# DEVELOPMENT OF AN OPEN-SOURCE AUTOMATIC DEFORMATION MONITORING SYSTEM FOR GEODETICAL AND GEOTECHNICAL MEASUREMENTS

Ph. Engel[a], B. Schweimler[a]*

[a] Faculty of Landscape Sciences and Geomatics, Hochschule Neubrandenburg, University of Applied Sciences,
(pengel, schweimler)@hs-nb.de

**KEY WORDS:** automatic deformation monitoring, engineering surveying, software development, open source

**ABSTRACT:**

The deformation monitoring of structures and buildings is an important task field of modern engineering surveying, ensuring the standing and reliability of supervised objects over a long period. Several commercial hardware and software solutions for the realization of such monitoring measurements are available on the market. In addition to them, a research team at the Neubrandenburg University of Applied Sciences (NUAS) is actively developing a software package for monitoring purposes in geodesy and geotechnics, which is distributed under an open source licence and free of charge. The task of managing an open source project is well-known in computer science, but it is fairly new in a geodetic context. This paper contributes to that issue by detailing applications, frameworks, and interfaces for the design and implementation of open hardware and software solutions for sensor control, sensor networks, and data management in automatic deformation monitoring. It will be discussed how the development effort of networked applications can be reduced by using free programming tools, cloud computing technologies, and rapid prototyping methods

## 1. INTRODUCTION

The automation of deformation measurements by means of computer systems started early. In the 1980s first efforts where made concerning the remote control of geodetical and geotechnical sensors, using personal computers and analog modems (Pelzer, 1988). Since then, many software and hardware systems for deformation monitoring have been developed in universities and corporations. Several proprietary deformation monitoring systems exist, either made by the manufacturers of sensors or by engineering offices for their own businesses. These solutions are often designed as isolated applications. The supported sensors are limited to specific types or producers, and interfaces for data exchange and remote control are not open to third party products to ensure a vendor lock-in.

Automated deformation measurements gain importance due to the increasing building density in public space and the advanced age of existing infrastructure. At the same time, the costs of sensors and information technology in general decrease, which may also facilitate their widespread usage in geodesy and geotechnics.

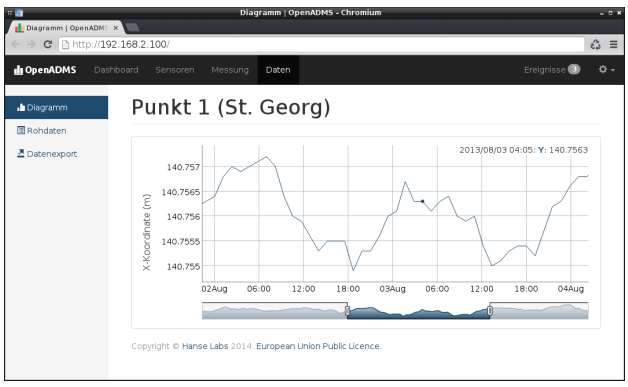


Figure 1: Web-based user interface of the upcoming version of OpenADMS

*Corresponding author

Despite the progress in terms of sensors and hardware appliances for remote control, an open software platform for deformation monitoring is still not available. For this reason, efforts are being made at NUAS to develop a universal open source monitoring solution with platform-independence, secure remote control functions using standardised interfaces for data exchange, and compatibility to cloud computing environments.

## 2. THE DABAMOS MONITORING SYSTEM

The development of an automatic deformation monitoring system at the Faculty of Landscape Sciences and Geomatics started in 2009 as a students project to simply control geodetical sensors remotely. The project was later called *Datenbank-orientiertes Monitoring- und Analyse-System* (DABAMOS). The first version of the software has been written entirely in Java SE and could be run on conventional personal computers without any special requirements, except the Java Runtime Environment. The integrated remote control interface allows users to start and stop the monitoring process through a TCP/IP network. All sensor data is stores in an object-oriented *db4o* database. Later on, the first steps towards a client–server architecture were taken with the development of an enterprise application written in Java EE for the storage and visualisation of monitoring data on Internet servers. But this application has never left prototype stage.

Coinciding with a rewrite of the software base in the Go programming language in 2013 the decision was made to publish the whole project under an open source licence. The new software was designed to be operated through a Web front-end with modern browsers (fig. 1) and to support ARM-based single-board computers, like the Raspberry Pi or the BeagleBone. These single-board computers can be used to control connected sensors.

The DABAMOS project consists of three parts (fig. 2):

*OpenADMS*: The *Open Automatic Deformation Monitoring System* is a platform-independent software for sensor control. The measured values of geodetical or geotechnical sensors are stored locally in a NoSQL database and then transmitted to an FTP server or an OpenSDMS instance.

*OpenSDMS*: The *Open Spatial Data Management Service* is used for the storage, analysis, and visualisation of sensor data in cloud computing environments. OpenSDMS collects and manages data from an arbitrary number of OpenADMS clients. The service is still in conceptual phase.

*Middleware*: In most cases, sensors have to be connected to a local computer system to be controlled remotely. Beside personal or industrial computers also embedded computers can be used to run OpenADMS. Such a "middleware" has to provide Internet access via Ethernet, WiFi, or wireless networks (3G/4G) and is possibly equipped with an uninterruptible power supply unit.

## 3. THE POTENTIAL OF OPEN SOURCE LICENCES

Free and open source development models sustain various software and hardware projects and make it easier for groups of people, who may not be acquainted, to work together in a collaborative way (Laurent, 2004).

The open source model leads to many advantages for users and developers of such projects. The fact that open source software is made available free of charge or at a low cost is thereby only the first perceived one, while even not exclusive to open source. More important is the availability of the source code and the right to modify it. It makes it feasible to improve the software by anyone and to port the application to new hardware or to adapt it to further conditions, which were not considered by the original authors.

Another advantage of the open source model is the possibility of splitting a project into subsequent ones ("forking"). The original project serves as a basis for ones with different aims. In case of an automatic deformation monitoring system this means that users can take the source code to develop software for their own purposes or to integrate it into a second, already existing project.

The DABAMOS software is published under the European Union Public Licence (EUPL) v. 1.1. The EUPL has been created on the initiative of the European Commission and is the first European Free/Open Source Software (F/OSS) licence, available in 22 linguistic versions. The EUPL is compatible to the popular GNU GPLv2 and shares its "copyleft" clause: derivated works of EUPL-licenced software must be published under the same or a compatible licence.

The decision to publish the DABAMOS software under the EUPL was pursued by the aim to establish a widely accepted and used platform for all kind of geodetical and geotechnical monitoring tasks. This is why all future work will be made freely available on the project website.

## 4. SOFTWARE DEVELOPMENT METHODS

Development teams often face several problems with changing and increasing requirements. This is even more of an issue when developing open source software. Traditional methods like the waterfall model are too rigid and prescriptive to handle small distributed teams and many different use cases. Software developed this way is only released when it is feature complete. This means, all of the functionality has to be developed when deploying the software. This is not an option for an open source project, as it is self-organised and often confronted with changing requirements, depending on the current user or contributor. Agile development methods can solve this problem.

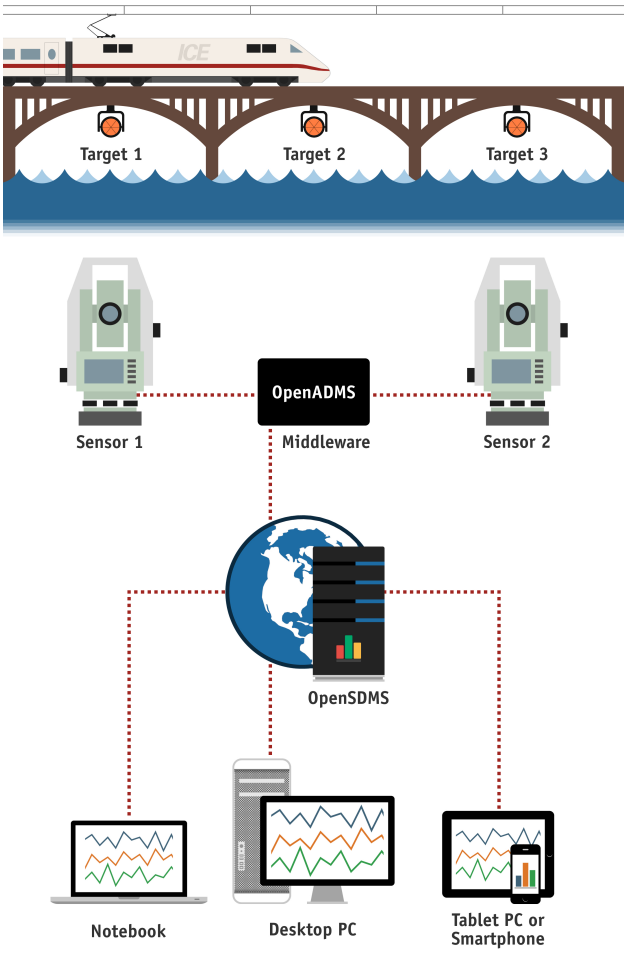


Figure 2: Schematic illustration of the DABAMOS platform

### 4.1 Agile development

Agile development offers a lightweight framework to feature evolving solutions. Those can grow with the problem and can adapt quickly to new situations. Agile development teams are generally self-organised and cross-functional. Many of the implemented methods focus on the user, forcing teams to deliver software early and continuous in frequent intervals. For measuring the work process, working software is almost all that matters. The principles are formulated in the Agile Manifesto (Agile, 2001).

### 4.2 Continuous Delivery

An actual implementation of Agile development is Continuous Delivery (CD). It describes a collection of tools, processes, and techniques to provide high quality software using a high degree of automation.

The actual development practice is the integration of source code of an application into a shared repository of a version control system at least daily. The code is provided to a processing chain, which builds the software immediately for all targeted platforms. All software tests are included and executed automatically by the processing chain.

By convention, every commit has to build on every platform. If a build fails, the developer gets an immediate feedback. The occurring error has to be fixed as soon as possible, which forces developers to commit small changes to the repository. This reduces

the time spend for debugging and backtracking significantly, as problems are detected early and can be located more easily. The outcome of the processing chain is the built bug-free software. The executables for the targeted platforms should be easy accessible and automatically be deployed to clients. (Humble/Farley, 2010)

### 4.3 DevOps

To build reliable software, developers have to understand what the software is used for and more importantly how it is used. Agile development describes procedures between developers. It does not overcome breaking points between the development and the actual operation of software.

DevOps, a clipped compound of Development and Operations, is an approach to build a bridge between developers and system administrators who run an application in a live environment. It solely aims for the delivery of reliable software and has to be implemented in a CD context. DevOps teams can fulfil all tasks needed, from developing to testing and finally administrating the software.

The communication is crucial for all team members to understand the underlying business logic of the software and to exchange information with actual users. To be part of a DevOps team, developers and operators need a wide skill set, as DevOps follows a multi-disciplinary approach. (Swartout, 2012)

The described paradigms and techniques are a small cut off of open source development methods. As the development includes many different persons, it is vital for the success of an open source project to react flexible to changing requirements and new situations. The following chapter describes the actual implementation of the described methods to manage and build DABAMOS.

## 5. MANAGING AND BUILDING AN OPEN SOURCE SOFTWARE PROJECT

A range of hardware and software tools for the management and development of open source engineering projects is available. The following section describes some of the tools used for the implementation of applications and appliances in the DABAMOS project.

### 5.1 Hardware

The control of sensors and the management of their collected data do require not only specialised software products but also hardware appliances for data transmission, persistence, and interaction. New technologies for rapid prototyping can shorten the development time of hardware solutions.

The analog or digital data of sensors can be read by several secondary devices, like personal computers or microcontrollers. For further processing, analog signals have to be turned into digital values first by using analog-to-digital converters (ADCs). Industrial-grade ADC modules are equipped with a serial port for data transmission to a connected computer, whereas geodetical sensors often have an internal digital interface for data exchange and remote control. Conventional personal computers are adequate for the communication with ADC modules and sensors but are often not suitable for a usage in harsh environmental conditions, such as dirt, heat, cold, wet, or vibration. For a higher level of resistance against environmental impact industrial or embedded computers are used.

In addition, so called single-board computers can also act as sensor control instances. The low-priced systems combine all required components (i. a., processor, memory, I/O, and permanent storage) of a computer on a single small-sized circuit board. Most single-board computers are based on the ARM or the MIPS processor architecture and can be run with GNU/Linux or open source Unix operating systems. Popular single-board computers are, beside others, Rasperry Pi (fig. 3), BeagleBone, and Cubieboard. With prices under 100 Euros per board they are a low-cost alternative to industrial and conventional embedded computers and a convenient basis for the rapid prototyping of sensor control units.

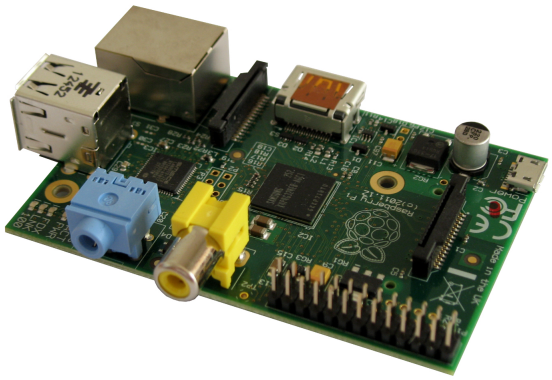

Figure 3: Low-cost single-board computer “Raspberry Pi”

For many applications custom-built cases, moutings, and adapters are required. 3D printers can be used to manufacture single units and small series of such physical components. The work pieces are constructed by 3D printers in layers of liquid or solid material, commonly plastics. These pieces are produced from a three-dimensional CAD model and can be of almost any shape. The Faculty of Landscape Sciences and Geomatics of the Neubrandenburg University of Applied Sciences owns a MakerBot Replicator Z18 3D printer for research and educational purposes (fig. 4). The printing is done in fused deposition modeling technique, where heated thermo-plastics are deposited on a movable table (Grimm, 2004). The printing resolution of 0.1 mm allows the precise construction of fine structures. Depending on the size of the work piece, the printing process can take several hours. Because the used polylactic acid filament is not weatherproof, the printed work pieces should be used indoor only and not be exposed to water and excessive UV radiation. The design of the models can be done in most CAD/CAAD software capable of exporting files in STL format, like Autodesk Inventor, SketchUp, or the open source programs FreeCAD and Blender.

### 5.2 Software

Modern programming languages and their tools are often published under an open source licence and therefore also suited to manage and build open source projects. This reduces the dependency to single vendors, as it is, for instance, the case for the .NET platform of Microsoft.

### 5.3 Programming language

The software of the DABAMOS project is mostly written in the Go programming language, which is published under a BSD-style licence. The source code of Go is hosted publicly on GitHub and all tools to build and test the software are available for free. The language enables developers to construct flexible and modular programs. It provides concurrency mechanisms, garbage collection, and run-time reflections. (Go, 2015)

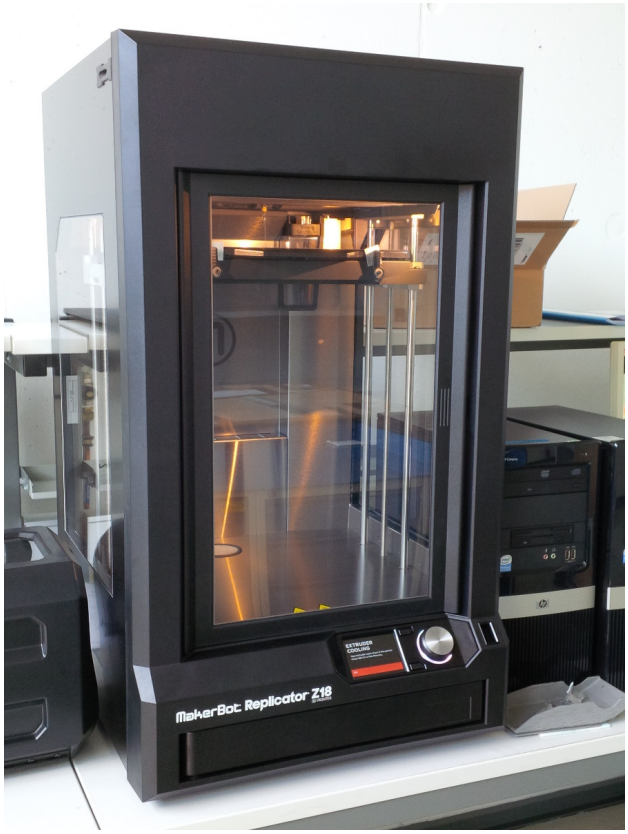


Figure 4: 3D printer “MakerBot Replicator Z18”

The language supports common computer architectures, including ARM, and most modern operating systems. It is not necessary to build the code directly on the specified computer platform, as it can easily be compiled by setting up flags in the development environment (cross-compiling). The languages concurrency mechanisms support techniques that are actively used in the DABAMOS software. It provides so called “Go routines”, which are similar to threads in other programming languages. Additionally, there are tools to synchronise and to handle the communication between Go routines.

Many processes are executed simultaneously in a monitoring system. The Go programming language makes it possible to manage those processes. As DABAMOS is a distributed platform, the development also benefits from integrated networking libraries, which are related to open standards, like Transport Layer Security (TLS) for encrypted communication, as well as Extensible Markup Language (XML) and JavaScript Object Notation (JSON) for data exchange.

### 5.4 Version control system

The source code of the DABAMOS software is managed with Git, an open source version control system. Git can handle software projects of any size and supports Continuous Delivery methods, especially by providing tools for tagging the source code. The typical workflow starts with the addition of a source code file to the local workspace. The change is described by a message and finally committed to the local repository. The commit is afterwards pushed to a central remote repository. As there is often more than one contributor, it is common to use branches, isolated from each other, for the development of features. After the implementation is finished, a branch can be merged into the master branch, which is the default branch of a Git repository. This way, many developers can evolve complex software without interfering each other. (Preiel, Stachmann, 2013)

GitHub, a repository hosting service, is used as the remote repository for the DABAMOS software which is free of charge for open source projects. The service adds additional so called “social coding” features to Git. Developers can set up several collaboration functions to manage their project and its contributors. (GitHub, 2015)

### 5.5 Networks

For the management and organisation of the DABAMOS project the team uses an Intel Xeon server with five hard disks, running FreeBSD 10, an open source Unix operating system. The collocation of the server is provided by the University, while the Internet access is available through the X-WiN backbone of the national research and education network DFN. The server is mainly used for serving the project website and the project documentation, for file exchange, project management, continuous integration, and instant messaging, but also as a SSH gateway for remote sensor nodes.

The project network infrastructure has also been equipped with a virtual private network (VPN) hardware gateway to establish encrypted tunnel connections based on IPsec/L2TP. A VPN can be used to integrate remote sensor nodes into a local network, which makes it possible to establish serial data connections through the Internet. To access sensors remotely through an existing TCP/IP network so called “serial device servers” are an easy to use option. The serial device servers are connected to a VPN hardware client which establishes an encrypted tunnel connection to the remote VPN gateway. In this way the local sensors become part of the whole VPN network. They can be accessed and controlled by any privileged user inside the VPN network, even if the user is far away from the sensor location.

The establishment of encrypted tunnel connections is not limited to the operation of hardware appliances. Software-based alternatives, like SSH and OpenVPN, also exist and do not require the acquisition of hardware solutions.

### 5.6 Documentation

An important task in software development is the writing of a comprehensive documentation for end users and developers. The documentation of the source code is often done by the respective programmer using special comments and tags within the source code. Depending on the programming language, the developer can then create a HyperText-based source code documentation with documentation generators, like “godoc” for Go, or “Javadoc” for Java.

The end user documentation can be created in several ways. In the past, user manuals were often available as a printed work only. But this kind of offline documentation is no more appropriate, since printing and distribution of user handbooks is both time-consuming and expensive. A further disadvantage of printed manuals is that the user has to receive an updated copy for each new software version, which makes the update process more complex.

Therefore, a user manual is often available twice: as a printed hard copy as well as a digital copy in the Portable Document Format (PDF). In comparison with HyperText-based online manuals PDF files are still geared to be printed on paper, which is why they are based on a fixed format while being less interactive for the user.

For a modest online publication of user manuals the HyperText Markup Language (HTML) is more suitable. The manual is simply written or transformed to HTML and uploaded to a web server, so that all users have access to it. Apart from HTML, further markup languages exist, some particularly for the creation of online manuals. These languages are less complex than HTML and therefore easier to learn (e. g., Markdown, reStructured-Text, Textile, and AsciiDoc).

A further alternative to printed books are wikis. They are Web-based applications which allow users to work on content in a collaborative way. The content is structured by a markup language and later exported to HTML. All changes on the content are tracked by the internal version control system of the wiki and can be set back anytime. Popular open source wiki systems are MediaWiki, which is also used for the Wikipedia encyclopedia, and Doku Wiki. Within the DABAMOS project a wiki, based on the DokuWiki software, is used for end user and developer documentation.

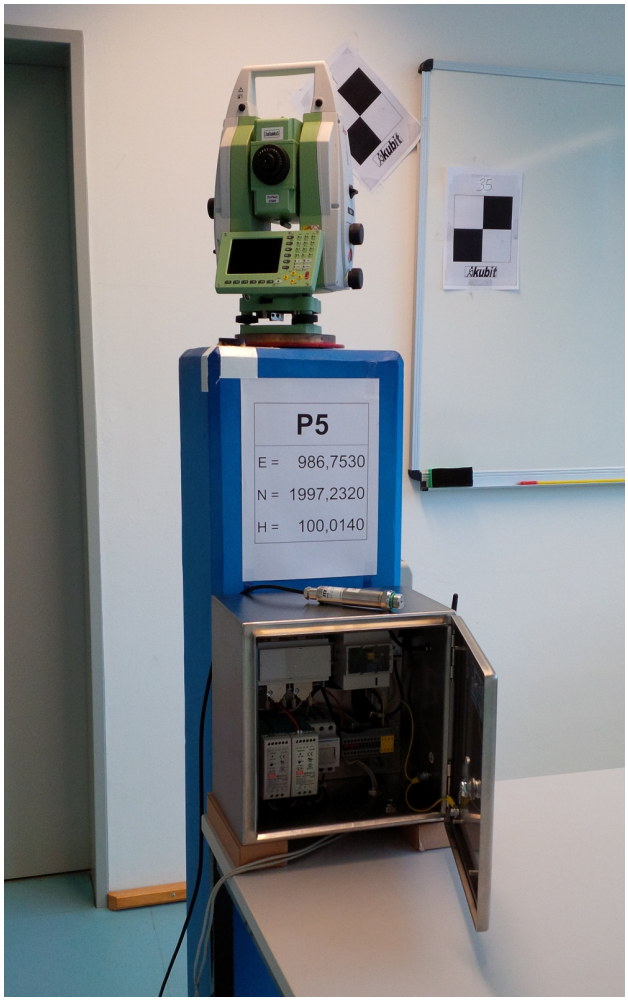


Figure 5: Monitoring station with Leica TM30 totalstation and embedded sensor control unit, based on a Raspberry Pi

## 6. A PLATFORM FOR GEO-SENSOR NETWORKS IN CLOUD-COMPUTING ENVIRONMENTS

A geo-sensor network consists of sensors with a spatial reference, connected to wired or wireless transmission technologies. A single sensor node is generally formed by the sensor itself and a computer to request, store, and forward collected measurement data. The sensor can transmit the data on its own without caching it on a computer. Usually, information from all sensor nodes is collected in a central storage and processed afterwards.

The central storage of DABAMOS is called OpenSDMS. It is integrated into geo-sensor network and connects all system components. Conceptualised as a platform, OpenSDMS is used to manage monitoring projects and configure sensor nodes centrally. The platform has an expandable toolset to analyse the collected data and to create reports summarising the results.

Core features of such a platform are accessibility and scalability. As the grade of utilisation is unknown before operating, it is important for the system to scale dynamically with changing requirements. This is a typical scenario for cloud computing environments. The cloud represents an abstract model of configurable computing resources, which are accessed over a network.

The management of the cloud infrastructure is handled by a service provider. For DABAMOS the service model “Platform as a Service” (PaaS) is used. PaaS provides the cloud infrastructure and additional tools and services, supporting the operation of the software. The user does not manage nor control the underlying resources (e. g., network, servers). The application installed in such an environment can acquire resources to match higher utilisation as well as release resources when being confronted with lesser load. This process is automated.

The actual application, deployed in the cloud, consists of two components: the application programming interface (API) and the graphical user interface (GUI). The API provides the general functionality to execute specific tasks, the so called business logic. It is based on “Representational State Transfer” (REST), a software architecture model for distributed systems. REST is designed for machine-to-machine communication, which makes it applicable for the automatic communication between sensor nodes and OpenSDMS. The architecture uses the HyperText Transfer Protocol (HTTP) and its methods to access resources via a uniform resource identifier (URI).

When implementing REST with the constraint of “Hypermedia as the Engine of Application State” (HATEOAS) any client is able to navigate the interface via URIs that are exclusively provided by the server. This technique reduces dependencies between the server and the clients. It allows developers to evolve the server API without having to worry about client incompatibilities. (Fielding, 2000)

## 7. CURRENT TASKS AND UPCOMING FEATURES

The development of the monitoring system DABAMOS is still work in progress. There is likely to be incompatibilities with different platforms or certain sensors, which will be fixed as soon as possible. However, in the future, the DABAMOS development team has two priorities to improve the project.

One future task is the integration of OpenSDMS into consisting geo-sensor networks. A live system is running on the roof top of the faculty. The system consists of a totalstation, which monitors points in the surrounding area. For the detection of temperature and air pressure corresponding sensors are integrated as well. The current set-up will be replaced by a new prototype of the embedded sensor control unit (fig. 5). At the moment, the station is configured locally. In the future, the station will be connected to the cloud service OpenSDMS, to allow extensive remote control.

The main purpose to operate such a system is to test new features under real conditions. Another intention of the team is to use the

project as a showcase for possible users and contributors. The developed solution will allow interested to view and change the configuration settings, after OpenSDMS is integrated. The data will be shown in interactive charts and additionally be provided to the public for download.

As the monitoring with total stations is rather common, it is another priority to extend the feature list by integrating low-cost GNSS receivers. These improvements are part of an ongoing research project at NUAS.

## 8. REFERENCES

Agile, 2015. Manifesto for Agile Software Development. `http://www.agilemanifesto.org/`. Visited in September 2015.

Fielding, R., 2000. Architectural Styles and the Design of Network-based Software Architectures. University of California, Irvine.

Humble, J.; Farley, D., 2010. Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. Addison-Wesley. Mnchen.

GitHub, 2015. GitHub source code hosting service. `https://github.com/`. Visited in September 2015.

Go (2015). Go programming language. `https://golang.org/`. Visited in September 2015.

Grimm, T., 2004. Users Guide to Rapid Prototyping. Society of Manufacturing Engineers, Dearborn, MI.

Laurent, A. M. St., 2004. Understanding Open Source and Free Software Licensing. OReilly Media, Sebastopol, CA.

Pelzer, H., 1988. Ingenieurvermessung. Deformationsmessungen. Massenberechnung. Ergebnisse des Arbeitskreises 6 des Deutschen Vereins fr Vermessungswesen (DVW) e. V. Herausgegeben von Hans Pelzer. Wittwer, Stuttgart.

Preiel, R.; Stachmann, B., 2013. Git: Dezentrale Versionsverwaltung im Team Grundlagen und Workflows. dpunkt.verlag, Heidelberg.

Swartout, P., 2012. Continuous delivery and DevOps: A Quickstart Guide. Packt Publishing, Birmingham.