# Introduction to ROS
# Group Project: Autonomous Quadruped

## 1 General Information

### 1.1 Administration

As it is impossible to match the three offered tasks in complexity, we will account for it by matching the grading distributions of the three tasks.

### 1.2 Timeline

- Today: Announcing the projects
- 25.06.2022: Release of code
- 02.08.2022: Submission and presentations

### 1.3 Submission

The project submission consists of three parts:

- Source Code (Deadline: 02.08.2022 - End of day)
- Documentation (Deadline: 02.08.2022 - End of day)
- Presentation (Deadline: 02.08.2022 - 1pm at the common Zoom-Link)

The presentation will not be graded but shall help the reviewers to quicker understand your code structure, limitations and issues you had during implementation.

Your source code and documentation must be submitted via a link to a github repository. In both cases you should include the src folder of your catkin workspace as well as the source code folder(s). In case you modified your unity simulations, please communicate that clearly in your documentation and give us access to your simulator source and binary.

#### 1.3.1 Source Code

While writing code, please pay attention to writing clean code, think about how to structure your code into individual ROS nodes. Think about structuring your code into individual ROS packages.

Please also include a `readme.md` file with detailed step by step installation instructions necessary to run your system.

You are allowed to use any free available code. Please indicate in the documentation which part of the code is written by yourself and where you use external code - please as well report that in the documentation.

#### 1.3.2 Documentation

The documentation shall be approx. 2 pages but is not limited to that. The goal of the documentation is to help us understanding your code. In your documentation you should include a short description of every ROS node and its functionality. Please let us as well know in case you were not able to fulfill all requirements and explain the problems. Document where you used external code, not written by yourself. Finally, discuss who of the team members worked on which components. Beyond that, you might include

- a ROS graph,
- figures, and plots presenting your results,
- a bibliography.

#### 1.3.3 Presentation

Please prepare an oral presentation of your documentation. The presentation length shall not be longer than 6 minutes. An introduction to the field and topic is not required. The goal of the presentation is to give us a quick overview over your project, explain how you implemented the individual components, discuss limitations and issues and give us feedback about the project work.

# 2 Task

## 2.1 Goals

The goal of the project to pass a parkour with your robot in minimal time.

The core parts, including but not limited are:

- A Unity simulation environment. A base version will be provided to you.
- A ROS-Simulation-Bridge providing ROS interfaces (topics, services). It will communicate with the simulation via TCP while at the same time providing relevant information to other ROS nodes. A base version will be provided to you. You are free to adjust the code.
- A basic position controller for the single joints of the robot. For simplicity we will assume the we can control the robot kinematically.
- A state machine for your robot.
- A perception pipeline that converts the depth image, first to a **point cloud** and second to a **voxel-grid** representation of the environment.
- A path planner that generates a path through the environment.
- A trajectory planner that plans a trajectory based on the found path.

## 2.2 Grading

You can reach a maximum of 100 points in the following categories:

- Functionality and Performance (max. 50p)
  - Successfully working perception pipeline (max. 10p)

  - Successfully working path planning (max. 8p)
  - Successfully working trajectory planning (max. 8p)
  - Successfully working planner for coordinated motion (max. 6p)
  - Successfully crossing the parkour (max. 12p)
  - Time to complete the mission (max. 6p)
- Code and Architecture Quality (max. 30p)
  - Sensible separation in packages, nodes and functions (max. 10p)
  - Sensible separation in services, topics, etc (max. 10p)
  - Code quality, readability, and traceability (max. 10p)
- Written Summary (max. 20p)
  - Sound explanation of your architecture, model, and design choices (max. 15p)
  - Clear documentation who did what, which code is your own, which code is reused (max. 5p)
- Bonus Points (max. 10p)
  - Solving the problem on the hint legs only (max. 10p)

**Very important: We will subtract up to 30p if your code does not build or performes differently then documented.**

### 2.3 Tips

Here are a couple of hints regarding the implementation. The hints are just suggestions, you are free so solve the task differently:

- Generating point cloud from depth image: use *depth_image_proc* in http://wiki.ros.org/depth_image_proc.
- Generating occupancy Grid: use *Octomap* in http://wiki.ros.org/octomap.
- **Please ping us in case you have any questions or if you get stuck in some subtasks.**