Topic	Description
Module	Robotics
Module Code	ROB
Semester	SS 2025
Lecturer	Daniel T. McGuiness, Ph.D
ECTS	5
SWS	1
Lecture Type	ILV
Teaching UE	15
Coursework Name	A Assignment
Work	Individual
Suggested Private Study	20 hours
Submission Format	Submission via SAKAI
Submission Deadline	29th Jun 23:59
Late Submission	Not accepted
Resubmitting Opportunity	No re submission opportunity

> No lecture time is exclusively devoted to the aforementioned assignment.

A portion of the mark for every assignment will be, where applicable, based on style. Style, in this context, refers to organisation, flow, sentence and paragraph structure, typographical accuracy, grammar, spelling, clarity of expression and use of correct IEEE style for citations and references. Students will find *The Elements of Style (3rd ed.)* (1979) by Strunk & White, published by Macmillan, useful with an alternative recommendation being *Economist Style Guide (12th ed.)* by Ann Wroe.

Question	Maximum Point	Result
Go Turtlebot Go!	100	
Sum	100	

[Q1] Go Turtlebot Go! — 100

Here you will create a ROS 2 application which creates a turtlesim environment, where the robot must do the following:

1. The robot (i.e., the turtle) should randomly start moving in a direction chosen randomly,
2. Once the robot approaches an edge, it will stop, print to the terminal `edge detected!`, and then rotate **90° degrees clockwise** and continue.

   > The rotation should be away from the wall.

3. This operation will continue **as long as the code is in operation**.
4. This package will be submitted to SAKAI with a **bash script** which automates the installation.

`turtlesimAutomata` will be **automatically** installed using a bash script which does the following:

5. Creates a workspace (`ros2\_ws/src`) in the Home folder,
6. Moves the package `turtlesimAutomata` to the appropriate folder,
7. Compiles the package using `colcon`,
8. Executes the package,
9. Removes the downloaded package inside the `Downloads` folder.

Rules and Requirements of your Assignment

This goes without saying, but your code must be executable in a Linux environment **without any modification**. To be able to do this assignment, you need to have Ubuntu installed with ROS 2 Humble. If you need more information, please consult the lecture documents. Before starting the assignment, you should make sure you understand ROS 2 concept and should be familiar with the CLI. In addition, go over all the code samples from ROS 2 tutorials and make sure you understand them thoroughly.

1. Make sure that your code is tidy and well-commented.
2. Your package must be written in Python.
   a) If you are going to use a code from someone, **cite** the part of the code.

3. **You should do this work on your own**. All the work you turn in should be yours, and not done in collaboration with anyone else. If you use any external sources of inspiration, other than official ROS 2 documentation, you must declare it in a `README` file.
4. You should **hand in everything needed to run your code**. This means:
   a) Your source code, adequately commented, so that each distinct part of the code is clearly explained.
   b) Your `.bash` script to automate the installation or the package and execution.
   c) The launch file (if necessary),
   d) Your automation bash script (`turtlesimAutomata.bash`)
5. **Failure to hand in the source code will result in immediate failure**
6. A document including a 1-page cover sheet with title, background of the assignment, followed by discussion of: procedures followed; what worked and what did not; the testing carried out.
7. The rest of your document should include screenshots of the turtlesim and terminal output. These should be presented as numbered figures, referred to (where relevant) in your discussion.
8. The document should be written in LaTeX.

> You should not hand in executable files, or any other files that can be regenerated.

> Your code should run with only one single command, using the terminal in the correct directory `bash tutlesimAutomata.bash`

> If you only submit ROS 2 part of the assignment, the maximum attainable points for the assignment will be 60.

Style Guidelines

Formatting

1. Code should be single spaced and use readable font sizes.
2. Add spacing around operators if it improves clarity.

> To have a good overview of industrial standards of python programming, please have a look at PEP 8 - Style Guide for Python Code.

Commenting

1. The goal of commenting is to make your code easily understandable by someone else.
2. There should be a block comment at the start of your program explaining what it does and including: your name, the assignment and the date.
3. Every method should have a block comment explaining what is does. I suggest that you write the block comment before you write the method.
4. Add inline comments where needed to explain subtle or tricky code.
5. Add blank lines and comments to separate sections of large methods that perform several different tasks.

Naming Variables, Methods, and Classes

1. Take the time to choose meaningful names.
2. Follow naming conventions (i.e., PEP).
3. You can use abbreviations, but add comments explaining them.

> Many people comment out old code that wasn't working correctly. Clean up and remove any old code before turning it in.