commands to start the node that interprets the previous node to talk with the controller node.

    ```
    $ cd ~/catkin_ws/mocap_listener
    $ .  start_mocap_listener_node.bash
    ```

11. If you wish to use the planner that sends a pre-calculated trajectory keep following this list. If you wish to use the Lattice planner that calculates a trajectory online, follow the instructions under Section 5.5.

12. Open a new terminal and run the following commands to start the node publishing the precalculated trajectory. How to modify this can be found in the README-file found in the lego_truck_simple_planner folder. Make sure you have the terminal connected to the EV3 ready since this is where you

can stop the LEGO truck. If everything is set up the LEGO truck should soon start driving:

```
$ cd ~/catkin_ws/src/lego_truck_simple_planner
$ .  start_simple_planner_node.bash
```

13. To stop the LEGO-truck go to the terminal connected to the EV3 and press "CTRL+C".

5.5 Running the lattice planner node

Note: Before performing the steps below, please run all steps in section 5.3 first. To use the lattice planner for the LEGO truck, perform the following steps:

1. Run the lines following in a new terminal, one at a time:

   ```
   $ cd ~/catkin_ws/src/lego_truck_planner
   $ .  start_planner_node.bash
   ```

2. The ROS-node for the lattice planner should now be running, but it will not do anything until the user publishes a message to the topic */send_start_goal*. This is done by typing the following in a new terminal:

   ```
   $ cd ~/catkin_ws/src/lego_truck_planner
   $ .  send_start_goal.bash
   ```

   This bash script publishes a message to the rostopic */send_start_goal* where four starting states and four goal states are specified (x_3, y_3, θ_3, α). In order to change the starting and goal states, simply edit the eight values in the bash script file on the line where `data:` is specified. The first four values correspond to the starting states of the truck, and the final four values correspond the the goal states of the truck. **Note:** Make sure that the states x_3 and y_3 are rounded to a tenth, otherwise no solution will be found!

3. If the lattice planner has sent the entire plan to the LEGO truck, and the LEGO truck has driven to the goal, the ROS-node for the lattice planner has to be restarted if a new plan is to be sent to the truck. To do this, navigate to the terminal where the lattice planner ROS-node is active and do the following:

   - Press `CTRL+C` on your keyboard.
   - Start the node again by following step 1.

- Edit the starting and goal states in "`$ . send_start_goal.bash`"

6 Common Issues

Below are some common issues that can occur when trying to run the LEGO truck.

The LEGO truck is not moving.	Make sure all devices are connected to the "Visionen" network.
The truck is moving in a weird pattern.	Make sure the LEGO truck is detected properly in QTM.
The truck is moving only a short way when using the precalulcated trajectory.	Make sure the variable *nr_of_time_instances_* is set accordingly in the file *simple_planner.cc*.
The Linux computer is running hot and no measurements are received.	Check the *mocap_qualisys_node*, it crashes when you switch networks. Restart it if it has.
You get either of the messages: *No motors connected*, *No sensors connected*.	Make sure both motors and sensors are connected properly. If you go into *Device browser* on the EV3 you can see all sensors detected under *Sensors* and all motors detected under *Motors*.

7 Good to know info

- The password to the EV3 is: "racetrack", this is used when using ssh to log in to the LEGO truck.
- The password to the desktop Linux computer is: "race".
- The recommended speed forward is 150 (this is in degrees/second on the driving motor). This is set in the file constants in the folder ~ev3/catkin_ws/src/lego_truck_controller/src.
- The recommended speed backward is -75 (this is in degrees/second on the driving motor). This is set in the file constants in the folder ~ev3/catkin_ws/src/lego_truck_controller/src.

- The recommended steering speed is 400 (this is in degrees/second on the steering motor). This is set in the file constants in the folder ~ev3/catkin_ws/src/lego_truck_controller/src.