IoT4CPS – Trustworthy IoT for CPS

FFG - ICT of the Future
Project No. 863129

Deliverable D6.1b
First Prototype demonstrator for safe and secure automated driving platform

The IoT4CPS Consortium:
AIT – Austrian Institute of Technology GmbH
AVL – AVL List GmbH
DUK – Donau-Universität Krems
IFAT – Infineon Technologies Austria AG
JKU – JK Universität Linz / Institute for Pervasive Computing
JR – Joanneum Research Forschungsgesellschaft mbH
NOKIA – Nokia Solutions and Networks Österreich GmbH
NXP – NXP Semiconductors Austria GmbH
SBA – SBA Research GmbH
SRFG – Salzburg Research Forschungsgesellschaft
SCCH – Software Competence Center Hagenberg GmbH
SAGÖ – Siemens AG Österreich
TTTech – TTTech Computertechnik AG
IAIK – TU Graz / Institute for Applied Information Processing and Communications
ITI – TU Graz / Institute for Technical Informatics
TUW – TU Wien / Institute of Computer Engineering
XNET – X-Net Services GmbH

For more information on this document or the IoT4CPS project, please contact:
Mario Drobics, AIT Austrian Institute of Technology, mario.drobics@ait.ac.at

Document Control

Title:	First Prototype demonstrator for safe and secure automated driving platform
Type:	public
Editor(s):	Edin Arnautovic
E-mail:	edin.arnautovic@tttech.com
Author(s):	Denise Ratasich, Edin Arnautovic, Heribert Vallant, Christoph Schmittner
Doc ID:	D6.1b

Amendment History

Version	Date	Author	Description/Comments
V0.1	06.11.2019	Edin Arnautovic, Denise Ratasich	Initial version prepared
	26.11.2019	Heribert Vallant	Internal review
	04.11.2019	Christoph Schmittner	Internal review
V1.0	12.12.2019	Edin Arnautovic	Final

The IoT4CPS project is partially funded by the "ICT of the Future" Program of the FFG and the BMVIT.

Federal Ministry
Republic of Austria
Transport, Innovation
and Technology

FFG
Forschung wirkt.

Content

Abbreviations

ADAS	Advanced Driver Assistance Systems
AEB	Automated Emergency Braking
API	Application Programming Interface
ASIL	Automotive Safety Integrity Level
CPS	Cyber-Physical Systems
ECU	Electronic Control Unit
ECU	Electronic Control Unit
ROS	Robot Operating System
SHSA	Self-Healing by Structural Adaptation

Executive Summary

This deliverable is an accompanying report to the first demonstrator for a safe and secure automated driving platform. It contains a description of the integrated platform as well as the mobile rover robot demonstrator. The mobile rover robot is used as an example of the vehicle computing architecture used for automated driving containing a network of computers (electronic control units - "ECUs") providing related information by various sensors. From use-case perspective a *Collision Avoidance application is demonstrated*: stop when minimum distance to obstacle falls below a threshold. Software and hardware architecture are described elaborately in deliverable D6.1a: "Architecture for safe and secure automated driving platform demonstrator" and the scientific background is outlined in [RPSG2018] as well as the overall description of the prototype demonstrator. The focus was on integrating simple driving and obstacle avoidance functions on a safety platform. These functions represent the functions to be deployed in future vehicles. Future work will focus on further integration of software features of the platform and other results from WP3, WP4 or WP5.

1. Introduction

This document is an accompanying report to the first demonstrator for a safe and secure automated driving platform. Consider an automated car on the highway which tries to avoid collisions with the car in front of it. Typically, such a car uses range measurements (e.g., radar or laser) to emergency stop when a safety margin is violated. The safety margin will be chosen such that the car stops before crashing into the obstacle considering its current velocity and distance towards the car ahead. Observation components of a cyber-physical system (CPS) – such as the range finder of the autonomous car – may fail due to internal or external influences. Examples are timing or concurrency issues (e.g. late data from the range finder), hardware or software errors (e.g. missing communication link, platform/task crash), unexpected environmental conditions (e.g. rain) or inappropriate usage (e.g. incorrect mounting angle of the radar). Since these components provide inputs to CPS controllers, the CPS may fail (the car crashes into the car in front) or its performance, reliability or usability may considerably decrease [RPSG2018].
This document contains a description of the integrated platform as well as the demonstrator the mobile robot. A mobile rover robot is used as an example of the vehicle computing architecture used for automated driving containing a network of computers (electronic control units - "ECUs") providing related information by various sensors. From the use-case perspective a *Collision Avoidance* application is demonstrated: stop when minimum distance to obstacle falls below a threshold.

2. First demonstrator for safe and secure automated driving platform

The first demonstrator showcases the safety-related platform with the use case of avoiding collisions with obstacles and moving objects demonstrated in a lab environment. Software and hardware architecture are described in deliverable D6.1a: "Architecture for safe and secure automated driving platform demonstrator" and the scientific background is outlined in [RPSG2018] as well as the overall description of the prototype demonstrator.
The demonstration is built with the mobile rover robot but the demonstrated concepts and technologies are applicable to real automotive environment and should be exploited for automated driving and ADAS systems in the future. Figures 1 and 2 show the mobile rover robot equipped with embedded computers and sensors. The rover is capable of driving autonomously or can be tele-operated via a computer.

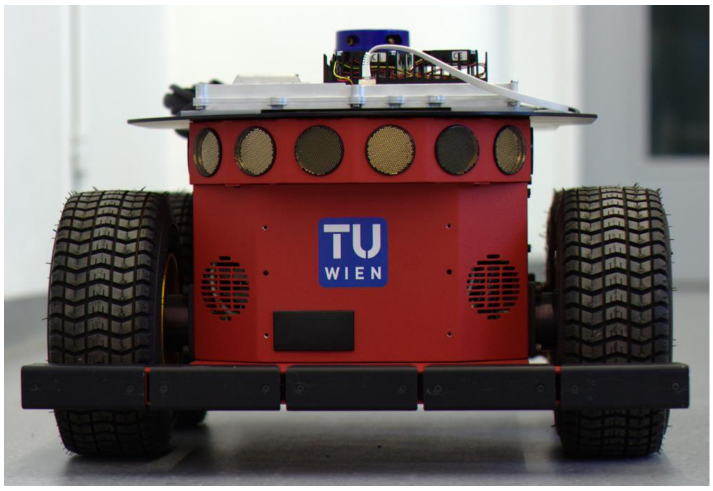

Figure 1: Front view of the mobile rover robot showing sonars.

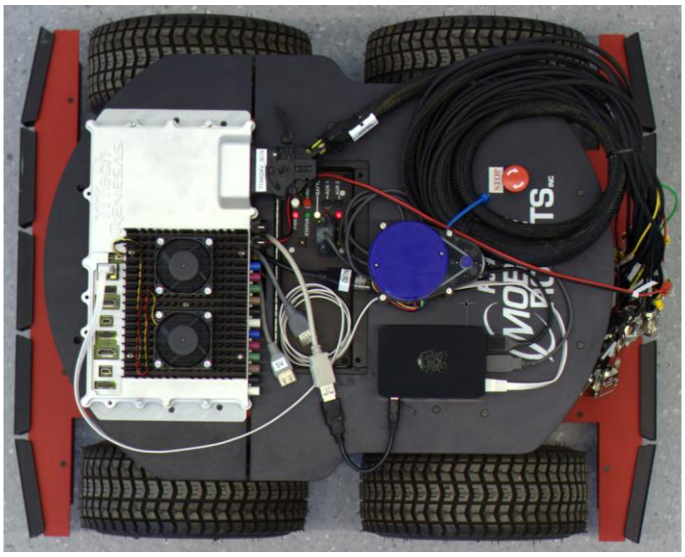

Figure 2: Mobile rover robot equipped with TTTech's safety platform, a Raspberry Pi and the LIDAR measuring the distance to obstacles.

As described in D6.1a the demonstrator contains two computing platforms, a TTTech's safety platform and a Raspberry Pi and the collision avoidance applications are deployed in a distributed manner.
The hardware architecture of the safety platform consists of different computing modules: a safety microcontroller ans two high-performance CPUs based on ARM architecture (Figure 3). These devices are connected by a Deterministic Ethernet (DE) switch. External interfaces such as CAN or Ethernet are also provided.

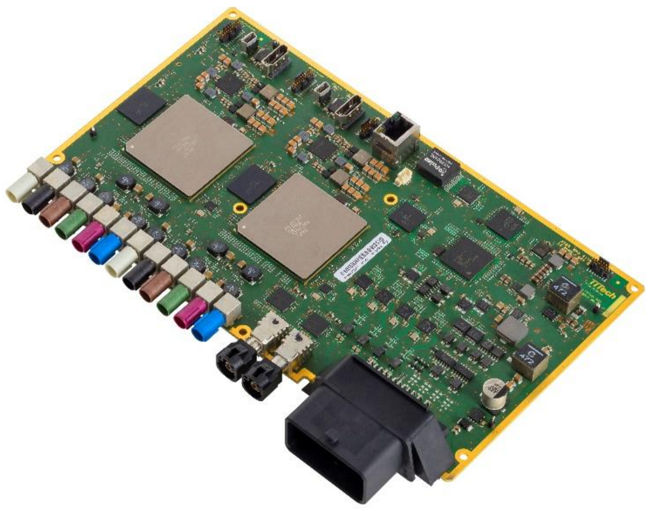

Figure 3: Safety platform (D6.1a)

Figure 4 shows the platform's external interfaces with 4 groups of interface connectors: (1) Vehicle connector (power feed, communication busses, IOs) on the far right, (2) Video input connectors (camera inputs) on the left, (3) Video output connectors (display outputs for dashboard and infotainment) in the middle, and (4) Programming and debugging interfaces connectors (internal, accessible via a hatch in the housing) on the top.

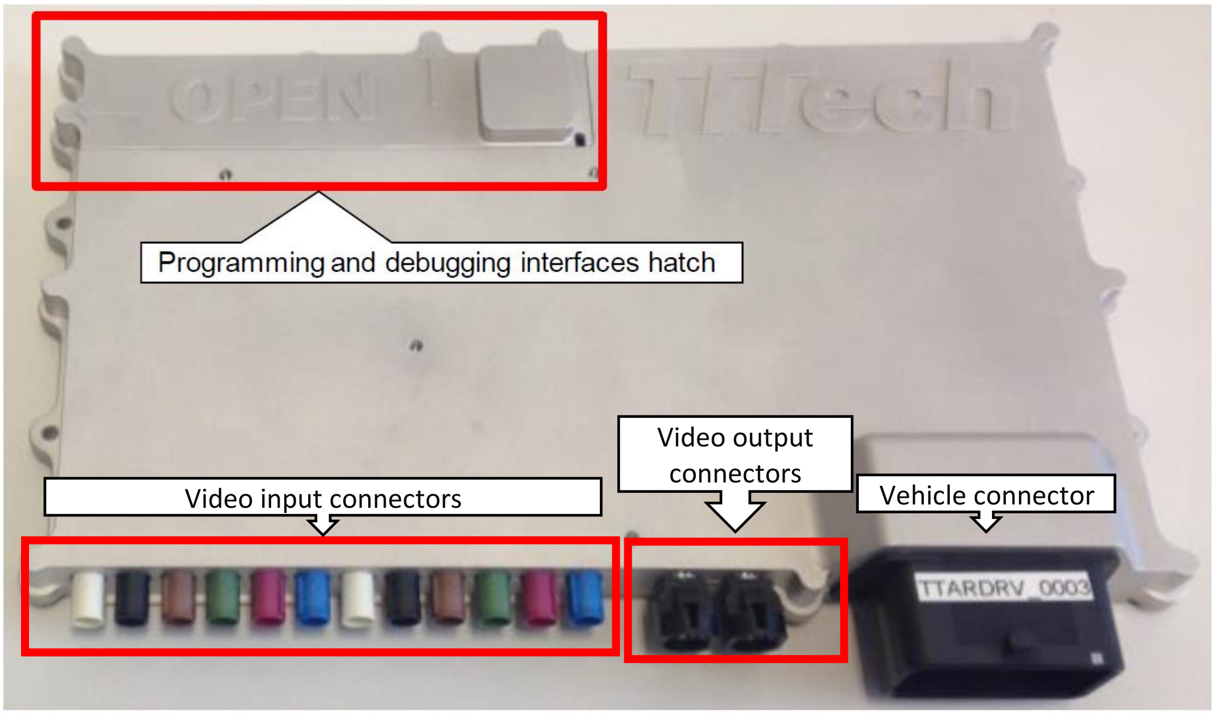

Figure 4: Safety platform

Both platforms run Linux and the Robot Operating System [ROS as middleware connecting different software components (ROS nodes). Nodes communicate via a message-based interface over TCP/IP. In particular, ROS nodes subscribe and publish to ROS topics (cf. named channels). ROS can start new nodes and reconfigure the communication flow of existing nodes during runtime. For example, there are nodes for lidar data */ydlidar_node,* calculation of the minimal distance */dmin_calculator* or a watchdog */min_watchdog* (as shown Figure 5).

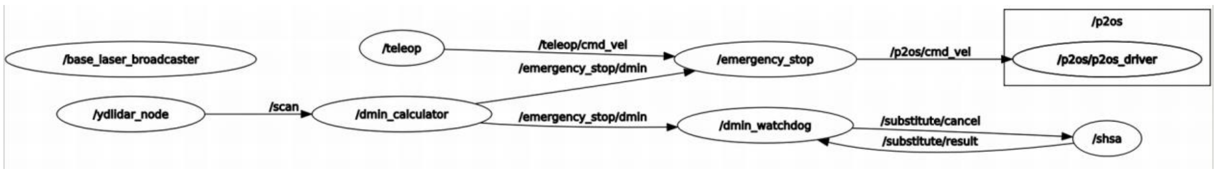

Figure 5: ROS nodes

The application nodes are distributed across two hosts where more critical tasks are planned to run the safety platform. Figure 6 shows different hosts and the distribution where components or nodes represent drivers to the sensors and actuators, as well as controllers. For instance, the node *emergency_stop* is a “critical” ROS node subscribing and publishing topics and it runs on the safety platform. The motors of the rover are controlled by a microcontroller *Robot uC*. The Pi connects to the microcontroller and LIDAR via UART. A controller running on a computer sends the desired linear and angular velocity (v,ω) to the robot’s microcontroller (Robot uC) controlling the wheel motors. The LIDAR (or laser scanner) on top of the rover provides distance measurements of 360°. When the minimum distance in front of the rover (dmin) – calculated by another ROS node dmin_calculator – falls below a threshold, the mobile rover robot is stopped and the velocity commands from the controller are replaced by (0,0) (which is implemented by the node emergency_stop). Acceptance of the controller commands is resumed when dmin again exceeds the threshold.

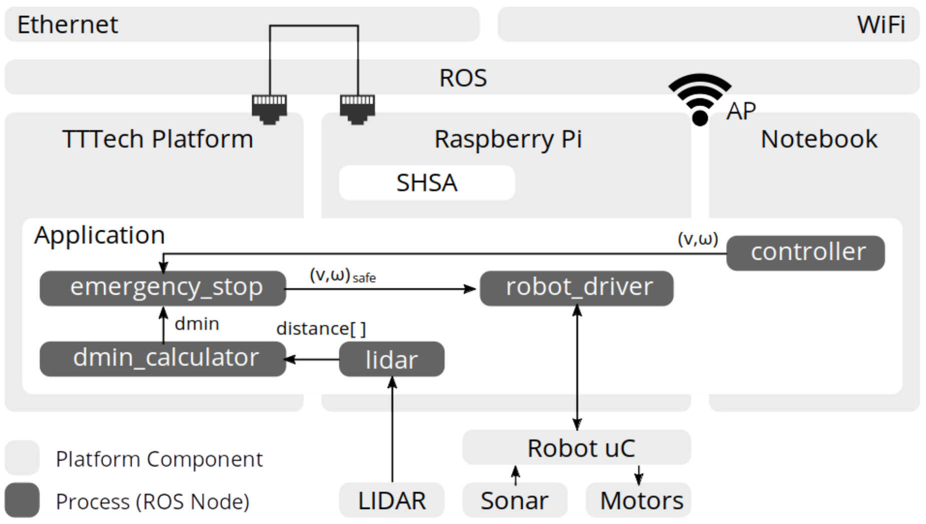

Figure 6: Platform overview and nodes of the application.

The scenario demonstrated is the application of (robot) automated driving with collision avoidance and the case of a safety-relevant failure of a lidar sensor. The system contains a smart watchdog to monitor the functionality of components and a self-healing engine The failure is simulated by powering off the laser scanner (Figure 7). The failure is detected and healed by the SHSA (Self-healing by structural adaptation, see D6.1a), by replacing a failed component with a substitute component.. After the failure is detected, a substitute node is generated, and the sonar is used to measure the distance. Sonar is considered as an emergency operation mode (for an utilization in the field, timings for fault detection / transition to sonar operation will be specified).

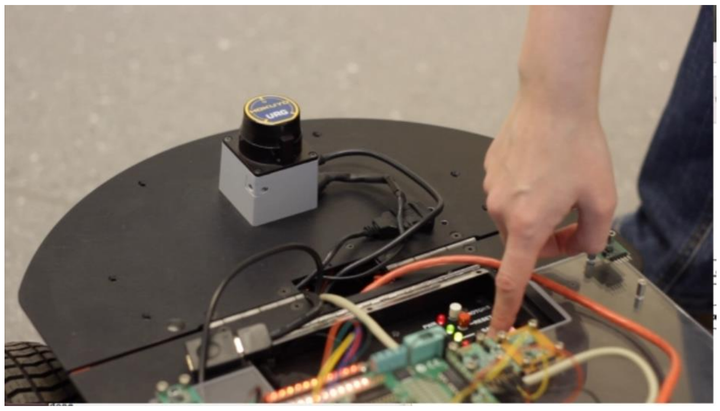

Figure 7: Simulation of a laser failure.

3. Conclusion

This report has given some details about the hardware and software of the first platform demonstrator. The focus was on integrating simple driving and obstacle avoidance functions on a safety platform. These functions represent the functions to be deployed in future vehicles. Future work will focus on further integration of software features of the platform and other results from WP3, WP4 or WP5.

4. References

[ROS] M. Quigley et al., “ROS: an open-source Robot Operating System,” in ICRA workshop on open source software, 2009, vol. 3, p. 5.
[RPSG2018] D. Ratasich, T. Preindl, K. Selyunin, and R. Grosu. Self-healing by property-guided structural adaptation. In 2018 IEEE Industrial Cyber-Physical Systems (ICPS), pages 199-205, May 2018.