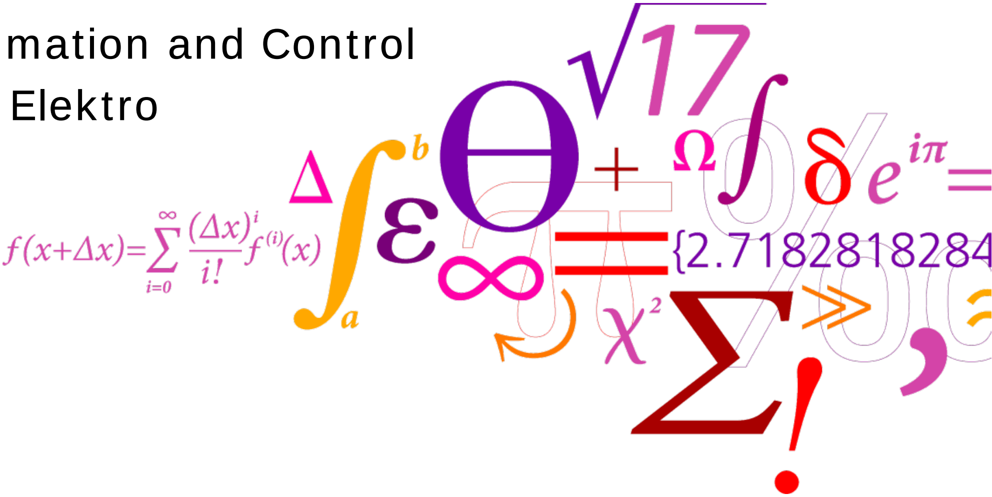
Project management and documentation

What has been done and how

Ole Ravn
Automation and Control
DTU Elektro


$f(x+\Delta x)=\sum_{i=0}^{\infty}\frac{(\Delta x)^i}{i!}f^{(i)}(x)$
$\sqrt{17}$
{2.7182818284

DTU Electrical Engineering
Department of Electrical Engineering

Outline

- Project phases
- Project planning
- Programming style
- Debugging
- Documentation
- Competition

Practical

- Schedule from Monday

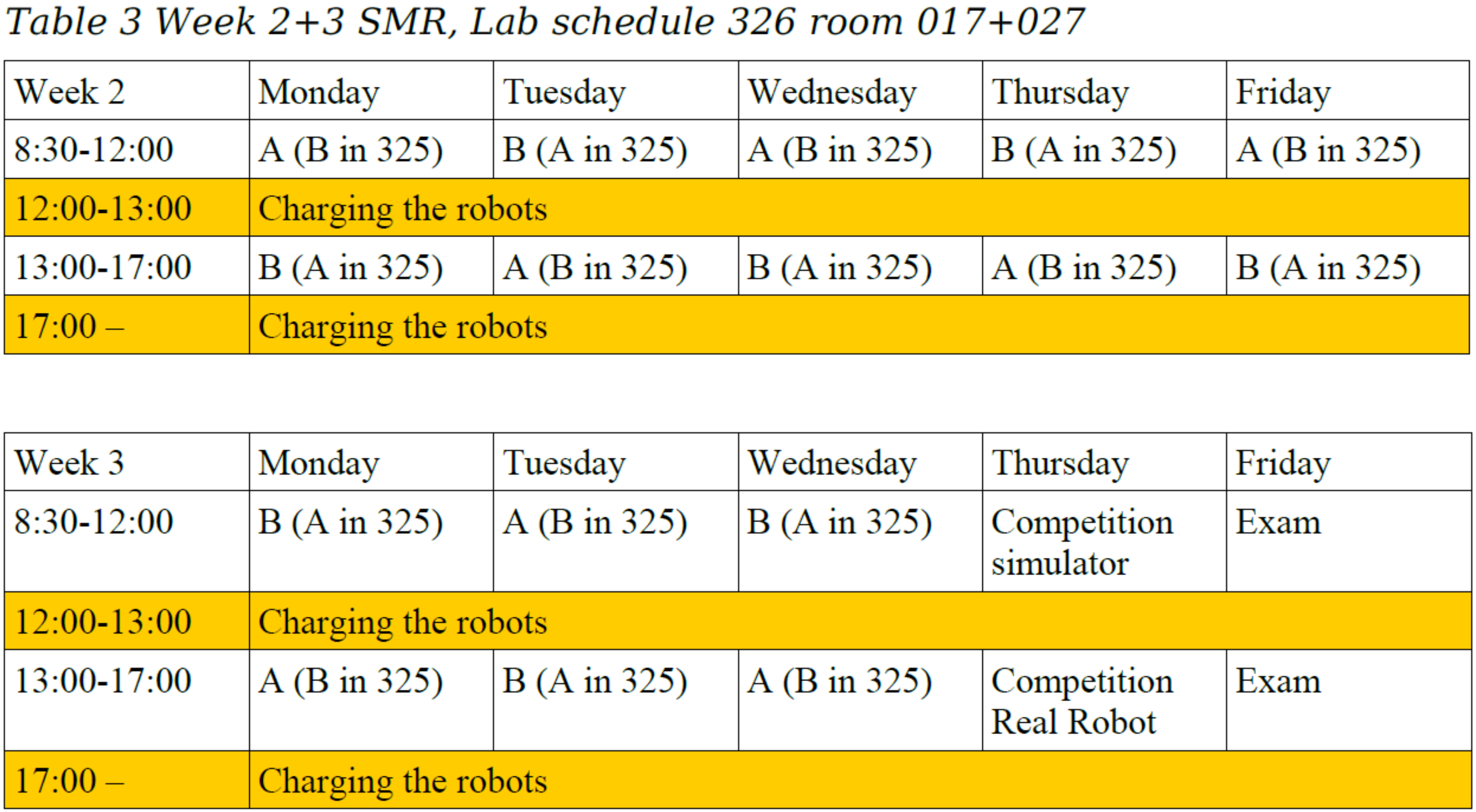


Table 3 Week 2+3 SMR, Lab schedule 326 room 017+027

Week 2	Monday	Tuesday	Wednesday	Thursday	Friday
8:30-12:00	A (B in 325)	B (A in 325)	A (B in 325)	B (A in 325)	A (B in 325)
12:00-13:00	Charging the robots				
13:00-17:00	B (A in 325)	A (B in 325)	B (A in 325)	A (B in 325)	B (A in 325)
17:00 –	Charging the robots				

Week 3	Monday	Tuesday	Wednesday	Thursday	Friday
8:30-12:00	B (A in 325)	A (B in 325)	B (A in 325)	Competition simulator	Exam
12:00-13:00	Charging the robots				
13:00-17:00	A (B in 325)	B (A in 325)	A (B in 325)	Competition Real Robot	Exam
17:00 –	Charging the robots				

Practical

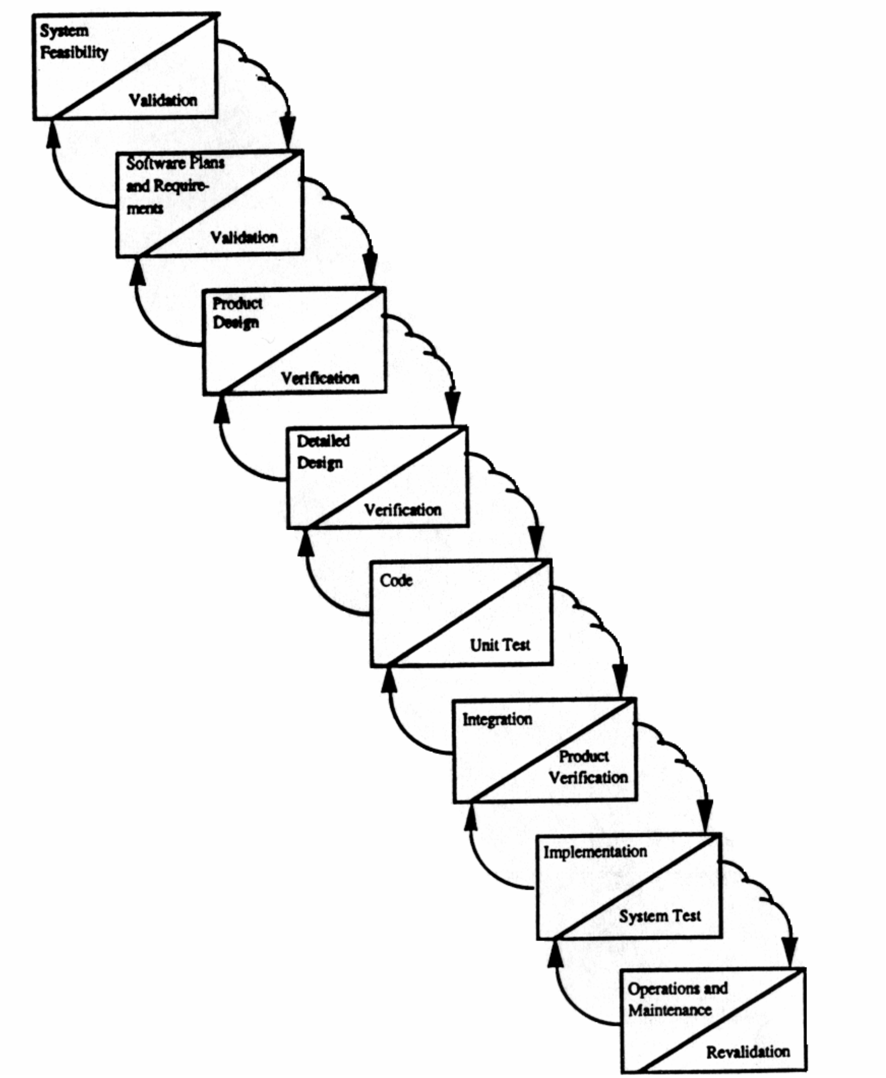
- Essential to recharge the robot at **every** opportunity
- Cool-line help from next week
- Split tasks, plan and use simulator
- All first week exercises on web by Monday evening
- Poster hand-in electronically at 17.00 on the last day of the course

First week evaluation

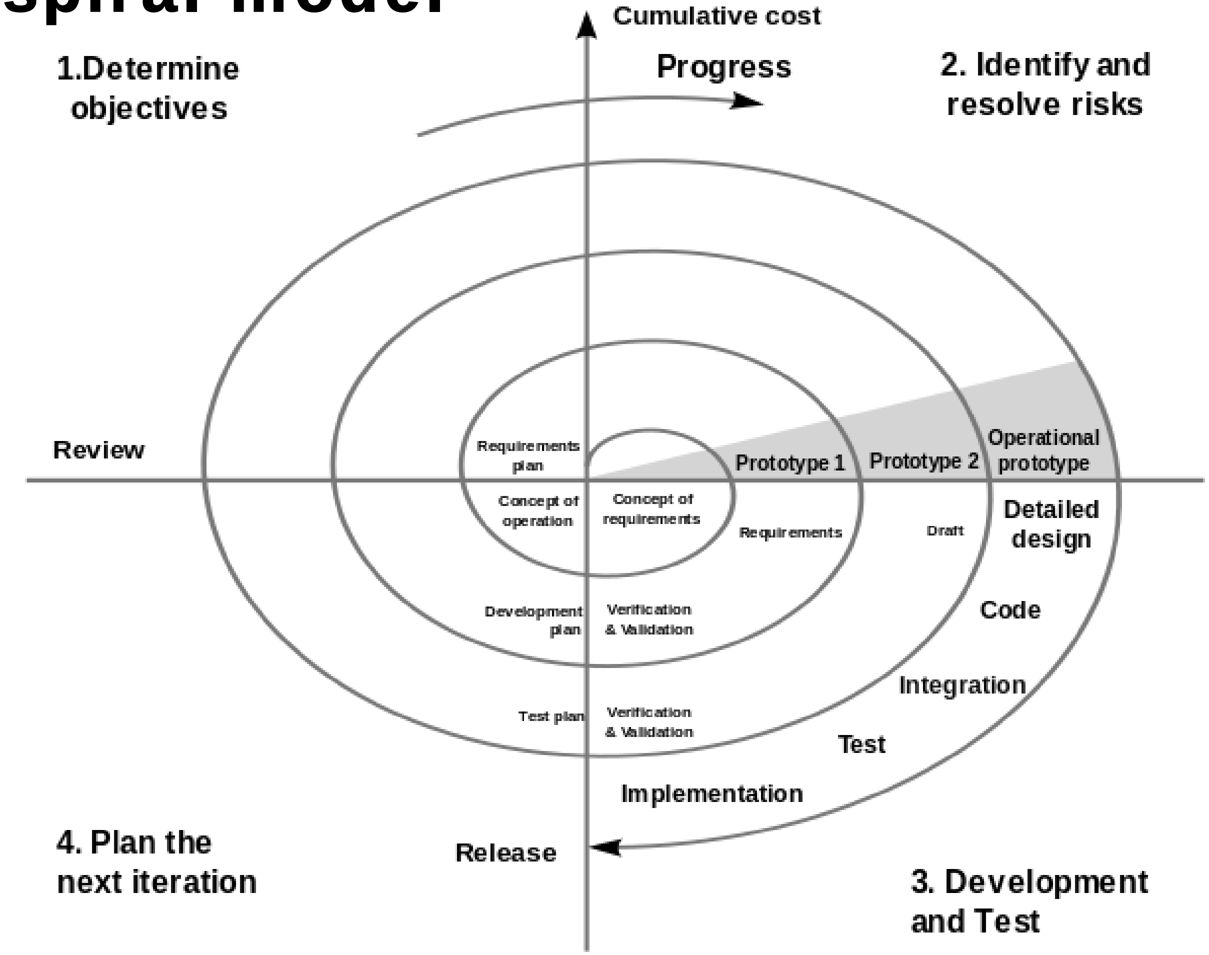
- Too little to do Monday afternoon and Tuesday morning?
- Slides on CN in advance

- Your input ....

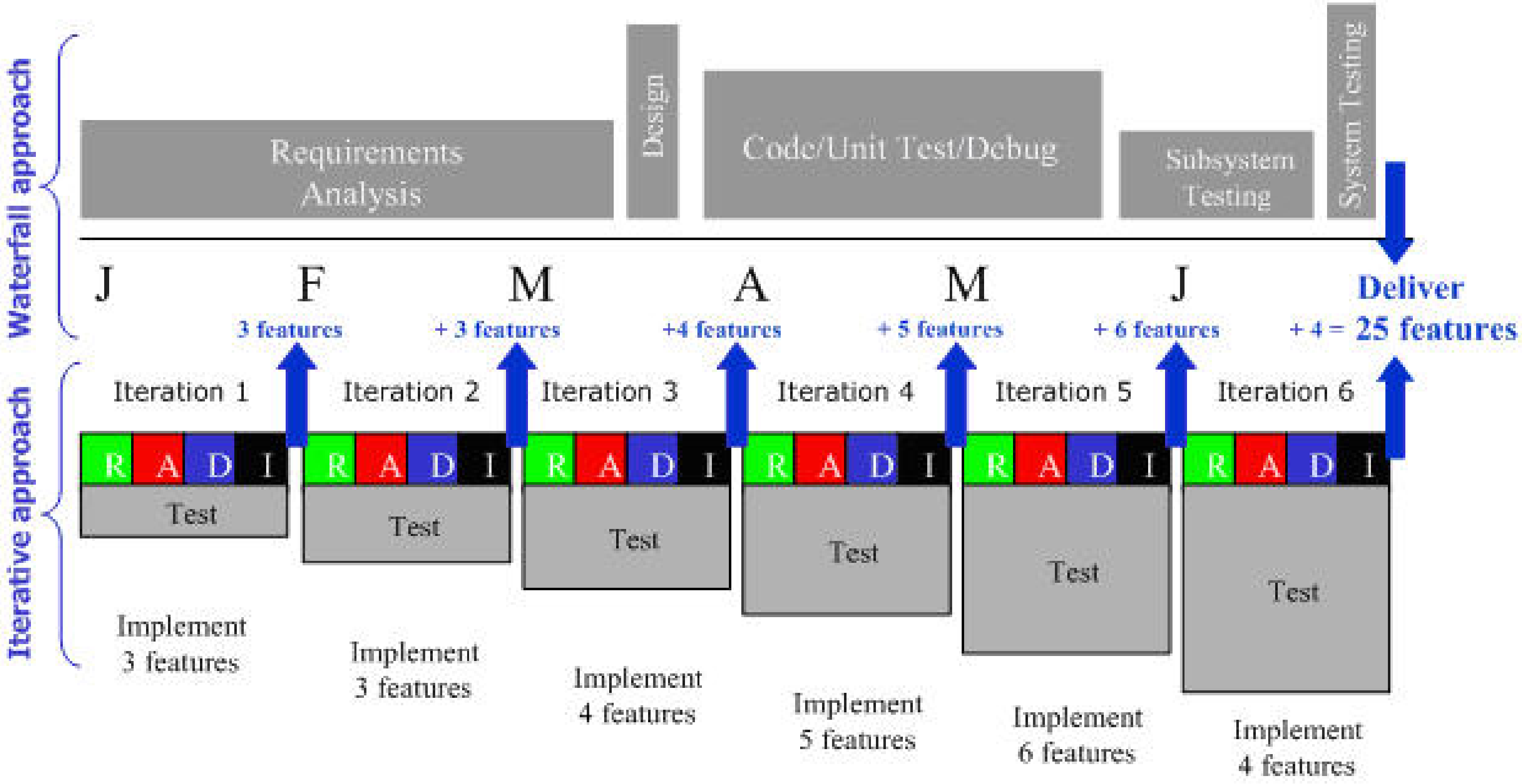
The water fall model



The spiral model



Time line



Waterfall approach
Requirements Analysis
Design
Code/Unit Test/Debug
Subsystem Testing
System Testing
J
F
M
A
M
J
Deliver
3 features
+ 3 features
+4 features
+ 5 features
+ 6 features
+ 4 = 25 features
Iterative approach
Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
Iteration 6
R A D I
Test
Implement 3 features
Implement 3 features
Implement 4 features
Implement 5 features
Implement 6 features
Implement 4 features

Ideas

- Pair programming
- Integration testing at specific times each day

Style

- Names
  - Use descriptive names for globals, short names for locals
  - Be consistent
  - Use active names for functions
  - Be accurate
- Expressions and Statements
  - Indent to show structure
  - Use the natural form for expressions
  - Parenthesize to resolve ambiguity
  - Break up complex expressions
  - Be clear
  - Be careful with side effects

Style

- Consistency
  - Use a consistent indentation and brace style
- Function macros
  - Parenthesize the macro body and arguments

Style

- Magic numbers
  - Give names to magic numbers
  - Define numbers as constants, not macros
  - Use language to calculate the size of an object
- Comments
  - Don’t belabour the obvious
  - Comment functions and global data
  - Don’t comment bad code, rewrite it
  - Don’t contradict the code

Good clues, Easy bugs

- Look for familiar patterns
- Examine the most recent change
- Don’t make the same mistake twice
- Debug it now, not later
- Get a stack trace
- Read before typing
- Explain your code to someone else

No clues, Hard bugs

- Make the bug reproducible
- Divide and conquer
- Study the numerology of failures
- Display output to localize your search
- Write self-checking code
- Write a log file
- Draw a picture
- Use tools
- Keep records

Documentation

- Journal
  - Time oriented diary
  - Helps memory
  - Internal communication
- Report
  - Logical description of a problem and the solution.
  - Should enable evaluation of the solution.
  - What and how

Documentation

- Time schedule
  - A time schedule for the project containing duration and sequence of the project sub-tasks should be made.
  - Note that implementation and test always take longer than expected.
  - Utilize the fact that you are 3-4 persons on the team to divide the tasks.
  -
- Software tests:
  - Describe the tests results and improvements made.

- On the web
  - Results of the first week exercises

- On the poster
  - Description of the project

MATLAB figures on the web

- If you use jpg make sure that the ‘quality’ is 100 % or you get bad plots.
- Generally png (or gif) works well.
- print -rxxx -dpng name
- Use the following commands the produce a (better) jpg file:
- print -rxxx -djpg100 name
- For best results the size of the plot should be adjusted using the -rxxx option to print, NOT later in the html code.

Tasks

- Think->design->implement->test
- Decompose into smaller subtasks
- Test, test og test

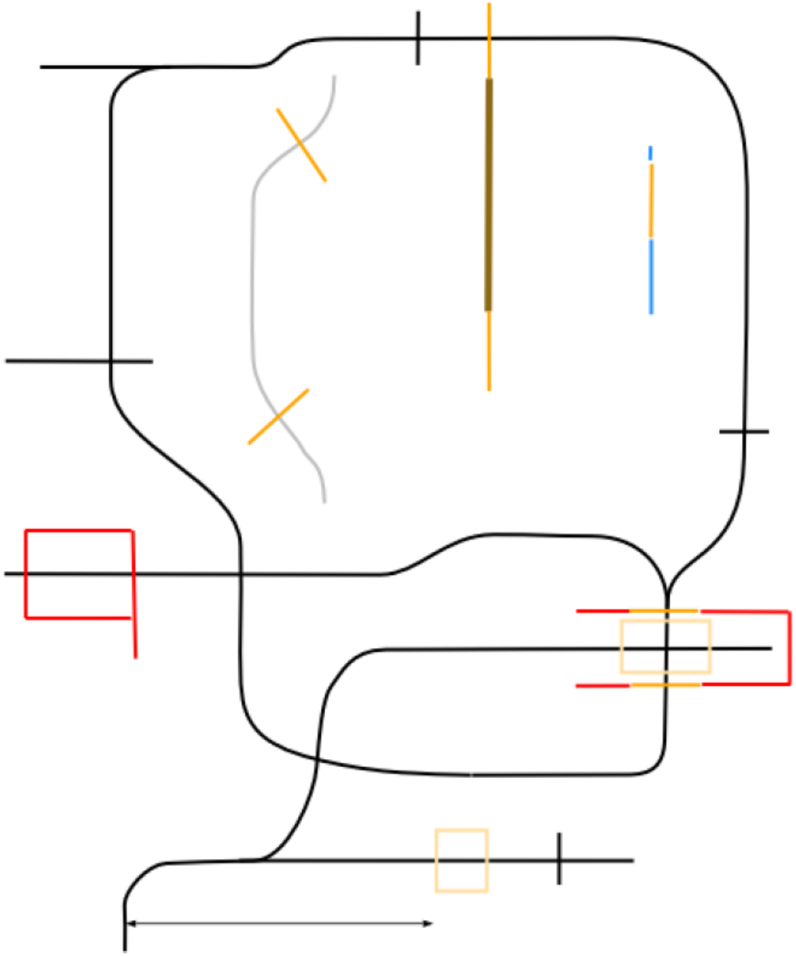
```
%goto "dock"
goto "box"
%goto "wall"
%goto "findgate"
label "dock"
......
label “box”
```

- Robustness

Autonomous systems

- Systems that are able to handle unforeseen events in a robust and elegant way
- Examples are mobile robots, industrial plants Heating systems

Sample Competition Track



Points in Robot Competition

There are 6 tests that each gives between 0 and 2 points.

- Distance measurement: The x-distance from start to box is measured. Error less than 3 cm gives 2 points, error less than 7 cm gives 1 point. The distance should be read aloud immediately after the measurement
- ’Box gate’, 1point pr. Gate.
- ’Gate on the loose’: 2 points for going through the gate.
- Wall: 1 point pr. Gate
- White line: 1 point pr. Gate
- Goal: 1 point for opening and 1 point for vehicle in ’tunnel’

Competition

- All robots must start at the appointed start point and end at the goal.
- After 180 sec or when the goal is reached points will be given according to the tests passed.
- Each team should make two simulator runs, one using SMR-CL and one using C. The final result is the mean of the two runs.
- Each team has two real robot runs, one using SMR-CL and one using C. The final result is the mean of the two runs.
- Each result is rounded to one decimal.