Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Eidgenössisches Departement für Umwelt, Verkehr, Energie und Kommunikation UVEK
Département fédéral de l'environnement, des transports, de l'énergie et de la communication DETEC
Dipartimento federale dell'ambiente, dei trasporti, dell'energia e delle comunicazioni DATEC

Bundesamt für Strassen
Office fédéral des routes
Ufficio federale delle Strade

Low Power Wireless Sensor Network for Monitoring Civil Infrastructure

Drahtloses Sensornetzwerk zur Infrastrukturüberwachung

Réseau de capteurs sans fil pour l'observation de grands ouvrages

Empa, Swiss Federal Laboratories for Materials Testing and Research
Structural Engineering Research Laboratory
Aleksandar Novakovic
Jonas Meyer
Reinhard Bischoff
Dr. Glauco Feltrin
Prof. Dr. Masoud Motavalli

CSEM, Swiss Center for Electronics and Microtechnology
Dr. Amre El-Hoiydi
Dr. Antonio Restrepo
Prof. Jean-Dominique Decotignie

Forschungsauftrag ASTRA2005/008 auf Antrag des Bundesamtes für Strassen (ASTRA)

January 2009 **1232**

Preface

Structural monitoring is a term that increasingly occurred in the last decade within the civil engineering community and which refers to a great variety of measurements systems implemented on full-scale civil structures with the purpose to inform infrastructure operators about the actual state of their structures. The current concern in the western countries is driven by the increasing number of structures that infrastructure operators have to manage and maintain, the increasing number of structures that are approaching their design lifespan, and which require a decision about their future use, and, last but not least, by the rapid progress in sensor, data acquisition, computing and communication technologies that permit to deploy monitoring systems at reasonable costs. These factors are generating a small and innovative market for structural monitoring technology and services that is steadily increasing year by year.

Monitoring with a wireless sensor network is a young and emerging technology that has the potential to mix up and boost the structural monitoring market. The advantages of wireless sensor networks over conventional wired technology are fast deployment, great flexibility, easy scalability and self-organisation. These advantages, however, are obtained by introducing a major handicap concerning power management, because wireless sensor networks have to operate from batteries for several months up to a year or even more. Minimization of power consumption is therefore a key issue. Since data communication dissipates much more energy than data processing, it is vital to process the acquired raw data in each sensor node with the aim to reduce significantly the data sent to the base station. This data reduction step is particularly important when data is acquired with high sampling rates, e.g. when monitoring dynamic processes.

This project investigates the feasibility of structural monitoring with wireless sensor networks. It focuses on monitoring tasks that have a life-time of several months to years and where high data rates have to be acquired and processed. Despite the obvious practical relevance, however, this ambitious goal has been hardly explored so far. This project provides therefore important new results which will influence the further development of structural monitoring with wireless sensor networks.

These achievements were made possible because of the fruitful collaboration between CSEM and Empa, which shared their expertise on wireless sensor networks and structural monitoring. The contribution of CSEM was the low-power wireless communication solution for sensor networks (WiseNET), which was adapted and optimized for civil infrastructure monitoring applications. The contribution of Empa's Structural Engineering Research Laboratory was the development of low-power sensing modules, in-mote implementation of data reduction algorithms, network management and data collection tools of the remote operator console, and carrying out the field deployment. This collaboration between CSEM and Empa was made possible by the generous financial support of the Federal Roads Office, which we acknowledge.

Prof. Dr. Jean-Dominique Decotignie
CSEM, Swiss Center for Electronics and Microtechnology

Prof. Dr. Masoud Motavalli
Empa, Swiss Federal Laboratories for Materials Testing and Research

Duebendorf, January 2009

Summary

This project investigates the feasibility of a structural monitoring system based on low power wireless sensor network technology. The targeted maintenance free system lifetime was several years. This target life time requires not only the use of low power sensing, data processing and communication hardware but also an energy efficient communication protocol and an extremely economical data communication policy. Since data communication is the most energy consuming process, the raw data acquired with the sensors have to be significantly reduced in the nodes with specific data processing algorithms to extract the relevant information. This data processing has to be performed with small computational hardware and memory resources.

To achieve this goal, WiseNET is used to provide an ultra low-power wireless communication link to the nodes. WiseNET, which was developed by the Swiss Center for Electronics and Microtechnology (CSEM), combines a dedicated duty-cycled radio with WiseMAC, a low-power MAC protocol designed for wireless sensor networks with low duty-cycle. WiseNET consumes about 100 times less power than comparable solutions available today. Since the processing of the acquired raw data can be very demanding for the limited computational and memory resources, this data processing was separated from the data processing for establishing the network functionality. The benefits of this design are a easier management of resources, simpler interaction between data reduction and network functionality processes, and more computational and memory resources for data processing. Low power MEMS sensing devices were used to reduce power consumption. Furthermore, an additional saving of energy was achieved by powering the sensing and data acquisition devices only during the data acquisition phase.

For minimizing the in situ maintenance effort, several tools have been developed to administrate the monitoring system by an operator from a remote control centre. The measurement tasks management tool enables to schedule new measurements and cancel or re-schedule the existing ones. The network management tool allows setting up and modifying the network topology, to regulate the power of the radio, to set the duty-cycle and to issue a statistics of the network operation. The reprogramming tool enables the operator to remotely reprogram the nodes. This functionality allows uploading new versions of application firmware, new processing algorithms or just improved software versions. The control center, which is linked to the on site wireless sensor network by a cell phone network or the internet, stores all incoming data, issued commands, and network statistics and provides tools to extract and visualize the logged data.

The monitoring of tensile forces in cable-stays of a road bridge based on ambient-excited cable vibration was selected as demo application. This application requires the sensing of low level ambient vibrations with high sampling rates and a powerful signal processing for reducing the acquired data to the relevant information: the natural frequencies of the cables. These were extracted from a spectrum that was computed using

an integer-only implementation of the Fast Fourier Transform (FFT) to increase the execution speed and reduce the memory size. Instead of transmitting the raw data, which consisted of thousands of data items, only a few natural frequencies were communicated by the network, thus reducing dramatically the communicated data. In a first phase, the monitoring system was validated on the indoor cable-stayed bridge at Empa. In a second phase, the system has been moved to the Stork Bridge in Winterthur, where six cable-stays were instrumented.

Based on the experience from the field deployment and according to the simulation results, the monitoring system promises to operate maintenance free for several years. The modular design enables easy adaptation to various monitoring applications and allows expanding functionality on module level without excessive redesign effort. In spite of the low cost hardware deployed, the results demonstrated that the natural frequencies were estimated by the monitoring system with an accuracy of about 2%. This figure perfectly matches the accuracy requirements within a typical assessment of civil structures.

The project has clearly demonstrated that it was possible to monitor civil structures on a continuous basis without maintenance for several years. A number of problems are still open that call for further research and development. Among those, there are long-term reliability and packaging, extension to large networks, design of deployment, commissioning and observation tools that can be used by non-specialists. At longer terms, sensor miniaturisation (including communication electronics) will allow to insert the wireless sensors directly in the structure at building time.

Kurzfassung

Dieses Projekt untersucht die Realisierbarkeit drahtloser Zustandsüberwachungssystemen. Die angestrebte wartungsfreie Lebenszeit des Systems soll mehrere Jahre betragen. Diese Lebensdauer erfordert einerseits den Einsatz von energiesparenden Komponenten, andererseits die Implementierung von energieeffizienten Datenverarbeitunsalgorithmen und Kommunikationsprotokollen. Da die Datenübertragung den energieaufwändigsten Vorgang darstellt, müssen die erfassten Rohdaten vor der Übertragung in den Knoten verarbeitet werden um die zu übertragende Informationsmenge zu reduzieren. Diese Verarbeitung muss auf Plattformen mit knappen Rechen- und Speicherressourcen bewältigt werden.

Um dieses Ziel zu erreichen wird die WiseNet Kommunikationstechnologie verwendet um energieeffiziente Kommunikationsverbindungen zu den Messknoten aufzubauen. Die WiseNet Technologie, welche am Centre Suisse d'Electronique et de Microtechnique (CSEM) entwickelt wurde, integriert ein dediziertes Radio, ein energiesparendes Media Access Control Protokoll (WiseMac) und bezüglich Energieverbrauch optimierte Routing Algorithmen. WiseNet verbraucht etwa 100 Mal weniger Energie als aktuelle, vergleichbare Lösungen. Da die Implementierung der Datenerfassung und Verarbeitung mit den limitierten Hardwareressourcen äusserst anspruchsvoll ist, wurde diese von der Kommunikation getrennt. Die Vorteile dieser Lösung sind ein vereinfachtes Ressoucenmanagement, eine vereinfachte Interaktion zwischen Datenreduktion und Kommunikation und eine erweiterte Rechen- und Speicherkapazität. Um den Energieverbrauch weiter zu reduzieren, werden energieeffiziente MEMS Sensoren verwendet und die Mess- und Datenerfassungseinheiten nur während der Messung eingeschaltet.

Um die Unterhaltsarbeiten vor Ort zu minimieren, wurden verschiedene Softwarewerkzeuge zur Fernwartung des Systems implementiert. Ein Werkzeug zur Verwaltung der Messaufgaben erlaubt neue Messungen zu disponieren und bestehende zu ändern bzw. zu löschen. Ein Werkzeug zur Administration des Netzwerks ermöglicht die Netzwerktopologie zu erstellen und zu modifizieren, die Sendeleistung der Kommunikationsmodule anzupassen und statistische Informationen über den Netzwerkzustand abzurufen. Ein Werkzeug zur Reprogrammierung der Knoten ermöglicht das Einspielen von Software-Updates, welche die Knotenfunktionalität verbessern oder erweitern. Im Kontrollzentrum, welches über eine Mobilfunk- oder Internetverbindung mit dem drahtlosen Zustandsüberwachungssystem verbunden ist, werden die Messdaten, sowie ausgeführte Kommandos und Netzwerkstatistiken gesammelt und gespeichert. Die Auswertungswerkzeuge erlauben auf diese gespeicherten Daten zuzugreifen und diese zu visualisieren.

Als Demonstrationsanwendung wurde die Überwachung der Zugkräfte der Kabel einer Schrägseilbrücke gewählt. Die Überwachung basiert auf der Messung ambient

angeregter Kabelschwingungen. Diese Anwendung erfordert das Erfassen von Vibrationen mit sehr kleinen Amplituden und eine anspruchsvollen Datenverarbeitung, um die Rohdaten auf die wesentlichen Informationen, die Kabeleigenfrequenzen zu reduzieren. Diese wurden aus dem Spektrum, welches mittels einer Integer-FFT berechnet wurde, bestimmt. Die Integer-Implementation wurde gewählt um die Verarbeitungsgeschwindigkeit zu erhöhen und den Speicherbedarf zu reduzieren. Anstatt der Rohdaten welche aus tausenden Datenpunkten bestehen, werden lediglich einige Eigenfrequenzen übertragen. Dies reduziert die zu kommunizierende Datenmenge erheblich.

Basierend auf den Erfahrungen aus den Feldinstallationen und übereinstimmend mit Simulationsresultaten verspricht das Zustandsüberwachungssystem mehrere Jahre wartungsfrei betrieben werden zu können. Der modulare Aufbau erlaubt, das System für verschiedene Überwachungsanwendungen einfach anzupassen und Erweiterungen auf Modulebene ohne komplette Neukonstruktion vorzunehmen. Obwohl das System aus billigen Komponente aufgebaut wurde, zeigen die Resultate, dass die Eigenfrequenzen mit einer Genauigkeit von 2% bestimmt werden können. Diese Genauigkeit entspricht den typischen Anforderungen, welche eine Zustandsbewertung von Bauwerken stellen.

Das Projekt demonstriert klar die Machbarkeit mehrjähriger, wartungsfreier Zustandsüberwachung von Bauwerken. Einige Probleme bleiben bestehen und erfordern weitere Forschungs- und Entwicklungsarbeit. Dazu zählen Zuverlässigkeit, Erweiterung hin zu grossen Netzwerken, effiziente Planung der Feldinstallation und Bedienungs- und Beobachtungswerkzeuge für Nicht-Spezialisten. Langfristig wird die Miniaturisierung von Sensoren und Kommunikationselektronik die Integration von Überwachungssystemen während der Bauphase erlauben.

Résumé

L'objectif de ce projet est de vérifier la faisabilité d'un système d'observation de grands ouvrages à l'aide d'un réseau de capteurs sans fil. La durée d'opération sans maintenance visée est de plusieurs années. Une telle cible implique de réduire la consommation de tous les éléments de la chaîne tant matériels que logiciels pour le captage, le traitement et la transmission sans fil. On aura besoin notamment de protocoles de communication très efficaces en terme d'énergie et d'une politique de transmission d'information très économique. Comme la communication sans fil est un gros consommateur d'énergie, la quantité d'information acquise par les capteurs ne peut pas être transmise telle quelle mais réduite à l'aide d'algorithmes de traitement qui extraient l'information pertinente. Ce traitement doit bien évidemment se faire sur des plates-formes dont les capacités de traitement et en mémoire sont très réduites ce qui constitue un défi supplémentaire.

Pour atteindre l'objectif visé, le projet se base sur le système de communication sans fil à très basse consommation WiseNET développé par le Centre Suisse d'Electronique et de Microtechnique (CSEM) qui combine une radio dédiée avec un protocole d'accès au milieu de transmission à très basse consommation (WiseMAC) et des algorithmes de routage spécialement étudiés. Dans cette configuration, un noeud WiseNET consomme environ 100 fois moins que les solutions disponibles sur le marché. Dans le cadre du projet, pour des raisons de disponibilité, seuls les protocoles de WiseNET seront utilisés, le gain étant encore très significatif. Comme le traitement des données des capteurs peut être très demandant en terme de puissance de calcul et de mémoire, l'unité de traitement des données a été séparée de l'unité de communication. L'avantage supplémentaire est de réduire la gestion des ressources, de simplifier les interactions entre la partie de traitement et réduction des données et la communication et d'offrir plus de ressources. Pour réduire la consommation de la mesure, des capteurs à technologie MEMS ont été utilisés. De plus, seuls les éléments nécessaires sont maintenus sous tension. Le système de mesure est notamment éteint en dehors de la phase d'acquisition.

Pour minimiser les efforts de maintenance sur site, plusieurs outils ont été développés qui permettent d'administrer le système depuis un centre de contrôle à distance. L'outil de gestion des tâches de mesure permet d'établir de nouveaux échéanciers de mesure, de modifier ou de supprimer des échéanciers existants. L'outil de gestion du réseau autorise l'établissement d'une topologie du réseau différente de celle trouvée automatiquement et sa modification, le modification des puissances d'émission et la demande de statistiques d'utilisation du réseau. L'outil de reprogrammation permet à l'opérateur de changer le code des noeuds à distance. Cette fonction permet de charger de nouvelle versions du logiciel d'application, des nouveaux algorithmes de traitement ou tout simplement des versions améliorées du logiciel. Le centre de contrôle est relié au site par un lien téléphonique ou par Internet. Il stocke toutes les données reçues, envoie des

commandes de gestion au réseau et aux application, récupère les statistiques et offre des outils pour extraire et visualiser les données acquises.

Les concepts développés ont été validés sur une application de démonstration, l'observation des forces de tension dans les câbles d'un pont routier suspendu ou haubané sur la base des vibrations naturelles. Cette application requiert la mesure des vibrations ambiantes à une haute fréquence d'échantillonnage et un traitement du signal sophistiqué pour extraire de ces données l'information pertinente. Cette extraction se base sur du spectre fréquentiel calculé à l'aide de transformée de Fourier rapide implantée par des calculs sur des entiers. Cette technique permet d'accélérer la traitement et de réduire la mémoire nécessaire de manière significative. Au lieu de transmettre les données acquises, ce qui nécessiterait une très large bande passante, seules quelques fréquences naturelles d'oscillation sont transmises ce qui réduit la charge sur le réseau de manière drastique. Le système a d'abord été validé sur une un pont haubané dans une halle de l'EMPA. Dans une seconde phase, le système a été installé sur le pont Stork à Winterthur où six câbles porteurs ont été instrumentés.

Sur la base des expériences in-situ et des résultats de simulation, il est possible d'espérer un fonctionnement sans intervention sur plusieurs années. La conception modulaire du système permet une adaptation aisée à d'autres applications d'observation et permet l'extension des fonctionnalités du module d'acquisition sans effort important de conception. Malgré l'utilisation de matériel bon marché, les résultats démontrent que les fréquences de résonance naturelles ont pu être mesurées avec une précision d'environ 2%. Ce résultat correspond parfaitement aux besoins en précision pour l'observation des structures en génie civil.

Le projet a montré la faisabilité d'une observation permanente d'ouvrages de génie civil sur de longues durées et sans intervention. Plusieurs problèmes restent ouverts qui qui devraient faire l'objet de recherches et développements complémentaires. On citera la fiabilité du système à long terme, l'extension à des grands réseaux et la création d'outils de déploiements et d'observation pour des non-spécialistes. A plus long terme, la miniaturisation des capteurs avec leur électronique de communication permettra de les intégrer directement dans la structure (dans le béton par exemple) lors de la construction de l'ouvrage.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ADC ..... Analog to Digital Converter
bps ..... bits per second
COTS ..... Commercial of-the-shelf
CPU ..... Central Processing Unit
CSEM ..... Swiss Center for Electronics and Microtechnology
DAC ..... Digital to Analog Converter
DC ..... Direct Current
DFT ..... Discrete Fourier Transform
EDGE ..... Enhanced Data rates for GSM Evolution
Empa ..... Swiss Federal Institute for Material Testing and Research
FEDRO ..... Swiss Federal Road Office
FFT ..... Fast Fourier Transform
GPRS ..... General Packet Radio Service
GSM ..... Global System for Mobile communications
GUI ..... Graphical User Interface
HCI ..... Host Controller Interface
I2C ..... Inter-Integrated Circuit Bus
IO ..... Input / Output
IP ..... Internet Protocol
MAC ..... Medium Access Control
MB ..... megabyte
MEMS ..... Micro Electro-Mechanical System
NRMSE ..... Normalised RMSE
PC ..... Personal Computer
PCB ..... Printed Circuit Board
PCMCIA ..... Personal Computer Memory Card International Association
RAM ..... Random Access Memory
RF ..... Radio Frequency
RISC ..... Reduce Instruction Set Computer
RMS ..... Root Mean Square
RMSE ..... Root Mean Square Error
ROM ..... Read Only Memory
SHM ..... Structural Health Monitoring
TCP ..... Transmission Control Protocol

UART ..... Universal Asynchronous Receiver / Transmitter
UI ..... User Interface
UMTS ..... Universal Mobile Telecommunication System
USB ..... Universal Serial Bus
µC ..... Microcontroller
WLAN ..... Wireless Local Area Network
WSN ..... Wireless Sensor Network

1 Introduction

1.1 Structural Monitoring

During the last decades, concepts and technologies for the monitoring of civil structures have attracted a rapidly increasing interest by civil infrastructure owners and consultants. This concern was made possible by the rapid progress in sensor, computing and communication technologies that allowed to develop structural monitoring systems that are sufficiently robust and reliable to be suitable for autonomous medium and long term applications at allowable costs. This technological breakthrough intensified the research on structural monitoring techniques with the goal to improve the effectiveness of assessment procedures for structures, to identify the degradation of structures or to assess their performance under real operation conditions.

The research activity was based on the conception that a cost efficient maintenance and management of civil infrastructure requires specific information about the current condition state of a structure. Assessing the condition of a structure is traditionally done by periodical visual inspection. However, the aging of civil infrastructures demands for an increased inspection effort producing considerable costs. Furthermore, periodical visual inspections do not provide up to date information. This yields an increasing concern for rationalized, autonomous and continuous inspection and monitoring concepts and technologies. Monitoring with physical sensors has the potential to provide up to date information. Furthermore, it furnishes information about transient processes like live loads, fatigue damage, vibrations, temperature, humidity etc. difficult to assess by using traditional inspection methods. In principle, this offers the potential to perform inspections on demand, reducing significantly the costs.

A whole specialized field of structural monitoring, called structural health monitoring, was established with the goal to develop techniques that are able to, possible autonomously, identify damages in a structure [1]. Here, damage refers to changes of material, geometric or system properties that affect adversely the performance of a structure. Influenced by research and development efforts performed in the 1970s and 1980s for offshore platforms and aerospace structures, the investigations were mainly focused on vibration-based damage identification methods. A great variety of methods have been developed and tested for simple structures in the laboratory [2][3][4][5]. Field tests on full-scale structures, however, have demonstrated that the effectiveness obtained under controlled conditions in the laboratory cannot be achieved [6][7][8][9]. The intrinsic complexity of civil structures, immanent uncertainties of the structural models, uncontrollable environmental factors and noise diminish dramatically the likelihood of successful damage identification in a stage that is beneficial for the infrastructure operator

[10]. After two decades of intensive research, structural health monitoring has therefore found little acceptance by infrastructure operators and remained essentially a research topic within the academic community.

Recently, structural monitoring has attracted the attention of scientists traditionally focused on developing and implementing cost-efficient maintenance and management processes [11]. In view of the recent developments in monitoring technology, which permits deployments at allowable costs, they started to systematically investigate the benefits of monitoring in terms of cost-effectiveness at all stage of bridges' life-cycle management and particularly as part of bridge management. The investigations are mainly focused on the efficient integration of quantitative structural monitoring data into reliability assessment frameworks and the updating of prediction models [13][14]. The goal is not to supersede traditional inspections with monitoring but to complement and optimize the inspection and assessment process with modern tools. The expectation is that the additional quantitative information provided by monitoring will assist the efficient spending of available budgets by permitting more focused intervention strategies. This monitoring approach, which is less ambitious than the structural health monitoring approach based on damage identification, has the advantage to be easier to integrate into existing bridge maintenance and management processes and to be conceptually closer to the current engineering practise.

Today, however, in daily civil engineering practise, the application of structural monitoring is still an exception. Deployments are mainly limited to very large or severely deteriorated structures or are performed for demonstration and research scopes [15][16]. Important factors, which prevent the large scale deployments of monitoring is the lack of incentives provided by codes or accepted guidelines. Monitoring guidelines have been published by different national organisations [17][18][19]. These guidelines, however, are too monitoring technology oriented and do not provide a guide how to design and implement a monitoring system for a focused objective on a specific bridge. Furthermore, although less critical than a decade ago, costs (instrumentation, deployment and maintenance costs) are still an important issue since current monitoring technology and services are generally quite expensive. The typical instrumentation costs per channel are in excess of several thousands of Swiss Francs (deployment and maintenance costs are not included) which imply high investment costs for service providers and high service costs for bridge operators even for small size monitoring systems.

1.2 Wireless Sensor Network

1.2.1 Motivation

Within the framework of reliability based assessment, which does not require a permanent monitoring of a structure, conventional structural monitoring systems have serious drawbacks. The most important drawback is the cabling of the sensors to the data processing and storage unit. Since every single sensor has to be connected individually for data acquisition, in a star topology, the installation of such systems tends to be labor-intensive, time consuming and therefore expensive. Especially in the field of civil engineering where the structures are typically large, the sensors can be located long way away from the data acquisition unit which results in high installation costs (see Figure 1 and Figure 2). Wired sensor systems have limited applications, since the wiring during construction or the wiring on existing structures is complicated and the wires might interfere with the functionality of the structure. The installation costs have shown to be a major issue preventing large scale applications of short and medium term monitoring for civil infrastructures. Due to the wires, such monitoring systems are susceptible to failure due to cable cuts involving considerable maintenance effort, leading to loss of data and interruption of the monitoring process. Cabled systems also tend to offer a limited flexibility in terms of rearrangement of sensors and scalability.

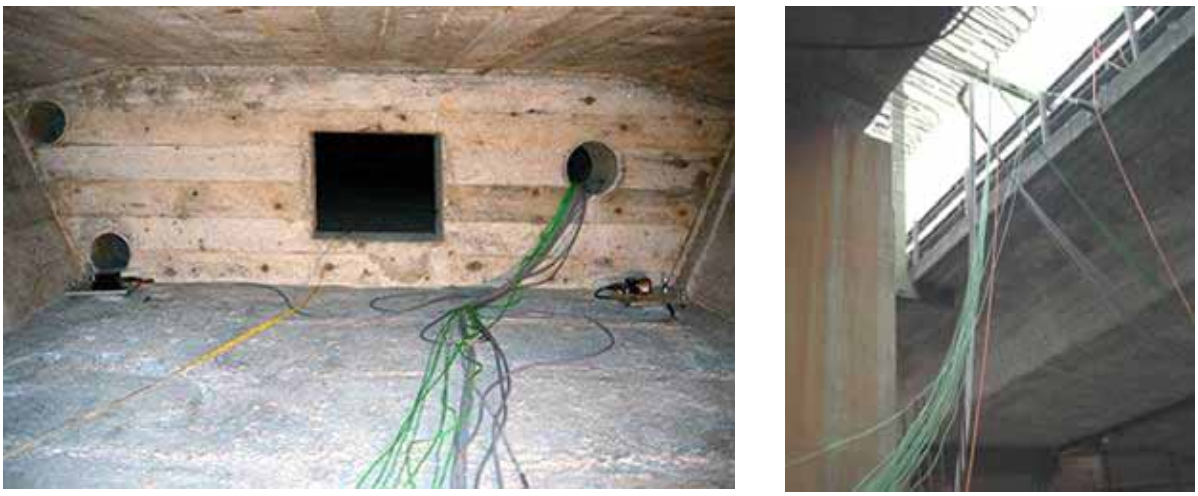

Figure 1: Cabling in the traditional wired SHM installation (1).

Figure 2: Cabling in the traditional wired SHM installation (2).

The adoption of wireless sensor network (WSN) techniques to monitoring applications promises to overcome these drawbacks. Wireless networks are much easier to install on existing infrastructures or in the environment, more flexible and are expected to be less costly than wired networks.

Recent advances in embedded system and radio technologies led to the idea of wireless sensor networks. The development of wireless sensor networks requires technologies from three different research areas: sensing, communication, and computing. Thus, advancements in each of these areas have driven research in sensor networks.

Although the elimination of the cables connecting the sensors to the central logging unit solves the cable related problems mentioned above, new WSN related challenges arise. Particularly the resources to power the wireless sensing devices are highly affected by the absence of cables. In fact, the nodes of the network have to be powered by autonomous sources like batteries or solar cells that are severely limited in capacity. The limited energy resources on each node present the most restricting factor in designing and implementing WSN based monitoring systems for long-term applications. Therefore, novel monitoring and measurement techniques as well as communication strategies, which take into account the limited energy resources, have to be developed.

1.2.2 Brief historical overview

The initial research in wireless sensor networks was mainly driven by military applications like battlefield reconnaissance and surveillance, nuclear, biological and chemical attack detection etc. These projects focused on ad hoc, multi-hop wireless sensor networks that consisted of thousands of immobile nodes randomly distributed over a large

geographical area (e.g. Smart Dust). The nodes were tiny (hardly noticeable), severely resource constrained and homogeneous (identical hard- and software).

Subsequently, the emergence of civilian applications of wireless sensor networks in different fields (environmental monitoring, home automation, health applications, production, inventory and delivery control etc.) produced a significant diversification of requirements with respect to deployment, mobility, size, cost, network topology, lifetime etc. and therefore a flourishing of academic and commercial wireless sensor network platforms. To cope with these requirements, the platforms increased in size, computational resources and hardware as well as software complexity.

The first commercial platforms appeared in the early 2000s. The most important platform was Crossbow's Rene mote, which emerged from the WeC mote developed at the University of California-Berkeley and which evolved later to the popular Mica platform. These platforms were the precursors of the recent Mica2 and MicaZ platforms [21]. A major reason for the popularity of Crossbow's early mote platforms was their open source policy with both hard- and software design open to the public. This policy built the base for the widespread diffusion of TinyOS as operating system for wireless sensor networks [22]. Today, various commercial platforms with different characteristics in terms of computing resources, sensor interfaces, software architecture etc. are available, which allow to cope with a wide spectrum of civilian applications [21].

1.2.3 Wireless sensor networks in structural monitoring

The investigation of wireless sensor networks for structural monitoring is a very young discipline. The first systematic studies started in the first years of this century [22]. Most research applications of wireless sensor networks in structural monitoring were performed with short-term deployments ([25], [26],[27],[28]). In these investigations, which were performed within the framework of vibration based damage identification, strategies for prolonging the lifetime of WSN and their implications for the hard- and software design were investigated only marginally. One of the first efforts addressing this issues was undertaken within the EC project "Sustainable Bridges" [24]. Within this project, a long-term field deployment, which is designed to address the issues related to medium and long term monitoring is presented in [29]. This monitoring system is specifically designed for applications with high data processing workload. The system targets an overall system lifetime of several months to more than a year and includes several methods for reducing energy consumption: ultra low power hardware components, multi-hop communication, low duty-cycle operation and significant data reduction. The latter method, which achieves a significant reduction of transmitted data by decentralized data processing, is a key aspect for enabling a long node lifetime.

As was demonstrated by strain monitoring of a steel railway bridge, for short time deployments, data reduction is less critical [30]. Nevertheless, the energy consumption

aspect still remained a main issue, since conventional resistive strain gages are power hungry sensors that severely reduce the lifetime of the monitoring system. To achieve the targeted lifetime, the strain gage was powered only when a train passed the bridge. An ultra low power acceleration sensor, which was permanently in operation, was used to recognize the approaching trains and to turn on the strain gage and the data acquisition unit for data recording.

1.2.4 Overall Structure

A wireless sensor network is essentially a computer network consisting of many small, intercommunicating computers equipped with one or several sensors. Each small computer represents a node of the network and is commonly called sensor node or mote. The communication within the network is established using radio-frequency transmission techniques.

All sensor nodes are equipped with specific sensors tailored to their measurement tasks. The data that is to be communicated is generated from the sensor measurements.

The sensor nodes typically form a multi-hop mesh network by establishing communication links to neighbour nodes. Such multi-hop network topology with one root node is shown on Figure 3.

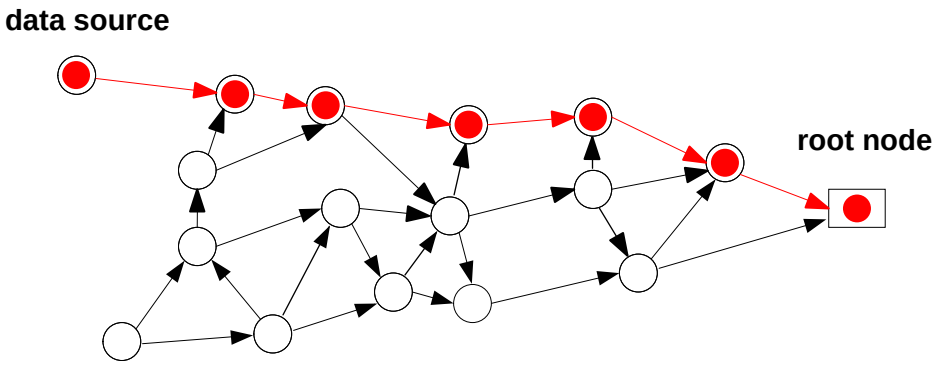

Figure 3: Multi-hop communication in a mesh network. The data packet from the data source is forwarded by the other network nodes towards the root node.

In a multi-hop network, sensor nodes act as data sources and as relaying stations, receiving and forwarding data from adjacent nodes. In most cases, all the data is communicated towards the so-called network sink (data sink, root node). The network may have one or more root node.

The main advantage of using multi-hop transmissions when communicating data to the root node is that it is more energy efficient than when each mode would make a direct

connection to the root node. The distances between some of the network nodes and the sink are very large, which would require too much power to directly transmit the data to the sink. Because the required transmission power is proportional to the square[1] of the transmission distance, it is more energy efficient to have multiple connections over shorter distances than having a single transmission over the whole distance.

Furthermore, sometimes, direct connection is not possible due to the signal degradation caused by physical properties of the signal path. Various physical objects, such as the metal elements of buildings, or other electronic devices, which emit interfering signals, negatively affect the signal propagation. Using multi-hop communication, data can be forwarded over alternative routes. In practice, regulations limit the emission power of radio transmitters thus limiting the possible reachable distances. Multi-hop transmission is hence mandatory in most practical settings.

A root node is connected to the base-station where the data from the network is collected and processed, or further forwarded to a remote data collection centre. The base station establishes a communication link to a data logging unit or remote sites (e.g. control centre) by using standard wired or wireless communication technologies like UMTS or WLAN. Figure 4 illustrates a schematic multi-hop network deployed on a road bridge.

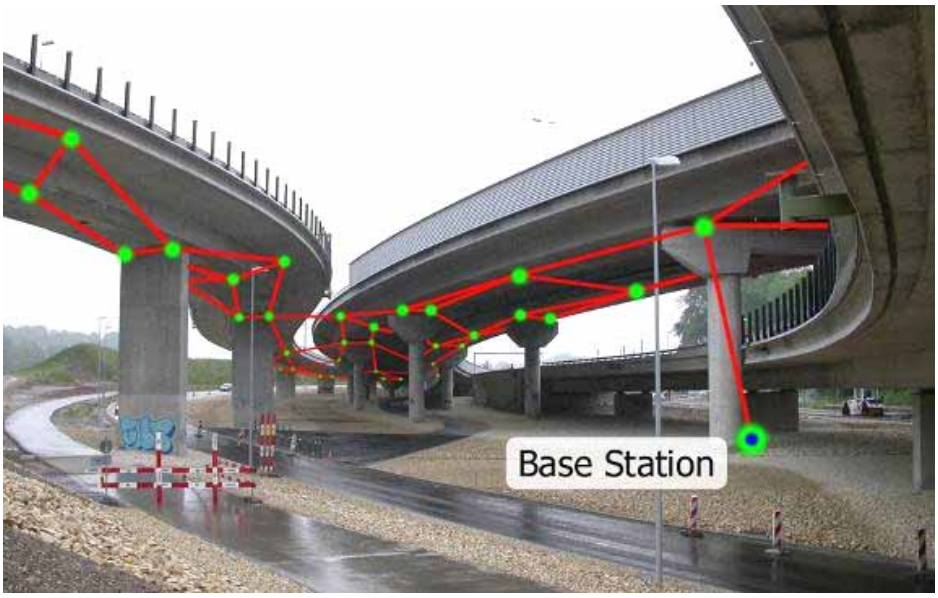

Figure 4: Schematic view of a multi-hop network deployed on a road bridge. The green spots illustrate the sensor nodes and the red lines illustrate the communication links.

[1] This is the attenuation in free space.

1.2.5 Hardware Architecture

The sensor nodes are the fundamental components of the wireless sensor network. In order to enable WSN based monitoring applications, the sensor nodes have to provide the following basic functionality:

- signal conditioning and data acquisition from different sensors
- temporary storage of the acquired data
- processing of the data
- analysis of the processed data for diagnosis and alert generation
- self monitoring (e.g. supply voltage)
- scheduling and execution of the measurement tasks
- management of the sensor node configuration (e.g. changing the sampling rate)
- reprogramming of data processing algorithms
- reception, transmission and forwarding of the data packets
- coordination and management of communication and networking

To provide all of the afore mentioned functionality, a sensor node is built up of one or more sensors, a signal conditioning unit, an analogue to digital conversion module, a processing unit with memory, a radio transceiver and a power supply, as illustrated on Figure 5. The integration of these components on one board is referred to as WSN platform.

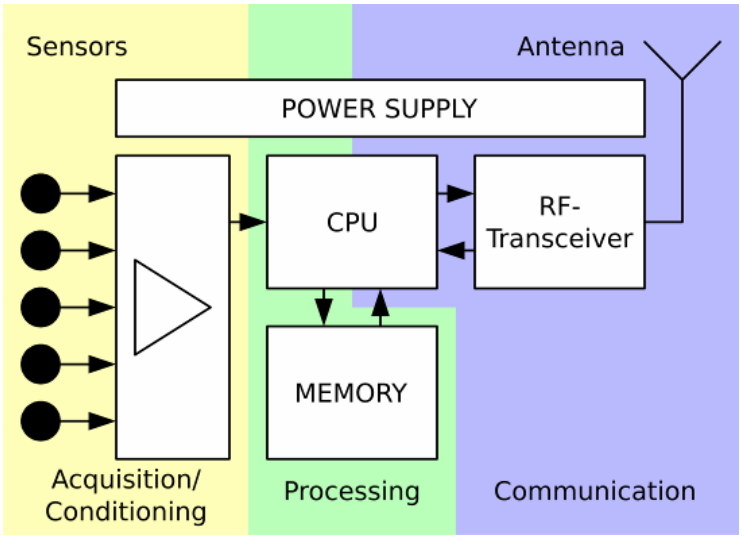

Figure 5: Hardware system architecture of a sensor node.

Various hardware platforms for wireless sensor networks are commercially available today and new ones emerge regularly. Many other prototype platforms have been designed

and produced for specific purposes by researchers to allow deploying of ad hoc WSN. The diversity of platforms offers the possibility to choose a platform which best fits the needs of a specific application. Some of these platforms are presented in [31].

One representative platform that was often used to build ad hoc networks is the Tmote Sky platform (see Figure 6). It is a microcontroller operated low-power system with an onboard radio and antenna, and with some already prepared sensor interfaces, making it a good base to build custom WSN nodes. It features an USB interface, which is used to program it in a convenient way.

Figure 6: Tmote Sky WSN platform

Tmote Sky platform was selected to build WSN nodes also for this project. The motivation for selection of this platform is described in section 0.

1.2.6 Power Consumption

A sensor node has to operate for an adequate period of time only from its own limited power source, usually a set of batteries. Therefore, optimally managing the power consumption is one of the main challenges in wireless sensor networks. This is especially true for long-term deployments.

For maintenance reasons, the nodes must be energetically autonomous by using batteries that need not to be replaced or recharged. In many application scenarios, the targeted node lifetime typically ranges from two to five years, imposing drastic constraints on power consumption.

This calls for a careful system design which requires an optimization regarding energy consumption across all of system layers; from physical, media access control, up to routing, transport, and application layers.

Network lifetime is maximized by reducing the power consumption with the following strategies:

1. **Low-power components**

   It is obvious that the selection of low-power components is the key factor to reduce power consumption in the sleeping mode as well as in the active mode. Section 2.1 describes the low-power components used in this project.

2. **Duty-cycling**

   Duty-cycling is one of the main methods to reduce the system energy consumption. Having the system sleeping for most of the time (power-down state of CPU, radio, and sensors) and awake only when it is required is referred to as duty-cycling. In order to achieve a lifetime of the WSN node of several years, it is required to have a very low duty-cycle, i.e. to keep the system active for only a few % of the time.

3. **Multi-hop communication**

   Multi-hop can be used to save power because transmitting information over several short hops consumes less energy than transmitting it over the complete distance at once. However, current transceiver offer little power savings when reducing emission power. On the other side, because the data need to be transported over relatively long distances, we use a chain of several short hops to the target. Therefore, each mote acts as data source as well as relaying station, forwarding data of other motes. This way all motes together form a multi-hop network by establishing communication links to their neighbour nodes.

   Another important benefit that makes the multi-hop networks attractive for monitoring applications is that the data can be forwarded over alternative routes when using the multi-hop communication. This improves the network robustness to sensor (relay) node failure.

4. **Data reduction**

   In terms of power consumption, wireless data transmission is much more expensive than the data processing. Therefore, one of the best ways to save energy is to reduce the acquired raw data before transmitting it over the wireless link. This is often feasible since the raw data contains a lot of redundant or even irrelevant information which can be discarded. It is possible to extract a small number of characteristics describing the physical process quite well.

Many recent WSN based SHM systems transmit the raw data streams to the base station and analyze them in the traditional centralized way. Without introducing huge batteries, this is not a viable solution if a system lifetime ranging from several months to several years is targeted. Distributed analysis algorithms for long term monitoring applications which allow decentralized data reduction or even condition assessment have to be introduced.

The goal of the processing on the motes is to reduce the amount of data. There are several methods to achieve this. One of them is data compression, which allows encoding the data in a new representation that uses fewer bits than the original un-encoded data. This is done by using specific encoding schemes, which are either lossless or lossy.

A further way to reduce the amount of data is to transform it into a new kind of information that requires less space in terms of bits. Simple data reductions can be maxima, minima, mean values, RMS, cycle counts or statistical distributions of a physical quantity.

A more advanced data reduction method is the extraction of just the key physical parameters out of the raw measurement data stream that characterize the physical process adequately enough. One example of such in-mote data processing is the extraction of the natural frequencies of vibration data. Such an approach of networked distributed data processing is used in this project to drastically reduce the acquired data. It is described in section 2.2.1.3.

Another way to further reduce the amount of transferred data is to perform even the condition assessment directly on the motes. The mote software would analyze the data according to the given criteria and decide if the result is relevant or not. Irrelevant results can be discarded already at mote level, leaving only results that qualify as relevant to be transmitted. For example, a simple criterion can be the use of a threshold. If the result of the data analysis exceeds a given threshold or range of values, it will be transmitted, and otherwise not.

5. Energy-efficient MAC protocol

The wireless media access control (MAC) layer plays the most crucial role in the overall energy efficiency of communication protocols, especially for networks that operate their radio modules in a low duty-cycle mode. Since radio communication requires by far the most power, low-power MAC protocols have a deep impact on the overall power consumption of the system.

The MAC protocol is responsible for energy-efficiency of the radio communication by coordinating the medium usage, i.e. synchronizing radio active times of the neighbouring nodes, reducing the energy loss due to packet collisions, idle listening and packet overhearing.

The low-power MAC solution used in this project is WiseMAC. This topic is covered in section A.2.3.

1.2.7 WiseNET

WiseNET is an ultra low-power platform for implementing wireless sensor networks that achieves low-power operation. WiseNET is developed by Swiss Center for Electronics and Microtechnology (CSEM). WiseNET is used in this project to provide a low-power wireless communication link to the sensing and processing boards, in order to form a low-power wireless sensor network.

WiseNET combines a dedicated duty-cycled radio with an optimised protocol stack. This stack is composed of a radio layer implementing the transmission and reception of frames, a MAC layer implementing WiseMAC and a routing layer transporting data packets following either predefined or dynamically discovered links towards the sink. The complete WiseNET solution consumes about 100 times less power than comparable solutions available today [32]. The WiseNET protocol stack has been adapted and optimised to fit the project's requirements.

A WiseNET node (WiseNode) consists of a basic board comprising a low-power radio, a microcontroller and some IO interface connectors (I2C, UART, etc.). The low-power WiseMAC multi-hop protocol is implemented on the microcontroller.

Application specific boards with sensors host processor boards running the main application software can be stacked on the basic board. The host processor communicates with the WiseNode's microcontroller through a serial interface using a packet based HCI (Host Controller Interface) protocol.

WiseNET provides the application with a wireless low-power communication link. It is used in this project as a low-power wireless networking solution. An overview of WiseNet is given in section 2.1.1.2. A detailed description of WiseNET can be found in appendix A.

1.3 Monitoring System Overview

The wireless sensor network is used to build a monitoring system. The scheme of the monitoring system used in this project is illustrated in Figure 7.

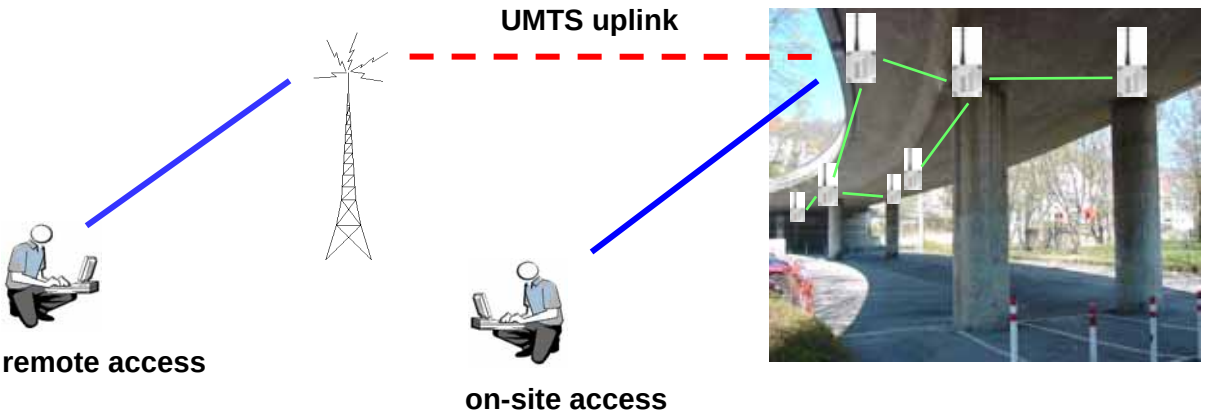

Figure 7: Schematic view of the monitoring system is composed of the installed WSN, local and remote access and the UMTS communication link in between.

The complete monitoring system is mainly composed of the following subsystems:

- **A wireless sensor network**

  This subsystem is installed on the structure and can be described as a series of strategically placed sensing units which are wirelessly connected. The functional role of this subsystem is to acquire measurements from the sensors, perform data processing, and communicate the results of the measurement analysis to the data sink. The wireless communication of this subsystem is provided by WiseNET, (see section 2.1.1.2 and appendix A).

- **A link between the sensor network and the control centre ***

  This link is particularly important if the control centre is not located on the site. It gives the operator the possibility to observe, control and configure the sensor network remotely. The remote access to the WSN is done by establishing a remote connection to the network base-station by using standard communication technologies and infrastructure like UMTS.

- **An operator console or a control centre ***

  This subsystem makes the information originating from the sensor network available to a user or operator. It implements the data visualization and condition representation, and it is responsible for long-term data storage. The WSN may also be accessed on-site through the operator console by establishing a direct connection to the network base-station. This subsystem is described in more detail in section 2.2.2.

As mentioned above the WSN in this project is built on top of WiseNET (see appendix A.3). WiseNodes are microcontroller-operated systems with an on-board radio and antenna. WiseNodes are wirelessly connected using their on-board radio and the necessary networking protocols such as WiseMAC (appendix A.2.3), an ultra-low power media access control protocol, and a dedicated routing protocol.

Sensing and processing boards called 'application board' are connected to the WiseNodes. These boards are a microcontroller-based low-power system equipped with sensors to measure the physical parameters of the structure. The application board is responsible for sensor management and measurement acquisition, signal conditioning and processing of the acquired sensor readings. It represents a host controller system for the WiseNode and runs the main application.

Each application board generates data, which is then passed-on to the WiseNode to be communicated over the multi-hop wireless network to the network sink at the base-station. Figure 8 provides a schematic overview of the WSN. The blue "WN" boxes represent the WiseNodes and the red "AB" boxes the application boards equipped with sensors.

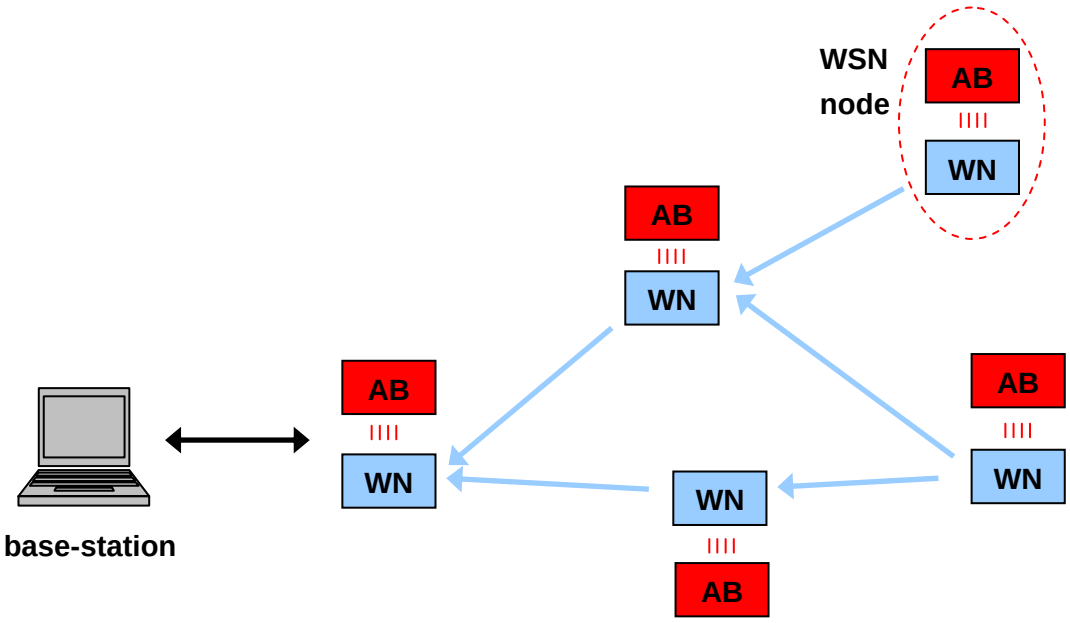

Figure 8: Schematic view of the developed WSN based on the WiseNET.

The systematic overview of the subsystems used to build the monitoring system is illustrated in Figure 9. The WSN is presented by the network motes on the right. These are sending and forwarding data packets to the network sink (mote 0) connected to the base

* These subsystems were developed previously as parts of another project, and are used within this project for the purposes of the field deployment (see section 3.2).

station. The base station provides the local (on-site) access to the network, as well as the remote access established by using the UMTS network infrastructure. The components of the remote control centre are displayed on the left. These components enable condition monitoring of the structure and the installed WSN, provide database storage for the incoming data and user access to configure the network and manage the measurements.

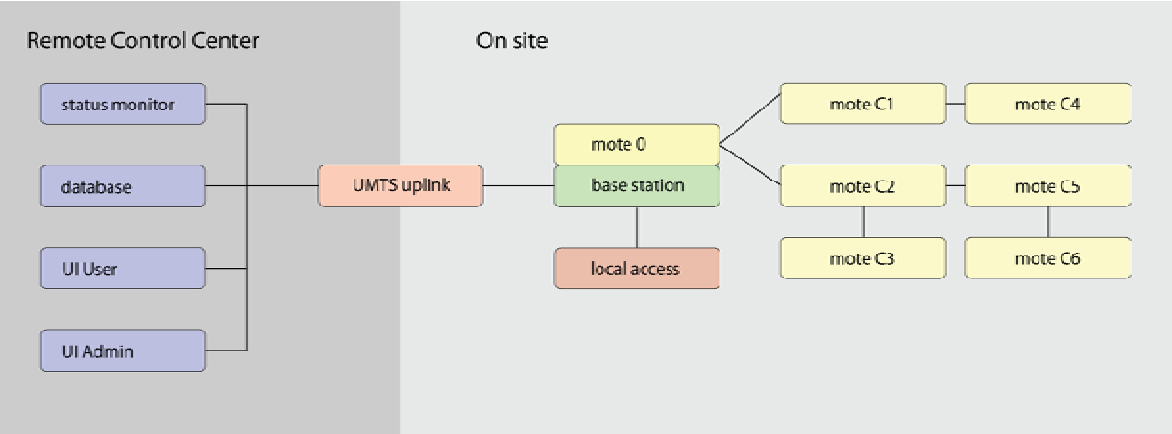

Figure 9: Overview of the developed monitoring system showing the wireless sensor network (right), the control components (left), and the link in between.

1.4 Tension Monitoring of Bridge Cable Stays

One potential application for wireless sensor networks is monitoring of cable tension force of cable-stayed bridges (Figure 10). Since a direct measurement of cable tension with load cells is complicated and expensive, cable tension is estimated indirectly by using natural frequencies obtained from measured cable vibrations. The cable tension is then estimated by comparing the measured natural frequencies with the predicted natural frequencies of a cable model. The first investigations of vibration-based cable tension evaluation were based on simple taut string theory [33][34]. More advanced identification methods include the effect of bending stiffness by simple approximation formulas of the natural frequencies [35][36]. A more refined method that does not rely on simple approximation formulas and considers also the effect of cable sag is presented in [37]. Exactly the same approach can be adopted for monitoring tendon forces on bridge with external post-tensioning.

Figure 10: Cable-stayed bridge (Millau viaduct, France).

The concept of vibration based cable tension monitoring is based on the taut string theory. A tensioned cable exited by a wide band excitation predominantly vibrates at its natural frequencies. The natural frequencies of a taut cable with negligible bending stiffness are given by:

$$f_k = \frac{k}{2L}\sqrt{\frac{T}{\mu}} \quad , \tag{1}$$

where f_k is the natural frequency of mode k defined as a function of the mode number k, T the tensile force of the cable, μ the cable mass per unit length, and L the cable length. A typical vibration power spectrum of a cable is shown in Figure 11. The peaks in the spectrum represent cable natural frequencies (modes) and can easily be determined.

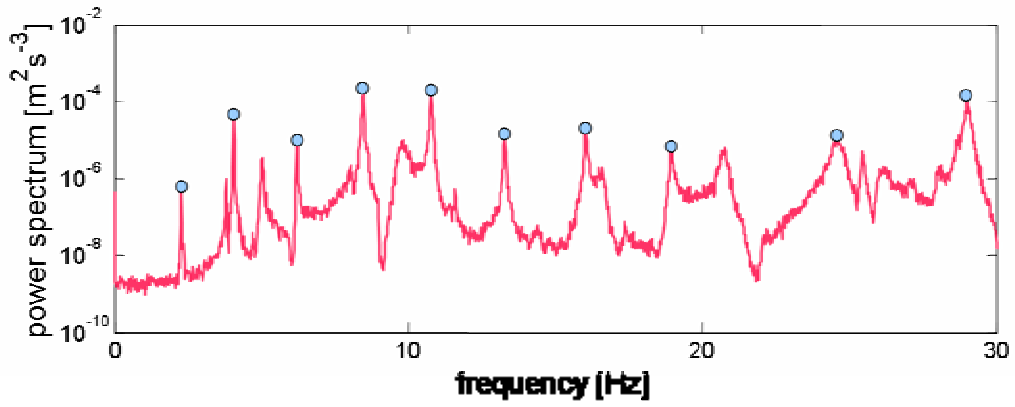

Figure 11: A typical power spectrum of the cable.

From relation (1) it can be seen that natural frequencies are defined by the cable tension. In fact, this dependence is used in various music instruments with strings, where the frequency of the string is set by applying the appropriate tension to the string. Therefore, changes in the cable tension result in changes in natural frequencies.

Equation (1) can be rewritten as

$$T = \left(\frac{2L}{k}\right)^2 \mu f_k^2, \tag{2}$$

what expresses the cable tension as a function of the cable frequency. Hence, by observing the natural frequency of the cable mode, a current tension of the cable can be estimated.

Moreover, the cable parameters *L* and *μ* do not need to be known in order to estimate the relative change in the tension force. If we assume that these parameters are constant, we can simplify expression (1) to the following:

$$f_k = C_k \sqrt{T}\ , \tag{3}$$

where C_k incorporates all the constant parameters in expression (1), and f_k symbolizes one specific natural frequency of the cable.

Using expression (3), the change in the monitored natural frequency can be expressed as:

$$f_{k1} = f_{k0} + \Delta f_k = C_k \sqrt{T_0} + \Delta f_k\ . \tag{4}$$

The change in a monitored natural frequency can also be expressed as:

$$f_{k1} = C_k \sqrt{T_0 + \Delta T}\ , \tag{5}$$

By equalizing the expressions (4) and (5), and using (3) the relation between the change in cable tension and the corresponding change in the cable frequency can be expressed as:

$$\Delta T = \frac{1}{C_k^{\,2}} \cdot \left(2 f_{k0} \cdot \Delta f_k + \left(\Delta f_k\right)^2\right). \tag{6}$$

If we are interested in a relative change of the cable tension expressed by the relative change of frequency, we can further modify the expression (6) using (3) into:

$$\frac{\Delta T}{T_0} = 2 \cdot \frac{\Delta f_k}{f_{k0}} + \left(\frac{\Delta f_k}{f_{k0}} \right)^2 , \tag{7}$$

which can be simplified by approximation to:

$$\frac{\Delta T}{T_0} \approx 2 \cdot \frac{\Delta f_k}{f_{k0}} , \tag{8}$$

when the relative changes in frequency are small ($\Delta f_k << f_{k0}$). Therefore, the observed percentile change in the monitored frequency (e.g. 2%) means that the percentile change in the cable tension force is two times bigger (e.g. 4%).

One test conducted at the Empa's Structural Engineering Research Laboratory aimed at observing changes in the natural frequencies of the cable stays of the indoor laboratory bridge (see Figure 12) caused by different load placements on the bridge deck. Figure 13 shows the change of one natural frequency over the time due to loading and unloading the bridge deck with additional mass.

Figure 12: Full-size pedestrian cable-stayed bridge at Structural Engineering Research Laboratory, Empa.

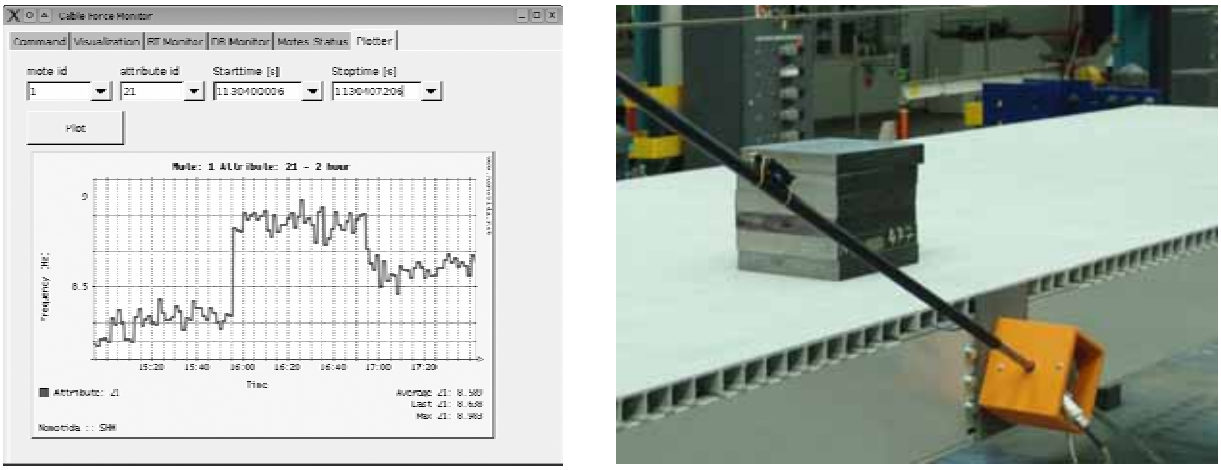

Figure 13: Plot of the estimated natural frequency of one cable stay (left). The steps in the plot correspond to three different loading states of the bridge deck (right).

2 Hardware and Software

This chapter describes hard- and software developed in this project to realize wireless sensor network based monitoring of natural frequencies of bridge cable stays. The system design of WSN modular motes, monitoring application, WSN base-station and control centre will be described.

2.1 Hardware

This section analyses the hardware used to develop the structural monitoring system that monitors natural frequencies of the bridge cable stays. The main focus was to design a modular WSN mote which can be used as a platform for various structural monitoring applications.

2.1.1 Mote Hardware

An overview of the system hardware architecture of a single mote is illustrated in Figure 14. The motes consist of three sub modules:

- communication module (WiseNode)
- processing module (application board)
- sensing module (sensor board)

The sensing module is a board with an accelerometer sensor and an analogue signal conditioning circuit. The application board allows connecting various sensors to the analogue to digital converter (ADC), but for this application a separate board with accelerometer sensor was required. The signal output from the sensor board is connected to the ADC of the application board.

The application board presents the processing module of the system. It is a microcontroller system with ADCs for sensor signal acquisition. The board is connected to the WiseNode and presents a host controller system to it. The application board establishes a bidirectional communication channel to the network sink over the WiseNET network. The results of the data analysis are transmitted over these channels to the network sink.

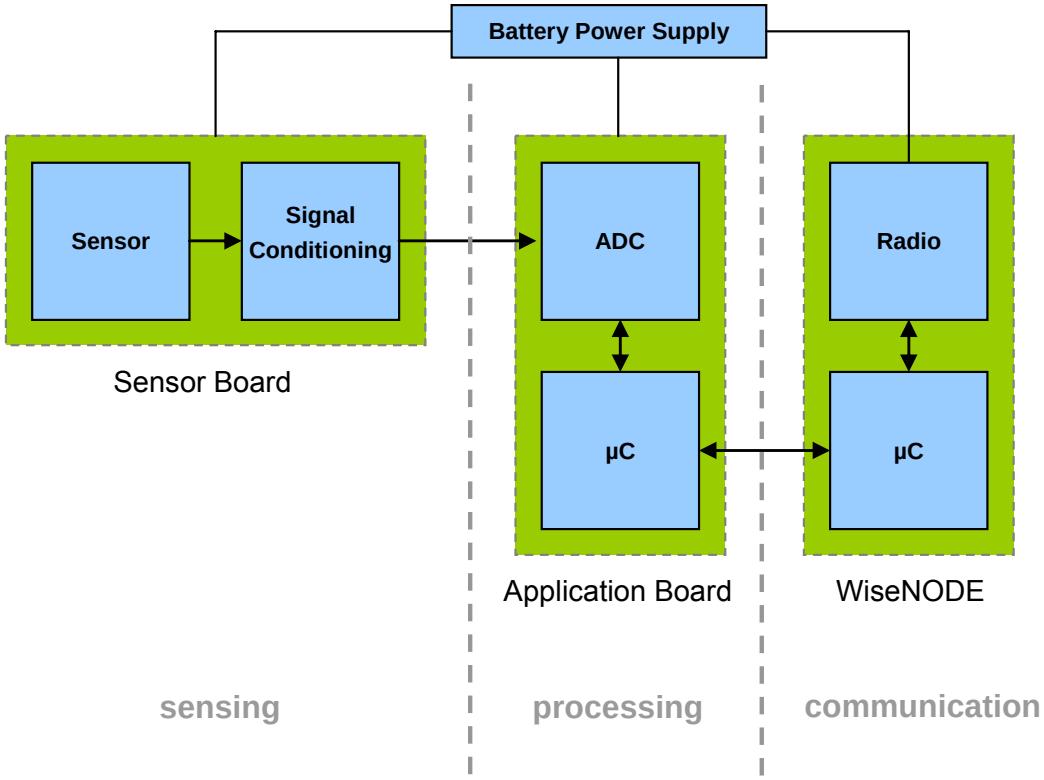

Figure 14: Each WSN mote consists of a WiseNode module (communication), an application board (processing) and a sensor board (sensing).

The WiseNode communication module is responsible for establishing the wireless communication links and managing the overall network structure. The WiseNode is a microcontroller based system equipped with a radio circuit. The low-power MAC protocol WiseMAC is implemented on this controller (see section A.2.3).

2.1.1.1 Application Board (Tmote Sky)

The application board is a microcontroller based low-power system with ADCs and various sensor interfaces. It is where the main application is implemented, responsible for the sensor management, measurement scheduling and measurement acquisition. The algorithms for digital signal processing of the measurement signal are implemented on this board as well.

Tmote Sky is a WSN platform from Moteiv Corporation and has been chosen in this project as the application board. Tmote Sky board is shown on Figure 15.

Figure 15: Tmote Sky platform was chosen as the application board.

Tmote Sky is in fact a very popular WSN platform, featuring its own on-board radio module and antenna. From the system overview in Figure 14 it is clear that the radio communication module of the Tmote Sky platform is not required as the wireless communication link is provided by WiseNode. In principles, the Tmote Sky board could have handled the communication with the sink. However, it would have been difficult to run the processing and the communication stack software concurrently. Tmote Sky has mainly been chosen as application module to avoid the development of a dedicated microcontroller board and because the boards were available at Empa. Furthermore, this splitting of communication and data analysis into two separated modules allowed developing the subsystems in parallel.

Select this platform instead of a low-power microcontroller system without radio functionality or developing a custom one is motivated by different reasons. The platform is ultra low-power and features convenient programming over USB. It offers already some interfaces for sensors and very good software support is available. Additionally, the on-board radio and antenna of Tmote Sky are in fact used for implementing the wireless reprogramming functionality. The communication link provided by this radio is established when a new firmware has to be uploaded and installed on the application boards. More about the wireless reprogramming feature is given in section 2.2.1.6.

The main components of Tmote Sky platform are the MSP430F1611 ultra low-power microcontroller from Texas Instruments, the Chipcon CC2420 low-power radio chip with on-board antenna and the USB interface for reprogramming and communication with the microcontroller.

The MSP430F1611 microcontroller features 10kB of RAM and 48kB of program memory. It is a 16-bit 8MHz RISC processor implementing several power-down modes with a very low sleep-current (max 21µA). The microcontroller can be powered-up from the sleep mode in about 6µs which allows a short reaction time upon the occurrence of an event. It has 8 integrated 12-bit analogue to digital (ADC) and digital to analogue converters (DAC). Detailed features of the MSP430F1611 can be found in its datasheet [37].

The Tmote Sky platform is equipped with the Chipcon CC2420 radio (2.4GHz, 250kbps) which is an IEEE802.15.4 compliant wireless transceiver. It features power management capabilities to ensure the low power consumption. However, since the wireless communication functionality is provided to the system by the WiseNode, this radio circuit is kept in power-off state to further reduce the power consumption.

The application board is referred to as Tmote Sky in the continuation of this document. Detailed features of Tmote Sky can be found in its datasheet [39].

2.1.1.2 WiseNode

WiseNode is a WSN platform used to build WiseNET, a low-power wireless network solution, as explained in detail in Appendix A. As previously shown in the modular mote system overview (Figure 14), the WiseNode provides the wireless communication functionality to the system. The latest version of the WiseNode module is WiseNode_V4 module shown in Figure 16.

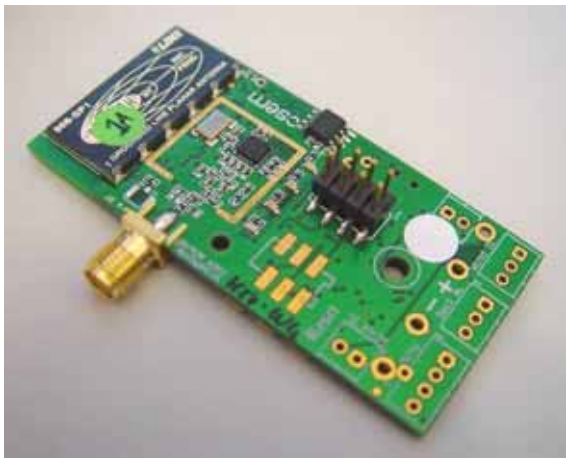

Figure 16: WiseNode_V4 module.

Similar to the Tmote Sky platform, WiseNode_V4 is also based on MSP430F1611 microcontroller, but it features a different radio chip, Chipcon CC1100 operating at 868 MHz. The WiseNode_V4 module is described in more detail in A.3.3.

2.1.1.3 Sensor Board

From the sensor board block diagram shown on Figure 17, it can be seen that the it features a signal amplification and signal filtering circuit. It is very important that the sensor board is as well a low-power solution since it significantly influences the power consumption of the whole system (see section 3.4.2.2). Hence, the required sensor and the associated

analogue signal conditioning circuitry are carefully selected and designed in order to reduce power consumption.

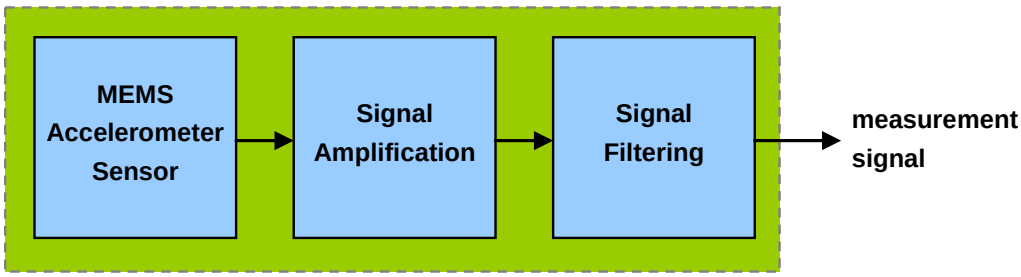

Figure 17: Block-scheme of the sensor board system.

The vibration measurements needed for the stay-cable tension force monitoring are acquired with an accelerometer sensor. A special MEMS[2] accelerometer sensor has been chosen since it has several advantages compared to conventional sensors. They are small, low power, and rather inexpensive. Current inexpensive MEMS accelerometers allow to measure ambient vibrations with peak amplitudes of several mg.

A 3-axis sensor LIS3L02 accelerometer from ST Microelectronics (see Figure 18) has been selected because of its good noise performance, low power consumption and low costs. More information about this accelerometer can be found in [40].

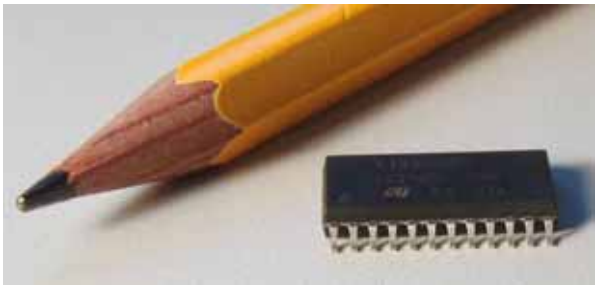

Figure 18: LIS3L02 MEMS accelerometer from ST Microelectronics.

The signal conditioning unit basically consists of amplification and of filtering circuitry. Both the amplification circuitry and the filtering circuitry are realized using low-power amplifiers AD8609 [41]. The signal amplification factor can be set by a resistor. It is currently set to 50. Detailed schematics of the signal amplification and filtering circuitry are shown in Figure 1 in appendix C.

[2] Micro-Electro-Mechanical Systems (MEMS) is the integration of mechanical elements, sensors, actuators, and electronics on a common silicon substrate through micro-fabrication technology

The analogue filter is a 1st order high-pass filter and two 2nd order low-pass Butterworth filters. The sensor output contains a DC signal, which limits the achievable amplification. The high-pass filter removes this DC signal and enables optimal amplification of the sensor signal. Currently, the filter high-pass cut-off frequency is set to 1.6Hz. The trade-off for the low cut-off frequency is a relatively long start-up time (see Table 1). The low-pass filters form the anti-aliasing filter for the signal acquisition. The low-pass cut-off frequency is set to 20Hz.

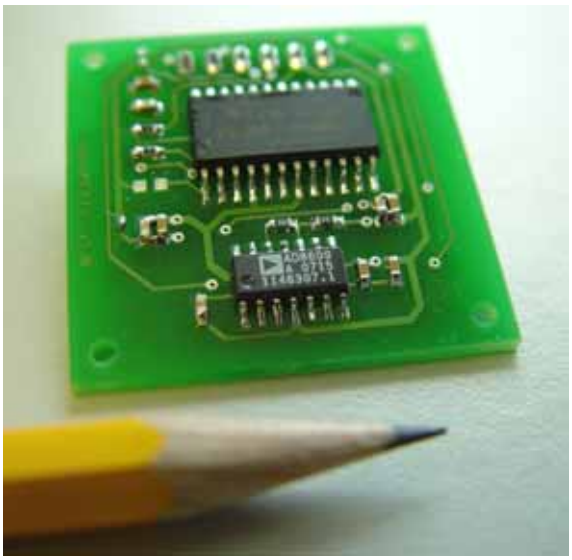

Figure 19: LIS3L02 MEMS accelerometer based sensor board with amplifying and signal filtering circuitry.

Table 1 summarizes some of the main features of the developed low-power sensor board. The parameters are set according to the characteristics of the acceleration signal of the monitored cables (see section 3.3.1).

Sensor Board Characteristic	**Rating**
Features	
Number of channels	2[3]
Power supply	1.8 - 5 V
Current consumption (max)	1.3 mA
Temperature range[4]	-40 – 85 ºC
High-pass filter order	1

[3] only X-axis and Y-axis of the sensor are in use.

[4] the sensor board is not tested to work in this temperature range, but it is expected to be capable of, as all of the used components are declared to operate at least in this temperature range -40 - 85ºC.

Low-pass filter order	4
Current Setup	
Signal amplification factor	50
High-pass cut-off frequency	1.6 Hz
Low-pass cut-off frequency	20 Hz
Acceleration sensitivity	33 mV/mg
Acceleration range	± 50 mg
Start-up time	~1 second

Table 1: Summary of the sensor board characteristics and setup

2.1.2 Base-station hardware

The base-station completes several tasks. It provides access to the wireless sensor network by a local wired, or a remote wireless Internet connection (for example GPRS or UMTS), as explained before and illustrated in Figure 9. Optionally, it provides the opportunity for local data logging, if Internet connection fails due to problems in the network of the telecommunication services provider. This requires a certain amount of local storage, e.g. a hard disk or a flash memory disk.

Since operating such a base-station requires much more power than the WSN, an unlimited power supply is advantageous. However, if this is not available the base station can optionally be powered by a solar panel with a rechargeable backup batterie.

The base-station functionality was implemented using an industrial embedded PC with USB, PCMCIA and Ethernet connectors and Aircard 850 from Sierra Wireless (UMTS, EDGE, GPRS, GSM capable, PCMCIA connector) with possibility to attach an external antenna for the case that the base station is located inside the structure where the radio signal could be too weak or the place is inconvenient to set up a connection. Figure 20 shows the industrial embedded PC used as base-station.

Figure 20: Base-station is an industrial embedded PC with a wireless Internet connection.

2.2 Software

This chapter describes the software components that have been developed and used to achieve the intended functionality of the WSN monitoring system.

2.2.1 Software Mote

The Tmote Sky and WiseNode platforms are both separated systems running their own firmware. They are communicating over a WiseNode's Host Controller Interface (HCI), using a 9600bps UART connection (see section A.2.2).

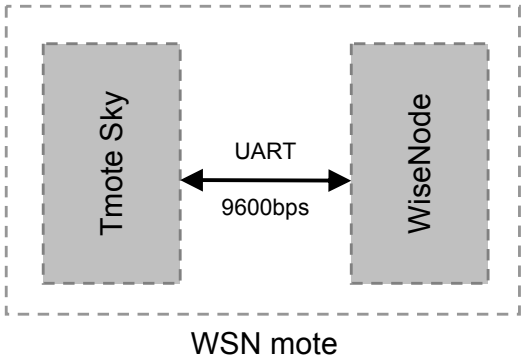

Figure 21: Two separate software systems on each mote.

The operating system running on the Tmote Sky is TinyOS. It is an open source component-based operating system designed for wireless embedded sensor networks [42]. Applications for TinyOS are written in nesC, an extension to the C language which enables event-driven component based programming [43].

2.2.1.1 System Overview

The modular software architecture running on the mote is illustrated in Figure 22. Some of the software components of the Tmote Sky platform are going to be discussed in this section. The software components of the WiseNode are described in appendix A.2. The layers of the WiseNET communication stack are illustrated here just in order to get an idea of the complete embedded software.

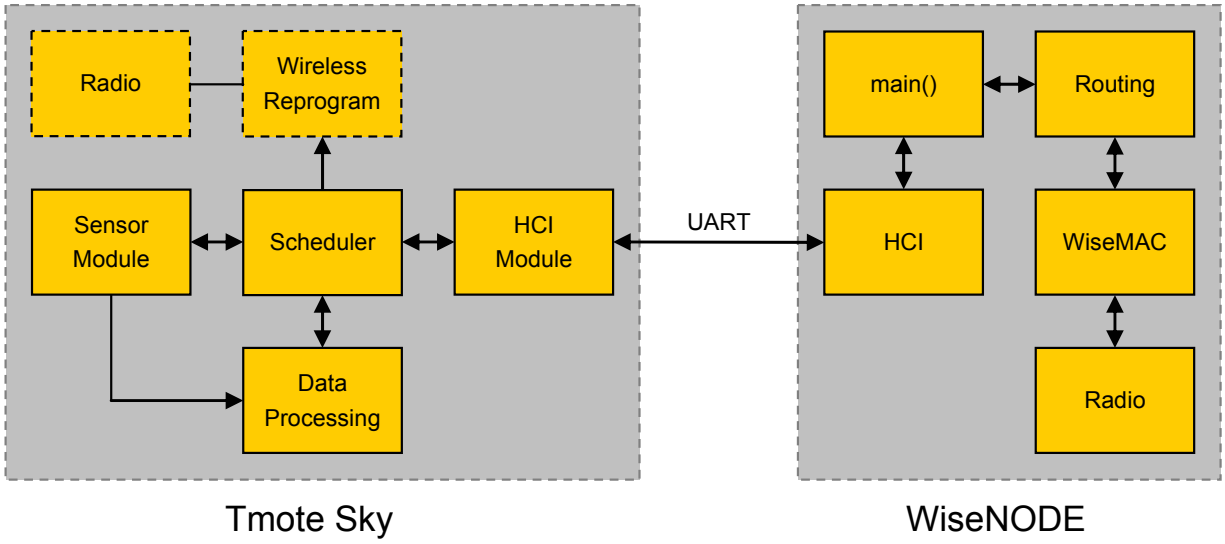

Figure 22: Mote software overview.

The central part of the Tmote Sky firmware is the scheduler module (see section 2.2.1.2), which manages the measurement acquisition. If a measurement has to be taken, the scheduler turns on the required sensor on the sensor module. After power-up the signal acquisition is activated. The sensor module is responsible for the control of the analogue to digital conversion of the conditioned sensor signal from the sensor board (see section 2.1.1.3).

After completion of the signal acquisition, the measured data is automatically provided to the Data Processing module where the in-mote data processing takes place (see section 2.2.1.3). In the proposed cable stay tension monitoring application, the data processing module actually represents the core component of the system. It is where the most crucial functionality for achieving the long system lifetime – a drastic data reduction, is implemented.

Finally, the results of the data processing are fed to the scheduler module which communicates them to the sink. The scheduler does so by sending it over the HCI module (see section 2.2.1.5) to the WiseNode, which transmits it over the multi-hop network to the network sink. The HCI module is where the HCI protocol stack for the communication with WiseNode is implemented.

After the measurement and the data processing completed and the results have been sent, the scheduler plans the next measurement, sets the internal alarm timer to wake-up the system, and puts the mote into the power-saving sleep mode.

The wireless reprogramming module is activated only on user request when the software running of the Tmote Sky microprocessor needs to be replaced. In this case the on board radio of Tmote Sky is turned on to receive the new program image which is injected into the network from the network sink.

2.2.1.2 Scheduler

The scheduler is the main application running on the Tmote Sky and is responsible for the measurement management. It includes a global scheduler for scheduling measurements tasks that need to be executed on the sensor nodes. Each task contains the global timestamp of its first execution time, the period and the number of times it must be repeated. This makes it possible to define points in time on a global time scale when measurement must be acquired.

The result of the task execution is a sensor reading or data computed from a sequence of sensor readings, which needs to be sent to the base station. Before transmitting the data the scheduler tags it with a timestamp indicating the corresponding acquisition time.

The present prototype allows collecting information about the structural condition based on acceleration measurements. Moreover, it is possible to receive information about the internal state of each mote (e.g. battery voltage) which describes the current condition of the monitoring system itself.

The scheduler implements several commands end events which form an interface to the main monitoring application running on the control centre (see section 2.2.3). These commands and events and their description are given in Table 2 and Table 3.

Scheduler Commands	Description
Measurement Management	
registerJob	Initiates a defined measurement task with a given starting time, repetition period and number of repetitions. Additionally, parameters for the measurement processing algorithm can be passed.
cancelJob	Cancels a defined measurement task.
cancelAllJobs	Cancels all active measurements tasks.
Application Management	
resetScheduler	Initiates the hardware reset of the whole WSN node.
resetWiseNode	Initiates the hardware reset of the WiseNode.
ping	Initiates the pong-event system response. Used to build a ping-pong mechanism, which is used to asses if the desired node is reachable.
reprogrammingMode	Puts the mote into the wireless reprogramming mode (2.2.1.6).

Table 2: Commands of the scheduler application layer interface.

Scheduler Events	Description
schedulerBooted	Signalled from the booting routine after reset of the node.
WiseNodeReset	Signalled after hardware reset of the WiseNode. It can occur as a result of the reset WiseNode command, or when the automatic recovery mechanism of the HCI module is triggered (2.2.1.5).
pong	Signalled after reception of the ping command. Additionally, it can contain information describing the current status of the main application.
globalTimeOverflow	Signalled after detection of a global time reset provided by the WiseNode. This can be caused by spontaneous reset of the WiseNode, or when anew global time is set by the sink.
reprogrammingModeAck	Signalled as a confirmation after reception of a reprogrammingMode command. It is very important to have this acknowledgement mechanism in order to be sure that all network nodes are entered the wireless reprogramming mode.
wdtFired	Signalled when a custom software watchdog timer fires.

Table 3: Events of the scheduler application layer interface.

2.2.1.3 Time Synchronization

Global time synchronization of the network nodes is a key part of the distributed monitoring system. The scheduler component needs information about the current global network time in order to schedule the measurements appropriately.

Keeping track of the global network time is a feature of WiseNET. It is achieved by propagating the current time information from the network sink to the network nodes through converge-cast data traffic or network flood mechanism (see appendix A.1.1.3).

The information about the current network global time is provided to the scheduler module by the HCI module when the *getGlobalTime* command is issued (see Figure 32). The HCI module uses the *getTime* command of the HCI interface (see Table 7) to get the current network time information from the WiseNode.

2.2.1.4 In-mote Data Processing

The most energy consuming component of the system is the radio module. Therefore the radio usage time should be reduced as much as possible. On the one hand this calls for efficient radio usage, which is the task of the advanced wireless MAC protocol (see section 1.2.6, point 5). At the same time it is crucial to reduce the amount of data packets that have to be transmitted by the radio.

Therefore, instead of sending all the raw data to sink for the centralized processing and analysis, the data processing is moved to the motes. These two principles of centralized and decentralized data processing are illustrated in Figure 23.

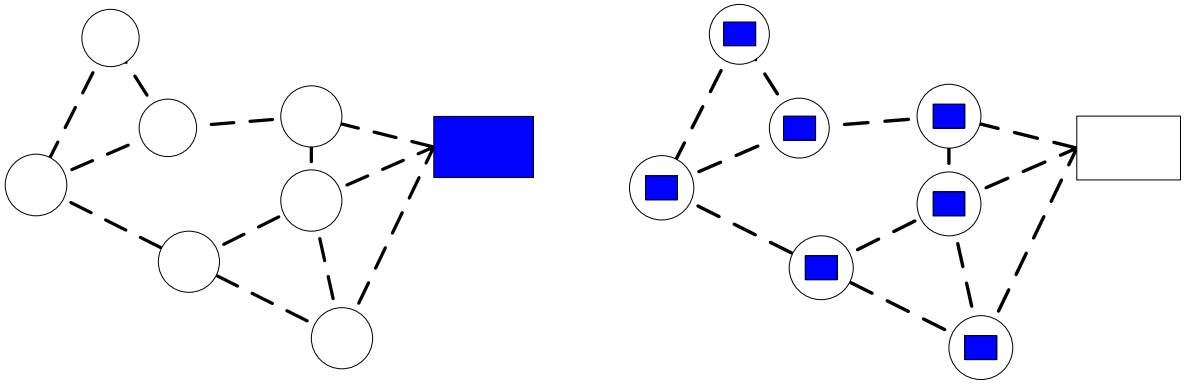

Figure 23: Centralized processing principle (left) and distributed processing (right).

By performing the data processing directly on the motes, the important and significant parameters are extracted from the measurement signal. Only this information is then communicated to the sink instead of sending the whole measurement signal. By using this approach, a drastic data reduction can be realized. In the application of cable tension monitoring this results in a data reduction factor of around 1000. The cable tension force is estimated from the natural frequency of the cable, therefore the only important parameter of the cable vibration measurement is the natural frequency of the signal. This simplified procedure is illustrated in Figure 24.

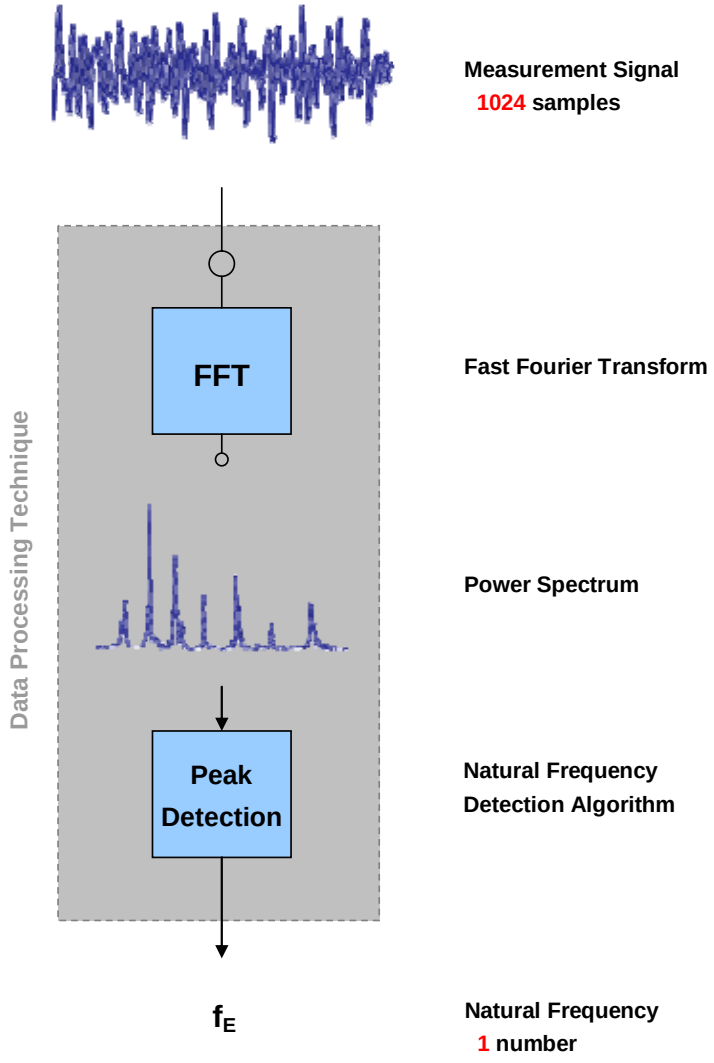

Figure 24: Data reduction (1024 to 1) is achieved by determining one of the cable natural frequencies from the vibration measurement signal.

After acquiring 1024 samples of the measurement signal, the data block is forwarded to the data processing module. Within the data processing module the signal is first transferred from the time domain to the frequency domain by a Fast Fourier Transform (FFT) algorithm. The resulting amplitude spectrum in the frequency domain reveals the natural frequencies of the cable vibration signal, as it is shown in Figure 25. The natural frequencies are represented by the peaks in the amplitude spectrum.

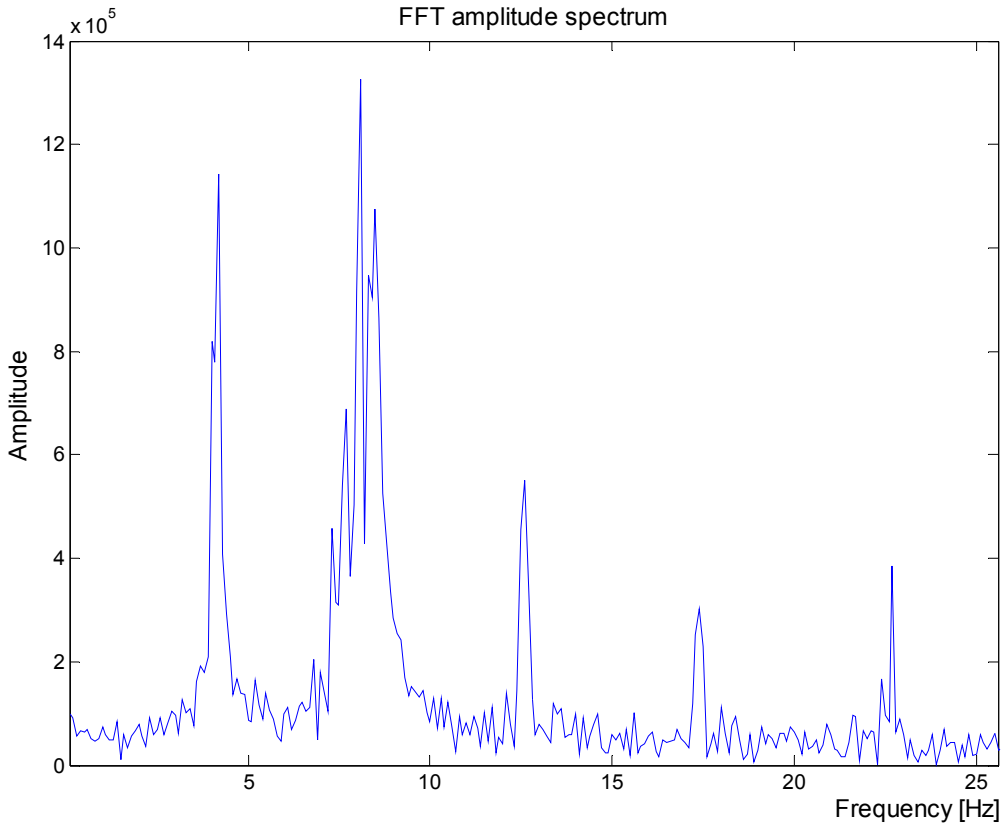

Figure 25: A typical amplitude spectrum of the acceleration signal acquired from a real bridge cable-stay. Spectral peaks correspond to the natural frequencies.

The next stage in the data processing is the detection of natural frequency peaks in the amplitude spectrum. After the peaks have been identified, a single or a couple of natural frequencies are selected according to the predefined criteria to be sent to the network sink within a single data packet.

This method of determining the natural frequencies of the signal is very efficient in data reduction, since the initial amount of data (1024 samples) can be reduced to a single number describing the cable tension. However, it in fact presents a challenge to implement it on the small computing resources of the microcontroller system (10kB RAM, 48kB ROM, 8MHz). This imposes the development of efficient solutions with the following features:

1. **small memory footprint (RAM)**

   The challenge arises from the limited RAM resource which totals 10kB for the whole operating system and the application. Since only the measurement signal takes 2kB of memory, the requirement is that the whole data processing has to be “in-place”. This means that the output of the data processing replaces the corresponding input data and therefore does not require additional storage.

2. **small program memory footprint**

   Another limitation is the program memory size which is 48kB used to implement the whole operating system and the application. This calls for an efficient implementation of the proposed data processing algorithms with regards to the used program memory space.

3. **fast execution time**

   The microcontroller central processing unit (CPU) operates on the internal 8MHz clock,. Therefore the proposed algorithms should be optimised in terms of execution time. Moreover, the execution time directly influences the duty-cycle of the system, i.e. the power consumption.

Fast Fourier Transform Implementation

Fast Fourier transform (FFT) is an efficient algorithm to compute the discrete Fourier transform (DFT). There are many existing FFT algorithms, but by far the most common is the Cooley-Tukey algorithm, and this is the one we use in our implementation as well.

The FFT uses a floating point arithmetic which is inappropriate for a microcontroller CPU since it has no floating point unit implemented in hardware. Therefore, all the floating point operations have to be emulated in software with integer operations. This is not very efficient in terms of program memory usage and execution speed, since the emulation of a single floating point operation requires several integer operations.

For example, a single operation of integer addition takes 1 clock cycle of the MSP430F1611 microcontroller CPU. An integer multiplication operation is as well emulated in software by using integer bit-shifting and adding operations and it takes 77 clock cycles[5] [44]. A special feature of MSP430F1611 microcontroller is that it has an integer multiplier implemented as a dedicated hardware module. Hence, the hardware multiplier allows completion of integer multiplication operation in a single clock cycle[6].

Floating point arithmetic on MSP430F1611 takes about 4 times more clock cycles than the integer arithmetic[7] [45].

[5] when using the standard C library for MSP430 family

[6] the actual amount of the required clock cycles is 3; 1st and 2nd clock cycle are used for loading the 1st and 2nd operand, and the 3rd clock cycle is when the multiplication is executed.

[7] The emulation requires four 16-bit integer multiplications to multiply the 23-bit fractions of the single-precision 32-bit floating point standard representation (IEEE 754).

Operation	Number of clock cycles
Integer addition	1
Integer multiplication (SW emulation)	77
Integer multiplication (HW multiplier)	3
Floating point addition	158
Floating point multiplication	332

Table 4: Number of clock cycles needed for some of the arithmetic operations.

Another downside of the floating point FFT implementation for the microcontroller system is that floating point data needs 32 bits of memory[8], which is twice as much needed for the 16-bit Integer data type. Hence, just the memory required for the storage and processing of the measurement signal would take 40% instead of 20% of total amount of RAM.

Since the acquired signal after the 16-bit ADC conversion is of an integer data type, it is reasonable to perform the data analysis by using just integer operations instead of floating point operations. The memory footprint of such an analysis is much smaller (both RAM and ROM), and the execution time is significantly lower. As a consequence, the power consumption of the data processing is reduced.

The main idea to achieve such an integer-only implementation of the FFT is to replace all floating point operations of the standard FFT implementation with integer operations. One way to achieve this is to approximate the floating point multiplication by a integer multiplication. Two 16-bit integer operands are multiplied to get the 32-bit result, but then only the higher 16 bits are taken as the final result and the lower 16 bits are discarded.

It might seem that such an approximation mechanism would result in big errors in the result. In order to see the effective error caused by this rounding, the results of the integer approximation were compared with the results of the standard floating point FFT, as it is illustrated in Figure 26.

[8] In single-precision representation (IEEE 754)

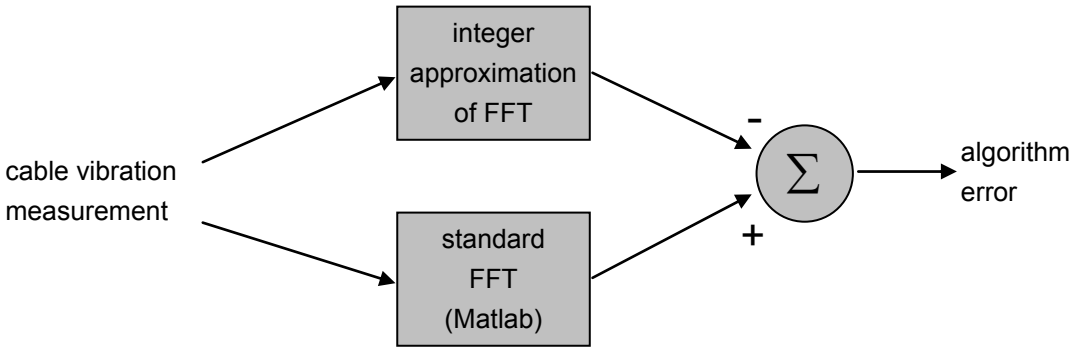

Figure 26: Analysis of the algorithm error.

The direct comparison of the two results is shown in Figure 27 and Figure 28, where the amplitude spectrums of the two FFT implementations are plotted. It is very important to mention that these amplitude spectrums do not present an absolute value of the spectrum:

$$|S_i| = \sqrt{\left(\mathrm{Re}\{S_i\}\right)^2 + \left(\mathrm{Im}\{S_i\}\right)^2}\,, \tag{9}$$

but instead, they equal the sum of an absolute value of the real part and an absolute value of the imaginary part of the signal spectrum:

$$|S_i| = |\mathrm{Re}\{S_i\}| + |\mathrm{Im}\{S_i\}|, \tag{10}$$

for every point of the spectrum S_i , which is very easy to calculate.

This approach does not calculate the amplitude spectrum, but is a very simple solution for microcontroller systems because of the simplicity of the calculation. However, it is important for this method to work properly that the natural frequency peaks have amplitudes which are more than about two times higher than the noise floor what is in general true for the cable stays.

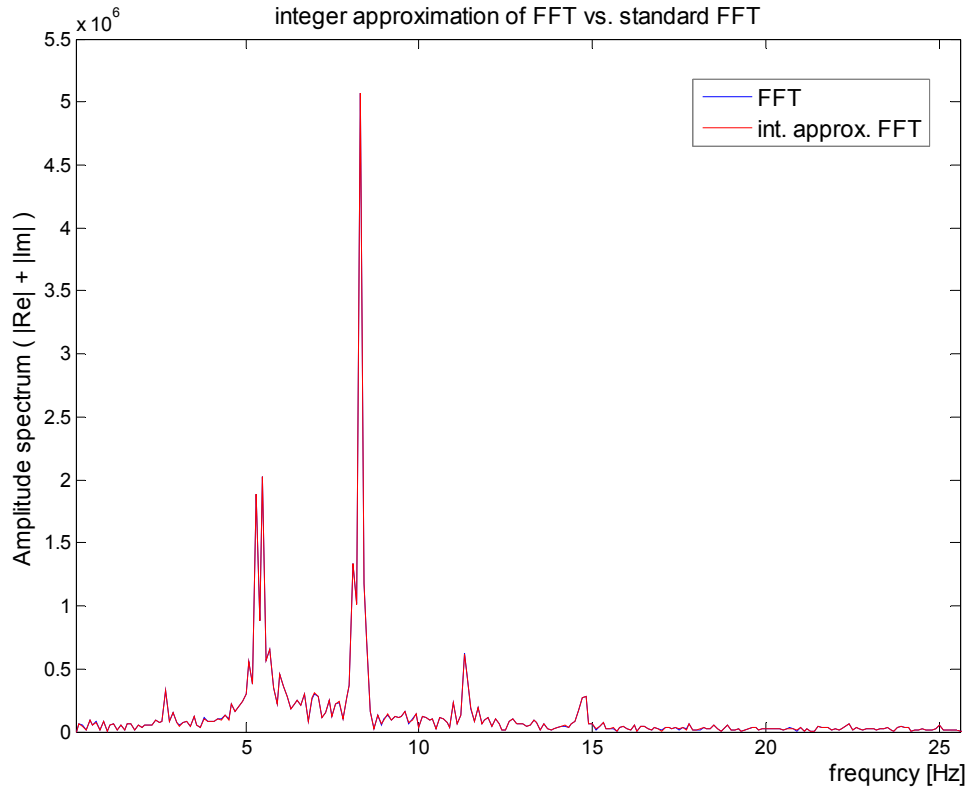

Figure 27: The results of the integer approximation (red) compared to the floating point FFT (blue). The input for the FFT analysis is an acceleration signal from a cable stay.

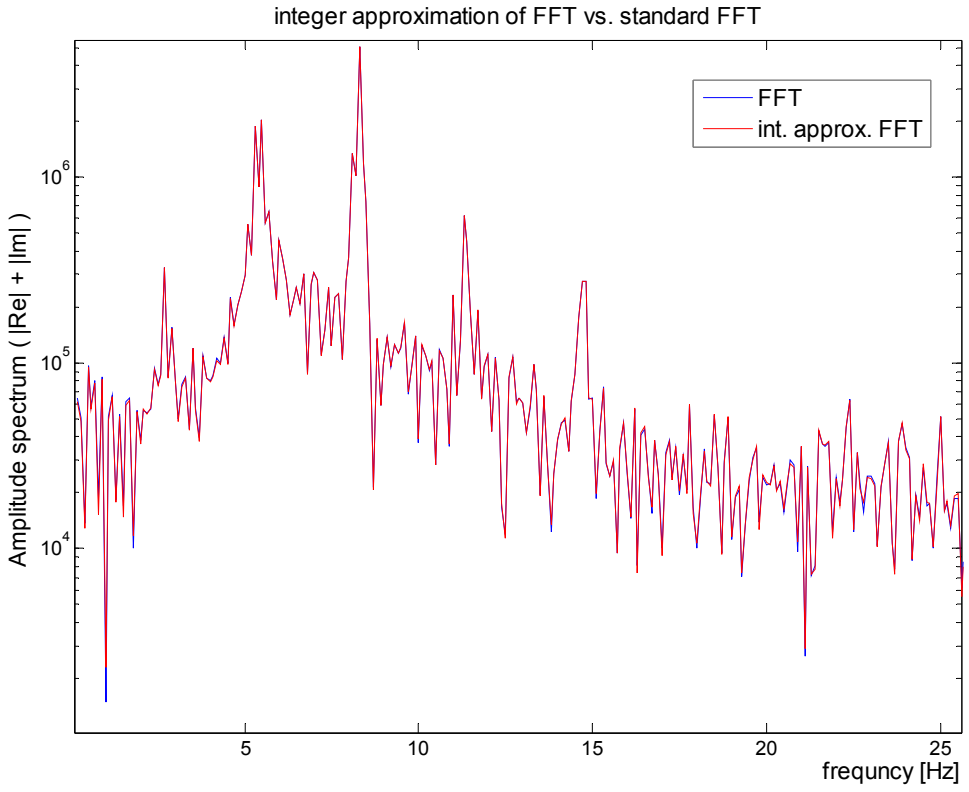

Figure 28: Logarithmic scale of the results of the integer approximation (red) compared to the floating point FFT (blue).

The integer approximation of the FFT achieves a very good result matching with the floating point FFT. In fact, in Figure 28 it is almost impossible to perceive error. Therefore, the results and the difference are plotted together in Figure 29 in the logarithmic scale in order to better see the approximation error. The relative error[9] is expected to be smaller in the area of the amplitude peaks, which is very good because this is in fact the important area for the analysis of natural frequencies.

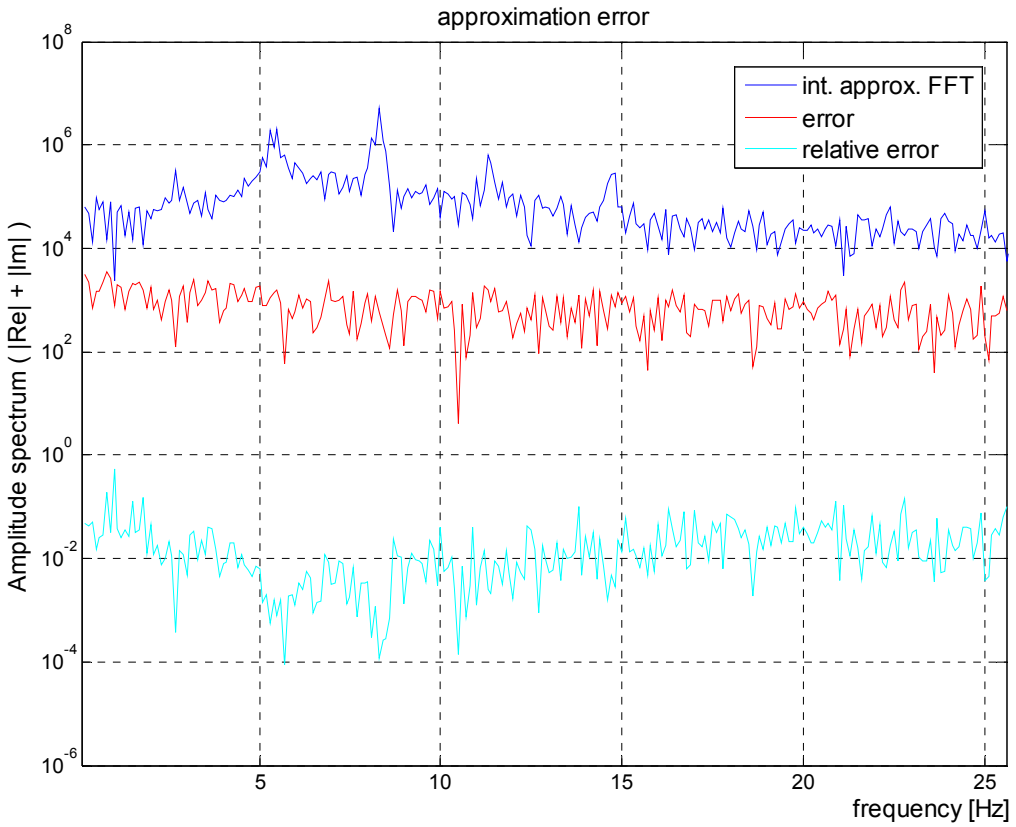

Figure 29: Integer approximation of FFT and the approximation error.

The root mean square error (RMSE)[10] in the presented example was 919, normalized RMSE (NRMSE) [11] was 0.018%, and RMSE normalized by the mean value of the observed signal (CV(RMSE))[12] was 1.10%.

[9] absolute error divided by the FFT value

[10] $RMSE = \sqrt{\frac{\sum_n (error)^2}{n}}$

[11] RMSE normalized by the range of the result, $NRMSE = \frac{RMSE}{x_{\max} - x_{\min}}$

[12] $CV(RMSE) = \frac{RMSE}{\bar{x}}$

The achieved performance of different implementations of the FFT algorithm regarding the execution time is presented in Table 5 and plotted on Figure 30. These times incorporate the time required to calculate the absolute amplitude spectrum from the real and imaginary part of the signal spectrum, which takes about 17ms for a 1024-samples version.

	Integer approximation of FFT		floating point FFT
Number of samples	**Computation time (HW multiplier) [ms]**	**Computation time (no HW multiplier) [ms]**	**Computation time [ms]**
128	58	147	531
256	125	338	1195
512	271	749	2654
1024	581	1646	5840

Table 5: Execution time of the integer approximation of the FFT algorithm compared to the standard floating point FFT for different input signal length.

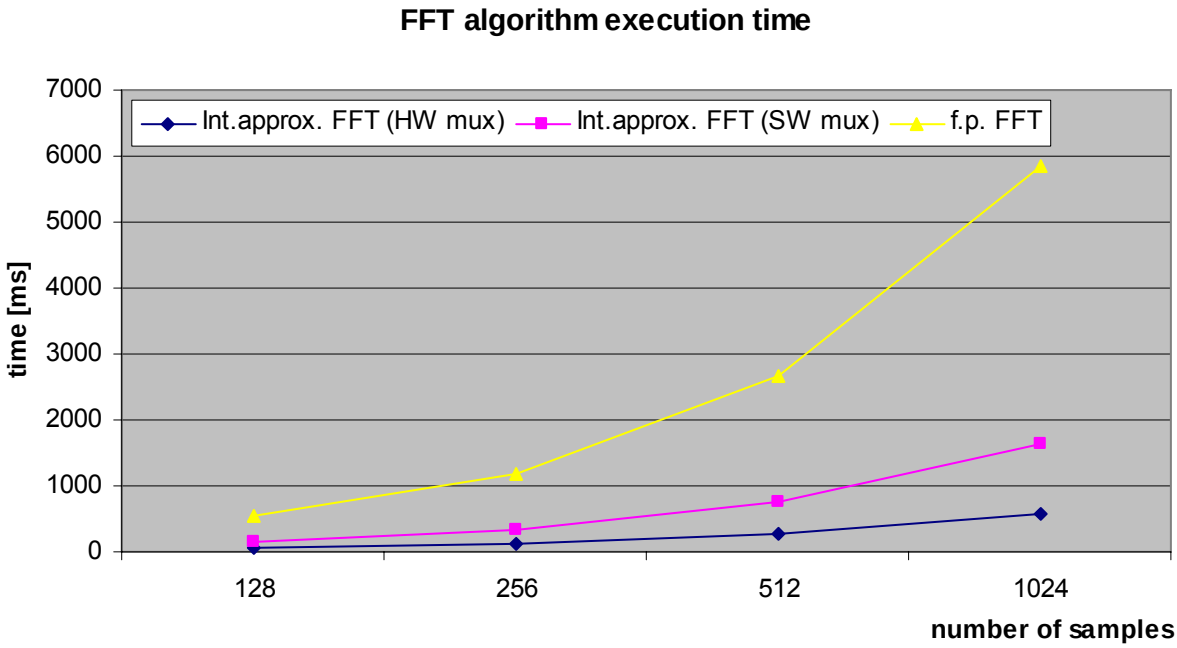

Figure 30: Execution time of implemented FFT algorithms.

It can be observed that the integer approximation of the FFT algorithm significantly outperforms the standard floating point FFT in terms of execution speed. As well, a significant decrease in the computation time of the integer approximation of FFT is achieved by using the hardware multiplication module of the MSP430F1611 microprocessor. From the values in Table 5 it can be concluded that the total computation takes about 3 times

longer when not using hardware multiplier, and about 10 times longer for the floating point implementation.

The execution time which is under 0.6 seconds is very much influenced by the requirement that the whole algorithm implementation should consist of small tasks which are not longer than 1 millisecond in order to maintain a fast system response time. Such an implementation with small tasks actually takes about 2 times longer to execute than the one without this feature requirement.

As seen before, the expected RMSE is around 1.10% which is sufficient for our application of natural frequency estimation, especially because the relative error is significantly smaller in the peak areas (see Figure 29). The execution time of about 0.6 second does not influence significantly the overall power consumption of the system. The impact of the FFT processing on the overall power consumption is discussed in section 3.4.2.2.

Natural Frequency Detection

As described above an FFT algorithm transforms the signal from time to frequency domain, but the amount of data remains unchanged. However, moving to the frequency domain is needed to reveal the natural frequencies in the spectrum. The final step in the data reduction process is to identify the natural frequencies by picking the corresponding peaks in the amplitude spectrum.

In Figure 31 a typical acceleration signal spectrum of one of the cables stays of the Stork Bridge in Winterthur is plotted. As it can be seen, the spectral peaks corresponding to the natural frequencies are well separated.

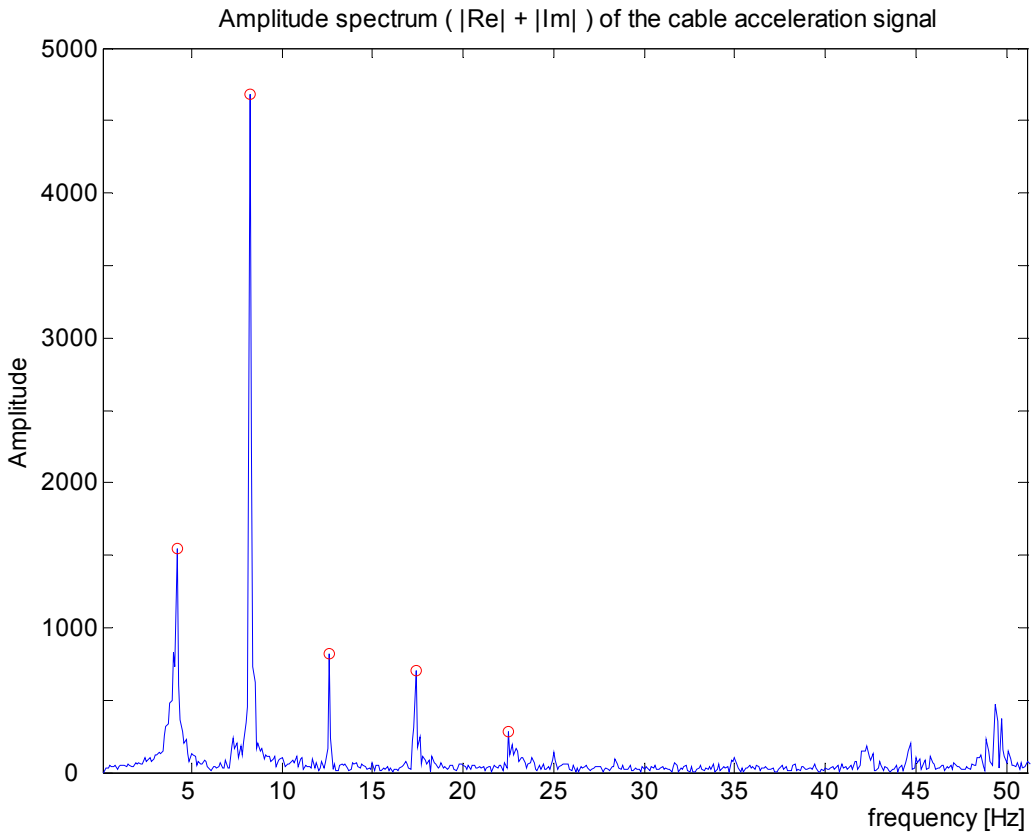

Figure 31: Detected peaks in the FFT spectrum of the acceleration signal are corresponding to the natural frequencies of one of the cable stays from the Stork Bridge in Winterthur.

Although the spectral peaks of the cable vibration are easy to see, an automatic peak detection is not trivial, especially if it has to be achieved on a resource constrained microcontroller system. Since the computing requirements of known peak detection algorithms exceed the resources of a microcontroller system a custom spectral peak detection algorithm has been developed. This algorithm is tailored to natural frequency detection of cable stays, based on the fact that the natural frequency peaks are evenly spaced. The main requirements on the algorithm were fast execution time and small memory footprint. An algorithm meeting these requirements has been developed and implemented. Table 6 provides information about its memory requirements and execution time. Execution time totals the time required to detect the first 4 natural frequencies.

Execution time	~3.2 ms
Used RAM	23 bytes
Used ROM	3138 bytes
Longest task	< 1 ms

Table 6: Features of the natural frequency detection algorithm (1024 samples version).

The implemented algorithm performs the peak detection in about 3.2 ms for 1024 input samples. The longest task runs in less than 1 ms. Considerations about the algorithm accuracy are given in section 3.5.

2.2.1.5 Host Controller Interface Stack

The HCI module provides the scheduler module with an interface to WiseNode (see Figure 22). It is the software component where the Host Controller Interface (HCI, see A.2.2) communication protocol stack is implemented.

The HCI protocol consists of commands sent from the host to WiseNode, and events sent from WiseNode to the host. Table 7 presents a brief summary of the commands and events of the HCI protocol that are supported by the current version of the HCI stack. More detailed information about the HCI commands and events and HCI protocol in general can be found in [46].

HCI Commands	Description
Network Group	
Network Flood	Starts a wave of broadcast transmissions used to build a tree topology rooted at the sink.
Host Flood	Starts a wave of broadcasts to disseminate a new application command to all network nodes.
Change Sampling Period	Changes the length of preamble sampling period.
Change Transmit Power	Changes the radio transmission power.
Change Receive Threshold	Changes the radio receive threshold.
Change Accept Route RSSI Threshold	Changes the threshold above which the routes are accepted during the routing floods.
Change Medium Reservation Window	Changes the length of the medium reservation window for the collision avoidance mechanism.
Disable Flood Forwarding	Used to avoid useless broadcast retransmissions when all network nodes are in reach of another.
Set Route To Sink	Sets the next hop (parent) towards the sink; used to manually change the routing table of a node.
Send Statistics	Starts a periodic transmission of statistics messages.
Reset	Resets the main application running on the WiseNodes, statistics or route to default one.
MAC Group	
Send Data	Starts transmission of data packet from node to sink.
Set MAC Address	Changes the MAC address of the node.
Miscellaneous Group	
Set Time	Sets the global time of the network.

HCI Commands	Description
Get Time	Reads the current global time of the network.
Get Configuration	Reads the WiseNode's configuration data.
Set Frequency	Changes the frequency of the WiseNode's radio.
HCI Events	
Received Data	Signalled upon the reception of the data packet.
Received Host Command	Signalled upon the reception of the application command disseminated from the sink.
Received Statistics	Signalled upon the reception of the statistics packet.

Table 7: Summary of supported HCI protocol commands and events.

Although all HCI commands and events given in Table 7 are supported, only a few of them are used during a normal network operation. On the network nodes, the HCI module uses just a couple of them to build an interface towards the main Scheduler module as shown in Figure 32.

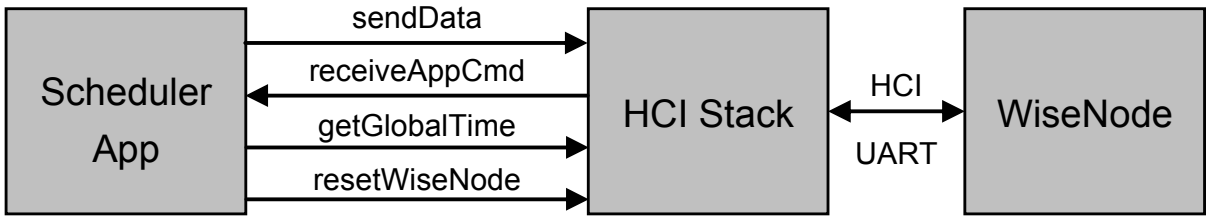

Figure 32: Interface of the HCI module provided to the main Scheduler application.

The interface provided to the main Scheduler application running on the network nodes consists of 3 commands and 1 event. The provided commands are used to send the data to the network sink, get the information about the global network time and to hard-reset the attached WiseNode. The only event that is implemented on the network nodes is signalled when an application management command is received by the means of host flood from the sink (see appendix A.1.1.4). Application management commands are used to send commands to the Scheduler application, e.g. command to schedule a measurement task.

The situation is different on the sink, as the Tmote Sky host board connected to the root WiseNode runs a different main application. Namely, the measurement functionality of the node is completely disabled. In fact, the network sink is not even equipped with the sensing modules, as it is not required. Therefore, the main application running on the sink host board is not the scheduler application, but a dedicated application required to establish the communication between the base station and the WiseNode sink.

The interface provided to the main application running on the sink node is illustrated in Figure 33, and it consists of 5 commands and 2 events. The provided commands are used to disseminate the application management commands into the network using the host flood HCI command, to disseminate the network management commands (Table 7, Network Group), to set and get the global time of the network, and to hard-reset the attached WiseNode. Events are signalled upon the reception of the data or statistics packets.

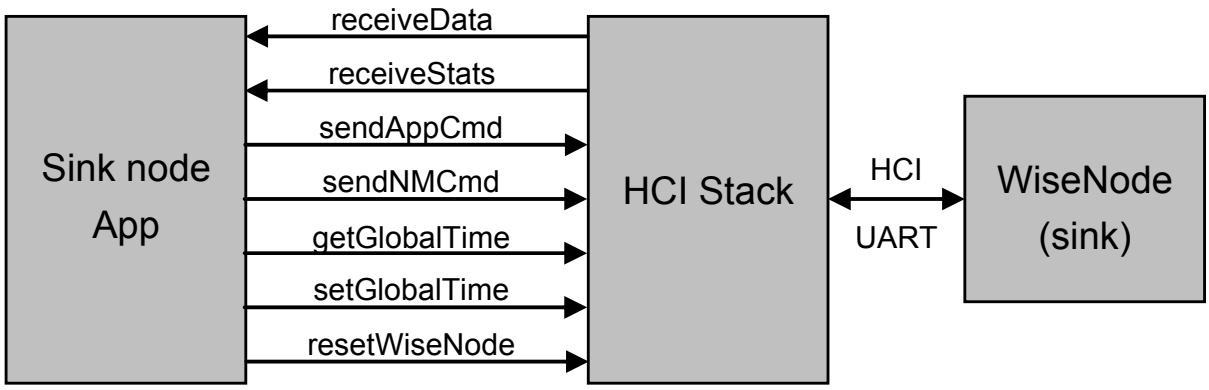

Figure 33: Interface of the HCI module provided to the main application on the network sink.

2.2.1.6 Wireless Reprogramming Module

The wireless reprogramming module is where the important functionality of wireless reprogramming of the Tmote Sky boards is implemented (see Figure 22). It is used to reprogram the network nodes with new versions of application firmware, which is very important in the situations where new features or new processing algorithms need to be implemented or when new improved software versions in terms of system efficiency, stability and robustness are available. However, in the current implementation it is only possible to wirelessly reprogram Tmote Sky boards. There is no such functionality to wirelessly upgrade the WiseNode firmware. Although this functionality is available at CSEM, it has not been included in the project firmware.

Wireless reprogramming is an important feature that enables uploading improved firmware versions to every network node without needing any cables and physical access. There are many scenarios where once the network has been installed the physical access to the nodes would require a lot of effort. Even more, sometimes it is not feasible since the WSN nodes are embedded into the structure, e.g. during the construction phase. Therefore, the possibility to reprogram the whole network once installed is very important and it definitely allows decreasing maintenance costs of WSN monitoring systems.

To achieve this functionality the module uses the radio chip and antenna of Tmote Sky. In normal network operation when the low-power monitoring application is running on the motes and the wireless communication network provided by the WiseNodes is used to establish the communication to the network sink, the radio circuitry and the dedicated software modules are in an inactive state. The normal network operation is illustrated in Figure 34 where the WiseNode module is used to establish a wireless network with tree topology rooted at the network sink, indicated by blue arrows.

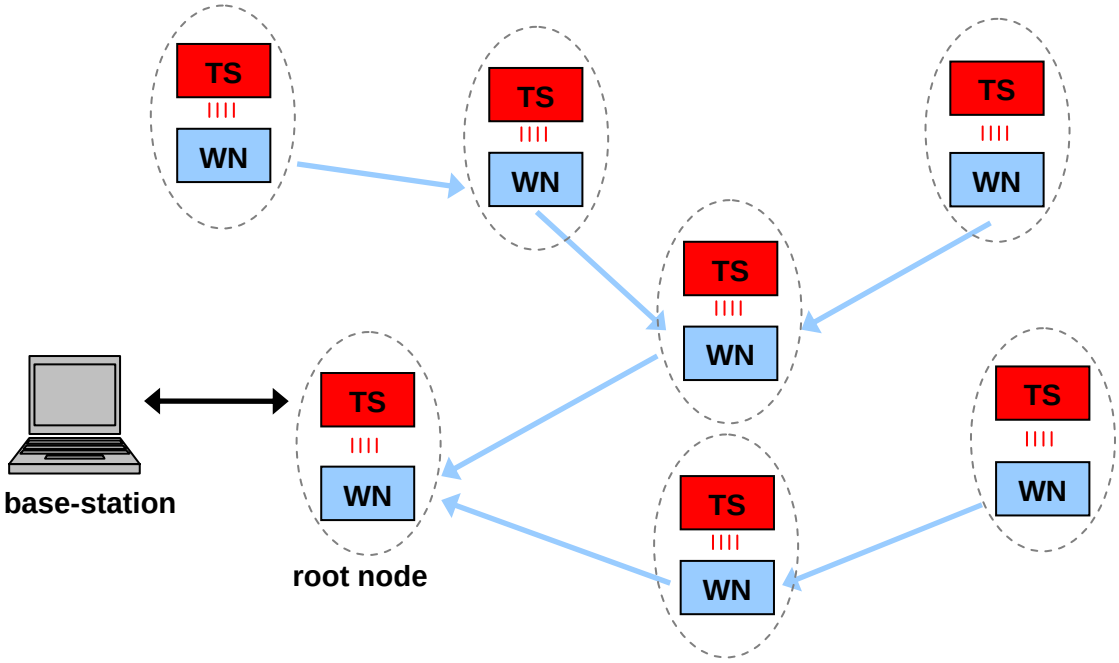

Figure 34: Normal operation: WiseNodes (WN) communicate the data originating from Tmote Sky (TS) to the base station.

The wireless reprogramming module is activated only on a special user request when the software running on a Tmote Sky board needs to be upgraded or reprogrammed. Upon such a request, the module switches Tmote Sky into the wireless reprogramming mode which turns on the Tmote Sky radio and a new multi-hop wireless network is established. This is illustrated in Figure 35 and indicated by red arrows.

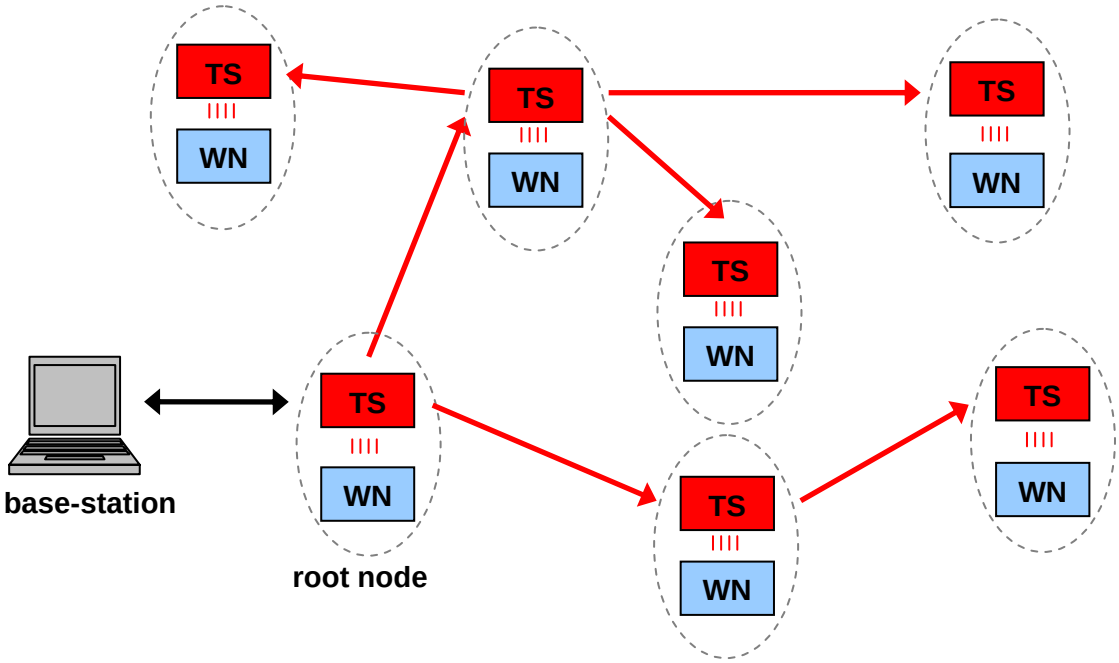

Figure 35: Wireless reprogramming mode: a separate multi-hop network is established using the Tmote Sky (TS) radio to propagate new application firmware to every mote.

The firmware upgrade functionality of the wireless reprogramming module additionally enables storing multiple firmware versions on motes, i.e. different program images or different versions. This enables network nodes to easily switch between different programs without the need of injecting a new program image every time when the network needs to switch into another mode. This increases the usability of the installed WSN, since different operating modes and functionality of the network can easily achieved. In fact, the reprogramming mode described here is also achieved by switching the Tmote Sky motes into a special preinstalled program image where the reprogramming functionality is implemented.

Wireless reprogramming feature is implemented by using the existing TinyOS Deluge wireless reprogramming module [47]. The Deluge module uses the Tmote Sky's on-board 1MB flash memory chip to store different program images. Since the maximum size of the program image is 48kB, up to 20 different program images can be stored on the WSN nodes.

2.2.2 Software Base-station

Debian/Linux was chosen as operating system and PostgreSQL as database software. Secure shell (ssh) is used for the secure transmission of the data and the remote login. The access to the WSN, database and Internet connection is provided by software

modules written in Java. Watchdog scripts were written to survey the WSN and the Internet connection. Additionally, a Dyndns client provides the access to the base station by an official domain name and circumvents the problem of periodically changing dynamic IP addresses assigned by the telecommunication company. Further software packages running on the base station are a NTP-client (Network Time Protocol) to get the current time from time servers in the Internet and a firewall to only allow connections from a known IP range.

2.2.3 Software Control Centre

An operator console enables access to the installed WSN monitoring system, both locally on site and remotely at the control centre. In this section the software components of such an operator console are described, mainly the developed graphical user interface (GUI).

The GUI presents a front-end of the whole monitoring system as it enables:

- the display of the incoming results
- incoming data visualisation
- display of the feedback information about the current network status
- network management and setup
- measurement scheduling and management
- reconfiguration of the system by switching the nodes to a different program image
- injecting a new program image into the network

Besides the GUI operator console, the topic of this section will cover the database of the remote control centre as well as the automatic data visualisation on a web-page.

2.2.3.1 Data Display and Visualisation Tools

Data display and visualization part of the GUI operator console enables the operator to easily observe the incoming analysis results and status reports from the installed WSN.

Figure 36 presents a screenshot of the display component for the incoming results. The results of the measurement analysis from the WSN are parsed and displayed in a user friendly form. The operator can observe the origin of each message, the type and result of the measurement and analysis, and the time when it was performed. A detailed screenshot of this tool is given in Figure 117 in appendix B.

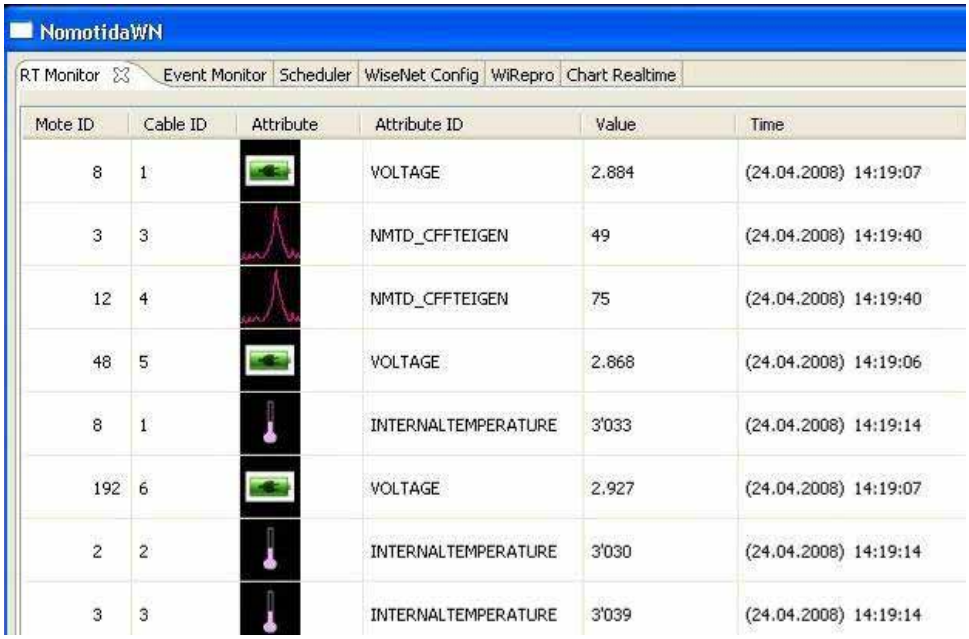

Figure 36: Screenshot of the display tool for the incoming results.

Figure 37 presents a screenshot of the display component for the incoming results visualization in form of a graphical plot. The results are plotted in real-time as they arrive to the control centre with the x-axis corresponding to the time when the measurement was taken. A graphical plot of the detected natural frequencies of the bridge stay cables is given in Figure 37, where each line represents the detected frequency of a specific cable.

Some of the plot options that the operator can set are: the plot of the results from just one particular cable, moving average of the results, manually setting the x-axis and y-axis range, zooming in and out at the desired area. A detailed screenshot of this tool is given in Figure 118 in appendix B.

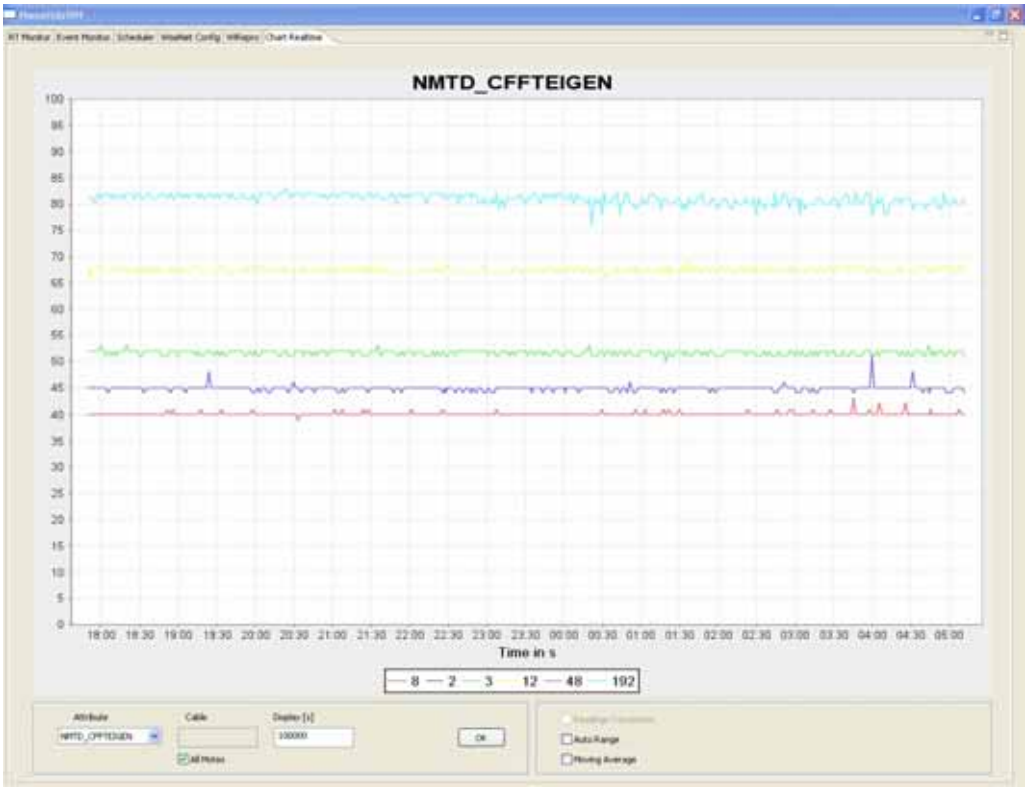

Figure 37: Screenshot of the chart tool for the incoming results.

Figure 38 presents a screenshot of the network status monitoring tool where the incoming events from the network nodes are displayed. Such an event signalling from the WSN nodes forms an important feedback mechanism in order to get the information about the current status of the monitoring system (see section 2.2.1.2). Incoming events are parsed and displayed in a user friendly form, so that the operator can easily observe the origin of each event, information about it and time when it happened. A detailed screenshot of this tool is given in Figure 120 in appendix B.

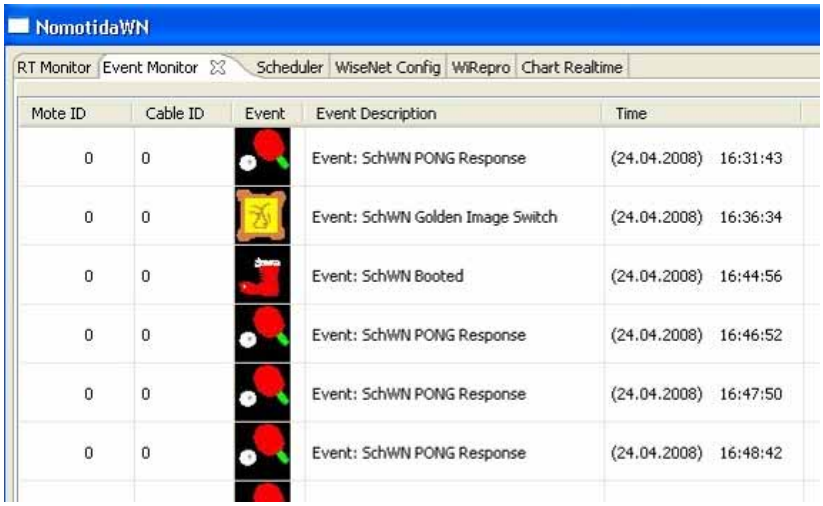

Figure 38: Screenshot of the display tool for the incoming status reports.

2.2.3.2 Measurement Management Tools

This part of the GUI operator console is dedicated to the management of measurements tasks which enables the operator to schedule new measurements, and cancel or re-schedule existing ones.

Figure 39 presents a screenshot of the measurement management tool. The operator can easily schedule different kinds of measurement and analysis, set a desired repetition period and a number of repetitions, or cancel the existing ones (see section 2.2.1.2). The measurement management GUI has 2 tables; one containing the list of all previously issued measurement management commands, and the other one with a list of all active measurement tasks. To have a better overview of the active measurements and their spacing in time, a clock-like component was implemented, with the indication of the current network time and the scheduled measurements. Different measurements are indicated by a different colour of the ticks, corresponding to the colour in the scheduler status table.

A detailed screenshot of this tool can be found in Figure 119 in appendix B.

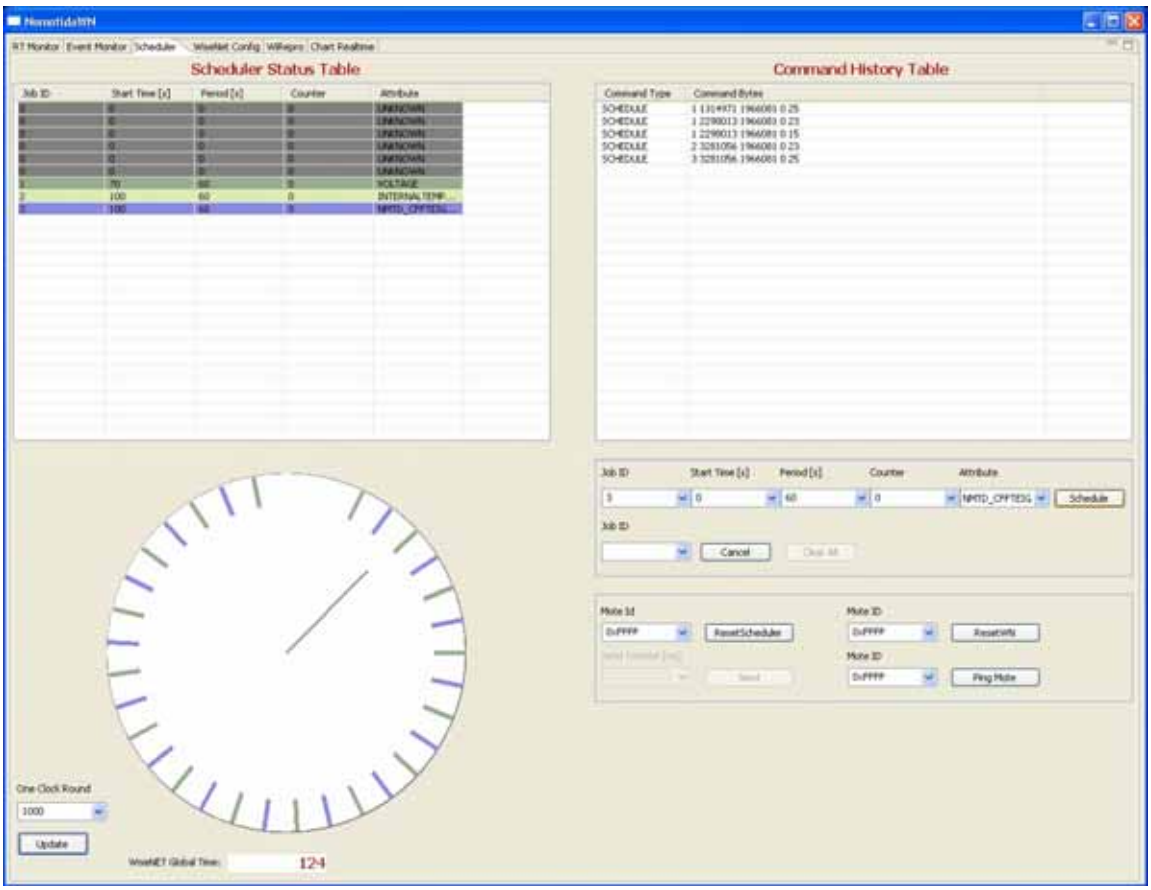

Figure 39: Screenshot of the measurement management GUI.

2.2.3.3 Network Management Tool

This part of the GUI is dedicated to the management of WiseNET parameters. Figure 40 presents a screenshot of the WiseNET management component.

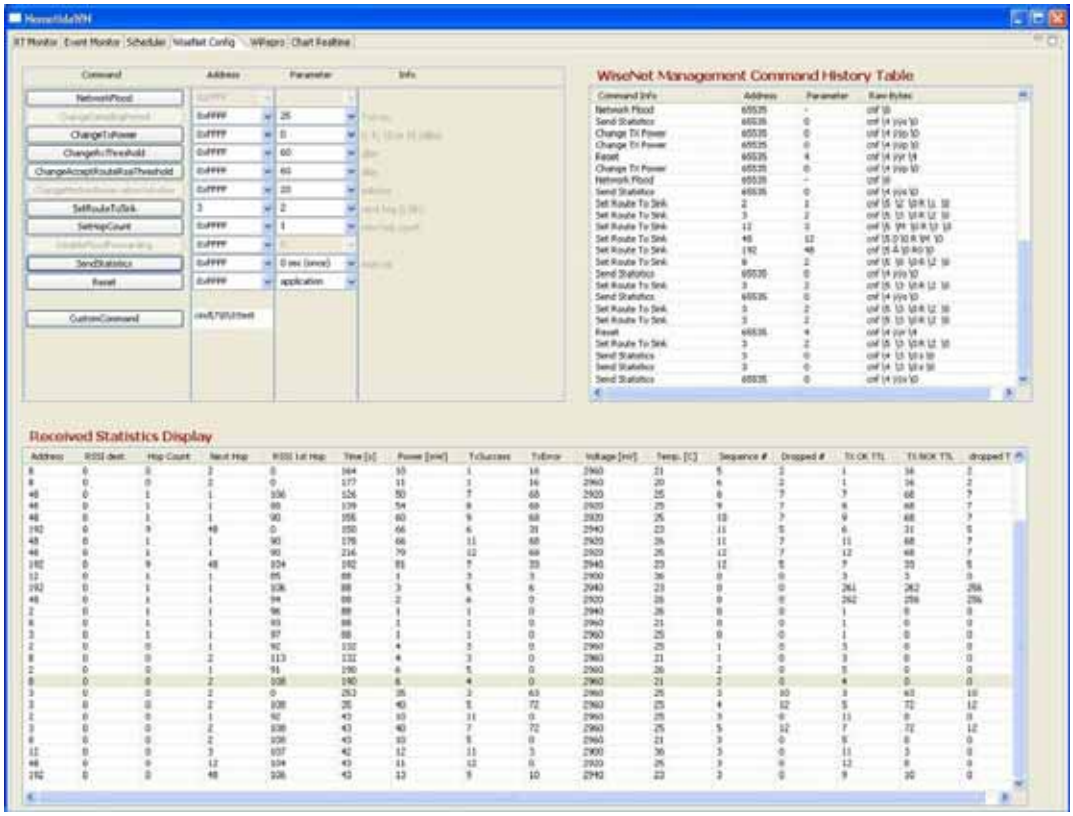

Figure 40: Screenshot of the tool for network parameters and topology management.

The network management group of HCI commands is dedicated to managing WiseNET (see Table 7). Some of the main properties of WiseNET which can be managed using this tool are:

- network topology
  - manually changing existing network topology
  - automatically building a new network topology
- radio transmit power
- radio packet receive threshold
- network sampling
  - change network sampling period
  - change the medium reservation window
- issuing WiseNode statistics

- reset
  - routing
  - statistics
  - WiseNode main application

The network management GUI features a special command history list holding the network management commands issued in the current session. As well, there is another list for displaying the incoming WiseNode statistics packets.

A detailed screenshot of this tool is given in Figure 121 in appendix B.

2.2.3.4 Wireless Reprogramming Tool

The wireless reprogramming tool of the GUI operator console enables the operator to remotely reprogram the nodes. This is accomplished by switching the nodes into a special reprogramming mode (see section 2.2.1.6), and injecting a new program image (firmware) to the nodes, or instructing them to switch to a different program image which is already stored in the flash memory module of the WSN nodes.

Figure 41 presents a screenshot of the wireless reprogramming tool, whereas a detailed screenshot is given in Figure 122 in appendix B.

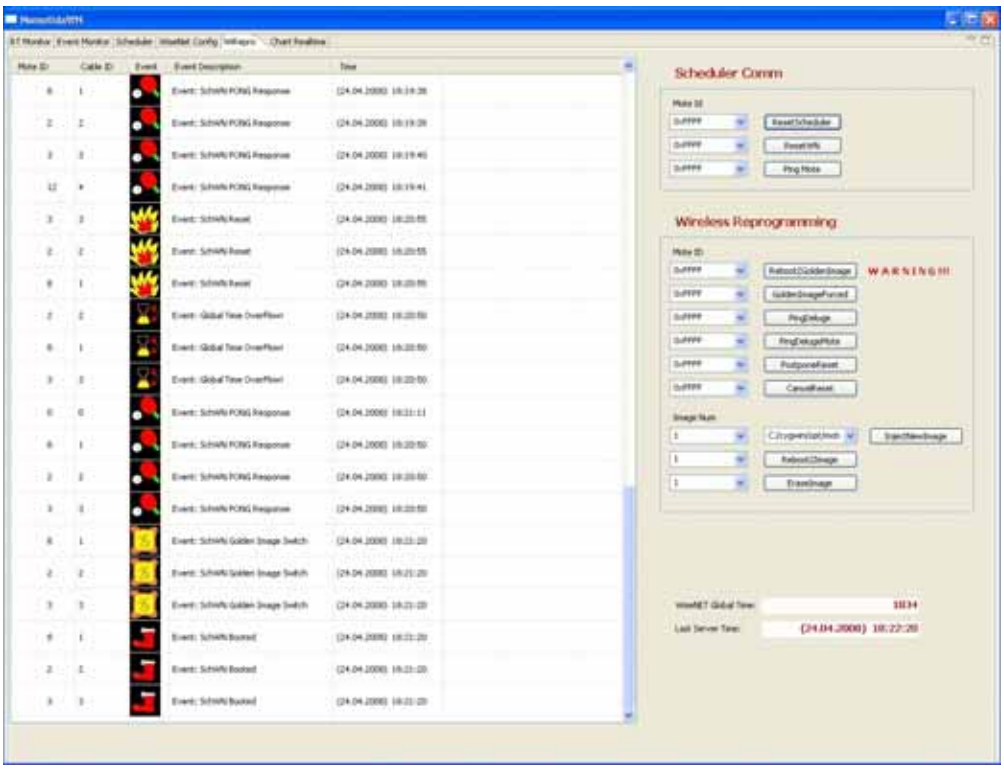

Figure 41: Screenshot of the tool for the wireless reprogramming of the network nodes.

Similar to the network status monitoring tool, this tool features the same incoming events display functionality. Important events from the network nodes are displayed in a user friendly manner, enabling the operator to observe important information about the status of the network during network reprogramming. A reprogramming command panel enables the operator to instruct the nodes to switch to a desired program image or to inject a new program image to the network nodes.

2.2.3.5 Database

All incoming data from the WSN, measurement analysis results, events and statistics, are logged at the remote control centre and stored as log files and in a PostgreSQL database. The applications access the database through JDBC (Java Database Connectivity), which provides methods for querying and updating data in a relational database. This connectivity component interfaces to a variety of databases and offers a high degree of flexibility.

All events and attributes received from the sensor network can be logged in the database. Furthermore, the data sent to the motes can also be stored in the database, i.e. commands which contain network configuration data. This makes it always possible to retrace a specific event.

2.2.3.6 Web-page

The objective of the web site is to provide background information about the ongoing monitoring activity and a tool for visualizing the data collected by the monitoring systems that are accessible from any computer connected to the internet.

The address of the web site is www.nomotida.net [*]. The most important part of the web site is the graphical visualisation of the data collected by the monitoring activities. The plots are automatically updated with data from the data base. Figure 42 shows a screenshot of the web site.

[*] The website was developed previously as a part of another project.

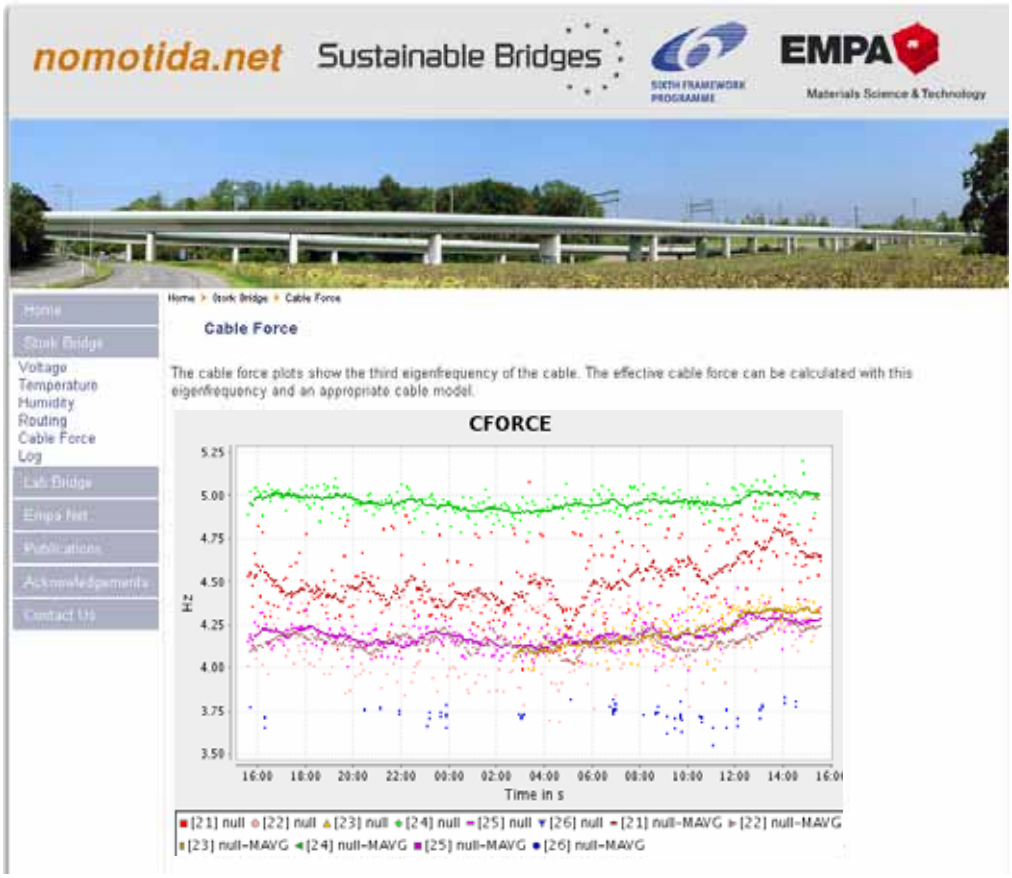

Figure 42: Screenshot of webpage where the data from the deployed WSN can be displayed.

3 System Evaluation

This chapter is dedicated to the evaluation of the developed system. System performance is evaluated from the experience gathered from short-term indoor demonstration tests with a laboratory network and from the outdoor deployment on a real cable-stayed bridge. The performance of the developed low-power wireless sensor network platform and of the developed structural monitoring application is presented and analysed.

3.1 Indoor Test

A short-term monitoring application was conceived in order to evaluate the first working version of the developed WSN system for monitoring cable natural frequencies, i.e. cable stay tension force (see section 1.4). The test was focused on the evaluation of the accuracy of the developed data processing algorithms, and the overall stability of the system.

The tests were performed on the laboratory indoor bridge at the Structural Engineering Research Laboratory at, Empa (see Figure 43). It is a full-scale pedestrian bridge with a cable supported bridge deck.

Figure 43: Full-scale pedestrian bridge where the WSN system was verified in a short-term demonstration test at the Structural Engineering Research Laboratory at Empa.

3.1.1 Laboratory Bridge Setup

For the purposes of the indoor test, a special enclosure for the hardware components was not necessary. Therefore, the Tmote Sky and WiseNode boards were simply coupled together using rubber-band and sealing tape, as shown in Figure 44. During this time a previous version of WiseNode (version v2) was used (see appendix A.3.2).

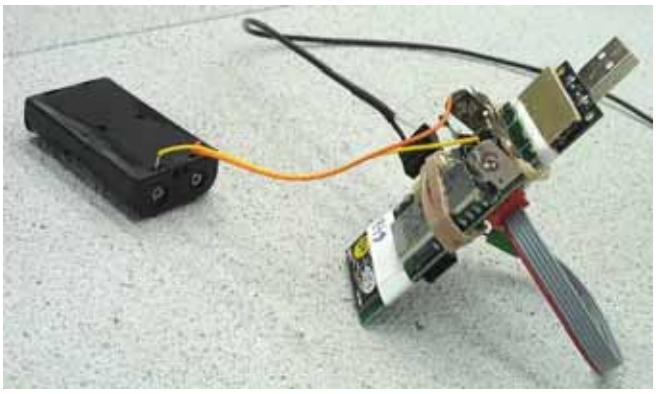

Figure 44: Tmote Sky and WiseNode_v2 coupled together with a rubber-band and sealing tape for the purposes of indoor short-term system verification.

At the time the custom sensor module (see 2.1.1.3) was still in the development process. Therefore, to obtain the acceleration signal from the bridge cable stays, a precision acceleration sensor (PCB 3711) was used instead. The sensors were mounted on the cables as shown in Figure 45 (left). The sensor signal was conditioned using a dedicated sensor amplifier (Figure 45 (right)).

Figure 45: (left) A precision acceleration sensor mounted on a cable stay of the lab bridge. (right) Dedicated sensor signal amplifier.

As it can be seen in Figure 45 (right) a. USB connection was established to every network node to be able to reprogram them. This was required in order to enable evaluation of different versions of developed natural frequency detection algorithms because at that

time a wireless reprogramming feature of the network (see section 2.2.1.6) was not yet implemented.

A WSN with a total of 6 network nodes and a sink was deployed. Each of 6 stay-cables was equipped with a precision acceleration sensor PCB3711 to acquire the acceleration signal from the cables. The implemented data processing algorithm (see section 2.2.1.4) evaluated the acquired measurement signal. The computed natural frequencies of each cable were thereafter communicated over the multi-hop WiseNET to the sink.

The network sink was connected to a laboratory PC which served as the network base-station. The data was forwarded by establishing TCP/IP connection from the base-station PC to one of the PCs in the department offices with the control operator console running (see section 2.2.3).

In order to produce sufficient vibration of the cable stays, a shaker was placed at one end of the bridge deck to excite the bridge deck vibrations (see Figure 46). The shaker was set to generate broad band random vibrations with frequency components up to 40 Hz. On the real bridges, the vibration of cables stays is sufficient to perform such analysis of natural frequencies, because the vibration is induced by wind and bridge deck vibration due to the traffic loads. These sources of vibration do not exist in the case of the laboratory bridge and had therefore to be induced artificially.

Figure 46: A shaker producing broad band random vibrations was used to excite the cable stays.

3.1.2 Short-term test

The monitoring system was installed on the laboratory bridge and left continuously running for about a week in order to evaluate the implemented algorithm and the stability of the system. The evaluation of the algorithm for the natural frequency detection resulted in several of improvements during the test, increasing the algorithm's accuracy and stability.

Figure 47 shows a typical signal spectrum acquired during the short-term indoor test. The red circles indicate the detected frequency peaks corresponding to the cable natural frequencies.

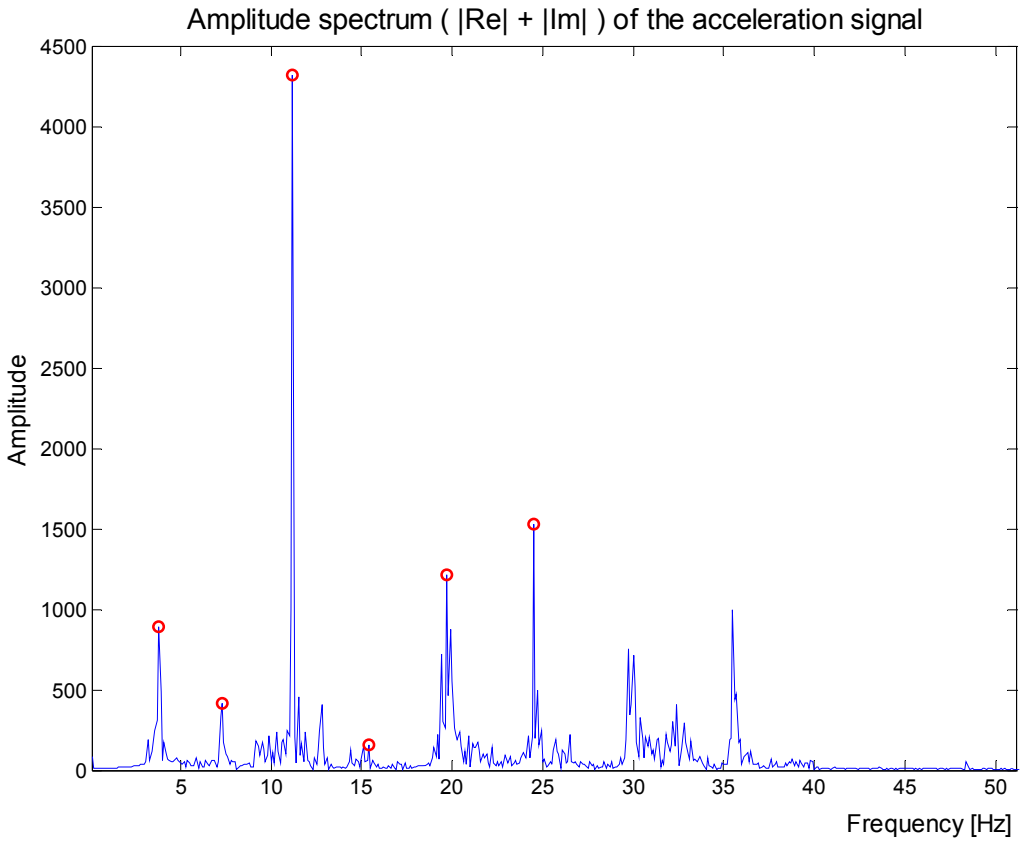

Figure 47: Typical signal spectrum and detected peaks.

3.1.3 Observing changes in cable tension

A test was performed to observe the changes in the detected natural frequencies due to the change in cable tension. In order to change the cable tension, a mass of about 500 kg was placed on the fork-lift trolley and moved across the bridge deck (see Figure 48). The load was repositioned several times during the test in order to observe changes in cable tension.

In this test the installed sensors were connected to a high precision data logger in order to obtain the complete time series of the cable acceleration signals. This data was then used as input data for the natural frequency detection algorithm to improve the detection algorithm. Simulations were conducted using Matlab.

The simulation was performed on data acquired from the test with the moving load. The detected natural frequencies of one of the bridge cables are shown in Figure 49. It is possible to identify two situations where the tension of cable C3 was significantly increased. The first situation starts just before 5 minutes after the start of the test, and the second one starts short before 20 minutes. Two spikes around 18 and 25 minutes are the result of false peak detection due to the transient response during the movement of the load and due to imperfection of the algorithm. The detected frequencies from other cables are presented in Figure 131 to Figure 136 in appendix C.

Figure 48: A mass of about 500kg was moved across the bridge deck in order to observe the changes in cable natural frequencies.

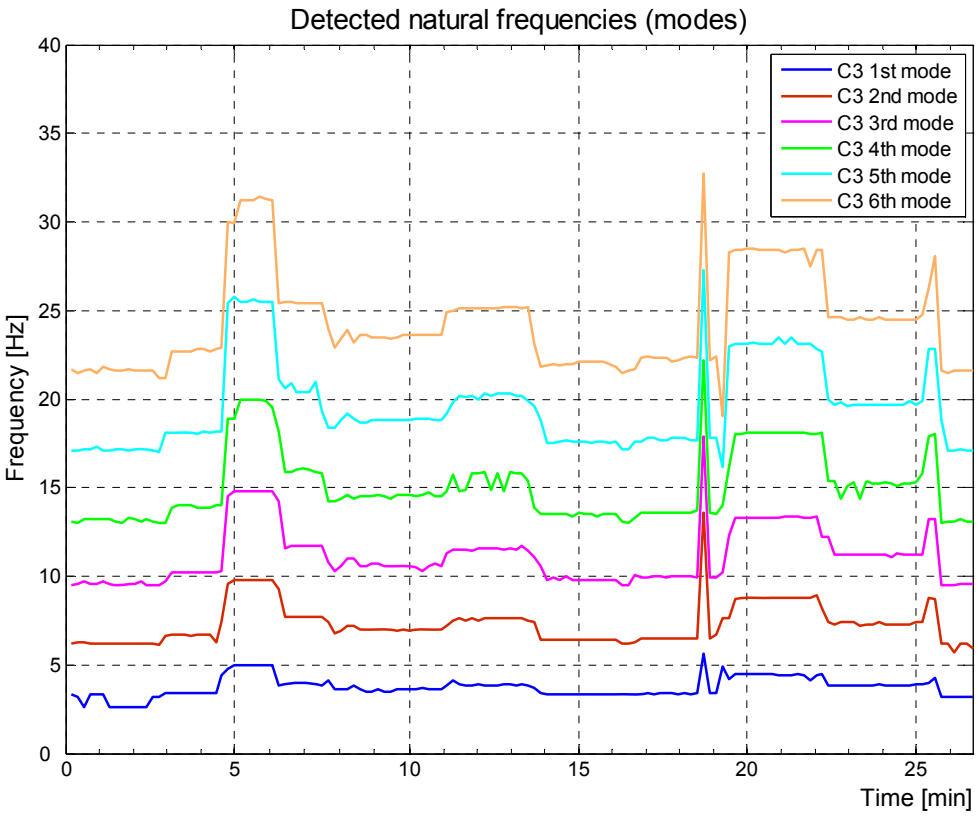

Figure 49: Detected natural frequencies of the bridge cable stay C3 during the test.

Figure 50 illustrates the top view of the bridge deck with six of the stay-cables. The position of the bridge load was illustrated with the corresponding time of the test. For instance, “p1” denotes the 1st position of the 500kg mass, which was left there for 2 minutes, between 3rd and 5th minute from the start of the test. The load then was moved to the 2nd position “p2”, and left there for 1.5 minutes and so on.

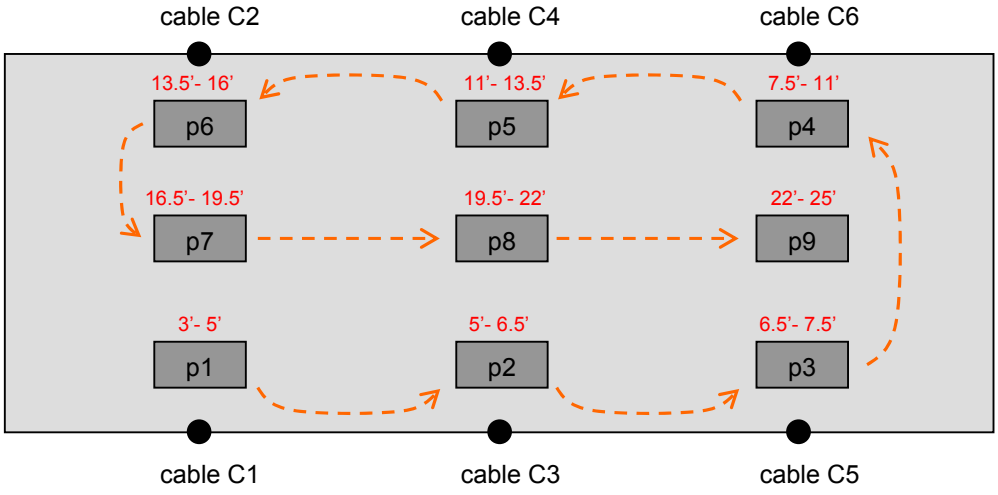

Figure 50: Position of the mass load on the longer side of the bridge deck and the corresponding time during the test.

It is obvious that placing the load near the cable results in a tension change in that particular cable. For example “p4” produces the biggest tension change in the cable C6. It can be concluded that the significant change in the detected natural frequencies of the cable C3 in Figure 49 exactly corresponds to the positions “p2” and “p8”.

In order to observe the changes of natural frequencies of every cable at once, a selected natural frequency from each cable is placed on the same plot.

Figure 51 shows such a plot of 6th natural frequency of all 6 cables. A 5-sample moving average was applied to the natural frequency data to remove the spikes caused by transient effects.

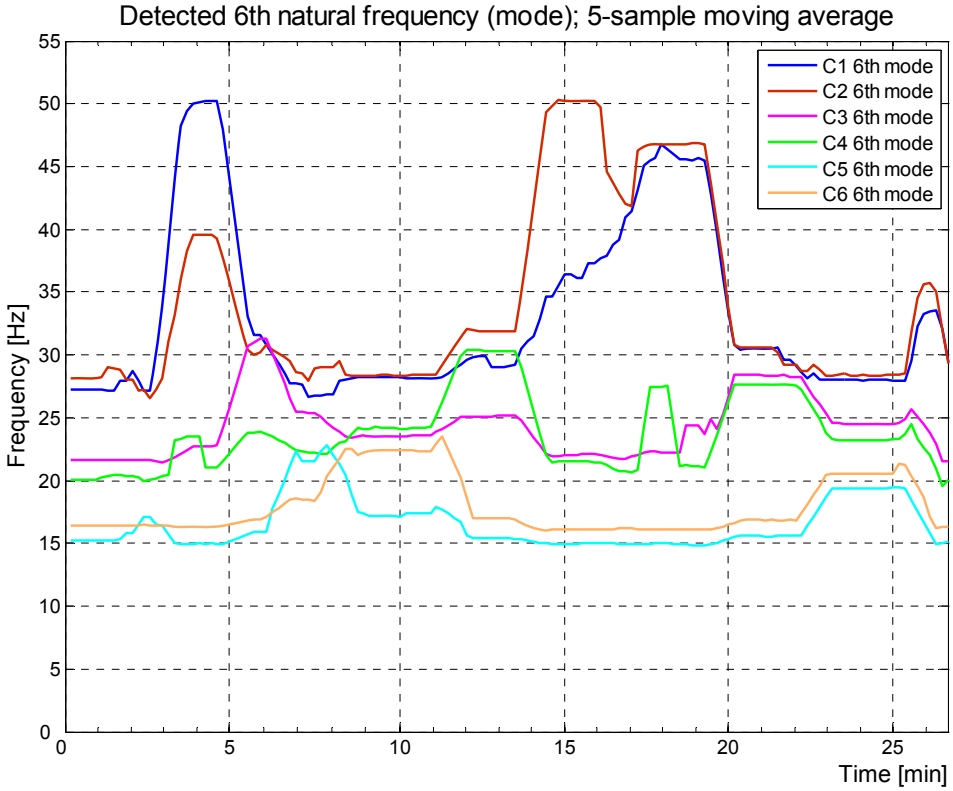

Figure 51: Detected 6th natural frequency of bridge stay-cables.

In order to compare the increase in the natural frequency of particular cable with the corresponding load position, the detected natural frequency results can be individually scaled to match each other. Such a plot allows observing the transition of tension (load) from one cable to another and it is shown in Figure 52. Figure 53 presents the detected natural frequencies normalized by their initial values at the test start what corresponds to the percentile change.

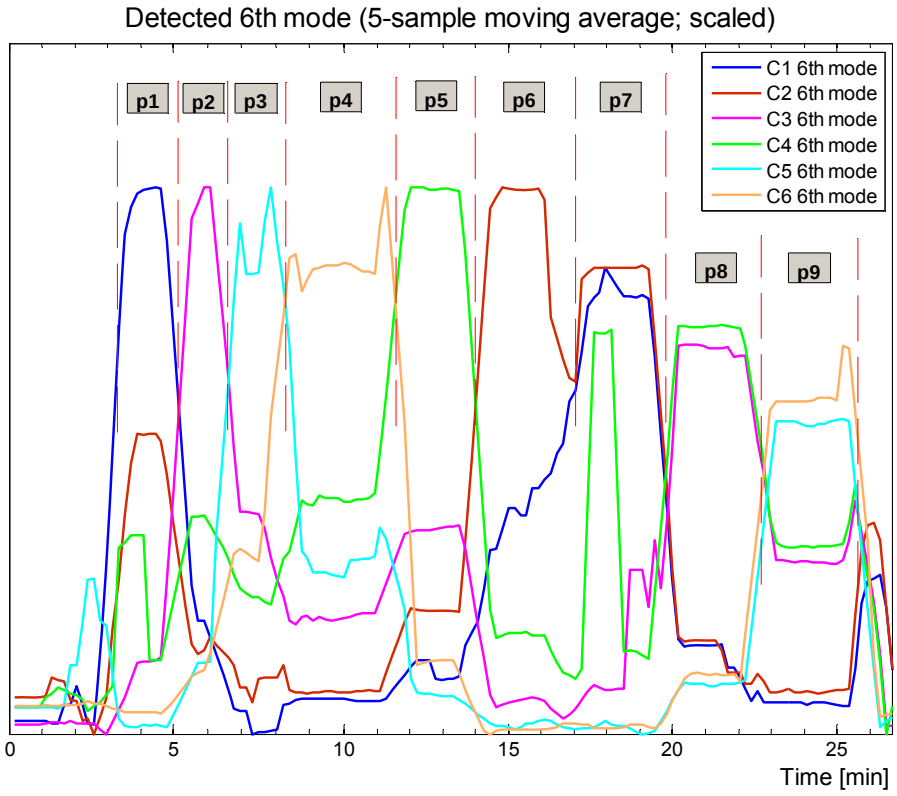

Figure 52: Scaled plot of the 6th natural frequency of the cable stays.

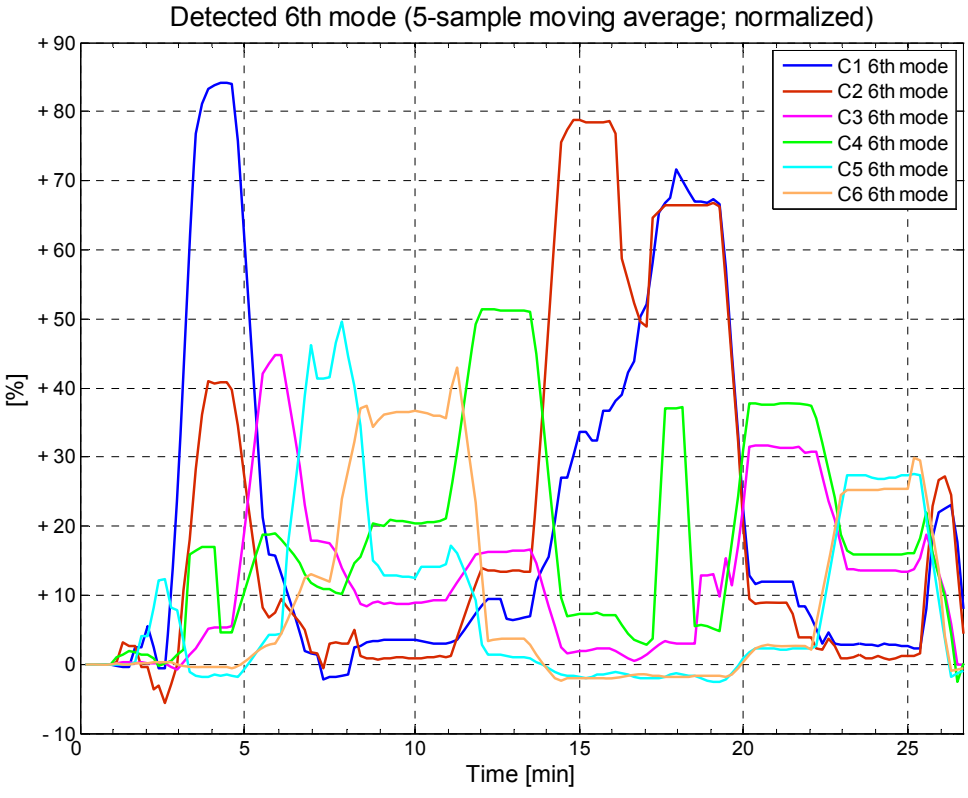

Figure 53: Percentile change of the 6th natural frequency of the cable stays.

Using a simplified cable model, the increase in cable tension could be estimated using (7) from chapter 1.4. Figure 54 shows such an estimation of the cable tension.

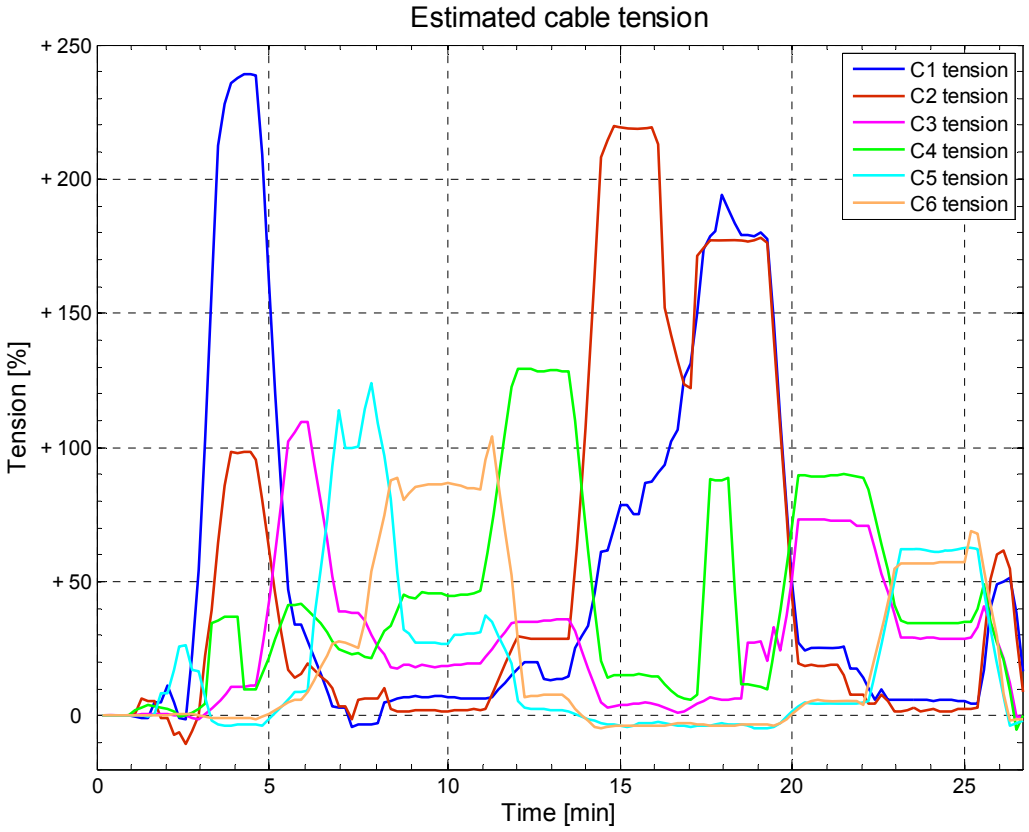

Figure 54: Estimated change of cable tension.

3.1.4 Demonstration Test

A short demonstration test was conceived as a part of the presentation of the developed system in the department meeting room. The idea was to observe in a real time the changes in the cable tension due to the repositioning of the bridge deck load.

Figure 55 shows the real time information from the WSN installed on the laboratory bridge in a user friendly form. Detected natural frequencies were illustrated by graphical bars which indicated the cable tension by their length and colour. By observing the graphic bars, the current position of the load could easily be estimated.

Figure 55: Visualisation of changes in cable tension force (natural frequencies).

3.2 Laboratory Test Network

A special WSN network was build to enable simulations of the real network deployed on the bridge. This network consists of the same number of nodes as the real network deployed on the bridge with the sink node connected directly to the laboratory computer. Figure 56 shows this test WSN running in the laboratory.

The laboratory WSN is used for the development and testing purposes and basically imitates the deployed network. Therefore, the nodes are built using the same platforms, have the same network addresses, are also battery powered and are organised in the same topology. The only differences are the reduced transmission power in order to achieve similar radio signal strength as in the deployed network and the absence of the sensor modules. As a consequence all data processing is done on a noise signal acquired from the unconnected input of the analogue to digital converter.

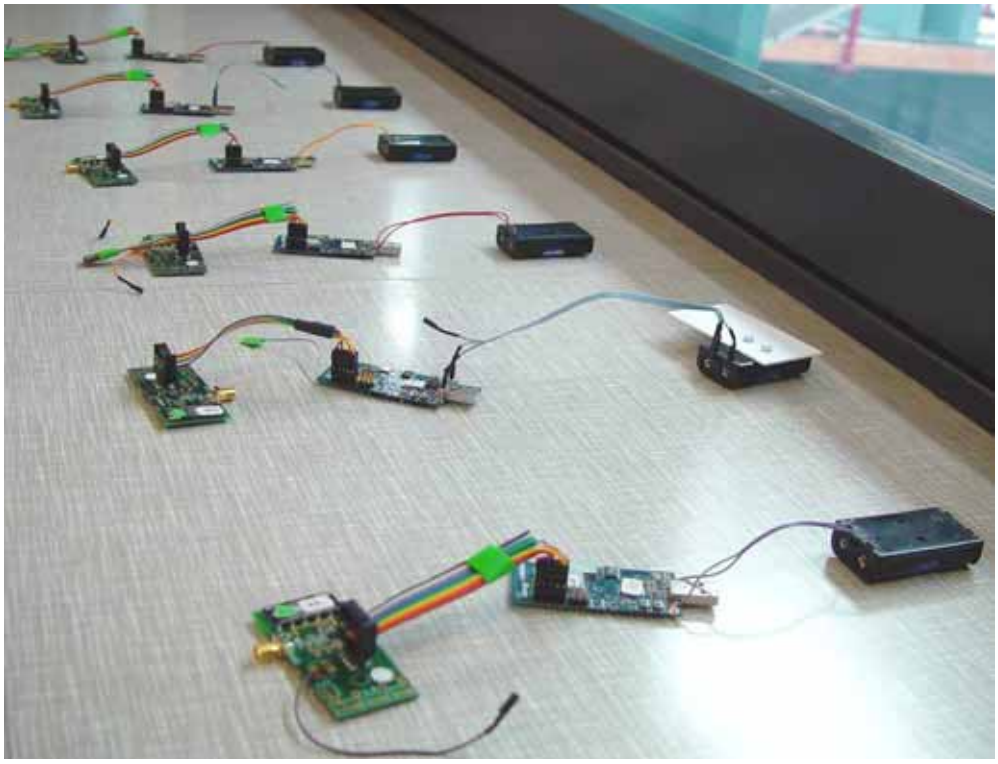

Figure 56: Laboratory WSN.

The laboratory network is used for the following tasks:

1. **Testing changes before performing them on the real network**

   Changes that might result in unwanted network behaviour (see section 2.2.3.3) can be evaluated before deployment.

2. **Development and testing of new features**

   The laboratory network was used during the development of new features of firmware as a testing tool. For instance, the laboratory network was used to develop and test the remote wireless reprogramming feature (see section 2.2.1.6).

3. **Testing new firmware versions**

   Before reprogramming the deployed network, new versions of developed firmware are first tested for stability and recovery on the laboratory network.

4. **Emulating error conditions**

   The conditions on the bridge that have potentially provoked the undesired behaviour of the network are simulated in the laboratory network to discover the potential sources of the problem.

5. **Discovering bugs**

   The laboratory network is constantly running with the same set-up as the deployed network to discover the occurrence of the system instability and potential threats.

6. **HCI protocol inspection**

   In order to get an insight in the communication between the WiseNode and Tmote Sky platforms, the HCI protocol is monitored within a particular or several motes. This is used for debugging and to discover error conditions of the protocol. The developed tools used to investigate the HCI protocol are the topic of the section 3.2.1.

3.2.1 HCI Protocol Testing Tool

For the debugging purposes the tool to monitor and test the communication protocol between WiseNode and Tmote Sky has been developed. It was used in the development of the HCI communication stack for the Tmote Sky (see section 2.2.1.5), and is still used in the laboratory test network (see section 3.2) to monitor the communication for irregularities.

Figure 57 shows the setup for the HCI protocol monitoring. The Tmote Sky board connected with a flat cable to the WiseNode is the host board. Two other Tmote Sky boards are used as sniffers on the RX and TX UART communication lines of the interface. Sniffed communication is forwarded through the USB port to the HCI protocol monitoring application.

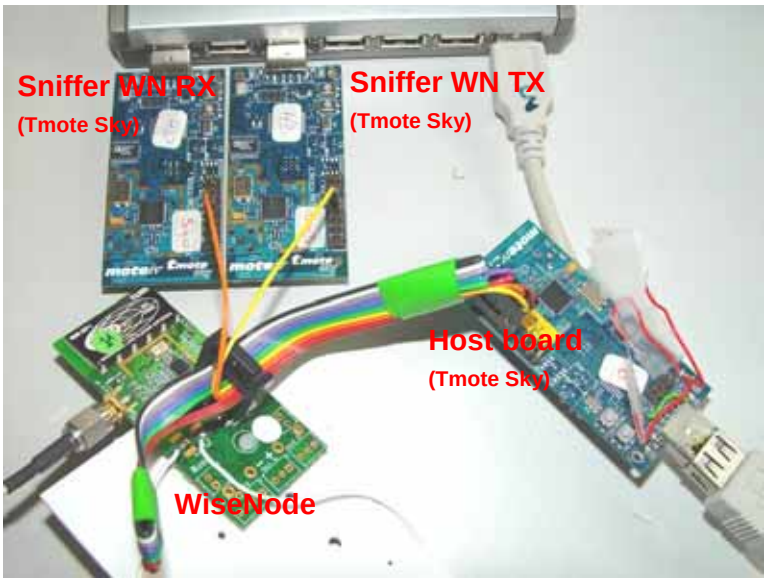

Figure 57: Tmote Sky boards are used as sniffers of the UART communication between WiseNode and host board.

The HCI protocol testing application is written in Java and features a graphical user interface. Figure 58 presents a screenshot of the HCI protocol testing tool, while a detailed screenshot of this tool can be found in Figure 123 in appendix B.

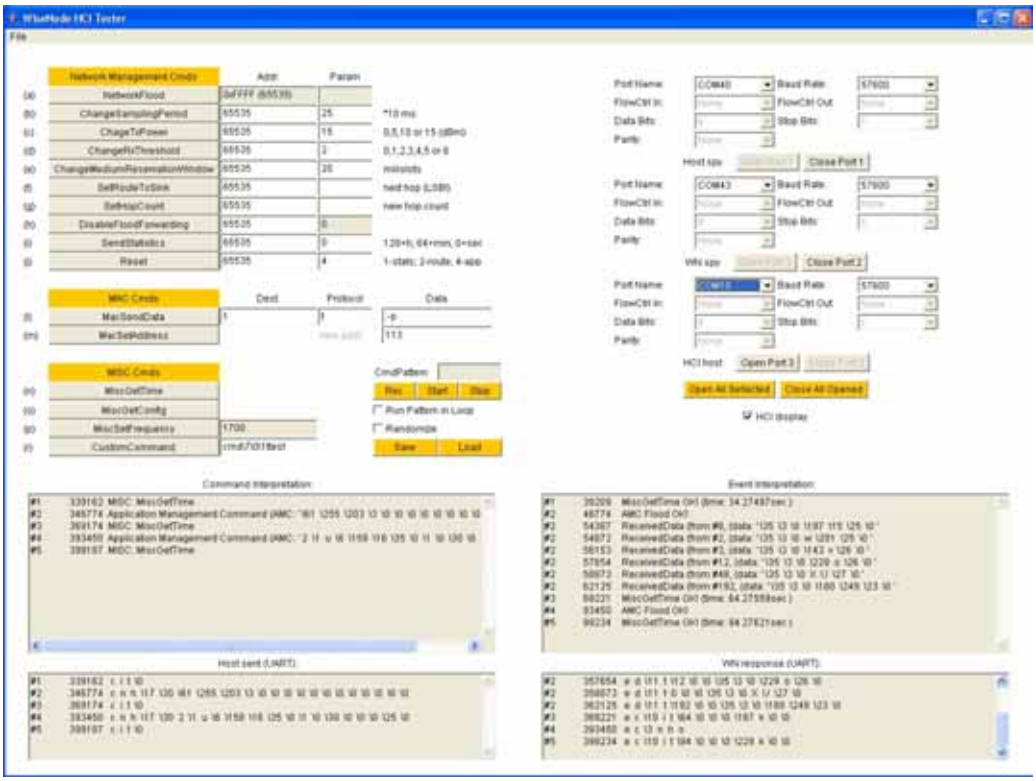

Figure 58: Screenshot of the HCI protocol inspection tool.

The developed HCI protocol inspection tool was used for the following debugging operations:

1. **Protocol monitoring**

   Besides the raw display of the UART communication, the protocol commands and events are displayed in an interpreted form. Figure 59 shows a screenshot of the event interpretation area of the protocol monitoring tool. The automatic characterisation of the communication packets enables the user to easily follow the protocol.

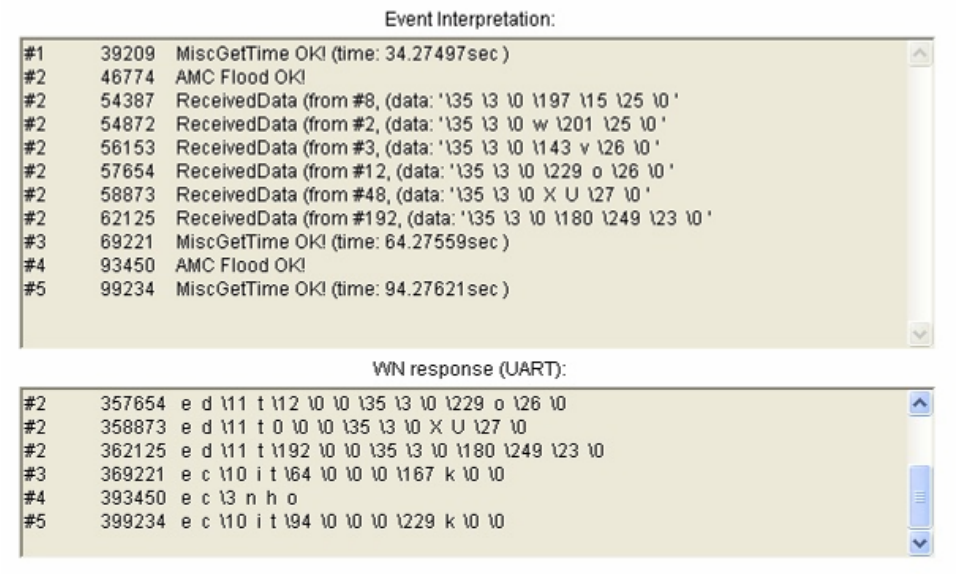

Figure 59: HCI event display area of the HCI protocol monitoring tool.

2. **Protocol logging**

   All of the received commands and events can be logged to a text file. The communication is saved both in its raw form and in its interpreted form. This enabled to investigate the protocol later from the generated logs after the system crash occurred. During the HCI stack development this feature was used to monitor the protocol for errors.

3. **Protocol testing**

   The functionality of generating and sending HCI commands has been developed to enable protocol testing. The user can send a HCI command simply by setting the desired command parameters and by pressing the corresponding button.

   A dedicated application has to be installed on the host board in order to perform the protocol testing mode. This application simply forwards the HCI commands received from the USB to the WiseNode.

4. **Automated protocol testing**

   The tool enables to record a desired sequence of HCI commands, which are then automatically sent. The command sequence can be run once or in a loop and the command order can additionally be randomised.

   This feature was used to test the stability of the HCI protocol and to make automated protocol test routines for new firmware versions. Generated communication was stored to a log and investigated after the completion of the test routines.

3.3 Field Test

The developed WSN-based monitoring system is deployed on a real cable-stayed bridge in Winterthur (see Figure 60). The planned duration of the deployment was 3 months, which is estimated to be long enough to provide information about long term system performance.

Figure 60: Stork Bridge in Winterthur.

The field deployment studies the performance of the system in a real world application and under outdoor conditions. There are several aspects of the system performance which are evaluated:

1. **overall system robustness**
    a. **software stability**

        The deployment should reveal long-term stability of the software running on the nodes and its ability to recover in cases of software malfunction back to normal operations. This includes firmware running on the application board (Tmote Sky) as well as the firmware on the WiseNodes. Both systems incorporate recovery mechanisms that reboot the system in the cases of such undesired behaviour.

    b. **hardware robustness**

It is usually required longer period of time to be able to evaluate the stability and durability of the hardware components. The overall robustness of the hardware system depends on the robustness and quality of the used components and system design. Main focus is to reveal the possible negative effects of harsh outdoor environment on the system and to evaluate used hardware enclosure.

2. **natural frequency detection algorithm**

   The performance of the implemented natural frequency detection algorithm is going to be evaluated. The experience gained from the field deployment is going to be used to enhance the algorithm performance, and to reveal the rationalities of such natural frequency detection system by discovering its limitations.

3. **wireless communication**

   a. **signal quality**

      A prerequisite for a good communication links between the nodes is a good quality of the wireless communication signal. The information about the received signal strength will be regularly sent from each node.

   b. **packet error rate**

      The percentage of unsuccessful attempts to send packets is the best measure to describe the link quality between sending and receiving nodes, and the MAC efficiency in general. Low packet error rate (PER) presents a prerequisite for low power consumption of the system, because additional energy is used for each retransmission.

4. **power consumption**

   The field deployment should provide enough information to model the real power consumption of the system. For instance, PER is very important parameter that is going to be used to build a model of the node power consumption in order to estimate the lifetime of the developed network (see section 3.4.2.5).

5. **wireless reprogramming functionality**

   One important feature of the developed system is the remote wireless reprogramming functionality. It is tested in the field and used to upgrade the firmware.

3.3.1 Preliminary Tests

Before deploying the monitoring system, a preliminary investigation was performed. The cable stays were instrumented with PCB3711 accelerometers (see Figure 61) wired to a 24-bit high precision data recorder and the ambient vibrations of the cables were recorded.

Figure 61: High precision accelerometers mounted on a cable.

The acquired acceleration data was analysed to obtain the information needed to appropriately set the parameters of signal acquisition, signal conditioning and processing algorithm. Using this information the sampling frequency for the signal acquisition on WSN nodes is set to 100Hz, and parameters of the sensor board were set according to Table 8.

Sensor Board Setup	Rating
Number of used channels	1
Signal amplification factor	50
High-pass cut-off frequency	1.6 Hz
Low-pass cut-off frequency	20 Hz
Acceleration sensitivity	33 mV/mg
Acceleration range	± 50 mg
Start-up time	~1 second

Table 8: Current setup of the sensor board.

3.3.2 Node Hardware Overview

To protect the hardware from environmental effects, all of the components had to be placed in an appropriate housing. The hardware enclosure for the field deployment is shown in Figure 62.

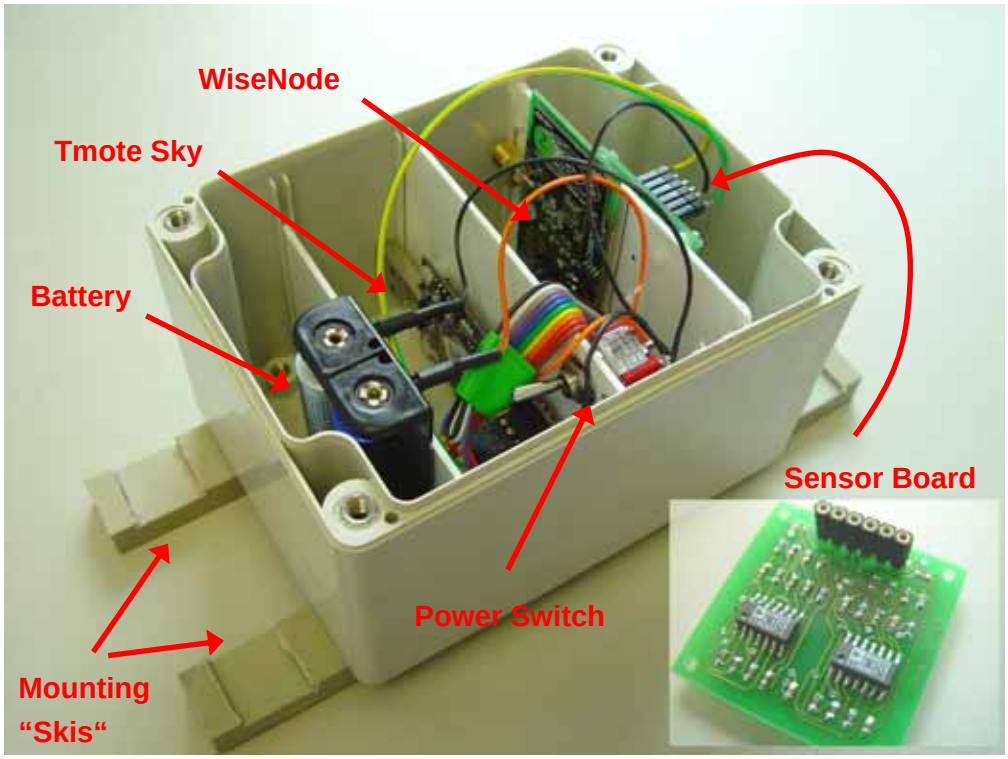

Figure 62: WSN node hardware is enclosed in a special plastic waterproof housing.

All of the hardware modules were mounted on small plastic boards that were then inserted into the box. It can be seen that the acceleration sensing module and the WiseNode were mounted on one plastic board, the Tmote Sky board and the power switch are on the middle board and the third one holds the battery pack.

The design of the hardware enclosure had to fulfil the following:

1. **non-metal housing**

   In order to use the on-board planar antenna of the WiseNode, the hardware enclosure had to be made of non-metal materials so the RF signal can pass the housing. Otherwise an external antenna would be necessary what would increase the costs significantly. Namely, the external antenna, RF extension cable and the dedicated RF connectors are quite costly components.

   Additionally an opening in the housing would have to be made for the external antenna, which could diminish the waterproof sealing of the selected

housing. Weakening the robustness of the hardware and durability of the solution contradicts point 3 below.

Therefore, a housing made from ABS plastic was selected for the deployment [48]. It has IP66 rating which offers sufficient waterproof protection for the outdoor deployment. It features internal rail mounting tabs for the convenient placement of the hardware modules. The dimensions of the selected model are indicated on Figure 66 (right).

2. **simple design**

The design should be simple in terms of the fast prototype assembly and the ability to efficiently perform changes to the hardware.

All the hardware components were mounted on the plastic boards which were then easily inserted into the enclosure (see Figure 63). Plastic boards and nylon screws were selected to minimize interference with the propagation of the RF signal. The use plastic boards presented a good choice for fast and cheap prototyping.

Figure 63: Hardware components were mounted on the plastic boards which were inserted into the mounting rails of the enclosure.

Such a design enabled all the hardware components to easily be accessible, allowed easy change of the hardware design and component rearrangement. For instance, replacing the old batteries during the servicing periods is easily performed by pulling out just the board holding the battery pack. Another example is the hardware redesign that was performed on the system power

supply by adding an additional battery pack (see section 3.3.3.2). Other possible changes to the system could be adding of additional components, i.e. additional sensors, or changing of the battery source (see Figure 65).

The only plastic board that had to be glued to the enclosure was the one holding the acceleration sensor, to guarantee the proper transfer of the vibration from the housing to the sensor module. Figure 64 shows how the board was glued with hot glue.

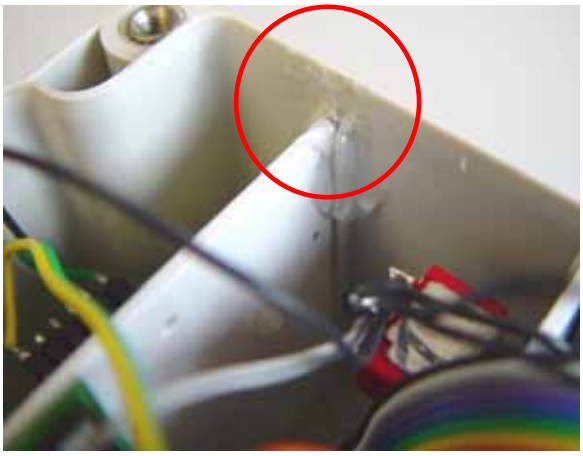

Figure 64: Board holding the sensor module is glued to the housing.

3. sufficient space

An enclosure with some extra space was selected to allow for hardware design changes during the prototype development. Some of the performed and potential hardware changes are mentioned in the previous point 2. Figure 65 shows the possibility of replacing the batteries by bigger ones.

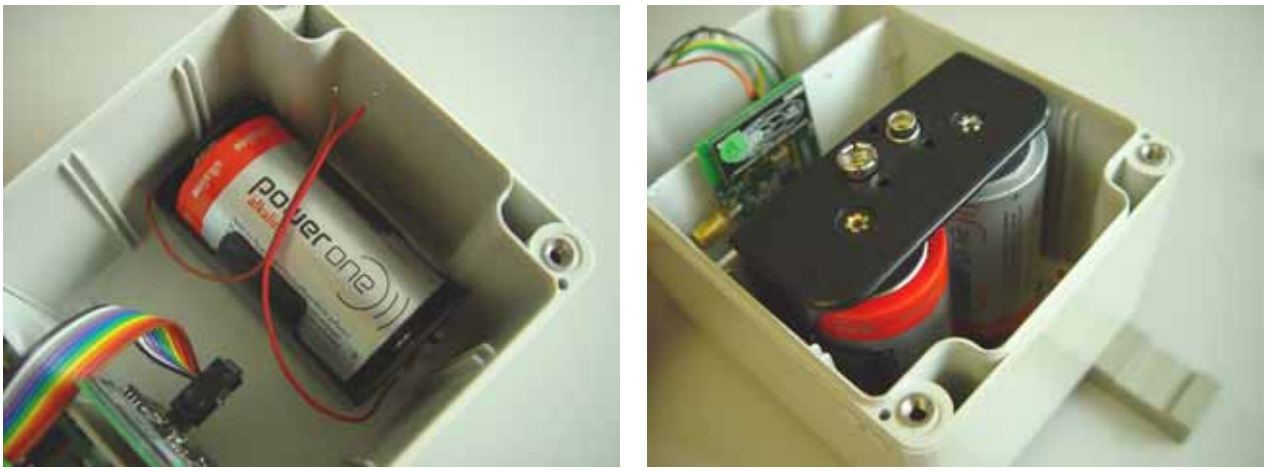

Figure 65: Enclosure provides enough space to place a bigger battery source.

4. **robustness and durability**

   The enclosure has to protect the hardware from external impacts like hail storms. Very important is to provide a sufficient waterproof protection so that the hardware is protected against rain and humidity. Such protection must not degrade due to the environmental effects, with temperature being the most important one. It should sustain the big range of temperatures from winter to summer.

After taking into account the mentioned criteria, WSN nodes for the field deployment were constructed. Figure 66 shows the outside look of one WSN node with indicated dimensions.

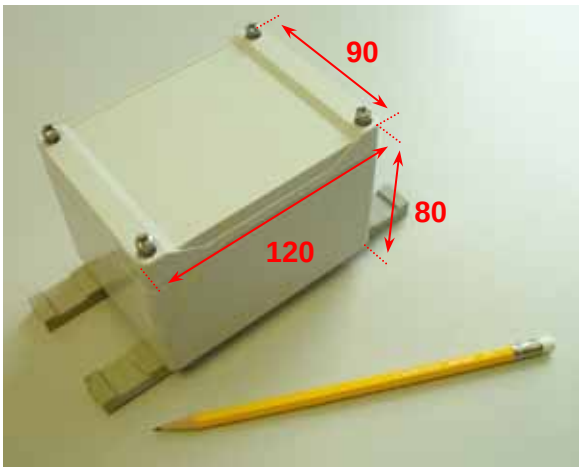

Figure 66: Outside look and dimensions of the developed WSN node [mm].

3.3.3 Power Supply

It is very important to choose the right battery as power source for the system. The battery must fulfil the following requirements:

1. **energy capacity**

   It is obvious that the battery should have a very high energy capacity to enable long system lifetime.

2. **stable voltage**

   The battery voltage shouldn’t vary much with operating temperature, age and remaining charge. However, for our application the most important is that the battery voltage during its lifetime does not drop below the 2.7 V

which is the minimum voltage required for the reprogramming functionality.

3. **good temperature characteristics**

   As all the other hardware components, the battery should as well be able to sustain a certain range of temperature that could develop inside the hardware housing. The most important is that the temperature operating conditions shouldn't significantly degrade the self-discharge characteristic of the battery and the voltage.

4. **long shelf-life**

   The shelf-life of the battery is determined by its self-discharge current, which is very important characteristic of the battery for the low power systems with long lifetime.

5. **good impulse current characteristics**

   The whole system operates at a low duty-cycle, i.e. the system is active only for a short period of time and needs higher power from the battery. Since the radio communication requires by far the most power, it is very important for the battery to deliver higher impulse currents required by the WiseNode board during the short radio activity.

In all the requirements above lithium batteries are outperforming alkaline batteries, especially at low temperature operation. Therefore, Lithium batteries have been chosen as the solution for the deployment.

In particular, two AA-size Lithium batteries with a nominal voltage of 1.5V from the manufacturer Energizer have been selected because of their superb performance (see Figure 67). The L91 model features 3000mAh capacity, 15 years of shelf-life*, big operating temperature range (-40 to 60 ºC) and good discharge characteristics. More information about this particular model and a comparison with the performance of a typical alkaline battery can be found in [51].

Figure 67: Selected Energizer L91 AA-size Lithium battery.

* The time until the capacity drops to 90% of the rated one

3.3.3.1Single Power-Supply Issues

After the first deployment of the WSN on the Stork Bridge, problems with natural frequency detection have occurred. The detected natural frequencies didn't match the expected values. In fact, it was very strange to see that sometimes all of the nodes were sending the same result!

In order to investigate this behaviour, the full spectrum of the measurement signal had to be acquired for analysis. To do so, a new program image was injected into the network which was dedicated to send the complete measurement time series to the network sink. The network nodes were reprogrammed using the implemented remote wireless reprogramming feature (see section 2.2.1.6). After the measurement acquisition, the nodes were one by one instructed to send the complete measurement time series to the sink, without any signal processing or compression performed. Instead of using WiseNET, a new multi-hop wireless network was built using the Tmote Sky's radio without the duty-cycled operation. This solution was used instead of WiseNET because it offered a higher throughput since the WiseMAC is developed to serve as a low traffic protocol. After the reception of the complete measurements, the nodes were instructed to reboot back to a standard low-power operation which uses the WiseNET.

A spectral analysis was performed on the received data. This revealed that all of the measurement signals had the spectrum similar to the one shown on Figure 68. All spectra had well defined peaks at frequencies of about 4 Hz and higher harmonics.

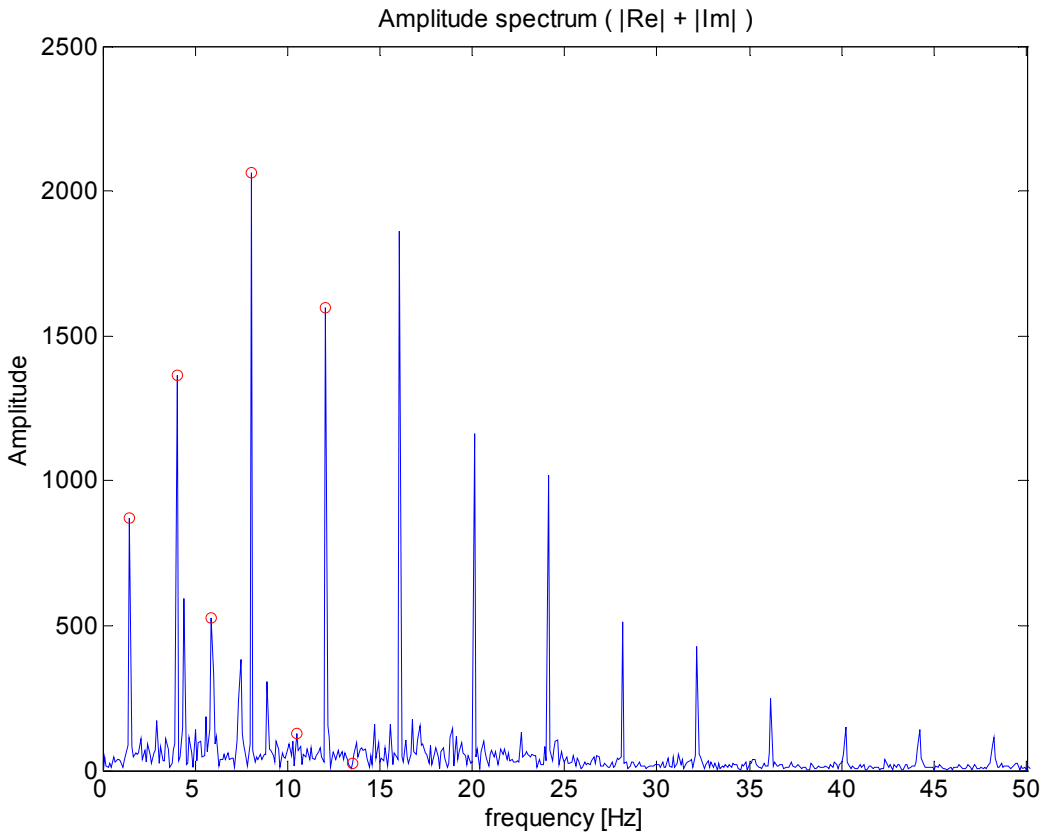

Figure 68: Artificial peaks in the signal spectrum were caused by regular network sampling performed by WiseNode which resulted in incorrect natural frequencies detection.

The explanation for this was the voltage drop caused by the network medium sampling, i.e. low-power listening, performed by WiseNode. WiseNode periodically turns on the radio in receiving mode to listen to the medium for a very short period of time to check for the network activity, i.e. to check for packets or broadcast messages. The power consumption of the radio causes the voltage drop of the power supply. More about WiseMAC and described technique can be found in the appendix A.2.3.

The default network sampling interval is 250 ms, which means that a resulting short voltage drop is happening at a frequency of 4 Hz. This influenced the signal output of the sensor module and the analogue to digital conversion. Thus, the spectrum of the measurement signal contained as well the spectrum of the voltage drop. Usually the network sampling voltage drop signal was actually stronger than the measured acceleration signal. This resulted in stronger peaks of the artificial signal which shadowed the peaks corresponding to the natural frequencies of the cable stays. As it can be seen on Figure 68, the identified peaks in the spectrum are not the wanted ones.

One solution to suppress the detection of the artificial peaks is to filter them out. This is accomplished by skipping detected peaks at a multiple of the network sampling interval, i.e. at 4 Hz, 8 Hz, 12 Hz, and so on. The output of such a modified algorithm is presented in

Figure 69. As it can be seen, by ignoring the expected peaks of the artificial signal, the detected peaks again correspond to the cable natural frequencies.

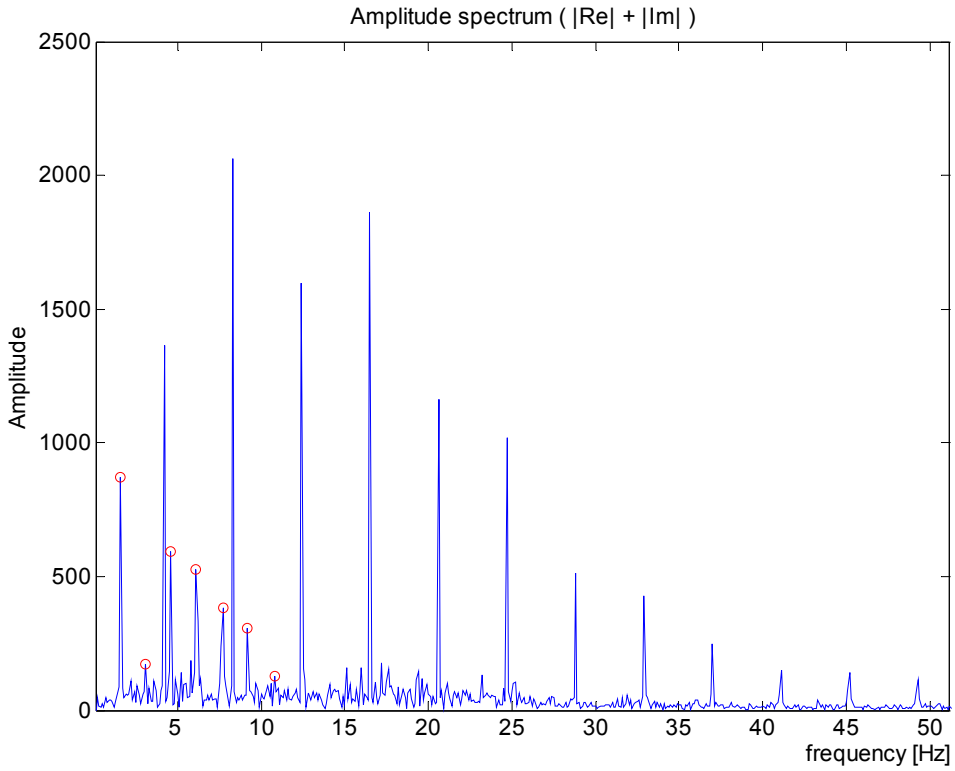

Figure 69: The natural frequency detection algorithm was modified to ignore the artificial natural frequencies, which resulted in the correct detection of the real natural frequencies.

This presented a good example for usefulness of the remote wireless reprogramming functionality since the whole investigation was performed without accessing the bridge. The functionality of the network was first changed by reprogramming the nodes with a new program image. After the successful execution of the new program the nodes were rebooted back to the standard low-power monitoring mode.

Moreover, after identifying the problem and developing a new modified version of the algorithm, the whole network was reprogrammed again with the monitoring application running the new algorithm.

3.3.3.2Split Power-Supply Solution

The method of ignoring the spectral components at the defined frequencies presented above is rather a workaround. For instance, the problem arises when some of the natural frequencies are at the same place as the peaks caused by the network sampling activity. These natural frequencies are then going to be ignored as well and not detected as they should be.

The best solution to remove the artificial signal from the measurement signal is to remove the voltage drop from the sensing and acquisition modules of the system. The easiest way to achieve this was to redesign the power-supply of the node. Figure 70 illustrates the original power supply system of the node and the new solution.

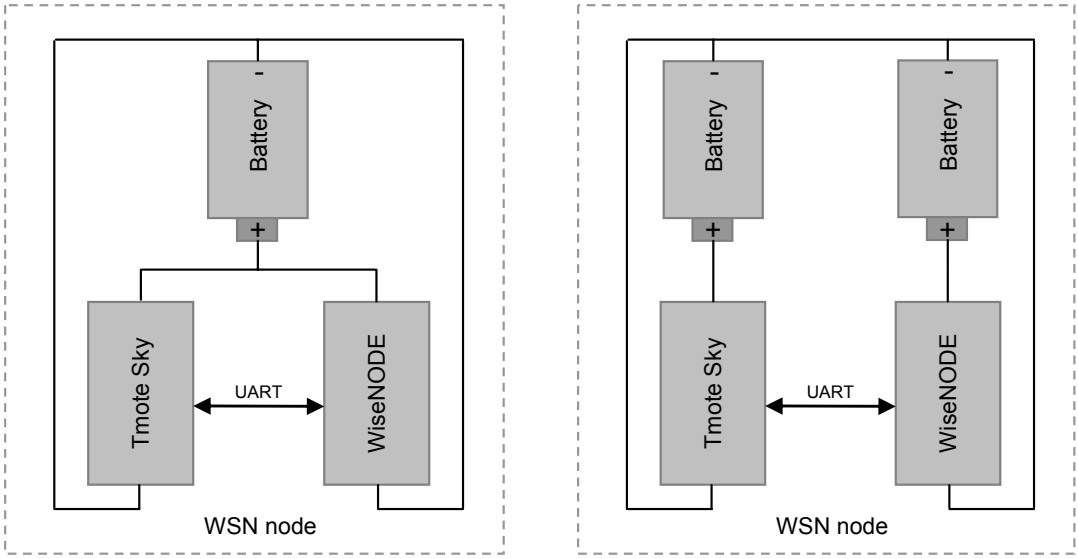

Figure 70: Power supply of the node: (left) Tmote Sky and WiseNode share the same battery. (right) A separate power supply for each of the platforms.

The idea is to have a separate power supply for the wireless communication and sensing and processing modules (Figure 70, (right)), instead of a common power supply where the voltage drop is reflected into the sensing circuitry (Figure 70, (left)).

This completely removes the voltage drop due to the network sampling from the measurement circuit. Figure 71 shows the signal spectrum after implementing such a split power supply. As it can be seen, it is not any more possible to observe the peaks corresponding to the voltage drop, which resulted in a proper detection of the natural frequencies.

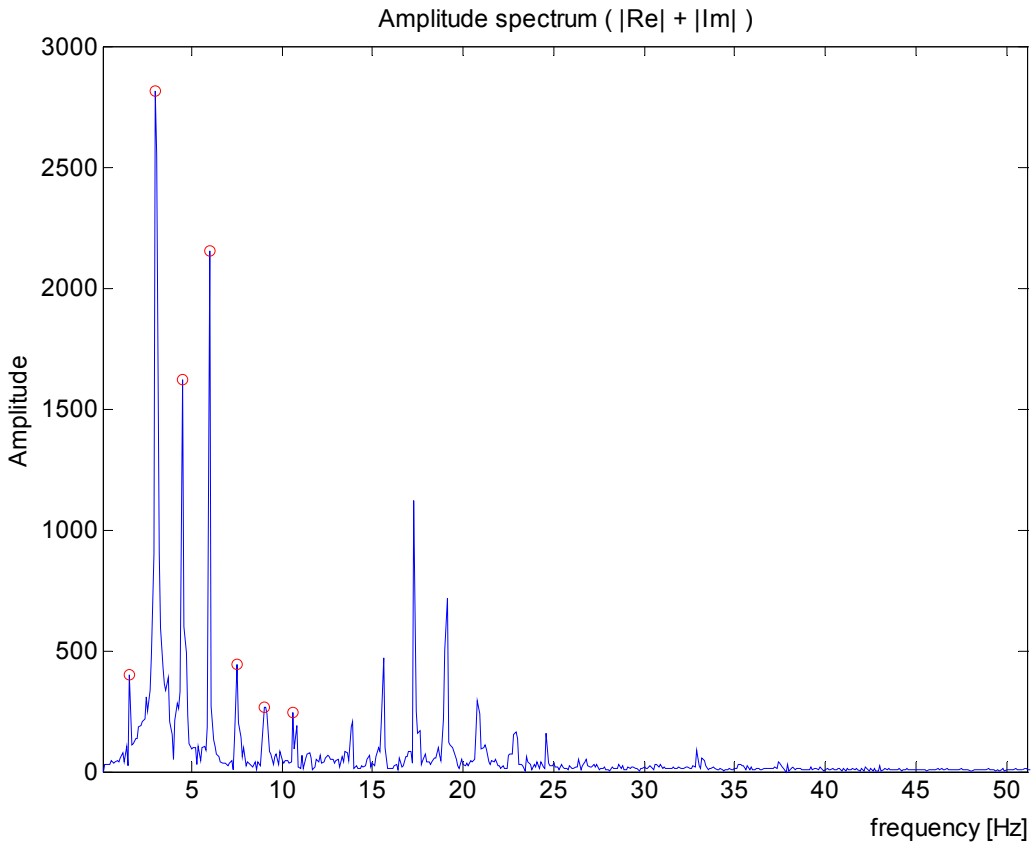

Figure 71: Signal spectrum does not contain anymore artefacts after the introduction of the split-power supply.

A node with the redesigned power supply is shown in Figure 72. The additional battery pack for split power supply system can be easily seen. This hardware change was easy to carry out since there was enough space within the hardware enclosure.

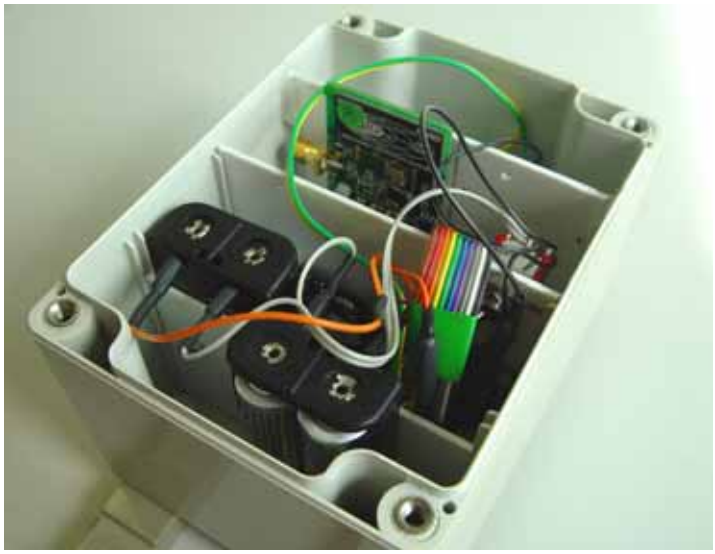

Figure 72: WSN node hardware after the change in the power supply system.

3.3.4 Deployment

Six measurement nodes (see Figure 73) and a sink node were built for the deployment on the bridge.

Figure 73: Six nodes for the field deployment.

Six adjacent cables on one side of the bridge were selected to be equipped with nodes C1 to C6 as shown in the bridge drawing in Figure 74. The sink node C0 was placed inside the abutment. The motes are mounted 3 meters above the bridge deck. This corresponds to about 10% of cable length.

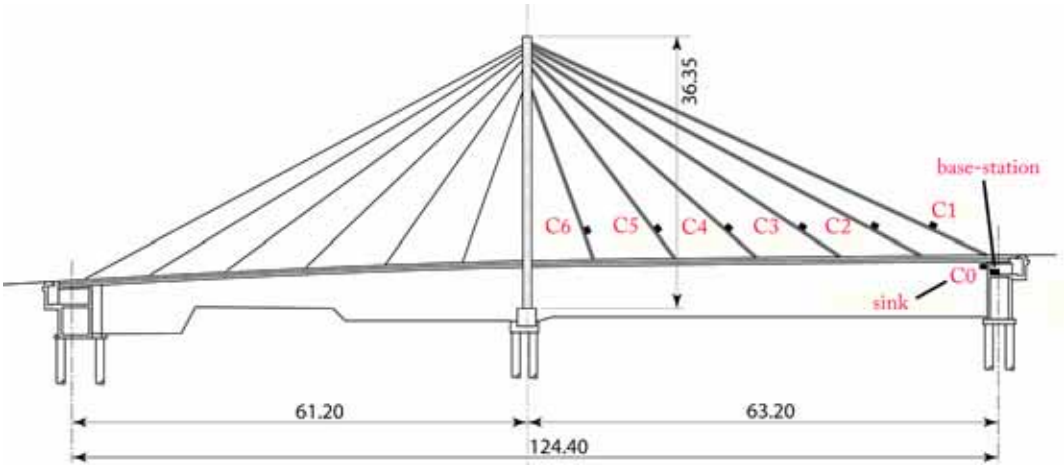

Figure 74: Drawing of the Stork Bridge indicating the six instrumented cables and the positions of the motes.

The nodes were set to measure the in-plane acceleration of the cable by using only one channel of the acceleration sensing module. They were mounted on the sides of the cables and not on the top of them in order to not have the body of the cable directly in the transmission path between the neighbouring nodes.

Figure 75 shows the sink node located under the bridge deck. It is in the proximity of the base station to which it is directly connected over an USB connection. The sink node is powered from the base-station over the USB cable.

Figure 75: Root node located under the bridge deck.

3.3.4.1 Base Station

The base-station (see Figure 76) is located under the bridge deck. It is connected to an unlimited power supply provided by the mains power supply available inside the bridge abutment.

The base-station was implemented using an industrial PC with USB and a wireless UMTS card. For more information about the base-station hardware components please see the section 2.1.2.

Figure 76: (left) Base-station located under the bridge deck. (right) Laptop running the operator console is directly connected to the base-station in order to access the WSN on-site.

The remote connection between the base-station and the control centre is over the UMTS communication infrastructure and IP technology. This connection is used to forward the incoming data from the WSN and to manage the network from the control centre.

It is also possible to establish a direct connection to the base-station using Ethernet. This approach is used when the network needs to be locally administrated, e.g. during the adaptation works on the network. Figure 76 (right) shows such a local connection to the base-station, where the portable computer running the operator console is used to administrate the WSN.

3.3.4.2 Network Topology

The topology of the network refers to the way the nodes are connected to each other. It represents the structure of the multi-hop communication implemented by WiseMAC. Figure 77 illustrates the default topology of the installed WSN that the nodes use to communicate and forward the data to the sink node.

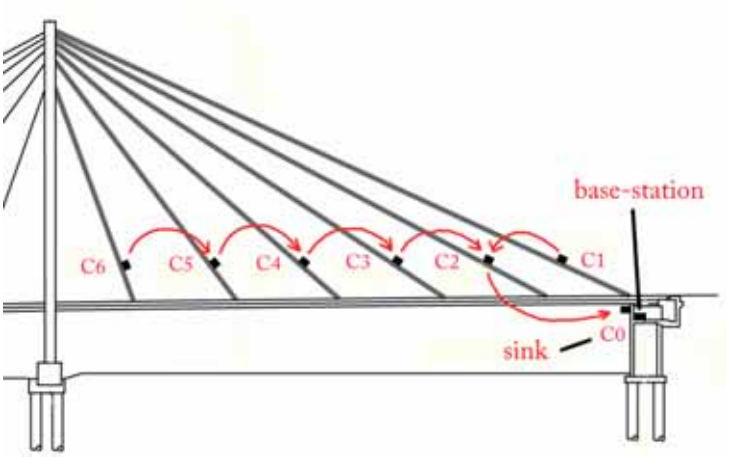

Figure 77: Topology of the WSN installed on the Stork Bridge.

For instance, node C3 sends its data and forwards the data received from node C4 to node C2. Node C2 presents the first hop to the sink, and it is the only one in the default topology meaning that all the data generated by the network has to be forwarded to the sink by this node. Such a topology places this node in the position with the highest power consumption, which is the main consideration when analysing the network lifetime (see section 3.3.6.1).

This topology has been selected because of the good signal quality between adjacent nodes mounted on the cables. The connection from the measurement nodes to the network sink has a much lower signal quality, since it is located under the bridge deck. The node with the best signal quality towards the sink has been selected as the first hop (C2). The connection quality in between the nodes is the topic of the following section.

3.3.5 Signal Strength

As mentioned above, a prerequisite for a good communication link between nodes is a good quality of the wireless communication signal. The received signal strength indicator (RSSI) is the measure in defining the strength level of the signal. It is defined as:

$$RSSI = 10\log_{10}\left(\frac{P_S}{1\ mW}\right) \quad [dBm], \tag{11}$$

where P_S is the power of the received signal in milliwatt.

Figure 78 shows typical RSSI levels of the WSN nodes towards its next hop node on the Stork Bridge. As it can be seen, the signal quality is more or less constant, usually at or above -73 dBm for the measuring nodes and usually at or above -78 dBm for the last hop to the sink (C2 to root node).

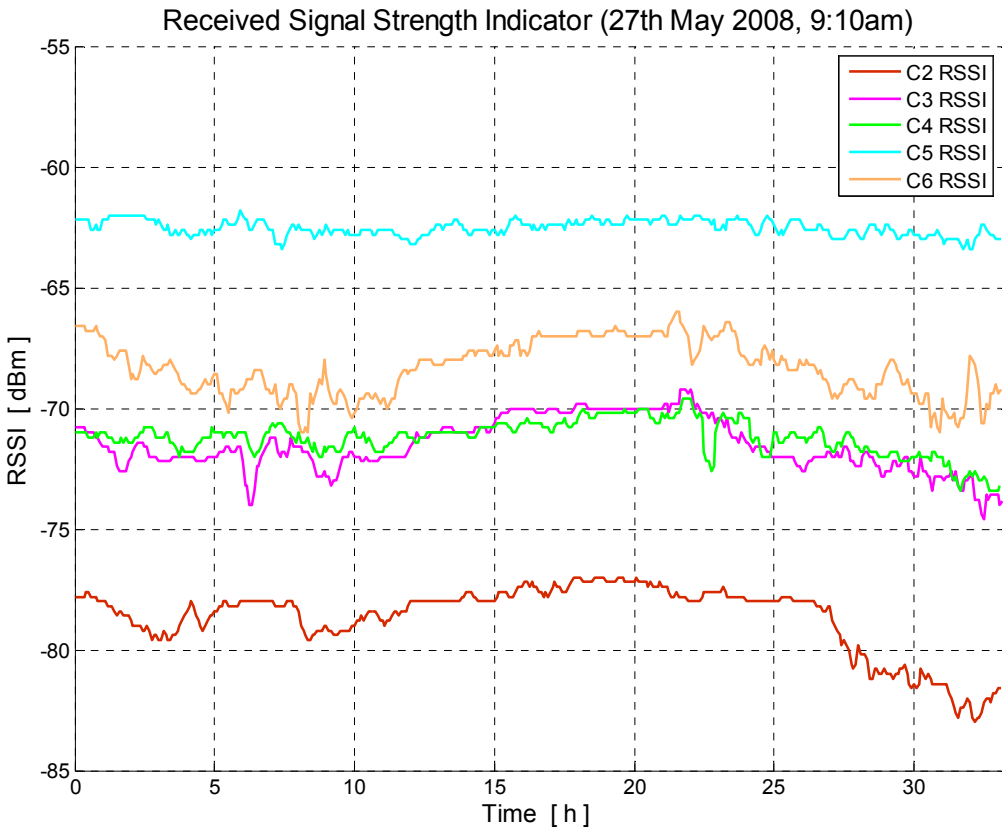

Figure 78: Usual signal strength towards the parent node from the default topology.

Some other data regarding the RSSI from the Stork Bridge can be found in appendix F.1.

3.3.6 Packet Error Rate

Packet transmission is considered unsuccessful if no acknowledgement of the packet reception is received. This could be due to a bit error that resulted in a corrupt packet. Bit errors in a transmission can happen due to the bad signal to noise ratio, because of the existence of interference signals, or because of packet collisions (hidden terminal problem). Interference can be caused by the multi-path propagation of the signal or from other devices operating in the same frequency band and acting as interferers.

Since the MAC protocol implements duty-cycled radio and synchronized preamble sampling, the packet will also not be received and acknowledged if it was missed due to synchronization errors between sender and receiver.

Information about the PER presents a very valuable information as it is going to be used to build a more realistic model for estimating the energy consumption of the WSN nodes and the resulting network lifetime. More about the modelling of the power consumption and network lifetime is presented in the following section 3.3.6.1.

The information about the number of successfully and unsuccessfully transmitted packets is provided by the special diagnostic packets generated by WiseNodes. Those packets are received at certain points in time determined by their transmission interval and are logged at the control centre.

3.3.6.1 Deployed Network

One of the diagnostic logs from the WSN deployed on the Stork Bridge is going to be observed in this section in regard to transmission errors. The data presented here is from a 33 hour long diagnostic log from 27th May 2008 starting at 9:10am. Some other data regarding PER from the Stork Bridge can be found in the Appendix F.1.

Figure 79* shows the accumulated number of unsuccessful transmissions from individual WSN nodes. The plot shows how the number of transmission errors was growing from the start of the session. A monotonous trend can be observed with some parts where there is an increase in the slope indicating an increased packet error rate. Figure 78 shows the corresponding RSSI levels during the time when this network diagnostic was performed.

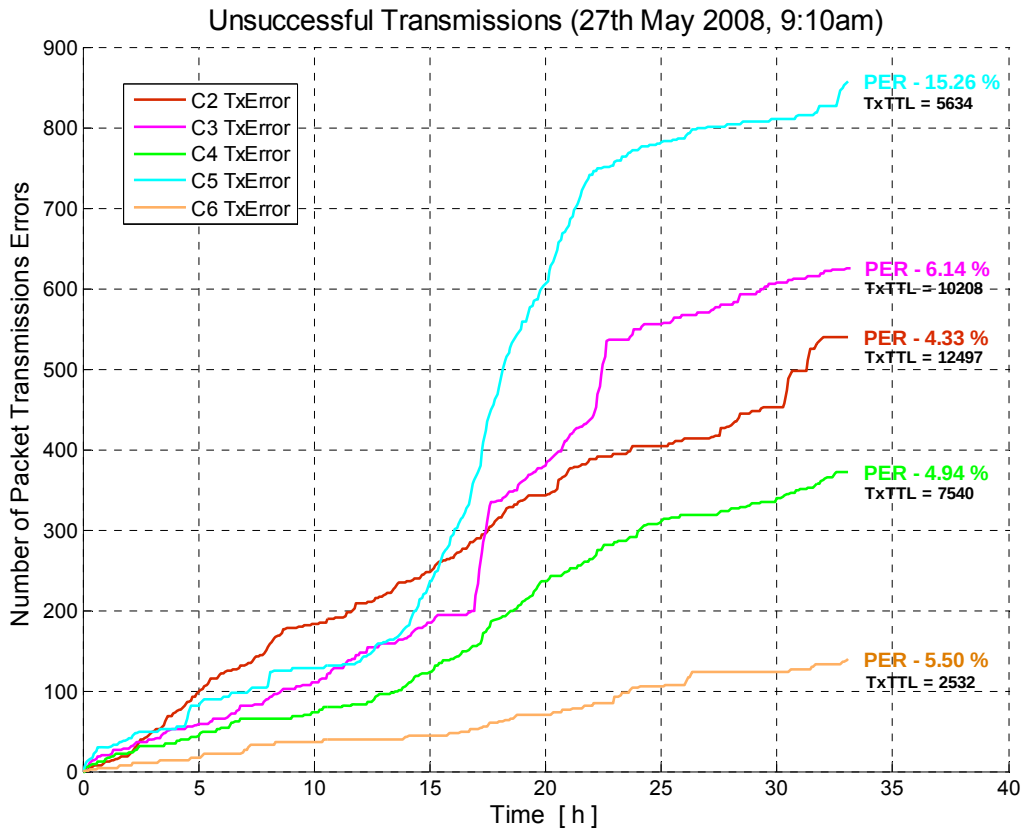

Figure 79: Number of unsuccessful packet transmissions from the deployed WSN.

* Node C1 is not represented at the plots as it was not functional at that moment.

The total PER during the observed period of time is ranging from 4.33% for node C2 to 15.26% for node C4. The PER is given as the ratio of the number of unsuccessful packet transmissions to the total transmissions in a given time interval:

$$PER(t_k) = \frac{\Delta N_{TX\_Error}(t_k)}{\Delta N_{TX\_Error}(t_k) + \Delta N_{TX\_Success}(t_k)} \cdot 100\,[\%], \tag{12}$$

where $PER(t_k)$ is the percentile ratio of the increase in transmission errors $N_{TX\_Error}(t_k)$ and the total number of transmissions in the interval at the discrete point of time t_k , when the diagnostic packet is generated.

Figure 80 shows how the PER of individual WSN nodes was changing. As it can be seen, the packet error rate sometimes reached even 50% for node C5. It is obvious that the increased levels of PER correspond to an increase of the slope of the total number of transmission errors (Figure 79), that can be observed on the previous plots, especially for nodes C3 and C5.

A longer period of an increased PER is noticeable for node C5 during the 14th and 22nd hour from the beginning of the network diagnostic, what corresponds to the time from about 11pm till 7am.

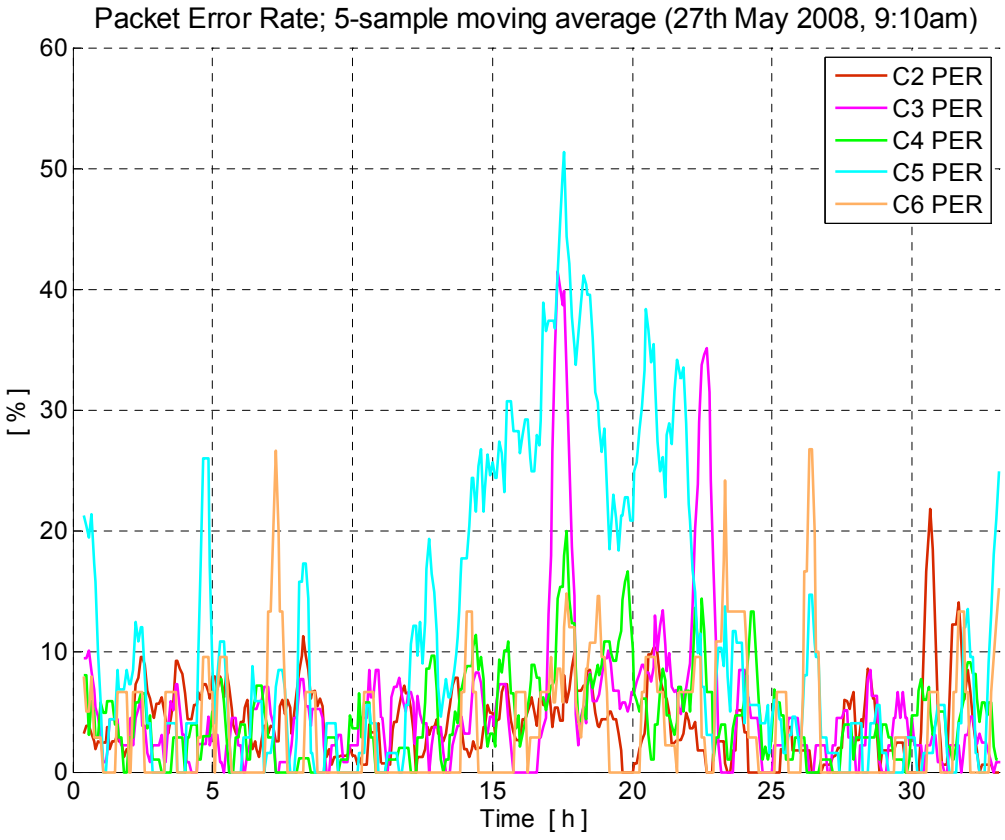

Figure 80: Packet error rate from the deployed WSN.

3.3.6.2 Laboratory Network

One of the diagnostic logs from the laboratory network is going to be observed in this section with respect to transmission errors. The data presented here is from a 86 hour long diagnostic log from 9th May 2008 starting at 6:44pm.

Figure 81 shows the RSSI achieved in the laboratory network in an attempt to simulate conditions on the deployed WSN in terms of RSSI. In comparison to the RSSI levels on the bridge (see Figure 78) it can be seen that the achieved signal levels have similar values in terms of dBm.

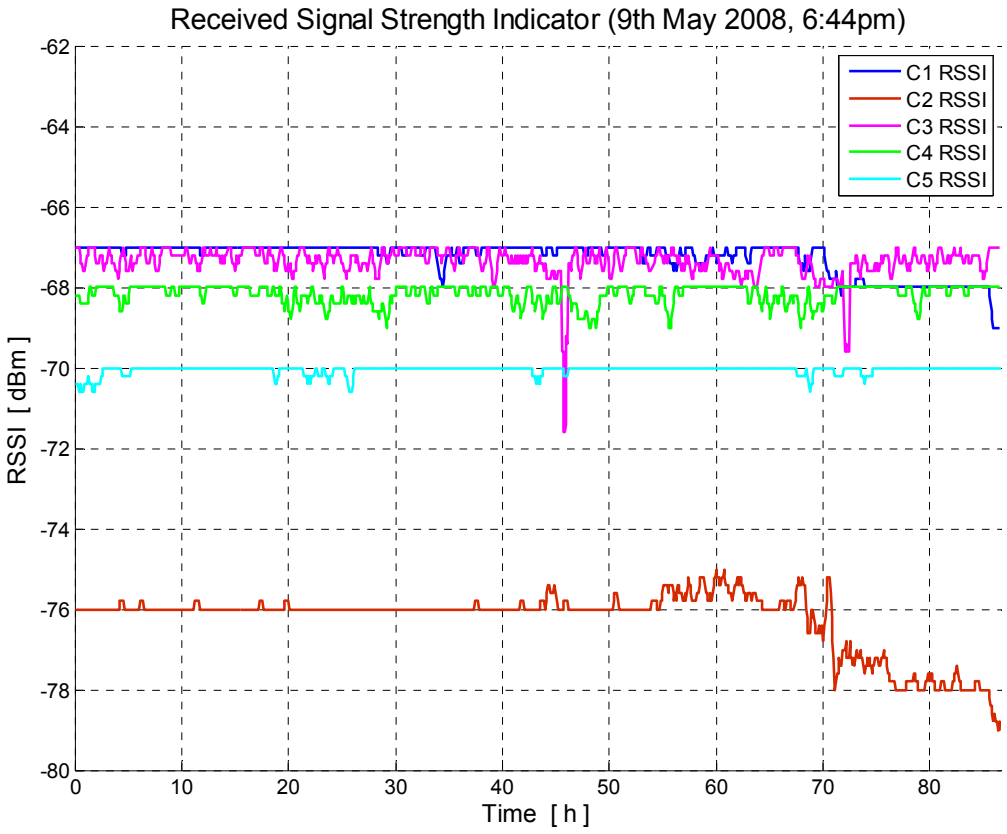

Figure 81: RSSI levels from the laboratory network.

Figure 82* shows the total number of unsuccessful transmission attempts from the nodes in the laboratory network. The plot shows how the number of transmission errors was growing from the start of the session. A monotonous trend can be observed with more or less constant slope.

* Node C6 is not represented at the plots as it was not available at that moment due to a hardware defect.

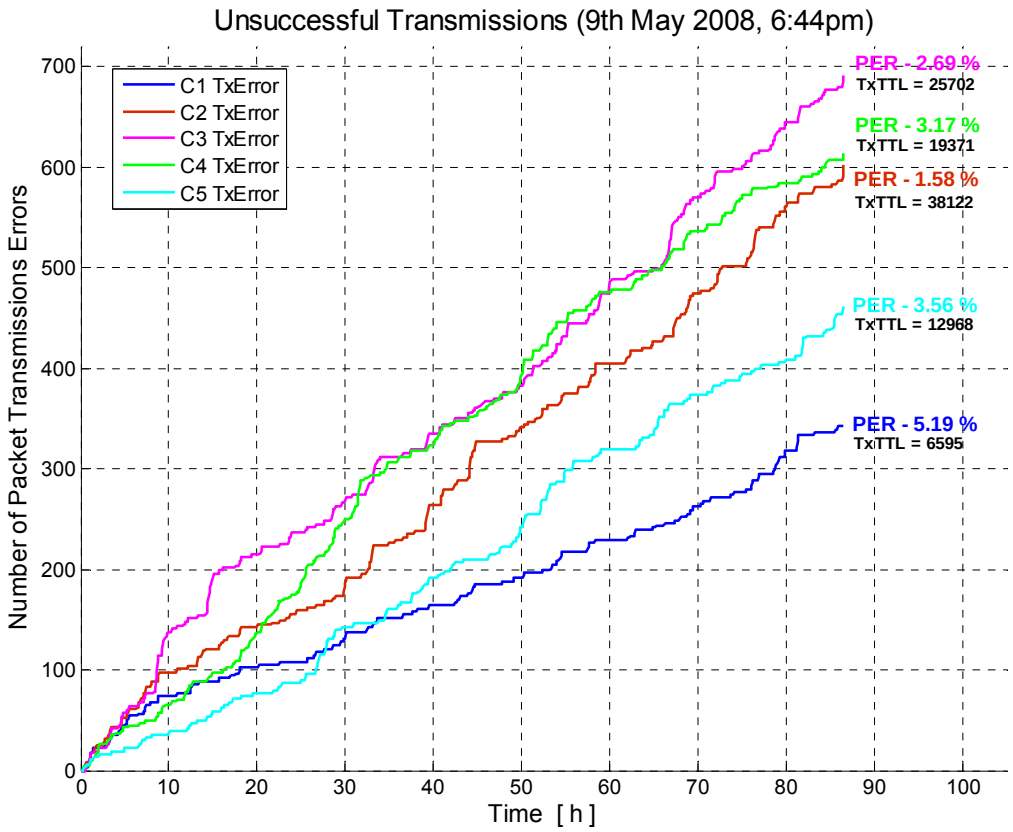

Figure 82: Number of unsuccessful packet transmission in the laboratory WSN.

The total PER during the observed period of time is ranging from 1.58% for node C2 to 5.19% for node C5.

The interval PER, given by expression (12), for the laboratory network nodes is presented on Figure 83. No longer periods of an increased PER are noticeable as it is the case with the PER from the bridge network presented in the previous Figure 80.

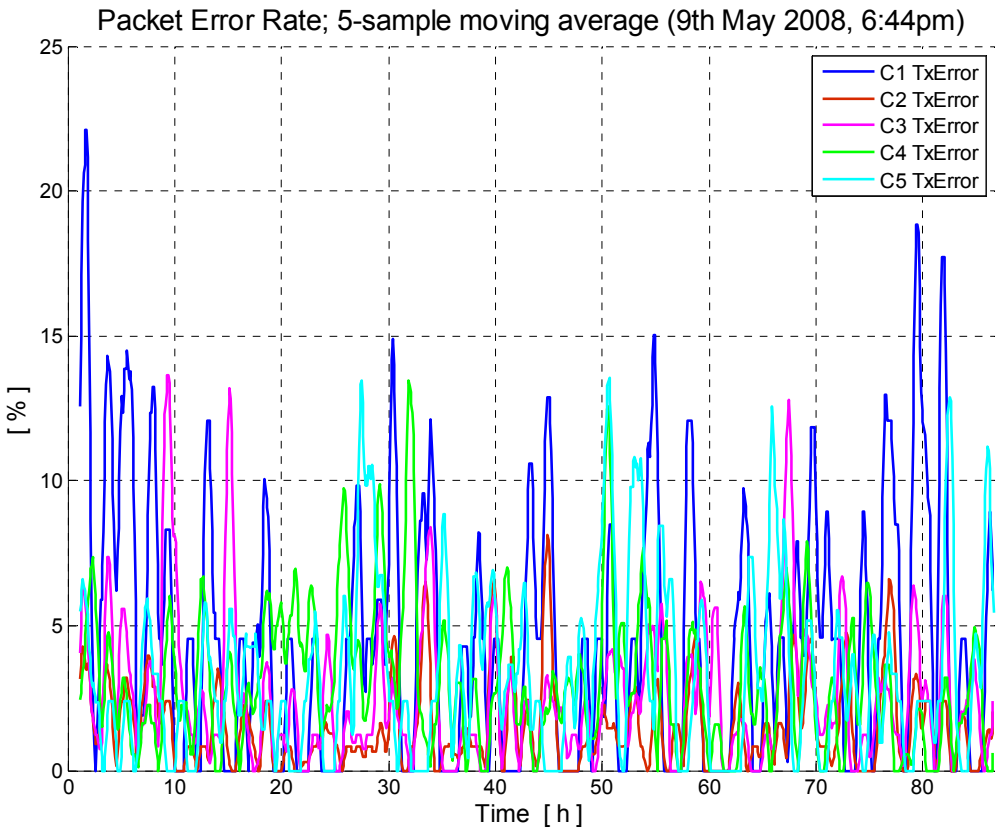

Figure 83: Packet error rate in the laboratory WSN.

3.3.6.3 Conclusion

First of all it is important to point out that the provided figures reflect the first experience with the field deployment on the Stork Bridge. Another version of WiseNode firmware is ready to be installed on the bridge to address some of the potential synchronization problems. The new version should provide similar or better PER figures as those achieved in the laboratory. So far the deployed network did not provide the same performance as the one achieved by the laboratory network, a packet error rate below 5%. The following considerations address the potential causes of the increased PER.

Looking at Figure 78 and Figure 79, a correlation between the increase in the slope of the total number of transmission errors and the RSSI levels cannot be made. The only correlation that can be made is valid for the last hop to sink node (C2) which has the lowest RSSI levels. Namely, we can see that the increase in the slope of transmission errors after the 30th hour from node C2 corresponds with the lower levels of RSSI. The same could be concluded when observing Figure 146 and Figure 147.

In fact if we observe the situation for other nodes than C2 more precisely, we can even see a slight decrease in the transmission error trend when the RSSI is slightly lower, and vice versa. As this is contradicting, a conclusion could be made that the RSSI levels

greater than -80dBm do not present the main factor for the PER. To support this conclusion, additional tests should be carried out using the laboratory network under controlled conditions regarding temperature in order to see how the change in the RSSI levels affects PER.

Although the last hop towards the sink (C2) has the lowest level of signal strength (RSSI) in both the deployed and laboratory network, it is still able to achieve significantly better PER then any other node in the network. That should rule out the RSSI as the main cause of PER for the other nodes of the network and can indicate that the majority of the packet errors are due to MAC synchronization issues and not because of the bit error rate due to the signal quality. However, this does not exclude other factors related to the signal quality, like possible signal interferences.

It can be seen that the last hop exhibits lower PER. This is due to the fact that the sink node is most of the time a receive-only node. In the WiseMAC protocol, a node that is waiting to transmit cannot receive a packet in the interval between the carrier sensing and the transmission. A node that transmits fewer packets will hence have fewer “blocking” periods and thus a lower probability to be unable to receive a packet (this is interpreted as a packet error at the sender side).

Uncompensated temperature drift could be one of the reasons for the possible synchronization problems. Operating temperature influences the frequency of the crystal oscillators used to generate the system clock. When nodes experience different local temperatures, their system clocks drift apart depending on the temperature difference. Such temperature drift must be compensated in order to achieve the required accuracy for the synchronization. As in the deployed WSN all of the hardware components are placed in a plastic housing (see section 3.3.2), the temperatures within the enclosure could differ significantly and produce the uncompensated temperature drift, which could explain the longer periods of increased interval PER (as seen in Figure 80, node C5). To investigate this possibility, nodes would have to be enhanced with additional temperature sensors to measure the temperature within the enclosure. An other approach would be to simulate different temperature conditions of the nodes on the laboratory network and see if the temperature induced drift will degrade the performance in the terms of PER.

Diagnostic logs of the laboratory network reveal that in some sessions some of the nodes have an increased total PER of about 10%, while in some other sessions all of the nodes achieve total PER of less than 5% (see Figure 82). Currently the reason for this is unknown but the explanation might lie in the fact that nodes are placed along the office window. During the sunny days some nodes are in the shadow while the other ones are in the direct sunlight (see Figure 84).

Figure 84: Some of the laboratory network nodes are in a direct sunlight (left), while the others are in the shadow (right).

This causes unequal operating temperature conditions at neighbour nodes, which result in temperatures difference between the nodes to become significant enough to produce the clock drift greater than the one being compensated for, thus resulting in the problems with synchronization. Further tests need to be carried out to investigate if the effect described above is present and how the network performs under controlled conditions regarding the operating temperature.

The packet error rate influences the lifetime of the network. Such an effect can be roughly approximated with the simple model of the node power consumption that includes the information about the PER which is presented in section 3.4.2.5. Further investigation is required to identify and characterize the reasons for the periods of increased PER. However, as discussed, a strong indication exists that the majority of transmission errors are due to synchronization problems.

Laboratory experiments at CSEM have recently confirmed this assumption. A decrease of inaccessibility periods (when a node has some message to transmit) and correcting some timing issues has reduced the observable PER to negligible values (less than 0.5%). This must however be confirmed using field experiments.

3.4 Network Lifetime

In the following sections, a simple power consumption model for estimating the network lifetime is going to be analysed. Different scenarios will be simulated to observe how different effects influence the lifetime of network.

3.4.1 Power Consumption Analysis

This section focuses on finding of the average power consumption of the node. The average power consumption and the capacity of a specific battery determine the network lifetime. Such an approach has already been made for the power consumption of the WiseNode (see appendix A.4).

The total average power consumption of a node can be divided into two parts:

$$P_{TTL} = P_{WN} + P_{TM} \,, \tag{13}$$

where P_{WN} is the average power consumption of the WiseNode and P_{TM} the average power consumption of the sensor module and the Tmote Sky processing board. The power consumption of the WiseNode is taken from the calculations in appendix A.4. We can re-write equation (25) from appendix A.4 in the following form:

$$P_{WN} = P_Z + P_{LPL} + E_{FWD} N\lambda \,, \tag{14}$$

where P_Z is idle power consumption of the WiseNode, P_{LPL} is the average power in the low power listening state, i.e. network sampling. The energy for forwarding a packet E_{FWD} is given by equation (26), N is the number of network nodes and λ is the measurement rate. P_{LPL} is determined by the network sampling period T_W and for all of our considerations it is going to be set to the default value of 250 ms.

The average power consumption of the Tmote Sky can be written in the same :

$$P_{TM} = P_I + \left(E_{MEAS} + E_{DSP}\right)\lambda \,, \tag{15}$$

where P_I is idle power consumption of the Tmote Sky, E_{MEAS} is the energy required to perform ameasurement, and E_{DSP} is the energy consumed by the measurement processing. The measurement energy consists of the energy required to power the sensor module and the energy consumed by the Tmote Sky board to perform A/D conversion. Hence, it can be written as follows:

$$E_{MEAS} = E_S + E_{ADC} = \left(I_S T_S + I_{ADC} T_{ADC}\right) \cdot U \,, \tag{16}$$

where I_S and I_{ADC} are the currents required for sensor module and A/D conversion, respectively, T_S and T_{ADC} are the corresponding times and U the supply voltage. The energy consumption for signal processing is given by:

$$E_{DSP} = I_{DSP} T_{DSP} \cdot U \,. \tag{17}$$

Table 9 gives the parameters required to determine the energy consumption for signal acquisition and processing. The values are based on the 1024-sample FFT implementation and 100 Hz sampling frequency. The A/D conversion takes about 10 seconds. The power on state of the sensing module 1 second longer, since it requires a 1 second of start-up time.

Task	Current Consumption [mA]	Duration [s]
ADC	0.72	10
Sensor Board	1.3	11
DSP	1.9	0.6
idle	0.02	-

Table 9: Power consumption parameters of the sensing and processing part.

A worst-case topology for the power consumption will be assumed for the simulation. The worst topology is the chain topology illustrated in Figure 85. In this situation, every node has to forward data packets from all the other nodes that have a higher number of hops to the sink. The most critical is the last hop node because it has to forward all the data from the other nodes. Thus, the power consumption of the last hop node is the highest in the network and the network lifetime is determined by the lifetime of the last hop node.

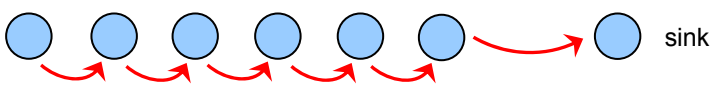

Figure 85: Worst-case network topology.

Figure 86 shows the average power consumption of the last hop node in relation to the measurement interval ($1/\lambda$). The power consumption is higher for networks with more nodes (N) since the last hop node has to forward more data to the sink.

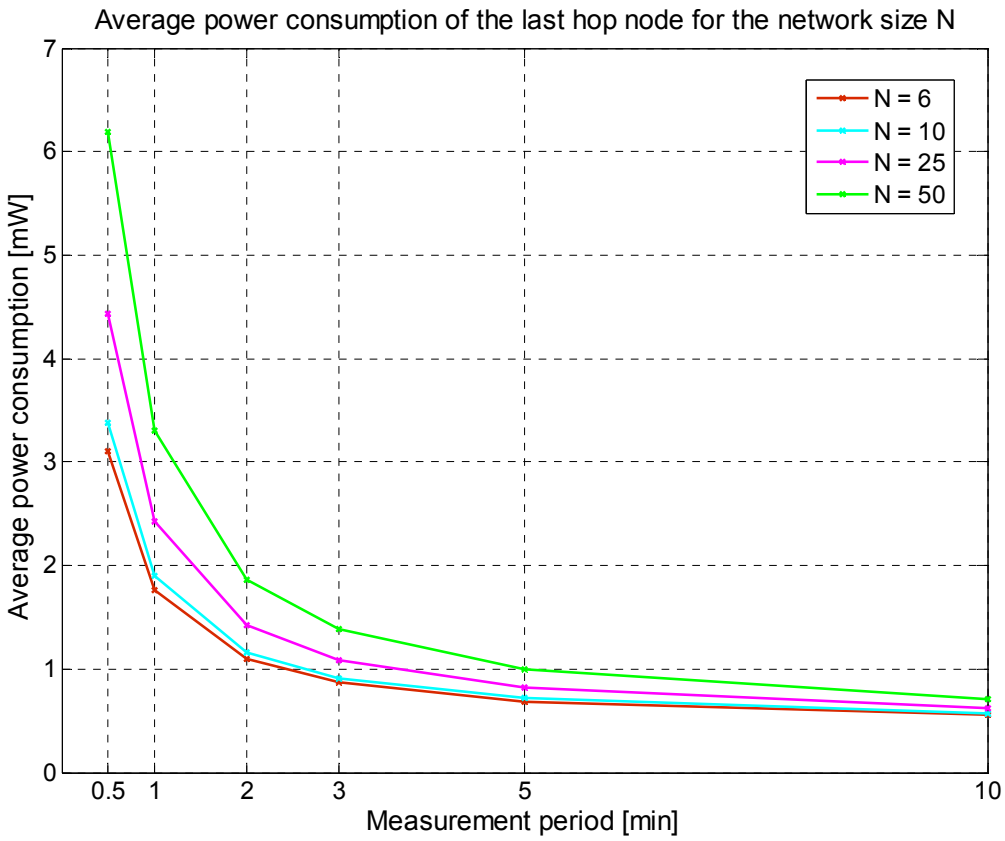

Figure 86: Average power consumption of the last hop node.

The presented model of average power consumption for the last hop node is based on a theoretical model where the connection is considered to be ideal. It is important to emphasize that the model assumes that there are no transmission errors and consequent re-transmissions, and does not take into account the energy consumed by packet overhearing. However, it presents an approximation to provide an upper bound of a network lifetime estimate.

A more realistic model which includes the packet error rate is presented and simulated in the following section 3.4.2.5.

3.4.2 Network Lifetime Estimation

The lifetime of the network is going to be simulated using the previously defined model of the average power consumption of the last hop node and using equation (28). Different scenarios and setups are simulated to obtain the upper bound of the achievable network lifetime.

The battery model used in the simulations assumes ideal operating conditions. It will be based on the SL-2880 battery from Tadiran [49], if not stated otherwise. It is a 3.6 V Lithium battery with 19Ah capacity that features a superior shelf-life with a capacity loss due

to self-discharge of less than 1% per year [50]. The battery size is a standard D-size which easily fits the hardware enclosure (see section 3.3.2, Figure 65).

3.4.2.1 Network Topology Considerations

The lifetime of the last hop node is simulated with the described 19Ah Lithium battery and the chain topology. The result is shown in Figure 87.

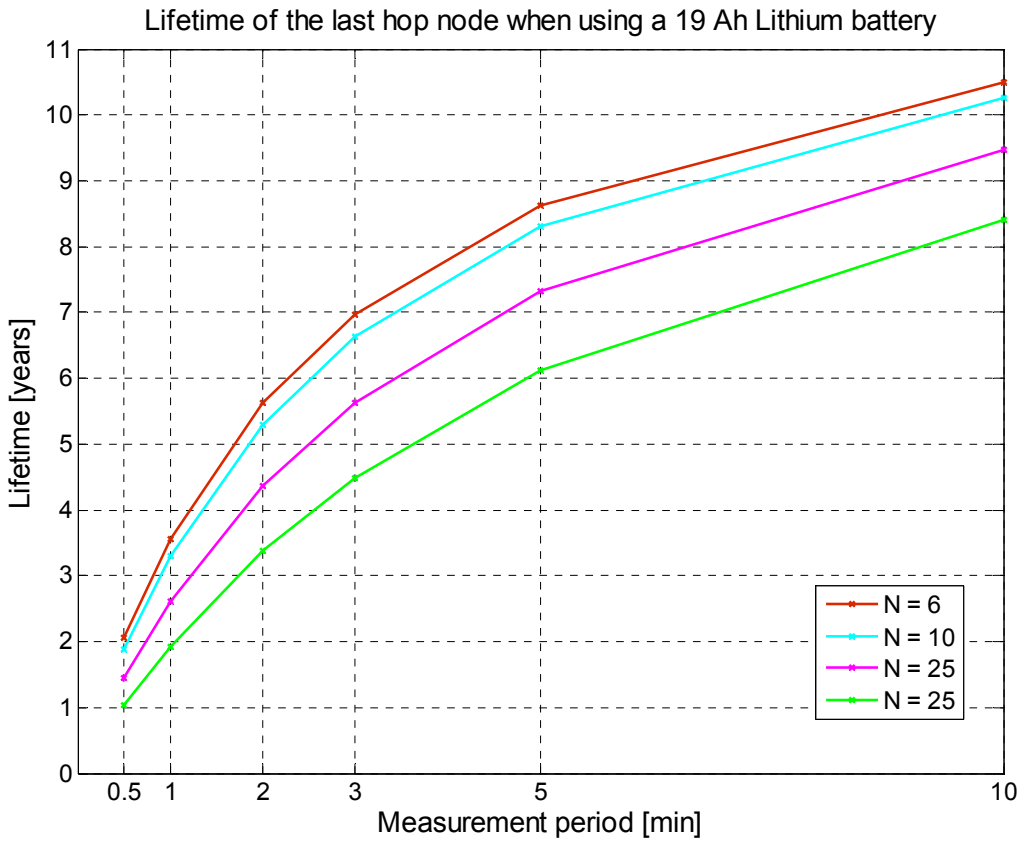

Figure 87: Lifetime of the last hop node the default setup.

As it can be seen, the measurement rate and the size of the network are determining the lifetime of the network. For a network size of 25 nodes, which would correspond to the network size required to fully equipping the Stork Bridge and its 24 cable stays and measurement rate 1 per minute, the maximum lifetime that can be achieved is about 2.5 years. If the measurements would be performed every 5 minutes, the network lifetime could reach about 7 years.

These considerations are for a network with a single last hop node, which means that, for the example of fully equipped Stork Bridge, we are considering that only one of the network nodes establishes connection to the network sink and forwards all the data packets originating from other nodes. This topology can easily be changed to improve the lifetime of the network by introducing several last hop nodes instead of a single one. For instance,

there could be 4 last hop nodes in the 24-node network. One example of such a network topology is illustrated in Figure 88.

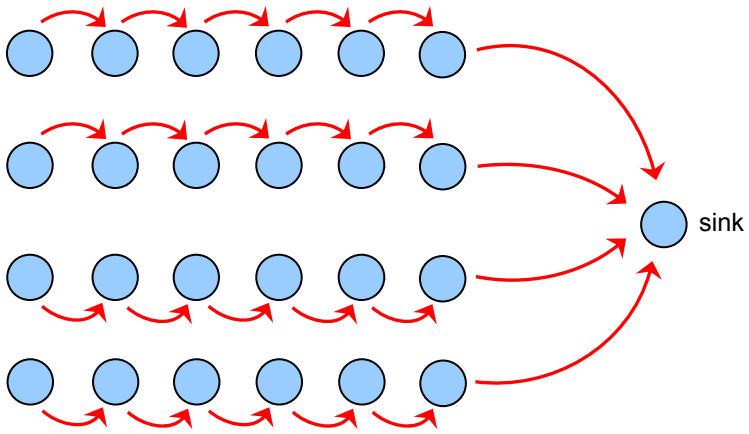

Figure 88: Network topology with four last hop nodes.

The illustrated network topology consists of four 6-node chains, where each chain corresponds to 6 adjacent cables of the Stork Bridge. By forming such a topology, the maximum network lifetime is determined by last hops, but in this case each of the last hop nodes has to forward the data packets originating from 6 nodes instead of 24. Therefore, the lifetime of such a network can be estimated by observing the case of the 6-node network.

According to the lifetime simulation results shown in Figure 87 the network lifetime would increase from about 2.5 to about 3.5 years for the measurement interval of 1 minute, and from about 7.5 to about 8.5 years for an interval of 5 minutes. Such a network topology is going to be presumed for the rest of the simulation scenarios

Assuming that all 24 bridge cables of Stork Bridge and monitored and that the topology is optimised as shown in Figure 88, we can estimate the upper network lifetime bound by observing the curve for 6-node network. With a measurement period of 1 minute a lifetime up to about 3.5 years could be achieved, with a period of 5 minutes up to about 8.5 years, and for 10 minutes up to 10.5 years.

3.4.2.2 Energy Consumption Distribution

The power consumption of the different node components depends on the measurement rate and the number of forwarded packets. Figure 89 shows the energy consumption of the last hop node with a measurement rate of 1 per minute and the 6-node chain network topology.

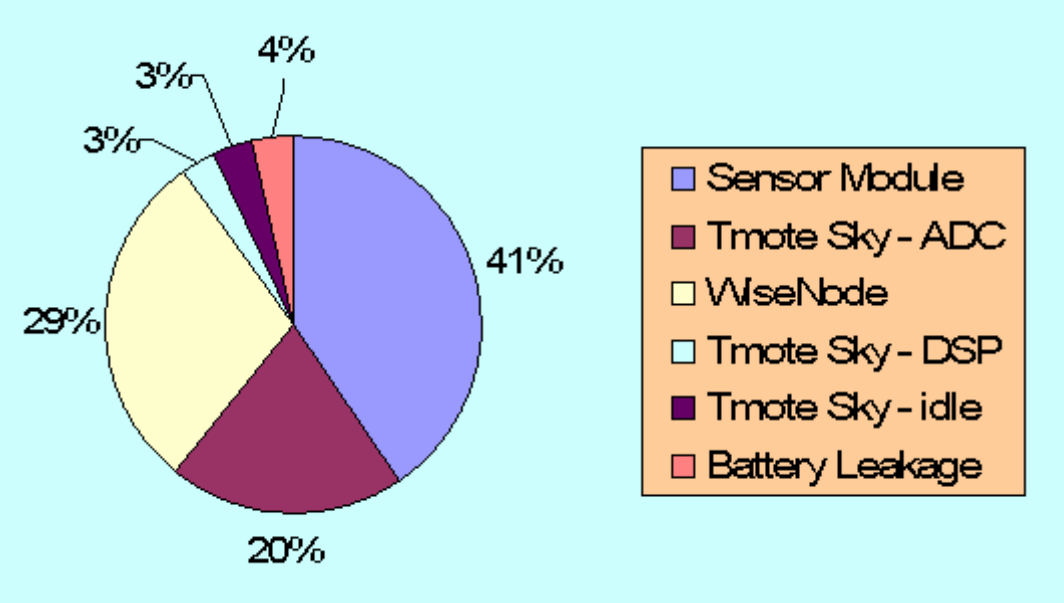

Figure 89: Energy consumption of the last hop node in the case of one measurement per minute and 6-node chain topology.

The most energy consuming module of the last hop node is the sensor board which contributes with 41% to the total energy consumption. The A/D conversion contributes with 20%. This results in a consumption of 61% for the signal acquisition. The wireless communication (WiseNode) consumes 29% of the energy. It is interesting to see is that the signal processing energy consumption is comparable to the idle mode consumption, i.e. 3% each.

The power consumption for measurement periods of 1, 5 and 10 minutes is shown in the network lifetime graph in Figure 90. It can be seen that for decreased measurement rates the percentile power consumption of the measurement acquisition (sensor board and A/D conversion) is decreasing. The major part of the power consumption shifts towards maintaining the wireless communication link. As the overall power consumption goes down with decreasing the measurement rate and the lifetime of the network goes up, the constant idle power consumption and the battery leakage are becoming more and more important. In reality, the consumption contribution of the wireless communication will be larger than the theoretically modelled one, because the model does not include the increased consumption due to packet errors, overhead and broadcast transmissions during the network flood and host command dissemination (see appendix A.1.1.4). The power distribution presented here is however good enough to give us an insight and some ideas.

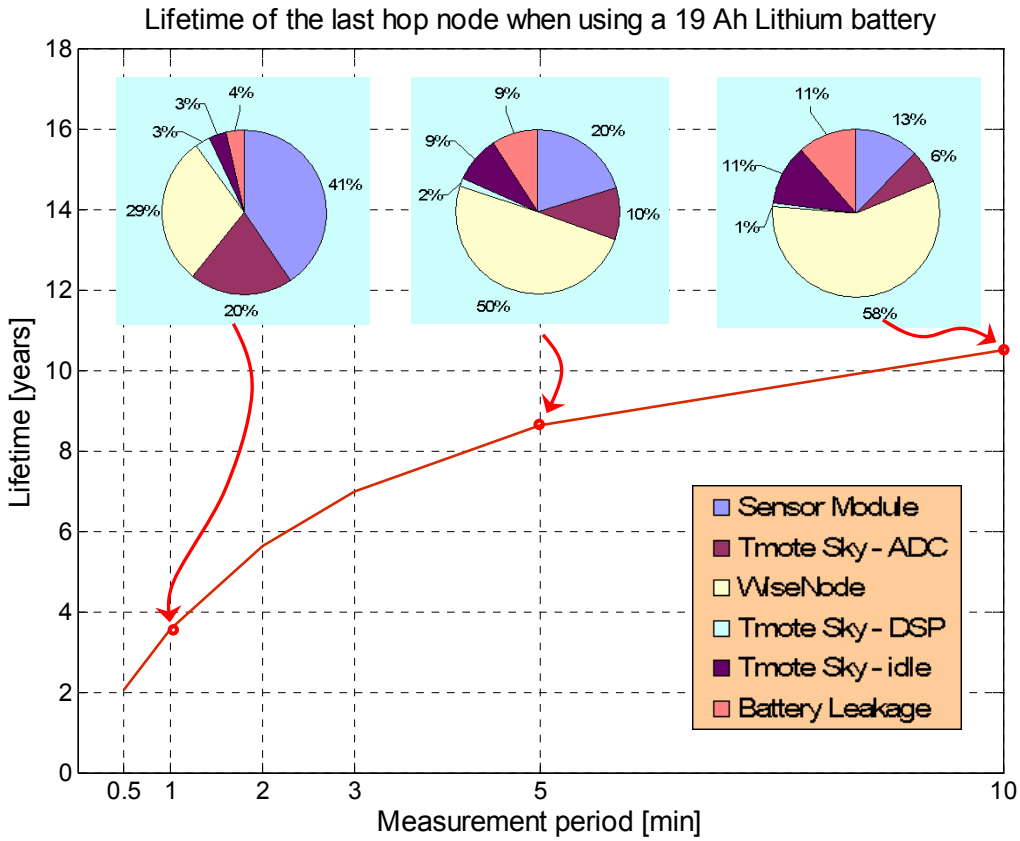

Figure 90: Lifetime of the last hop node together with the characterization of the power consumption for measurement rates of 1, 5 and 10 minutes.

The power distribution will vary from node to node, depending on their location in the network which determines their packet forwarding activities. The presented power distribution is for the node with the highest power consumption, i.e. last hop node. Clearly, the average power consumption of all network nodes will be different because of a different WiseNode activity. The WiseNode contribution on the last hop node is 30% of the total consumption. All other nodes have less packets to forward and the WiseNode consumption will therefore be less than 30% of the overall consumption. This is a rough approximation because a part of the WiseNode consumption goes for the network sampling activity, which is independent of the actual traffic.

In reality, the consumption contribution of the wireless communication will be larger than the theoretically modelled one, because the model does not include the increased consumption due to packet errors, overhead and broadcast transmissions during the network flood and host command dissemination (see appendix A.1.1.4). The power distribution presented here is however good enough to give a rough estimate of the system lifetime.

The measurement acquisition presents the major power consumer when the measurement period is short. This is expected since the measurement acquisition takes about 10 seconds, during which the sensor module is fully powered and the Tmote Sky

board is in the active mode performing the A/D conversion. To further reduce the power consumption of the measurement acquisition, a sensor module with lower power consumption would have to be developed or the duration of the measurement would have to be decreased. Our sensor module (see section 2.1.1.3) presents a low-power solution as it consists of a low-power MEMS acceleration sensor and low-power amplifiers used to build the filtering and conditioning circuitry. Therefore, it is difficult to conceive how the power consumption of the sensor module could be significantly decreased, other than using other components with significantly lower power consumption. The measurement duration depends on the application and therefore cannot be changed without degrading the frequency resolution of the analysis (see section 2.2.1.4).

The energy consumption of the data processing part (DSP) is comparable to the energy loss in the inactive state (idle) state and the battery leakage. For instance, if the measurement period is set to 5 minutes, the energy consumption of the processing part (2%) is smaller than the energy loss due to the battery leakage (9%). We can conclude that the processing algorithms are very efficient in the terms of power consumption and that further algorithm optimization with respect to execution time can not increase the system lifetime significantly.

To see the gain of the advanced processing with respect to system lifetime, it is compared to a standard floating point FFT implementation. In this case, calculating the signal spectrum takes about 10 times longer (see section 2.2.1.4, Table 5). Figure 91 compares the expected lifetime of a system when the FFT is implemented using floating-point operations and when it is realized with integer operations only. The lifetime is plotted for both cases, and the presented power consumption distribution is shown for the floating point FFT implementation.

The power consumption of the signal processing part is about 10 times larger. The system lifetime is about 30% longer with the integer FFT implementation when the measurement period is 60 seconds, enabling the system to reach 4 years instead of 3 years for the floating-point implementation. But when the measurement period is 5 minutes, the lifetime difference is only 10% (9 years instead of 8).

If the measurement acquisition would be less costly in terms of energy, the processing part would figure as a major factor in the power consumption when the measurements are frequent. In the current implementation, optimizing the processing might be considered as uncritical because the gains (10 times smaller consumption) are somehow shadowed by the measurement acquisition consumption.

This consideration has been made just for one part of the processing part, the FFT implementation, while the second part regarding the natural frequency detection algorithm was not changed.

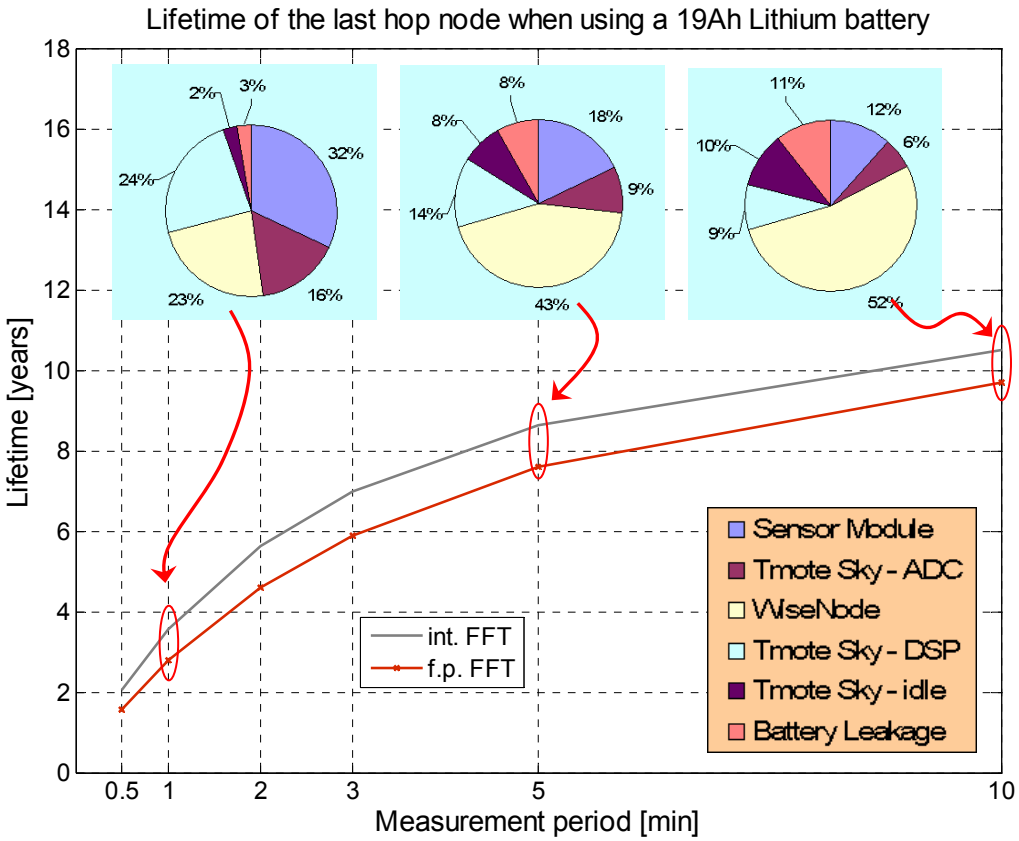

Figure 91: Lifetime of the last hop node for different processing algorithms.

When observing the power consumption of the WiseNode in Figure 90, it can be seen that as the measurement period increases the contribution of the WiseNode to the overall consumption increases and it starts to represent the major part of it (50% at 5 minutes period, and 58% at 10 minutes). It can be argued that if the packets are generated every 1, 5 or 10 minutes, it doesn't make sense for the WiseNodes to sample the network medium 4 times a second as it is in the default case. Increasing the network sampling period (T_W) would decrease the power consumption of WiseNode while increasing the latency of the data packet delivery. The application of monitoring cable tension forces allows the network sampling to be less frequent. The increased latency is tolerable in this application.

Figure 92 shows the lifetime of the network for different network sampling periods (0.25, 1 and 5 seconds) together with the power consumption distribution for the simulated network sampling periods in a case when measurements are performed every 5 minutes.

As it can be seen from the network lifetime in Figure 92, reducing the sampling period would make sense for longer measurement periods. For a measurement period of 1 minute it can be seen that the gain is not significant. For a 5 minute measurement period, the lifetime is raised from about 8.5 years to 11 years (~+30%) when T_W is changed from 250 ms to 1 second. For a network sampling period of 5 seconds the lifetime rises to about 12 years (~+40%).

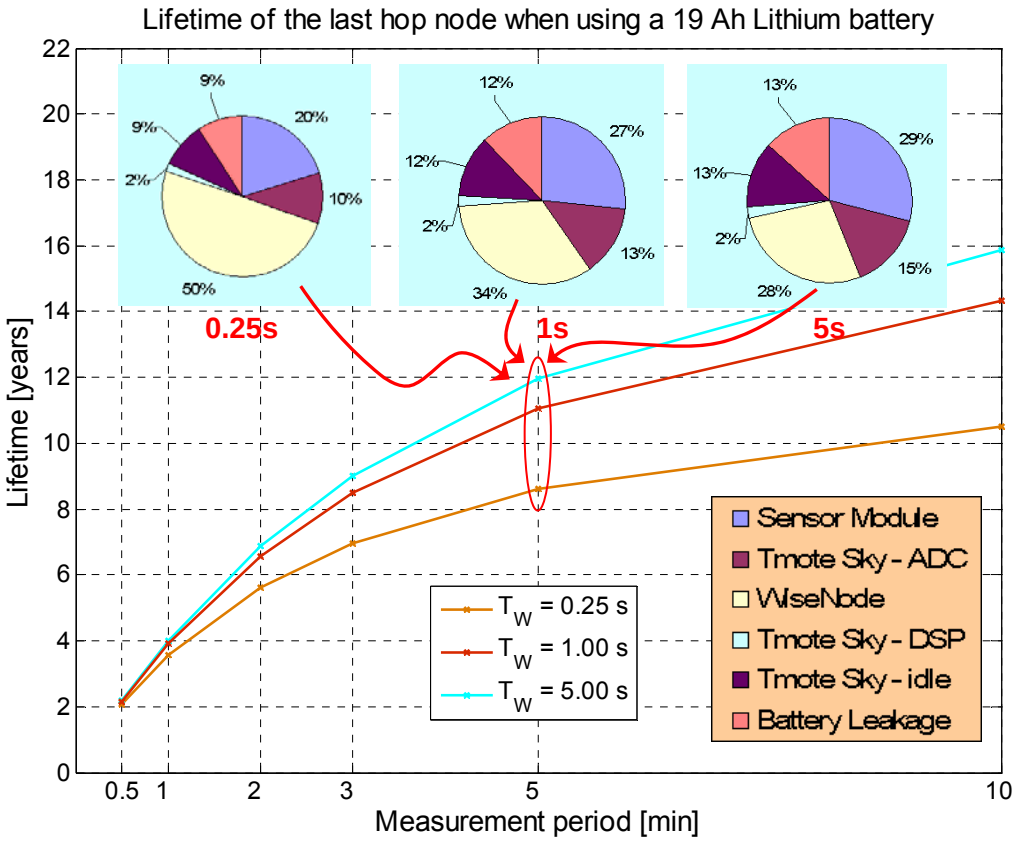

Figure 92: Lifetime of the last hop node for different network sampling periods.

However, these considerations do not take into account the broadcast transmissions. The energy consumption from the broadcast transmission is directly related to network sampling period since the broadcast transmissions have to be longer to match the network sampling period. Broadcasts are used in the network flood mechanism to disseminate the network management commands and host commands (see appendix A.1.1.4 and A.1.1.5). This would proportionally increase the energy costs of the broadcast transmissions, e.g. by 4 times if the network sampling period is increased from the default 0.25 to 1 second.

Hence, it is possible that the network lifetime could even be degraded when reducing the network sampling if the network floods are used relatively often. This leads to the conclusion that if the network sampling period is increased by X times, the network floods would have to be used X times less frequently to assure that the network lifetime will not be degraded. Therefore, the decision to reduce the network sampling period has to include the considerations about the intended rate of broadcast transmissions.

3.4.2.3 Alkaline Battery Power Supply

Alkaline batteries have a bigger self-discharge than lithium batteries. The network lifetime was also simulated with alkaline batteries to see how this reflects on the system

lifetime. The battery model for the simulation is based on the E95 Energizer alkaline battery (D-size, 1.5V) which has a capacity of 20.5 Ah and a self-discharge rate of about 3% per year [58]. Figure 93 shows the comparison of system lifetimes achievable when using alkaline batteries instead of the 19Ah Lithium battery used in the previous simulations. As it can be seen, the network lifetime does not differ significantly. It would therefore be reasonable to use alkaline batteries to reduce system costs costs.

Alkaline batteries do not perform well at low temperatures. According to the datasheet it operates down to -16°C while the lithium battery still operates at -55°C. The remaining battery capacity drops significantly at lower temperatures for alkaline batteries [51]. Another reason why lithium batteries were selected for this project is the minimum voltage level of 2.7 V (2 x 1.35 V) during network reprogramming. The fully charged lithium battery voltage is 3.6 V compared to 3 V (2 x 1.5 V) for the alkaline one, what gives a larger operational voltage range. Discharge characteristics of the lithium batteries are by far superior to alkaline batteries [51].

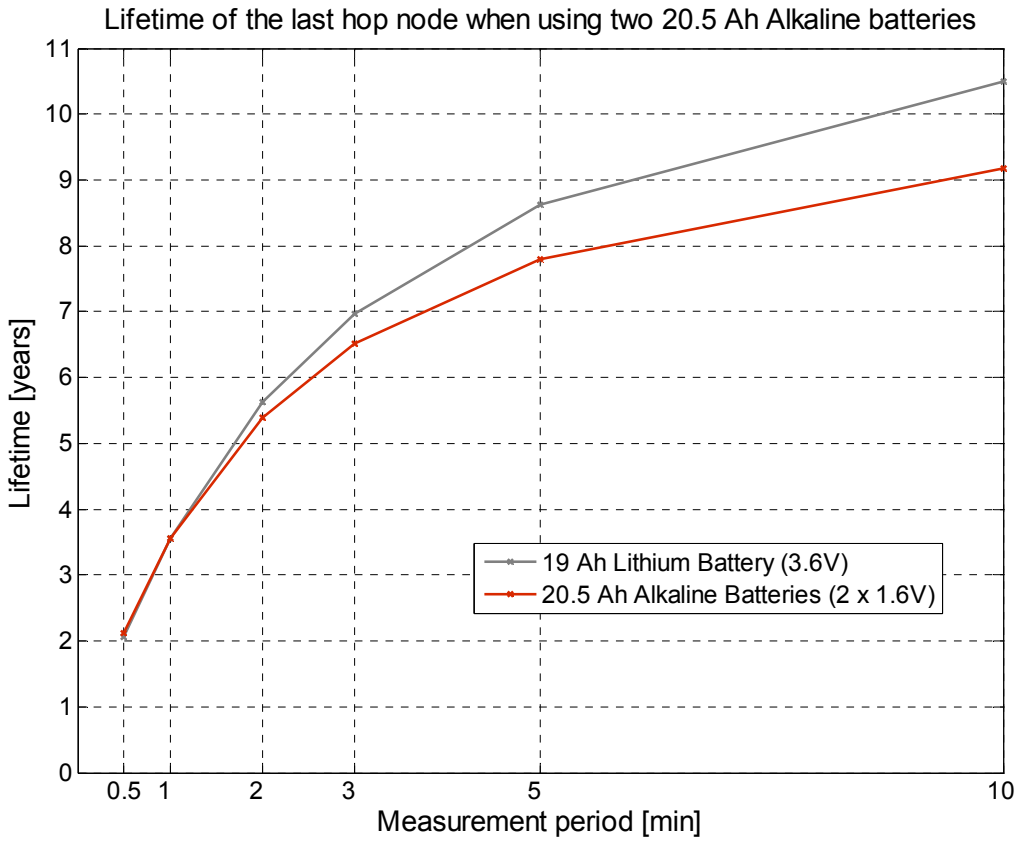

Figure 93: Lifetime of the last hop node with selected alkaline battery.

3.4.2.4 Installed Power Supply

The currently installed power supply consists of two independent battery power sources, one for the measurement and processing module (Tmote Sky + sensor module), and the other one for the WiseNode module (see 3.3.3.2). Each source consists of two Energizer L91 AA-size lithium batteries with a nominal voltage of 1.5V and a capacity of 3000 mAh [51].

The lifetime of the node will therefore be determined by the power source that discharges the first. Thus, both of the subsystems were simulated independently to get the maximum lifetime of each of them. So instead of using expression (13) as power consumption model, the simulation was run once using the power consumption given by expression (14) and thereafter using expression (15). The results of these two simulations are plotted in Figure 94.

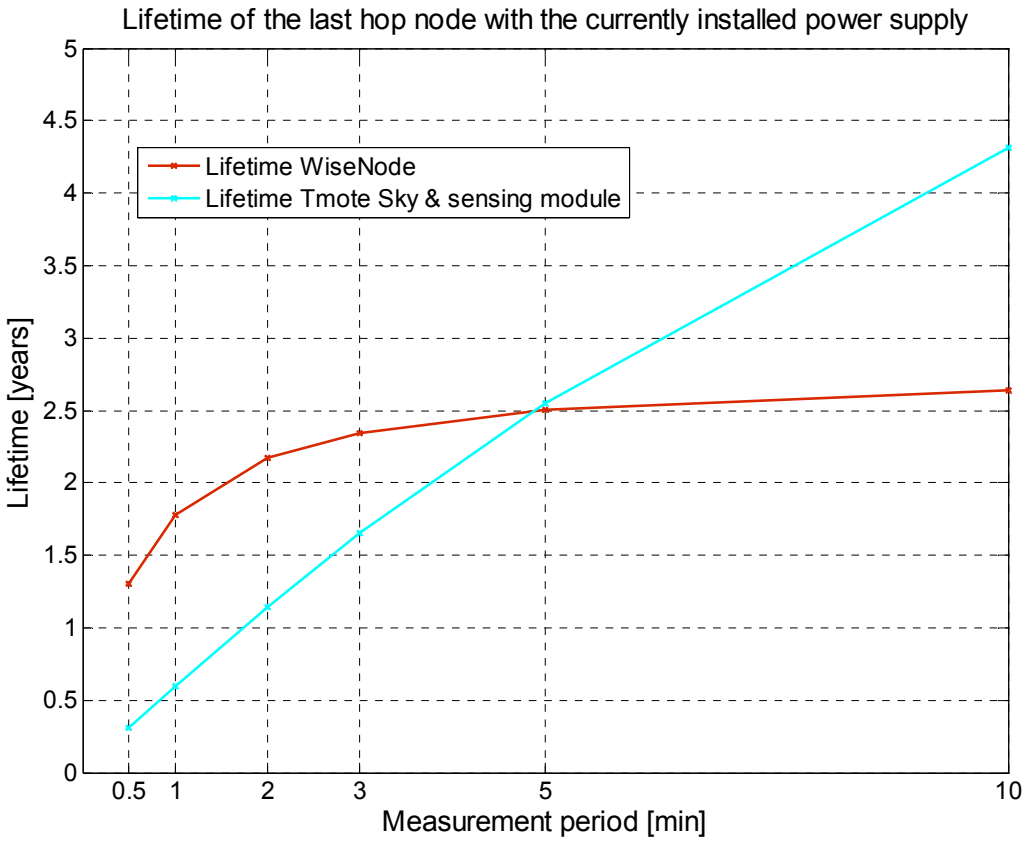

Figure 94: Lifetime of the last hop node for the currently installed power supply system.

As it can be seen, the lifetime of the Tmote Sky and the sensor module is almost linear with the measurement rate. It is because the power consumption of this part is determined exclusively by the measurement acquisition and processing, and the idle energy consumption which causes the slight decrease in the lifetime slope. For shorter measurement periods the WiseNode consumption is determined by the power consumption

used to forward the data packets and for longer periods by the network sampling consumption.

If using separated power supplies for measurement processing and communication the network lifetime is determined by the supply that exhausts first. The lifetime of the Tmote Sky and sensor module is matching the lifetime of the WiseNode module at a measurement period of approximately 5 minutes at which it reaches 2.5 years. For shorter measurement periods, the system lifetime equals the lifetime of the measurement and processing subsystem (Tmote Sky + sensor module), for longer measurement periods, the lifetime of the system is determined by the lifetime of the WiseNode. It should be noticed that with low measurement rates, the network sampling period may easily be increased leading to higher lifetimes (see Figure 92).

3.4.2.5 Packet Error Rate Modelling

The packet error rate is one important factor to describe the efficiency of the MAC protocol with regards to energy, since packet transmission retries present an unwanted energy overhead. A packet transmission retry can be very costly in terms of energy consumption since repeated transmissions are done with a long preamble after 3 consecutive failures. Long preambles are necessary to compensate for the presumed lack of synchronisation when unsuccessful transmissions persist.

In this analysis, we are going to modify the existing simple power consumption model to include the increased power consumption due to packet retransmissions. After an unsuccessful packet transmission the WiseNode first attempts to resend the packet 2 times with a shorter preamble (70 ms), and then if it is still not successful the transmissions are repeated with a longer preamble that equals the network sampling period (250ms).

In this consideration, we are going to be conservative and consider that every retransmission is performed with a long preamble. By modifying equation (26) from appendix A.4, we can write forwarding energy in the case of retransmission to be:

$$E_{Fwd\_TX\_Error} = E_{FwdMin} + (P_R / 2 + P_T) \cdot T_{TX\_Error}, \tag{18}$$

where T_{TX_Error} is a duration of the long preamble (250ms). To include it in the model of average power consumption of WiseNode, we can modify equation (14) to:

$$P_{WN} = P_Z + P_{LPL} + N\lambda \cdot \left(E_{Fwd} + E_{Fwd\_TX\_Error} \cdot K_{PER}\right), \tag{19}$$

where K_{PER} is the PER.

Using this simple model we can simulate the lifetime of the network to see how the PER influences the lifetime of the network. Figure 95 shows the results of the simulation when using a single 19Ah power supply. Table 10 shows the achievable lifetime in years for different measurement periods and packet error rates.

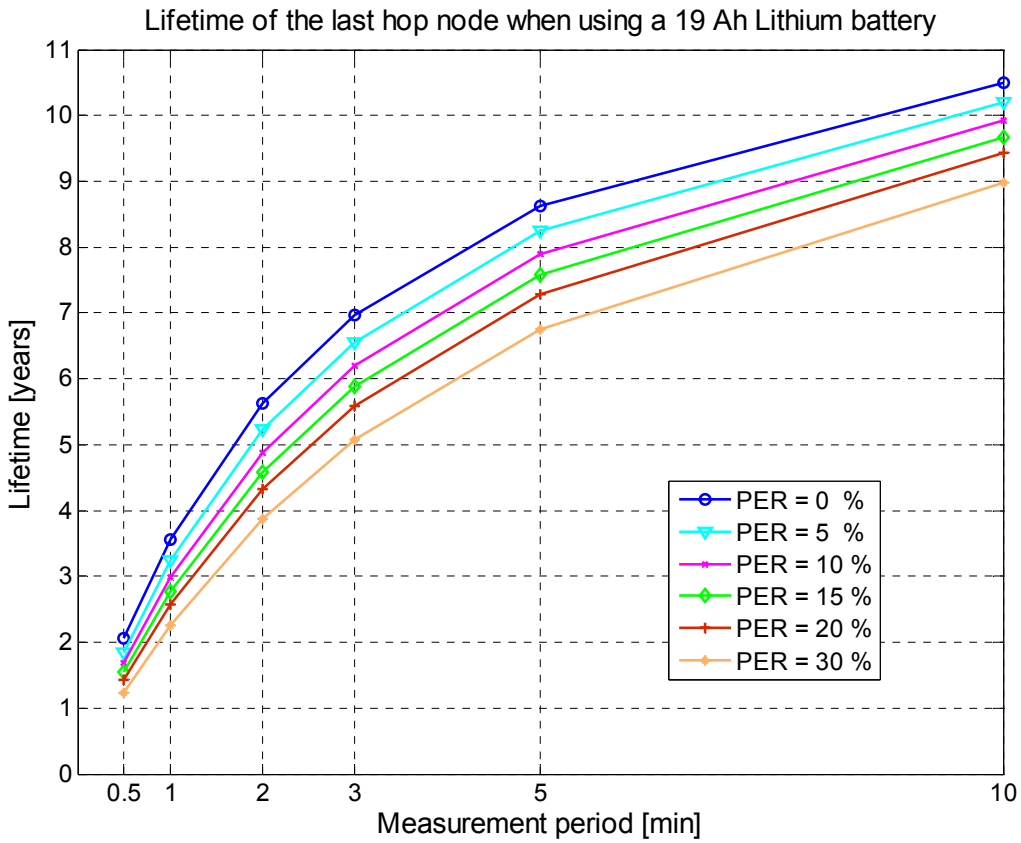

Figure 95: Lifetime of the last hop node for different packet error rates.

Network Lifetime [years]	**Packet Error Rate**					
Measurement Period [s]	**0%**	**5%**	**10%**	**15%**	**20%**	**30%**
30	1.7	1.6	1.5	1.4	1.3	1.1
60	3.1	2.9	2.7	2.5	2.3	2.1
100	4.5	4.2	3.9	3.7	3.5	3.1
200	6.7	6.4	6.1	5.8	5.5	5.1
300	8.1	7.7	7.4	7.1	6.9	6.4
600	10.1	9.8	9.6	9.3	9.1	8.7

Table 10: Network lifetime in years for different packet error rates.

So far the idealized model was simulated presuming ideal link conditions in terms of transmission errors. This ideal case is presented here for a PER of 0%. The lifetime of the

network is more affected by the PER if the measurement rate is higher and therefore forwarding is more frequent. For instance, for a measurement period of 1 minute, the lifetime of the network would be degraded from about 3.5 years in the ideal case to about 2 years in the case of 30% PER, which gives a decrease of lifetime in this case of about 40%. For a measurement period of 5 minutes the system lifetime only decreases about 20%.

According to the network diagnostics data and PER analysis in section 3.3.6, it can be expected that the PER on the bridge deployment ranges from 5% to 15%. Therefore these values should be observed when estimating the lifetime of the developed system. As it can be seen, this would correspond to a lifetime decrease of 10% to 20%.

Longer preambles do not only increase the energy consumption of the transmitting node but also the one of nodes in the transmission range. This is because of the increased probability that they are going to be awoken by the longer preamble of the packet and listen to a whole packet which is not intended for them (idle listening). Because of the difficulty of modelling such effect since it depends on a certain network deployment, this effect was not taken into account.

With the present PER in the field deployment (up to max 15%, see appendix F.1), the network can still achieve a reasonable system lifetime. Improvements in the WiseNode firmware should allow bringing the PER on the bridge down to the values achieved in the laboratory (up to 5%, see appendix E).

3.4.3 Deployed Network

To estimate the lifetime of the deployed network on the Stork Bridge with a dual power supply and four 1.5V 3000mAh Lithium batteries, the approach from section 3.4.2.4 is used. The only difference is that the power consumption model for the WiseNode is going to incorporate the PER equation (19). The results of these two simulations are plotted in Figure 96. Table 11 shows the achievable lifetime in months of the currently installed system for different measurement periods and packet error rates.

If we observe the case with PER of 5%, the lifetime of Tmote Sky and sensor module is matching the lifetime of the WiseNode at a measurement period of approximately 2.5 minutes. For shorter measurement periods, the system lifetime equals to the lifetime of the measurement and processing subsystem (Tmote Sky + sensor module), for longer measurement periods, it is determined by the lifetime of the WiseNode.

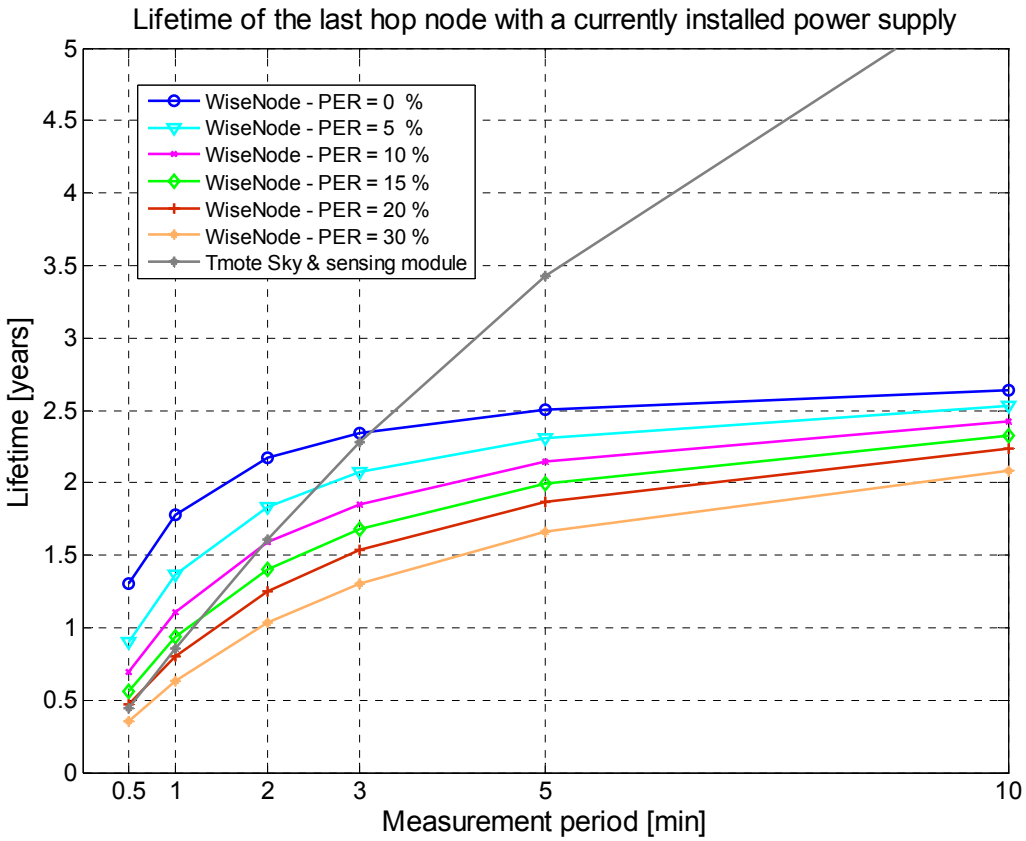

Figure 96: Lifetime of the last hop node of the currently deployed network for different packet error rates.

Network Lifetime [months]	**Packet Error Rate**					
Measurement Period [s]	**0%**	**5%**	**10%**	**15%**	**20%**	**30%**
30	4.3	4.3	4.3	4.3	4.3	4.3
60	8.3	8.3	8.3	8.3	8.3	7.6
100	13.5	13.5	13.5	13.5	13.4	11.0
200	25.0	25.0	23.1	21.0	19.3	16.6
300	30.1	27.8	25.8	24.0	22.4	19.2
600	31.8	30.4	29.1	28.0	26.9	25.0

Table 11: Network lifetime [months] of the deployed system for the different packet error rates.

In the current setup, the measurements are acquired every 60 seconds. In this case the lifetime of the network is determined by the lifetime of the measurement and processing subsystem, except for the simulated case of 30% of PER. This lets conclude that even with a bad WiseNode performance in the terms of the PER, the network lifetime will reach approximately 8.3 months. If considering sending measurements every 5 minutes, the lifetime of the system is determined by the WiseNode, and will reach about 19 months for a

PER of 30%. A PER of 5% could be achieve in the laboratory network. Improvements in the WiseNode firmware will bring the PER on the deployed network down to this value (see 3.3.6.3).

The network lifetime provided by this simulation is an estimation of the upper lifetime bound. The main deficiency of this model is that the overhearing effect is not taken into account. However, the PER model is conservative as it presumes that every packet error rate causes long retransmissions preambles, although the first two attempts are performed with a short ones. Additionally, network floods used to disseminate network management commands and host commands are not modelled (see section 3.4.2.5).

3.5 Field Test Data

The data from the deployed WSN on the Stork Bridge is received at the control centre and saved to log files. The natural frequencies (modes) will be presented in this section. Figure 97 shows the first detected natural frequency of the cable-stay C2. The measurement was taken on January 25th 2008. Each dot represents frequency of the 1st mode of the cable detected by the algorithm.

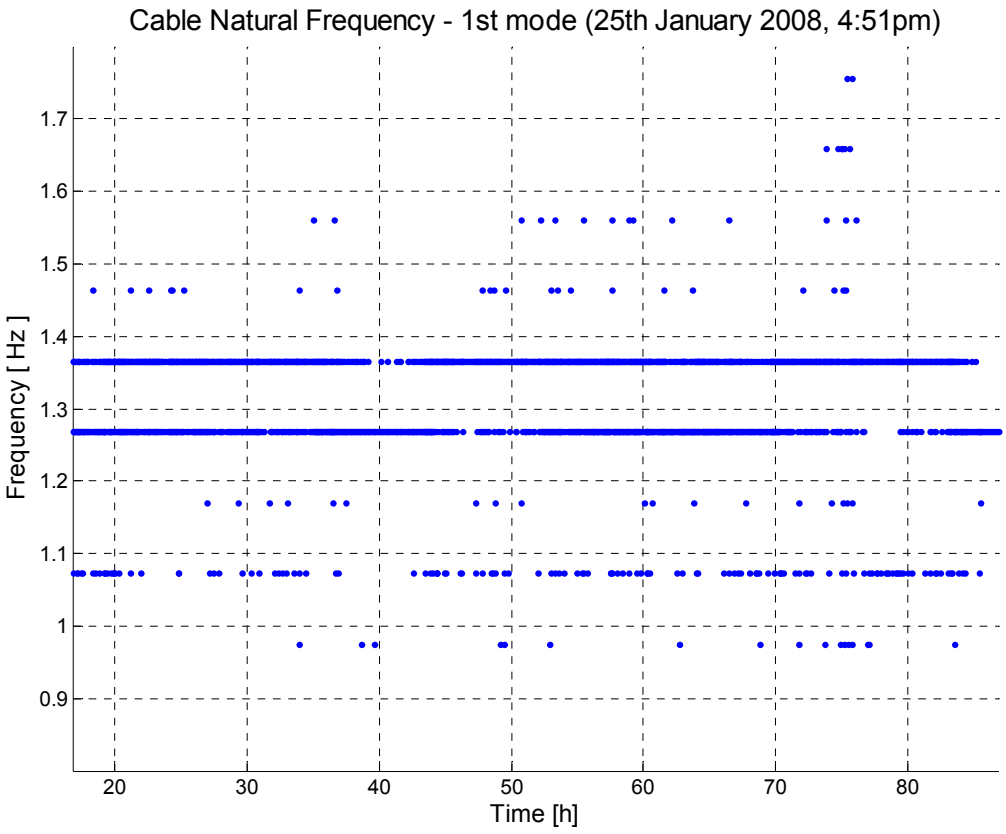

Figure 97: Detected 1st natural frequency of stay-cable C2.

Two main frequency values can be identified at 1.27 and 1.37 Hz. The other scattered dots could be caused by the imperfection of the peak detection method and needs further investigation. The detected natural frequencies of 1.27 and 1.37 Hz in fact are the 13th and 14th frequency sample of the FFT spectrum. The difference of 0.1 Hz corresponds to the resolution of the implemented algorithm. The frequency resolution depends on the number of samples and the sampling frequency:

$$\Delta f = \frac{F_S}{N_S}, \tag{20}$$

where F_S is the sampling frequency and N_S is the number of samples. The implemented 1024-sample algorithm and the sampling frequency of 100 Hz result in a frequency resolution of about 0.1 Hz. To increase the resolution in the frequency domain, the number of samples must be increased or/and the sampling frequency must be decreased. However, the sampling frequency is usually determined by the characteristics of the monitored signal and cannot be lowered under a certain value.

On the other hand, increasing the frequency resolution would mean increasing the measurement duration t_M according to the relation:

$$t_M = N_S \cdot T_S = N_S \cdot \frac{1}{F_S} = \frac{1}{\Delta f}. \tag{21}$$

Since the power consumption is directly related to the measurement duration, frequency resolution and energy consumption has to be traded-off.

When estimating relative changes in cable tension, for instance by using the cable model from section 1.4, the relative change of the natural frequency has to be known. For small changes in natural frequencies, the relative change in cable tension can be estimated by using (8). Thus, the frequency resolution directly determines the precision of the relative change in cable tension that is possible to detect:

$$\frac{\Delta T}{T_0} \approx 2 \cdot \frac{\Delta f_1}{f_{10}}. \tag{22}$$

If we set the initial value of the monitored natural frequency to 1.32 Hz what corresponds to the average of 1.27 and 1.37 Hz from Figure 97 and the frequency resolution is 0.1 Hz, the minimum detectable relative change in frequency would be approximately 7.6%, which according to expression (22) would correspond to about 15% relative change in cable tension. This value, being the minimum relative change in cable tension that can be detected might not be satisfactory.

A simple solution to mitigate this problem is to monitor a higher natural frequency. Frequencies of higher modes can be expressed as multiples of the first mode frequency:

$$f_k = k \cdot f_1 . \tag{23}$$

Using this we can modify expression (22) to:

$$\frac{\Delta T}{T_0} \approx 2 \cdot \frac{\Delta f_k}{f_{k0}} = 2 \cdot \frac{\Delta f_k}{k \cdot f_{10}} . \tag{24}$$

By monitoring the *k-th* natural frequency, the smallest detectable relative change in cable tension becomes *k* times smaller. For instance, if we would monitor the third natural frequency instead of the first one, we would get a three times better resolution of the relative change of the cable tension. Currently the installed system monitors the 3rd natural frequency what results in relative cable tension resolution of about 5%.

Figure 98 shows the 3rd natural frequency on June 6th 2008 (for additional plots see appendix F.2). The measurements were performed every minute. Nodes C1 and C3 were not operational at that moment.

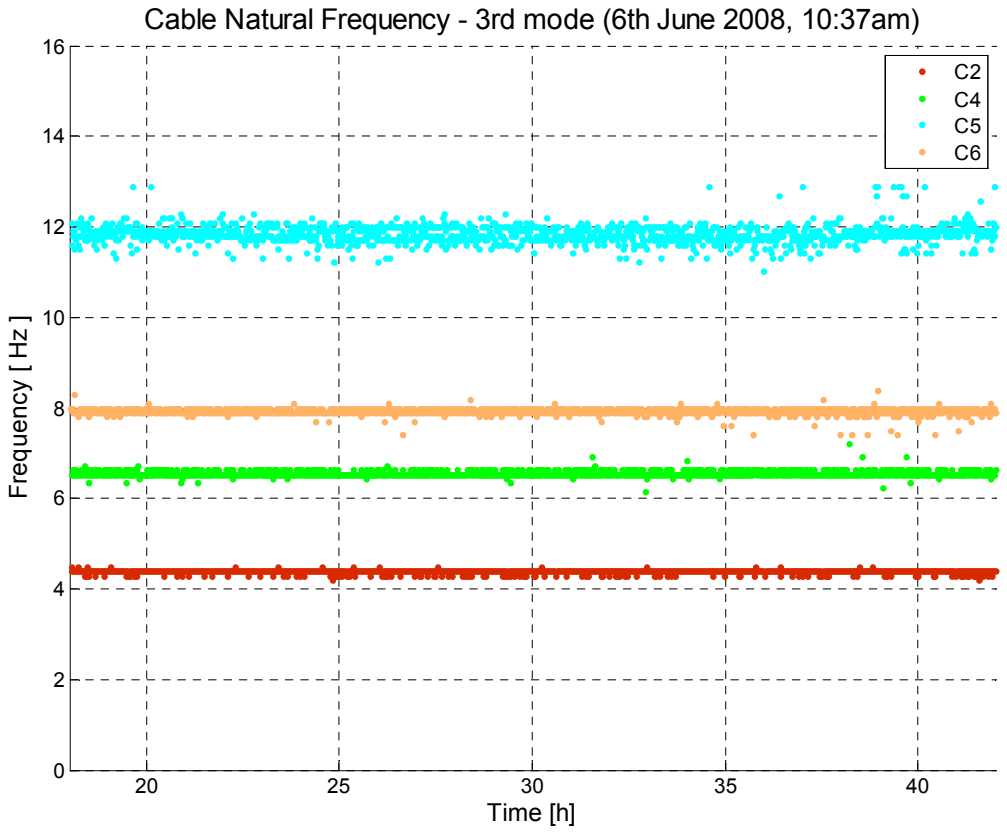

Figure 98: Detected 3rd natural frequency of stay-cables at Stork Bridge.

The scattering of the detected frequency is not as big as the one in Figure 97. The reason is that the 3rd mode is more excited than the 1st one and the algorithm can perform the detection more accurately. Cable C5 shows significantly bigger scattering compared to the other cables. One reason for the different behaviour of cable C5 could be the fact that it is made of carbon fibre reinforced plastic.

During day time the ambient vibrations are larger than during night time. This is most probably due to varying traffic loads. This results in a changing scattering pattern over a day. During day time cables are more excited and natural frequency peaks in the spectrum are well defined and above the noise level. This enables detecting the peaks more reliably. During night time when there is little traffic on the bridge the detected frequencies scatter more. This can be seen in Figure 99 which shows more scattering in the period from around 25 and 32 hours, which corresponds to time between 1am to 8am on April 26th 2008.

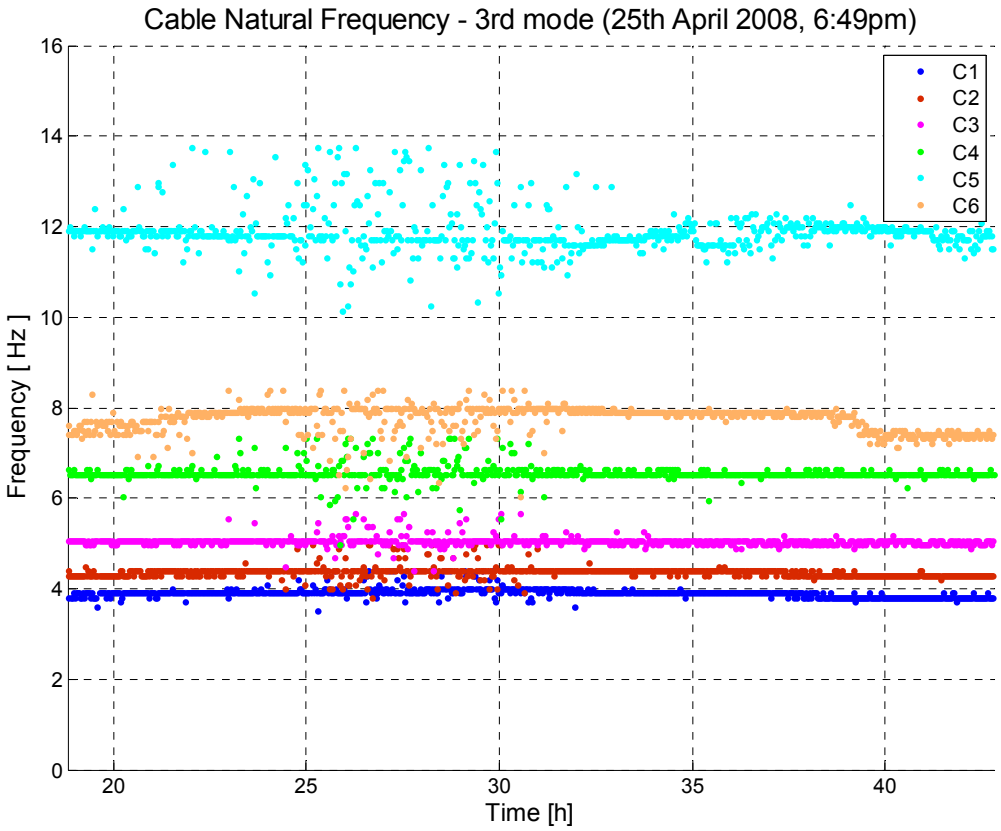

Figure 99: Detected 3rd natural frequency of monitored stay-cables.

A possible solution would be to introduce a criterion which compares the amplitude of the detected peak and the noise level in the spectrum. If the level of the detected peak is not sufficiently above the noise level, another peak corresponding to a cable vibration mode that more excited could be selected and monitored. This would provide a sort of reliability estimate of the peak detection.

4 Conclusion

Monitor large civil structures like bridges for long periods of time, or even their entire life, without maintenance is becoming a need for safety and maintenance reasons. Since many structures are reaching their life time end the need for monitoring systems and condition assessment procedures will increase in the near future. Traditional techniques have so far been limited to short-term monitoring or diagnosis applications and are associated to high costs of purchase, installation and operation.

This project explored the feasibility of using low power wireless networks in conjunction with low-cost sensors and sophisticated signal processing techniques as a means to provide an economical monitoring tool for large civil structures that can operate for years with little maintenance. The longevity of the network was reached by a careful system design. Low-power operation has been achieved using the WiseNET platform providing a low-power wireless networking solution, by applying low-power MEMS sensing and signal conditioning components and by implementing efficient processing algorithms. Based on the first experiences from the field deployment and according to the network lifetime simulations, the developed system has the potential to achieve several years of maintenance-free operation.

The results clearly show that the objective of long term maintenance free operation may be achieved with the technology developed within this project. This is demonstrated by the implemented cable tension force monitoring application deployed on a cable stayed bridge which was operated over a few months. The analysis of the results shows that, depending on the measurement frequency, maintenance-free operation up to 10 years may be reached. The system is modular, allowing integrating different types of sensor and processing modules, and very easy to install due to the absence of wires and the network self configuration. Remote monitoring is provided using the public mobile telephone network. A variety of configuration and observation software tools have been implemented to provide easy data access and observation methods.

This first and important step demonstrates the feasibility of long-term maintenance-free monitoring of civil structures such as bridges, tunnels or power plants. A number of problems are still open that call for further research and development. Among those, there are long-term reliability, extension to large networks and the design of deployment, commissioning and observation tools that can be used by non-specialists. At longer terms, sensor miniaturisation including communication electronics will allow to insert the wireless sensors directly in the structure at construction time.

Appendixes

A. WiseNET

B. Screenshots of the Control Centre Operator Console

C. Sensor Board Design

D Analysis Results from the Indoor Test at the Laboratory Bridge

E. Performance Analysis of the Laboratory Network

F. Performance Analysis of the Deployed WSN

A. WiseNET

This part presents the contribution of CSEM in the project “Low Power Wireless Sensor Network for Monitoring Civil Infrastructure”.

A.1. Introduction

The contribution of CSEM in this project is in the wireless communication infrastructure (wireless communication hardware modules and protocols). The WiseNET communication platform has been adapted and optimized for the civil infrastructure monitoring application.

The contributions made by CSEM within this project are:

- design and implementation of a network wide synchronization scheme
- design and implementation of the host flood mechanism
- debug and optimization of the WiseNET stack
- design and realization of the WiseNode_V4 boards

A.2. WiseNET Protocol Stack

A.2.1. Overview

The WiseNET communication protocol stack, illustrated in Figure 100, is composed of a radio layer implementing the transmission and reception of frames, a MAC layer implementing WiseMAC and a routing layer, routing packets following either predefined or dynamically learned routes.

On top of the three communication layers are the HCI layer and/or an embedded application. The HCI layer is meant for communication with a host controller. On the sensor nodes, the host controller performs measurements (including the required signal processing) and generates data packets that shall be forwarded to the sink. On the sink node, the host controller is the data collection computer.

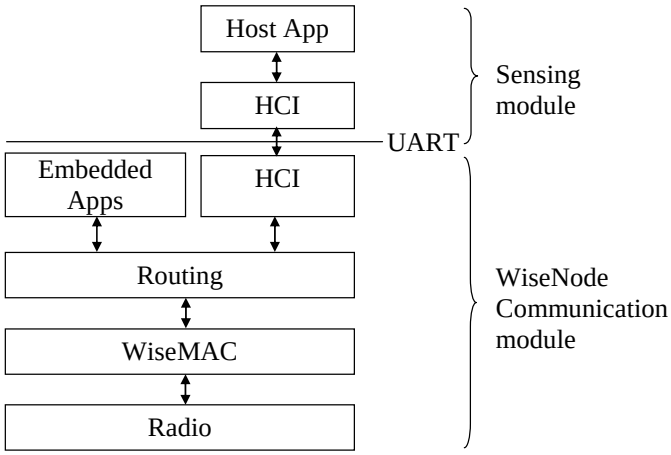

Figure 100: WiseNET Protocol Stack.

A.2.2. Host Controller Interface

The Host Controller Interface (HCI), documented in [46], is a convention to transmit commands and events between a host controller and an embedded device across a serial interface.

A.2.3. MAC Layer - WiseMAC

WiseMAC [52][53][54] is based on the preamble sampling technique. This technique consists in regularly sampling the medium to check for activity. By *sampling the medium*, it is meant listening to the radio channel for the duration required to measure the received power (i.e. a few symbols). All sensor nodes in a network sample the medium with the same constant period[13]. Their relative sampling schedule offsets are independent. If the medium is found busy, a sensor node continues to listen until a data frame is received or until the medium becomes idle again. For the first transmission towards some destination, the transmitter adds a wake-up preamble of size equal to the sampling period in front of the data frame to ensure that the receiver will be awake when the data portion of the packet arrives. The WiseMAC acknowledgement packets are not only used to carry the acknowledgement for a received data packet, but also to inform the other party of the remaining time until the next sampling time. In this way, a node can keep a table of sampling time offsets of all its usual destinations up-to-date. Using this information, a node

[13] If different wake-up periods are used among the network nodes, a maximum period should be defined for the first contact, and the wake-up period duration should be added in the acknowledgement message.

transmits a packet just at the right time, with a wake-up preamble of minimized size, as illustrated in Figure 101.

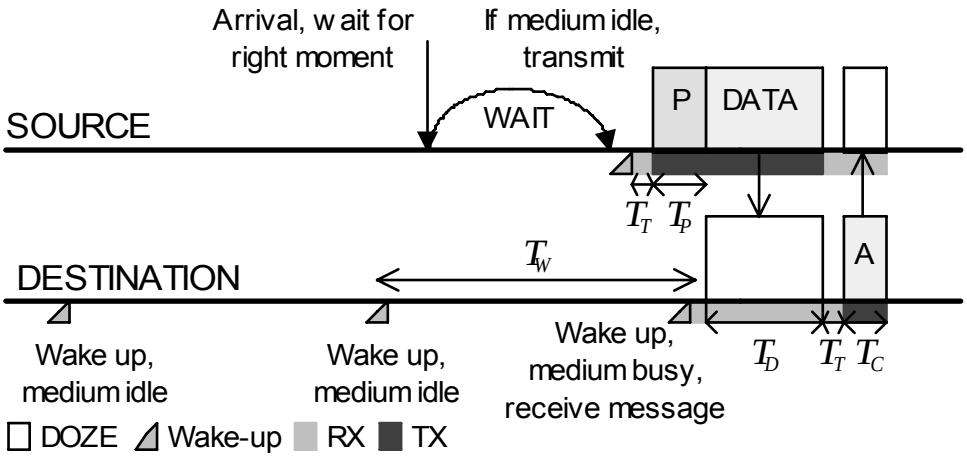

Figure 101: Principle of operation of WiseMAC.

The duration of the wake-up preamble must cover the potential clock drift between the clock at the source and at the destination. This drift is proportional to the time since the last re-synchronization (i.e. the last time an acknowledgement was received).

The synchronization mechanism of WiseMAC can introduce a risk of systematic collision. Indeed, in a sensor network, a tree network topology with a number of sensors sending data through a multi-hop network to a sink often occurs. In this situation, many nodes are operating as relays along the path towards the sink. If a number of sensor nodes try to send a data packet to the same relay, at the same scheduled sampling time and with wake-up preambles of approximately identical size, there are high probabilities to obtain a collision. To mitigate such collisions, it is necessary to add a medium reservation preamble of randomized length in front of the wake-up preamble. The sensor node that has picked the longest medium reservation preamble will start its transmission sooner, and thereby reserve the medium.

Additional schemes include: the repetition of data frames in long preambles to minimize overhearing, the transmission of data packets in bursts thank to “more to come” indication in the packet header, the avoidance of interrupting data-ack transactions through a mandatory inter-frame space larger than the data-ack interval, the selection of a receive threshold well above the noise floor to minimize useless wake-ups and the selection of a carrier sensing threshold below the receive threshold to mitigate the hidden node effect.

A.2.4. Communication Services

The following communication services are available:

- Low power multi-hop data transmission
- Routing
- Network synchronization
- Host flood
- Network management (control and monitoring)

They will be detailed in the next sections.

A.1.1.1. Low power multi-hop data transmission

Data packet are generated by the sensing modules and forwarded transparently by the WiseNode modules to the data collection computer (see **Figure 102**). Transmissions are made using the WiseMAC low power medium access control protocol which has been designed at CSEM specifically for low power multi-hop wireless networks. This protocol is today, to our knowledge, the most energy efficient medium access control protocol for low traffic multi-hop transmissions.

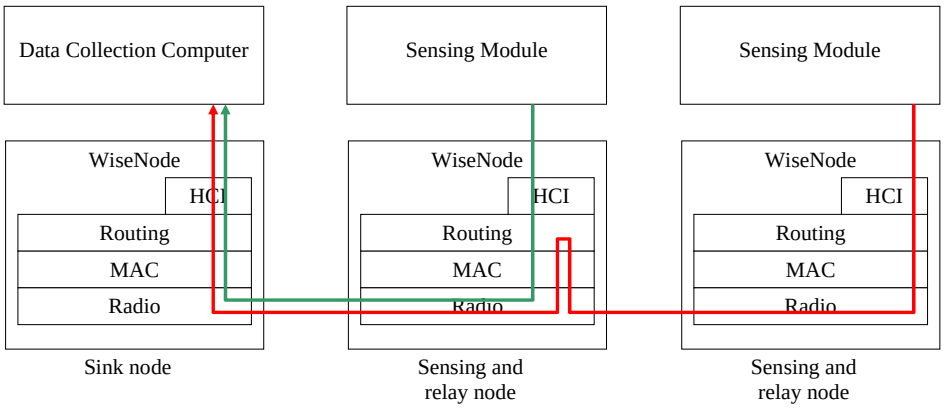

Figure 102: End-to-end transmission of measurements across the WiseNET protocol stack.

A.1.1.2. Routing

At power-on, routes towards the sink (node with address 1), are predefined using the rule “the address of the next hop towards the sink is my binary address shifted by 2 to the right”. For example, 64, 65, 66 and 67 goes to 16.

A dynamic routing algorithm has also been implemented to permit the automatic construction of a convergecast tree, allowing the multi-hop transmission from every sensor node to the sink.

A flood is the repetition (only once) of a broadcast message by every node of a multi-hop network. The initial source of the flood and a sequence number managed by this source must be transmitted with the flood in order to uniquely identify a flood and avoid endless repetitions.

The convergecast routes are established using a single flood originated at the sink. Every node repeats the flood once, and remembers the address of the source of the first received copy of the flood as the next hop towards the sink. Because of the temporal constraint (first received copy), loop are not possible.

The flood mechanism is illustrated in Figure 103. The number on the nodes represents the temporal order with which the flood messages are transmitted. The black arrows represent the link on which the flood messages are received. The large orange arrows represent the links that are selected to form the convergecast tree. In this example, the flood is started by the sink (node with number 1). It is first repeated by node number 2 and then by node number 3. Because node number 2 has repeated the flood before node number 3, node number 5 selects the node number 2 has the next hop, whether or not link towards 2 is better that link towards 3.

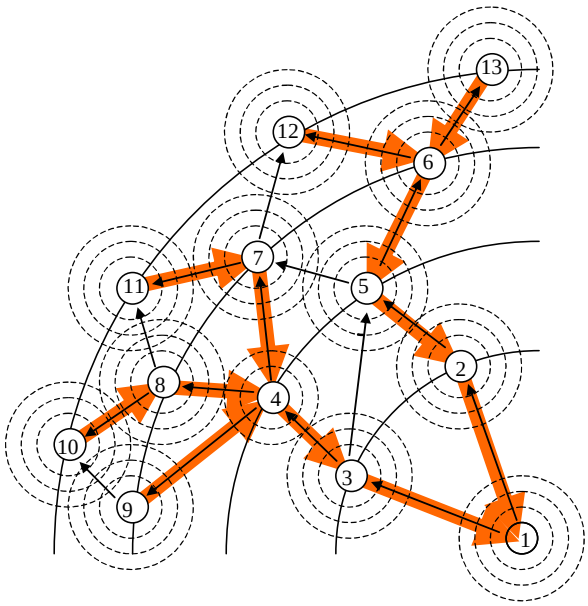

Figure 103: Using a flood to build a convergecast tree.

This routing algorithm is suitable for static networks. Nodes added to the network can be detected through a periodic route establishment (e.g. every hour). Link breaks can be detected at application layer through a malfunction of the convergecast data collection. A re-establishment of routes can then be initiated.

In order to avoid that links of low reliability are selected to be part of the convergecast tree, one can set a threshold below which received flooding routing messages are ignored[14].

A.1.1.3. Network Synchronization

In order to permit measurement time-stamping, a network-wide synchronized clock is provided by the WiseNodes to the sensing modules. The clock can be read using HCI commands.

Network-wide synchronization occurs:

1. with every flood. The source of a flood message inserts its clock value. A node receiving the flood adopts the transmitted clock value.
2. with convergecast traffic. Acknowledgements contain the value of the clock of the source of the acknowledgement. If the application consists uniquely in convergecast traffic, acknowledgements are always going away from the sink and therefore propagate a more accurate clock.

A.1.1.4. Host Flood

A host-to-host command transmission mechanism has been implemented within this project to let the data collection computer transmit a command to all sensing modules. The host command is transported using the flood mechanism. The host command is transmitted by every WiseNode once, and only once, to the host sensing module via the HCI interface (see Figure 104).

[14] Only flooding routing message are subject to this threshold. Flooding network management messages are never ignored.

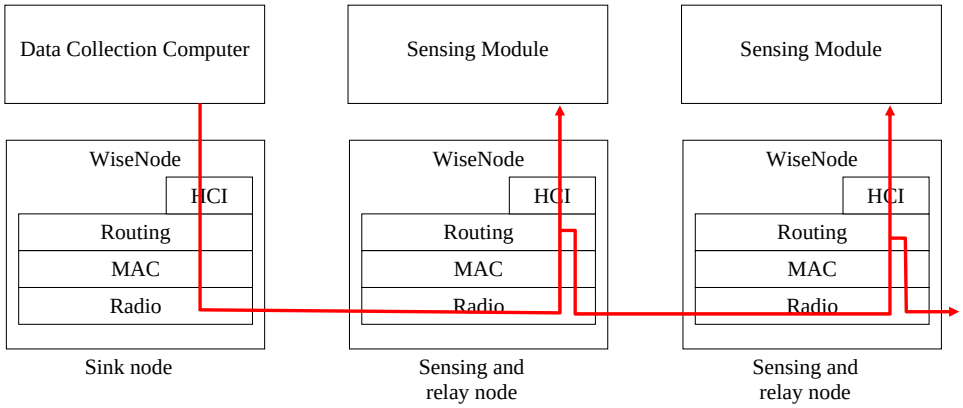

Figure 104: Transmission of a host command from the data collection computer to each sensing module.

A.1.1.5. Network Management

Network management commands can be issued by the data collection computer to configure certain parameters of the protocol stack (i.e. transmit power, sampling period, routing tree). Network management commands are transported using the flood mechanism and executed by an embedded application layer, as illustrated in Figure 105.

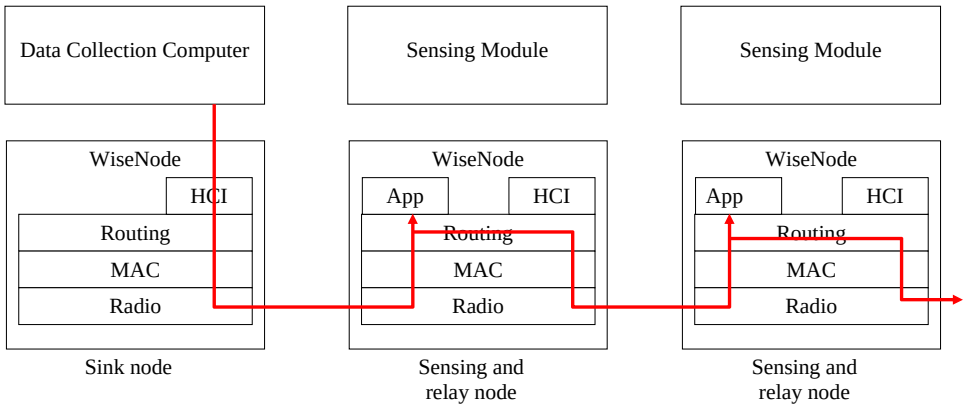

Figure 105: Transmission of a network management command.

To allow the monitoring of the network performance, it is possible to request from the embedded application the periodic transmission of statistics message, containing information such as the link receive power, the packet error rate and the energy consumed by the radio.

A.3. WiseNode Communication Modules

A.3.1. Introduction

The first generation of WiseNode modules (WiseNode_v1) has been developed at CSEM within an internal research project (WiseNET project). A second version (WiseNode_V2), with improved antenna efficiency has been produced within this project and delivered to Empa. Because of the very tight memory constraints of the used processor (24 kB program memory and 512 bytes RAM), it was not possible to reach the desired functionality. A new processor with more memory has been used in WiseNode_v4. This version has been delivered to Empa as well. The WiseNode_V2 and WiseNode_V4 hardware platforms are described in the following two sections.

A.3.2. WiseNode_V2

The WiseNode_V2 module (Figure 106) is based on a XE88LC06A microcontroller and a XE1203 radio, both from Semtech (formerly Xemics, a spin-off company of CSEM). The XE88LC06A microcontroller offers 24 kB program memory and 512 bytes RAM.

Figure 106: WiseNode Platform V2

Communication with the sensing module occurs via HCI over the UART interface. The UART interface is available either through the UART flat cable connector (red) or through the white board-to-board connector (white).

More details about the WiseNode_V2 platform can be found in [55].

An adaptation board between the board-to-board connector and a flat-cable connector has been produced by CSEM to permit access of Empa's sensing module to all

the potentially required signals. Figure 107 shows the description of the flat-cable connector available on this adaptation board. UART communication occurs through pins 6 (UART output from WiseNode) and 7 (UART input to WiseNode). Power can be given to the WiseNode via pin 15. The WiseNode can be reset by pulling pin 13 low.

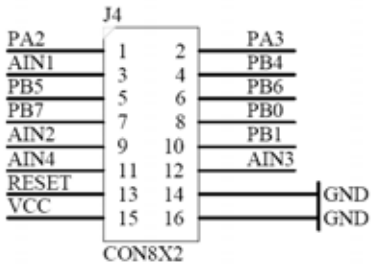

Figure 107: Description of flat cable extension connector of the WiseNode2_Adapter_v3 board.

Programming of the WiseNode boards occurs via the black connector, using a programming board (XE8000MP) and the XELoader programming software provided by Semtech.

A.3.3. WiseNode_V4

The WiseNode_V4 module (see Figure 108) is based on a MSP430F1611 microcontroller and a CC1100 radio at 868 MHz. The MSP430F1611 offers 48 kB program memory and 10 kB RAM, which represent twice as much program memory and 20 times more RAM than with the WiseNode_V2. The main reason for the change of the CPU between WiseNode_V2 and WiseNode_V4 was the memory limitations with the XE88LC06A microcontroller which proved to be too constrained for this application. In addition, the change of the radio chip between versions 2 and 4 has permitted to increase the transmission range from about 500 meters to 2 kilometres (in line of sight).

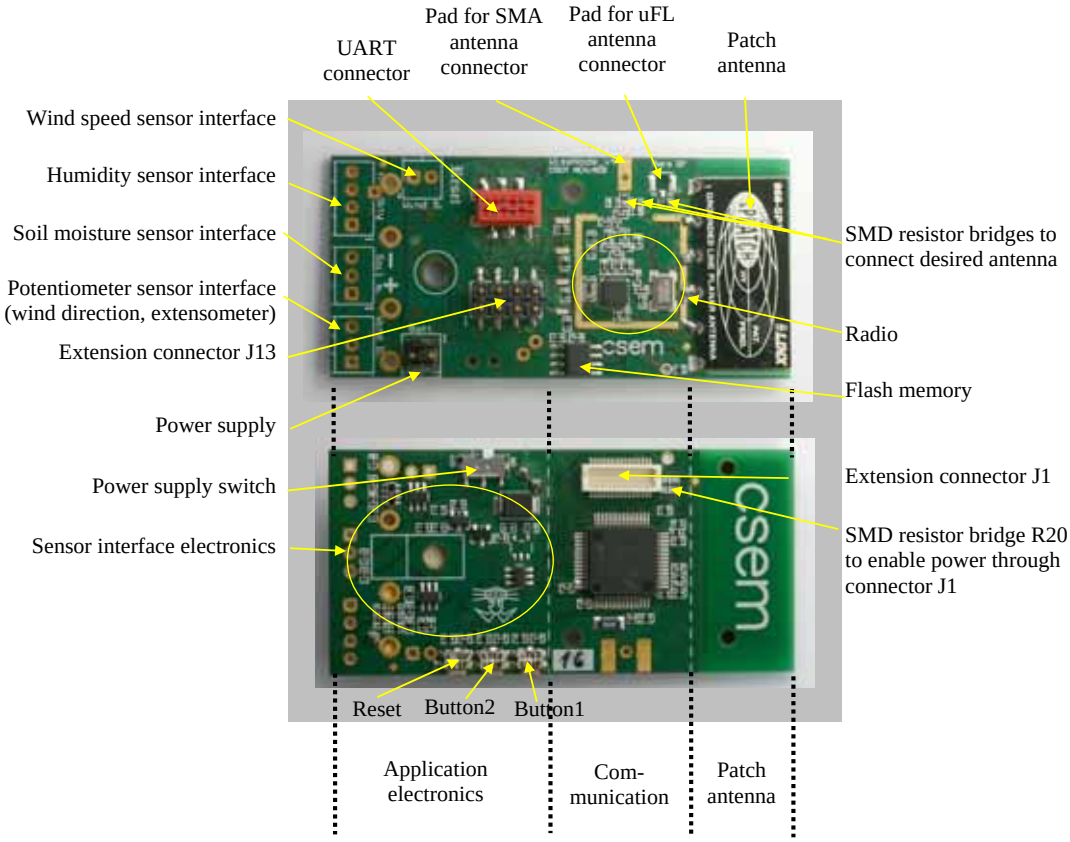

Figure 108: WiseNode_V4 module.

The integration with the sensing board used by Empa occurs through the flat cable extension connector, illustrated in Figure 109.

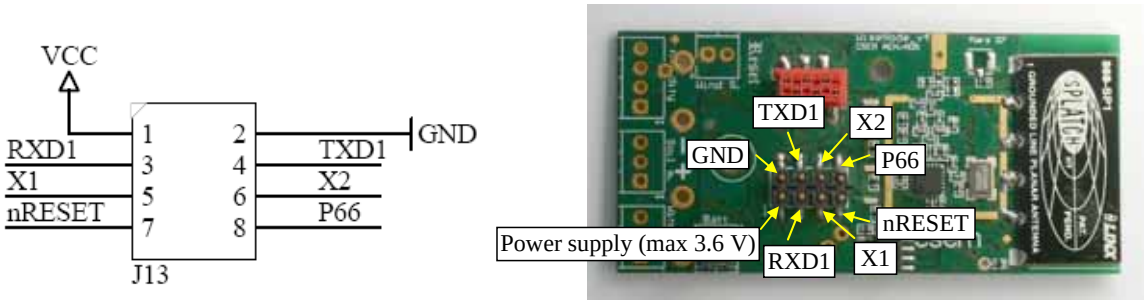

Figure 109: Flat cable extension connector.

More details about the WiseNode_V4 hardware platform can be found in [56].

A.3.4. Hardware Parameters

This following table presents measurement results of characterization parameters important for the theoretical power consumption evaluation. Power consumption values are computed from the measured current with a multiplication by the supply voltage of 3 volts.

Description	Symbol	WiseNode_V2	WiseNode_V4
Power consumption in transmit mode (10 dBm)	P_T	129 mW (43 mA)	103.8 mW (34.6 mA)
Power consumption in receive mode (0 dBm)	P_R	42 mW (14 mA, 25 kbps)	54 mW (18 mA, 10 kbps)
Energy consumption of channel sampling	E_S	50 uJ (see Figure 110)	55.3 uJ (see Figure 111)
Power consumption in idle mode	P_Z	7.2 uW (2.4 uA)	81 uW (27 uA)

Table 12: Power consumption of the WiseNodes modules.

The current consumption during a channel sampling using WiseNode_V2 and WiseNode_V4 is shown in Figure 110 and Figure 111 respectively.

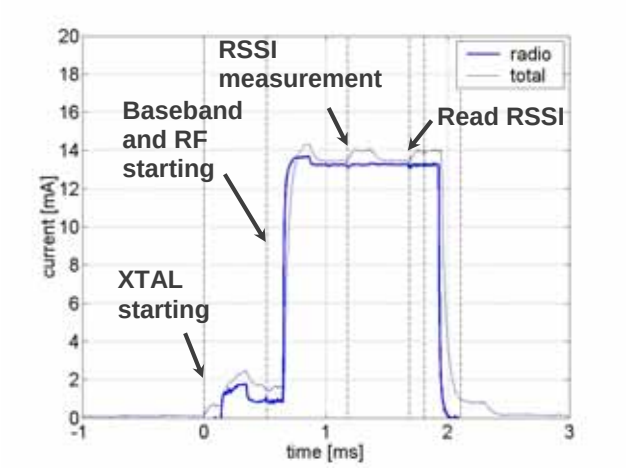

Figure 110: Consumed energy for wake-up and RSSI integration with XE1203 (total 50 uJ).

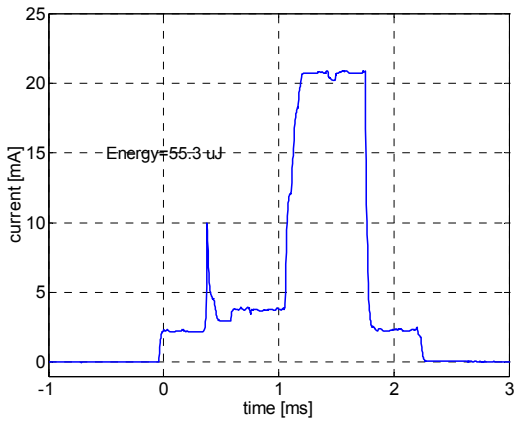

Figure 111: Consumed energy for wake-up and RSSI integration with CC1100 (total 55.3 uJ).

A.4. Theoretical Power Consumption

Let's consider, as illustrated in Figure 112, a multi-hop network with N sensor nodes and one sink ($N = 6$ in this example). Every sensor node generates λ packets per second. The node with the highest power consumption will be the last node before the sink (the sink, being attached to a computer can be expected to be powered from the mains). This node receives $(N-1)\lambda$ packets per second and transmits $N\lambda$ packets per second.

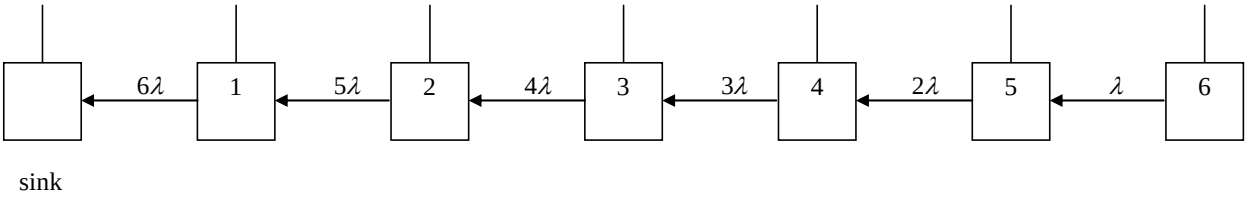

Figure 112: Network topology for power consumption estimations.

For simplicity, we will consider that the last relay node forwards $N\lambda$ packets per second. This generates an upper bound on the power consumption, very close to the correct value if the network is large. The theoretical power consumption of the last relay node is given by

$$P = P_Z + \frac{E_S}{T_W} + E_{Fwd} N\lambda \,, \tag{25}$$

where the first term P_Z is the base power consumption[15].

The second term E_S / T_W is the average power used for the sampling activity. T_W is the sampling period. A large T_W reduces the power consumption of the sampling activity but increases the hop delay and decreases the maximum throughout (as a node can at most forward as many packets as its memory can hold per T_W).

The third term $E_{Fwd} N\lambda$ is the average power needed to forward packets. E_{Fwd} can be computed completely theoretically, but it is simpler and more precise to calibrate the minimum forwarding energy consumption with measurement and add the variable part. We have[16]

$$E_{Fwd} = E_{FwdMin} + (P_R / 2 + P_T)(T_P - T_{P\min}) \,. \tag{26}$$

The length of the wake-up preamble depends on the traffic. We have

$$T_P = \min\left(T_{P\min} + \frac{4\theta}{N\lambda}, T_W \right), \tag{27}$$

where θ is the quartz tolerance (typically 100 ppm for low cost 32kHz watch crystals). $T_{P\min}$ is the minimum length of the wake-up preamble required to cover the jitter.

The minimum forwarding energy has been measured to be $E_{FwdMin} = 2106\ \mu\text{J}$ in the current implementation (see Figure 113).

[15] If P_Z is not negligible compared to P_R and P_T, then P_R and P_T should be replaced by $\hat{P}_R = P_R - P_Z$ and $\hat{P}_T = P_T - P_Z$ in the equation (26), in order to be accurate.

[16] $T_{P\min}$ is subtracted from T_P because it is already included in E_{FwdMin}.

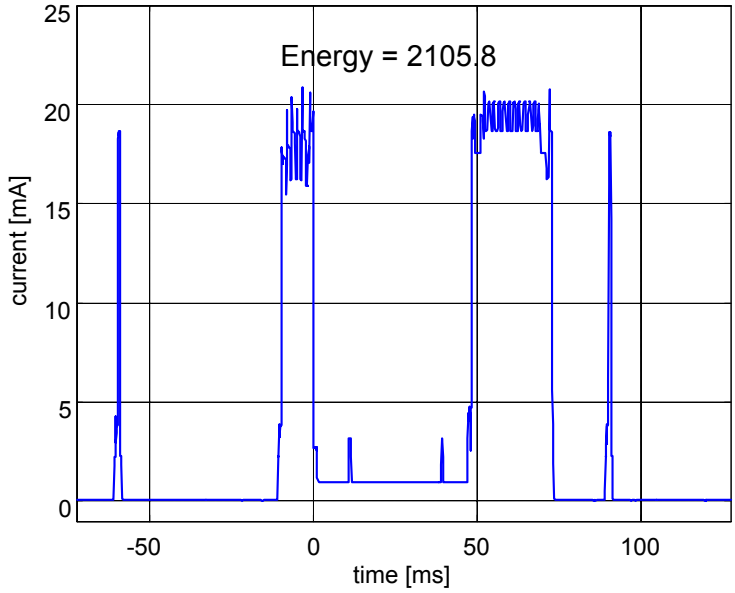

Figure 113: Measurement of the forwarding energy when clock synchronization is precise (minimum forwarding energy).

For the lifetime computation, we will consider the hardware parameters of the WiseNode_V4 module (the version used in the deployments) that are listed in the table of section A.3.4. Additional parameters are listed below:

$\theta = 100 \cdot 10^{-6}$

$T_W = 250 \text{ ms}$

$T_{P\min} = 6.4 \text{ ms}$ (200 bytes at 250 kbps, subject to be lower after further optimization).

One can compute using (25) the average power consumption of the last hop relay node as a function of traffic (see Figure 114).

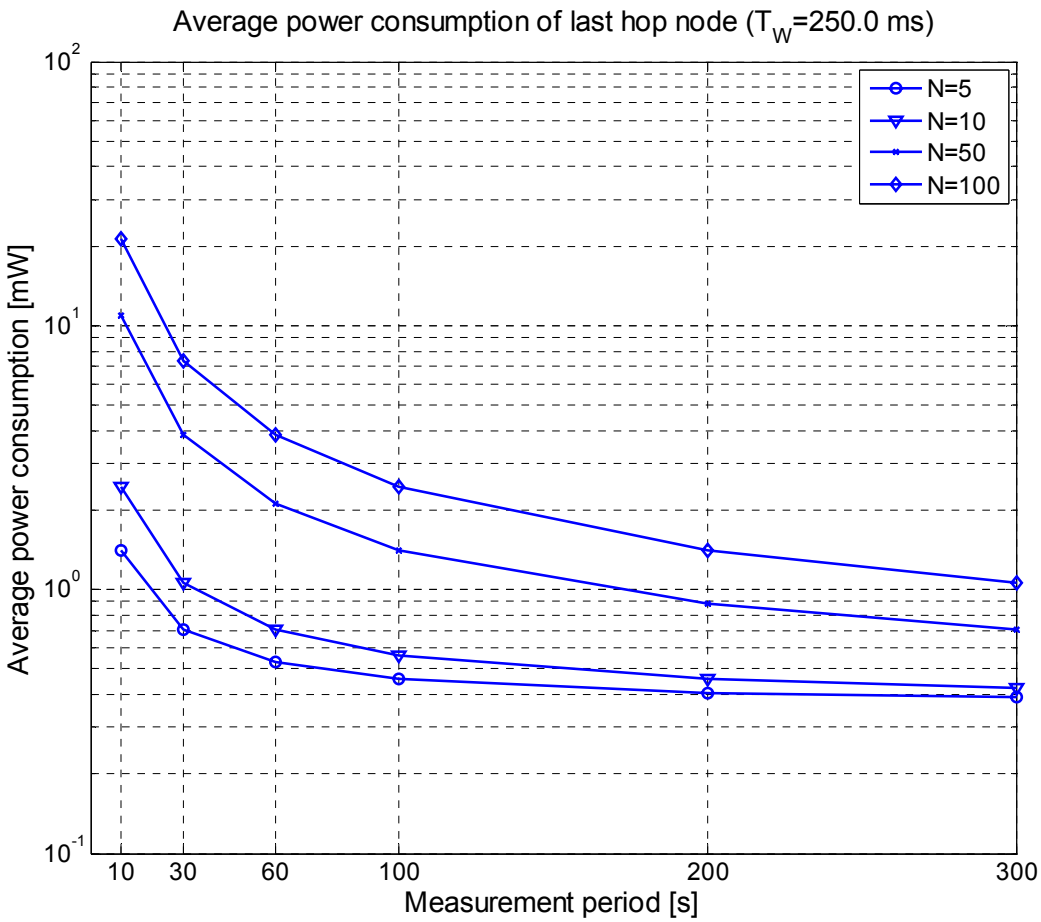

Figure 114: Average power consumption of the last hop WiseNode as a function of the generated traffic (measurement period) and for different network sizes (N).

To compute a lifetime, one needs a battery model. We will use the battery model described in [57] which considers a continuous discharge current across the lifetime of the battery. This model is pessimistic as the discharge current can be expected to become smaller as the energy contained in the battery decreases. We have

$$T = \frac{E}{P + P_{Leack}} = \frac{E}{24 \cdot 365 \cdot P + \xi E} \text{ [years]}, \tag{28}$$

where E is the capacity of the battery in Wh and ξ the percentage of the battery energy lost due to leakage in the first year.

Considering a 17 Ah 3.6 V Lithium battery from SAFT (LS3360 [57]), and assuming a self-discharge of 1 percent every year (which is pessimistic for a Lithium battery), we obtain the expected lifetime as illustrated in Figure 115.

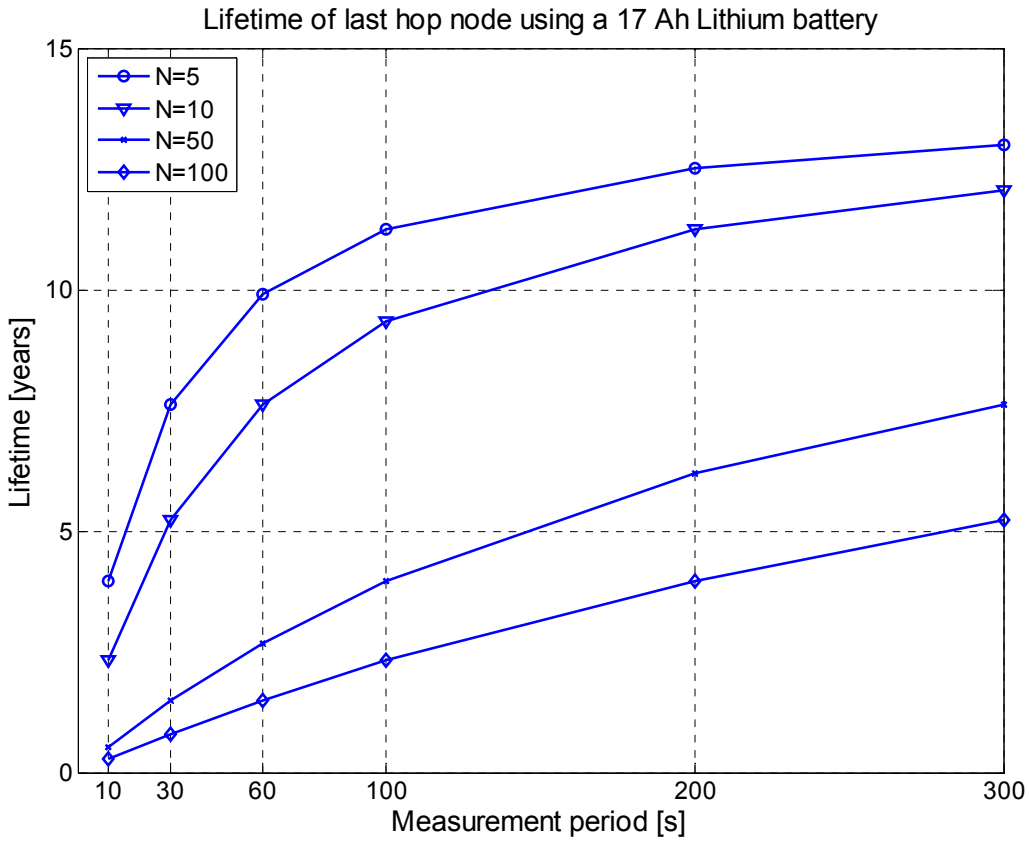

Figure 115: Expected lifetime for the last hop node using a single 17 Ah 3.6 V LS3360 SAFT Lithium battery as a function of the generated traffic (measurement period) and for different network sizes (N).

It can be observed that a lifetime in the order of 5 years can be obtained with a large network (N=100) if the measurement generation period is in the order of one every 5 minutes. With a small network (N=5…10) the measurement period can be reduced down to about one every 30 seconds while keeping the same lifetime.

The use of Alkaline batteries for this application permits a cost reduction with the drawbacks of a lower tolerance to low temperatures. The E95 Energizer Alkaline battery (see [58]) offers 20.5 Ah capacity with a self-discharge of 3 % (20 % in 7 years). It is advertised to operate down to -18 degrees. As it can be observed in Figure 116, a similar lifetime can be obtained when using two E95 Alkaline batteries as when using one LS3360 Lithium battery.

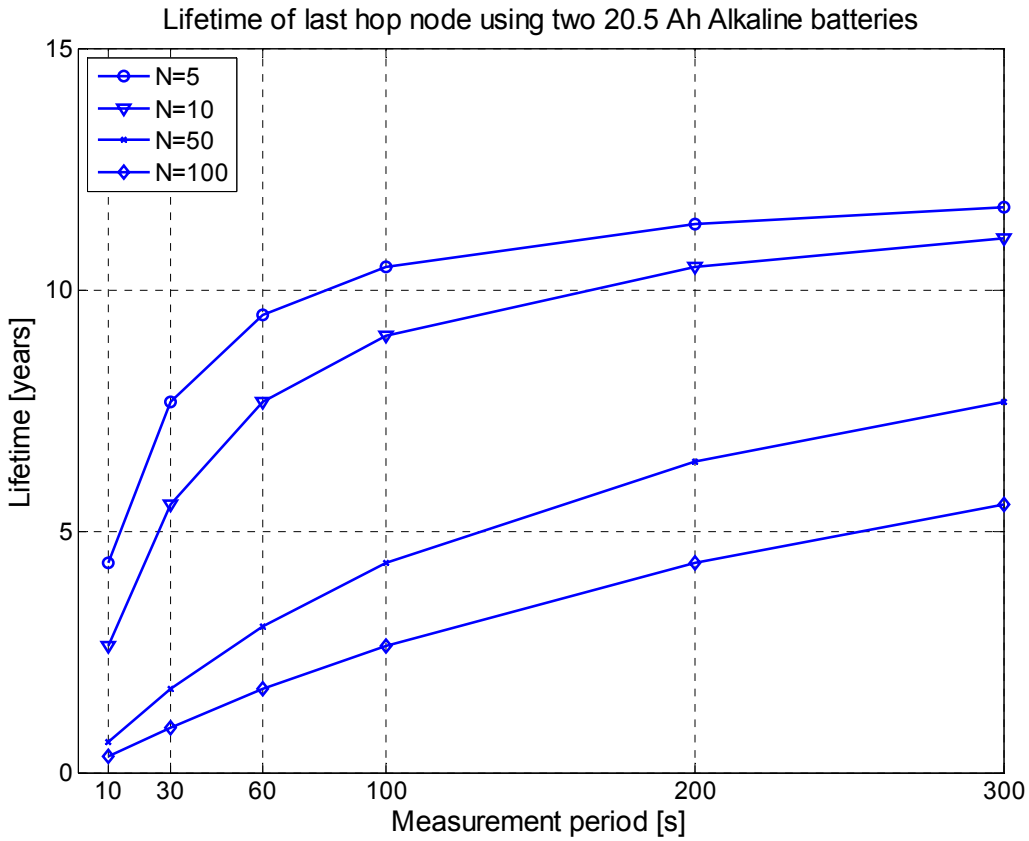

Figure 116: Expected lifetime for the last hop node using two 20.5 Ah 1.5 V E95 Energizer alkaline batteries as a function of the generated traffic (measurement period) and for different network sizes (N).

The power consumption and lifetime estimation presented in this section did only take into account the power consumption of the communication module. The power consumption of the sensing module should be included to obtain the overall power consumption.

A.5. Communication Infrastructure Test

A.5.1. Low Temperature Tests

A network with 5 WiseNodes_V5, powered with a LS3360 SAFT battery [57], are in outdoor operation since December 2007. Multi-hop convergecast data collection is made, with a transmission period of 30 seconds by every node. Low temperature tests have also been conducted using freezers. The tolerance to low temperature is excellent. No voltage drop has been observed on the Lithium batteries down to -20 degrees (these batteries are conceived for operation down to -60 degrees).

A.5.2. Transmission Range

Maximum reliable (packet error rate of about 1 %) range of 2 km achieved at 10 kbps (100 kHz channel bandwidth), a RSSI level at the receive side of -100 dBm, a transmission power of 10 dBm and using LINX antennas on both sides.

B. Screenshots of the Control Centre Operator Console

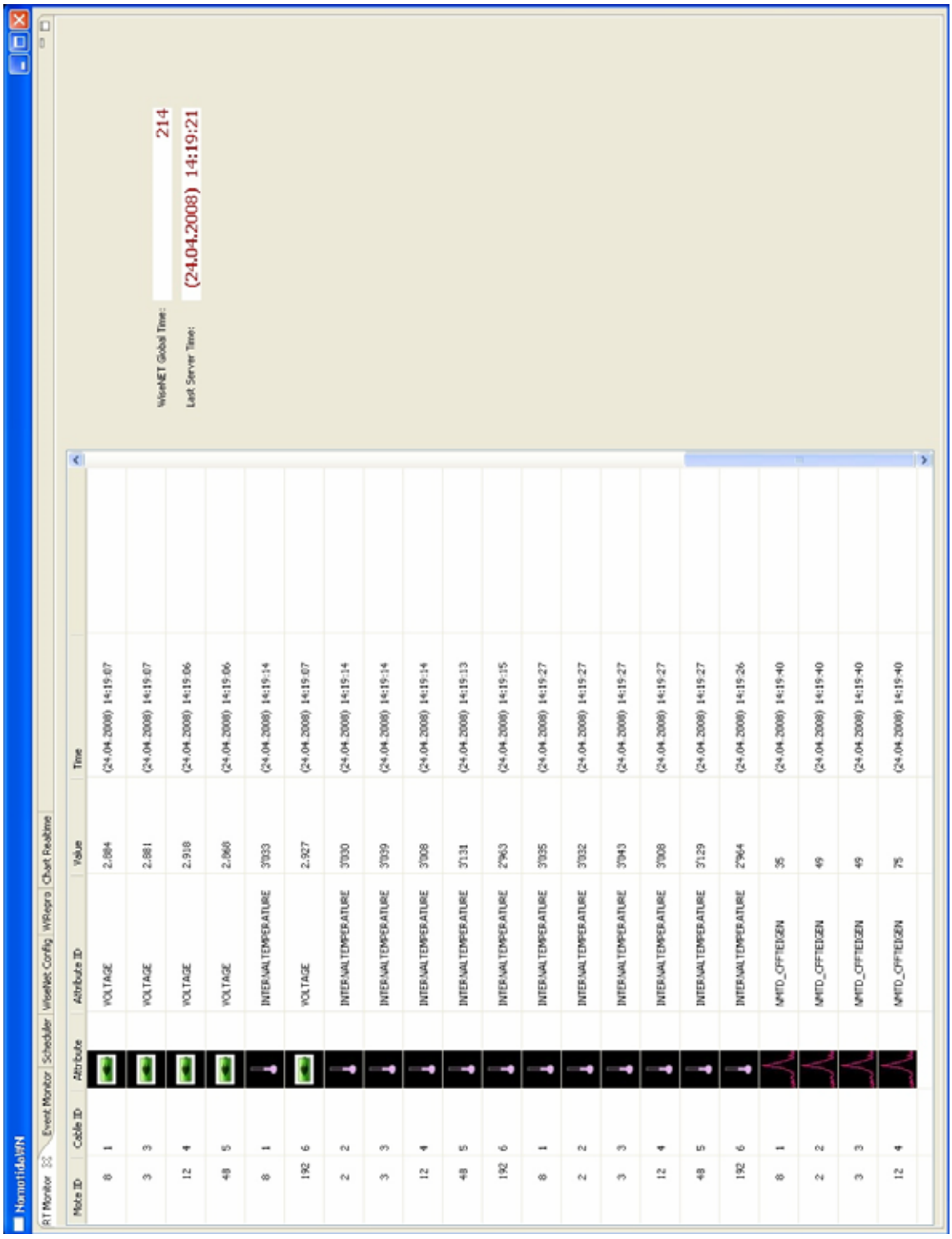

Figure 117: Screenshot of the incoming data display tool of the operator console at the remote control centre.

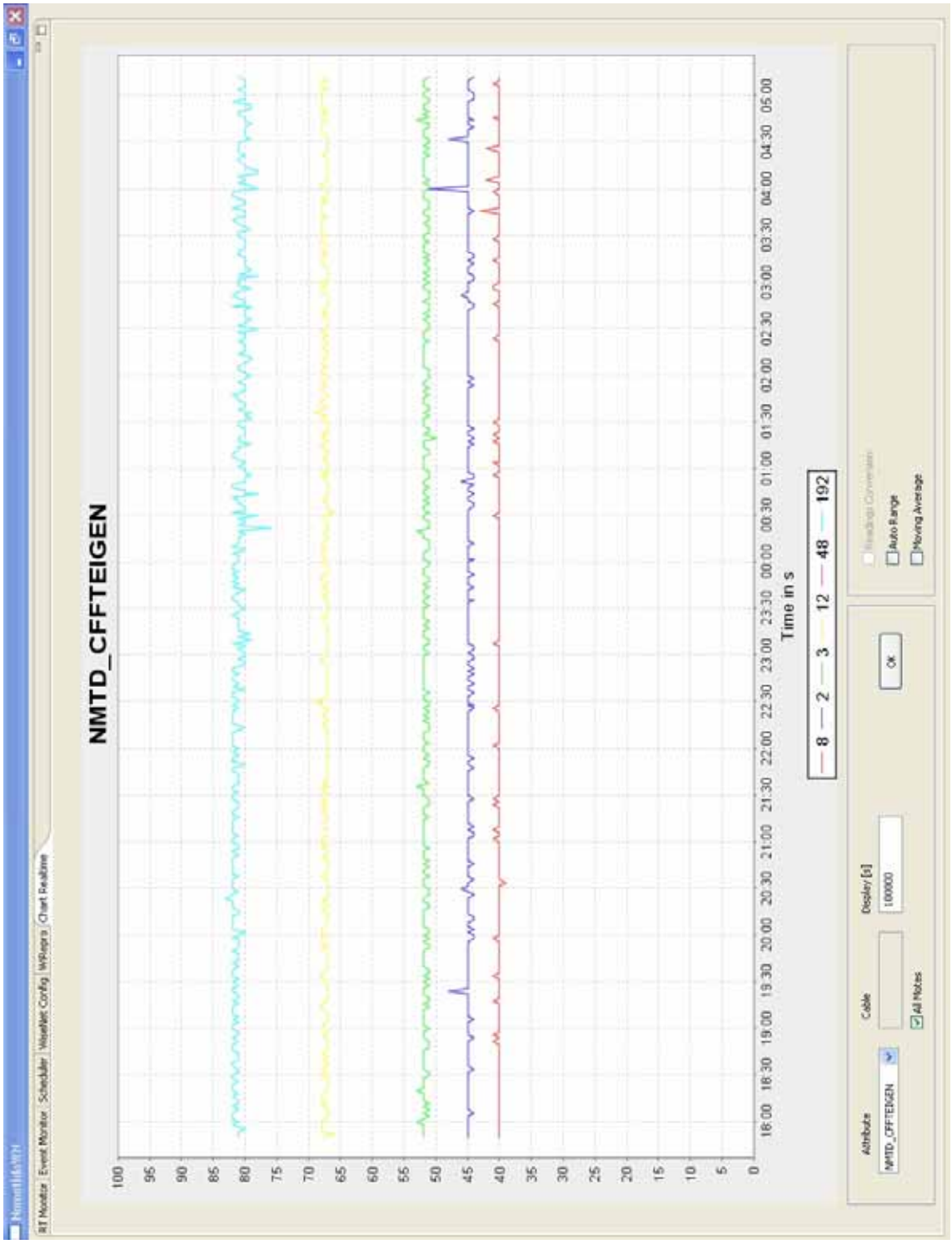

Figure 118: Screenshot of the incoming data plotting tool of the operator console at the remote control centre.

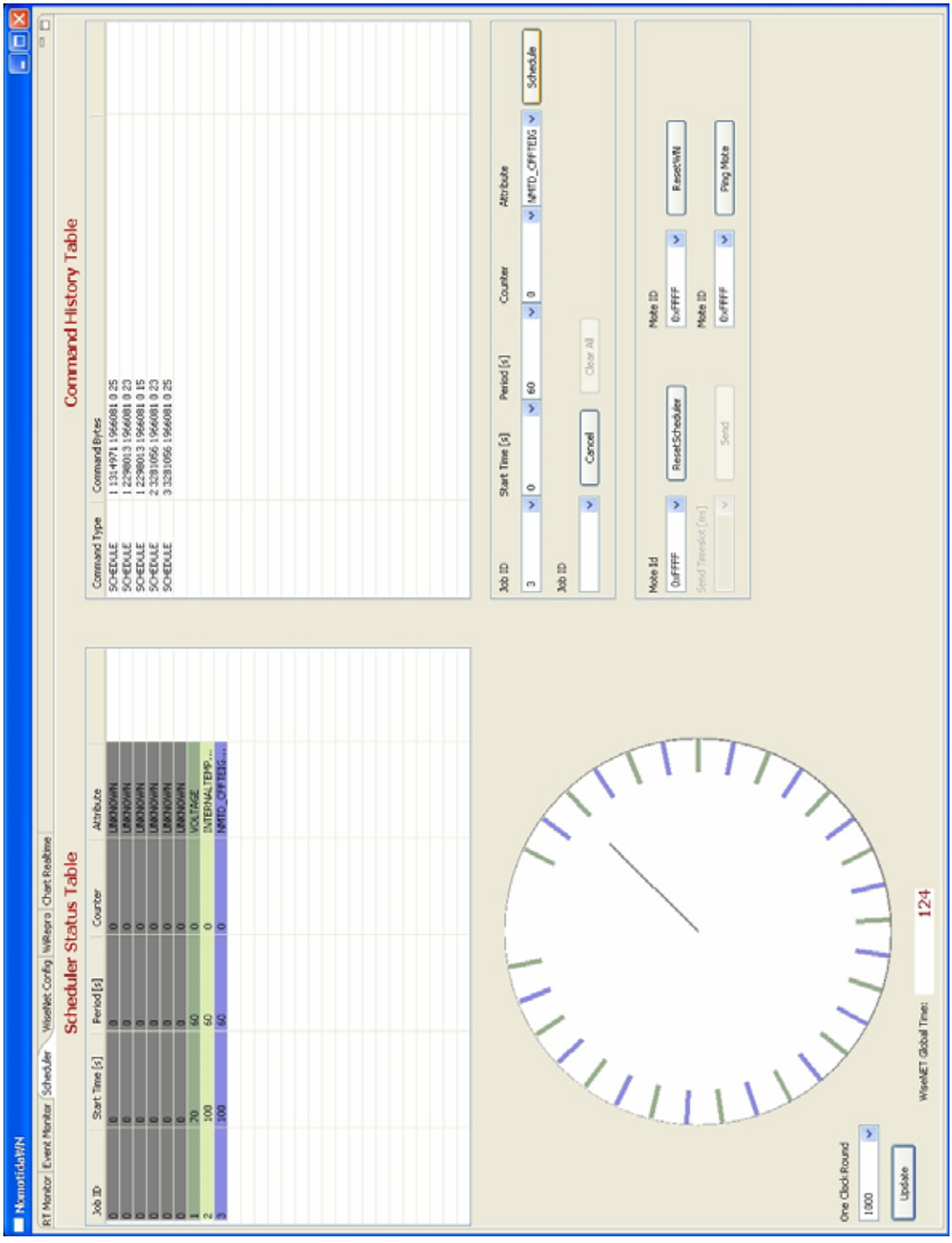

Figure 119: Screenshot of the measurement management tool of the operator console at the remote control centre.

Figure 120: Screenshot of the incoming event visualization tool of the operator console at the remote control centre.

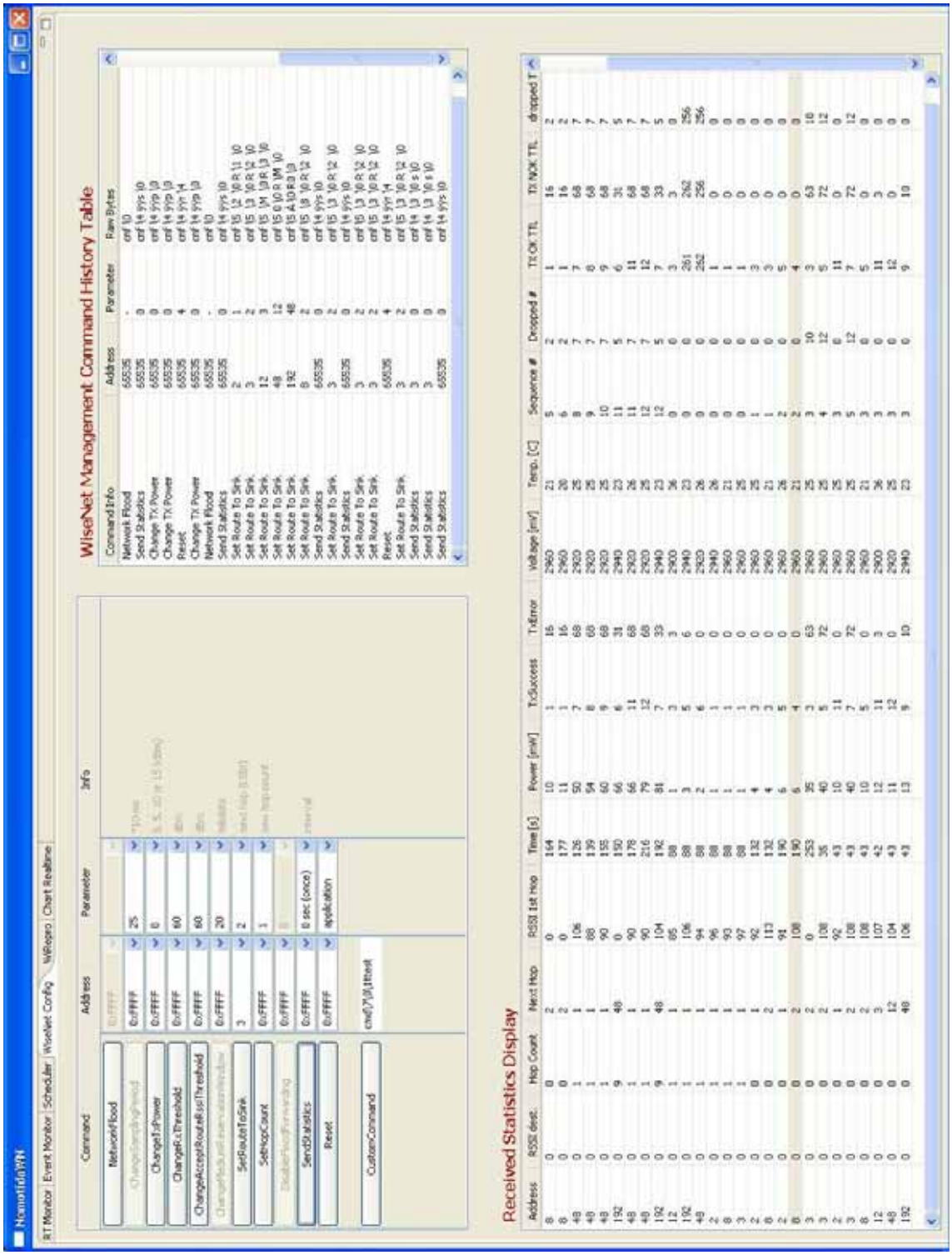

Figure 121: Screenshot of the network management tool of the operator console at the remote control centre.

Figure 122: Screenshot of the remote wireless reprogramming tool of the operator console at the remote control centre.

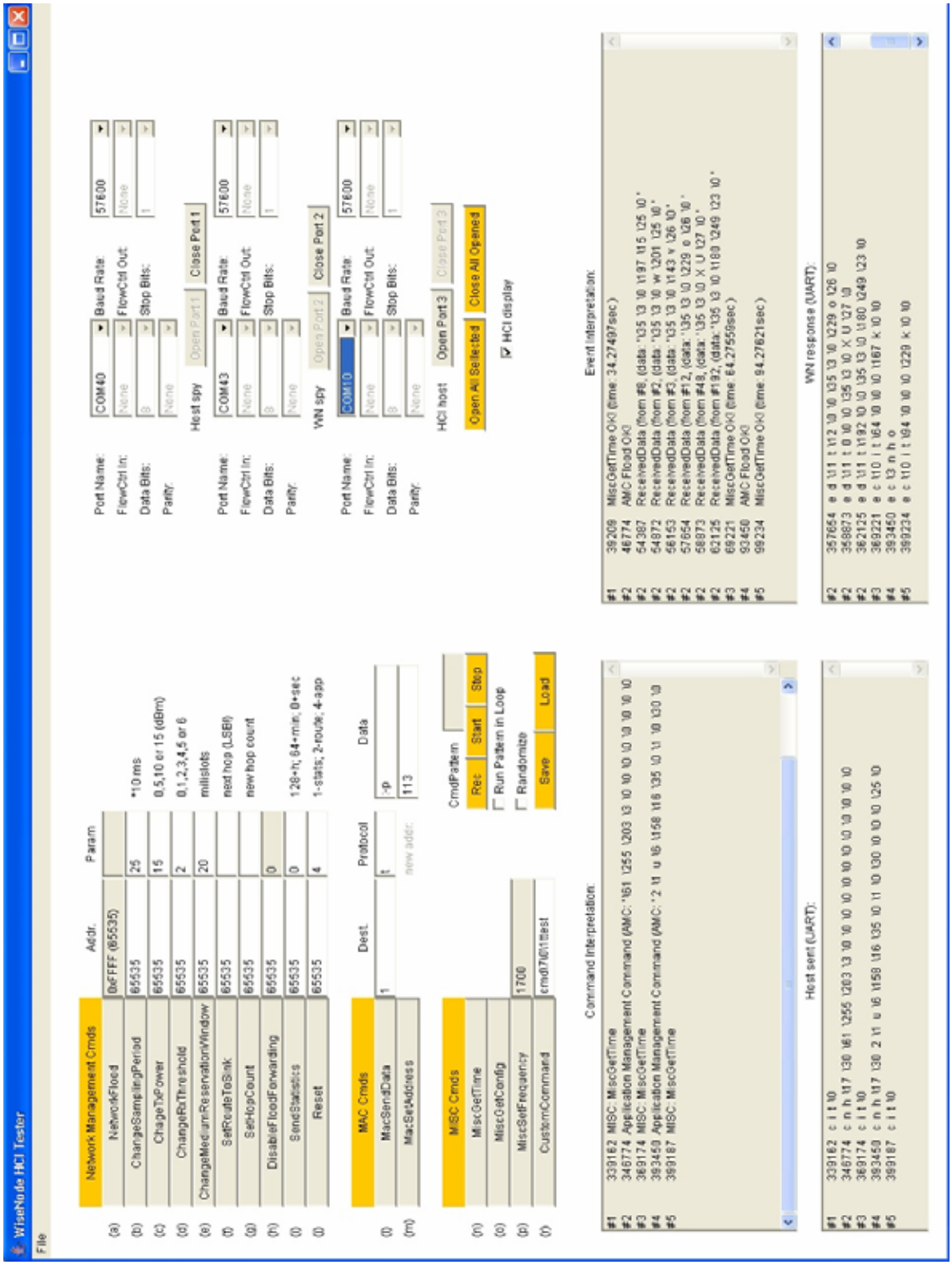

Figure 123: Screenshot of the HCI protocol testing and monitoring tool for protocol interpretation, communication logging and automatic protocol testing.

C. Sensor Board Design

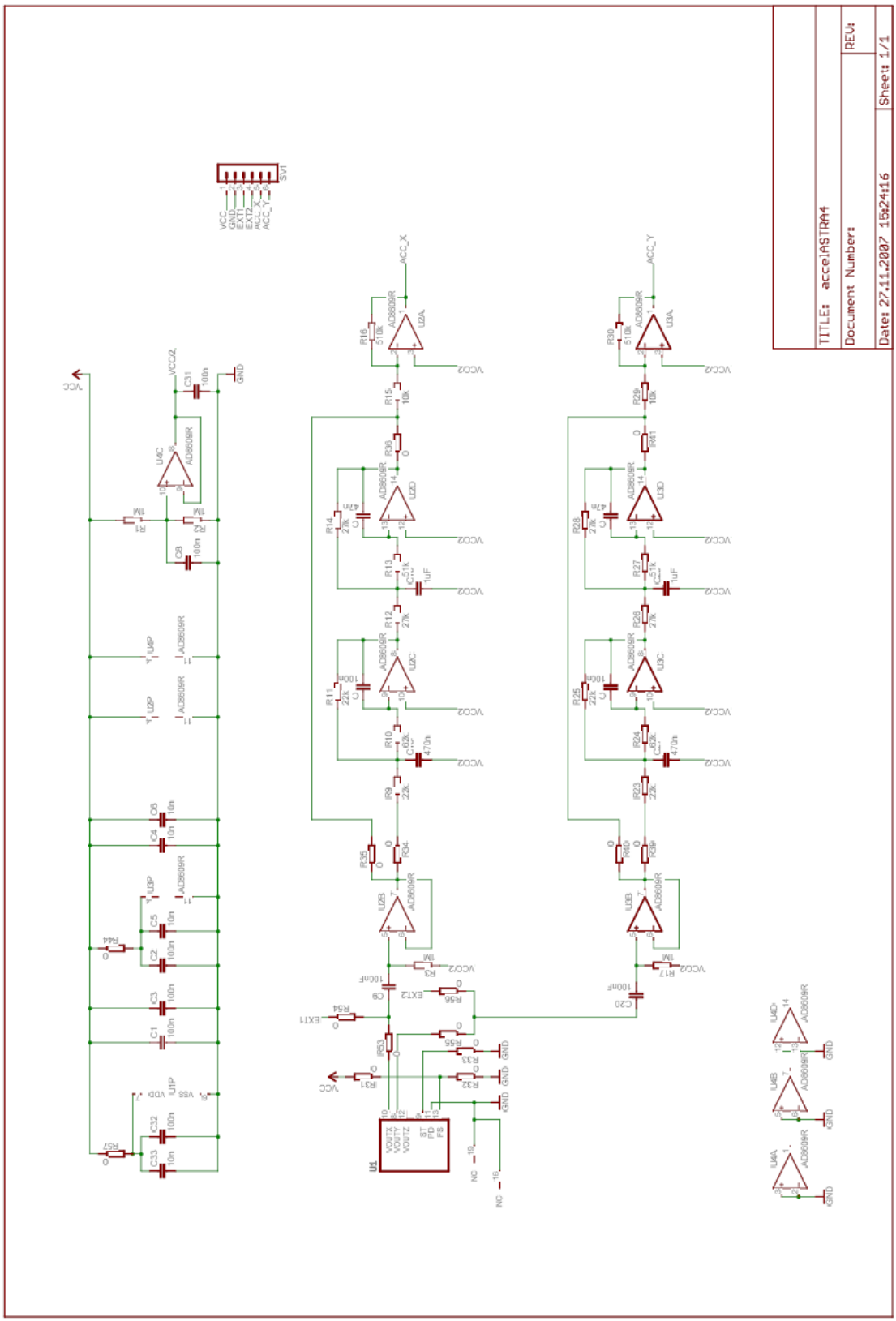

Figure 124: Schematics of the MEMS accelerometer sensor board.

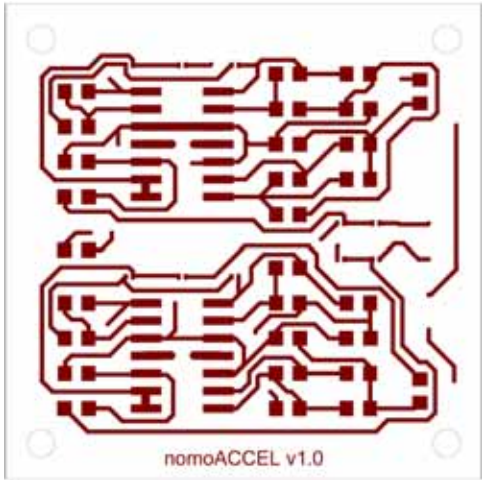

Figure 125: Zoomed view (200%) of the sensor board PCB top signal layer layout.

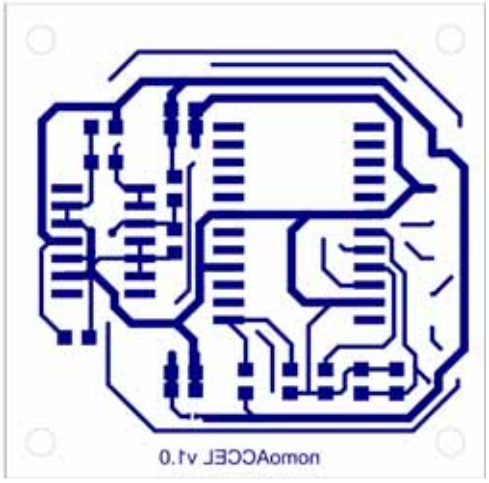

Figure 126: Zoomed view (200%) of the sensor board PCB bottom signal layer layout.

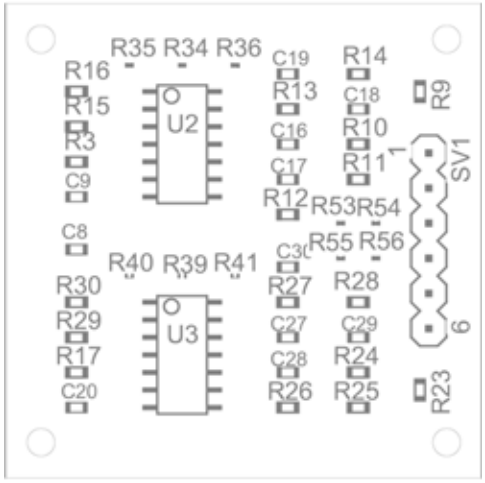

Figure 127: Zoomed view (200%) of the sensor board PCB top component placement.

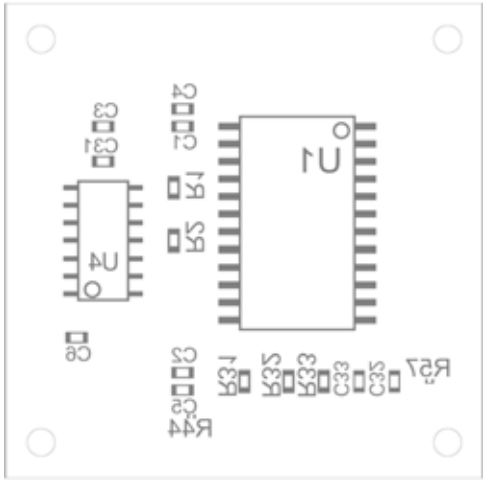

Figure 128: Zoomed view (200%) of the sensor board PCB bottom component placement.

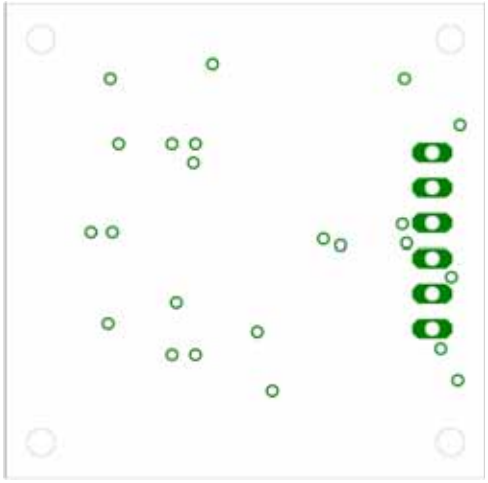

Figure 129: Zoomed view (200%) of the sensor board PCB drill layout.

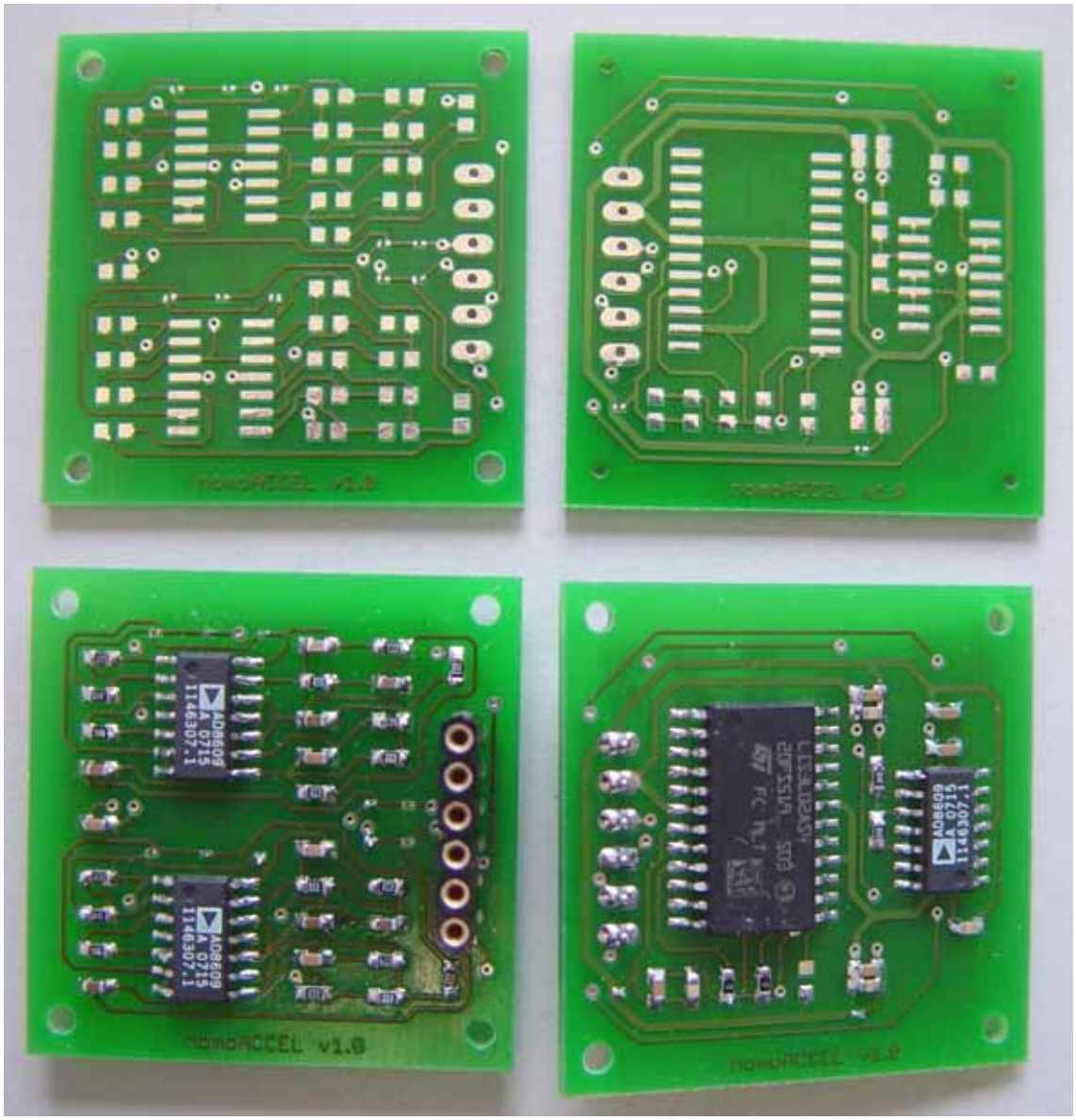

Figure 130: Photograph of the low-power sensor board.

D. Analysis Results from the Indoor Test at the Laboratory Bridge

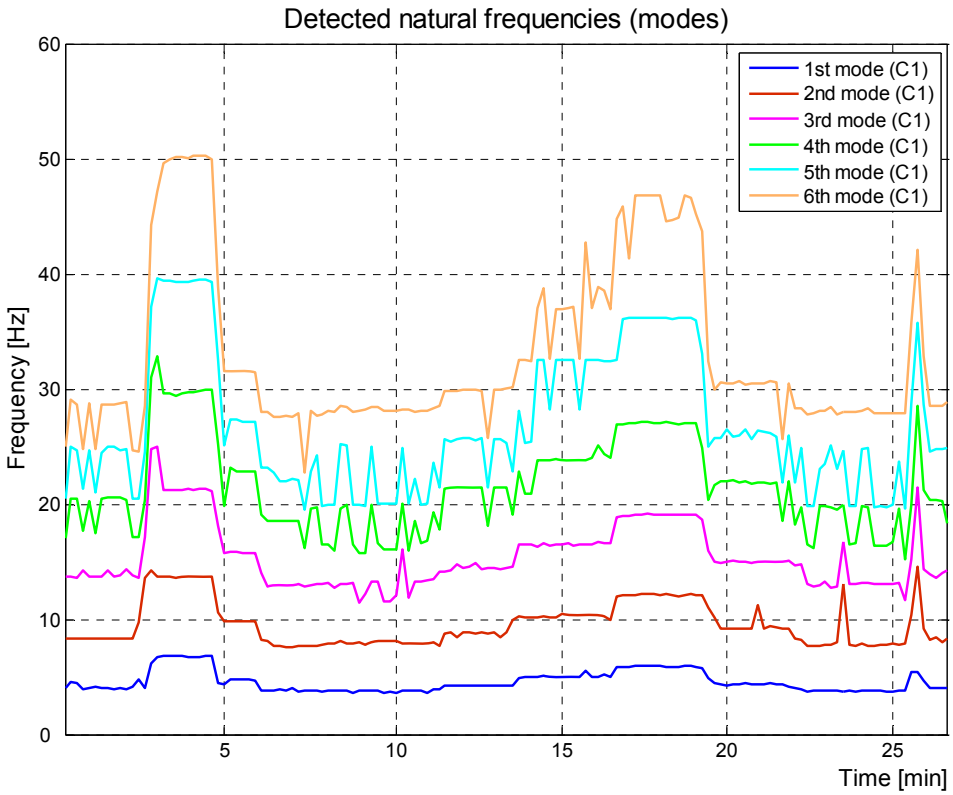

Figure 131: Detected natural frequencies of cable #1 of the laboratory bridge.

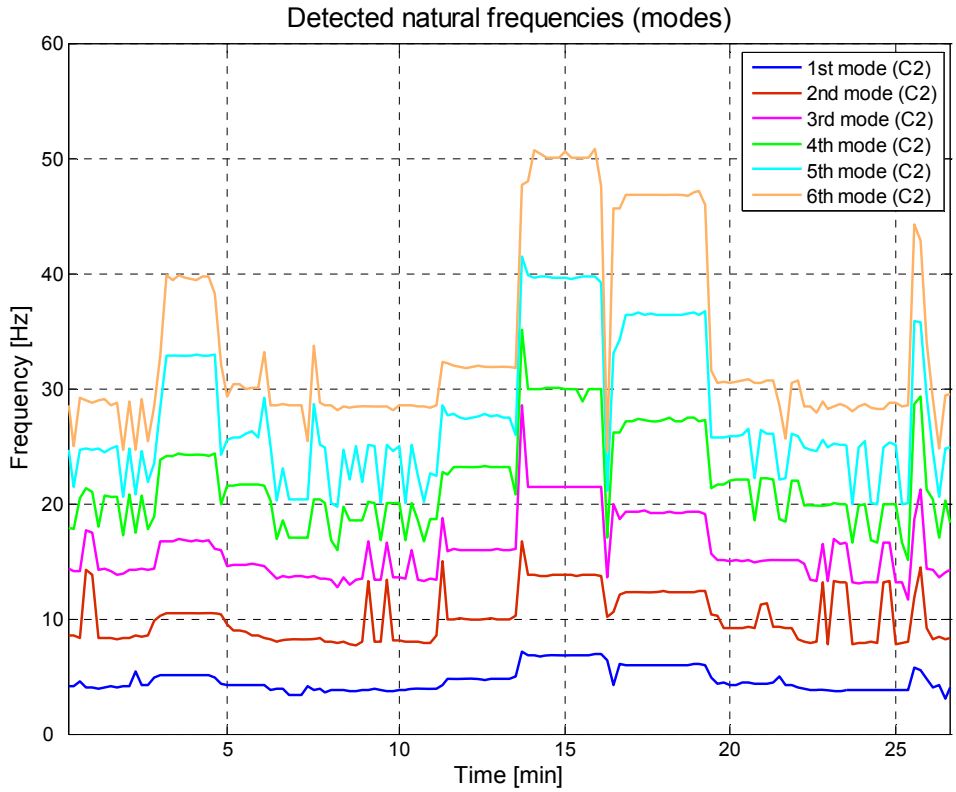

Figure 132: Detected natural frequencies of cable #2 of the laboratory bridge.

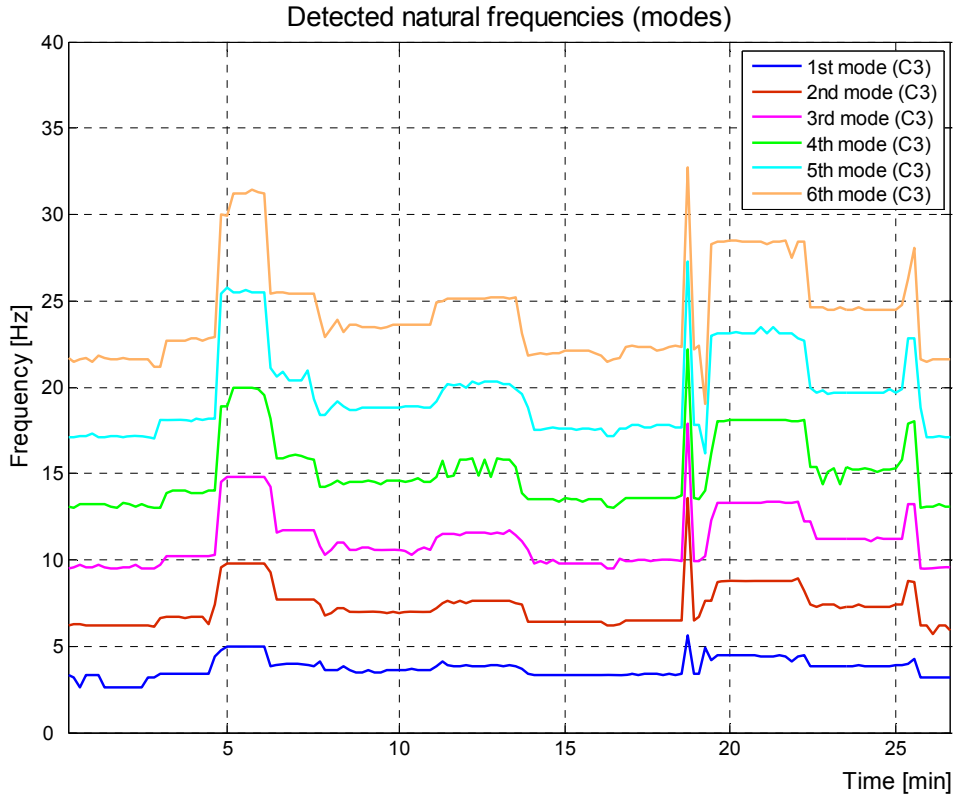

Figure 133: Detected natural frequencies of cable #3 of the laboratory bridge.

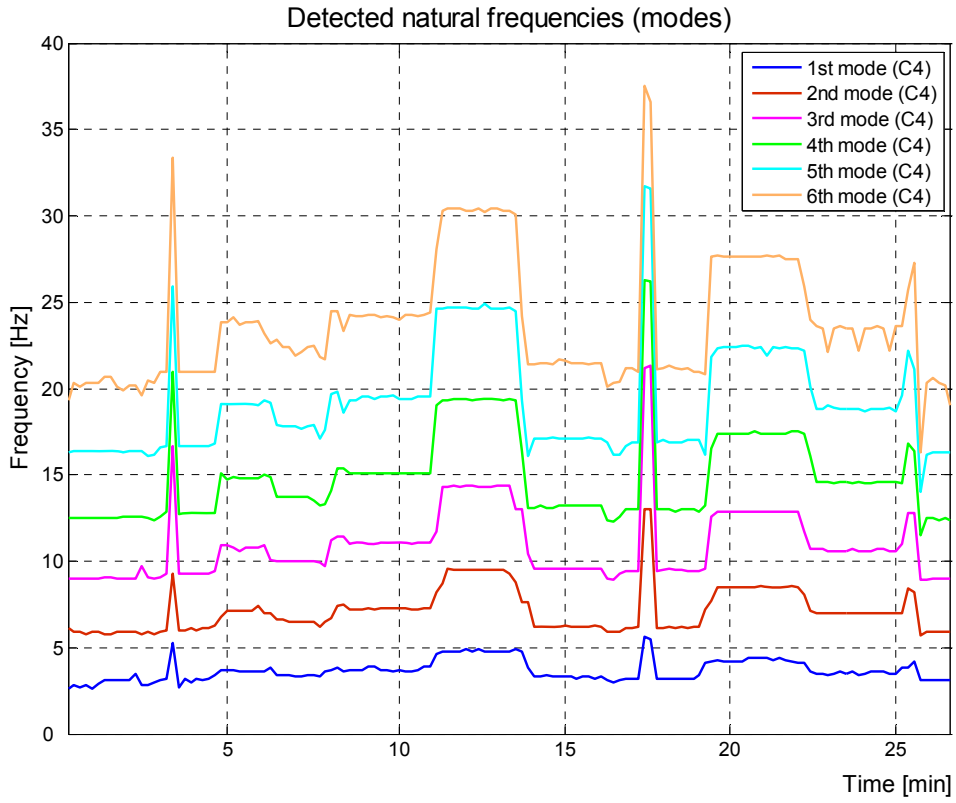

Figure 134: Detected natural frequencies of cable #4 of the laboratory bridge.

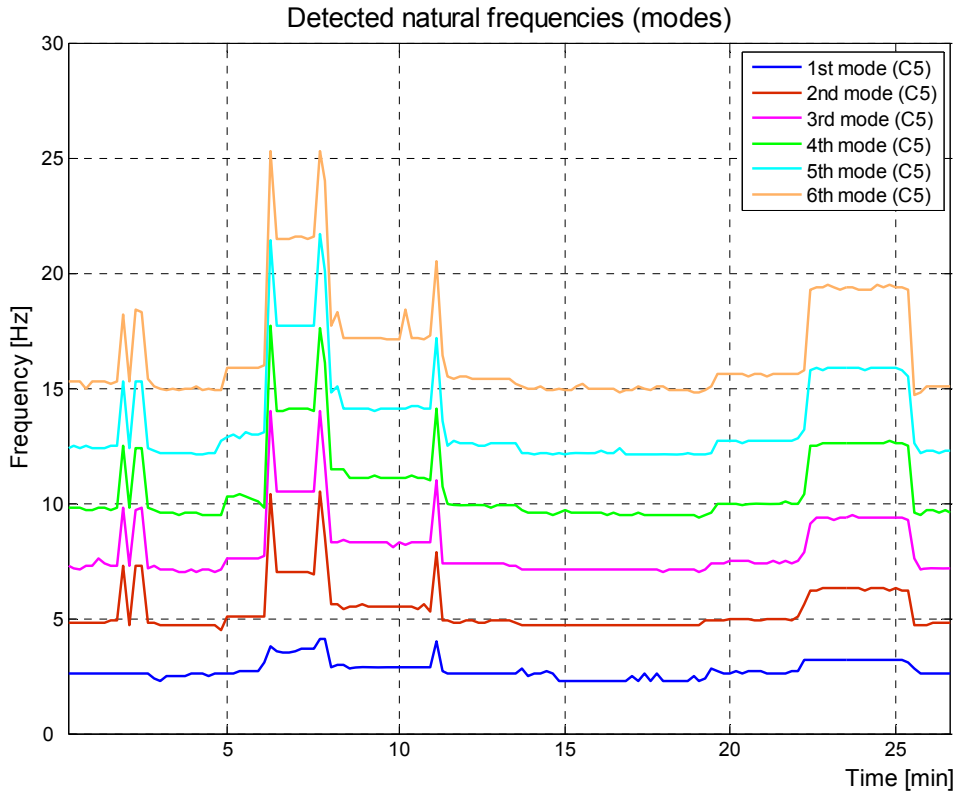

Figure 135: Detected natural frequencies of cable #5 of the laboratory bridge.

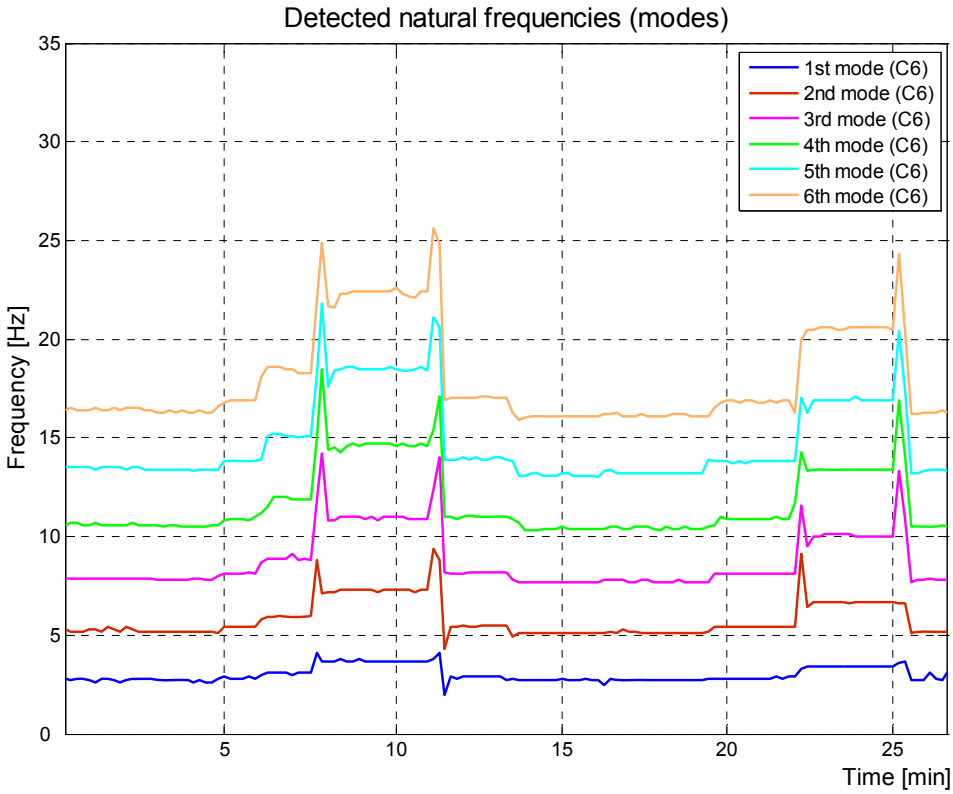

Figure 136: Detected natural frequencies of cable #6 of the laboratory bridge.

E. Performance Analysis of the Laboratory Network

May 9th 2008

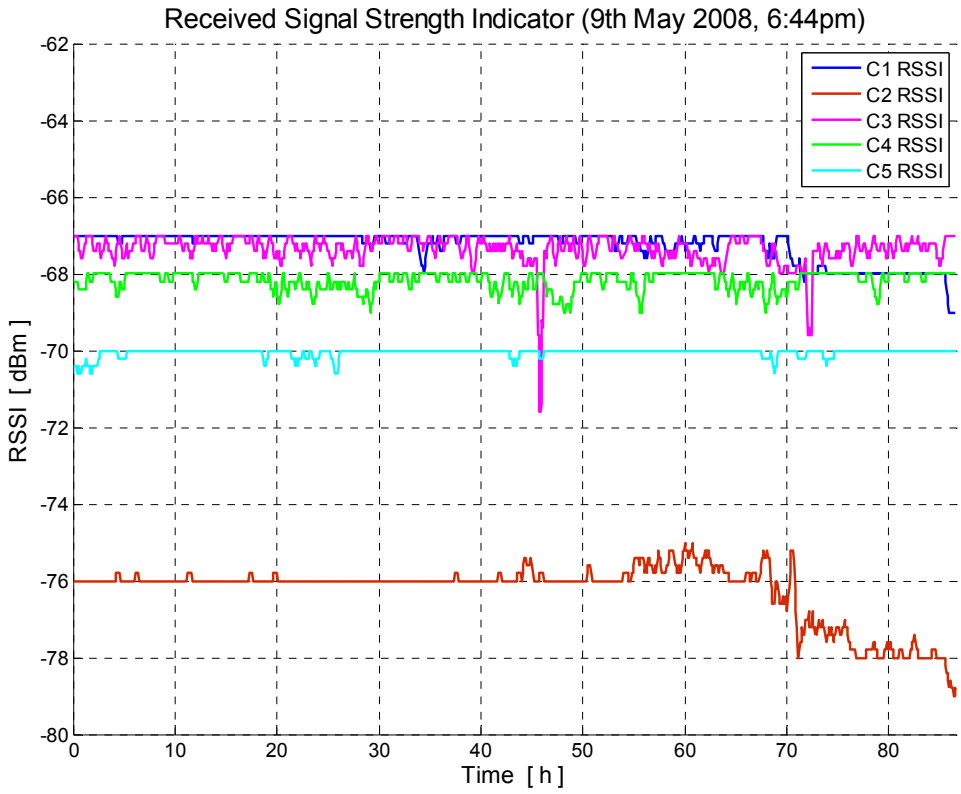

Figure 137: Signal strength towards the parent node in the default topology.

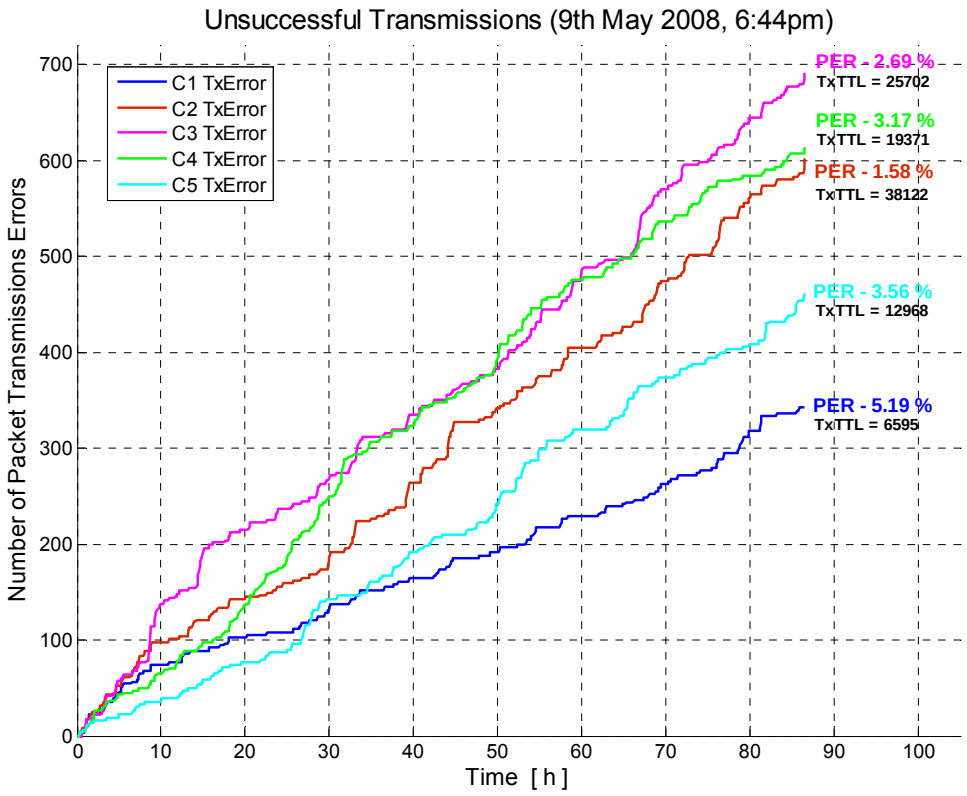

Figure 138: Number of unsuccessful packet transmission for individual nodes.

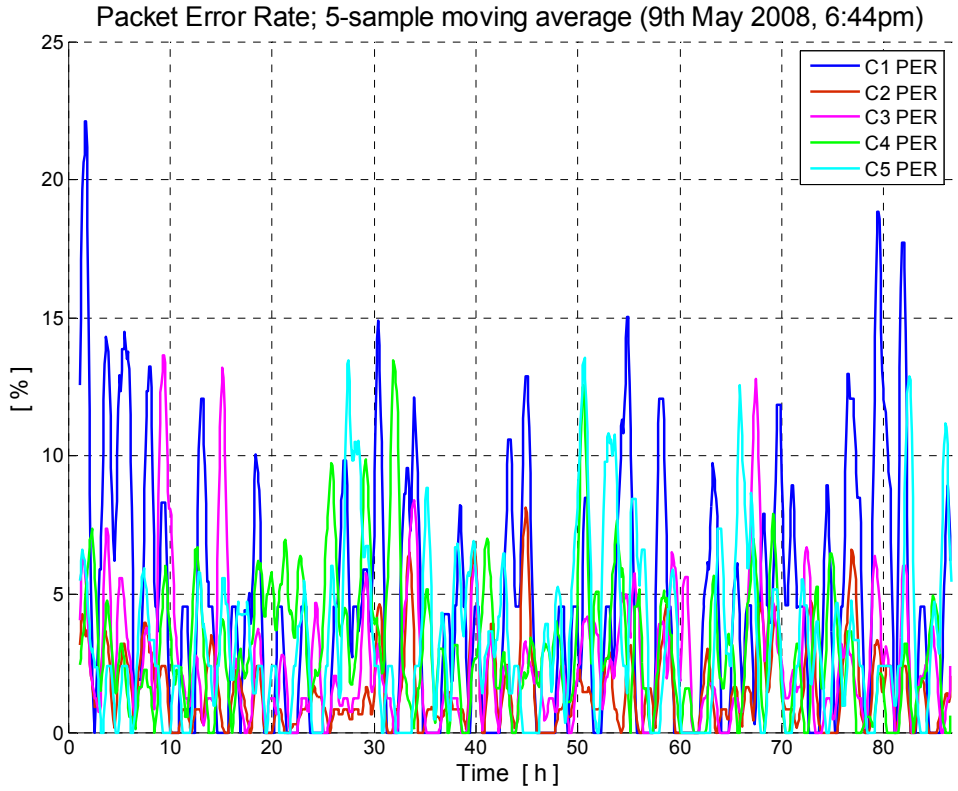

Figure 139: Packet error rate.

June 16th 2008

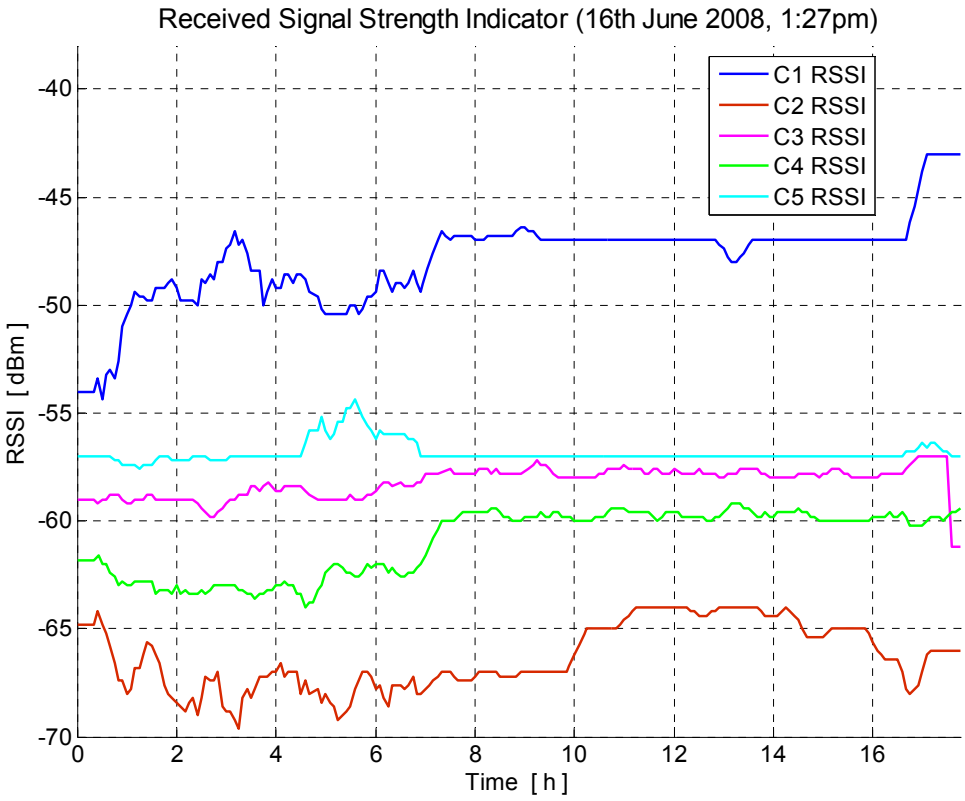

Figure 140: Signal strength towards the parent node in the default topology.

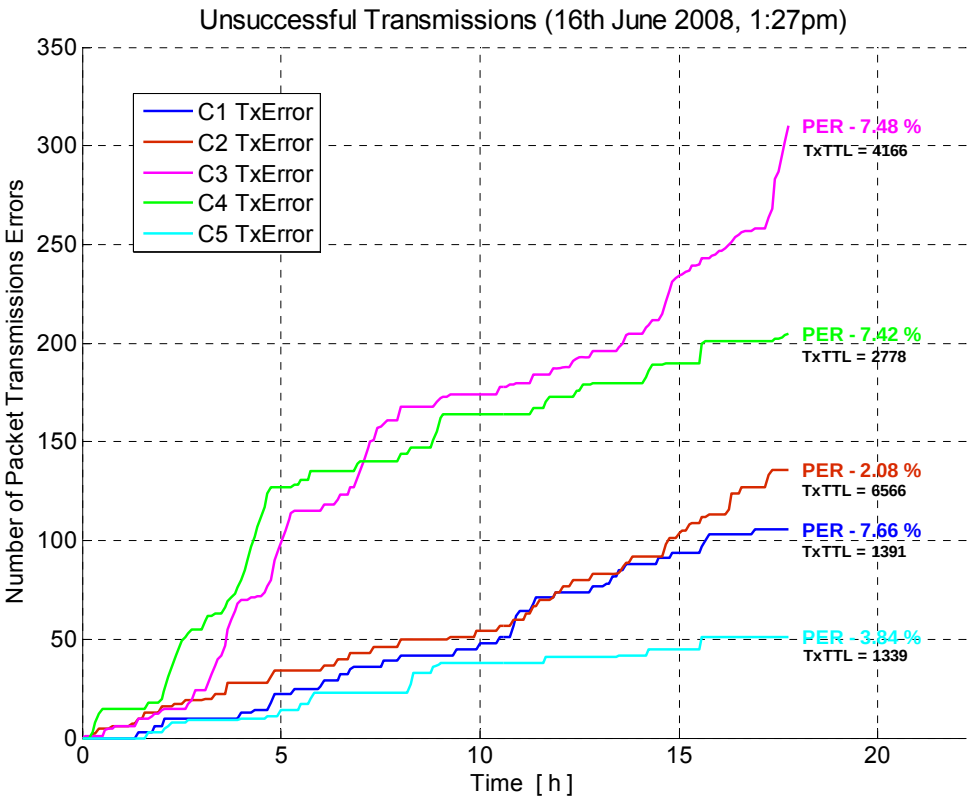

Figure 141: Number of unsuccessful packet transmission for individual nodes.

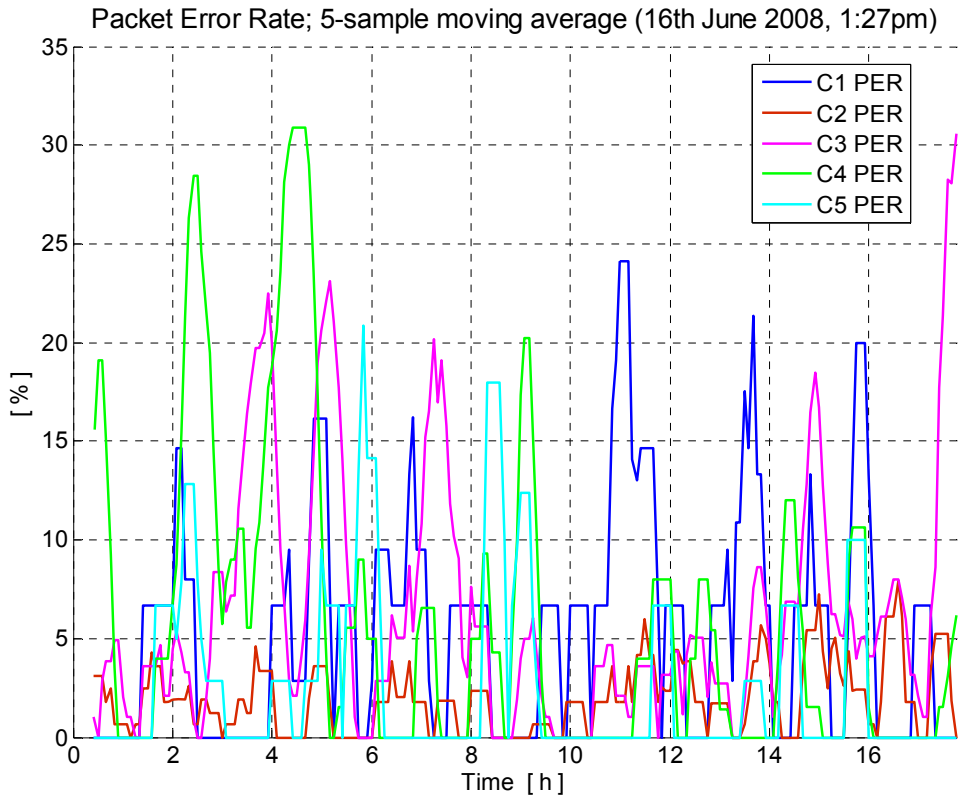

Figure 142: Packet error rate.

F. Performance Analysis of the Deployed WSN

F.1. Wireless communication – RSSI and PER

May 27th 2008

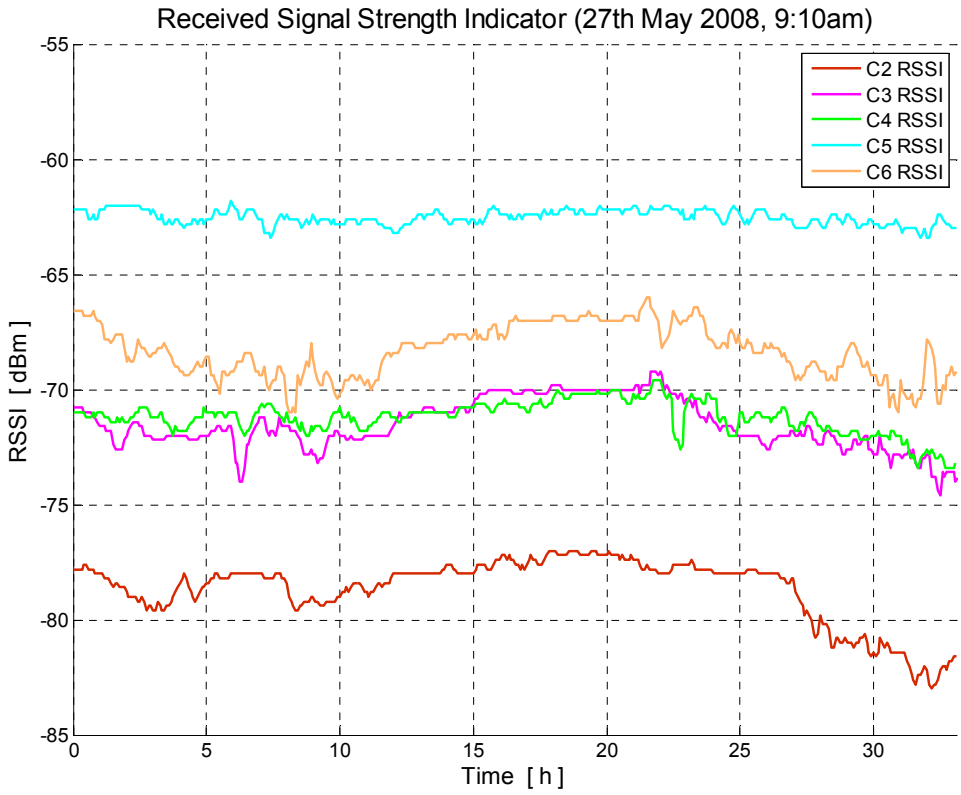

Figure 143: Signal strength towards the parent node in the default topology.

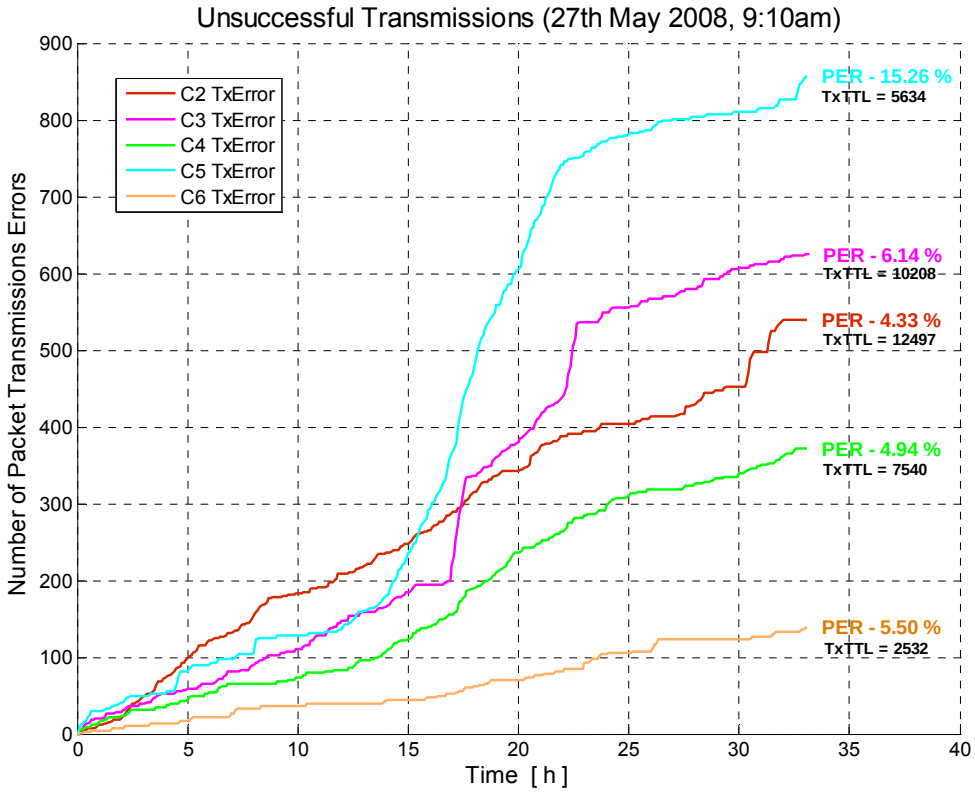

Figure 144: Number of unsuccessful packet transmission for individual nodes.

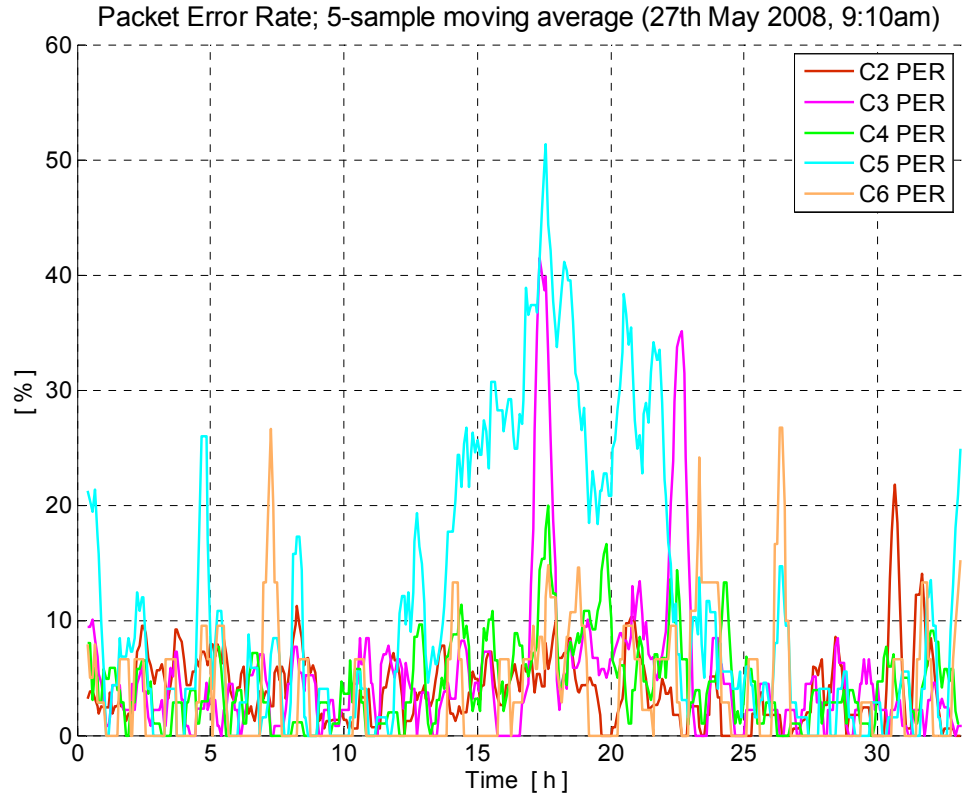

Figure 145: Packet error rate.

May 20th 2008

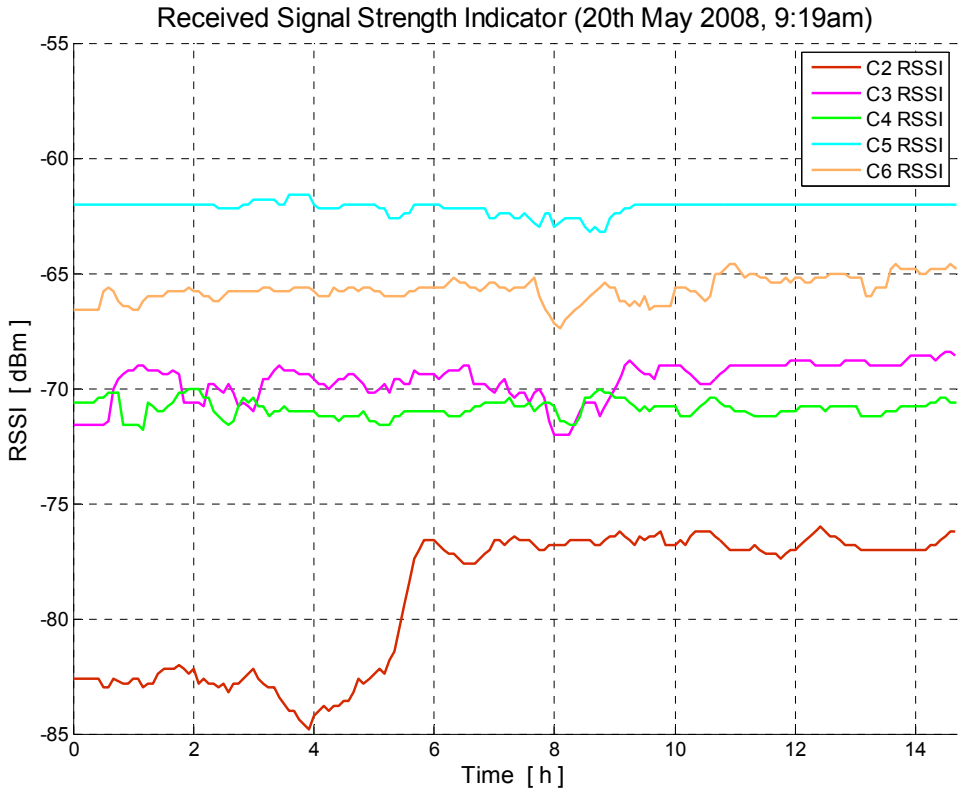

Figure 146: Signal strength towards the parent node in the default topology.

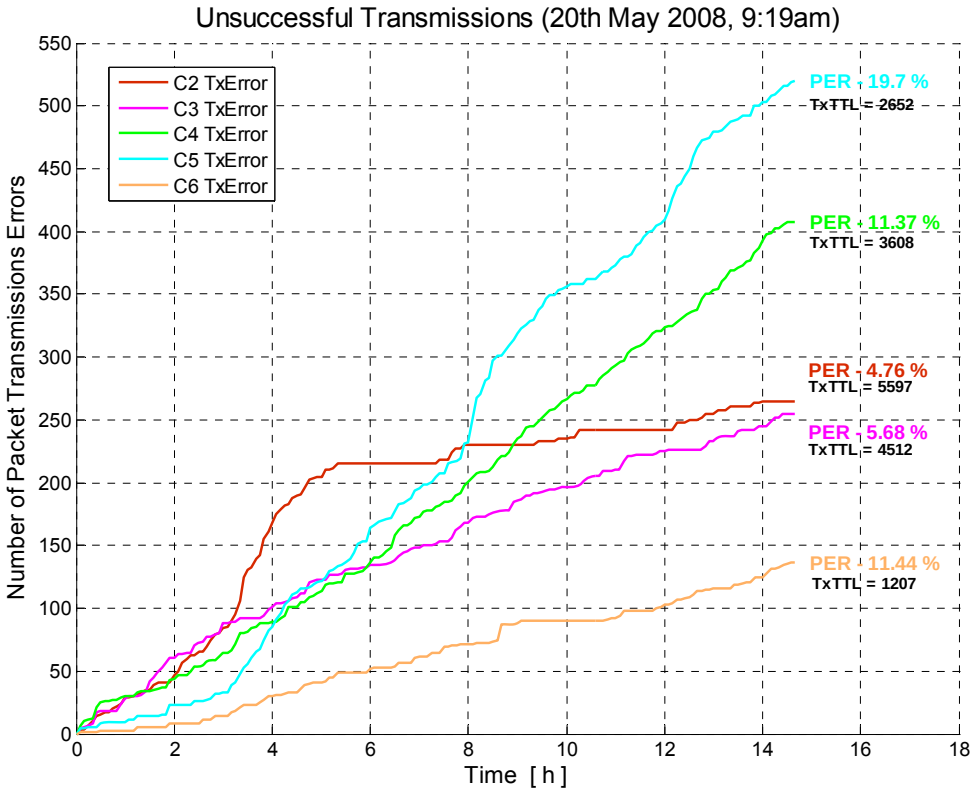

Figure 147: Number of unsuccessful packet transmission for individual nodes.

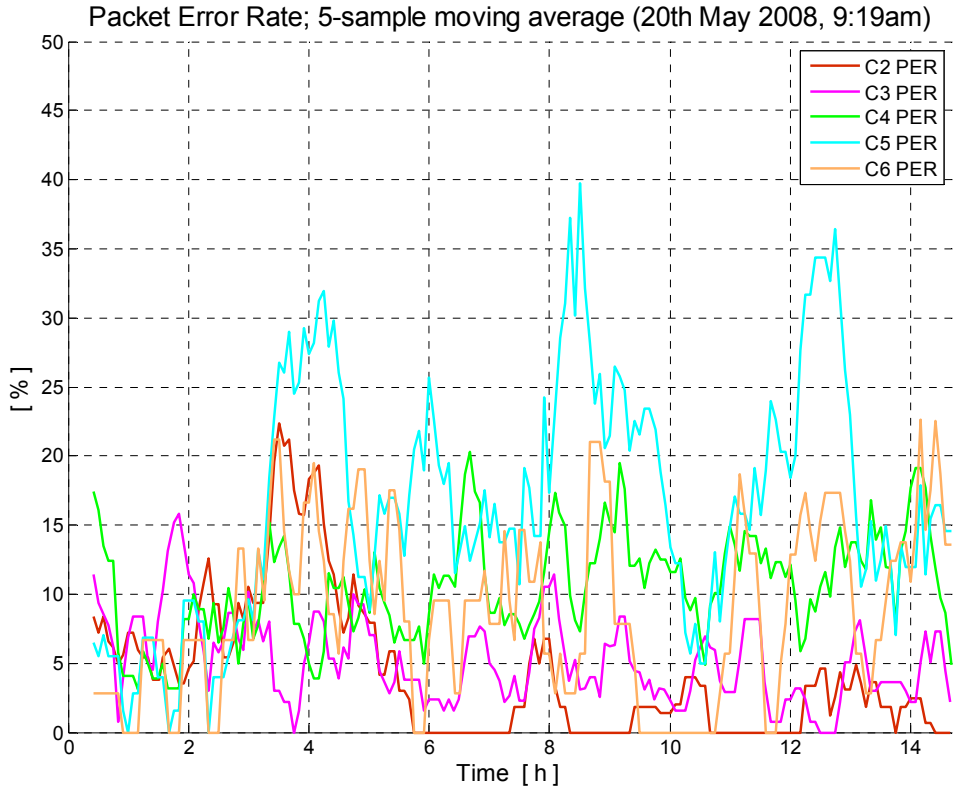

Figure 148: Packet error rate.

F.2. Detected Stay-Cables Natural Frequencies

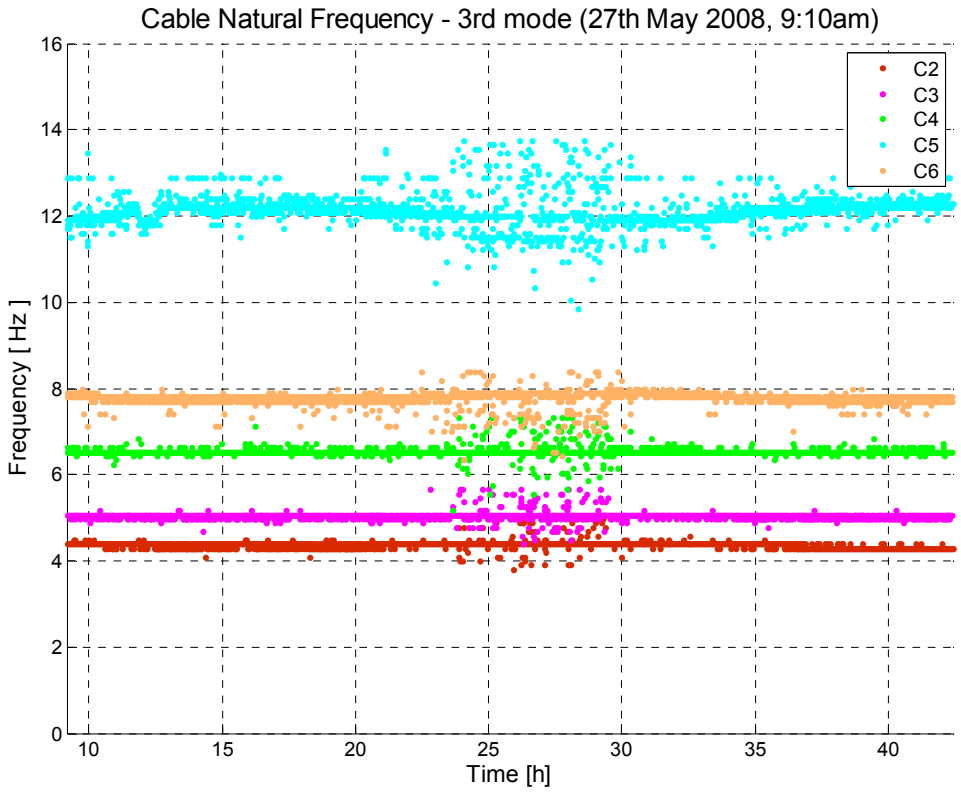

Figure 149: Detected 3rd natural frequency of monitored stay cables.

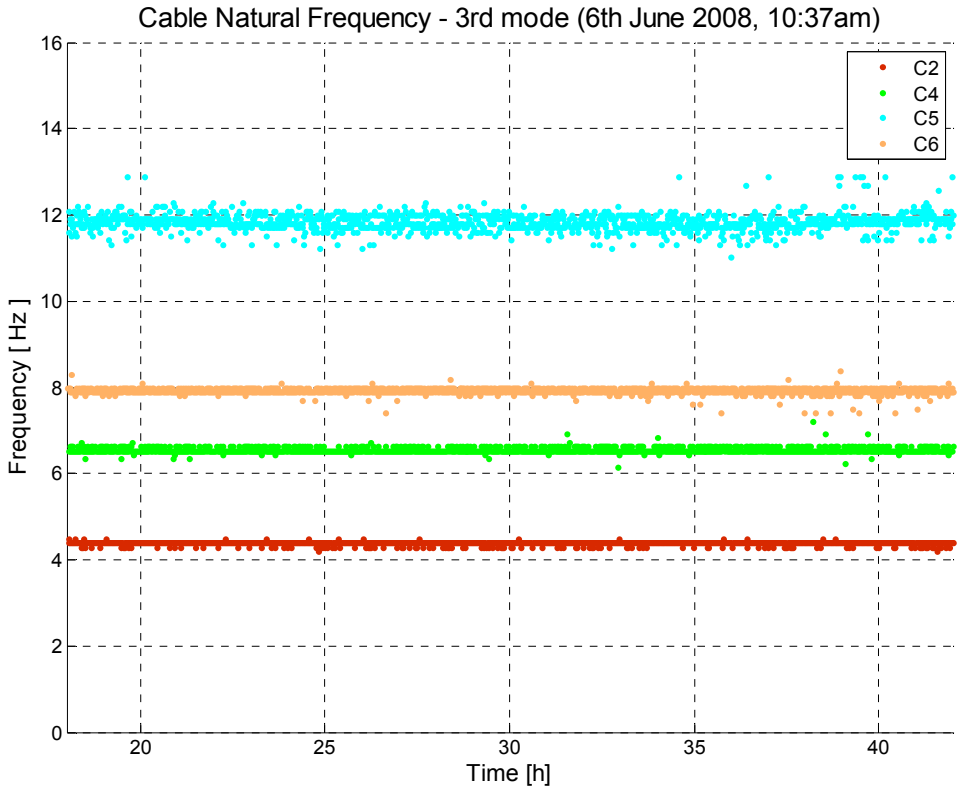

Figure 150: Detected 3rd natural frequency of monitored stay cables.

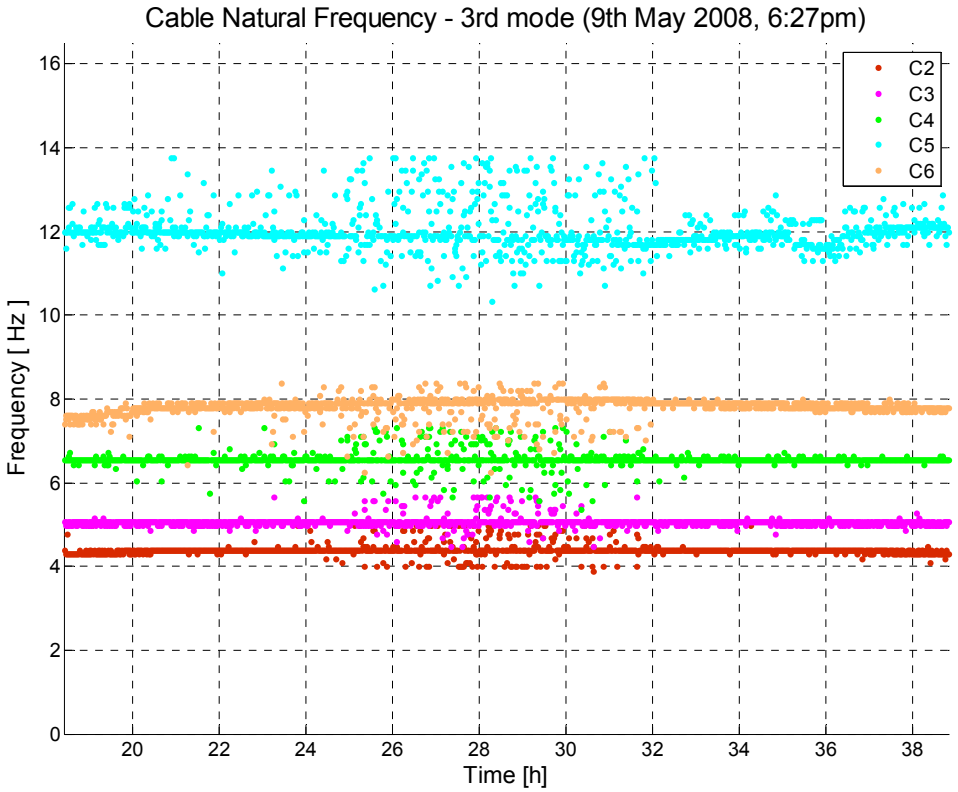

Figure 151: Detected 3rd natural frequency of monitored stay cables.

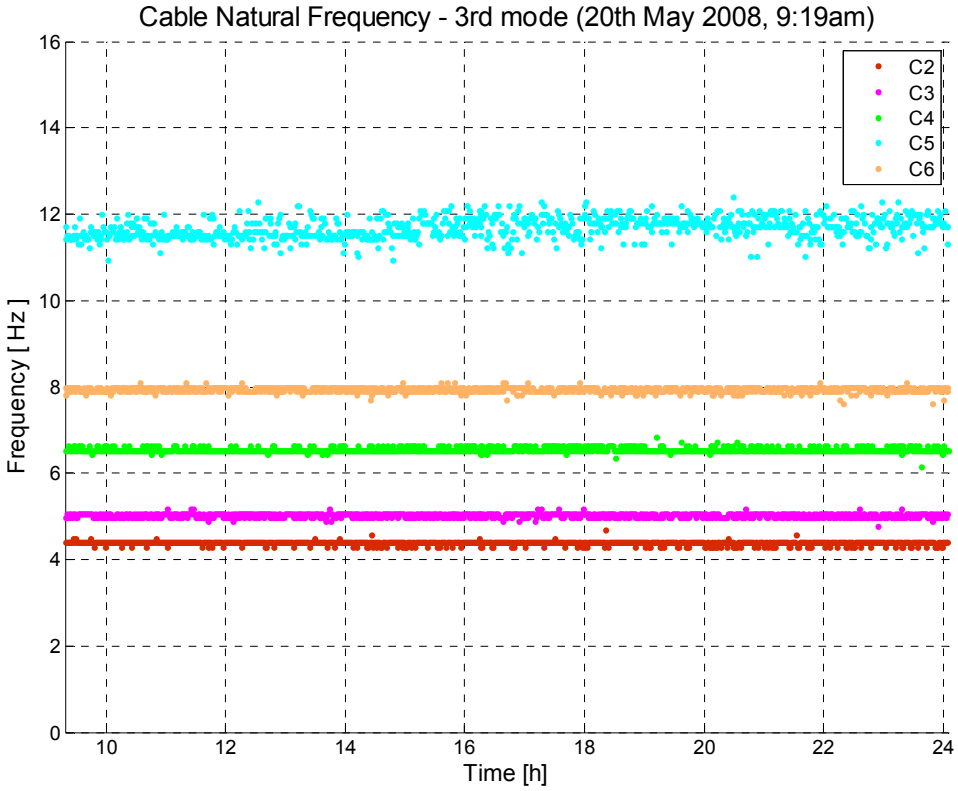

Figure 152: Detected 3rd natural frequency of monitored stay cables.

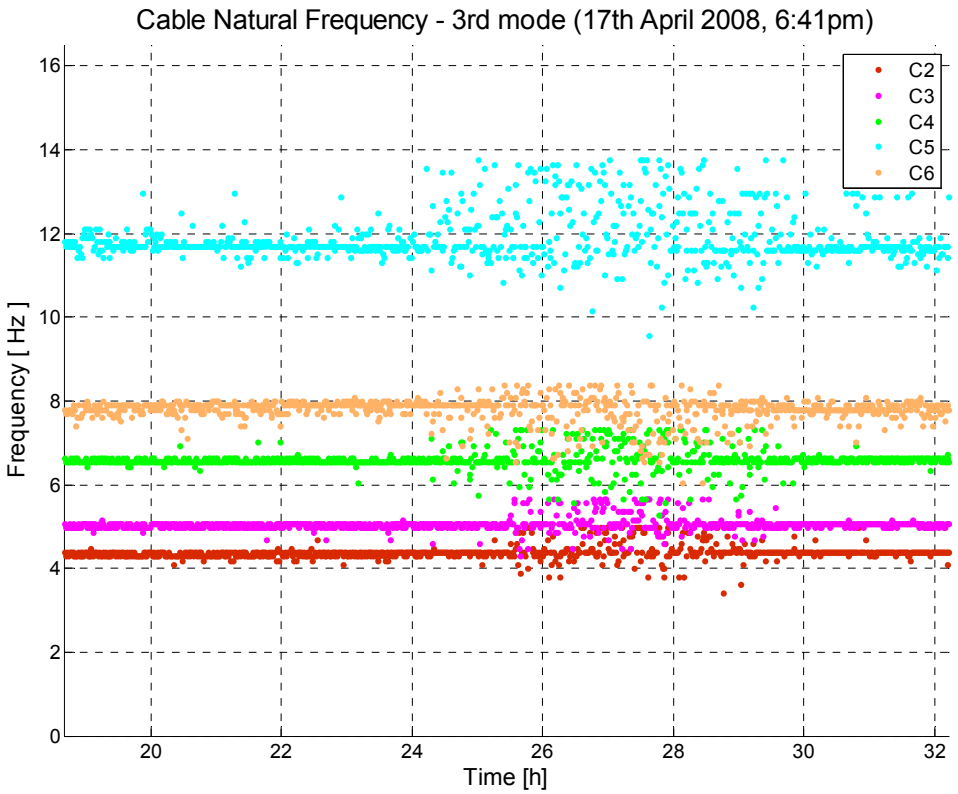

Figure 153: Detected 3rd natural frequency of monitored stay cables.

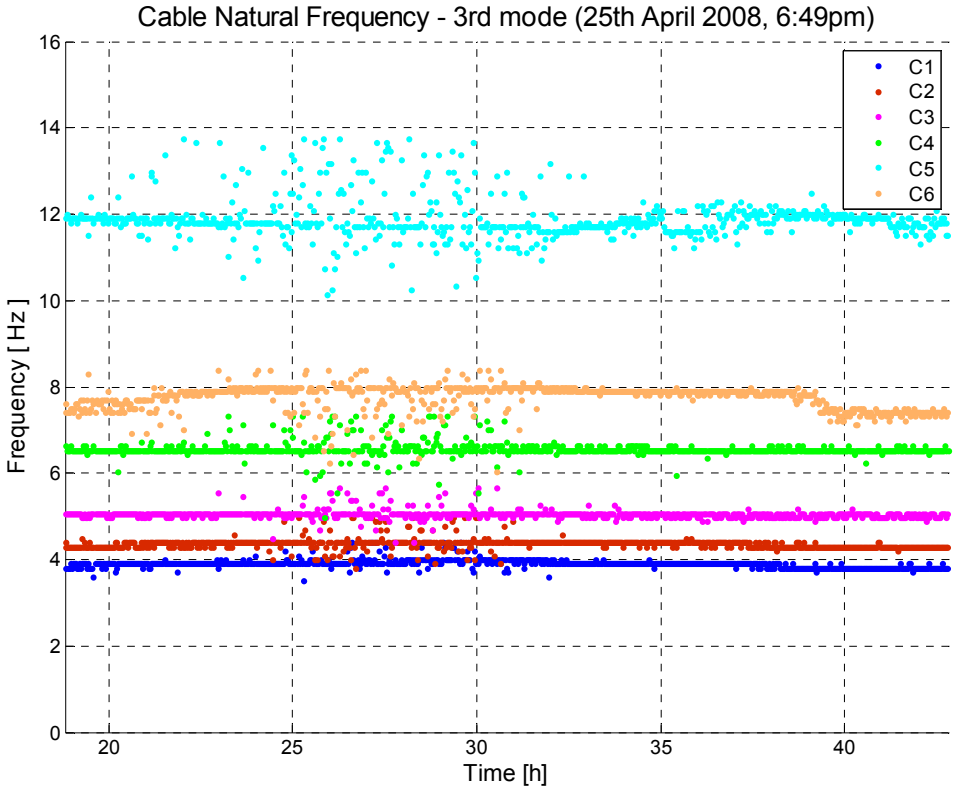

Figure 154: Detected 3rd natural frequency of monitored stay cables.

References

[1] Farrar, C.R. and K. Worden, An introduction to structural health monitoring. Philosophical Transactions Of The Royal Society A-Mathematical Physical And Engineering Sciences, 2007. 365(1851): p. 303-315.

[2] Doebling, S.W., et al., Damage Identification and Health Monitoring of Structural and Mechanical Systems from Changes in Their Vibration Characteristics: A Literature Review. 1996. LA-13070-MS, Los Alamos National Laboratory.

[3] Farrar, C.R., S.W. Doebling, and D.A. Nix, Vibration-based structural damage identification. Philosophical Transactions Of The Royal Society Of London Series A-Mathematical Physical And Engineering Sciences, 2001. 359(1778): p. 131-149.

[4] Sohn, H., et al., A Review of Structural Health Monitoring Literature form 1996-2001. 2004. LA-13976-MS, Los Alamos National Laboratory.

[5] Brownjohn, J.M.W., Structural health monitoring of civil infrastructure. Philosophical Transactions Of The Royal Society A-Mathematical Physical And Engineering Sciences, 2007. 365(1851): p. 589-622.

[6] Farrar, C.R., et al. Variability of Modal Parameters Measured on the Alamosa Canyon Bridge. in Proceedings of the 15th International Modal Analysis Conference: p. 257-263. 1997. Orlando, FL.

[7] Rohrmann, R.G., et al. Structural Causes of Temperature Affected Data of Civil Structures Obtained by Long Term Monitoring. In Proceedings of the 18th International Modal Analysis Conference: p. 1-7. 2000. San Antonio, Texas.

[8] Peeters, B. and G. De Roeck, One-year monitoring of the Z24-Bridge: environmental effects versus damage events. Earthquake Engineering & Structural Dynamics, 2001. 30(2): p. 149-171.

[9] Feltrin, G. Temperature and damage effects on modal parameters of a reinforced concrete bridge. In Proceedings of the Fifth European Conference on Structural Dynamics, EURODYN 2002: p. 373-378. 2002. Munich.

[10] Huth, O., et al., Damage Identification Using Modal Data: Experiences on a Prestressed Concrete Bridge. Structural Engineering ASCE, 2005. 131(12): p. 1898-1910.

[11] Yanev, B. Structural health monitoring as a bridge management tool. In Proceedings of the 1st International Conference on Structural Health Monitoring and Intelligent Infrastructure: p. 87-95. 2003. Tokyo, Japan.

[12] Farhey, D.N., Bridge instrumentation and monitoring for structural diagnostics. Structural Health Monitoring-An International Journal, 2005. 4(4): p. 301-318.

[13] Frangopol, D.M., A. Strauss, and S. Kim, Bridge Reliability Assessment Based on Monitoring. Journal of Bridge Engineering, 2008. 13(3): p. 258-270.

[14] Strauss, A., D.M. Frangopol, and S. Kim, Statistical, Probabilistic and Decision Analysis Aspects Related to the Efficient Use of Structural Monitoring Systems. Beton- und Stahlbetonbau, 2008. 103(S1): p. 23-28.

[15] Wong, K.Y., Instrumentation and health monitoring of cable-supported bridges. Structural Control and Health Monitoring, 2004. 11(2): p. 91-124.

[16] Desjardins, S.L., et al., Real-time data processing, analysis and visualization for structural monitoring of the confederation bridge. Advances in Structural Engineering, 2006. 9(1): p. 141-157.

[17] Rücker, W., et al., Merkblatt über die automatisierte Dauerüberwachung im Ingenieurbau. 2000. Deutsche Gesellschaft für Zerstörungsfreie Prüfung e.V.

[18] Mufti, A., et al., Guidelines for Structural Health Monitoring, Design Manual No. 2. 2001. ISIS Canada Corporation.

[19] Aktan, A., et al., Development of a Model Health Monitoring Guide for Major Bridges. 2002. Drexel Intelligent Infrastructure and Transportation Safety Institute, Drexel University.

[20] Culler, D.E. and H. Wei, Wireless sensor networks. Communications Of The Acm, 2004. 47(6): p. 30-33.

[21] Bischoff, R., J. Meyer, and G. Feltrin, Wireless Sensor Network Platforms, in Encyclopedia of Structural Health Monitoring, C. Boller, F. Chang, and Y. Fujino, Editors. 2008, John Wiley & Sons: Chichester.

[22] Levis, P., et al., TinyOS: An operating system for wireless sensor networks, in Ambient Intelligence. 2005, Springer-Verlag: New York.

[23] Lynch, J.P. and K.J. Loh, A summary review of wireless sensors and sensor networks for structural health monitoring. Shock and Vibration Digest, 2006. 38(2): p. 91-128.

[24] Olofsson, I., et al., Assessment of European railway bridges for future traffic demands and longer lives – EC project “Sustainable Bridges”. Structure and Infrastructure Engineering, 2005. 1(2): p. 93-100.

[25] Glaser, S.D. Some Real-World Applications of Wireless Sensor Nodes. In Proceedings, SPIE Symposium on Smart Structures & Materials/ NDE 2004 2004. San Diego, CA.

[26] Lynch, J.P., et al., Performance monitoring of the Geumdang bridge using a dense network of high-resolution wireless sensors. Smart Materials and Structures, 2006. 15(6): p. 1561-75.

[27] Kim, S., et al., Health Monitoring of Civil Infrastructures Using Wireless Sensor Networks. 2006. Technical Report No. UCB/EECS-2006-121, University of California, Berkeley.

[28] Mechitov, K., et al., High-Frequency Distributed Sensing for Structure Monitoring. Transaction of the Society of Instrument and Control Engineers (SICE), 2006. E-S-1(1): p. 109-114.

[29] Feltrin, G., J. Meyer, and R. Bischoff. Wireless Sensor Networks for Long Term Monitoring of Civil Structures. In EVACES 07, 2nd International Conference on Experimental Vibration Analysis for Civil Engineering Structures: p. 95-111. 2007. Porto, Portugal.

[30] Kiviluoma, R., et al., SB-8.2 Demonstration of bridge monitoring. 2007. Prepared by Sustainable Bridges - a project within EU FP6. Available from: www.sustainablebridges.net.

[31] The Sensor Network Museum^{tm} http://www.btnode.ethz.ch/Projects/SensorNetworkMuseum

[32] Enz C., El-Hoiydi A., Decotignie J.-D., Peiris V., WiseNET: An Ultralow-Power Wireless Sensor Network Solution. IEEE Computer, Vol. 37, Nr 8, august 2004, p. 62-70.

[33] Kroneberger-Stanton, K.J. and B.R. Hartsough, A monitor for indirect measurement of cable vibration frequency and tension. Transactions of the ASAE, 1992. 35: p. 341-346.

[34] Casas, J.R., A combined method for measuring cable forces: the cable-stayed Alamillo Bridge, Spain. Structural Engineering International, 1994. 4: p. 235-240.

[35] Ladret, P., M.A. Belinchón Márquez, and J.R. Casas. Inspection of cable forces in cable-stayed bridges using a modified taut string method. First International Conference on Bridge Maintenance, Safety and Management (IABMAS 02): p. 171-172. 2002. Barcelona.

[36] Geier, R., G. De Roeck, and J. Petz, Cable force determination for the Danube Channel bridge in Vienna. Structural Engineering International, 2005. 15(3): p. 181-185.

[37] Feltrin, G., J. Meyer, and R. Bischoff. A Wireless Sensor Network for Force Monitoring of Cable Stays. Third International Conference on Bridge Maintenance, Safety and Management (IABMAS'06): on CD, 2006, Porto.

[38] MSP430x15x, MSP430x16x, MSP430x161x Mixed Signal Microcontroller, Datasheet, Texas Instruments
http://focus.ti.com/lit/ds/symlink/msp430f1611.pdf

[39] Tmote Sky: Ultra low power IEEE 802.15.4 compliant wireless sensor module, Moteiv

[40] LIS3L02AS4 MEMS Inertial Sensor: 3-Axis - +/-2g / +/- 6g Linear Accelerometer, ST Microelectronics
http://www.st.com/stonline/products/literature/ds/10221.pdf

[41] Precision Micropower, Low Noise CMOS Rail-to-Rail Input/Output Operational Amplifiers AD8603/AD8607/AD8609, Analog Devices
http://www.analog.com/UploadedFiles/Data_Sheets/AD8603_8607_8609.pdf

[42] Philip Levis: TinyOS Programming

[43] David Gay, Philip Lewis, David culler, Eric Brewer: nesC 1.1 Language Reference Manual

[44] Efficient Multiplication and Division Using MSP430, Texas Instruments
http://focus.ti.com/lit/an/slaa329/slaa329.pdf

[45] Greg Morton, Kripasagar Venkat: MSP430 Competitive Benchmarking (Rev. B)
http://www-bsac.eecs.berkeley.edu/~slanzise/courses/100/labs/docs/microcontroller benchmarks.pdf

[46] A. El-Hoiydi, P. Volet, “Host Controller Interface”, CSEM Technical Report, Version 1.3 File HostControllerInterface.pdf.

[47] Jonathan Hui: Deluge 2.0 – TinyOS Network Programming
http://www.cs.berkeley.edu/~jwhui/deluge/deluge-manual.pdf

[48] Hammond Watertight Enclosure, Technical Specification
http://www.farnell.com/datasheets/66487.pdf

[49] Tadiran, SL-2880 Datasheet
http://www.tadiran-batterie.de/download/eng/LBR06Eng.pdf

[50] Tadiran, Technical Broschure
http://www.tadiran-batterie.de/download/eng/SL-2880.pdf

[51] Energizer L91, Product Datasheet, Energizer
http://data.energizer.com/PDFs/l91.pdf

[52] A. El-Hoiydi and J.D. Decotignie, WiseMAC: An Ultra Low Power MAC Protocol for Multi-hop Wireless Sensor Networks. In Proceedings of the First International Workshop on Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS

2004), Lecture Notes in Computer Science, LNCS 3121, pages 18-31. Springer-Verlag, July 2004.

[53] A. El-Hoiydi, Spatial TDMA and CSMA with Preamble Sampling for Low Power Ad Hoc Wireless Sensor Networks. In *Proc. IEEE Int. Conf. on Computers and Communications (ISCC)*, pages 685-692, Taormina, Italy, July 2002.

[54] A. El-Hoiydi, Energy efficient medium access control for wireless sensor networks, PhD Thesis, EPFL, no 3285 (2005)

[55] A. El-Hoiydi, P. Volet, “WiseNode Platform Overview”, CSEM Technical Report, File WiseNodeV2_Overview.pdf, Document version 1.1.

[56] A. El-Hoiydi, “WiseNode Platform Overview”, CSEM Technical Report, File WiseNodeV4_Overview.pdf, Document version 1.2.

[57] LS 33600 SAFT Lithium Battery. Datasheet http://www.accu-profi.de/doku/ls_33600.pdf

[58] Energizer E95, Product Datasheet http://data.energizer.com/PDFs/E95.pdf

[59] MSP430x1xx Family User’s Guide, Texas Instruments http://focus.ti.com/lit/ug/slau049f/slau049f.pdf