SCIENCE
PASSION
TECHNOLOGY

fROS: A Generic Fieldbus Framework for ROS

Daniel Schneider

- Daniel Schneider

- M.Sc in Advanced Driver Assistant Systems
- PhD ongoing at Graz University of Technology

- Research Assistant at Kempten University of Applied Sciences
- Application Engineer at AVL Deutschland GmbH

- Scenario specific performance evaluation for ADAS
- Data-mining, Big Data processing and scenario extraction

Agenda

1. **Introduction**
   1. Use case
   2. Middleware
   3. Fieldbus

2. **fROS**
   1. General Overview
   2. Node Architecture
   3. Implementation

3. **Temporal Validation**
   1. Validation principle
   2. Validation results

Agenda

1. Introduction

1. Use case
2. Middleware
3. Fieldbus

2. fROS

1. General Overview
2. Node Architecture
3. Implementation

3. Temporal Validation

1. Validation principle
2. Validation results

1.1 Use case

- Data-driven development and validation for ADAS and AD requires real-world data with high-precision sensors – so-called reference sensors

- Especially in sensor development and testing, reference sensors are strictly required for evaluation of sensor-performance, software-comparison, and application

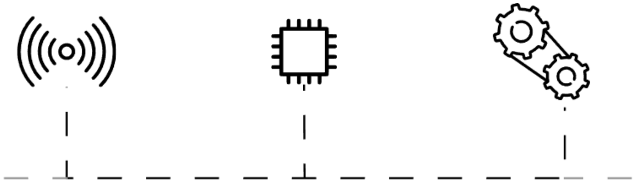
- Combining both, reference sensors and vehicle-internal communication is often challenging and cost intense

- We are presenting an open-source framework for reading vehicle-internal communication alongside with additional sensor signals such as reference LiDAR, IMU, camera, …

1.2 Middleware

- To connect multiple sensors and actuators, a central software framework (so-called middleware) is required
- Beside sensor-system-communication, timing, visualization, calibration but also message-recording and message-processing is a central part of the middleware
- In the field of ADAS, automotive data and time-triggered framework (ADTF) is often used
- Due to the license model, but also due to the excellent support open-source such as the robot operating system (ROS) are often used in research and development
- To cover the specific requirements of the automotive industry, real-time and failsafe communication, but also end-to-end encryption is available (ROS 2)

1.3 Fieldbus Protocols



- For communication between sensors, ECUs, and actuators, various Fieldbus protocols are used
- Controller Area Network (CAN) as the most common protocol was standardized in 1993
- Extensions such as CAN-FD, but also different protocols such as LIN, (Automotive)Ethernet, EtherCAT, MOST, …. available and mostly tailored for its application
- Deterministic behavior only for few protocols such as FlexRay (FR)
- For all protocols, principles for ensuring secure transmission, encryption, etc. are applied
- Few Fieldbus nodes in ROS available

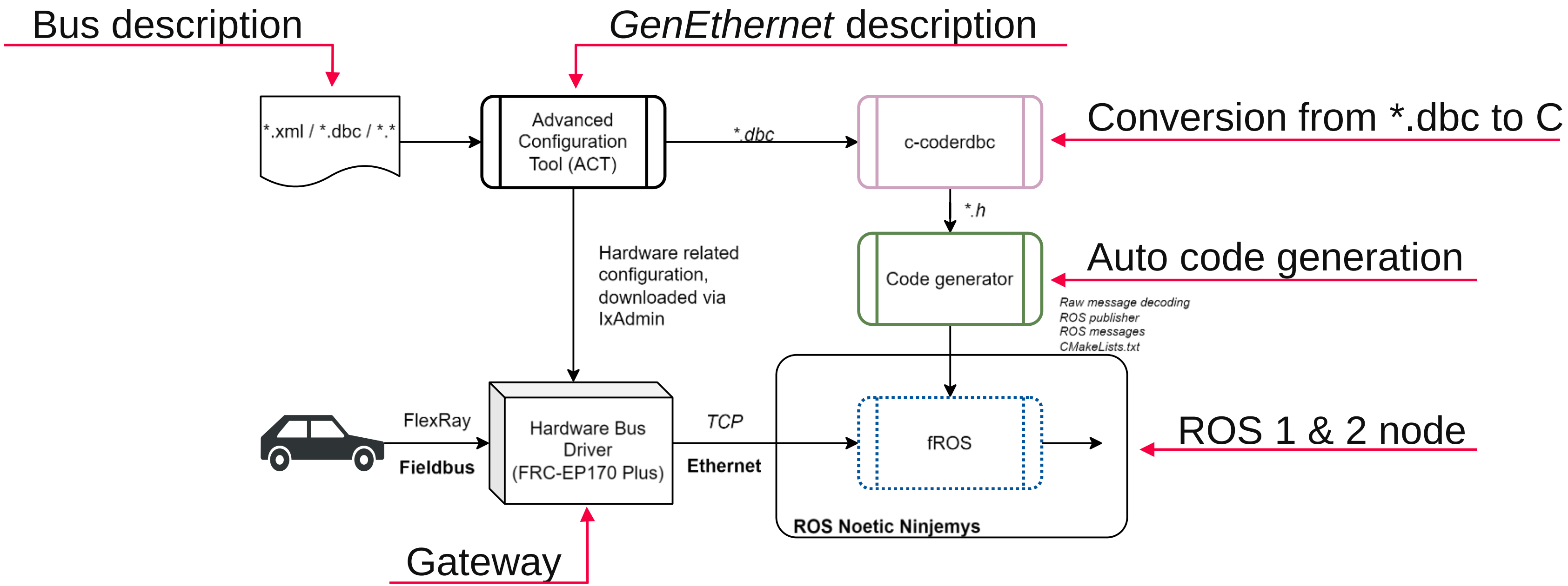
Agenda

1. **Introduction**
   1. Use case
   2. Middleware
   3. Fieldbus

2. **fROS**
   1. General Overview
   2. Node Architecture
   3. Implementation

3. **Temporal Validation**
   1. Validation principle
   2. Validation results

2.1 General Overview



github.com/AdriveLivingLab/fROS

2.1 General Overview

- IXXAT defines *GenEthernet* (GE) as generic bridge between Fieldbus and TCP-message

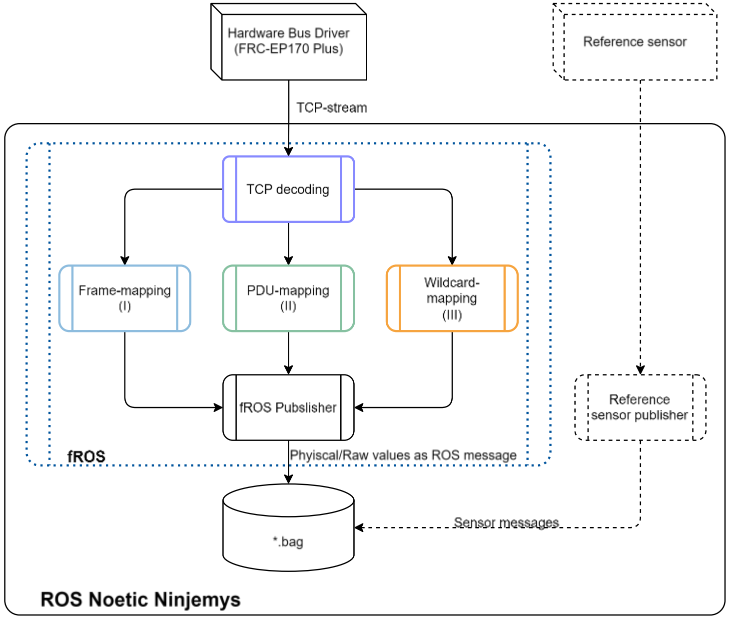
- The idea behind GE is that all messages are mapped into defined binary structure:

```
typedef struct {
  uint16_t w_msgSign; // B space
  uint16_t w_msglen; // message complete length
  uint32_t dw_id; // message ID
  uint8_t b_format; // message frame format info
  uint64_t qw_timestamp; // message timestamp
  uint16_t w_reserved; // reserved
  uint16_t w_datalen; // num of data bytes
  uint8_t ab_data[]; // data bytes
  } IXXAT_BIN_PKT;
```

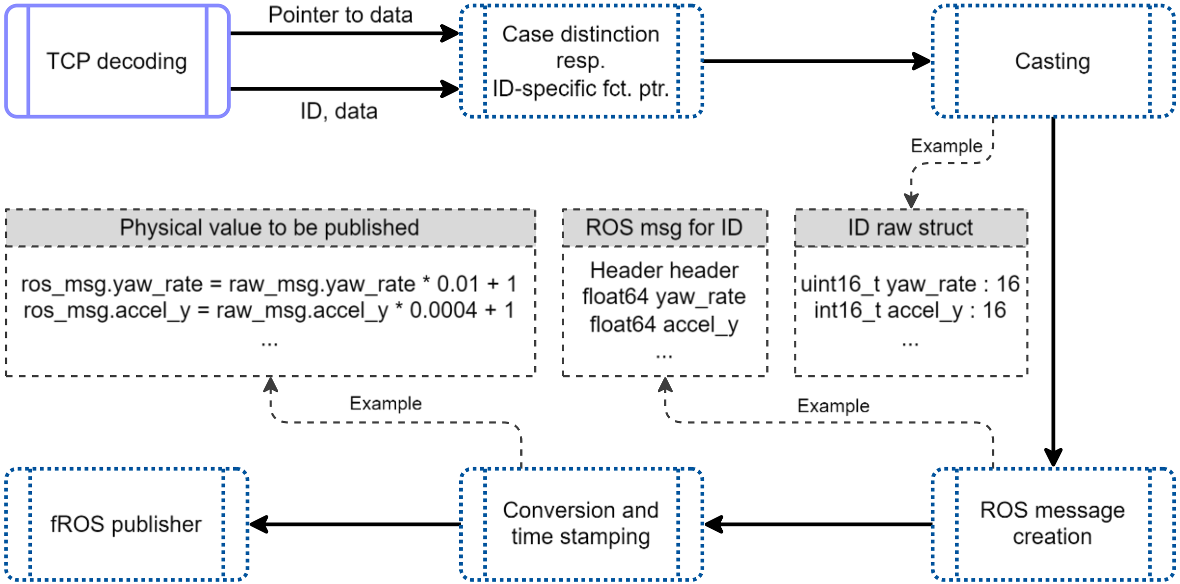
- With that, LIN, CAN, FlexRay, and EtherCAT can be converted into *GenEthernet*

2.2 Node Architecture

- Overall architecture can be adapted according to the use case
- Entire bus description file is not required (compliance) for data recording
- Gateway generates TCP-stream via *GenEthernet*
- TCP-stream is decoded and mapped to ROS message according to the selected principle:
  - Frame-mapping
  - PDU-mapping
  - Wildcard-mapping
- The resulting message is published and saved



2.3 Implementation



Agenda

1. **Introduction**
   1. Use case
   2. Middleware
   3. Fieldbus
2. **fROS**
   1. General Overview
   2. Node Architecture
   3. Implementation
3. **Temporal Validation**
   1. Validation principle
   2. Validation results

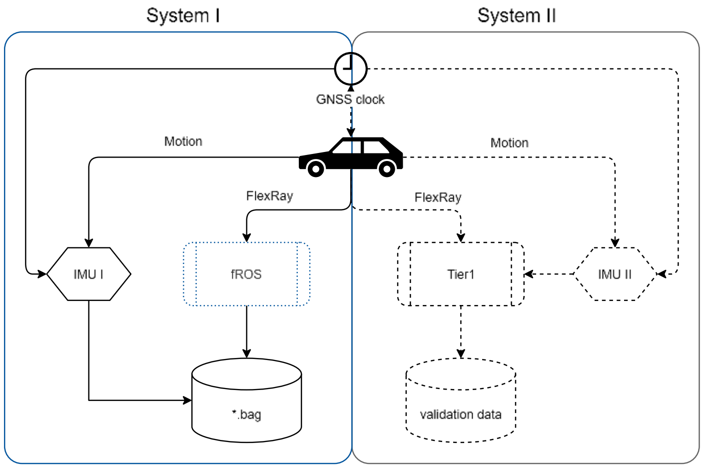
3.1 Validation Principle

- For temporal validation of fROS, a vehicle dynamics-based validation process was applied

- Two independent systems are installed in the same vehicle, measuring the same motion, FlexRay, and time information

- Weave-test according to ISO 13674-1 was conducted at 60 $\frac{km}{h}$ with $\pm 22°$ steering wheel angle to achieve $\pm 2\frac{m}{s^2}$ lateral acceleration

- Cross-correlation identifies temporal difference between the two sinusoidal signals of the yaw rate

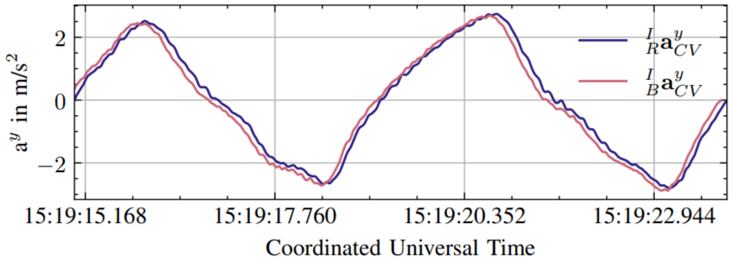


Hardware	System *I*	System *II*
FlexRay Interface	Ixxat FRC-EP170 Plus	Vector VN7640
Inertial Sensor	GeneSys ADMA-G-PRO+	GeneSys ADMA-G-PRO+
GNSS Antenna	NovAtel VEXXIS GNSS-502	NovAtel GPS-702-GG
RTK-GNSS	✓	✓
CPU	AMD EPYC 7401p	Intel Core i7 7820HQ
RAM	256 GB	16 GB
Software		
Operating System	Linux Ubuntu 20.04	Windows 10 Professional
Framework	ROS 1 Noetic Ninjemys	Vector CANoe 15 SP2

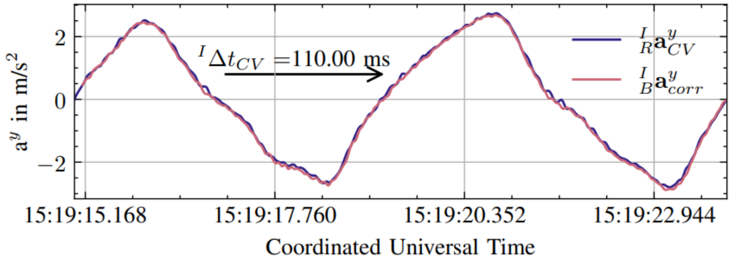
3.1 Validation Principle Verification

- All data are mapped to the common GNSS-clock via INS
- The intersection of the abscissa for the yaw rate in the first rising half-wave are defined as the event start point (esp) of the validation
- The event end point (eep) was identified exactly two periods later, formally:

$$
{}^{I}_{B}\dot{\Psi}_{TV} = \left\{ {}^{I}_{B}\dot{\Psi}_{i} | i \in \mathbb{N},\ {}^{I}_{B}\mathrm{esp}_{TV} \leq i \leq^{I}_{B} \mathrm{eep}_{TV} \right\}
$$
$$
{}^{I}_{R}\dot{\Psi}_{TV} = \left\{ {}^{I}_{R}\dot{\Psi}_{i} | i \in \mathbb{N},\ {}^{I}_{R}\mathrm{esp}_{TV} \leq i \leq^{I}_{R} \mathrm{eep}_{TV} \right\}
$$



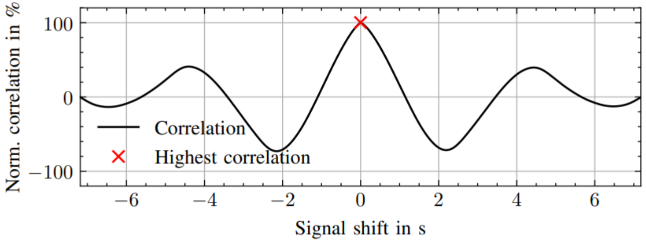
(a) Filtered lateral acceleration signals obtained from the on-board sensor (red) and IMU (blue) on system *I*.



(b) Time-corrected acceleration signal (red) and IMU signal (blue) on common UTC axis.

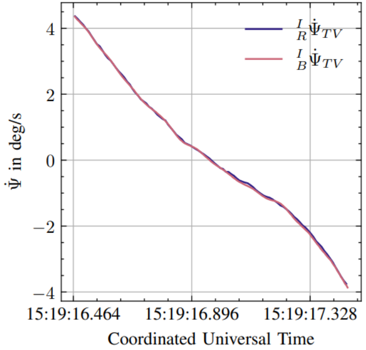
3.1 Validation Results

- Cross-correlation between the yaw rate of system *I* and system *II* determines $0\ ms$ time shift between it

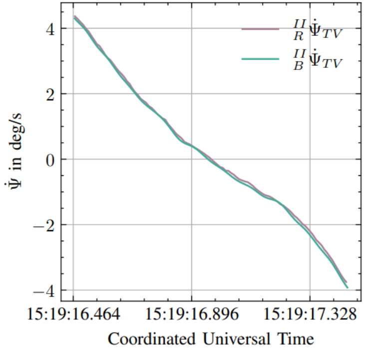


- However, system *II* shows more precise timing for the 200 Hz sampled data

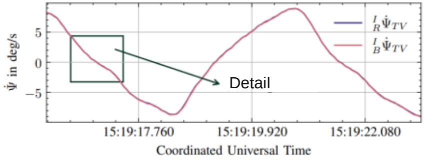
System	Mean cycle time in s	Cycle time standard deviation in s
I	5.00×10^{-3}	3.53×10^{-4}
II	5.00×10^{-3}	3.00×10^{-6}

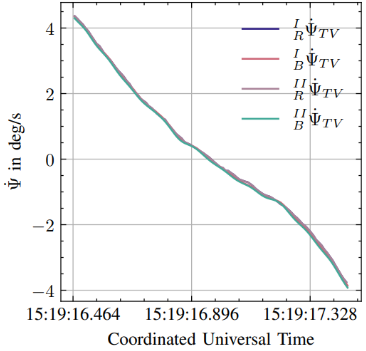


(a) Yaw rates measured on system *I*.



(b) Yaw rates measured on system *II*.





(c) Yaw rates of both systems combined.

- This work was funded by the Bavarian Ministry of Economics Affairs, Regional Development and Energy [grant number DIK0229/02] and supported by AVL/DE

- Daniel Schneider and Ludwig Kastner worked on the presented topic together. Ludwig mainly developed the fROS node, whereby Daniel developed the methodology around it

github.com/AdriveLivingLab/fROS

github.com/schneider-daniel

github.com/ludwig-kastner