COMPLEX CYBER-PHYSICAL SYSTEMS DESIGN

Ansgar RADERMACHER, Asma SMAOUI
(based on work of Jérémie Tatibouët and Shuai Li)

mars

Future moon rovers could be teleoperated from Earth. (Image credit: NASA/Daniel Rutter)

”Teleoperated rovers could soon be working on the moon, with human controllers on Earth manipulating the rovers' tools virtually, allowing for greater dexterity when taking samples, digging or assembling..”[1]

We will design a small demonstrator inspired by the needs of a real robot as described above. The system consists of the following components which have partly already introduced in a previous exercise

- A controller running a real-time OS
- An ultrasonic sensor able to detect an obstacle and measure its proximity in centimeters (in combination with a software component, please note that the sensor also emits sound).
- Four servomotors: the motor has a built-in rotation sensor measuring speed and distance. Several motors can be aligned to drive at the same speed. The actuator can rotate forward and backward
- A temperature sensor.
- The heating system. The purpose of this component is to prevent electronic parts from getting too cold (lunar night has temperatures as low as -180°).

The light sensor that was part of the original system is no longer used.

[1] https://www.space.com/moon-mars-robots-remote-control-technology

Setup

Copy folder ExamStudentModel from the git

1. Import the project ExamStudentModel
2. Open the model.

The model is divided into a top-level class and three main packages:

1. **Rover**: this top-level package represents the system component to design
2. **Requirements**: this sub-package contains the requirements
3. **System**: this sub-package shall contain the definition of software components of your systems, including signal definition and interfaces
4. **PowerModel**: a sub-package for power-related calculations

Structure

System

System design requirements:

A **Rover** is composed of the following **software** components

1. A **commandSystem**. The main control component of the rover. It reads the temperature sensor, gets the distance to a potential obstacle from the detector, and controls the motion as well as steering wheels.
1. A **heating** module
2. An **obstacle detection** module
3. The **steering** subsystem and the motion wheels

Hardware design requirements

The previous document already mentions the following requirements:

Following are a few preliminary requirements:

- ▷ Rovers can be either directly operated by in-orbit astronauts or commanded by a program from an orbit station for routine tasks.
- ▷ The system is a four wheels rover, two of them are powered by a servomotor integrated into the wheel
- ▷ The rover must be able to steer left and right (2 servomotors control the angle of the two non-powered wheels)
- ▷ The rover must be able to go forward and backward.
- ▷ The rover sends operation errors to the remote base.
- ▷ Operations and errors must be logged and accessible at any time.
- ▷ The minimal and maximal power consumption (stored in an attribute of a constraint definition) must be greater than 0.
- ▷ The actual power consumption shall be less than or equal to the maximal power consumption.
- ▷ The actual power consumption shall be greater than or equal to the minimal power consumption.
- ▷ Power consumption for the steering of the Rover must be 60W +/- 5%.
- ▷ The steering angle of the left and right wheel must be between -1/3pi and 1/3pi.

In addition, consider the following ones

- ▷ If an obstacle is detected in the current path of the rover, the motors should stop. The control system should then log and error recalculate a new path.
- ▷ If the temperature is above 90°, the system should enter a sleep state (reducing energy consumption and thus internal heating) and only return from it, once the temperature drops below 70°.
- ▷ If the temperature is below -20°, the heating system should be activated.

Communication protocol

Use port definitions and, optionally, interface definitions in SysML v2 to define the communication between the different components.

System design instructions

Complete the **SysML v2 Architecture** model of the Rover. Only model the rover itself, not the environment (the mote base).

1. Add missing requirements
2. Add parts, ports, and connections. The ports should reference port definitions (to be added) and, optionally, interface definitions.
3. A state machine that describes the behavior of the movements (in reaction to remote commands). Please note that obstacles need to be detected, including error logging and path re-planning.
4. A state machine describing the heating system
5. Complete the power model. Add functions to calculate the power consumption from the resistance of the motors and the electrical current (use P = U x I and U = R x I)

Code generator (A subset of SysML v1 to SysML v2)

In this section, write a simple code generator that is capable of producing a SysML v2 specification from a SysML v1 requirements and blocks. The former only needs to support the “doc” part of a requirement. The latter should take parts, ports, and connectors into account. We will provide a UML student model that contains a subset of the rover system that you should use for testing. Compared to previous work in an exercise, we will notably add support for requirements and connectors.

SysMLv2 Metadata

In this section, write a SysML v2 Library (with metadata definitions) and use it to design the equivalent SysML v2 file (keep it simple, identify attributes from the part definition that could be part of the stereotype-like definitions).