Development of the Mars Rover robot

Judith van Stegeren and Mirjam van Nahmen

December 19, 2014

Description of the work on the Mars Rover

The main goal of the robot is to find all three colored lakes and measure the temperature.

MoSCoW overview of requirements

MoSCoW	requirement
Functional requirements	The robot must be able to ...
M	find all lakes.
M	find a lake and determine its color.
M	navigate around lakes.
M	avoid falling off the table.
M	notice obstacles in the area and avoid them.
M	measure the temperature of all lakes.
M	see the difference between black, white, red, green and blue.
S	recognize bumping against an obstacles and drive around it.
S	show the overview with the information of all lakes.
Usability	The robot must ...
M	be programmed by stating its behavior in a DSL and then generating code from that specification.
S	not drive faster then 0,20 m/s , so that if the robot is heading for the edge of the table, the user still has time to interfere.
S	provide feedback for the user about the state of the program for debugging purposes.
C	provide feedback for the user about measurements it makes.
M	stop when the robot is finished.
M	provide feedback for the user when the robot is finished and the user press a button.
C	provide feedback for the user about errors/bugs.
Reliability	
M	The robot components must be tested before it will work with the whole system.
S	The robot must stop if an error/bug is detected to avoid to go off of the table.
S	The sensors must be calibrated before the program starts.
M	The robot must have enough power and memory to run the programs.
C	The robot must recognize its starting position.
Performance	
M	If the light sensors spot the white border, the robot must react immediately.
S	If the light sensors spot the color border of a lake, the robot must react immediately.
C	The robot should find all lakes in 10 minutes.
S	The robot should not measure the same lake twice.
Supportability	
W	The DSL can also be used for the generation of code in another language or for use with a different API.

Proposal deployment

List of the actuators:

2 motors

1 motor for the temperature sensor

1 lamp

List of the sensors:

1 temperature sensor

2 light sensors

1 color sensor

1 ultra sonar

2 touch sensors

There can be 3 actuators and 4 sensors on 1 brick. Since it is vital that the motor does not fall off the table, we want that the communication between the light sensors and the motors happens without bluetooth. Since collision with other objects is an important issue, it would seem logical to connect the touch sensors directly with the master brick as well.

Since the color sensor, the motor for the temperature sensor and the temperature sensor all have to do with the same functionality, it would seem logical to put these sensors and actuators on the same brick. Finally, as the measurement of the distance with the ultrasonic sensor is not crucial for the functionality of the robot, we suggest to put that sensor on the (slave) brick for the measurements as well.

Deployment diagram

brick	NXT1 (master)	NXT2 (slave)
tasks	wandering, navigating, avoiding collisions	recognizing colours, spotting far-away objects, temperature sensor control, lamp
actuators	motor 1, motor 2	temperature sensor motor
sensors	light sensor 1, light sensor 2, touch sensor 1, touch sensor 2	ultra sonic sensor, temperature sensor, color sensor

Actual deployment diagram

Brick 1 (Master):
crucial processes (wandering without collisions or falling off the table)

A	Motor left
B	Motor right
C	Lamp yellow
S1	Light sensor left
S2	Light sensor right
S3	Touch sensor left
S4	Touch sensor right

Brick 2 (Slave):
measurements (color of a lake, temperature and distance detection)

A	Motor Temperature
B	
C	Lamp green
S1	Color sensor
S2	Ultrasonic sensor
S3	Temperature sensor
S4	

Use cases

Wandering within white line The robot drives in a straight line until it encounters a white line with one of its light sensors. If it encounters the white line, it will back up a bit, make a turn away from the white line and proceed in a straight line. For this use case, we need both motors and both the light sensors.

New lake detection and measurements On detection of a colored lake, the robot first checks if it hasn't encountered the lake before. If not, the robot positions itself before the lake. It then lowers the temperature sensor, measures the temperature and stores it together with the color of the lake. We need the color detector, the motor for the temperature sensor and the temperature sensor itself.

Old lake detection On detection of a colored lake, the robot checks if he encountered this lake before. If this is the case, the robot drives around the lake. For this we need the color detector and the two motors, but it doesn't matter whether the color sensor is on the same brick as both motors.

Wandering without collisions The robot drives in a straight line. If one of the bumpers touches an object, the robot backs up, makes a turn and proceeds in a straight line. For this we need the bumpers and both motors. Since reaction time is important in this use case, it would be best if the motors and the bumpers are connected to the same brick. The ultrasonic sensor can also play a role in this use case. The reaction time of the ultrasonic sensor is less important since it can spot obstacles from further away.

Ready If the robot has measured the temperature in a lake, it check if all lakes have been measured. If that is the case, it stops wandering and shows the table with the temperatures of all lakes on his display.

Risk analysis

Id	Type	Description	Probability	Severity	Weight	Mitigation	Contingency
1	Technical risk	The computer with the installed software breaks down and we lose access to our work.	3	4	High	Push all work to the repository, regularly.	Find a replacement for the broken laptop or use one of the lab computers.
2	Organizational risk	The robot is not available for testing.	3	4	High	Regulary test small pieces of code, so that debugging is not a lot of work.	Postpone testing until later or wait until the robot is free. Use the lab when when we know there won't be other students around (early morning).
3	Technical risk	The DSL does not fit with the implementation.	3	2	Medium	Change the DSL to a lower level of abstraction.	
4	Organizational risk	One of our team members become ill.	3	2	Small	Healthy living	The other one continues working on the project and the ill person works as much as possible from home. Keep communicating via email/chat.
5	Requirements risk	The requirements contains ideas which are not compatible with the idea of the DSL.	2	2	Medium		Change requirements until it is possible to implement.
6	Technical risk	Problems with the bluetooth communication between the two bricks	5	3	Medium	Test bluetooth functionality as soon as possible.	Ask for help from the teacher or other students.
7	Organizational risk	Lab partners have different schedules which makes it hard to find time to work together	5	3	Medium	The lab partners don't plan anything on Wednesday afternoon, so that there is at least one moment in the week when they are both free to work on the project	Reschedule work or work together during the weekends.

DSL of the mars rover

We had two options for the structur of the DSL of the Mars Rover. First option was to create a DSL which is close to the behavior programming concepts of LeJOS. The advantage of this approach is the experience with during the last weeks of the class 'Design of embedded Systems' and the thread/task management is already made in form of the Arbitrator class.
The second option is the implementation of the state machine approach which was the first idea for our DSL. We see the advantages here in the abstraction level which can be create. That makes it easier for us to implement such complex things like the Bluetooth communication, where we had a lot of problems with the Arbitrator. The disadvantage is the implementation itself. We do not have many experience with it, such that it will be approximately cost more time to implement the robot.
The main reason why we chose for the state machine approach was the possibility to have more control over the threads and task of the robot and so a better overview what the does and why. Another reason is, that it is also more abstract and therefore also easier to make the implementation modular. That is in the future useful to use it again for comparable projects which is in the business world quite often the case. Furthermore seems it for us more interesting to implement a state machine with all the challenges.

The idea behind the DSL is a state machine. In the practice it is an often used system, which uses states and arrows to describe the transitions and conditions in which the robot can be.

Syntax

Instances for the Mars Rover

Main structure of the generated code

Realization of the Mars Rover

The first steps were to design and implement the basic structure of the DSL and the code generation. We decided to use the state diagram and designed the start/end state and the robot should be able to wander an area without falling off the table.

Final result