Universiteit van Amsterdam

Practical Project

Athena Line Follower

December 9, 2018

Students:
Rocco Andela
11745673

Anna Valachi
12301922

Nico Tromp
11699353

Sjoerd van der Heijden
10336001

Vasilis Gemistos
12318264

Coordinator:
Sebastian Altmeyer

Course:
Embedded Software & Systems

Course number:
5364EMSS6Y

1 Introduction - The fight for a new dawn

The year is 2018. Mankind is at the brink of extinction. Earths resources are almost depleted. The only way to survive is to colonize other planets. A candidate planet named Traal has been found. The best five engineers in the world have been put together to build a rover to explore the new planet.

The name of the courageous robot is Athena. Athena was a patron goddess associated with wisdom, handicraft, and warfare. These are the qualities that are need in an explorer that wanders the dangerous planes of the planet Traal.

The ultimate goal of Athena is to find a field of resources. However the planet is inhabited by the Ravenous Bugblatter Beast. The five engineers decided not to exterminate the whole population of Traal so they do not equip Athena with weapons. The only option, when the beast is encountered, is to flee.

2 Contributions

The engineers came up with a specific plan. The plan consists of some steps. First, the rover is manually driven to the starting position. Then autonomous mode is engaged. The robot drives forward until it encounters the Ravenous Bugblatter Beast. It has to stay at an safe distance of 10-12cm from the beast, beep for half a second and after 5 seconds it has to flee at a 90°angle to the right until it finds the black line that leads to the green field of resources. There it has to raise a flag and phone home to earth.

The engineers explored many different solutions before visiting the Traal Planet. They wanted to be sure that they were able to perform the tasks in the best possible way.

2.1 Responsibilities

Each of the engineers has their own field of expertise.

- Engineer Vasilis is an excellent communications expert. He was assigned the tasks of phoning home and raising the flag when the rover has found the resources. Whenever there is an encounter with a beast his work makes sure that Athena flees in the right direction.
- Engineer Anna worked one of the solutions for following the black line and made a library for the color sensor.

- Engineer Sjoerd took up the responsibility to combine all of the work of the other engineers in one complete package.
- Engineer Rocco worked on a second approach for following the black line using a PID controller. He was also responsible for the manual controls of the robot.
- Engineer Nico worked on a third solution for following the black line, a solution for detecting the green resources field and he was also responsible for making sure that the robot would drive completely straight in both the autonomous and manual mode.
- Engineers Vaslis and Nico worked on recognizing the safe zone once the black line is being followed.

2.2 Grade distribution

All of the engineers contributed equally to the project (ordered alphabetically on first name).

- 20% Anna Valachi
- 20% Nico Tromp
- 20% Rocco Andela
- 20% Sjoerd van der Heijden
- 20% Vasilis Gemistos

Not all contributions made it to the final version of the project so there might be a seemingly discrepancy between the final project and the provided code/stateflows.
Any simulink model/stateflow that is not enlisted in the 'project' view are models/stateflows that did not make it into the final version.

3 Motivation of model

3.1 Overall design

Most parts of the robot are created as separate libraries with one large model (*Athena.slx*) that combines these libraries into the final product. This modular setup makes changing or even replacing the smaller sub-systems very easy.
In the beginning, each engineer was responsible to create one library which

would control one of the aspects of the robot and send events. For example, they had created the distance sensor library which was, initially, sending back an event when the robot was close to the beast. When they tried to compose all the libraries, they run into the problem that Simulink does not allow subcharts to send events. To overcome this problem, they changed the output of each library and instead of sending back events they decided to send back variables of type *boolean* to the other parts of the system. By default, its value is **false** and when the library sends the 'event' its value is set to **true**. Any other part of the system that listens to this event defines an input variable (also of type *boolean*) and uses that in the guards for transitions between states. These two variables are bind with each other by means of the **Subchart Mappings...** feature of Simulink.

3.2 Line following

For the line following three approaches were designed.

- Approach 1 (Anna) involved a model purely written in Stateflow using the optical sensor in color mode. The robot kept following the black color. When it detected something else than black it would swivel in increasingly bigger arches until it found the black line again. When the black line was detected again it would keep track whether it was approaching the left or right side of the line in order to use it to swivel in that direction first whenever it would lose the black line again.

- Approach 2 (Rocco) involved a PID controller and the optical sensor in reflected light intensity mode. This approach allowed for accurately tracking the border of the black line. After some tuning of the PID values this approach seemed to work. However issues arose when integrating the external PID library in the final model.

- Approach 3 (Nico) involved a controller written in Stateflow and the optical sensor in reflective mode. This controller follows the black line on the left side. When the value from the sensor directs is to higher then a pre-set value (this happens when the robot is to far to the left) the robot is instructed to turn to the right. If the value from the sensor is to low (which means the robot is to much to the right) the robot is instructed to the left. The directional corrections are divided into three different steps. Depending on the difference between the current value from the sensor and the pre-set value the correction is either small, medium or large.
  Due to the geometry of the robot (the light sensor is on the right side of the robot) the correction for turning left and right is not completely symmetrical. When turning to the left the corrections for the motors

tend to be larger since a change is speed of the left wheel has a small impact on the position of the light sensor.

3.3 Safe zone detection (home detector)

The detection of the safe zone (green circle) is done by measuring the time the light sensor detects the expected 'green value' within a predefined range for a certain amount of milliseconds. The exact value for this time-out and the speed of the Traal rover needs to be carefully balanced. Engineer Sjoerd took care of balancing these values. Once the value from the light sensor falls outside the predefined range of 'green' the timer is being reset.

3.4 Line following and safe zone detection

In order to keep the Traal rover to drive in a straight line once the safe zone (the green circele) has been reached, the Traal rover is instructed to aim at the reflective value of the safe zone. This ensures that the (home detector) has enough time to detect that the safe zone has been reached.
To prevent false positives while the Traal rover follows the black line, it wiggles as a dug while driving. By wiggling the value of the light sensor continuously changes and thereby resets the timing of the safe zone detection.

3.5 Driving straight

When driving in a straight line the Traal rover tries to minimize the difference between the rotation encoders between the left and the right engine. Depending on the difference and the current power settings one of the engines is corrected.

3.6 Constants

Simulink offers the possibility to define 'constants'. We made use of this in multiple models, for example the maximum speed for a engine or the aiming value for the reflective sensor.

References

[1] MathWorks Documentation. Retrieve from
`https://nl.mathworks.com/help/`

[2] J. Sluka. (n.d). A PID Controller For Lego Mindstorms Robots. Retrieve from
`http://www.inpharmix.com/jps/PID_Controller_For_Lego_Mindstorms_Robots.html`

[3] Embedded Software and Systems 2018, Practical Project: Traal-Rover.