10th ESA Workshop on Advanced Space Technologies for Robotics and Automation - ASTRA 2008

Description of the Locomotion Control Architecture on the ExoMars Rover breadboard

M. Höpflinger[(1)], A. Krebs[(1)], C. Pradalier[(1)], C. Lee[(2)], R. Obstei[(2)], R. Siegwart[(1)]

[(1)] *Autonomous Systems Lab*
Institute of Robotics and Intelligent Systems
Swiss Federal Institute of Technology (ETHZ)
CH-8092 Zurich (Switzerland)

{markus.hoepflinger, ambroise.krebs,
cedric.paralier, roland.siegwart}
@mavt.ethz.ch

[(2)] *von Hoerner & Sulger GmbH*
68723 Schwezingen
Germany

{lee, obstei}@vh-s.de

Abstract: The ESA ExoMars mission calls for the development of a Mars orbiter, a descent module and a surface mobility device, the ExoMars Rover. The Autonomous Systems Laboratory (ASL) of the Swiss Federal Institute of Technology Zürich has been chosen as team member of the ExoMars Rover Vehicle Chassis and Locomotion Subsystem Team lead by Oerlikon Space AG. Within the second project phase of the ExoMars project, phase B1, the ASL was responsible for the development of the Locomotion Control Architecture, the software integration and the performance testing of the rover breadboard. This paper will present an overview of the Locomotion Control Architecture and how the software has been integrated in the ExoMars Rover breadboard.

1. Introduction

The ExoMars Rover breadboard[1] is a six wheel drive and six wheel steered vehicle. The suspension system bases on a modified version of the RCL-E ([2]).
The Locomotion Control System Architecture has to allow to operate the rover breadboard as a stand alone item for the performance testing. As inputs, the locomotion control system has to take higher level information, e.g. waypoints as provided in later project phases by the Navigation System. The Locomotion Control System has to output position or velocity information to the motor controllers of the steering and driving motors. Because of the missing Navigation System, a Men Machine Interface (MMI) was developed to provide the Locomotion Control System with the necessary information and to log the data of the various sensors on the rover.
The software has been implemented first on the CRAB[2] breadboard (ETHZ, [3]) and finally on the ExoMars Rover breadboard (fig. 1). Table 2 lists the main dimensions of the two used rover breadboards.

[1]The ExoMars Rover breadboard produced by our consortium ([1]) in the context of the phase B1. For simplicity called ExoMars Rover breadboard in this paper.
[2]http://www.asl.ethz.ch/robots/crab

Figure 1: Images of the used rover breadboards on obstacles.
Left: ExoMars Rover Breadboard, Right: ETHZ CRAB

Figure 2: Dimensions of the breadboards

	ExoMars Breadboard	*ETHZ CRAB*
Mass	94 kg	34 kg
Track width	1.2 m	0.78 m
Distance between wheels	0.7 m	0.42 m
Wheel diameter	0.25 m	0.196 m

1.1 Computer hardware and software

A standard personal computer (x86 architecture) has been used as Electrical Ground Support Equipment (EGSE) . The MMI to send high level locomotion commands to the rover was executed on this standard PC. The software was implemented in C/C++ and uses the wxWidgets library for visualisation. A data logger writes the incoming data of the rover to files on the hard disk of the PC.
Two main tasks are running on the PC: A task for user input and visualisation and a task for the communication with the rover breadboard.

As on board computer (OBC) on the rover a computer board equipped with an Leon 2 processor (ESA, Gaisler Research) was used. The board has several interfaces, among which an ethernet, used to programm the board, and two SpaceWire, allowing the communication with the lower level hardware (breadboard motor controller and sensor interface electronics).
On the OBC, the operating system RTEMS (Real Time Executive for Multiprocessor Systems, [4]) is used. This operating system has been used for real time military or space applications (e.g. [5]). The operating system supports various platforms as the SPARC architecture of the Leon 2 board and uses a POSIX API for user applications. RTEMS in our case is used in single-processor mode. For the locomotion control software , eight different tasks are running with different priorities. Since there is no memory protection by the OS, exclusive access to memory has to be guaranteed by using semaphores. The synchronisation between the different tasks is event based.

2. Implementation of the locomotion control software

Figure 3 shows the overall architecture of the rover control. The rover level locomotion control sends high level drive commands (e.g. way point coordinates) to the subsystem. The OBC computes the lower level motor commands (e.g. steering motor positions) out of the driving commands. The motor commands are then transmitted via SpaceWire to the six wheel nodes manufactured by vH&S. The wheel nodes contain the motor power converter to power the motors and the electronics to interface the motor level sensors. The EBOX, another SpaceWire node, integrates the power supplies and electronics to interface the chassis level sensors as an inertial measurement unit (IMU).

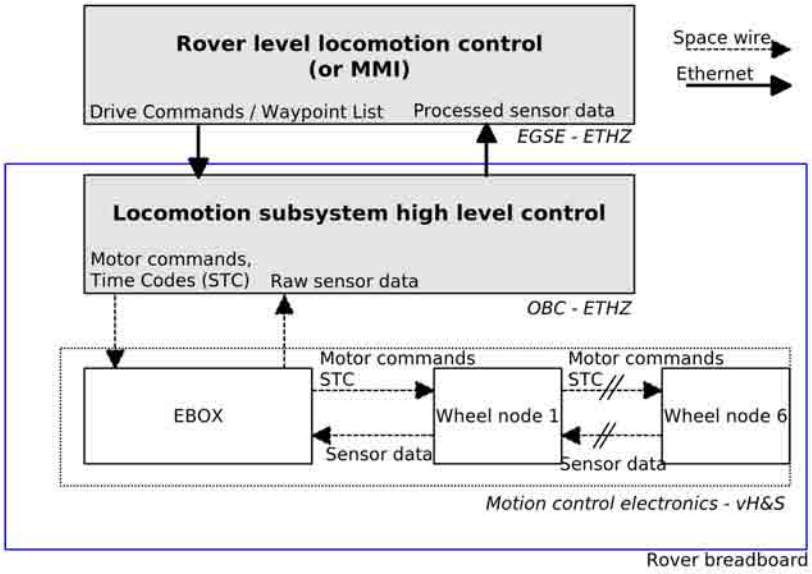

Figure 3: Data flow of the ExoMars Rover breadboard control architecture

Among other tasks the wheel node electronics handle the SpaceWire communication with the OBC and read out the sensor values of various sensors (see figure 4).

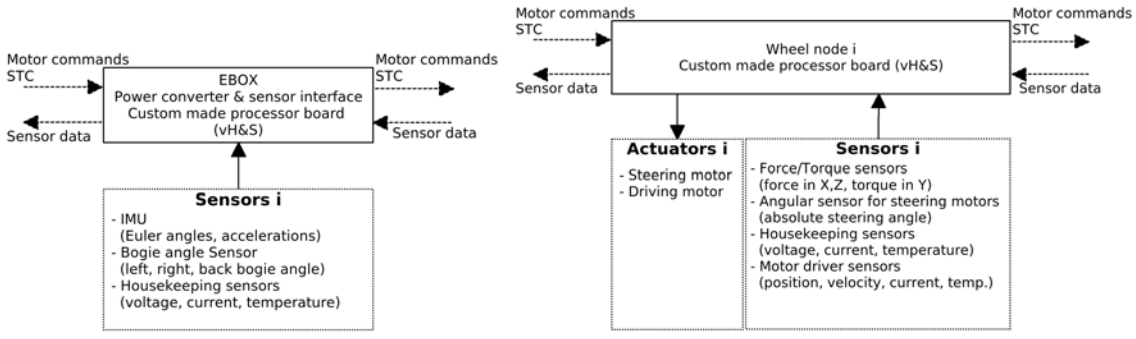

Figure 4: Left: Data flow between the EBOX and the chassis level sensors
Right: Data flow between the wheel nodes and the motor level sensors/actuators

Figure 5 shows a sequence diagram of the communication between the on board computer and the wheel node electronics. On receiving a SpaceWire time code (STC) from the locomotion control module, the wheel nodes respond by sending all the collected sensor data to the OBC. The STC

is sent once at every execution of the locomotion control module to every SpaceWire node. Wheel node one will first receive the STC and thus answers first. The OBC waits for 2 ms until all the wheel nodes had time to respond to the time code. Then it sends the low level motor commands to all the different wheel nodes.

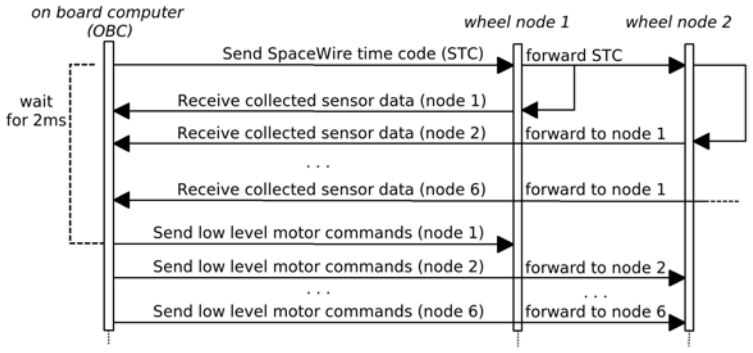

Figure 5: Sequence diagram of the communication between OBC and wheel nodes (without EBOX)

Figure 6 shows the most important tasks of the locomotion control software running on the Leon 2 board on the rover. The *Socket Manager* task is responsible to handle the communication with the EGSE PC.
The *Motion Control Manager* task executes a state machine to switch the control modes and control mode sub states. The *Motion Control Loop* task generates the low level motor commands and considers the rover data to decide on state changes. A desired state change is signalled by sending an event to the *Motion Control Manager* task. The *Hardware Rx Manager* task is responsible to receive and process RMAP ([6]) messages from the SpaceWire bus, thus making the new data available in the shared data structure for all other tasks.
The transmission of the message is done by the *Hardware Tx Manager* task. It sends the SpaceWire time code (STC) as well as the low level motor commands to the SpaceWire nodes.

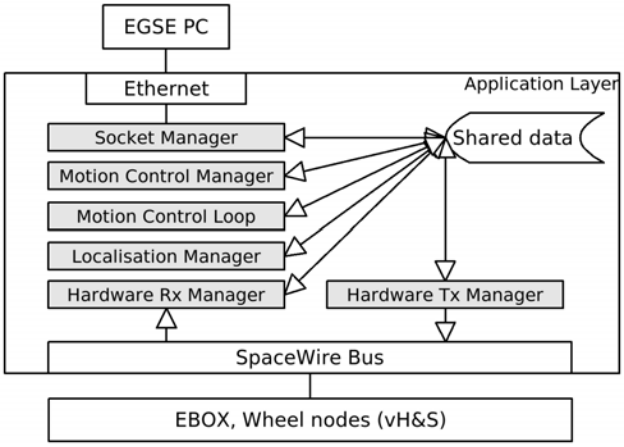

Figure 6: Simplified view of the main tasks run on the on board computer

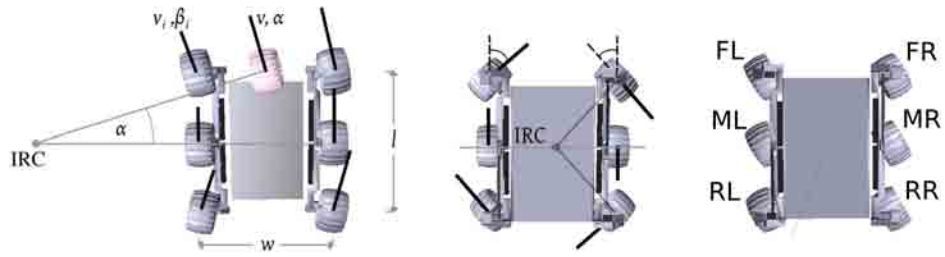

Figure 7: Drawing of the rover in different control modes: Normal (Ackermann), Turn on Spot, Lateral (Symbols: α, β_i: Steering angles; v, v_i: Driving speeds; FL, ML, RL, FR, MR, RR: Indices for wheels l: distance between front and rear wheel; w: track width)

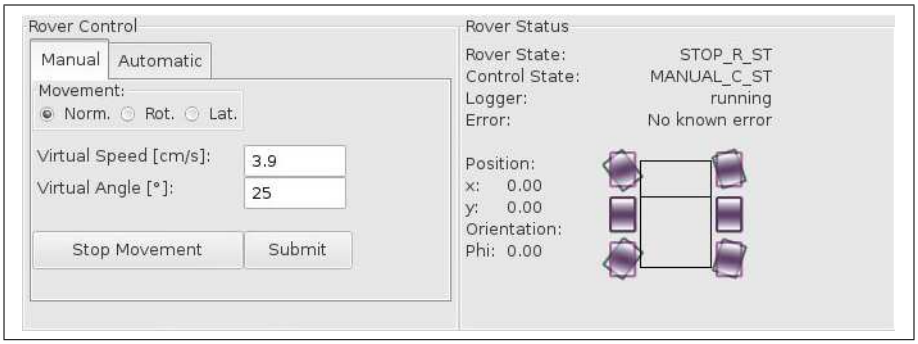

Figure 8: Part of the MMI GUI showing the fields to input parameters for the normal mode (ackermann steering)

2.1 Control modes

The locomotion control software allows to control the rover in manual and in automatic modes. In automatic mode the rover follows a given trajectory. In manual modes, which are of the number of three, the rover teleoperated directly by the user. Figure 7 shows a possible wheel configuration for three different control modes: *Ackermann* Mode (also called *Normal Mode* since it is the default control mode), *Turn on Spot* Mode and the *Lateral* Mode.

2.1.1 Input to control modes

The high level rover drive commands are generated by the GUI on user input (Figure 8). Table 9 lists the parameters the user has to input to control the rover in the given control mode.
For all manual modes, the user defines the motion of a virtual wheel (in this implementation the virtual wheel is located on the middle of the front axle of the rover (see first schematics of figure 7)). With the restriction that the instantaneous rotation centre (IRC) is located at the middle wheel axle of the rover, the IRC is defined by the steering angle α of the virtual wheel and the rover dimensions. The angles of the steering wheels and the rotational speeds of the driving wheels depend on the rotational speed v and the steering angle α of the virtual wheel: $\beta_i = f(\alpha)$, $v_i = f(v, \alpha)$.

Figure 9: Table showing the input parameters for the MMI GUI

Ackermann Mode	*Turn on Spot* Mode	*Lateral* Mode	*Automatic*
Virtual speed [cm/s]	Rot. speed [deg/s]	Speed [cm/s]	Waypoint list [m]
Virtual steering angle [deg]	Rot. Angle [deg]	Steering Angle [deg]	

2.1.2 Ackermann *Mode*

In *Ackermann* Mode, the rover wheels are steered so that the rover rotates around a given instantaneous rotation centre (see figure 7). The steering angles are calculated by the following formulas:

$$\beta_{FL} = -\beta_{RL} = arctan\left(\frac{l}{l/tan(\alpha) - w}\right) \qquad \beta_{FR} = -\beta_{RR} = arctan\left(\frac{l}{l/tan(\alpha) + w}\right) \qquad \beta_{ML} = \beta_{MR} = 0$$

The rotational speed of the driving wheel can be computed as follows:

$$v_{FL} = v_{RL} = v \cdot \frac{sin(\alpha)}{sin(\beta_{FL})} \qquad v_{FR} = v_{RR} = v \cdot \frac{sin(\alpha)}{sin(\beta_{FR})} \qquad v_{ML} = v \cdot \frac{\sin(\alpha)}{\tan(\beta_{FL})} \qquad v_{MR} = v \cdot \frac{\sin(\alpha)}{\tan(\beta_{FR})}$$

2.1.3 Turn on Spot *Mode*

The second schematics shows the rover in *Turn on Spot* Mode. The IRC is located at the centre of the rover. This allows the rover to turn on spot.
The steering angles are calculated as follows:

$$\beta_{FL} = -\beta_{RL} = -\beta_{FR} = \beta_{RR} = -\arctan(\frac{l}{w}) \qquad \beta_{ML} = \beta_{MR} = 0$$

The speed of the driving wheels is given by the following formula:

$$v_{FL} = v_{RL} = -v_{FR} = -v_{RR} = -\frac{\omega}{2} \cdot \sqrt{w^2 + l^2} \qquad v_{ML} = -v_{MR} = -\omega \cdot \frac{w}{2}$$

where ω corresponds to the rotational speed of the rover.

2.1.4 Lateral *Mode*

The third schematics of figure 7 shows a rover in *Lateral* Mode.
By steering all the wheels, the rover is able to move laterally. In this control mode, all the steering angles and the rotational velocities of the driving wheels are the same:

$$\forall i \quad \beta_i = \alpha$$
$$\forall i \quad v_i = v$$

2.1.5 Automatic mode

In the automatic mode, the rover follows a given trajectory. The trajectory can be generated with a waypoint editor of the GUI. After generation, the waypoint list is transmitted to the rover. By comparing the estimated position through wheel odometry and the target position, the rover motion can be controlled. It was specified, that the rover should follow the waypoints by driving along arcs connecting two waypoints. This restriction simplifies the complex problem of trajectory generation.

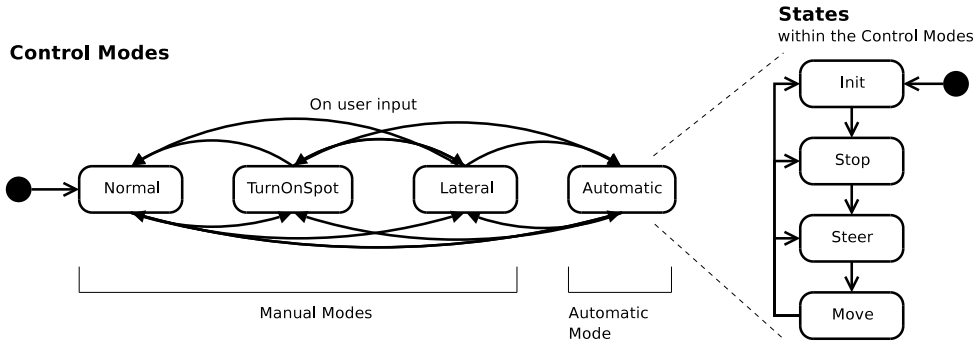

Figure 10: Control modes and sub states of the control modes

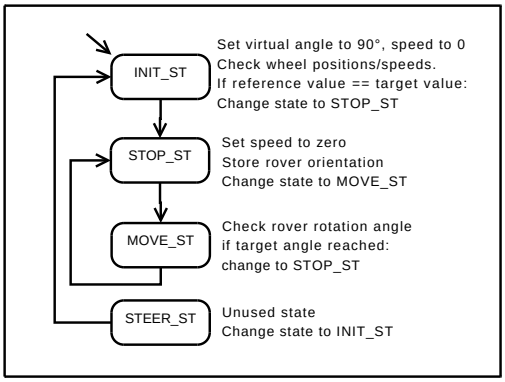

Figure 11: Simplified state machine for *Turn on Spot* Mode

2.2 Description of the control sub states

Every control mode includes four sub states: *INIT* State, *STOP* State, *STEER* State and *MOVE* State (see figure 10). In *INIT* State, the control mode is initialised. After the initialisation, the rover is ready to start the motion.

Figure 11 shows an example of a state machine with the conditions for transition from one state to another. In *INIT* State the rover is commanded to stop the driving motors and to rotate the steering wheels so that a turn on spot is possible. As soon as the wheels are steered correctly, the state is changed to STOP. By comparing the rover input it is verified that the rover speed is zero. Then the actual orientation of the rover is recorded and the state changed to MOVE. The rotational velocity of every wheel is calculated and transmitted to the wheel nodes. While in *MOVE* State, the robot turns on spot until the integrated variation of orientation reaches the target angular distance to be travelled.

3. Conclusion and Future work

The locomotion control software has been implemented and tested on the CRAB breadboard as well as on the ExoMars Rover breadboard. The ExoMars Locomotion System test campain showed the ease of the usage of the software and the robustness. The software recorded valuable data to support the engineering phase of the flight model.
It could be previewed and observed that the implementation of the path follower limits the user definable waypoints. For better path following capabilities, the specification should be changed in that way, that the rover not only follows arcs between two waypoints. This would induce an infinite number of possible trajectories between two waypoint. The infinite number of solutions would have to be reduced by optimisation, which could require more data about the environment.
For Phase B2, a visualisation tool has been developed. This greatly reduces the effort to test and debug the locomotion control module.

4. Acknowledgement

This work was supported by ESA and Oerlikon Space AG in the context of the project *ExoMars Rover Vehicle*.

References

1. C. Lee, J. Dalcolmo, S. Klinkner, L. Richter, G. Terrien, A. Krebs, R. Siegwart, L. Waugh, C. Draper, "DESIGN AND MANUFACTURE OF A FULL SIZE BREADBOARD EXOMARS ROVER CHASSIS", 9th ESA Workshop on Advanced Space Technologies for Robotics and Automation, 2006
2. V. Kucherenko, A. Bogatchev, M. van Winnendael, "Chassis Concepts for the ExoMars Rover", 8th ESA Workshop on Advanced Space Technologies for Robotics and Automation, 2004
3. T. Thueer, P. Lamon, A. Krebs, R. Siegwart, "CRAB-Exploration rover with advanced obstacle negotiation capabilities" 9th ESA Workshop on Advanced Space Technologies for Robotics and Automation, 2006
4. T. Straumann, "OPEN SOURCE REAL TIME OPERATING SYSTEMS OVERVIEW", 8th International Conference on Accelerator & Large Experimental Physics Control Systems, 2001
5. T.A. Ely, C. Duncan, E.G. Lightsey, A. Mogensen, "Real Time Mars Approach Navigation aided by the Mars Network", American Institute of Aeronautics and Astronautics, 2006
6. S. Parkes, C. McClements, "SpaceWire Remote Memory Access Protocol", University of Dundee, 2005