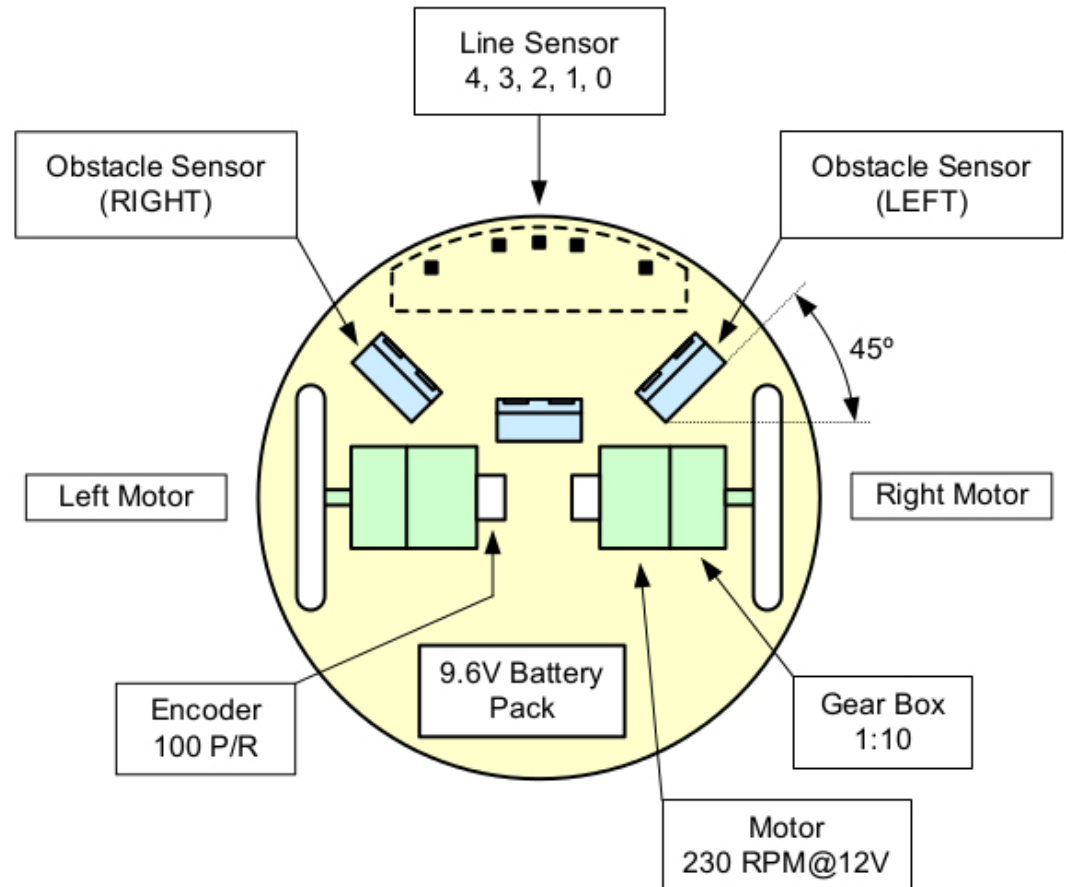# Development of an agent for a robot that solves labyrinths PathFinder

Intelligent and Mobile Robotics – Final Project

Vedran Semenski (75345), Aveiro, February 2015.

Intro

- C

- Basic text editor

- Robot

- pcompile

Line Sensor
4, 3, 2, 1, 0

Obstacle Sensor (RIGHT)

Obstacle Sensor (LEFT)

45º

Left Motor

Right Motor

Encoder 100 P/R

9.6V Battery Pack

Gear Box 1:10

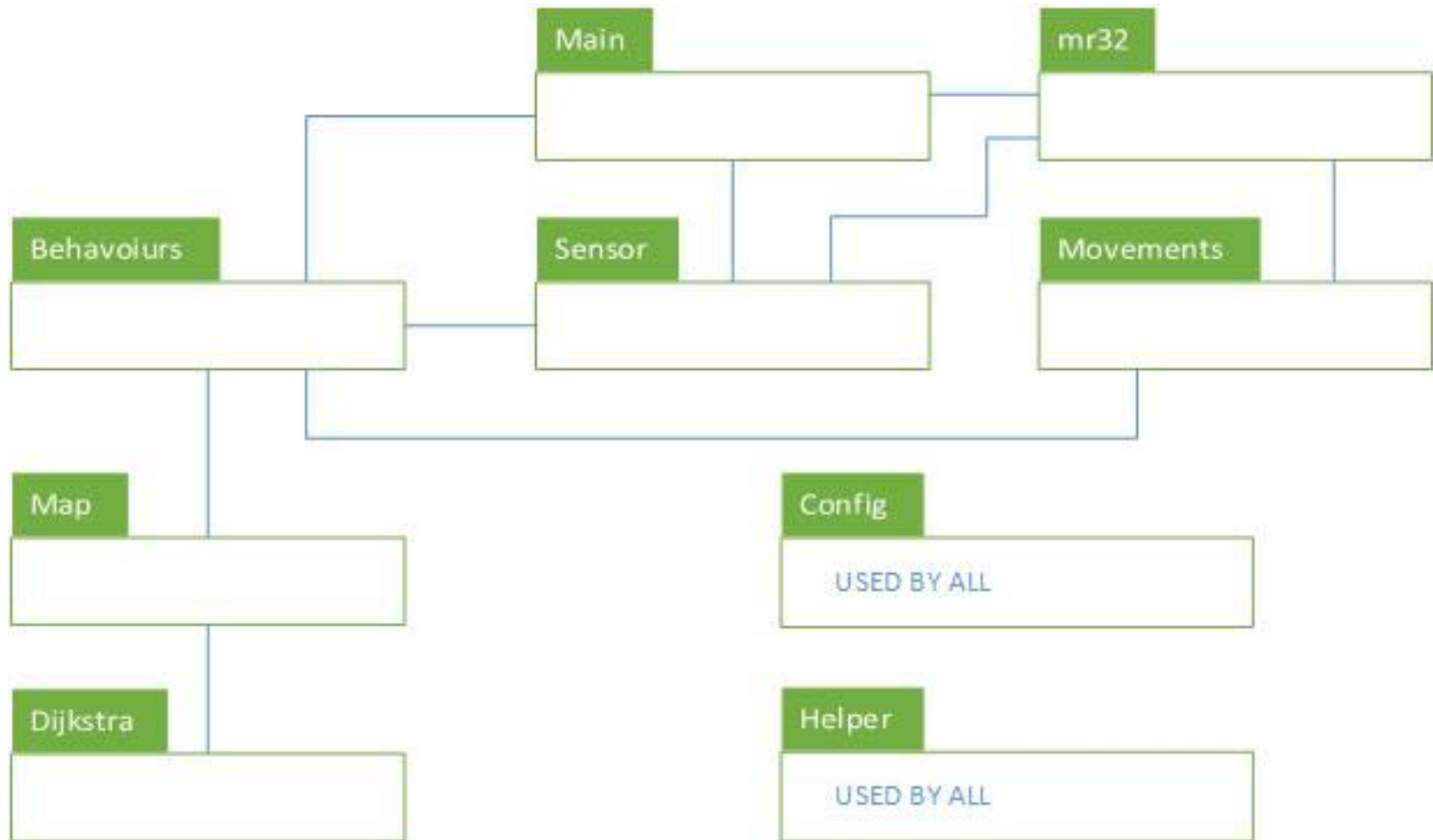Motor 230 RPM@12V

University of Aveiro, February 2015.

Basic concept

- **Abstracting the robot**

  1. Movements

  2. Sensors

- **Subsumption Architecture**

  - Prioritized list of behaviors (dinamic)

- **Behaviors**

University of Aveiro, February 2015.

Solution architecure

Sensor module

Feature list:

1. Compass
2. Circular buffer (5)
3. Normalization
4. Error compensation
5. Filtering readings
    1. Weighted average
    2. Median

University of Aveiro, February 2015.

Movements module

Feature list

1. Buffered movements

2. Many types of movements
   1. Forward
   2. Rotate
   3. Move in curve
   4. …

University of Aveiro, February 2015.

Map module

Feature list

1. MapField (Node) and MapConnection(Vertex)

2. Dijkstra's search algorithm

3. Checking for existing fields

4. Printing out map

5. Flexibe (number os connections, fields…)

University of Aveiro, February 2015.

Behaviours

Behaviour list

1. StopAtBeaconArea
2. AvoidCollision
3. StopAtStartingPoint
4. FollowPath
5. ReturnHome
6. ExploreLabyrinth
7. CorrectPosition

University of Aveiro, February 2015.

Behaviours

Explore labyrinth sub behaviours:

1. Move to next point
    1. Check if fields on sides are occupied
    2. Go to next behavoiur
2. Find next unexplored area
    1. If area isn't here set shortest path to unexplored area
    2. Go to next behaviour
3. Discover/Evaluate an unexplored field
    1. Get accurate readings
    2. Determine if field is free or occupied
        1. Occcupied – correct position and go to previous b.
        2. Free – set field as next field and go to first b.

Behaviours

Correct position:

1. Adjust angle and check if a wall is in front
2. Get sensor readings
3. Check distance from wall
4. Robot is to far or to close to the wall
    1. Report error offset to the sensor module
    2. Move to new position

University of Aveiro, February 2015.

Initial and return priority lists

Priority lists:

1. StopAtBeaconArea

2. AvoidCollision

3. CorrectPosition

4. FollowPath

5. ExploreLabyrinth

1. StopAtStartingPoint

2. AvoidCollision

3. CorrectPosition

4. FollowPath

5. ReturnHome

Workflow

▶ **BasicWorkflow:**

1. Initialisation

2. Start loop

    1. Refreshing sensor readings

    2. Testing behaviors

    3. Execution of behavior with highest priority

        1. Updating map

    4. Check if finished

3. End

University of Aveiro, February 2015.

Resulting behaviour

1. Initialisation
2. Robot moves from field to field
3. Checks is surounding fields are free or occupied
4. Updates the Map
5. Travels to unexplored areas by shortest path
6. When beacon is found - setup return settings
7. Calculate shortest path
8. Return to starting point
9. Print out Map
10. Finished

University of Aveiro, February 2015.

Development difficulties

1. Programming in C

2. Noisy sensor readings

3. Integrating the compass

4. Testing and changing the behaviour

5. Accumulation of error on the position readings

6. More reliable behaviour and readings = slower movement

Results and Limitation

Results:

- Good results
- Flexible
- Mapping
- Compensation for error
- Optimizations
- Code Documentation (Doxygen style)

Limitations

- Memory
- Sensor readings
  - Noise
  - Error accumulation
  - Speed

University of Aveiro, February 2015.