A Generic Data Processing Framework for Heterogeneous Sensor-Actor-Networks

Matthias Vodel, René Bergelt, and Wolfram Hardt
Dept. of Computer Science
Chemnitz University of Technology
Chemnitz, GERMANY
Email: { vodel | berre | hardt }@cs.tu-chemnitz.de

***Abstract*—This paper presents a generic, energy-efficient concept for synchronised logging, processing and visualisation of arbitrary sensor data. The proposed framework enables a chronological coordination and correlation of information retrieved from different, distributed sensor networks as well as from any other self-sufficient measurement system. By relating data from different sensor data sources the overall information quality can be improved significantly. The implementation allows for easy cross-platform communication and facilitates readability by humans by employing the XML data format for data storage. Furthermore, the aggregated, heterogeneous sensor information can be converted into an extensible list of output formats. Depending on the application-specific requirements for visualization it is possible to consider additional meta-information from the test environment to optimise the data representation. The utilization of advanced data fusion techniques and pre-processing mechanisms enables a selective data filtering to reduce the network load. In order to evaluate basic usability requirements and the effectiveness of the proposed concept, an automotive sensor network is presented as a test system for the framework. For this demonstration, the available on-board measurement systems of a vehicle were extended by high-precision sensor nodes establishing a wireless sensor network and aggregating environmental and behavioural data on several test drives. Subsequently the correlated measurement information was converted and visualised for use with several professional data analysis tools, including jBEAM, FlexPro as well as Google Earth.**

***Keywords*-data aggregation; data fusion; data synchronisation; heterogeneous wireless sensor networks; sensor actuator systems**

I. Introduction

Current research projects in the field of wireless sensor networks operate on different, proprietary hardware platforms and contain multifaceted types of sensors. The main objective of activities in this area is to establish an extensive knowledge base which concentrates data of different kinds and allows its user to gain further information out of the entirety of the collected data. In order to accomplish this goal, the fusion of data from different sources as well as methods for a uniform analysis are essential. This scenario is illustrated in *Figure* 1. Currently, most measurement scenarios consist of several application-specific and independently operating processes for the collection, storage and analysis of sensor data. There are no uniform synchronisation techniques between autonomous sensor systems in existence. Accordingly, a detailed and target-oriented post-processing of the data sets within a shared knowledge base is not feasible. In consequence, it requires much effort to create unambiguous, novel relations between different measurement information or this is simply not possible at all. Due to missing relations, it is very hard to create a common primary index for the given, heterogeneous sensor platforms.

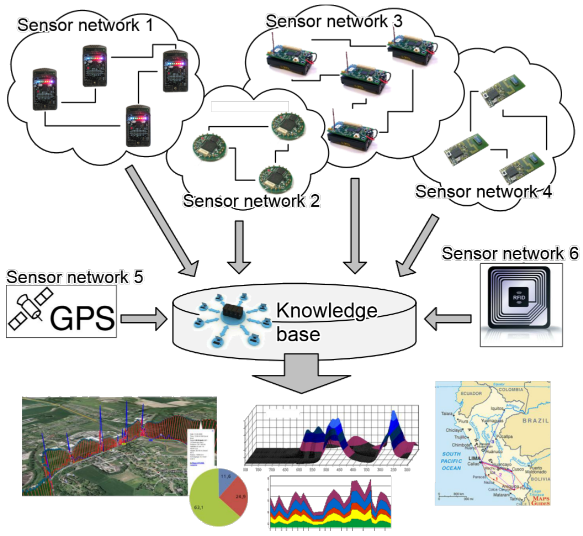


Figure 1: Gathering sensor data into a single knowledge base where it is analysed uniformly and independent of its origin

To solve this problem, we developed *GREASE* - a Generic Reconfigurable Framework for the Evaluation and Aggregation of heterogeneous Sensor Data [1], [2]. In order to introduce this integrated data processing concept, this paper is structured as follows: After this introduction, section *II* provides an overview about heterogeneous, distributed sensor environments, the data processing flow and respective challenges. The proposed GREASE framework is introduced in Section *III*, including conceptual fundamentals, basic requirements, system parameters and the top level structure (Section *IV*). Section *V* provides implementation details of the GREASE software architecture as well as the overall application flow within the framework. Section *VI* specifies a sample application scenarios with all integrated components and environmental conditions. The corresponding data analysis is described and discussed in Section *VII*. Finally, the paper concludes with a summary and an outlook for future work in this research project.

II. Related Work

During the last two decades, a couple of commercial tools for measurement, data recording and monitoring were developed. Unfortunately, most of these tools have functional or conceptual restrictions. Some sensor hardware vendors offer exclusive, hardware-specific analysis tools, which require additional devices or do only cover specific product series. Other sensor systems do not offer any special software tools for extracting the measured data sets. Thus, there is no particular support for further post-processing steps.

In consequence, software solutions targeting this problem have been developed. These include *LabView* from National Instruments [3], *jBEAM* from AMS [4], or FlexPro [5], which offer multiple features to enhance the restricted vendor tools. These more general toolkits provide a larger set of generic data recording and handling functions. Furthermoere, the applications allow an interpretation of offline data from databases or files as well as the live analysis from a given data source. Both jBEAM and LabView operate platform-independently. The software applications support many established data formats and related communication interfaces. Especially jBEAM, which integrates the *ASAM* standard *(Association for Standardisation of Automation and Measuring Systems)* [6], enables an easy and modular extension with user-defined components whereas FlexPro already includes a lot of additional visual plugins and offers a complete visualisation framework for the given measurement data.

Nevertheless, the very high system requirements of all given software applications represent a critical disadvantage. Accordingly, these tools are not suitable for the usage in resource-limited data recording environments. Consequently, such frameworks have to be used in a second data processing step on dedicated workstations with sufficient hardware components and resources. Hence, small and energy saving hardware systems, which are used exclusively for collecting and storing multiple data from different sensor sources, are not able to use data aggregation and visualisation features of these software frameworks [7]. Due to these circumstances, most ongoing sensor system projects use proprietary software solutions to organise and synchronise the collected measurement data [8]. Additionally, in practice there are many critical compatibility problems between such software tools. In consequence, modifications of the measurement scenario or the system configuration take a lot of time and bind important resources. The user therefore demands a universal software tool for collecting and analysing the entire pool of sensor information in an application-specific and resource-efficient way. Automated or semi-automated data evaluation and visualisation techniques represent further essential requirements, especially for unattended long-term aggregations of environmental sensor data or test runs.

III. Concept

For the concept we had to find generic methodologies and standards to route information from different sensor systems into a common data processing unit in a synchronised way, considering scenario-specific configuration schemes and sensor parameters. In this respect, synchronised time stamps for the heterogeneous sensor data sets are very important in order to allow the correct identification of correlations in the common knowledge base later on. Furthermore, such techniques allow the definition of user-specific data analysis procedures during the measurement runtime. This also includes advanced data fusion techniques [9], [10], [11] to shrink the data volume directly within the sensor nodes or prior to storage.

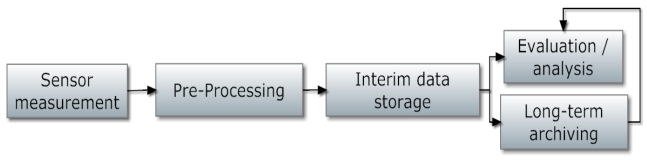


Figure 2: Data flow for typical measurement scenarios

The general data flow for (long-term) measurement scenarios is illustrated in *Figure* 2. Most available solutions only cover one half of this flow either the data aggregation or the evaluation of already collected and stored data. With GREASE we wanted to develop a framework which governs the whole data flow in such a measurement system in a generic and extensible way. To provide such features, GREASE represents a software framework based on a capable and lightweight data management concept, which is able to bypass the aforementioned mentioned disadvantages. It combines advanced sensor network configuration features with resource-efficient operating parameters. GREASE integrates the entire data processing flow, including all stages like data measurement, processing, storage and finally data analysis tasks. This flow is illustrated in *Figure* 3.

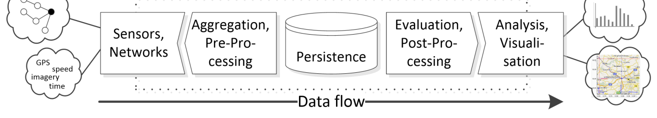


Figure 3: Integrated data processing flow for heterogeneous measurement topologies

Our concept focuses on resource-limited systems and has to be feasible for a wide variety of application scenarios. Thus, the primary objective is a dynamic and flexible processing environment, which is adaptable to modifications in configuration or analysis requirements. Furthermore, the data processing core has to be separable into two spatial, chronological and platform-specific operating modes. All components for the data measurement are working within the first mode. All relevant modules for the data analysis as well as possible visualisation components operate independently within the second mode. Based on these dedicated modes, we are able to map different data processing functions to predefined configuration scenarios. In contrast, other related software tools do not separate the data handling process into such phases, especially with respect to the efficient concept of operations. Basically, GREASE acts as mediator between existing individual hardware sensors as well as sensor networks and evaluation processes including for instance visualisation tools. In this position it offers a standardised way to represent and transport measured data. The framework provides configurable synchronisation parameters for collecting information from several distributed sensor components. Accordingly, changes in the data analysis process do not affect the components of the data measurement and (in most cases) vice versa. This feature offers significant benefits, especially for complex sensor systems or inaccessible measurement environments.

In addition, all GUI *(Graphical User Interface)* actions, which can be accessed by the user, also have to be executable and controllable in an automated or semi-automated way. This feature represents another important difference to other related software tools, which do not provide any script-based operating modes without GUI. But especially for continuous maintenance-free and unattended sensor measurement scenarios, the scripting of user-defined activities is essential.

Hence, all central requirements for a synchronised data logging, processing and visualisation framework, particularly in the field of heterogeneous sensor network systems, can be summarised as follows:

- Synchronisation of different, autonomous sensor systems
- Modular extensibility through plugins and easy modification / adaptation
- Usage of a common data exchange format (e.g., *XML ((Extensible Markup Language)*) instead of proprietary data types
- Off-line and live data analysis from files, network file systems or databases
- Graphical User Interface for configuration and maintenance
- Automated or semi-automated data analysis and data representation mechanisms

Based on the proposed concept, the developed framework acts as coordinator between a given set of application-specific sensor and actuator components.

IV. Structure

As already mentioned, the structure of the proposed concept is basically divided into two operating phases. The first one encapsulates the data recording, synchronisation and correlation whereas the second phase provides processes for the data analysis generating a user-defined representation of the information sets. These two phases are connected through the so called persistence layer which is in charge of permanently storing the measured data in a well-defined format. This layer also restores data for future analysis and evaluation tasks. The described structure ensures a strict separation of data measurement and analysis tasks and is illustrated in *Figure* 4. Based on this approach GREASE exhibits a modular operating concept, making the whole framework independent of the given application-specific sensor configuration. Therefore, the environment uses an end-to-end communication design, called the *hourglass architecture*, which enables maximum interoperability between

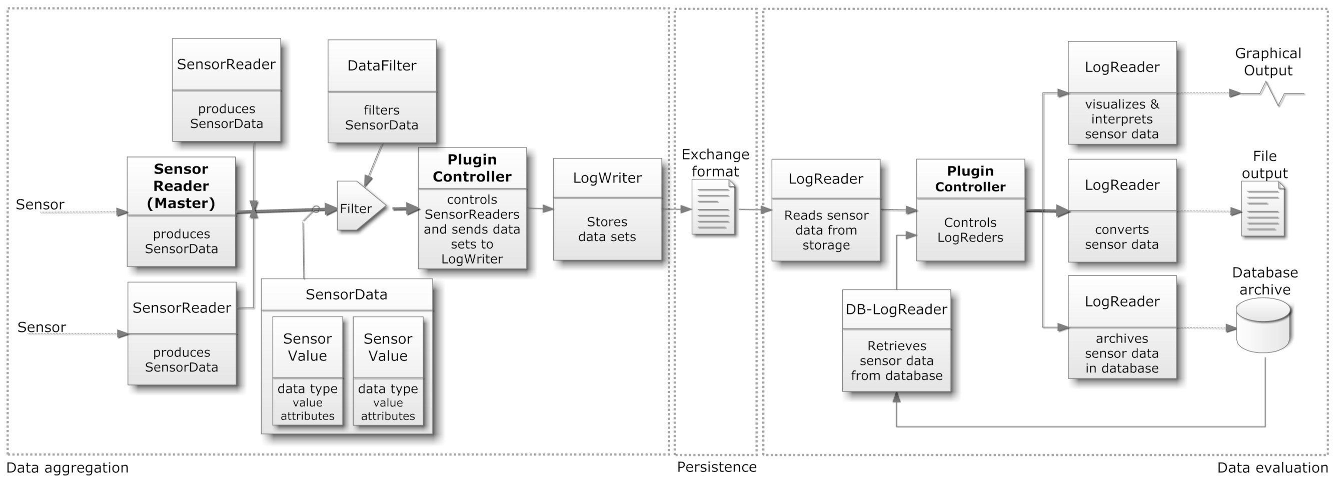


Figure 4: GREASE framework structure which allows the strict separation into two operation phases (measurement and evaluation)

the several components, i.e. modules or plugins. This results in a high diversity with respect to both components for data input (measurement) as well as data output (analysis). In contrast, the uniform connecting middle part is designed simple and light-weight. Correspondingly, the GREASE framework consists of data type definitions, interfaces and a plugin architecture. There exist five main types of plugins, three on the data input side (*SensorReader, DataFilter* and *SensorLogWriter*) and two on the data output side (*SensorLogReader* and *LogReader*). On both sides the coordination is handled by a dedicated *PluginController*. Every plugin type deals with exactly one of the phases illustrated in *Figure* 2 and *Figure* 3, respectively. Consequently, the SensorReader plugins are of particular importance as they transform measured sensor data of sensors into data types which adhere to the framework definition. They are able to combine several data values into data packets (*SensorData* objects) which consist of one or more *SensorValues*. A SensorValue includes at least sensor name, data type and the measured value and an arbitrary amount of descriptive meta-information may be added. These data packets are sent to the *controller* where *DataFilter plugins* enable the pre-processing of received sensor data. If any filter plugin has been registered at the *controller*, the packets will be forwarded to these filters in order to carry out further actions like editing or rejecting data. The *controller* then correlates the received data packets and will group them into datasets according to the scenario configuration. Within this process all internal and external parameters of the environment as well as special meta-information regarding the measurement scheme are correlated together with the datasets. Accordingly, researchers are able to reconstruct the whole test scenario with synchronised data, timestamps and a detailed system configuration. Therefore, the reuse factor, for instance in the field of automotive testing scenarios, increases substantially. In order to save the created datasets a single *SensorLogWriter* module has to be registered at the *controller*. Such a plugin allows the *controller* to write the datasets into a universal exchange format, e.g., an XML file. The gathered sensor data is now at the *persistence level* and may transferred to the evaluation part or archived for later assessment.

For the analysis of the sensor data, a corresponding counterpart of the used *SensorLogWriter* has to exist, a *SensorLogReader* which retransforms the data of the internal storage format into framework-specific data objects. This mechanism is - again - coordinated by a *PluginController*, which hands the retransformed sensor data to user-defined evaluation plugins (LogReader plugins). These plugins allow the user to evaluate, assess, visualise or archive the sensor data. A conversion to other file formats is also possible and can be used to import the measured data into third-party applications. An important advantage of the modular architecture is that the archived data, which has been saved for instance into a database, may be retrieved later on by an appropriate *SensorLogReader* plugin. Accordingly, the data can be processed by a second evaluation plugin. Due to the framework-defined data types, the evaluation modules do not know the origin of the sensor data (e.g., log file, database or any other storage type). Therefore, the development of a single plugin for GREASE can be carried out independently of other framework parts and future measurement scenarios. The persistence level represents the link between aggregation and evaluation and is one basic requirement for the exchange and editing of arbitrary measurement configurations. Existing configurations may now be altered easily by simply adding or removing components (i.e. plugins) without affecting the data flow of the overall monitoring system. Thus, changes within the measurement components are possible without having to update data analysis components and vice versa.

In principle it is also possible to connect data measurement and analysis modules directly, without using the persistence level. This leads to a kind of real-time data processing scenario. It may be noted that GREASE has not been designed for real-time processing and in the current implementation real-time qualities or maximum delays cannot be guaranteed. Therefore, the use of the framework in real-time decision making systems or systems requiring similar qualities is neither advisable nor wanted at the moment, since this is not the main application focus of GREASE. The framework performs best in long-term data measurement projects and in the domain of resource-limited systems without user interaction during measurements.

V. Implementation & Application Flow

To enable a platform-independent operation, the proposed concept has been implemented in *Java*. Nevertheless, additional modules can be written in other programming languages depending on the used controller. A precondition for both high flexibility and easy extensibility is a common interface specification within the controller. This feature of the proposed framework implementation is provided in the form of a *central core library*, which encapsulates the entire data processing logic. The communication between the controller and its set of modules is realized through dedicated protocols and a common syntax, which are both designed as generic as possible to allow a universal and flexible usage. This flexibility also simplifies the integration of third party modules and ensures the compatibility during further developments [12]. Due to the fact, that the sensor configurations and all kinds of scenario parameters are also transmitted within the XML representation, possible enhancements for customer-specific applications can be realized in an easy way.

Regarding the application flow, the initialisation of the *controller* commences with loading an application-specific

Table I: Supported data types by GREASE version 1.3

Type	Description
INT	Simple integer
FLOAT	Decimal
DATE	A date value without time information
TIMESTAMP	A timestamp (date & time information)
STRING	Arbitrary characters
IMAGE	Image data

configuration file. This file contains all information for the current project as well as the structure of all corresponding *SensorReader* components. Accordingly, the *controller* loads and activates all necessary sensor components and starts the data recording. Each *SensorReader* module operates simultaneously as a dedicated thread. When a *SensorReader* receives sensor data, a *SensorData* object will be generated. The format of this object is predefined by the framework configuration scheme. Such a *SensorData* object consists of a descriptive name (tag), e.g., *gps* for location data, and a list of *SensorValue* objects. A *SensorValue* object describes a single measured value and includes a descriptive name, the data type of the value, the data value itself and attributes which may provide further information about the value, for instance the physical measurement unit or special indicators. The latest GREASE version (GREASE.Core version 1.3) supports the data types listed in *Table* I.

The created *SensorData* object is sent to the *controller*, which forwards it to the registered filter plugins (only if available). Subsequently, it correlates and groups the received *SensorData* objects into datasets. In the following step, the object will be permanently stored through the persistence plugin the user has chosen. Due to the fact, that the *controller* has no knowledge about the interpretation of the transported sensor data, this process is fully application-independent. In contrast, jBEAM for instance uses project files which have to be adapted to changes in the measurement scenario since data measurement and analysis directly build on each other.

The standard storage plugin of GREASE saves datasets in an XML representation. The XML format has been chosen mainly for test and demonstration purposes as it also allows for human-readability. However, the user may replace it by any other third-party or custom plugin which uses different file formats or storage methods altogether. In more resource limited environments and applications this is advisable, since XML exhibits a rather high verbosity and leads to comparatively large file sizes.

Furthermore, the whole framework supports the translation and localisation of content and provides corresponding interfaces to plugins. In the current version, all standard plugins support both the English and German language. The basic setup of GREASE, including all plugins, is listed in *Table* II.

Table II: All plugins which are part of the basic setup of GREASE

SensorReader plugins	
MTS310Reader	Retrieving sensor data from Crossbow MTS310 sensor boards
GpsReader	Retrieving location data through the NMEA 0183 standard
ObdReader	Querying data from the OBD (on board diagnosis) interface of vehicles
CamImageReader	Capturing the image stream of image devices such as webcams
UdpSensorReader	Retrieving sensor data over UDP
DataFilter plugins	
LiveDataViewer	Visualisation of the data flow of the framework
NetVis	Visualisation of sensor network activity
Persistence plugins	
XML-LogWriter	Saves datasets as XML representation
XML-LogReader	Reads datasets from XML representation
DB-LogReader	Reads datasets from a database
LogReader plugins	
LogFuse	Fusing multiple log files
CSVOutput	Converting datasets to CSV format
ImageExtractor	Extraction of image data
DBOutput	Archiving of log files to a database
GEarthOutput	Visualisation of sensor data in Google Earth

With respect to the communication tasks in the sensor environment, we also have to discuss security features. Due to the fact, that GREASE focuses on research and development environments, we actually do not consider further security aspects for the distributed handling and storage of the sensor data. Within the different development stages of a given system, engineers design and implement complex test environments for getting valid and high-quality measurement results. Accordingly, the risks, which result from general communication threats are negligible. Nevertheless, we currently cooperate with related German car manufacturers in regards to this weak point. Several research projects focus on the development of advanced, energy-efficient security features for embedded, resource-limited sensor network topologies. In this context, the main challenge is the maintenance of a lightweight software architecture, which provides stable and flexible modules for diversified application scenarios. Here, advanced security mechanisms have a direct impact on the data throughput and the resource consumption. Accordingly, our goal is to find a good trade-off between runtime performance and security capabilities within GREASE. A first approach could then be the usage of an encrypting persistence module in non-time-critical applications. However, this leads to other questions as to which encryption algorithms to use and how to handle key security. The next step would then be to actually secure the data traffic inside the GREASE framework and its surrounding infrastructure.

VI. Application Scenario

The proposed concept has been developed to manage several sensor network scenarios at our computer engineering

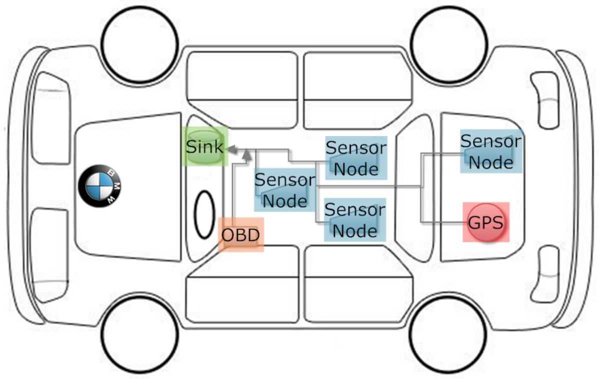
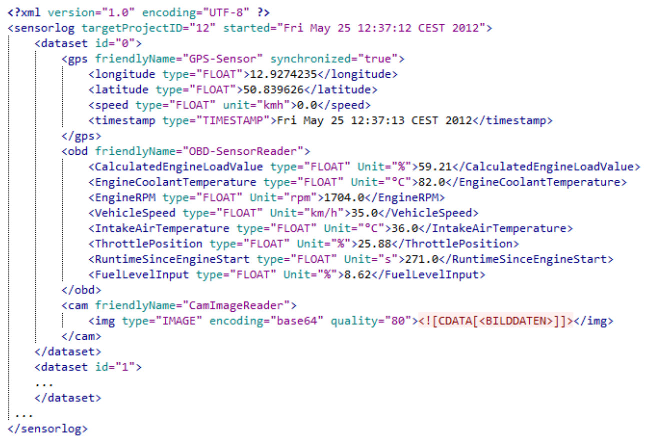


Figure 5: Measurement system - All data from the sensor nodes and the GPS module is transmitted to the data sink in the vehicle



```
<?xml version="1.0" encoding="UTF-8" ?>
<sensorlog targetProjectID="12" started="Fri May 25 12:37:12 CEST 2012">
    <dataset id="0">
        <gps friendlyName="GPS-Sensor" synchronized="true">
            <longitude type="FLOAT">12.9274235</longitude>
            <latitude type="FLOAT">50.839626</latitude>
            <speed type="FLOAT" unit="kmh">0.0</speed>
            <timestamp type="TIMESTAMP">Fri May 25 12:37:13 CEST 2012</timestamp>
        </gps>
        <obd friendlyName="OBD-SensorReader">
            <CalculatedEngineLoadValue type="FLOAT" Unit="%">59.21</CalculatedEngineLoadValue>
            <EngineCoolantTemperature type="FLOAT" Unit="°C">82.0</EngineCoolantTemperature>
            <EngineRPM type="FLOAT" Unit="rpm">1704.0</EngineRPM>
            <VehicleSpeed type="FLOAT" Unit="km/h">35.0</VehicleSpeed>
            <IntakeAirTemperature type="FLOAT" Unit="°C">36.0</IntakeAirTemperature>
            <ThrottlePosition type="FLOAT" Unit="%">25.88</ThrottlePosition>
            <RuntimeSinceEngineStart type="FLOAT" Unit="s">271.0</RuntimeSinceEngineStart>
            <FuelLevelInput type="FLOAT" Unit="%">8.62</FuelLevelInput>
        </obd>
        <cam friendlyName="CamImageReader">
            <img type="IMAGE" encoding="base64" quality="80"><![CDATA[<BILDDATEN>]]></img>
        </cam>
    </dataset>
    <dataset id="1">
    ...
    </dataset>
 ...
</sensorlog>
```

Figure 6: An excerpt from an XML log file produced by GREASE's standard persistence plugin

department. To clarify functional aspects of the implemented framework, we demonstrate the data processing flow using a real-world automotive measurement system [13]. For this monitoring scenario, the existing sensor components of a given research vehicle were upgraded with high-definition sensor nodes. These nodes are placed at predefined positions to monitor the entire environment and provide independent measurement data about the current temperature, light intensity as well as the acceleration in two axes and the magnetic field strength. Thus, the established wireless sensor network provides information about the measurement environment and external parameters. *Figure* 5 illustrates this measurement scenario.

The wireless sensor communication infrastructure is based on the *IEEE 802.15.4* and *ZigBee* [14][15] standards. Additionally, mobile sensor nodes are worn by the passengers. In order to realise localisation features for these nodes, they are equipped with *nanoPAN* ultra-low power network interfaces [16], which provide *RSSI*-based *(Received Signal Strength Indication)* distance information. Both communication technologies use the 2.4 GHz frequency spectrum for data transmission. A multi-interface, multi-standard data sink is able to handle both communication standards simultaneously. Robust communication stacks with adapted layer 2 and layer 3 protocols minimise interference-based influences on the communication behaviour.

In addition, we integrated a high-resolution *GPS (Global Positioning System)* sensor, which enables the correlation between absolute positioning information, speed, altitude and the available on-board vehicle data. We also established a connection with the vehicle's *OBD (On-board diagnostics)* interface in order to retrieve further information specific to the manufacturer and vehicle model.

By providing a synchronised knowledge base of all sensor information, a detailed analysis of specific driving situations and the driver behaviour is possible. Thereby, the GPS data allows a verification of these situations based on available track information. Accordingly, we are able to calculate and predict driver profiles. The results are used to adjust and to optimise the characteristics of the entire vehicle, for instance the engine management system or the suspension dynamics. This leads to the creation of in-depth driver profiles to adjust said car characteristics even prior to the engine's ignition. For this a driver must already be known to the system and enough data needs to be collected about their driving style beforehand. By fusing the location data with other sensor readings it is also possible to link peculiar measurement values with similar driving situations. This data can then be used to detect or predict similar route characteristics based on information retrieved from the knowledge base and thus critical situations may be identified more easily when assessing real-time data and appropriate countermeasures can be taken more quickly. Furthermore, an analysis of the wear measuring quantity provides interesting statements about the vehicle lifetime. For instance alterations in both the noise and vibration behaviour of components in the engine bay of a vehicle may indicate a damaged or soon-to-break part when compared to information in the knowledge base. For the measurement environment described here, particular *SensorReader* modules for the data sink communication interfaces were implemented. Incoming data from the sensor network is classified and converted into abstract data objects, which are transmitted to the *controller*. Another *SensorReader* module implements the necessary features for the GPS (Global Positioning System) data input. Thereby, the *NMEA 0183 (National Marine Electronics Association 0183)* protocol for the positioning data is put to use. Accordingly, all kinds of GPS hardware, which support the NMEA protocol and the serial port as communication interface, are supported.

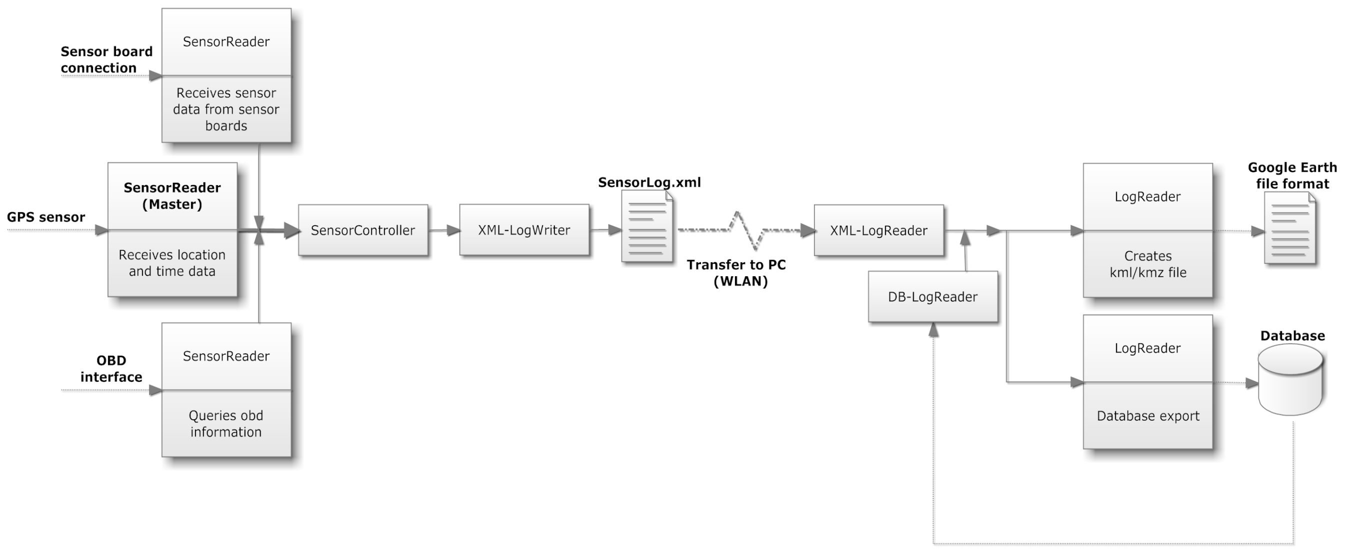


Figure 7: Sample application scenario of the proposed framework.

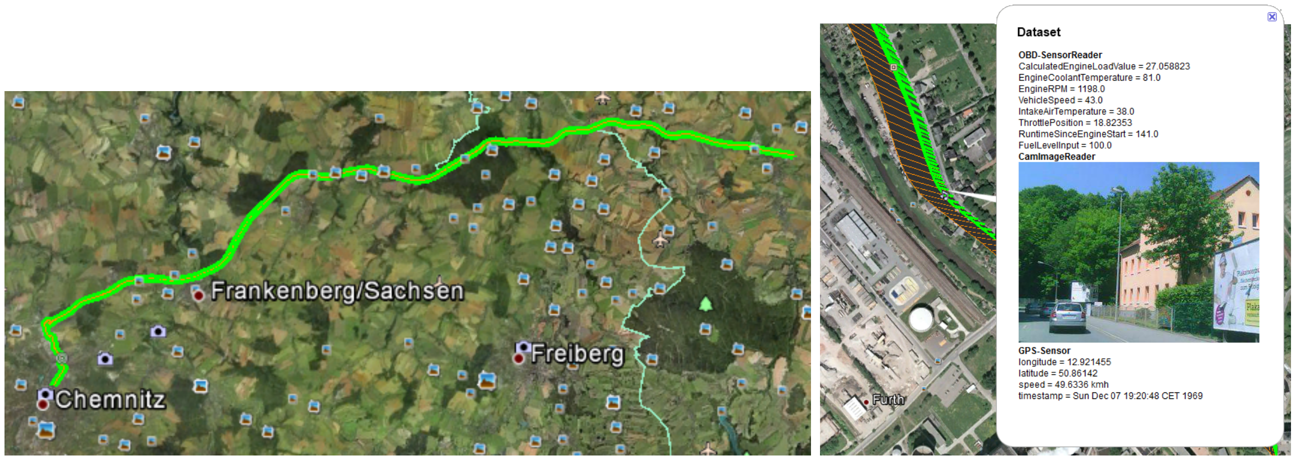
For the synchronisation of the sensor data, one specific *SensorReader* has to be predefined during the initialisation of the measurement scenario. In our example case, the system contains a GPS unit which provides positioning information as well as an accurate time signal. In consequence, the given timestamps from the GPS sensor represent the global *synchronisation master*. However, the framework is not restricted to using a timestamp as master index. Especially for the integration of multiple, autonomous sensor systems and a missing central scheduling entity, a user-defined choice for the synchronisation master provides important benefits. For instance this could be the activity of a specific sensor which detects predefined events or starting selective aggregation when certain sensor values exceed predetermined thresholds. On the one hand this helps in reducing the amount of sensor data to collect and store, whereas on the other hand it allows to group pieces of information which correlate based on time or event-wise. The storage of the retrieved datasets is handled by the standard persistence plugin of GREASE, which produces an XML file. An example of such a log file can be seen in *Figure* 6.

For post-processing and evaluating the collected data, two data analysis components were implemented. The first one is an export module, which prepares the sensor data sets for the storage in a given database system and accordingly transmits the chosen information. A second module is responsible for converting the sensor data with dedicated visualisation plugins, e.g., for Google Earth. Hence, the data output of this module is a *KML* (Keyhole Markup Language, file format of the Google Earth software) representation of all correlated sensor information or KMZ (zipped KML including resources such as images), if additional data has to be included. *Figure* 7 describes the data flow in this specific scenario. All developed modules are as generic as possible and may be used for a broad range of possible application scenarios and hardware-configurations in the future. This also includes a high compatibility level for both hardware and software environments [17][2]. For a detailed evaluation the import of measured data into JBEAM is possible and reasonable. This can be accomplished by using either the *CSV* (Comma Separated Values) export module or a specific jBEAM plugin, which implements the ASAM standard. With the latter the log file data can be imported directly into jBEAM which makes it even easier to assess and visualise large amounts of measured data in a comfortable and appealing way.

VII. Data Analysis

The following figures show some visualisation possibilities which represent the basis for further analyses. However, the actual evaluation and interpretation of the sensor data collected in the scenario described in *Section* VI is not part of this paper. Therefore, the figures may only be seen as examples with respect to the wide range of evaluation and visualisation possibilities the framework provides.

The Google Earth export module offers the conversion of the sensor data into the Google Earth file format (KML) which allows the user to use the Google Earth application for further analysis of the data. For this process, every dataset has to contain positioning information which can be retrieved from a respective sensor (for instance a dedicated GPS module). The module then creates a corresponding route for the GPS data (see *Figure* 8a) and a waypoint for every dataset. If the distance of two waypoints is smaller than a given threshold they may be fused for the sake of clarity. The user is now able to get detailed information for all waypoints in a corresponding pop-up window (see *Figure* 8b), this includes all sensor values contained in this



Dataset

OBD-SensorReader
CalculatedEngineLoadValue = 27.058823
EngineCoolantTemperature = 81.0
EngineRPM = 1198.0
VehicleSpeed = 43.0
IntakeAirTemperature = 38.0
ThrottlePosition = 18.82353
RuntimeSinceEngineStart = 141.0
FuelLevelInput = 100.0
CamImageReader

GPS-Sensor
longitude = 12.921455
latitude = 50.86142
speed = 49.6336 kmh
timestamp = Sun Dec 07 19:20:48 CET 1969

(a) The whole route (b) The details of a waypoint

Figure 8: Visualisation of a driven route and a single waypoint in Google Earth

dataset and any attached meta-information such as physical units or environmental circumstances. Furthermore, sensor value curves, such as the course of the vehicle speed, can be shown alongside the route as an additional three-dimensional altitude track. The altitude values of such a curve represent the respective sensor values as shown in *Figure* 9. Based on the timestamp data in the log file, Google Earth allows the user to view the course of the measurement by using its time bar. Only the appropriate waypoints for the selected time range will be shown. By animating this process the user can be provided with a comprehensive impression of the test drive. Additionally, each measurement scenario can be separated into different parts, for instance based on times when the vehicle did not move for a specific amount of time or if two adjacent waypoints have a distance greater than a defined value.

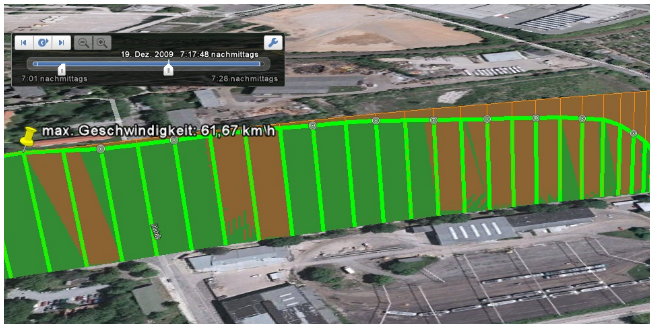

19. Dez. 2009 7:17:48 nachmittags
7:01 nachmittags
7:28 nachmittags
max. Geschwindigkeit: 61,67 km\h

Figure 9: Value curves along the driven route in Google Earth

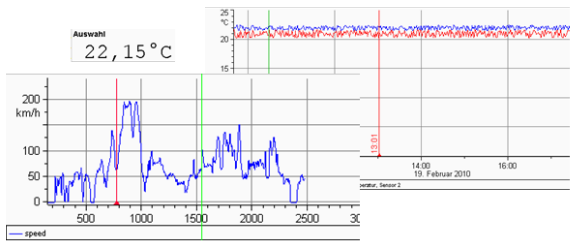
Other commercial software tools, e.g., FlexPro or jBEAM, are able to import the sensor data either by using export plugins such as the default CSV export module or through application-specific plugins which can the storage format produced by a GREASE persistence plugin. These third-party tools allow advanced, sectoral data post-processing tasks and offer advanced possibilities for the statistical analysis as well as the visualisation of the collected sensor data. Yet, the GREASE data flow and data measurement tasks remain unaffected regardless of the software tool, which shall be used for further tasks. A further sensor visualisation within jBEAM is shown in *Figure* 10a. In this figure there are two graphs which contain speed versus time and temperature versus time curves of a test drive. With jBEAM, classical value curves as well as maps with overlay information (as shown in *Figure* 10b) can be generated very easily. The entire data processing flow integrates all proposed features for the data recording, handling and visual representation. Since GREASE's aggregation phase and data flow are completely independent of the successive data evaluation the user can still benefit from the great variety of already existing software tools for this task. They may even change the used solution after already being halfway through a measurement project without problems or use several of such tools alongside each other. Besides, the framework also allows for a great flexibility for instance in adopting ever-changing measurement requirements and exposes a great reusability for upcoming projects.

VIII. Conclusion and Future Work

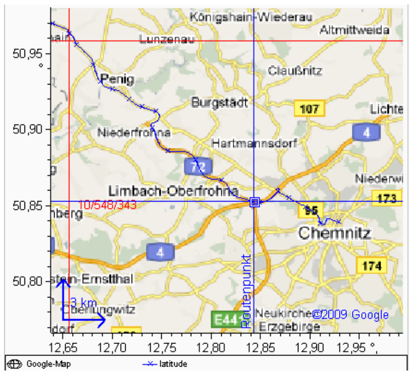
The proposed research work described the concept and implementation of a comprehensive data processing environment for heterogeneous sensor or sensor-actor-systems. The basic concept provides generic structures for many further research projects in the field of novel data aggregation and data fusion techniques. For an easy data collecting and data analysis process, we are now able to synchronise and correlate the single data sets also on resource limited and embedded computer systems. The result is a common and extensive knowledge base, which integrates all information sources into complex data sets.

In comparison to other related software tools, the proposed framework fulfils essential requirements for a flexible usage, a resource-efficient runtime behaviour as well as an automated or semi-automated operating mode. We developed a standardised process for monitoring and archiving data from a heterogeneous network topology in a synchronised way. Besides storing basic information from given sensor data sets, the system also integrates meta-information from the environment to increase the reuse factor of the measurement scenario. The universal XML data representation and a modular plugin system ensure a generic usage for all kind of sensor scenarios. Multiple data input and output interfaces provide a high level of compatibility to other software tools and data formats. The presented framework is used for several wireless sensor network projects at the Chemnitz University of Technology.

Regarding the presented automotive application scenario, the proposed framework enables correlations between the measured sensor data from the test track and specific driver profiles. Accordingly, these information allow dynamic adaptations of the driving parameters within the vehicle. This offers novel and interesting possibilities to optimise a vehicle for the specific characteristics of its driver.



(a) Value curves with markers for the currently selected waypoint



(b) A map with the driven route, showing the selected waypoint

Figure 10: Visualisation of sensor data in jBEAM

REFERENCES

[1] M. Vodel, R. Bergelt, and W. Hardt. Grease framework - generic reconfigurable evaluation and aggregation of sensor data. In *Proceedings of the 2nd International Conference on Smart Grids, Green Communications and IT Energy-aware Technologies (ENERGY2012 / InfoSys2012)*, page no pages given. IARIA, March 2012.

[2] M. Vodel, R. Bergelt, M. Glockner, and W. Hardt. Synchronised data logging, processing and visualisation in heterogeneous sensor networks. In *Proceedings of the International Conference on Data Engineering and Internet Technology*. Springer, March 2011.

[3] National Instruments. LabView. http://www.ni.com/labview/, 2010. [Online, retrieved: January, 2012].

[4] AMS GmbH. jBEAM. http://www.jbeam.de/german/produkte/jbeam.html, 2010. [Online, retrieved: January, 2012].

[5] Weisang. FlexPro. http://www.weisang.com/, 2010. [Online, retrieved: January, 2012].

[6] ASAM Consortium. Association for Standardisation of Automation and Measuring Systems. http://www.asam.net/, 2010. [Online, retrieved: January, 2012].

[7] L. Krishnamachari, D. Estrin, and S. Wicker. The impact of data aggregation in wireless sensor networks. In *Proceedings of the 22nd International Conference on Distributed Computing Systems Workshops*, pages 575–578. IEEE Computer Society, November 2002.

[8] M. Vodel, M. Lippmann, M. Caspar, and W. Hardt. Distributed high-level scheduling concept for synchronised, wireless sensor and actuator networks. *Journal of Communication and Computer*, 11(7):27–35, November 2010.

[9] V. Gupta and R. Pandey. Data Fusion and Topology Control in Wireless Sensor Networks. *WSEAS Trans. Sig. Proc.*, 4(4):150–172, 2008.

[10] H. Qi, S. S. Iyengar, and K. Chakrabarty. Distributed Sensor Fusion - A Review Of Recent Research. *Journal of the Franklin Institute*, 338(1):655–668, 2001.

[11] H. Qi, X. Wang, S. S. Iyengar, and K. Chakrabarty. Multisensor Data Fusion In Distributed Sensor Networks Using Mobile Agents. In *Proceedings of International Conference on Information Fusion*, pages 11–16, August 2001.

[12] A. Brown. *Component-Based Software Engineering*. Wiley-IEEE Computer Society Press, 1996.

[13] M. Vodel, M. Lippmann, M. Caspar, and W. Hardt. A Capable, High-Level Scheduling Concept for Application-Specific Wireless Sensor Networks. In *Proceedings of the World Engineering, Science and Technology Congress*, pages 914–919. IEEE Computer Society, June 2010.

[14] IEEE Computer Society. Part 15.4: Wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (WPANs). http://standards.ieee.org/getieee802/download/802.15.4-2006.pdf, 2007. [Online, retrieved: January, 2012].

[15] Zigbee Alliance. Zigbee specification. http://www.zigbee.org/en/spec_download/zigbee_downloads.asp, 2007. [Online, retrieved: January, 2012].

[16] Nanotron Technologies. Nanotron's transceiver enables iso compliant real time locating systems. In *The International Organization for Standardization and the International Electrotechnical Commission*, volume 24730-5:2010. New standard for Real Time Locating Systems (RTLS), April 2010.

[17] M. Vodel, W. Hardt, R. Bergelt, and M. Glockner. Modulares Framework fr die synchronisierte Erfassung, Verarbeitung und Aufbereitung heterogener Sensornetzdaten. In *Proceedings of the Dresdner Arbeitstagung Schaltungs- und Systementwurf*, pages 67–72. Fraunhofer Institute for Integrated Circuits, May 2010.

Dr. Matthias Vodel was born in Germany in 1982. He received the German Diploma degree (equal to M.Sc.) in Computer Science from the Chemnitz University of Technology with the focus on computer networks and distributed systems in 2006. In 2010, he received his Ph.D. degree in Computer Science from the Chemnitz University of Technology / Germany.
Currently, he works as a postdoctoral research fellow at the Department of Computer Science, Chair of Computer Engineering at Chemnitz University of Technology, Germany. His latest research projects focus on energy-efficient optimisation strategies in distributed embeddet systems. Additional fields of interest include network security topics, protocol engineering and wireless communication standards.
For his Ph.D. thesis, he received the "Commerzbank Award" for outstanding research work. He has published more than 40 journal and conference papers as well as two books. In 2008, Dr. Vodel received the best paper award for the conference paper "EBCR - A Routing Approach for Radio Standard Spanning Mobile Ad Hoc Networks". In 2012, the paper "WRTA - Wake-Up-Receiver Optimised Routing and Topology Optimisation Approach" received the best paper award at the 12th International Conference on ITS Telecommunications.
Dr. Vodel is member of the Association for Electrical, Electronic and Information Technologies (VDI/VDE) as well as IEEE member.

Dipl.-Inf. René Bergelt was born in Germany in 1988. He received the German Diploma degree (equal to M.Sc.) in Applied Computer Science with focus on embedded systems at the Chemnitz University of Technology in 2012. Currently, he is a Ph.D. student at the Department of Computer Science, Chair of Computer Engineering. The main fields of his research are methods for energy-efficient data aggregation and data fusion in wireless sensor networks. He also works on automotive applications for wireless sensor networks in the area of vehicle-to-vehicle and vehicle-to-environment communication strategies.

Prof. Dr. Wolfram Hardt is professor for computer science and head of the Computer Engineering Group at the Chemnitz University of Technology. He was born Germany 1965 and received the German Diploma degree (equal to M.Sc.) in Computer Science in 1991 from the University of Paderborn. Accordingly, Prof. Hardt received the Ph.D. degree in Computer Science from the University of Paderborn in 1996.
From 2000 to 2002 he was chair of the Computer Science and Process Laboratory at the University of Paderborn / Germany. Since 2003 Prof. Hardt became chair of the computer engineering Dept. at the Chemnitz University of Technology / Germany. He is editor of a scientific book series about self-organising embedded systems and has published more than 100 papers.
Prof. Hardt is member of the Association for Electrical, Electronic and Information Technologies (VDI/VDE), the Association for Computer Science (GI) and the Association for Computing Machinery. Since 2006 he is committee member of the DATE conference - "Design Automation and Test in Europe". His research interests include Hardware/Software Co-Design and Reconfigurable Hardware.