Endterm Presentation - Safety Car

HW/SW-Co-design with (LEGO)Cars

Florian, Chris and Lukas

Technische Universität München

3rd February 2014

Content

Hardware Overview

Key-Components:

- *wooden chassis*, aprox. 40 cm x 35 cm
- *12.6 V battery* with continuous 90 A ($\Rightarrow$ 1134 W!)
  Central power-management
  (generating 5 V for FPGAs / Linux-PC and 9 V for Ethernet-Switch)
- *4 CMWUnits* (ControlMotorWheel-Unit) see next slide...
- *Linux-PC* which controls every CMWUnit and sensor
- *Ultrasound-Sensors*

Car Design - Overview

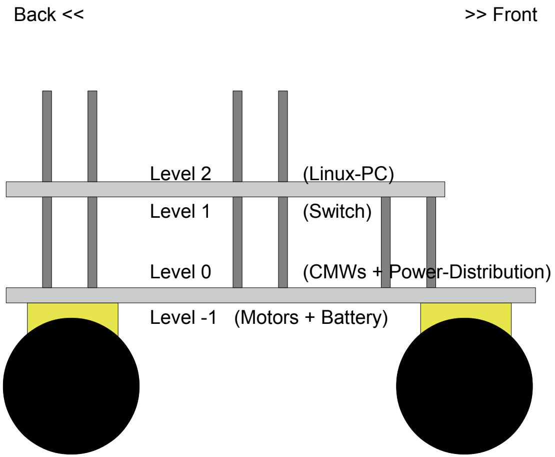


Figure : The car is divided into four levels. Each level has a different

Level -1

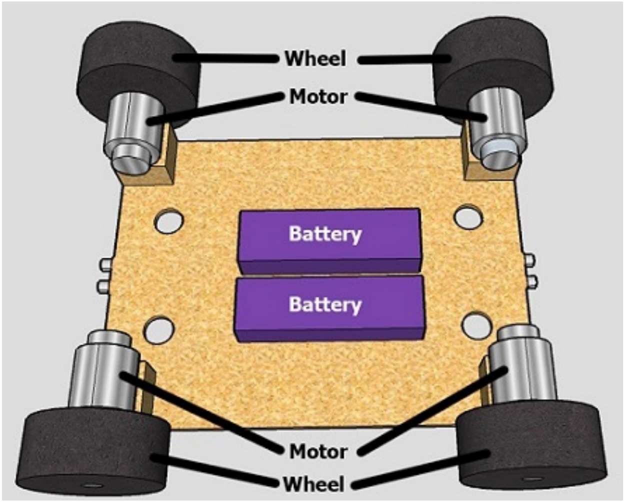


Figure : The lowest level contains the four motors and the battery.

Level 0

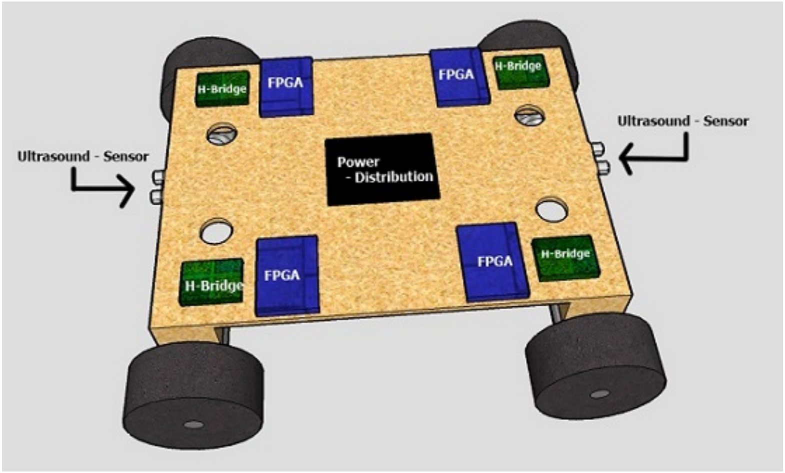


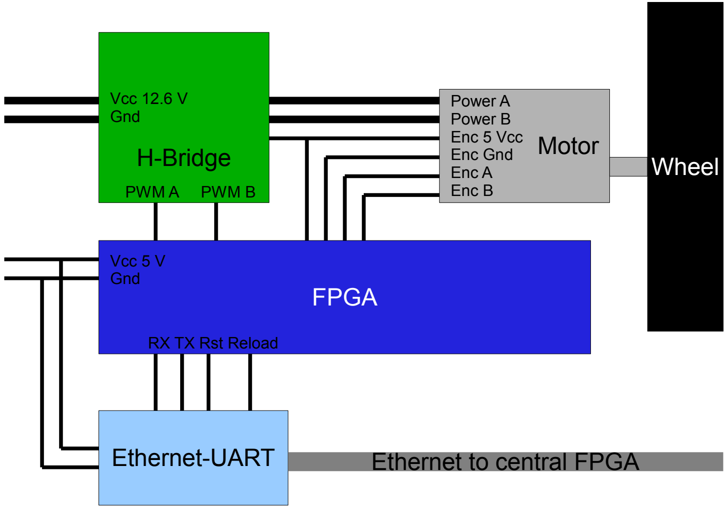
Figure : Level 0 contains the CMW-Units and the Voltage-Distribution

CMWUnit - Control-Motor-Wheel-Unit

Each Control-Motor-Wheel(CMW)-Unit consists of:

- One *Ethernet-UART* connected to the central FPGA
- One *DE0Nano-Boards* (FPGA)
- One *H-Bridge* (dual-channel but we only use one channel)
- One *Pololu Motor* (max. power: 60 W @ 12 V, 5 A).
  Problem: Many components can not take over 2 A!
- One *Soft-Wheel* (diameter: aprox. 12 cm)
  Problem: Each Soft-Wheel can take max. 3 kg

CMWUnit - Control-Motor-Wheel-Unit



Level 1

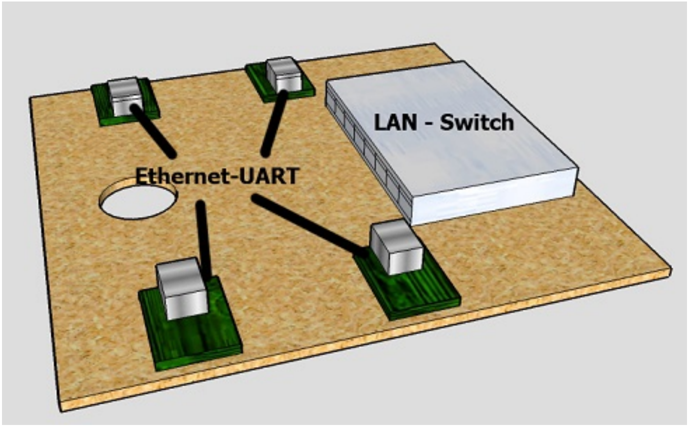


Figure : Level 1 contains the Ethernet-Switch

Level 2

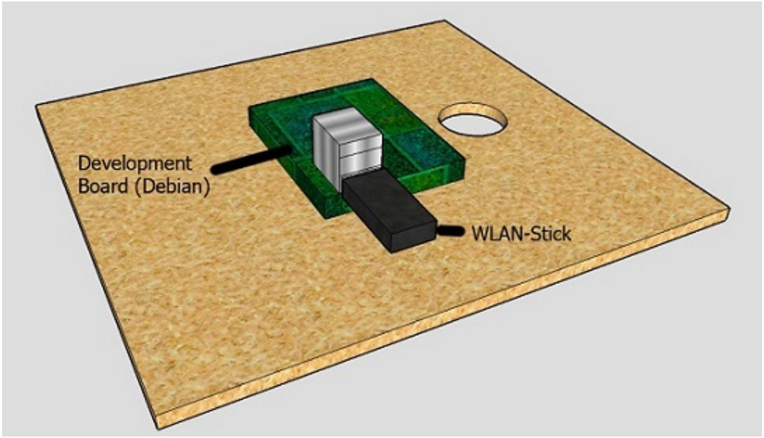


Figure : Level 2 contains the (big) Sabre-light i.mx6 (Linux-PC)

The (great) assembling

- First issues: Wood or aluminium? Which size?

  ⇒ Not enough proper aluminium, so take wood!
  ⇒ We resized the plank three times...
- Building the first version of power-management.
- Then see following process...

The (great) assembling

```
while(true){
        WaitForComponents();
        GetNewComponents();
                            // Yay!
        RealiseThatNewComponentsSuitNotForWorkflow();
                            // :-(
        ChangeEverythingToImplementNewComponents();
        SolderANewPowerDistribution();
                            // Burn fingers ;)
}
```

Our Aims

We want to reach these architectural aims:

1. hierarchical and distributed system
   (e.g. separated Motor-Control)
2. self-maintaining car
   (PID calibration, no hardcoded constants, ...)
3. simple programming of the master-controller (Linux-PC)

CMW-Unit SW design

Processing Unit: *Nios II* embedded core.

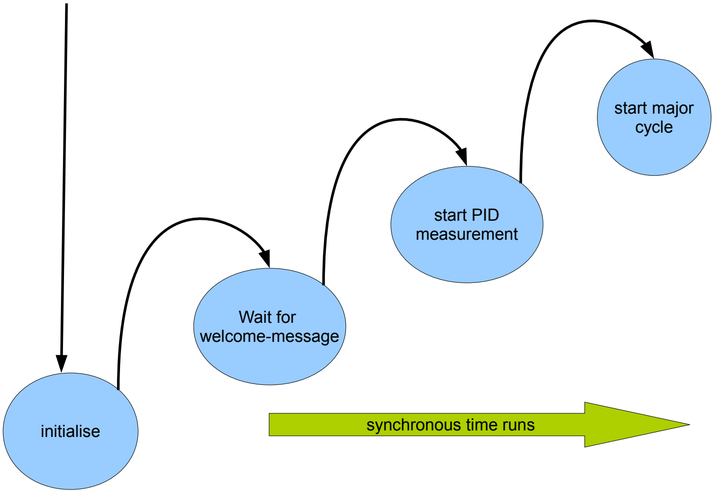
Main tasks:

1. Controlling the motor-speed (PI-Controller)
2. Communicate with Central-Linux-PC
3. Polling the sensors

Doing this tasks in a **hard timed cycle** (as for a real-time system).

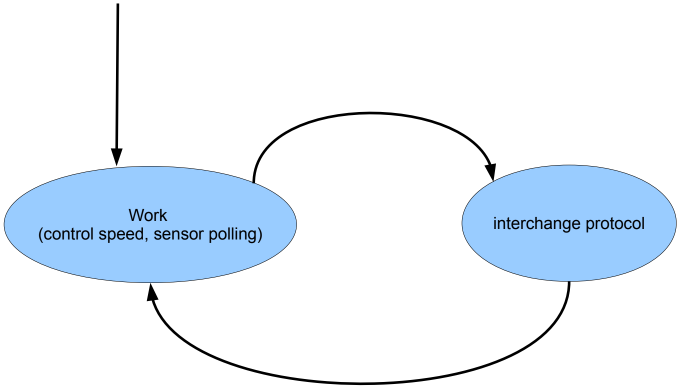
CMW-Unit task cycle

Start sequence:



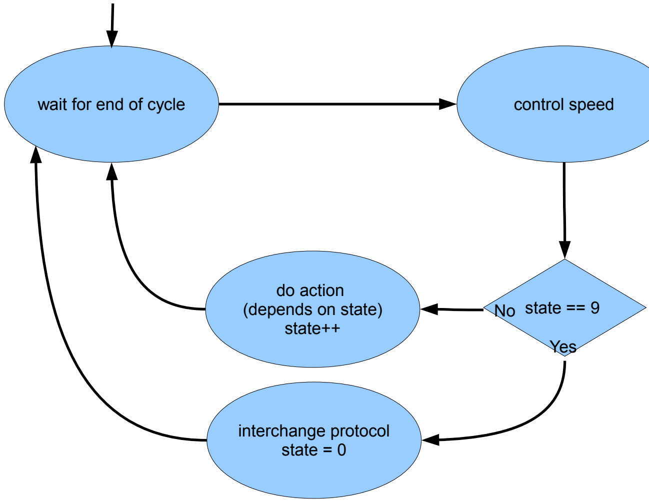
CMW-Unit task cycle

Major cycle:

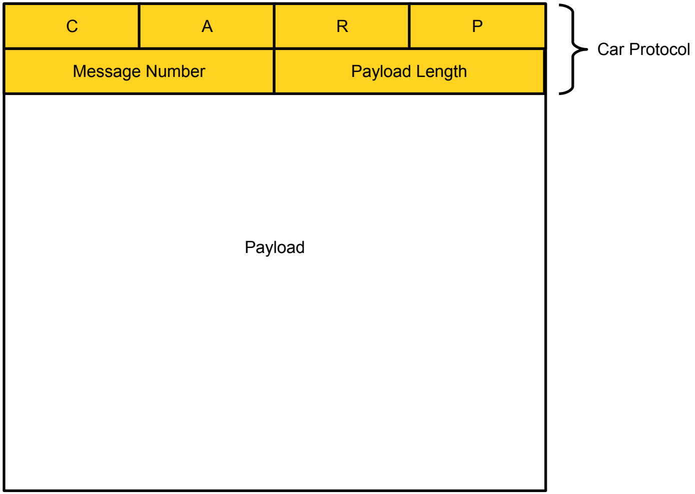


CMW-Unit task cycle

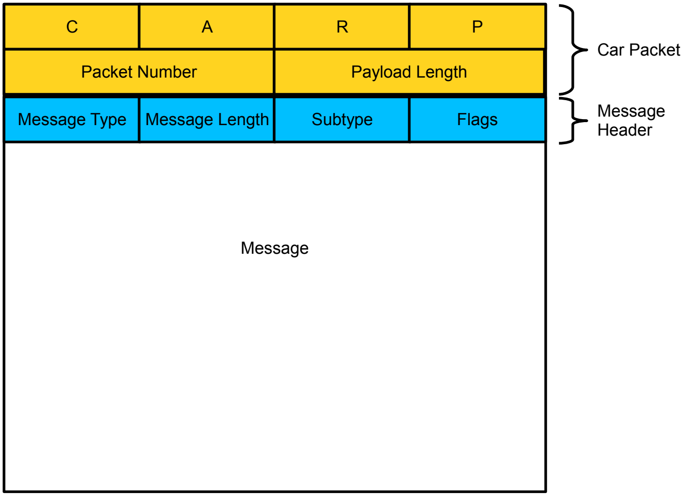
Minor cycle:



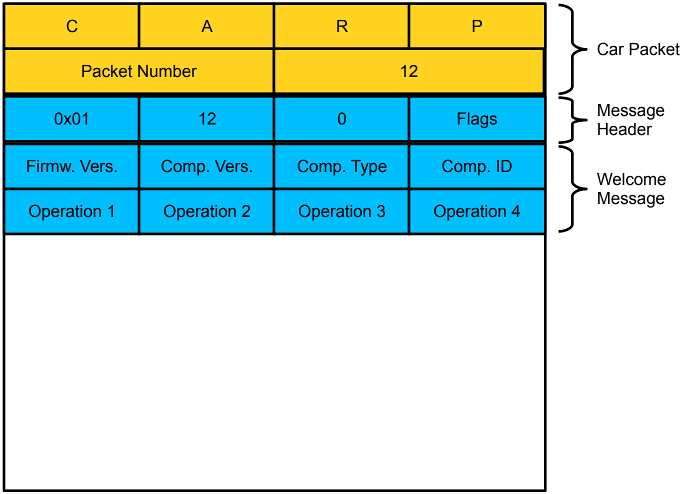
Networking Protocol



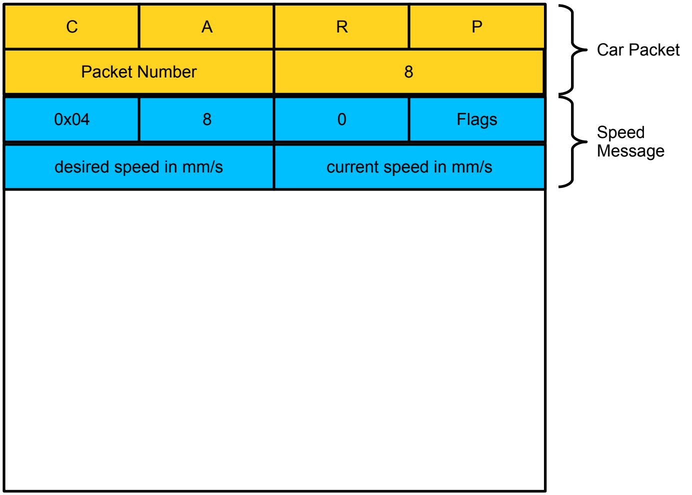
Networking Protocol



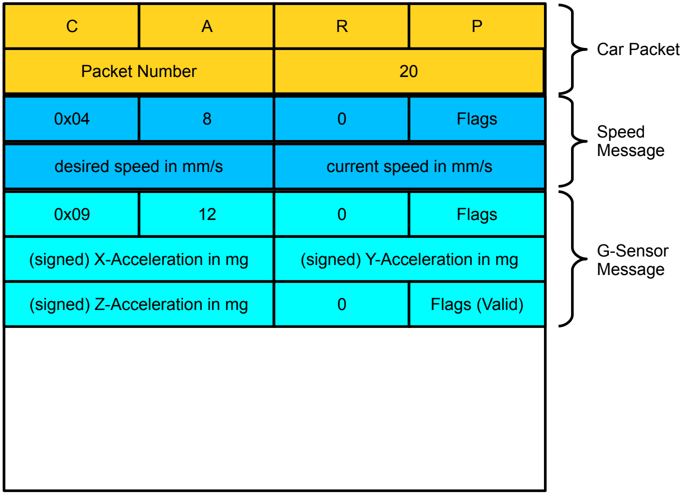
Networking Protocol



Networking Protocol



Networking Protocol



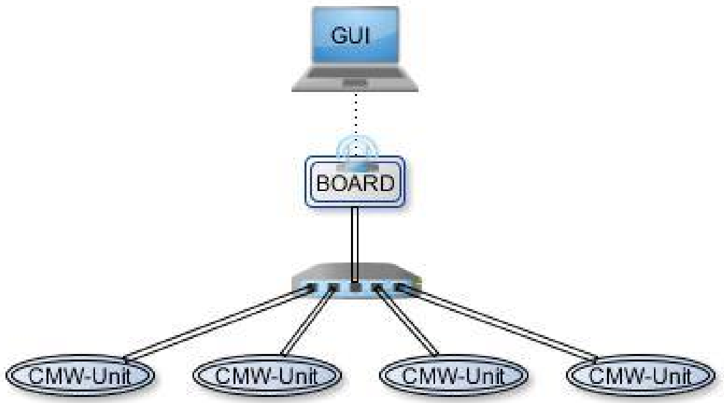
C	A	R	P	Car Packet
Packet Number		20		
0x04	8	0	Flags	Speed Message
desired speed in mm/s		current speed in mm/s		
0x09	12	0	Flags	G-Sensor Message
(signed) X-Acceleration in mg		(signed) Y-Acceleration in mg		
(signed) Z-Acceleration in mg		0	Flags (Valid)	

Network Architecture

- single star-architecture
- components communicate only with the central Linux-PC, not each other
- components only understand the protocols they need

  exception: CarPacket (main protocol) and messages with type lower 8
- similar components can use different protocols
  (e.g. CMW-Units from different producers)

Network Architecture



Central ECU: Linux-PC

Processing Unit: *Sabre-light i.mx6* development board (4 cores).

Main tasks:

1. Control the speed-controller
2. (Polling the sensors)
3. Calculate next behavior

Central ECU: Linux-PC

SW Architecture:

- Multi-Process / Multi-Thread
- seperated Behavior-Planning and Network-Communication

Current Setup:

- Main application:
  - 1. Thread: network-communication
  - 2. Thread: web-communication
  - Main Thread: behavior-planning
- Web-Server for control-ui
- dhcp-server, os, much more ;)

Central ECU: Linux-PC

Current Behavior:

- Human user controls movements via GUI
- GUI sends data to a hidden server (not NSA)
- If car is in danger (see wall ;)) then the car will speed down.
- If the distance between car and obstacle is lower 20 cm then the car will stop.

Central ECU: Linux-PC

Possible Behavior:

- Car discovers the world around it
- ABS, ESP via G-Sensors (data is already available)
- Web-Cam for little NSA-agents ;)
- ...

Thank You!

Thank you for your attention!

Any (even silly) questions???