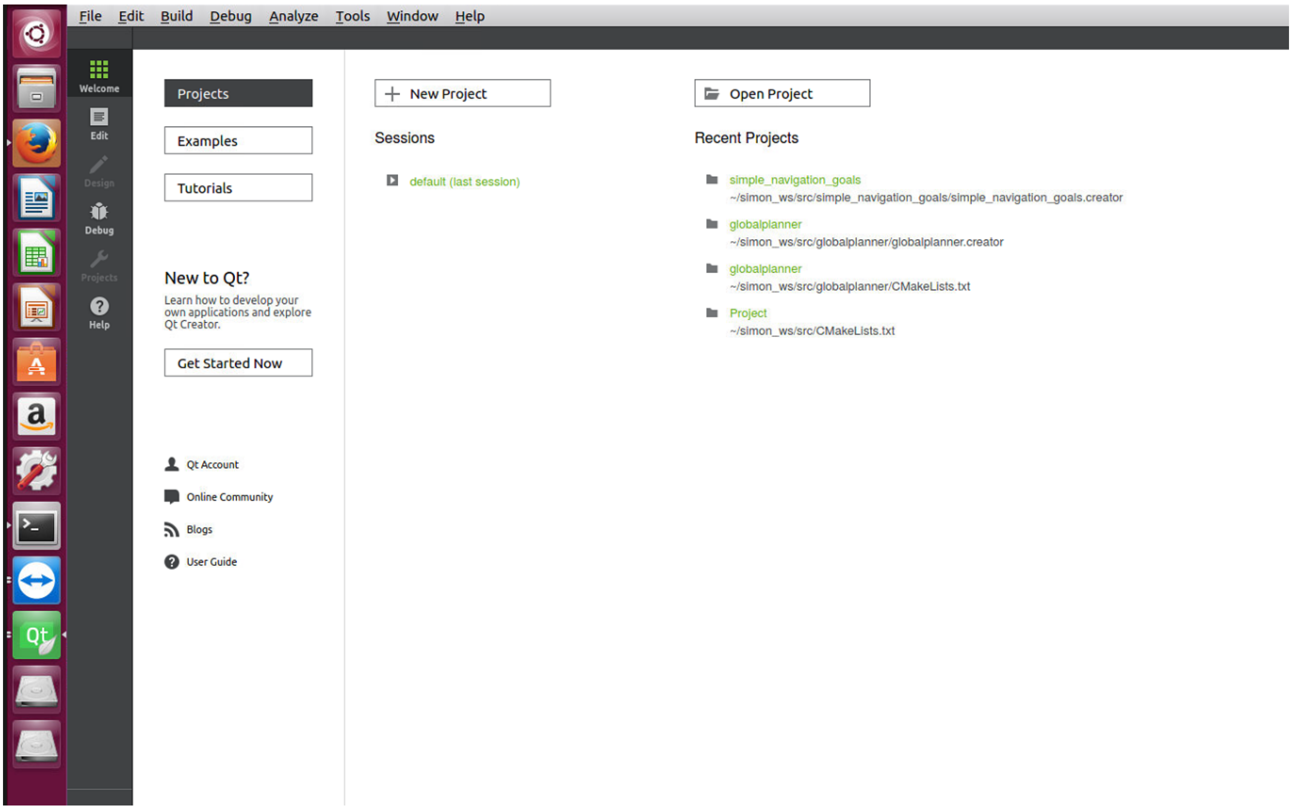
1 Download

Download Qt Creator from the official release at this link https://download.qt.io/official_releases/qt/, and select version **5.6**. Lower versions can be unsupported and might not be able to work with the debugger. The specific version used for this documentation is **5.6.1**.
For Linux systems run the file **qt-opensource-linux-x64-5.6.1.run**, and follow the installation wizard straight to the end. This will install Qt Creator in your machine.
Test the application by launching it from the terminal, simply typing **qtcreator &**. This erases the starting window, like the one seen below.



Learn how to develop your own applications and explore Qt Creator.

Figure 1: Welcome window opening the Qt Creator application

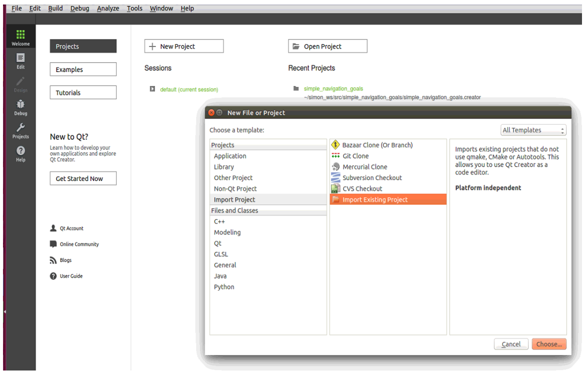
2 Importing a catkin package

Qt Creator is a versatile tool, naturally fitted to work with CMake build files. Any package under a **catkin workspace** can be built and run using Qt Creator, which resambles the functionality of the ROS **run** or **roslaunch** commands.
In order for Qt Creator to work in the catkin shell, the desired catkinized package (containing a **CMakeLists.txt and package.xml files**) should be imported into the application, by following the steps:

- Click on File →New File or Project. This will pop up the window shown in Figure 2.
- Select Import existing projects and click Choose.
- Give as Project Name the **exact** same name as the catkin package name you want to import. Browse to the path under the field Location. An example is visible in Figure 3 In

this case the name of the package is called **simple_navigation_goals** and the path is **simon_ws/src/simple_navigation_goals**. Click Next.

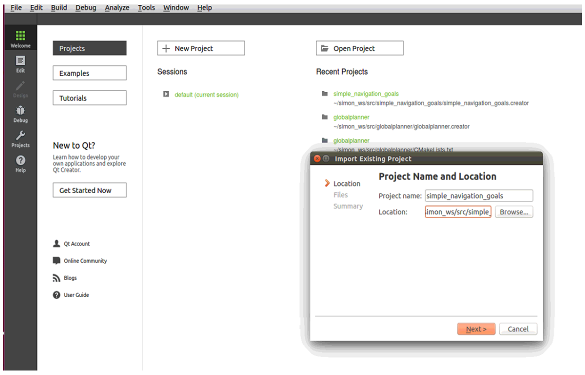
- Click Next to the following two windows that happear. Tick the checkbox **include** if the package contains header files in the include directory.



Imports existing projects that do not use qmake, CMake or Autotools. This allows you to use Qt Creator as a code editor.

Platform independent

Figure 2:



File Edit Build Debug Analyze Tools Window Help
Welcome
Edit
Design
Debug
Projects
Help
Projects
Examples
Tutorials
New to Qt?
Learn how to develop your own applications and explore Qt Creator.
Get Started Now
Qt Account
Online Community
Blogs
User Guide
New Project
Open Project
Sessions
default (current session)
Recent Projects
simple_navigation_goals
~/simon_ws/src/simple_navigation_goals/simple_navigation_goals.creator
globalplanner
~/simon_ws/src/globalplanner/globalplanner.creator
globalplanner
~/simon_ws/src/globalplanner/CMakeLists.txt
Import Existing Project
Project Name and Location
Location
Files
Summary
Browse...
Next >
Cancel

Project name: simple_navigation_goals

Location: imon_ws/src/simple_

Figure 3:

3 Setting up the debugger to work with catkin

If the process has terminated succesfully till this point, you should see something similar to the top left corner of Figure 6.
To allow Qt Creator to build and run the package, the build and run settings should resamble the ones in Figure 4 and Figure 5.

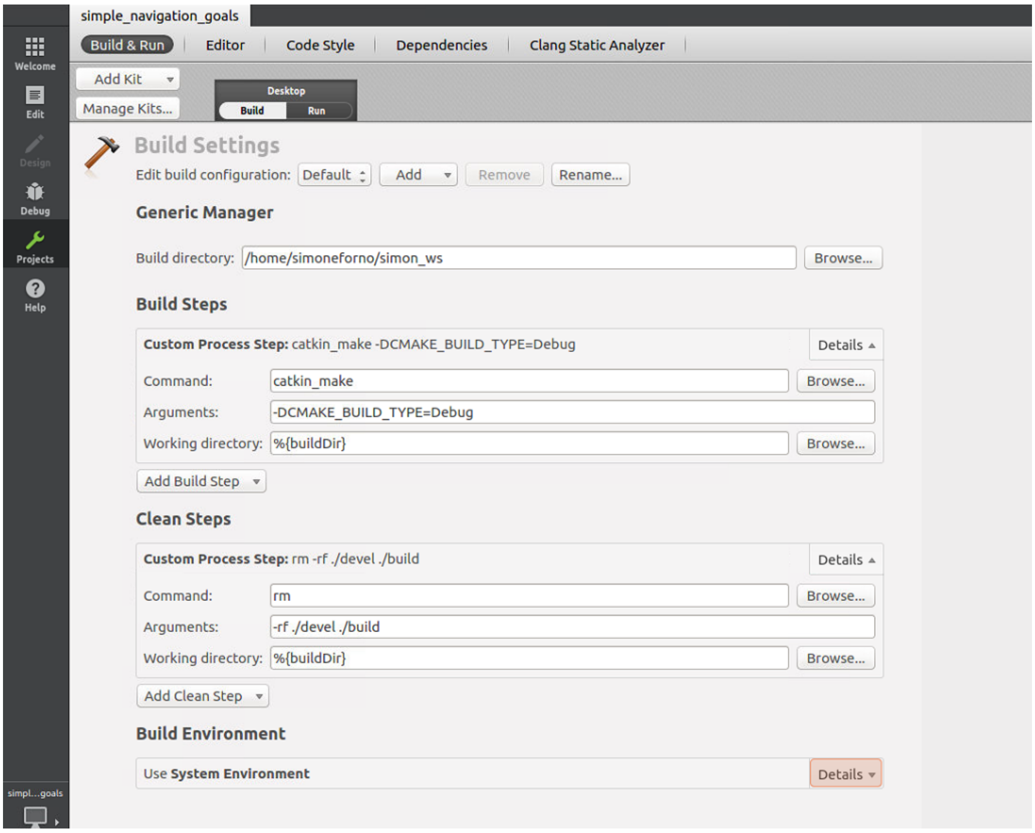
3.1 Build settings

- Change the Build Directory to the catkin root path and **do not** select the build folder under *workspace_name/build*.
- Under Build Steps, delete the default existing Process Step and add a new Process step by clicking the **Add Build Step** tab →**Custom Process Step**. Fill in the build steps as in Figure 4.
- Do the same also for the **Clean Steps** tab, delete the default Process Step and fill in the fields as in Figure 5.

3.2 Run settings

- Leave the **Deployment** part unchanged.
- In the **Run** part, for the Run configuration part, click on Rename and browse to the **devel/lib** path of your catkin workspace.
- Browse to the package Executable under the **lib** folder. Leave the remaining section as it is.

To debug the code, place breakpoints into the code and click on **Start Debugging** button, or **F5**. This will compile the package and create the targets. It is possible to visualize the variables in the top right panel, and see the changes by Stepping into the code, as see in Figure 6.



Build Settings

Edit build configuration: Default

Generic Manager

Build directory: /home/simoneforno/simon_ws

Build Steps

Custom Process Step: catkin_make -DCMAKE_BUILD_TYPE=Debug

Command: catkin_make

Arguments: -DCMAKE_BUILD_TYPE=Debug

Working directory: %{buildDir}

Clean Steps

Custom Process Step: rm -rf ./devel ./build

Command: rm

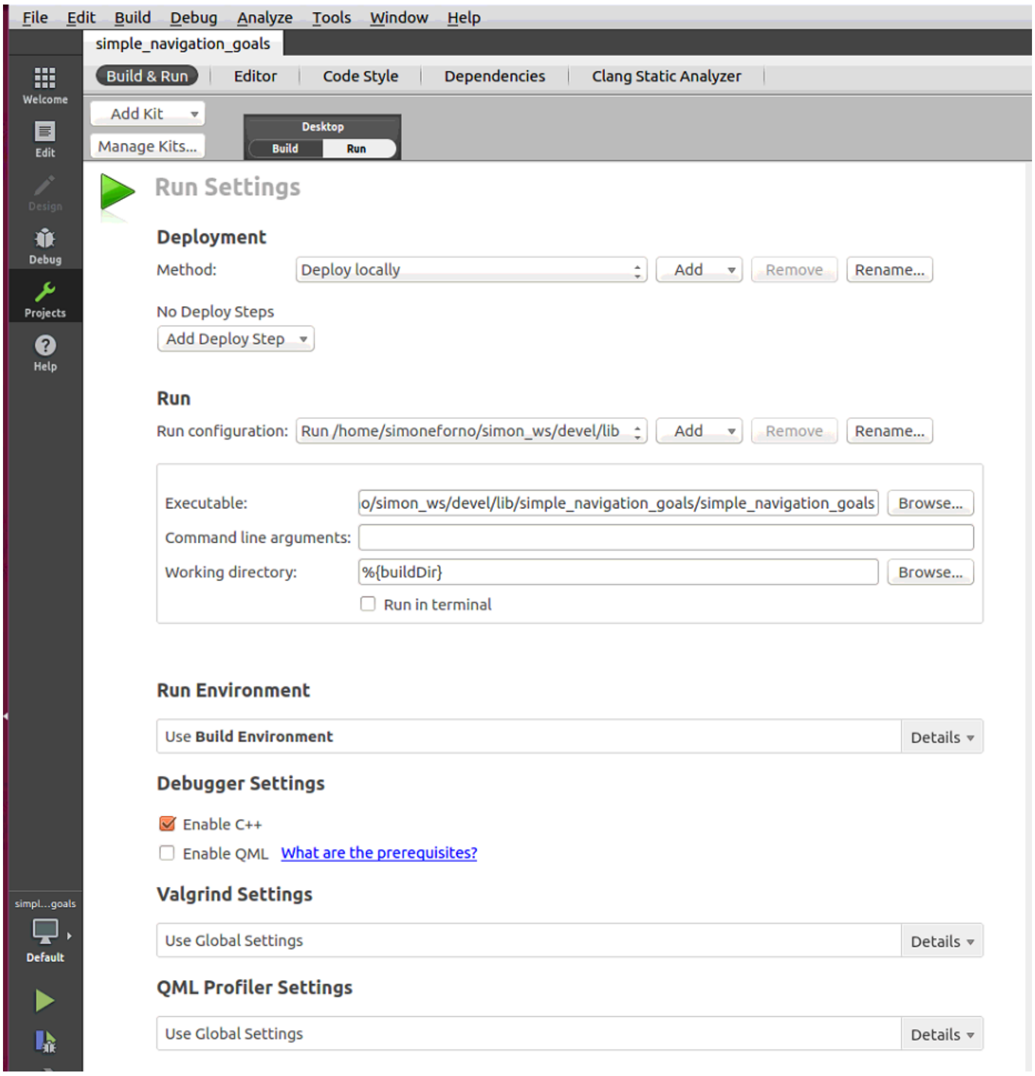
Arguments: -rf ./devel ./build

Working directory: %{buildDir}

Build Environment

Use System Environment

Figure 4: Build settings of the catkin package



Run Settings

Deployment

Method: Deploy locally

No Deploy Steps

Run

Run configuration: Run /home/simoneforno/simon_ws/devel/lib

Executable: o/simon_ws/devel/lib/simple_navigation_goals/simple_navigation_goals

Command line arguments:

Working directory: %{buildDir}

□ Run in terminal

Run Environment

Use **Build Environment**

Debugger Settings

☑ Enable C++

□ Enable QML What are the prerequisites?

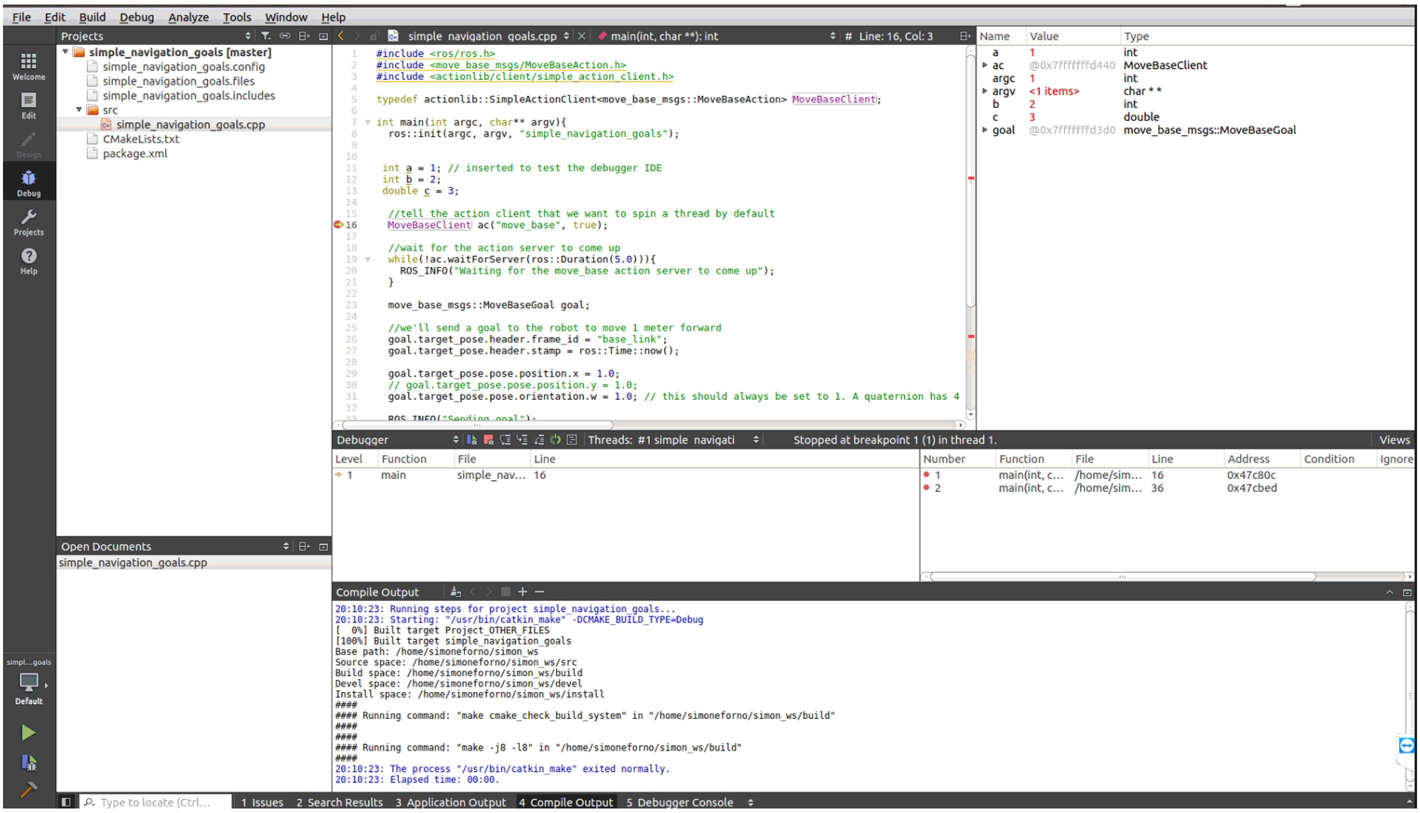
Valgrind Settings

Use Global Settings

QML Profiler Settings

Use Global Settings

Figure 5: Run settings of the catkin package



```
#include <ros/ros.h>
#include <move_base_msgs/MoveBaseAction.h>
#include <actionlib/client/simple_action_client.h>

typedef actionlib::SimpleActionClient<move_base_msgs::MoveBaseAction> MoveBaseClient;

int main(int argc, char** argv){
  ros::init(argc, argv, "simple_navigation_goals");


   int a = 1; // inserted to test the debugger IDE
   int b = 2;
   double c = 3;

  //tell the action client that we want to spin a thread by default
  MoveBaseClient ac("move_base", true);

  //wait for the action server to come up
  while(!ac.waitForServer(ros::Duration(5.0))){
    ROS_INFO("Waiting for the move_base action server to come up");
  }

  move_base_msgs::MoveBaseGoal goal;

  //we'll send a goal to the robot to move 1 meter forward
  goal.target_pose.header.frame_id = "base_link";
  goal.target_pose.header.stamp = ros::Time::now();

  goal.target_pose.pose.position.x = 1.0;
  // goal.target_pose.pose.position.y = 1.0;
  goal.target_pose.pose.orientation.w = 1.0; // this should always be set to 1. A quaternion has 4

  ROS_INFO("Sending goal");
```

Name	Value	Type
a	1	int
▸ ac	@0x7fffffffd440	MoveBaseClient
argc	1	int
▸ argv	<1 items>	char * *
b	2	int
c	3	double
▸ goal	@0x7fffffffd3d0	move_base_msgs::MoveBaseGoal

Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line
1	main	simple_nav...	16

Number	Function	File	Line	Address	Condition	Ignore
● 1	main(int, c...	/home/sim...	16	0x47c80c		
● 2	main(int, c...	/home/sim...	36	0x47cbed		

```
20:10:23: Running steps for project simple_navigation_goals...
20:10:23: Starting: "/usr/bin/catkin_make" -DCMAKE_BUILD_TYPE=Debug
[  0%] Built target Project_OTHER_FILES
[100%] Built target simple_navigation_goals
Base path: /home/simoneforno/simon_ws
Source space: /home/simoneforno/simon_ws/src
Build space: /home/simoneforno/simon_ws/build
Devel space: /home/simoneforno/simon_ws/devel
Install space: /home/simoneforno/simon_ws/install
####
#### Running command: "make cmake_check_build_system" in "/home/simoneforno/simon_ws/build"
####
####
#### Running command: "make -j8 -l8" in "/home/simoneforno/simon_ws/build"
####
20:10:23: The process "/usr/bin/catkin_make" exited normally.
20:10:23: Elapsed time: 00:00.
```

Figure 6: Debugger in action