# SOFTENG 306 Software Design 2
# Project 1: Group robotic behaviour simulation using Robot Operating System (ROS).

Department of Electrical and Computer Engineering, University of Auckland
Dr Bruce MacDonald, Mr Chandan Datta
Final Version. 23 July 2012

Changes from the draft are shown in blue. Some deletions are shown struck out; only if the content of the deletion was important.

In Swarm Robotics, a large number of simple robots are coordinated to achieve a collective behaviour. In this project, you will simulate small groups of robots that show simple collective behaviours. To coordinate the collective behaviour, robots need to exchange messages. To show the collective behaviour, they should be simulated in a graphical simulator.

## Key tasks

The key tasks of the project are as follows:

- Implementing individual robotic nodes
- Implementing group behaviour for subgroups of robots
- Message passing among robots
- Graphically simulating the collective behaviour

Above key tasks are briefly explained below.

In this project, individual robotic nodes are simple differential drive robots (robots with 2 independently driven wheels). Each robotic node is defined by a set of parameters; robot-id, pose ($x$, $y$, $\theta$ ) with reference to the simulation world frame, linear velocity ($\dot{x}$, $\dot{y}$), angular velocity ($\dot{\theta}$).

Each robot can transmit messages containing its current state (pose, velocity). Each robot can listen to messages transmitted by other robots and learn about the group. For message passing, ROS can be used.

For graphically simulating the robots, the Stage 2D robot simulator can be used.

## Expected outcome: alpha version

This is the minimum requirement for the progress demo:

One group of six robots with arbitrary poses should be placed in the simulation world (Stage). Once started, each robot should start transmitting and receiving messages. By processing the received messages, the group will elect a head node (cluster-head) for the group. The node that is closest to the $(x, y)$ origin (0,0). Robots that are exactly at the origin (0,0) are not eligible to be the cluster-head.

Once the cluster-head is elected, the cluster head should orient towards the origin and remain stationary. The other six robots should start moving to their new positions, which should be in a radial line out from the (0,0) origin through the cluster head position, and spaced evenly at 5 robot diameters apart.

After that, the group should move in to a circle and the circle should rotate. The movement should be by the cluster head leading a curving motion initially in the direction of the origin, and then curving to form a circle, and the other robots should follow (should not be by moving directly to circle positions).

## Multiple subgroup behaviour: final version

Once the above outcome is achieved, add multiple groups of robots. Each group should have at least six robots, and there should be at least four groups. When the robots are initialised, the $N$ subgroup cluster heads should be the $N$ robots that are closest to the origin. Other robots should decide which group to join and line up behind their clusters heads to begin (in the same way as for the alpha version). Once lined up, each cluster head should move to a predetermined position, register an instruction from something you sense at this position (imagine a bar code that defines a command, and arrange some kind of sensor to read sensor data to give the command). You can manually move objects to the relevant positions to give such commands. Then the cluster head should return to its subgroup and execute the command with its subgroup. Commands should include:

- Goto-Circle $x, y$ ~~$x, y, \theta$~~ (the cluster should circle this position and then remain stationary)
- Goto-Square $x, y, \theta$ (the cluster should form a square around this position)
- Rotate (the cluster should form a cirle at the original position and continually rotate around that circle). This is the same behaviour as the rotating circle in the alpha version.
- You may choose other behaviours also. At least one other behaviour should be implemented.

The requirements are intended to give each student a substantial software design and coding task. This could be divided by behaviours, reading the command, and initialisation, for example, or there may be a better way to divide up the tasks, according to a well designed software architecture, in to appropriate classes and methods.

Note: We advise you to setup the ROS environments as early as possible to smooth the development process and also think about your design before you code.

## Group organisation

The project will be carried out by six groups of seven students per group. The work must be divided up among group members as the project progresses, giving each group member a role in coding, and also at least one other role e.g. to work on software installation, planning, design, testing, trac wiki and documentation. However do not divide in to independent subgroups because that will lose marks for your group. Share the tasks around so everyone contributes across the whole project and is aware of the overall project development. All students should contribute to coding. We want to see clear evidence that the whole group is working together. There should be one robot class, and instances made for each robot, with subclasses as required to distinguish different kinds of robot.

## Software development process

Follow the following software development process, and document each step on your trac wiki:

- Project Plan. Make a project plan including the steps below, broken in to more detailed subtasks, estimating the time to be taken, milestones, resource allocation etc.
- Requirements elicitation and analysis. Analyse this handout. Ask questions on the Cecil forum for staff to answer as clients. Write a list of requirements.
- List the design options, discuss and choose a design
- Elaborate design details
- Design required tests
- Implementation (including tests)
- Testing and evaluation
- Documentation
- Repeat from any step as required
- Record the hours each person spends on which tasks

- Take turns to write meeting minutes

Please note that this is a collaboration project and that means every member should actively engage in and value conversation and meeting. It is expected that most of the planning and design decision will be made through effective meetings and they should be documented in the form of meeting minutes which should then be uploaded to the trac wiki. (If you do not already know how to do meeting minutes, please learn by searching the Internet).

## Schedule

Here is a plan of what each group should do:

| | |
|---|---|
| **Week 1** | Choose your groups and email the list to us, including the group leader's name. Select a group leader and document group members, leader, and roles on your trac. Comment on the project draft. Learn about ROS and install it in your own computer. ROS will be installed in lab computers by the IT department. Become familiar with ROS, Stage, Swarm robotics, trac, MS project. |
| **Week 2** | Make an initial project plan and overall software design, set up trac, start an alpha system for the first demo. |
| **Week 3** | Prepare and hand in your project plan. |
| **Week 3-4** | Prepare and present progress for assessment. Continue development. Marks: Equally divided between: Presentation/Demo, Code, Trac/Svn |
| **Week 4–5** | Iterate your software development process as required. |
| **Week 6** | Finish and present project for assessment. Update the plan, hours, and completion status. Marks: Equally divided between: Presentation/Demo, Code, Trac/Svn, Report |

See the course handout for the overall assessment weightings.

Note that we will be able to set up your trac only once your group is defined so please us your group as soon as possible. Then we will ask ECE IT to set up your Svn/Trac. Group Svn/Trac urls will probably be: `https://svn.ece.auckland.ac.nz/se306-2012-g1`, `https://svn.ece.auckland.ac.nz/se306-2012-g2`, etc. Do not add others to your svn access, and leave the staff permissions as set up so that we can see your activity.

## Deliverables

**Week 3: Project Plan**

**Week 4:**

- Setup of the ROS environment.
- Presentation of the project plan and current status, including requirements
- Demo of initial alpha version
- Svn and trac set up (initial code and documentation, plan, requirements)
- Evidence of regular work from svn history and trac tickets

**Week 6:**

- Presentation and demo in the lab of working final version
- Questions in the lab during the demo
- On Svn:
  - Code (including svn history)
  - Documentation (including user manual)
  - Svn log messages

  - Build environment (make files etc)
- Trac:
  - Wiki documentation for the project development process including requirements, plan, hours spent , design, tests
  - Ticket history and current status
- Others:
  - Peer evaluation forms (one for each student), using the given form on Cecil. Submit confidentially.

## Lectures

We will use all four lectures per week initially, for the first couple of weeks, to give introductory material, and then we will just meet for the Thursday lecture to discuss the project and to hear the group reports (worth a total of 10% of the final grade over the whole course). All group members are expected to attend in order to get the marks.

## Laboratory use

The TA will help people in the labs, as detailed on Cecil.

## Software tools

**ROS** Will be installed on ECE department remote linux. Instructions to install ROS on your laptops will be available in Wiki.

**Using ROS and Stage** See our robotics lab teaching pages at: `https://svn.ece.auckland.ac.nz/robotics-teaching`. There are details about using the tools for this project. This includes ROS installation and using Stage with ROS.

**Project planning** Use Microsoft project, which is available in Windows on the ECE and Engineering PCs. Define the tasks, and break them down into subtasks and subsubtasks, and so on. Assign your group members to each low-level task, and specify the number of hours for each person. Remember you are working part time (10 hours per week). Include group meetings, analysis, design, implementation, testing, documentation, and so on. Upload updated versions of the project plan to the svn regularly to show that you are keeping your planning up to date.

**Subversion** You have learned to use subversion already. If you have questions please ask. Remember you can get help from svn itself (eg "svn help log"). The svn book is also useful (`http://svnbook.red-bean.com/`).

If you are programming in pairs or in a group then make sure you take turns doing the coding, and committing to svn, so that we can see you are taking part when we look at the subversion logs. Also include the names of the people in the programming pair or group in the log message when you check in.

Make sure you include messages in your svn checkin, so that we can understand your contribution in each case.

**Trac** Trac is a web front end for subversion, and provides a wiki for documentation, a ticket (issue) tracking system for bugs and other issues that arise in your work, and also some other tools. There is documentation available in trac itself. You can add components, priorities etc in the admin section. We are looking to see a good structured Trac system set up for your project and will give marks for it.

**Alternative to svn and trac** If you want to you can use another tool, so long as you give all staff full access for marking (if we can't see your work we can't give marks for it), and so long as you provide the same functionality and documentation as required on svn and trac.