Software Engineering lab Team 8

Project Presentation

The Team

Vlad Frasineanu

Inti Mendoza

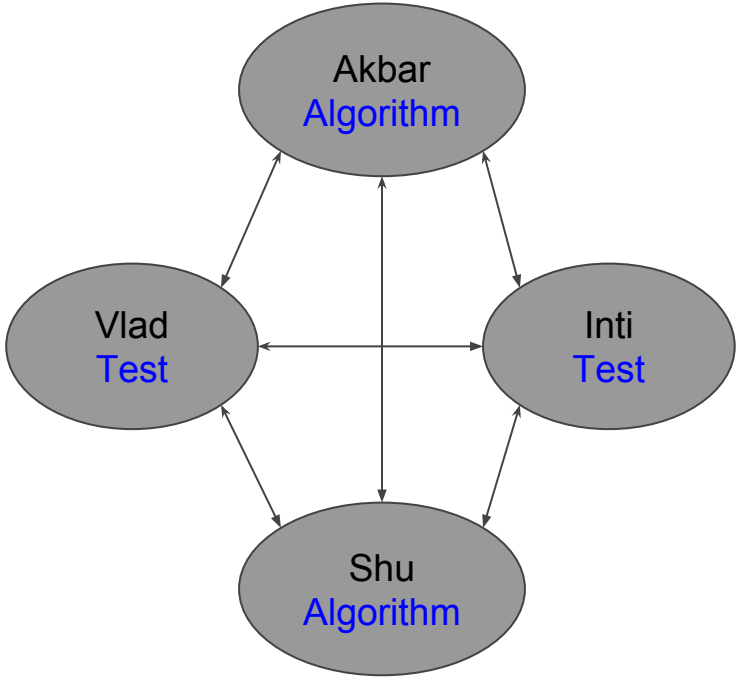
Akbar Oripov

Qifan Shu

Organization structure

Egoless (Democratic team structure)

- appropriate for less understood problems
- suitable for projects requiring less than five engineers
- programmers can share and review one another’s work



The Project

Robot with:

– laser range finder (LRF)

– tracked drive

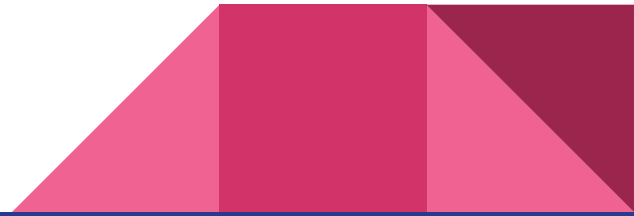
– Raspberry Pi

Playing field:

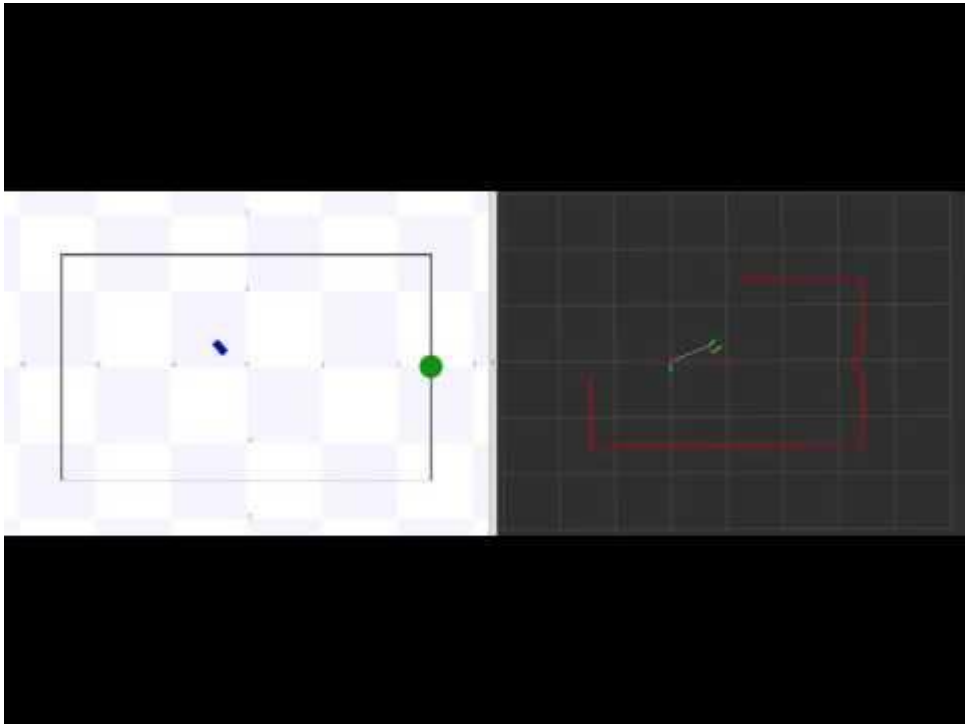
– given start position

– fixed position half-circle

Goal: find and touch half-circle without touching walls/obstacles

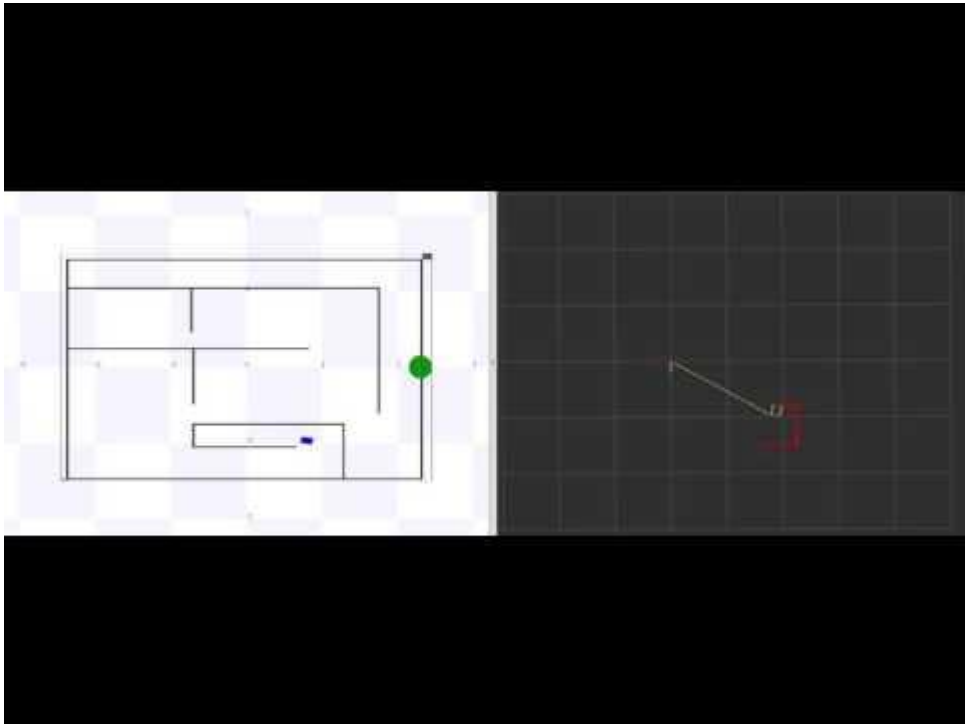


Basic version: free field



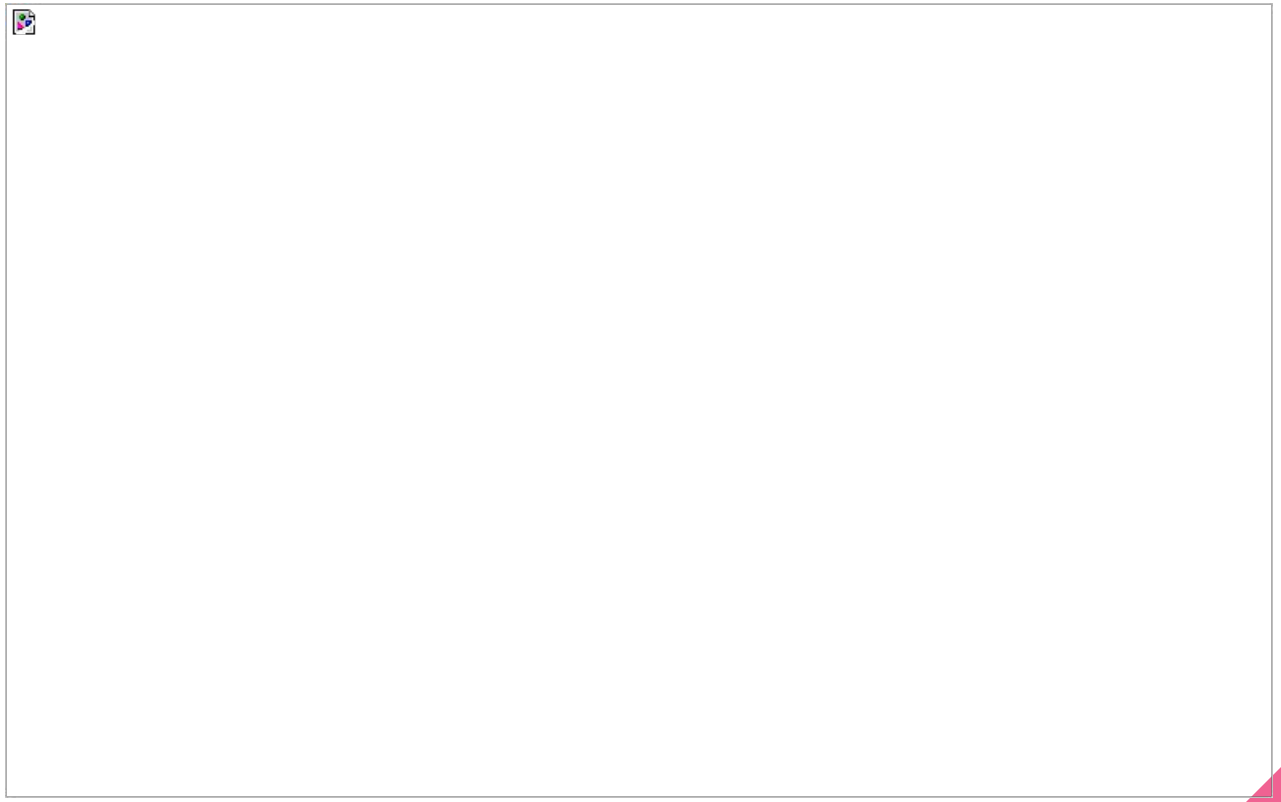
- Goal: Find the half-circle in a free field without hitting the walls
- Basic idea: Make a full scan (by rotating the robot) - if the goal is detected, then go directly to it, otherwise go in one direction until you find a wall and follow it until the detection of the half-circle.

Advanced version: with maze



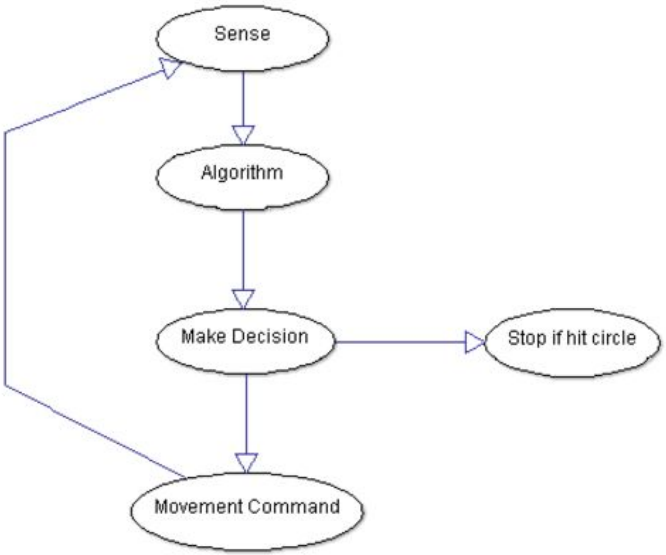
- Find the half-circle in a maze field without hitting the walls
- Basic idea: walk parallel with the walls at a some fixed distance and follow them until the scan recognize the goal

Structure in terms of ROS

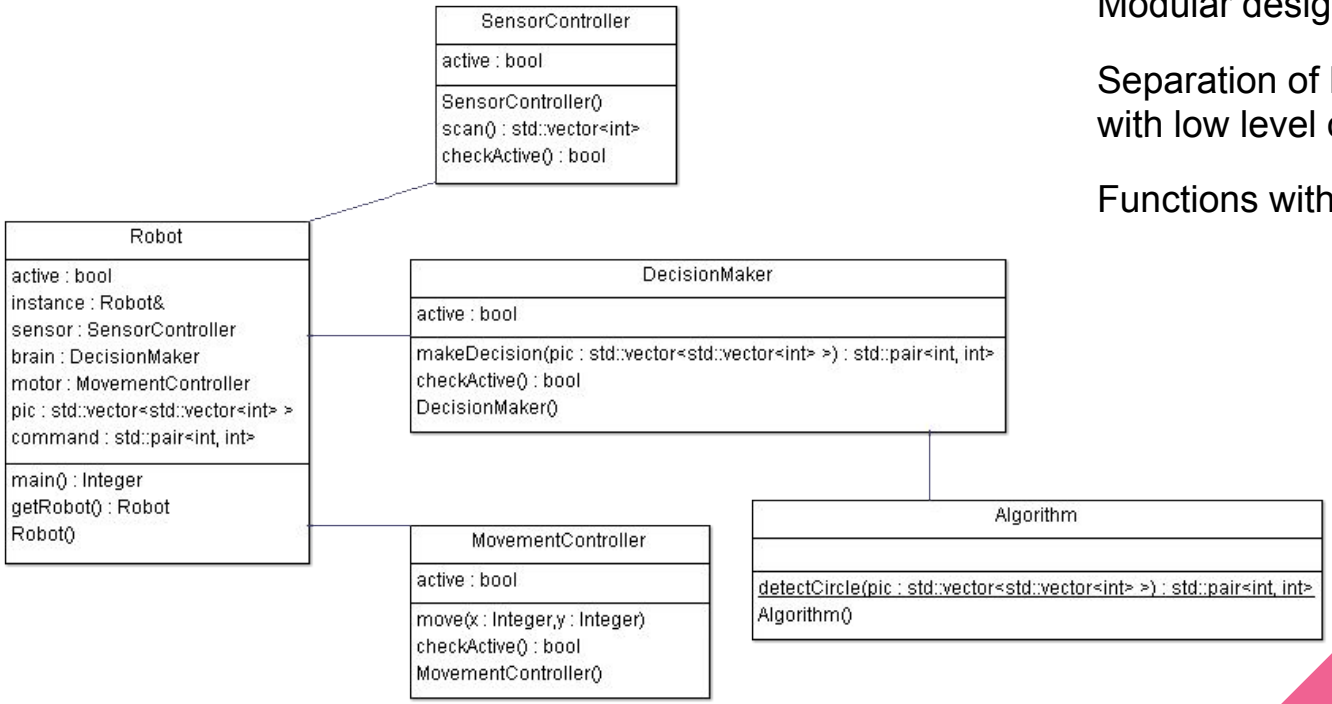


How does the software work?

- Sense module gets the data from the laser
- Manipulate the data such that it can be used by our algorithms
- The algorithms make a decision based on the received data
- If the circle is not detected, the decision is sent to the movement control which executes it
- After the movement command is done, the procedure is reloaded



Class Diagram



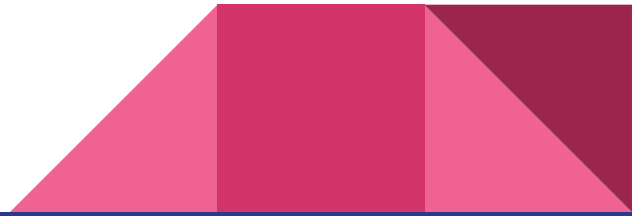
Modular design

Separation of high level functions with low level ones

Functions with simple functionality

Algorithms - object detection

- For corner detection: **Harris Corner Detector**
- For wall (straight line) detection: **Hough Line Transform**
- For semi-circle (arc) detection: **Hough Circle Transform**



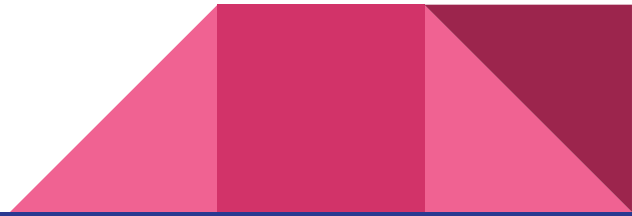
Libraries and packages

OpenCV - is a library of programming functions mainly aimed at real-time computer vision

- opensource
- provides optimized code for basic vision infrastructure
- provides all necessary algorithms for shape recognition that we want to use in our project

Testing methods

- Unit test - check all class methods and functions
- Integration test - combined modules tested as a group
- Validation test - check that the software system fulfills its intended purpose
- Performance test - check how the system performs in terms of responsiveness and stability



Scheduling

Tasks	% Done	Start	End	Effort PW
Code Sprint 1	**0%**	**Feb-24**	**Mar-8**	**8**
Documentation	0%	Feb-24	Mar-8	4
Sensor controller class	0%	Feb-24	Mar-1	1
Movement controller class	0%	Feb-24	Mar-1	1
Decision maker	0%	Feb-24	Mar-1	2
Robot class	0%	Mar-2	Mar-8	2
Test	0%	Mar-2	Mar-8	2
Code Sprint 2	**0%**	**Mar-9**	**Mar-29**	**8**
Documentation	0%	Mar-9	Mar-22	4
Algorithm for detecting walls	0%	Mar-9	Mar-15	4
Algorithm for detecting semi-circle	0%	Mar-16	Mar-29	2
Test	0%	Mar-16	Mar-29	2
Code Sprint 3	**0%**	**Mar-30**	**Apr-27**	**16**
Documentation	0%	Mar-30	Apr-27	4
Test on real robot	0%	Mar-30	Apr-5	4
Wall follower algorithm	0%	Apr-6	Apr-12	4
Test on mazes	0%	Apr-13	Apr-19	4
User manual, bug management, finalizing project	0%	Apr-20	Apr-26	4

Conclusion

Our design and overall strategy is...

- **Simple** - no need to write complex algorithms
- **Modular** - utilizes power of Object Oriented Programming
- **Expressive** - code intention is well defined
- **Follows Standards** - follows guidelines set
- **Clean** - follows Single Responsibility Principle

Thank You!

Questions?

References

http://nptel.ac.in/courses/106105087/pdf/m12L30.pdf

https://en.wikipedia.org/wiki/OpenCV

https://en.wikipedia.org/wiki/Integration_testing

Pictures

http://ecx.images-amazon.com/images/I/81HCJGn4HWL._SL1500_.jpg