A versatile software architecture for civil structure monitoring with wireless sensor networks

Kallirroi Flouri*[1], Olga Saukh[2], Robert Sauter[3], Khash Erdene Jalsan[1], Reinhard Bischoff[4], Jonas Meyer[4] and Glauco Feltrin[1]

[1]*Structural Engineering Laboratory, Empa Dübendorf, Switzerland*
[2]*Computer Engineering and Networks Laboratory, ETH Zürich, Switzerland*
[3]*Networked Embedded Systems Group, University of Duisburg-Essen, Germany*
[4]*Decentlab GmbH, Ueberlandstrasse 129, CH-8600 Dübendorf, Switzerland*

(Received January 5, 2012, Revised July 9, 2012, Accepted July 11, 2012)

Abstract. Structural health monitoring with wireless sensor networks has received much attention in recent years due to the ease of sensor installation and low deployment and maintenance costs. However, sensor network technology needs to solve numerous challenges in order to substitute conventional systems: large amounts of data, remote configuration of measurement parameters, on-site calibration of sensors and robust networking functionality for long-term deployments. We present a structural health monitoring network that addresses these challenges and is used in several deployments for monitoring of bridges and buildings. Our system supports a diverse set of sensors, a library of highly optimized processing algorithms and a lightweight solution to support a wide range of network runtime configurations. This allows flexible partitioning of the application between the sensor network and the backend software. We present an analysis of this partitioning and evaluate the performance of our system in three experimental network deployments on civil structures.

Keywords: structural health monitoring; wireless sensor networks; monitoring software; bridge monitoring; real-life deployment; TinyOS; cable stayed bridge

1. Introduction

In civil engineering practice, monitoring of civil structures is currently done with wired systems. These mature systems combine high-fidelity sensor values with a reliable and robust system performance. However, the installation (primarily the cabling) is very time-consuming and expensive. Many application scenarios benefit from using wireless sensor networks (WSNs) due to their attractive properties of being cable-free and easy to deploy, which allows minimizing installation cost and time. Furthermore, due to their self-configuring features, WSNs allow for a plug and play installation thus substantially simplifying the deployment process.

Although WSN technology allows considerable cost reduction, actual deployments of wireless systems are still very limited due to numerous challenges that need to be solved: handling reliably high sampling rates and large data volumes, providing the necessary support for parallel data acquisition with different sensors, assuring overall system reliability and stability, and achieving an

*Corresponding author, Dr., E-mail: kallirroi.flouri@empa.ch

acceptable system lifetime. Moreover, WSNs still need to offer enough transparency for civil engineers to concentrate on assessing the state of the structure by easily tuning acquisition parameters and reconfiguring parts of the measurement system. Several researchers have employed wireless sensor networks to monitor structures (Lynch *et al*. 2006, Nagayama and Spencer 2007, Kim *et al*. 2007, Jang *et al*. 2011), providing important insight into the opportunities and challenges of WSN technology for structural monitoring. Critical issues identified include: (i) power management, (ii) autonomous operation, (iii) high fidelity data acquisition, (iv) high sampling rates, (v) energy harvesting, (vi) fault tolerance, and (vii) environmental hardening.

Since the requirements of a WSN monitoring system are application-specific, existing systems are usually optimized for a certain type of measurements and operation mode. Jang *et al*. (2010) deployed a WSN monitoring system on the Jimbo Bridge, a cable-stayed bridge in South Korea, with excessive wind and vibration triggering the system to initiate monitoring. Lei *et al*. presented a two step Kalman estimator and least squares estimation approach, implemented on a hierarchical WSN for structural damage detection in a multi-store building. Another long term wireless monitoring system, developed by Kurata *et al*. (2011), for monitoring long-span bridges has been deployed on the New Carquinez Suspension Bridge in Vallejo, CA. This WSN system is organized in sub-networks with several base stations, which communicate the data using cellular modems to a powerful data base on the Internet. Casciati and Chen (2011) reviewed the recent development of several WSNs designs and applications, emphasizing the state of practice of WSNs in structural health monitoring. On the contrary to the previous approaches in the literature, our work emphasizes on the use of one WSN system for several applications with different requirements. We focus on the software characteristics which allow for an on-line reconfiguration of the network and we show that our system can successfully be applied on three deployments with different requirements.

Another critical issue is the cost and difficulty to physically access the deployed nodes for reconfiguration or reprogramming. Many WSN systems implement a limited set of commands that can be interpreted and executed by a sensor node, e.g., Talzi *et al*. (2007). Another approach which overcomes this limitation is based on executing remote procedure calls (RPCs) and modifying variables in RAM of a sensor node. Several recent papers discuss the usage of embedded RPCs (May *et al*. 2005, Cohen *et al*. 2007) in sensor network applications for debugging (Whitehouse *et al*. 2006), testing (Okola and Whitehouse 2010), and configuration (Yuan *et al*. 2008) purposes. In the context of our work, the Marionette system (Whitehouse *et al*. 2006) deserves special attention, since the authors go beyond reconfiguration of the system and state that the RPC-based approach allows the user to choose which functionality of the sensor network application should run on the PC and which on the sensor node. Marionette has shown to work successfully with third-party applications in combination with scripts that introduce new functionality to a sensor network without reprogramming the nodes.

Therefore, to successfully compete with wired monitoring systems, a WSN platform has to provide a high degree of flexibility to be able to adapt to the various application scenarios occurring in civil engineering practice, e.g., a WSN platform should allow to perform a short term vibration monitoring of a pedestrian bridge for assessing the natural frequencies and damping as well as a medium term (several weeks or months) strain monitoring of a railway bridge for assessing the remaining fatigue lifetime. This application profile requires a platform that is easily adaptable to different sensors (e.g., temperature, accelerometers, displacement, strain etc.), data acquisition policies (e.g., periodical, triggered) and in-node data processing algorithms. Furthermore, the platform should also provide a transparent control over the system and remote configuration and reprogramming tools

since many parameter values are not known in advance and have to be reconfigured during runtime.

In this paper we present a wireless monitoring system based on WSN technology, named STONE (STructural health mOnitoring NEtwork). Although traditionally WSNs require application driven hardware and software development, STONE is generic and allows for many application-specific customizations. Moreover, the software design allowing flexible coupling of node software and the backend application with an embedded RPC mechanism similar to Marionette Whitehouse et al. (2006) provides comprehensive remote reconfiguration capabilities of measurement settings and system parameters. While Marionette focuses on the development and debugging phase, STONE exploits similar concepts for system reconfiguration and application partitioning between the sensor network and the backend software. The reconfiguration technique developed in STONE, allows to overcome the limitations of Marionette concerned with control message loss, latency and safe updates of variables and RPC execution.

Due to the danger of significant financial losses and more importantly harm in case of errors, inspecting the condition of civil structures is a highly responsible task and is strongly tied to the expertise of civil engineers and the reputation of the companies. Therefore, it is essential for a monitoring system to insure data integrity, transparent system operation and a high reconfiguration level in order to compete with conventional monitoring systems. For this reason, the STONE design shifts application-specific complexity from resource-poor sensor nodes to the backend application leaving simple but reconfigurable measurement tasks composed of highly optimized and thoroughly tested components on the sensor nodes. This approach makes networks running STONE less complex and more transparent, while preserving the scarce resources of the sensor nodes.

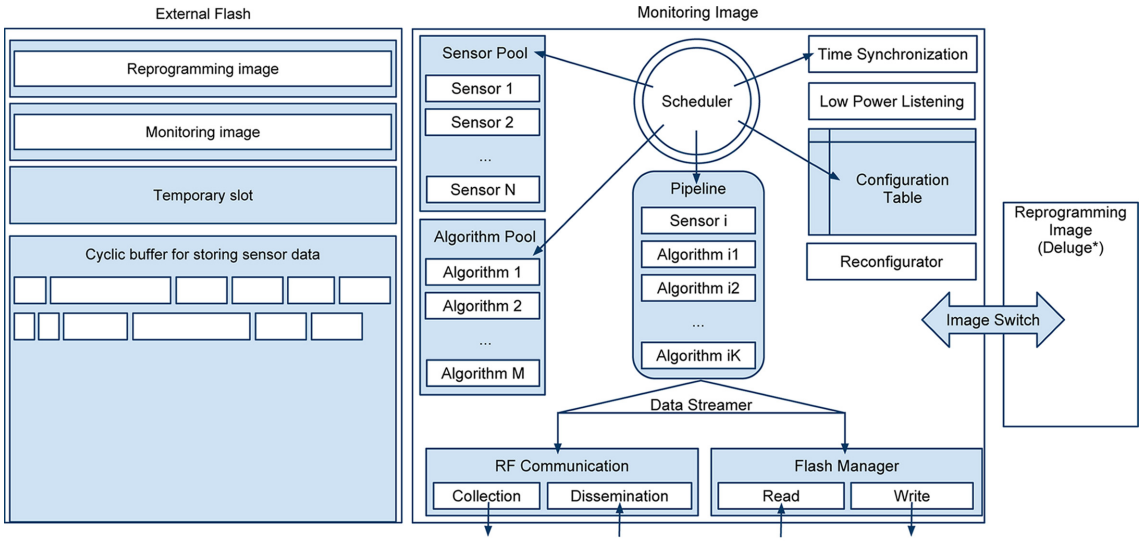
The main contributions of this paper are:

- We present STONE from the perspective of its software design requirements. The chosen design approach provides comprehensive remote capabilities, flexibility from a civil engineering point of view and energy efficiency to the overall application.
- We investigate flexible application partitioning as an essential core service of our platform. This enables easy and direct control of essential algorithms by the civil engineers on the backend. We discuss the advantages and usage of dynamic application partitioning based on three SHM deployments with different requirements.
- We describe and evaluate three deployments of the STONE system: The first deployment is in a two-story building on a shaking table in University of Athens (Greece), which focuses on estimating the damage degree of a civil structure by measuring the magnitude of the destructive forces during catastrophic events. The second application is dedicated to monitoring natural frequencies of cable stays of the Stork Bridge (Switzerland). The last application is in a house in Amphilochia (Greece) that aims to assess structural integrity after catastrophic events like earthquakes.

The rest of this paper is organized as follows: Section 2 describes the STONE architecture and design considerations. In Section 3 we present three applications in the field of structural health monitoring (SHM) where the STONE system is currently being used and show application-specific advantages of STONE. Finally, Section 4 concludes this paper.

2. STONE architecture

We use off-the-shelf components for the sensor node platform and for the sensors coupled by a custom-made component to reduce the cost of the system. The STONE sensor network hardware



(a) Flash organization (b) Monitoring system architecture

Fig. 1 STONE overview

is based on the Tmote Sky nodes which support up to 8 sensor channels. Currently, STONE has been tested with over 20 different sensors frequently used for structural health monitoring and environmental monitoring (Decentlab 2004). The following goals heavily influenced the software design of STONE: a generic architecture, simplicity and powerful reconfiguration.

Due to scarce resources, WSN systems are usually optimized for a certain type of applications. This approach leads to considerable changes to the system software to meet new requirements and to fit new applications. STONE is generic and extensible in the sense that there are a number of sensors, processing components and measurement schedulers with a supporting framework to allow assembling suitable components for a specific application.

Developing WSN systems is a non-trivial task due to the combination of resource scarceness of sensor nodes, message loss and the distributed nature of wireless networks making WSN systems hard to test and debug. Therefore, the software on the nodes of the proposed monitoring system is intentionally kept simple and the application logic is shifted to the backend.

Finally, traditional SHM systems are usually calibrated and configured once installed on the monitoring object, due to configuration dependency on the actual civil structure and limited experience of civil engineers with this technology. Therefore, powerful reconfiguration is essential for WSNs.

2.1 Program images

STONE builds on TinyOS (Levis *et al*. 2005) and includes two program images, the reprogramming image and the monitoring image, as shown in Fig. 1(a). Both images are stored in the external flash. The reprogramming image is a modified version of Deluge (Hui and Culler 2004) and allows reprogramming each sensor node individually. All sensor nodes except for the root node use the same reprogramming image. The monitoring image, however, is specially tailored to each sensor node depending on the set of physically attached sensors. The reprogramming image is used for

updating the monitoring software. The monitoring image can be distributed over the air and resides in the second slot of the external flash. It includes the software to access the sensor data, the necessary processing algorithms to extract relevant information, which is transmitted to the base station, a time synchronization protocol and a scheduler that executes measurement tasks.

The third slot allocated in the external flash is used for forwarding purposes as part of the image transfer protocol. Image switch is a tiny component (around 400 bytes) for replacing the currently running image with any other image available in flash and is included in both reprogramming and monitoring STONE images.

2.2 Monitoring system components

The monitoring image supports two modes: the execution mode and the configuration mode. The execution mode corresponds to the standard system operation for data acquisition and processing. In the configuration mode the system parameters can be safely changed. In order to switch to configuration mode, STONE waits for ongoing acquisition tasks to complete, removes future acquisition tasks from the task queue and stops the scheduler.

The software architecture of the monitoring image is presented in Fig. 1(b). The measurement tasks are executed by a scheduler and can be planed in local or in global time using the time of the root node as a reference. The configuration table contains a description of measurement tasks, sensor settings and processing parameters. The SensorPool provides a scheduler with a unified interface to access the sensor data from an attached sensor. Although STONE supports many different sensors, only those that are physically attached are included in the node's mon-itoring image. Similarly, AlgorithmPool contains the processing algorithm library. This allows constructing a processing sequence (pipeline) that reuses intermediate input/output buffers and processing code for various algorithms, treats every algorithm independently by turning it on/off or exchanging it with any other locally available algorithm. Together with the processing result, the pipeline returns the execution status of the processing in the form of feedbacks. Feedbacks can prevent the pipeline from executing further processing steps. This mechanism allows dismissing useless data in case further processing is not required (e.g., no event has been detected). Generated positive feedbacks are recorded by the pipeline and can be routed to the base station or stored in the external flash. For example, the Scale processing algorithm produces a scaling factor and an offset as feedback information when scaling a time series.

2.2.1 Sensing

STONE organizes data in buffers that can be allocated to use the complete RAM of a Tmote Sky (10 kB). Each measurement task has a buffer associated with it. A sensor component fills the buffer with sensor readings by running a sequence of measurements. STONE distinguishes four types of sensors: analog, digital, interrupts and virtual. A generic analog sensor component can access any analog sensor by reading the ADC values. We have gained practical experience with over ten different analog sensors, e.g., accelerometer, strain, acoustic emission, CO2, wind, light. A digital sensor generally requires an individual implementation of the underlying protocol for accessing sensor data. Tipping bucket is an example of an interrupt sensor. Virtual sensors are introduced to avoid a differentiation between the sensors and the STONE self-monitoring metrics, e.g., route quality. Finally, a test sensor fills a buffer with a test signal for being able to repeat the same input sequence. Conversion of sensor values is done by the backend software. This design decision

minimizes the number of floating point operations on a sensor node, makes node software simple and independent on the kind of sensor actually being read.

2.2.2 Processing

The processing algorithm library is mainly targeted at SHM applications (see Table 1 for some examples). All algorithms work with integer input/output and largely use fix point operations, which is a less memory demanding, much faster and less power consuming operation. The most challenging task here is the performance of the integer FFT. Evidently, an integer FFT implementation may introduce significant errors, e.g., overflow. By scaling the recorded time series with a suitable factor, however, the overflow can be avoided and the approximation error can be maintained within an acceptable range. Fig. 2 compares the frequency spectrum computed with a 16 bit integer approximation FFT with the spectrum obtained by the standard 32 bit floating point FFT. As can be

Table 1 Sample processing algorithms supported by STONE

Component	Input	ROM	RAM	Time	Time	Tasks	Description
Name	#	(bytes)	(bytes)	fix-pt.(ms)	float (ms)	#	
FixPointFFT	1024	2604	80	542.8	5840	3877	Fix-point FFT of time series
	512			252.8	2654	1825	
	256			117.3	1195	861	
	128			53.6	531	409	
LocalExtrema	1024	642	18	82.8		1023	Extracts local extrema
PeakPicking	1024	396	20	128.9		1023	Returns N highest maxima
PeakNormalization	1024	664	14		1813.2	1	Normalized spectrum for peak enhancing
Scale	1024	432	20	50.1		2	Shifts and scales an input
L_1 Norm	1024	252	10	7.1		1	Computes L_1 norm
L_2 Norm	1024	410	12	193.0		1	Computes L_2 norm
SW Trigger	1024	346	8	2.7		1	Returns 1 if threshold is exceeded
Statistics	1024	496	22	12.9		1	AVG, MIN, MAX and SUM

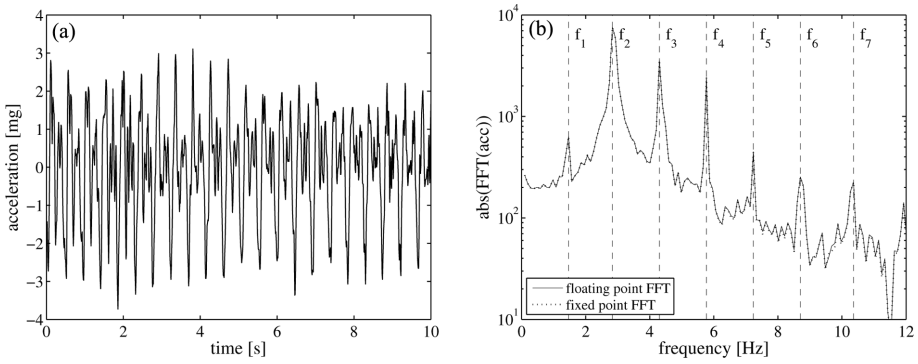


Fig. 2 (a) Ambient vibrations recorded on a stay cable and (b) the spectral amplitude of the integer approximation of FFT compared to the spectral amplitude of the standard floating point FFT

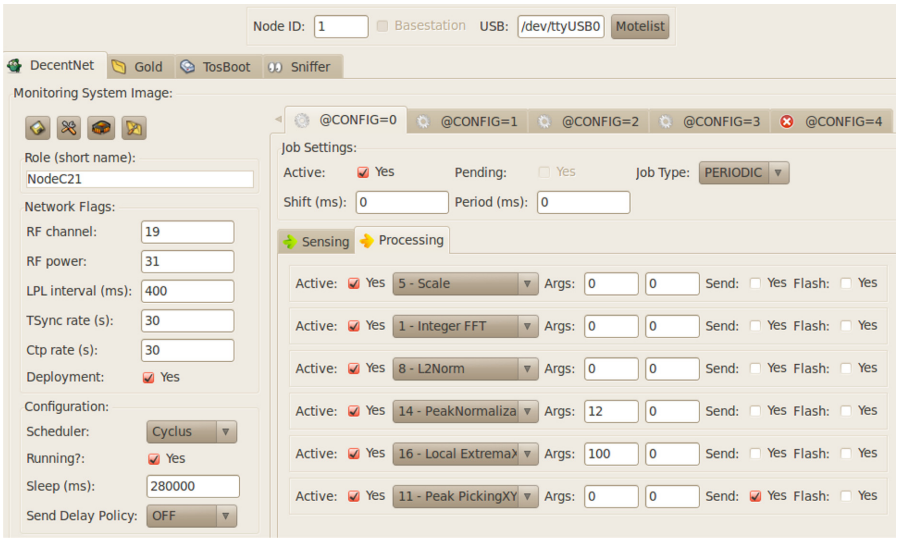


Fig. 3 Software configuration interface of the monitoring image

observed, the result of the integer approximation FFT matches very well with the floating point FFT.

In STONE the data can be preprocessed on a sensor node and further handled by the backend software, e.g., for converting measured data to physical units. STONE provides freedom to civil engineers to decide which part of the processing sequence should reside on the node and which must be relocated to the backend. Finally, power can be saved by processing the raw data on the node with the goal to reduce the amount of data that needs to be transmitted (sending of information instead of raw data), since sending 1 bit costs as much energy as executing about 1000 instructions of a low power microprocessor (Culler and Wei 2004). For example when monitoring vibrations which produces large samples of raw data, this strategy is the most powerful energy saving method and is mandatory if a system lifetime of several months should be achieved (Feltrin *et al*. 2006, Lynch *et al*. 2006).

Fig. 3 illustrates the software configuration interface of the monitoring image. On the left side the user can configure several network parameters (e.g., RF channel, RF power, scheduler). On the right side, the node configuration is depicted with an example of a specific pipeline. This pipeline consists of six algorithms, while only the output of the last process is sent to the base station. This is a screen shot of the deployment on the Stork Bridge (Section 3).

2.2.3 Time synchronization

The tasks are scheduled in global time having the local time of the base station as a reference. The clocks of individual sensor nodes are synchronized by a suitable time synchronization protocol, e.g., FTSP -Flooding Time Synchronization Protocol (Maroti *et al*. 2004), which provides time synchronization with 100 μs per hop accuracy. Accurate time synchronization allows simultaneous fetching of sensor data from spatially distributed sensor nodes and, therefore, alignment and comparison of data acquired at different locations. The time-stamped processed data are communicated to the base station over the multi-hop data collection protocol. While simultaneous measurements are important for data quality and processing algorithms spanning information from several nodes,

simultaneous transmission of data packets from multiple nodes leads to numerous collisions and packet losses. Therefore, the data communication protocol supports randomization of transmission start times of individual sensor nodes to minimize the probability of simultaneous transmission.

2.2.4 Schedulers

A measurement task contains configuration data, e.g., task on/off times, sensor ready time, sampling rate, allocated buffer, processing steps etc. This data defines how a scheduler should handle the task. The scheduler passes the configuration to a sensor component and then configures the pipeline by loading the processing steps. If the pipeline is busy handing another task, the processing is postponed. STONE includes three schedulers that follow the same interface. The Cyclus scheduler periodically executes all measurement tasks in a row. Cyclus has a small code size and does not rely on time synchronization. The Synchros scheduler executes a task to a global plan stored in the node's configuration table. The availability of a global time enables synchronized and parallel task execution by multiple nodes. Finally, the Woros scheduler continuously samples a sensor with high frequency and corresponds to the way conventional measurement systems work. All schedulers provide a state flag and the two primitives startScheduler() and stopScheduler() to control the scheduler execution.

2.3 Runtime reconfiguration and control

STONE supports flexible reconfiguration using basic mechanisms of reading and writing RAM content and executing functions with RPCs without the need to develop dedicated message structures and the associated parsing and processing overhead. The STONE configurator leverages the knowledge that the compiled executable for Tmote Sky follows the ELF format. Therefore, by analysing the symbol table of a node image it is possible to extract the locations in RAM and the sizes of the variables that have at least component-scope as well as function addresses in ROM. The following four commands are supported by the STONE configurator: READ and WRITE commands read and write a variable given its size and location in memory. RPC calls a function given its address and parameters. Finally, the SWITCH command forces the node to switch to the reprogramming image. The configurator is lightweight and uses only 2846 bytes in ROM and 140 bytes in RAM.

The configurator uses the standard TinyOS data dissemination protocol Drip to pass commands to the sensor nodes. End-to-end acknowledgements (ACKs) are used to ensure reception of every command. Every command packet includes a unique ID, which is sent back as part of the ACK message. The base station applies retransmission and timeout mechanisms to ensure that the command is received by the node. The node caches the result in order to allow resending it without re-executing the command if the original result message is lost. Therefore, STONE implements at most once semantics, which should be considered when coding the backend part of the application.

Updating or reading small variables (less than 10 words) that fit in one command/response message does not require to keep track of the update state. When dealing with big variables, the STONE configurator sequentially requests/sends small parts of the variable and keeps track of the separate pieces of it. This way, the state is always stored at the backend. The backend recognizes missing packets and requests for a repeated transmission. To keep the sensor node software simple, STONE sequentially performs READs and WRITEs to ensure that nothing gets lost. This allows using differential updates for big variables.

2.4 Application partitioning

Many WSN developers and researchers work on making wireless sensor nodes smart and networks self-organized. However, in the SHM field, WSN technology is primarily seen as a low-cost measurement instrument that enables fast and cable-free deployment. As for any measurement device, the data acquisition is expected to be foremost reliable. Moreover, the civil engineers are required to make decisions and take over the responsibility based on their knowledge and experience and, therefore, require full transparency and flexibility. This is the first argument for residing some part of the high-level application logic at the backend in a form of scripts or primitives that can be executed by domain experts. The second reason for application partitioning is energy efficiency of the overall application. STONE includes a simple script language with an additional support for 5 blocking functions: data( ), read( ), write( ), rpc( ) and wait( ) to interact with the application running on the sensor nodes. In the following subsections we give three examples of the application partitioning.

2.4.1 Automatic flash management

STONE uses the TinyOS implementation for flash management. The pipeline can directly store outputs of the processing steps into external flash. Additionally, STONE includes three "unconnected" functions for seeking, reading and erasing flash. On the one hand, these functions provide the user with freedom to decide when to read data from the flash and when the transmission should be repeated. On the other hand, a script running on the backend can perform necessary checks, read and empty flash on every sensor node separately. Therefore, this application logic is shifted to the base station.

It heavily depends on the application whether the flash access logic should be part of the sensor node application or reside on the backend. For example, in the deployment in Amphilochia, Greece, STONE is used for structural integrity assessment after an earthquake catastrophe, Section 3.3. Since earthquakes are quite infrequent events, we decided to shift the data extraction logic to the backend software.

2.4.2 Identifying outliers

STONE recognizes events at two levels: by interrupt sensors and event detection processing components and by scripts running at the backend. In this sense, event recognition is partitioned between the sensor network and the backend. Since sensor nodes have limited memory resources (RAM:10 KB, External flash: 1 MB for the Tmote Sky), they are only suitable for capturing short events, e.g., an earthquake. However, long-term changes resulting from material decay or seasonality can be detected only at the backend. As an example, identifying outliers in sensor data does not generate any input communication, does not reduce the output data (since frequent outliers might indicate sensor malfunction) but requires some code to be implemented and, thus, resides on the backend. We believe that STONE achieves a good application partitioning for SHM applications. STONE implements measurement logic as part of the monitoring image and leaves stubs for performing infrequent operations from the backend. As has been discussed above, this approach reduces the number of message exchanges and saves energy of sensor nodes.

2.5 Alarming service

Managing several WSN deployments does not allow for manual soundness checks and immediate

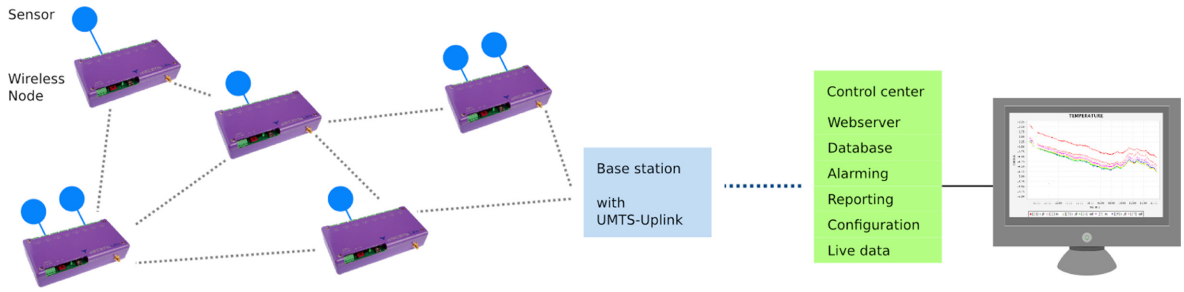


Fig. 4 Overview of the monitoring system architecture

malfunction detections. For this reason, STONE includes an alarming service running at the base station and implemented on top of the STONE configurator. The alarming service allows experts defining expressions that describe events of interest related to a WSN and list actions to handle these events. Moreover, the alarming service can control sensor nodes with commands by reading and updating variables or executing functions with RPCs.

2.5.1 Passive, active and time-based alarms

STONE differentiates three kinds of alarms: passive, active and time-based depending on their evaluation specifics. Passive alarms passively listen to the incoming messages. On reception of a data stream from the deployed WSN, the alarming service evaluates all passive alarms based on the received data items. If the result of the evaluation is true, the alarming service executes the corresponding alarm handlers. Active alarms are listening to the responses of command message execution which correspond to a variable read operation. Finally, time-based alarms are useful for detecting network and node malfunctions like node failures, bad communication links, unreachable nodes or parts of the network. Time-based alarms keep track of the time since the last event, e.g., last message reception form a node. All time-based alarms are periodically evaluated in contrast to active and passive alarm that wait for an incoming message.

2.5.2 Alarm handlers

We currently support the following actions handlers: writing into a log file, displaying a dialog on the screen, sending an email and a text message, sending a predefined RPC command to a sensor node, and updating a variable on a sensor node. The last two possibilities are also integral to allow the distribution of the application between sensor nodes and control center and allow easy automation of tasks.

3. SHM applications and experiences

In this section, we share our experiences with three deployments of the STONE network: a two-story building placed on a shaking table at the University of Athens, Greece, a cable stayed bridge in Switzerland and monitoring of a building in Greece. The first deployment requires a continuous vibration monitoring for assessing the performance of the structure when subjected to ground motion. The second deployment aims at long term monitoring of the natural frequencies of the bridge and, thus, periodic collection of data is required. The last deployment focuses also on a long

term monitoring of a house for quick structural integrity assessment after catastrophic events using an event detection mechanism. Although these applications have different requirements and goals, STONE ensures the successful monitoring of these structures and at the same time validates its contributions and advantages.

The architecture of the monitoring system is displayed in Fig. 4. The data acquired by the wireless sensor network is collected by the base station and afterwards transmitted to the control center located at Empa via an UMTS link or an Ethernet link.

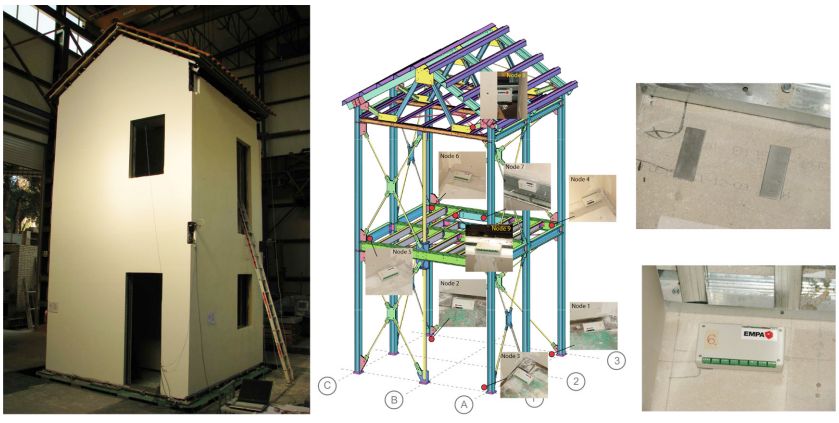
3.1 Shaking table test, greece

This deployment represents a typical short term monitoring application that requires rapid mounting and the acquisition of raw vibration data. In these applications scenarios energy efficiency is of secondary importance.

The installation of the STONE network took place in Athens, Greece in March 2010. The system was deployed in a two-story building made of lightweight gypsum dry-wall systems (height: 7.5 m, base: 3.3 × 3.3 m) placed on a shaking table at the University of Athens. During two days numerous earthquake simulations along all three axes and consecutive sweeps have been performed. Sweeps are vibration tests with a constant amplitude and growing frequency which are used to identify natural frequency of the building or its possible shift after an earthquake simulation.

Figs. 5(a) and (b) depict the test set-up with the location of the sensor nodes inside the mock-up building. The network included 9 battery-powered sensor nodes equipped with a 2-axis acceleration sensor (LIS2L06) mounted with magnetic feet at different locations of the building. More specifically, on the ground floor, sensor nodes 1, 2, and 3 were mounted directly on the steel table of the shaking table. Sensor nodes 4, 5, and 6 (located on the first floor), were mounted on two metallic plates, which were fixed on the gypsum panel with double faced adhesive tapes, Fig. 5(c). Sensor nodes 7, 8 and 9 were mounted directly on the steel components. Node 7 recorded vertical accelerations on one axis. All other nodes recorded horizontal acceleration in two perpendicular directions. The base station was located 7 m away from the house.

To avoid data loss, the data of the flash memory can also be read out using usb. This option



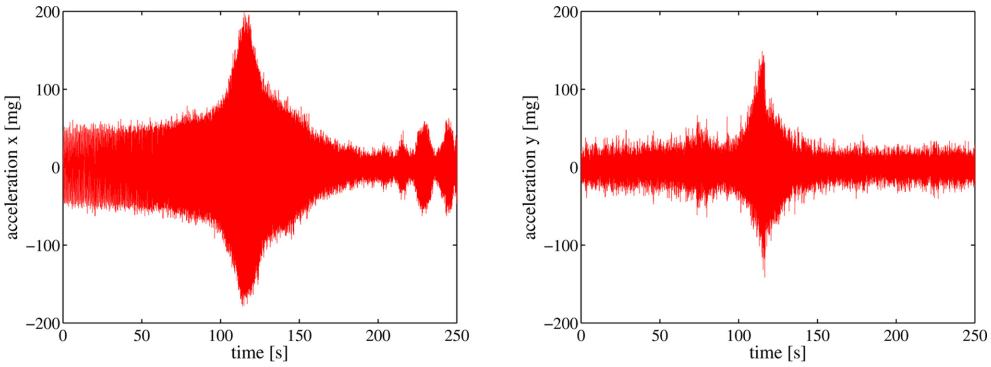
(a) (b) (c)

Fig. 5 (a) Mock-up building, (b) location of nodes and (c) sensor node with magnetic footing mounted on the metallic plates

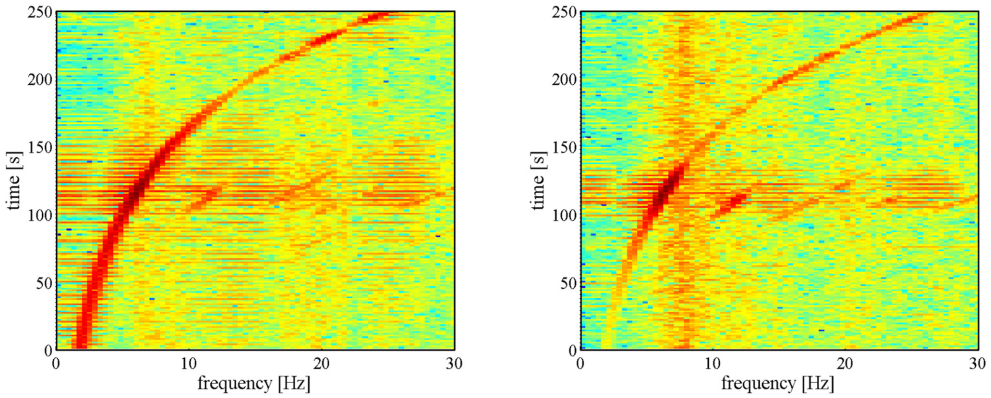
requires a periodical manual harvesting of the data and therefore a permanent accessibility of the sensor nodes. On the startScheduler() RPC command of the WorosP, the sensor nodes started sampling 2 channels of acceleration data with 200 Hz each. The stopScheduler() command stopped the measurements. The commands were distributed with broadcast messages since all nodes except for the root were running the same monitoring image. The recorded data were stored to the local flash memory. The reading was performed with an RPC command that addresses just one node to avoid data loss due to packet collision. The reading of 4 minutes of recorded raw data took approximately 30 minutes, clearly demonstrating the severe limitations of wireless transmission of raw vibration data.

Fig. 6(a) displays the recorded accelerations in X and Y directions of the sweep test with an amplitude of 0.02 g, where g=9.81 ms^{-2}. The spectrograms in Fig. 6(b) show the frequency shift with time. The time histories demonstrate that the chosen resolution was able to represent time histories with small scale vibration amplitude.

Fig. 7 displays the data loss due to the wireless communication in terms of data packets. A data packet contained 8 measurement samples. The total number of data packets represented the recording of 5 earthquake time series. The mean data loss rate was 0.24%. The greatest data loss occurred on node 4 and was 0.63%. Nodes 1, 6 and 8 had no data loss. There is no pattern that correlates data loss to the node position. During data downloading several persons were always



(a) Time histories of the accelerations in X and Y axis



(b) Spectrogram of the time histories of the accelerations in X and Y axis

Fig. 6 Sweep test results

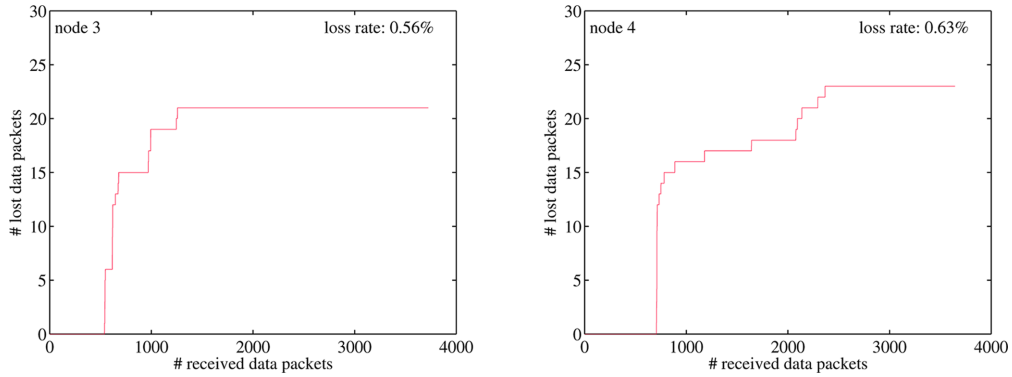


Fig. 7 Data loss of the wireless communications in nodes 3 and 4. The download of the stored data was performed node by node

between the nodes and the base station, therefore it is likely that the data loss was caused by the higher absorption of radio waves because of these persons. Nevertheless, the loss rate is quite small.

Concluding, the shaking table tests demonstrated that STONE is able to record two axial accelerations with a sample rate of 200 Hz. Its wireless data acquisition system has a resolution able to capture the motion due to small earthquakes with a sufficient granularity, but it can also be adapted to capture the motion due to strong earthquake, with a mean data loss that is smaller than 0.3%.

3.2 Deployment at stork bridge, switzerland

This deployment is radically different from the previous one since it is designed for long term monitoring. Long battery lifetime, long term system stability and in-node data processing were the challenging aspects. The Stork Bridge[1] is a two span cable stayed road bridge with a total length of 124 m and 24 cables, two of which are made of carbon fiber reinforced plastic. The STONE network deployed on the bridge comprises 6 battery-powered nodes, labeled C1 to C6, mounted on 6 cable stays, one forwarder node C7 and the root node C0 located under the bridge deck at the abutment, Fig. 8. The root node is connected to the base station computer placed inside an abutment and powered by the mains power supply.

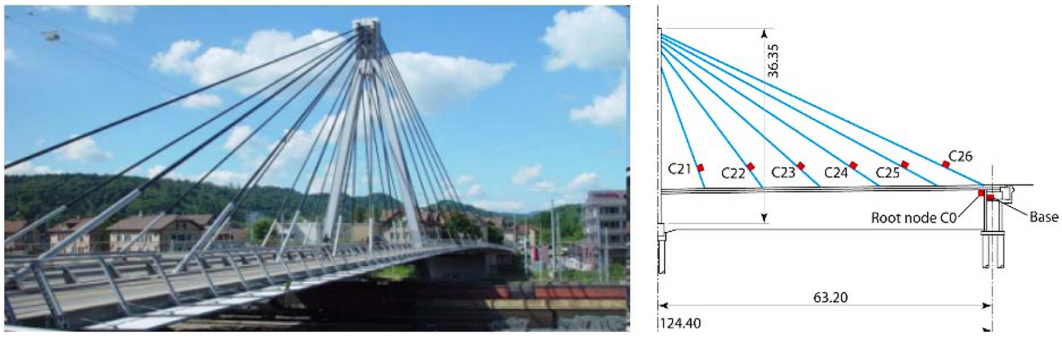


Fig. 8 Stork Bridge deployment setup

[1]Monitoring of the Stork Bridge started in 2006, (Feltrin *et al.* 2009). STONE substituted the old system in late 2009.

3.2.1 Natural frequency estimation

Cable tension is usually estimated by correlating the measured natural frequencies, which are extracted from ambient vibration recordings, with natural frequencies predicted with a cable model (Casas 1994, Feltrin *et al.* 2006). Since the natural frequencies are the only information needed to estimate the cable tension, they are extracted by processing the raw data in the nodes and transmitted to the network sink, while the raw data can be discarded. From the initial amount of data, e.g., thousands of samples, the data to be communicated is reduced to several natural frequencies. This approach results in a drastic reduction at the communication and the power consumption.

In our experiment nodes C1-C6 measure acceleration (LIS2L06 accelerometer, ST Microelectronics), temperature/humidity (Sensirion SHT11), voltage and route quality every 5 minutes. 1024 acceleration samples are acquired with 50 Hz and preprocessed by a sequence (pipeline) of six processing steps in order to identify the 8 strongest natural frequencies of the corresponding cables to be transmitted to the base station. The following processing sequence selected from the AlgorithmPool in the monitoring system component (Table 1 in Section 2.2.2), reduces data by 98%. This pipeline is also depicted in Fig. 3.

1. Scale the signal over the $(-2^{15}, 2^{15}]$ interval to minimize precision loss
2. Compute the integer FFT (FixPointFFT) to the scaled signal.
3. Compute the amplitude of the frequency spectrum using the L_2norm
4. Compute a normalized spectrum in order to enhance the peaks (PeakNormalization).
5. Extract the extrema of the spectrum that are greater than a specified threshold (LocalExtrema)
6. Choose the 8 highest peaks (PeakPicking) and their positions in the frequency spectrum

Fig. 9 shows the time evolution of the captured natural frequencies of cables C2 and C5. 6 natural frequencies could be regularly monitored for cable C2 and 3 for cable C5. Natural frequencies with a magnitude greater than ca. 12 *Hz* are more difficult to detect since the associated vibration modes are scarcely excited. The scattering of the natural frequencies estimations of cable C5 are greater than those of cable C2 due to the higher damping of the CFRP cable and the shortness of cable that limits the magnitude of the ambient vibrations, c.f. Fig. 8.

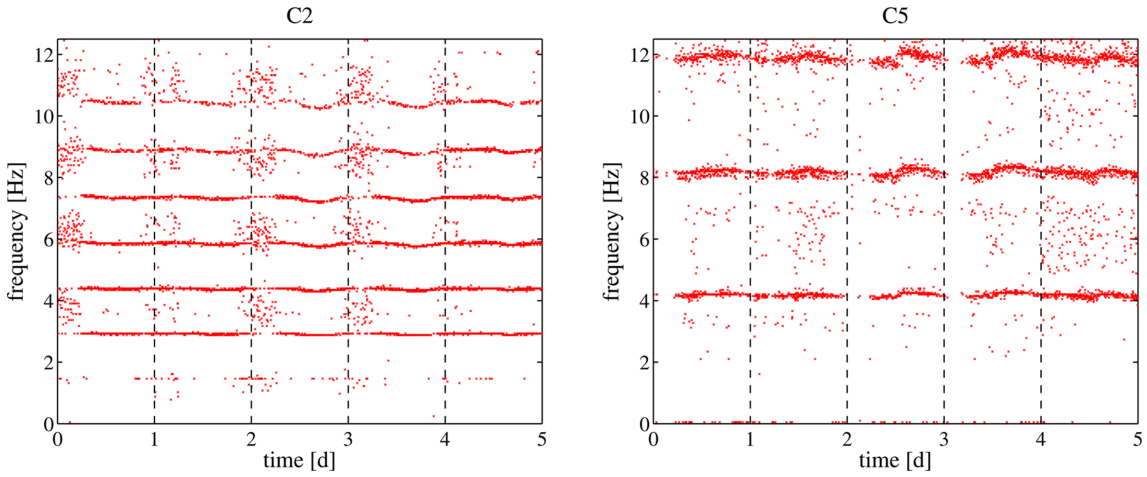


Fig. 9 Natural frequencies of cables C2 and C5

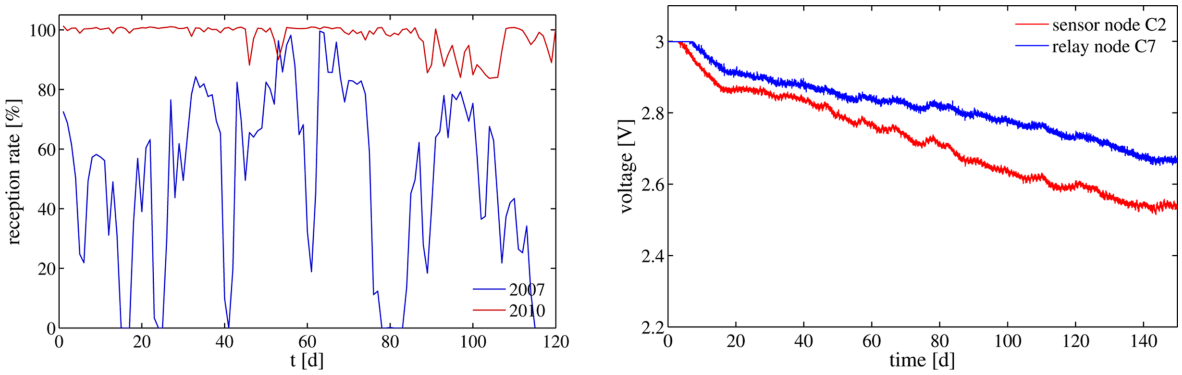


Fig. 10 (a) Delivery ratio of two test periods and (b) supply voltage evolution for nodes C2 and C7

3.2.2 Stability and reliability

One of the most challenging issues in a WSN monitoring system is to achieve system stability. STONE integrates data acquisition, data processing, time synchronization, low duty cycle, task scheduling etc. into one system which makes it efficient and flexible but also very sensitive. When operating the network with TinyOS 1.x/Boomerang the routing tree was changing continuously (e.g., sensor nodes disappeared and spontaneously reappeared after some time without any evident reason). In this highly dynamic situation quite often nodes failed to choose a parent node correctly and lost their link to the network. The system stability and reliability improved signifficantly after porting the software to TinyOS 2.x. This progress is shown in Fig. 10(a) that displays the hourly delivery ratio during 120 days of two test periods. The delivery ratio is defined as the percentage of natural frequencies received by the remote control unit with respect to the theoretical maximum.

During the period April-May 2007, the delivery ratio was rarely higher than 90% and several total break downs of the whole network. Total failures of sensor nodes to deliver data occurred very often. On the other hand, during the period January-February 2010, this pattern was not observed. The delivery ratio was 100% for most of the time and was only briefly interrupted with partial break downs that did not go below 80%. The mean delivery ratio of the two test periods was 53% and 99.5%, respectively.

3.2.3 Energy consumption

The average power consumption of the WSN node is determined by the time needed for sensing and data processing, the amount of data to be transmitted, the duty cycle period, the power management of the hardware and the network topology, since it determines the number of hops for reaching the data sink. Sensors and signal conditioning boards were switched-off after completion of the data acquisition. Furthermore, since the nodes were not equipped with a voltage regulator, the radio was switched-on during data acquisition in order to avoid signal corruption by the duty cycle.

Fig. 10(b) shows the battery voltage drop of the sensors node C2 and C7. Due to ambient temperature variations, the observed voltage curves are not decreasing monotonically but are oscillating signifficantly. The voltage drop of C2 was approximately 0.45 V in 150 days. Since a sensor node can be operated correctly provided the supply voltage is higher than 2.4 V, a lifetime estimation based on the observed voltage drop (and assuming an exponential decay) predicts a battery lifetime of approximately 210 days. The relay node C7, which was not equipped with

sensors, had half the voltage drop of sensor node C2 and thus twice the life-time of node C2. Therefore, roughly half of the energy of a node is consumed by data acquisition, data processing and the extra radio-on time during data acquisition.

3.3 Demo house deployment

In May 2010, STONE was deployed in a house in Amphilochia, Greece, in order to provide data for a quick structural integrity assessment after catastrophic events like earthquakes. The challenge in this deployment was to have an event based monitoring system and complete access to the data via a web interface. In this deployment 4 of the nodes that were used at the shaking table tests (Section 3.1) were integrated in the house in Amphilochia and 2 more nodes equipped with temperature and humidity sensors. One of these nodes was also equipped with a smoke sensor. The nodes equipped with temperature and humidity sensors had no fixed location and were designed to be easily movable from one location to another.

3.3.1 Storage and visualization

The data acquired by the wireless sensor network is collected by the base station and afterwards

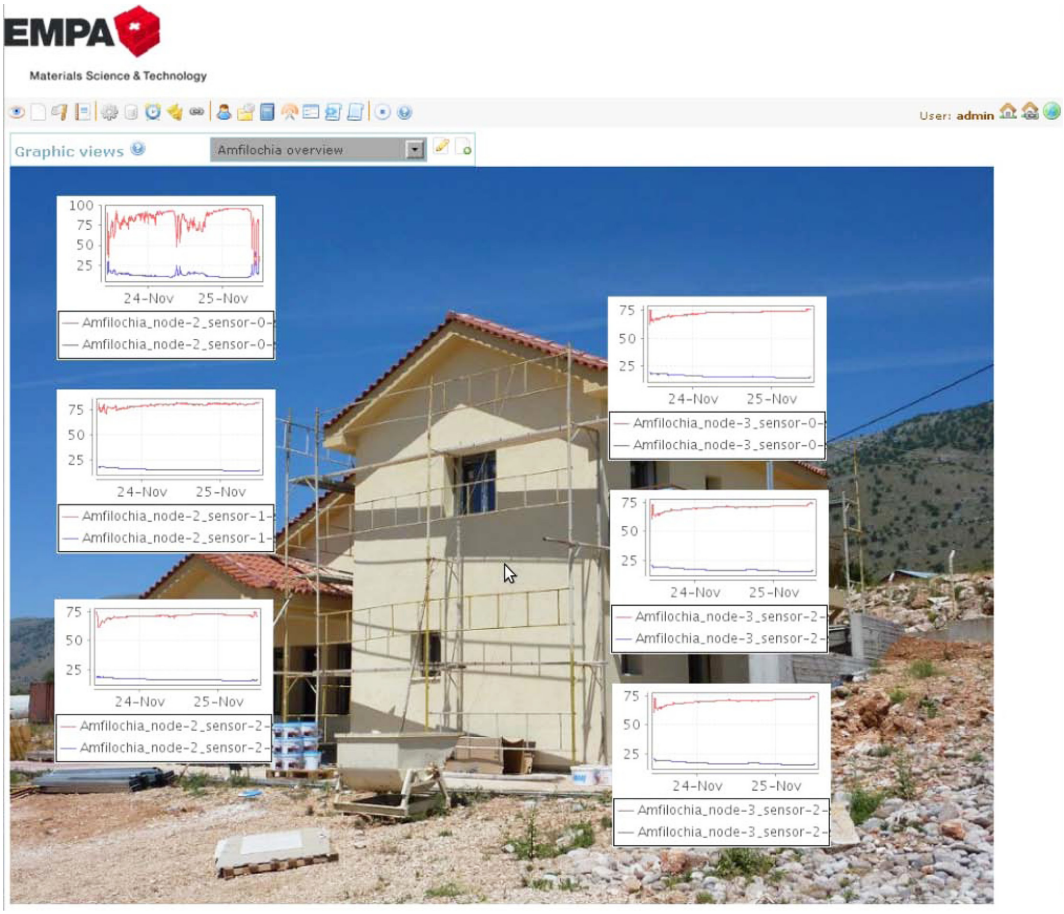


Fig. 11 Visualization of the recorded data via web interface

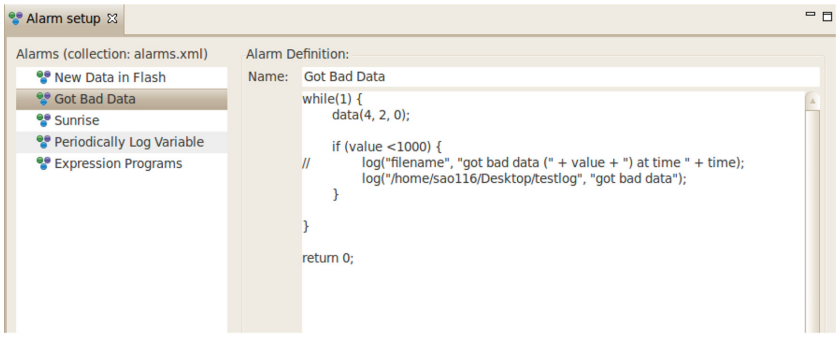


Fig. 12 Alarm setup interface. The left side shows the different alarm setups, while the right side shows the corresponding expressions in xml code

transmitted to the control center located at Empa via the internet. The control center stores the data in its database and provides tools for visualizing the data via a web-interface (Fig. 11), for configuring the wireless sensor network and the base station remotely, for reporting the monitoring process, and for defining and issuing alerts. One example of the alarming system is the creation of a log file whenever unfruitful data are measured, Fig. 12.

3.3.2 Event based monitoring

The accelerations were permanently acquired with a sampling rate of 100 Hz (less than the sampling rate at the shaking table test) and organized in data blocks with 1024 samples. After acquisition, a data block was analyzed in the node to identify the occurrence of an event. An event occurs if the acceleration exceeds the threshold value of 25 mg. If an event occurred then the whole data block was stored in the flash memory of the sensor node. If no event occurred the data block was discarded. The flash memory was organized as a ring buffer with a size of 786 kB. The data was stored in packets. Each packet consisted of the acceleration data, 16 data samples, and meta data that described the data packet (e.g., node number, sequence number, data type, etc.). Because of the overhead, the maximum acceleration data size that could be recorded in the ring buffer was 524 kB. Fig. 13(a) depicts the accelerations captured by STONE a specific time period. Depending on the application, the user can easily reconfigure the network and change the event detection

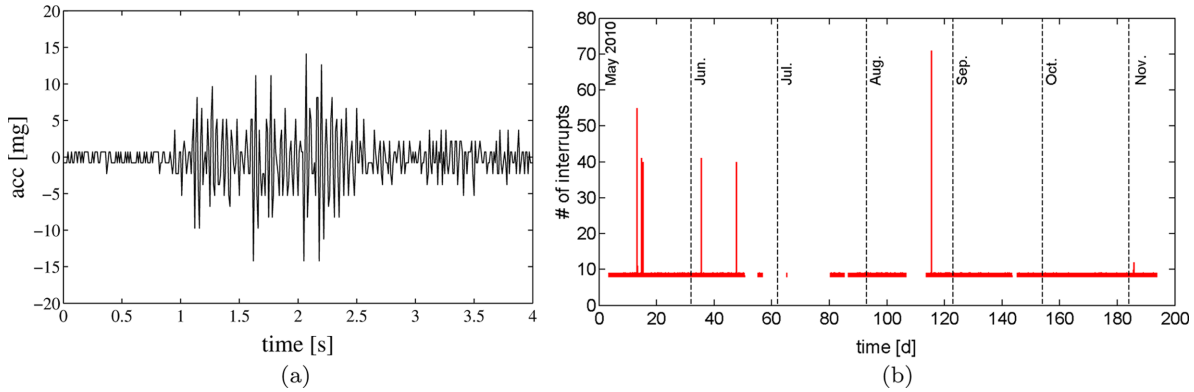


Fig. 13 (1) Time history of accelerations recorded at node 9 and (b) time history of smoke sensors output

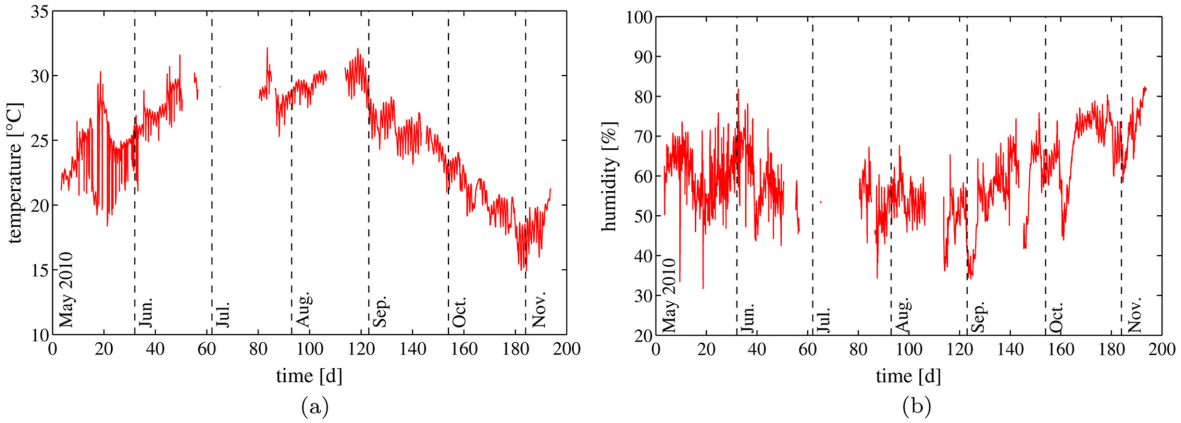


Fig. 14 (a) Time history of temperature and (b) time history of humidity

threshold in order to capture the vibrations due to a small earthquake or a big earthquake.

The permanent acquisition of accelerations requires a significant amount of energy. Therefore, the sensor nodes equipped with an accelerometer were connected to the main power supply. In addition, for bridging a break down of the main supply, the sensor nodes were also equipped with batteries.

The smoke sensor provided status data with a time interval of 5 minutes. The data consisted of the number of interrupts that occurred within this time interval. If any smoke was detected, the interrupts had a period of 35 seconds. That is, within a time interval of 5 minutes, the number of interrupts was 8 or 9. If smoke was detected, the interrupt period changed to 1 second for the time period with smoke detection. The output of the smoke sensor was then greater than 9 and provided information about the time period with smoke detection. Fig. 13(b) depicts the output of the smoke sensor, from May to November 2010. STONE has successfully recorded and detected 6 smoke events. The time histories exhibit several gaps which were due to problems related to the base station like failures of the Internet connection, accidental shut downs of the mains supply etc. The wireless network operated reliably during all the time.

The temperature and humidity was acquired periodically with a time interval of 1 minute. After 6 measurements, that is 6 minutes, the average of the 6 measurements was computed and the average value was sent to the base station. Figs. 14(a) and (b) display the temperature and humidity time histories recorded with two sensors from May 11, 2010 to November 10, 2010. Notice that the gaps in the time histories coincide in time with those of the time histories of the smoke displayed in Fig. 13(b).

4. Conclusions

Structural health monitoring is both a challenging and a promising application area for wireless sensor networks. Resource constraints and communication limitations that usually result in only moderately flexible and robust systems are in strong contrast to the demands of domain experts. In this paper, we presented STONE, a system that has been successfully validated in several deployments with different requirements: collection of raw data for continuous vibration monitoring, periodic collection of processed data for long term monitoring, and collection of data from heterogeneous

sensors for event based monitoring. Our experiences showed that the partitioning of the application between the sensor nodes and the base station is a promising approach to meet the demands of the domain experts while solving the resource constraints of the devices. Flexible schedulers in combination with a lightweight but powerful configuration and management component based on simple operations and RPCs provide an adaptable framework where generic data acquisition methods and heavily optimized processing algorithms can be configured and composed at runtime by the domain experts. The system fully supports the different phases inherent to structural health monitoring deployments including calibration, testing and operation. All deployments showed that STONE can be used for different applications and also for long term monitoring with very little maintenance.

References

Casas, J. (1994), "A combined method for measuring cable forces: the cable-stayed Alamillo bridge, Spain", *Struct. Eng*., **4**(4) , 235-240.

Casciati, F. and Chen, Z. (2011), "Design constraints for a wireless network architecture", *Proceedings of the International Symposium on Innovation and Sustainability of Structures in Civil Engineering*, October, Xiamen, China.

Cohen, M., Ponte, T., Rossetto, S. and de La Rocque Rodriguez, N. (2007), "Using coroutines for RPC in sensor networks", *Proceedings of the Paral. and Dist. Proc. Symp*.

Culler, D. and Wei, H. (2004), *Wireless Sensor Networks*, Communications of the ACM , **47**(6), 30-33.

Decentlab (2004), *Decentlab Gmbh: decentralized monitoring solutions*, URL www.decentlab.com

Feltrin, G., Meyer, J. and Bischoff, R. (2006), "A wireless sensor network for force monitoring of cable stays", *Proceedings of the 3nd International Conference on Bridge Maintenance*, *Safety and Management*, IABMAS.

Feltrin, G., Meyer, J., Bischoff, R. and Motavalli, M. (2009), "Long term monitoring of cable stays with a wireless sensor network", *Struct. Infrustruct. E*., **6**(5), 535-548..

Hui, J. and Culler, D. (2004). "The dynamic behavior of a data dissemination protocol for network programming at scale", *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, *SenSys*.

Jang, S., Jo, H., Cho, S., Mechitov, K., Rice, J.A., Sim, S.H., Jung, H.J., Yun, C.B., Spencer Jr. B.F., and Agha, G. (2010), "Structural health monitoring of a cable-stayed bridge using smart sensor technology: deployment and evaluation", *Smart Struct. Syst*., **6**(5-6), 439-459.

Jang, S., Spencer Jr., B.F., Rice, J.A. and Wang, Z. (2011), "Structural monitoring of a historic truss bridge using a wireless sensor network", *Adv. Struct. Eng*., **14**(1), 93-101.

Kim, S., Pakzad, S., Culler, D., Demmel, D., Fenves, G., Glaser, S. and Turon, M. (2007), "Health monitoring of civil infrastructures using wireless sensor networks", *Proceedings of the 6th Int. Conf. on Information Processing in Sensor Networks*, IPSN..

Kurata, M., Lynch, J.P., van der Linden, G., Hipley, P. and Sheng, L.H. (2011), "Long-term wireless monitoring system for the monitoring of long-span bridges", *Proceedings of the International Symposium on Innovation and Sustainability of Structures in Civil Engineering*, October, Xiamen, China.

Lei, Y., Liu, L., Jiang, Y., Tang, Y. and Luo, Y. (2011), "A smart wireless sensor network for structural damage detection", *Proceedings of the International Symposium on Innovation and Sustainability of Structures in Civil Engineering*, October, Xiamen, China.

Levis, P., Madden, S., Polastre, J., Szewczyk, R., Whitehouse, K., Woo, A., Gay, D., Hill, J., Welsh, M., Brewer, E. and Culler, D. (2005), *Tinyos: an operating system for sensor networks*, In Ambient Intelligence.

Lynch, J., Wang, Y., Loh, K., Yi, J. and Yun, C. (2006), "Performance monitoring of the geumdang bridge using a dense network of high-resolution wireless sensors", *Smart Mater. Struct*., **15**(6), 1561-1575.

Maróti, M., Kusy, B., Simon, G. and Lédeczi, A. (2004), "The flooding time synchronization protocol", *Proceedings of the Int. Conf. on Embedded Netw. Sens. Sys*.

May, T.D., Dunning, S.H. and Hallstrom, J.O. (2005), "An rpc design for wireless sensor networks", *Proceedings*

of the Int. Conf. on Mobile Adhoc and Sensor Systems.
Nagayama, T. and Spencer Jr., B.F. (2007), *Structural health monitoring using smart sensors*, Structural Engineering Laboratory Report Series 001, Newmark. URL http://hdl.handle.net/2142/3521.
Okola, M. and Whitehouse, K. (2010), "Unit testing for wireless sensor networks", *Proceedings of the Workshop on Software Engineering for Sensor Network Applications*.
Talzi, I., Hasler, A., Gruber, S. and Tschudin, C. (2007), "Permasense: Investigating permafrost with a wsn in the swiss alps", *Proceedings of the 4th Workshop on Embedded Networked Sensors*.
Whitehouse, K., Tolle, G., Taneja, J., Sharp, C., Kim, S., Jeong, J., Hui, J., Dutta, P. and Culler, D. (2006), "Marionette: Using rpc for interactive development and debugging of wireless embedded networks", *Proceedings of the 5th Int. Conf. on Information Processing in Sensor Networks*.
Yuan, F., Song, W.Z., Peterson, N., Peng, Y., Wang, L., Shirazi, B. and LaHusen, R. (2008), "A lightweight sensor network management system design", *Proceedings of the 6th Int. Conf. on Pervasive Computing and Communications*.
This means the chief editor of this paper.

FC