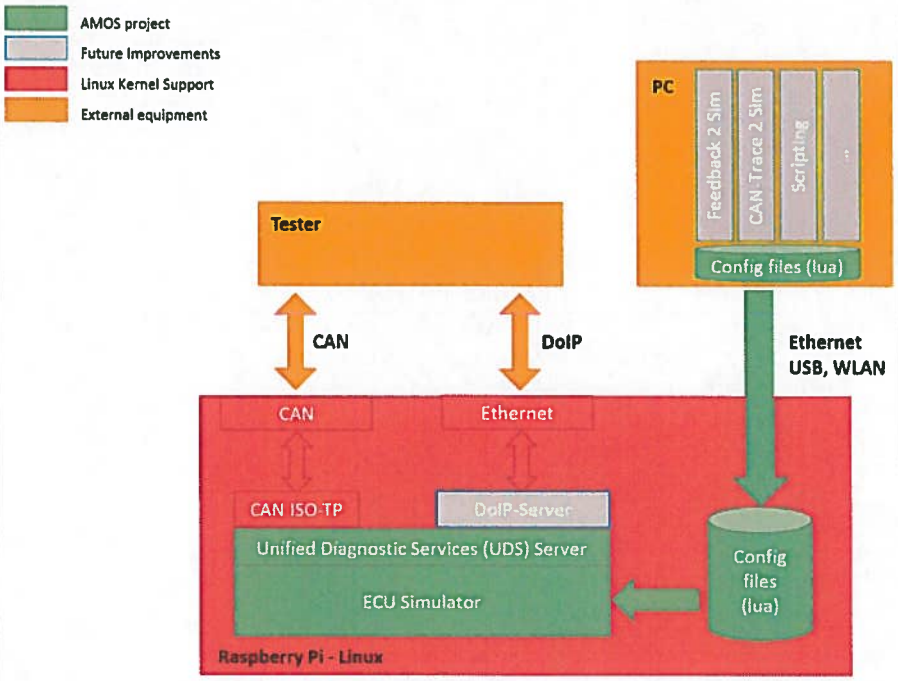
Anhang A

Project Information

Project Title:	Car-Simulation based on Raspberry Pi
Project Type:	AMOS
Industry Partner:	AVL DiTest GmbH
Contact Person:	Tobias Jähnel

Project Description

Project Summary:	One of the challenges in the development of Diagnostic Vehicle Testers is the availability of vehicles to verify the correct behavior of the software. Especially but not only for pre-series and development cars it is very hard to get access to the right vehicles. While low-level components can be tested using Unit-Tests, it is hard to perform integration or system tests. These are subject to availability of a car and time and money-consuming. And even if that is given, verification is limited to one or two vehicles the tester has access to. **Car Simulation** Cars consist of many different ECUs (Electronic Control Unit) that are networked using bus systems like CAN, FlexRay or even Ethernet. The goal of this project is to implement a scriptable ECU (Electronic Control Unit) simulation environment that can *simulate diagnostic communication*. Diagnostic Developers can thus model the diagnostic behavior of single ECUs or even whole cars inside a small box such as a Raspberry Pi. By using the CAN shield it is also possible to connect real ECUs while others are simulated. **Unified Diagnostic Services (UDS)** One of the modern communication protocols used in vehicle diagnostics is UDS (Unified Diagnostic Services). It is a simple Request-Response protocol where the Tester PC acts as the client and the ECU as the server. The ECU provides Services which can be called by the client, such as reading the error codes from the fault memory. To call a service, the client (Tester PC) sends a *request* to the server (ECU). After processing the request, the ECU sends back a *response* containing the requested information. The protocol UDS including all possible services and the request and response structure is specified in ISO 14229-1.
Project Details	The aim of this project is to implement a software component that simulates the UDS communication of one or many ECUs (Servers). This simulation shall be configurable and scriptable by Lua Scripts. The following picture shows an overview of the desired architecture, where

	the green boxes are the components that are new and need to be created. The central part of the implementation will be the scriptable ECU Simulator. This component should run on a Raspberry Pi. The configuration/scripting language to be used is Lua, as it is a lightweight interpreter and integrates well with native C/C++ applications. Very closely tied is the UDS protocol implementation. Furthermore we suggest that the students use the virtual CAN bus implementation provided by Linux for their own tests. There will also be the opportunity for integration meetings with a real tester application.  Keep in mind that communication to several ECUs at once is possible and needs to be scheduled accordingly
Project Tasks	• Define the configuration interfaces together with AVL • Set up virtual CAN busses and nodes via SocketCAN • Set up tests based on virtual CAN busses • Implement UDS Server for dispatching the tester requests to the simulated target ECU instance • Define viable defaults for the responses

Project Constraints

Core Technologies:	• We only consider the bus system CAN at the moment • Target platform is a Raspberry Pi 2/3 with a CAN-Shield (e.g. PiCAN2 by SK Pang Electronics) • All platform specific features shall be abstracted We might want to port this software to another Platform (even Windows) later

	• Programming language: C/C++ • Configuration shall be done using Lua • CAN/ISO-TP communication is provided by the SocketCAN Linux Kernel Module https://en.wikipedia.org/wiki/SocketCAN • UDS is specified in ISO 14229-1 Short info: https://en.wikipedia.org/wiki/Unified_Diagnostic_Services
Team Language:	Deutsch oder Englisch
Preconditions:	-
Resources:	-