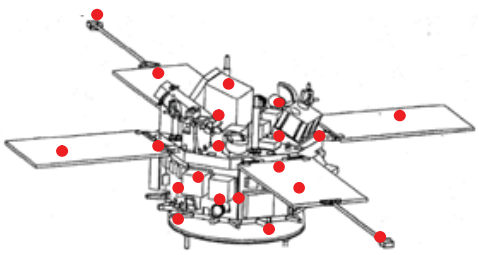
Computer Engineering
Mekelweg 4,
2628 CD Delft
The Netherlands
http://ce.et.tudelft.nl/

2011

MSc THESIS

Fault-Tolerant Platform for Intra-Spacecraft Modular Wireless Sensor Network

M. Beekema



CE-MS-2011-09

Abstract

Wireless sensor networks are becoming ubiquitous in terrestrial applications. In this thesis we present a platform for wireless on-board spacecraft sensor networks with focus on the sensors that bring high 'added value' to existing space systems while becoming wireless. A novel multi-gateway architecture that uses dual hardware redundancy is proposed in this thesis. The concept of an autonomous (self-describing) modular sensor node (i.e. exchangeable sensors and power sources) is investigated and a general provision is made for future implementations. Electronic Datasheets (EDS) are considered as a way to allow fast self-reconfiguration of the sensor node and its energy device. Next, a survey on commercial off-the-shelf components (COTS) wireless sensor network hardware and protocols is conducted. A test with a Zigbee-based gateway switchover is performed and its impact on the wireless communication is investigated with or without check pointing. The results of this thesis show that current wireless sensor network systems can be adapted for several space-borne applications and can significantly shorten integration-test time (during spacecraft assembly) with the help of a modular based sensor system. Additionally, the limitations and future directions are discussed to provide possible future implementations of the next generation wireless sensor networks for intra-spacecraft applications.

Faculty of Electrical Engineering, Mathematics and Computer Science

Fault-Tolerant Platform for Intra-Spacecraft Modular Wireless Sensor Network

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER ENGINEERING

by

M. Beekema
born in Doetinchem, Netherlands

Computer Engineering
Department of Electrical Engineering
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

Fault-Tolerant Platform for Intra-Spacecraft Modular Wireless Sensor Network

by M. Beekema

Abstract

Wireless sensor networks are becoming ubiquitous in terrestrial applications. In this thesis we present a platform for wireless on-board spacecraft sensor networks with focus on the sensors that bring high 'added value' to existing space systems while becoming wireless. A novel multi-gateway architecture that uses dual hardware redundancy is proposed in this thesis. The concept of an autonomous (self-describing) modular sensor node (i.e. exchangeable sensors and power sources) is investigated and a general provision is made for future implementations. Electronic Datasheets (EDS) are considered as a way to allow fast self-reconfiguration of the sensor node and its energy device. Next, a survey on commercial off-the-shelf components (COTS) wireless sensor network hardware and protocols is conducted. A test with a Zigbee-based gateway switchover is performed and its impact on the wireless communication is investigated with or without check pointing. The results of this thesis show that current wireless sensor network systems can be adapted for several space-borne applications and can significantly shorten integration-test time (during spacecraft assembly) with the help of a modular based sensor system. Additionally, the limitations and future directions are discussed to provide possible future implementations of the next generation wireless sensor networks for intra-spacecraft applications.

Laboratory : Computer Engineering
Codenumber : CE-MS-2011-09

Committee Members :

Advisor:	Dr. ir. Georgi N. Gaydadjiev, CE, TU Delft
Chairperson:	Dr. Koen Bertels, CE, TU Delft
Member:	Dr. ir. Anthony Lo, WMC, TU Delft
Member:	Dr. ir. Arjan van Genderen, CE, TU Delft
Member:	Ir. Maxime Castera, ISIS B.V.

"There are those that look at things the way they are, and ask why?
I dream of things that never were, and ask why not?"

– Robert F. Kennedy

Contents

List of Figures

List of Tables

Acknowledgements

This work was performed at Innovative Solutions in Space (ISIS) B.V. in Delft in collaboration with Aerospace-Wireless B.V. and Clyde Space in the UK. The author wishes to acknowledge the support of Clyde Space and the ISIS management.

ISIS is a specialist in the field of experimental nanosatellites. The company employs about 20 staff working on various nanosatellite projects, research and product development, product sales and services, as well as turnkey nanosatellite missions. The ISIS staff was highly involved in the design of the Delfi-C3 spacecraft in general and in the integration and tests of the Autonomous Wireless Sun Sensor experiment in particular. The spacecraft is still being operated by ISIS staff. The company is a workpackage leader in the Dutch MicroNED MISAT project which is aimed at developing MEMS and Microsystems Technology (MST) for use in small spacecraft.

For this thesis, I would like to thank M. Castera, E. van Breukelen, J. Rotteveel (ISIS) and special thanks go to R. Magness (Aerospace-Wireless), J. F. Dufour (ESA), J. Guo, R. Sun (TUDelft SSE) and S. Charlick (ClydeSpace). There are also a lot of other inspiring people at ISIS who I would like to thank for advice, support and help.

I would also like to thank all of my (current and previous) professors that brought me to where I stand now.

M. Beekema
Delft, The Netherlands
June 27, 2011

1 Introduction

1.1 Introduction

As of today, many wireless interfaces are already dominating in the terrestrial areas such as WLAN, bluetooth and others. In the aerospace domain wireless communication is successfully used many times before, from satellites to lander–rover (planetary, i.e. fixed-mobile) communications. The implementations of wireless sensor network applications however are just new and booming in the last few years.

Wireless solutions can be applied in many different subdomains in space engineering to solve challenging problems, ranging from lander-rover communications, manned space-flight mobility issues to inter-satellite communications for formation flying. This study, however is focused on using wireless sensor networks for intra-satellite sensor networks and the (re)use of wireless sensors on ground during Assembly, Integration and Testing (AIT) activities. From a technological implementation aspect, the focus lies on modular sensor nodes with a wireless radio-frequency (RF) link and drop-in modules for various types of sensors and power generation.

In the past years there are a lot of WSN developments, initiatives, flight tests, and standardization efforts for spacecraft and aircraft. Some examples include the demonstration of the Delfi-C3 Wireless Sun Sensor by TNO [16], new R&D projects by numerous companies and institutes like the structural heath monitoring (NASA) and Integrated Vehicle Health Monitoring (IVHM) for monitoring and analysis directly on the spacecraft. Already completed wireless standards documents by CCSDS, ESA and NASA provide guidelines and accepted standardization for possible applications inside and outside spacecraft systems.

For the space industry, wireless technology offers a number of specific advantages over existing (wired based) solutions:

- It simplifies development phase, and therefore reduces design-time and brings down development costs. E.g., wireless sensors allow cost reduction through their more flexible infrastructure. Also the use of wireless networks allows for more flexible placement of sensors, with a great freedom as the impact of the addition or removal of sensor nodes on the overall reconfiguration is minimal.
- It also simplifies AIT activities and therefore reduces design-time and brings down development costs. A strong reduction or even the elimination of cable harnessing, which is prone to workmanship errors and complicates spacecraft configuration and integration.

- Mass reduction: With the improvements of sensors and controllers miniaturization, smaller form factors are possible for sensor nodes, however, space approved power and data connectors do not scale down that easily and are currently often the determining factor in the sensor physical dimensions.

With all these advantages in mind, we conclude that wireless systems can avoid many of the problems during design time, however possible solutions to implement a wireless system into a spacecraft will require completely different design approach. Still, there are many significant challenges. Higher bandwidth technologies and increased complexity also significantly increase the time and effort to realize flight-ready wireless sensor systems.

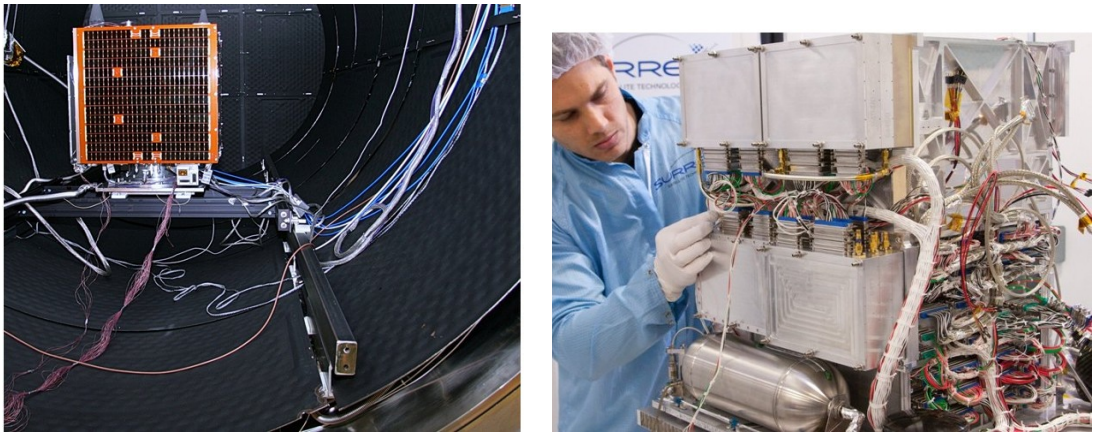
This thesis focuses on the first phases that define the concept and provides a basis for discussion to realize the vision into real hardware and software. In this thesis we focus on a low-data rate sensor network communication platform. Wireless sensors can be both fixed or mobile, the main focus (and goal) in this thesis lies in fixed (or static) sensor networks inside satellite systems. Mobility, however, is often required during the installation or configuration of a satellite, so additional focus in this thesis is for the application of easy reconfiguration of the network and its components. Information about quick and easy reconfiguration of the sensor network, can be found in the sensor network electronic datasheets. This can be applied to the connected sensors to a node and used to eliminate faults that could possibly be made by personnel during installation and assembly of the spacecraft. A review of most suitable existing (possibly wireless) electronic datasheets is performed and compared. A working setup of the platform composed of a dual-redundant gateway is tested out and followed by an evaluation of the impact on redundancy, availability, bandwidth and timing behaviour.

1.2 Problem Statement

Most of the problems in satellites are related to the physical aspects of having wires running across several locations, and the problems associated to the wires themselves. Furthermore, long-lengthy cables require different sensor conditioning circuitry than short wired cables. During AIT (Assembly, Integration & Test), deploying the cable harness required for a mission takes careful planning, as well as risk of damages during integration and test. Other harness bottlenecks are forcing designers to make use of bulky connectors or reroute to alternative (manually accessible) locations. During a test, the sensors and wires need to be taped down to be secured, locations have to be written down and dedicated holes inside test chambers have to be provided for accessing the sensors.

AIT phase wiring harness

The testing phase on the ground (a test bench) is one of the most important phases in the production of a satellite. Spacecraft assembly, integration & test is the process of assembling all the components of the flight vehicle and verifying that the spacecraft



(a) (b)

Figure 1.1: External wiring harness during AIT operation (a) and Intra-Spacecraft wiring harness, for in-orbit communications (b) [70]

operates correctly before launch. Testing requires that all sensors, buses and components are excited at least once and probed for correctness. Typically the spacecraft is shaken, frozen, heated-up, subjected to vacuum and irradiation. This testing is typically performed in (sealed) laboratory environments. Therefore, in a fully sealed spacecraft, cabling has to be taken outside the spacecraft (see Figure 1.1(a)). Furthermore, the test harness cabling often has to meet or exceed the requirements for in-orbit spaceflight.

Intra-spacecraft wiring harness

Many of the AIT phase problems can be directly seen back in the intra-spacecraft operational phase. The intra-spacecraft sensors are always routed in a more static way than during AIT, and imply more constrained placement requirements, as the devices have to withstand the launch and should operate flawlessly during its mission, far longer than during the AIT lifetime. Where the sensors during AIT are mobile, the sensors inside the spacecraft have static wire lengths and are optimized to operate on that length. Figure 1.1(b) demonstrates the internal wiring harness of the UK-DMC-2 satellite, built by Surrey Satellite Technology Ltd (SSTL). Other problems with wiring can be seen back in the deployables on the satellite (i.e. structures that are folded, swing out or collapsed by spring mechanisms on the spacecraft).
Examples of bottlenecks on deployables that cause most of the prominent problems inside the satellite:

- Rotating slip rings for deployable components, like solar panels;
- Antenna booms which require long lengthy cabling;
- Switches and latches (for verification of deployment status monitoring) require long wires.

1.2.1 Advantages of wireless communications

For the space industry and during AIT phase, wireless technology offers a number of specific advantages over existing solutions.

Simplified harness and reduced mass

For the current wired solutions, cables are long and with wireless solutions, fewer cables are used and worried about. As a result, harness is less bulky and can be more easily integrated into the harness.

Another improvement of wireless communication solutions is the mass reduction: with the improvements of sensor and controller miniaturization, smaller form factors are possible for sensor nodes, however, space qualified power and data connectors do not scale down that easily and are currently often the determining factor for the sensors physical dimensions.

While this was the initial goal of the thesis, it turns out that with the additional mass of the wireless sensor components (power supply and other components) the total mass reduction is only about 2% when applied to a typical spacecraft [20].

Improved visibility

In order to monitor the data on a wired spacecraft onboard bus it is necessary to physically attach the monitoring equipment. This influences the bus load immediately. Also, dedicated connectors (called *skin connectors*) are positioned on the rim of the spacecraft so that they are accessible also when the spacecraft is fully equipped (see Figure 1.2). Additional mass and harness comes from the fact that these connectors must be dedicated and reliable (the connectors chosen must allow regular mating and demating without any contact quality degradation).

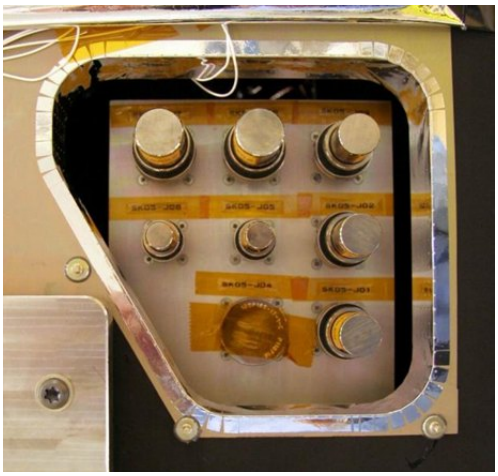


Figure 1.2: Example skin-connectors on the ESA Herschel/Planck satellite [5]

By contrast, wireless interfaces can be easily monitored, and do not require any physical connection. In the wireless network, a special configuration mode might be setup that transmits data also to a "test-node", located outside the spacecraft. Therefore total

visibility of data flowing across interfaces in each direction can be monitored and possibly be even stimulated with test-data. Also, the monitoring of discrete interfaces to sensors is even more difficult with current wired setups. It has often not have been attempted, instead, this testing part is placed on special software applications or software routines running on the flight-computers [53].

Easier retrofit & upgrading

For the above advantages, the main advantage is during assembly and integration phase of the satellite project. Retrofitting means the addition of components not foreseen in the original design of a spacecraft. In current wired solutions this is extremely difficult as it involves laying cabling and the provision of new connection points (like connectors/pins). Retrofitting on orbit is only possible when using man serviced missions, such as in space stations or serviceable satellites.
Upgrading wireless devices may be simpler than wired devices. It is basically easier to swap wireless devices as they:

- Do not need to be located at the same place the old sensor was;
- Do not require to mate up with old interfacing components (interoperability).

1.3 Thesis Overview

1.3.1 Purpose

The purpose of this thesis is the specification and development of a (modular) fault-tolerant wireless sensor network platform for space-borne applications, with the focus of medium to small-sized satellites. The end goal of the thesis should be considered as the first platform or prototype which will be further refined. The targeted final version of the platform lies well beyond the scope of this thesis. The thesis aims at working towards a fault-tolerant platform, which can be used for small-scale experiments and hands-on exploration with custom software, such as operating systems, protocols and middleware.

1.3.2 General structure

The introduction provides an overview of the challenge of a wireless sensor network for a satellite. It briefly explains the fundamental problems during integration-test phase of a satellite system. In addition, contemporary gives an overview of the benefits a wireless system will have over the conventional wired systems.

Chapter 2 covers a broad review of the intra-spacecraft sensors and networks in satellites as well as present wireless systems are reviewed and compared. Sections 2.1 to 2.5 defines the current wired sensor networks first and some related (flight-tested) wireless systems within the aerospace domain. Sensor candidates are listed and compared based on data rates and wireless applicability in section 2.6. Fundamentals of modularity for satellite instruments and typical sensors considered in typical small satellites are discussed in sections 2.7 to 2.8. Finally, section 2.9 covers a general introduction and fundamentals to fault tolerance and dependability in spacecraft systems.

Chapter 3 details a survey and selection of standards-based wireless, Commercial-Off-The-Shelf (COTS) protocols, hardware and characteristics. Section 3.1 lists most important requirements and goals for the system. Network topologies are analyzed and a proposed system topology for the sensor network is given in section 3.3 he selection of a generic protocol and a comparison with tradeoffs was part of the study and a summary is given in section 3.4. The last section discusses and validates the chosen development platform to propose a basic framework for the end application.

Chapter 4 describes the platform architecture design in detail. In section 4.3 hardware modularity is proposed for the sensor node and in section 4.3.3 examines the possibilities of the extension to a wireless modular solution. Section 4.3 provides research of viability of adding additional testing sensors during the test or verification phase in AIT. It also defines a concept wireless architecture for during AIT operations.

Chapter 5 details the platform test setup and in section 5.2 a general gateway redundancy scheme is proposed. Section 5.2 discusses how gateway redundancy was implemented. Performance results and discussion is given in section 5.3. The last section concludes with an overview of the platform and its envisioned architectural components.

Chapter 6 discusses the research findings, limitations and recommendations for the next phases of development and a conclusion. It also highlights areas of future research and presents a future (evolution) architecture for satellite construction.

2 Background & Related Work

In this chapter a survey related to the topic is given, to point out the many contributions of previous research and previous platforms that are already existing and operational. We organize this survey around the two main themes of the research on intra-spacecraft sensor systems: in-orbit (intra) satellite sensor systems, and integration-test based sensor systems (or during AIT operations). In both cases, background information is provided to the reader. The first section starts with an overview of current sensor systems and continues to describe relevant sensors for space applications in section 2.2. Section 2.3 describes details of the commonly used housekeeping sensors and systems in satellites. Other wireless related systems and sensors to satellites and/or a spacecraft are discussed in section 2.4. Lastly, relevant Attitude and Determination Control System (ADCS) sensors and current wireless systems related to this kind of sensors are discussed in section 2.5. An overview of all investigated sensors and candidates for wireless integration or during AIT is summarized in section 2.6. Modularity will be discussed in section 2.7, and section 2.8 provides a comprehensive overview of (current and wireless) interfacing in a spacecraft and for wireless sensors. Finally, an introduction to spacecraft system dependability and fault-tolerance is covered in section 2.9.

2.1 Introduction to Sensors, Networks and Wireless

Transducers

Transducers are devices that transform one form of energy to another. Any device which converts energy is called a transducer. A sensor is used to detect a parameter in one-form of energy and reports it itypically as electrical signals.

Actuators use the energy provided on their input and produce mechanical movement (actions). In a satellite system, there is also a possibility for wireless actuators, but in general, actuation implies higher reliability for the link between actuator and control-subsystem, the On-board Computer (OBC). Generally, actuation systems do require reliable links with high availability (like in launch systems[1]), or more robustness that might not be achieved with current wireless COTS components.

Since the primary focus of this thesis is a *sensor-network*, actuators are left for future use of the system and might require (significant) changes in system architecture.

[1]Typically, transducers in launch systems do not require real time links, but availability and accuracy is very important for the telemetry and telecommand system. Detailed information for wireless launch systems is shown in the CCSDS report, *"Wireless Network Communications Overview for Space Mission Operations"* (November 2010), E-6.

Wireless sensor networks

A wireless sensor network consists of physically small sensor nodes networked together by a common protocol. They may be embedded unobtrusively in their environment, and are typically distributed in order to monitor their surroundings (e.g., measuring temperature, or motion detection, barometric pressure, etc.). Many of the current Wireless Sensor Networks (WSNs) are aimed to deliver high-quality and accurate sensor data while operating during long stretches of time with minimal cost and maintenance needs. The majority of these networks assume a large number of nodes to be deployed in terrestrial applications, and are designed to cope with node-failure.

Due to the inherent fact that wireless sensor networks are highly application-specific, development and deployment is quite complex in the sense of architectural design and multi-constrained approaches. There is no unified, *"one-size fits-all"* solution for the current growing number of applications. Many details still have not yet been understood and limited research with simulation and actual experimentation has been conducted. Also, the case that the environment is not terrestrial, and the harsh-conditions of space add additional complexity to the system.

Wireless point-to-point sensors

Point-to-point RF communication links are notoriously variable and unpredictable. A wireless link that is strong today may be weak tomorrow due to environmental conditions, new obstacles, unforeseen interferers and other factors. Additionally, power surges, blackouts, or brownouts can cause nodes to fail. Any of these problems will bring down a point-to-point wireless link. However, with a network architecture designed to protect against these issues, the network can isolate individual points of failure and eliminate or mitigate the failure impacts, allowing the network as a whole to maintain very high reliability in spite of local failures.

An example of a dedicated point-to-point sensor link is the autonomous wireless sun-sensor of TNO [16] that is using a single channel wireless link. The aim for this thesis is however to form a network of multiple linked-sensors, to have improved robustness and a simpler (uniform) integration of sensors (during AIT phase).

2.2 Typical Applications of Wireless Sensing for Satellites

Wireless solutions can be applied in many different subdomains in space engineering to solve challenging problems. For example in stationary-mobile (e.g. lander-rover) communications for planetary exploration, manned spaceflight mobility issues (spacesuit health monitoring [43]) and inter-satellite communications for formation flying where communication-latencies issues can be challenging [79, 86, 18]. This study, however is focused on using wireless sensor networks for intra-satellite sensor networks and the use of wireless sensors on ground during AIT activities. From a technological implementation point of view, the focus lies on modular sensor nodes with a wireless RF link and drop-in modules for various types of sensors and power generation.

The monitoring of the health, status and behavior of a spacecraft or its subsystems is an important aspect during the spacecraft mission. Many sensory inputs are used for health monitoring in space, as much as 400 sensors for a medium class mission. Sensor types include: thermistors for temperature measurement, accelerometers for attitude control and measuring launch loads and radiation sensors to monitor the particle environment. The current state of the art is to interconnect these sensors and the central computer using wires.

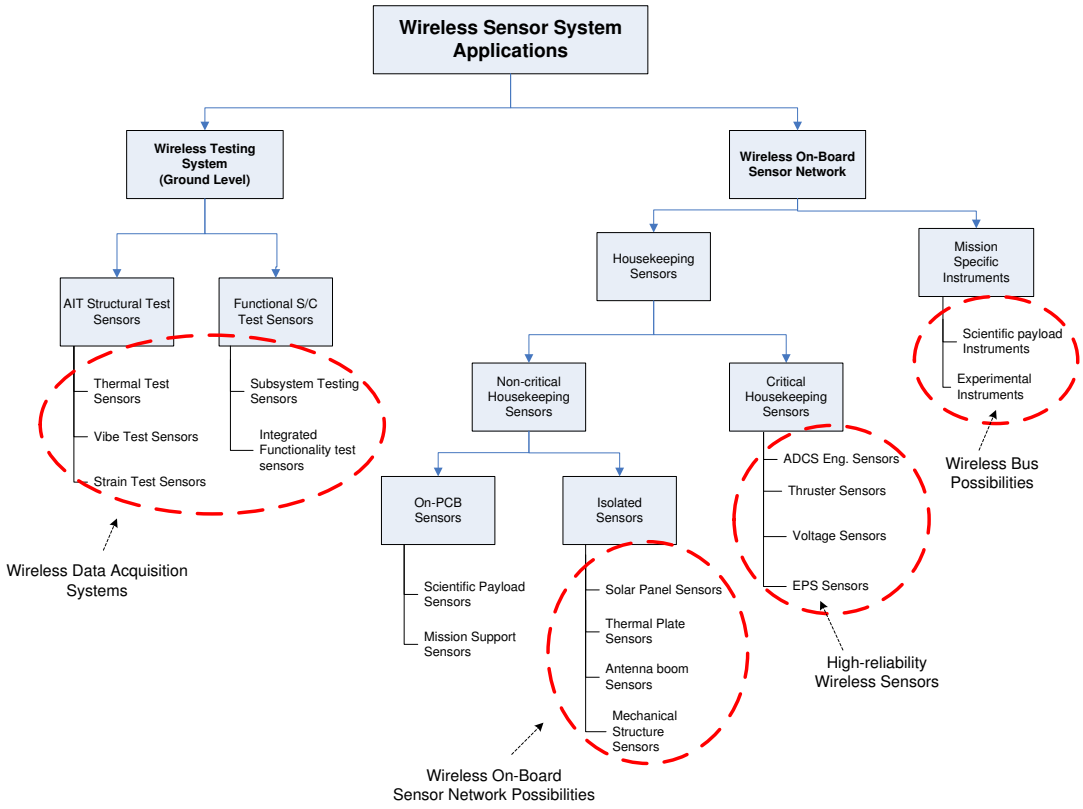


Figure 2.1: Applications of a possible wireless sensor networks for spacecraft

Using wireless networks to connect these sensors to each other and to the central computer of the spacecraft can greatly reduce the cable harness required for a mission and thus reduce mass, as well as damage risks during integration and test. In addition, the use of wireless networks can greatly simplify the use of additional sensor in places that are hard to reach for wired sensors, such as at the end of external appendages or deployables. Some typical applications for wireless sensors inside a satellite system can be seen back in figure 2.1. Typical applications range from Wireless Data Acquisition systems to sensors for functional tests and various on-board sensors used during the spacecraft mission. They all pose different characteristic demands on different layers of the wireless communication systems. Details, example scenarios and the impact of the applications on the wireless system are discussed in the following sections.

Mission specific instruments

Mission specific instruments, like the imaging (payload) sensors for scientific earth-observations systems, are most of the time dedicated instruments with high-speed data rates and propriety protocols. Other examples of mission specific sensors include:

- Hyperspectral imaging sensors for earth-observation satellites (e.g. LandSat [35]);
- Magnetic field sensor with a dedicated digital processing unit (from the NEAR satellite [48]).

The data acquired by the spacecraft's scientific instruments, in support of scientific experiments, are commonly referred to as "science" data. The other set of data, often referred as "engineering data" is composed of the state, health, safety and diagnostic data that is transmitted to the ground an used in control and monitoring of the satellite. Sometimes, spacecraft engineering data is necessary for science data reconstruction, e.g. an instrument should monitor its attitude, and inserts this data into the instrument data headers. For these general mission specific (or science) sensors, the best wireless solution might be to implement a wireless-bus based system, dedicated to the functionality of the mission sensors.

Spacecraft housekeeping

Spacecraft housekeeping data includes sensors that monitor the health of the aircraft, such as temperature and battery voltage, flight conditions, such as airspeed, pointing directions, etc. Housekeeping data is typically stored in the computer memory of the satellite.

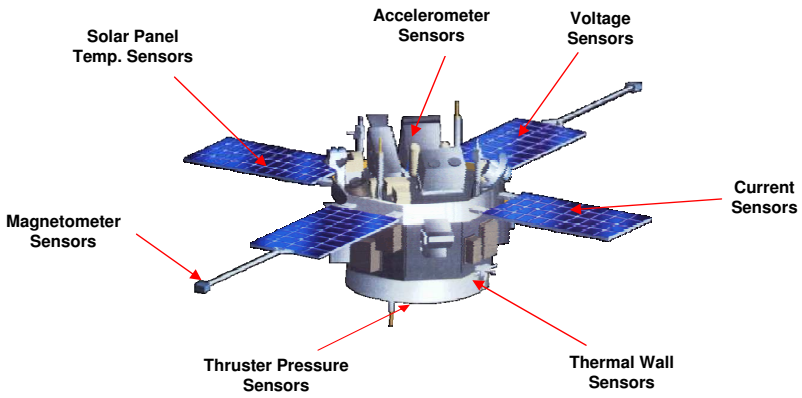


Figure 2.2: Example sensors in the ACE (Advanced Composition Explorer) satellite

Typically, the housekeeping sensors are scattered over all locations in the satellite, as shown in figure 2.2, yet all sensors are relatively densely populated in a compared to terrestrial sensor networks, where spaces between sensors can be hundreds of meters. Best suitable applications for a wireless network, are these housekeeping sensors, and possibly the low-priority (non-critical) housekeeping sensors. Other parts of the spacecraft typically have a behavior that is not suited for wireless networks. Examples are

high-data rates when testing spacecraft components (e.g. accelerated life tests imply high bandwidth data acquisition streams), or critical subsystems inside the spacecraft that can not deal with additional delays, or require a firm hard real-time deadline. Data for housekeeping is almost always sampled in recurring intervals. For instance, spacecraft housekeeping data might be sampled on a regular 2.5 second interval, while science data might have arbitrarily spaced time intervals.

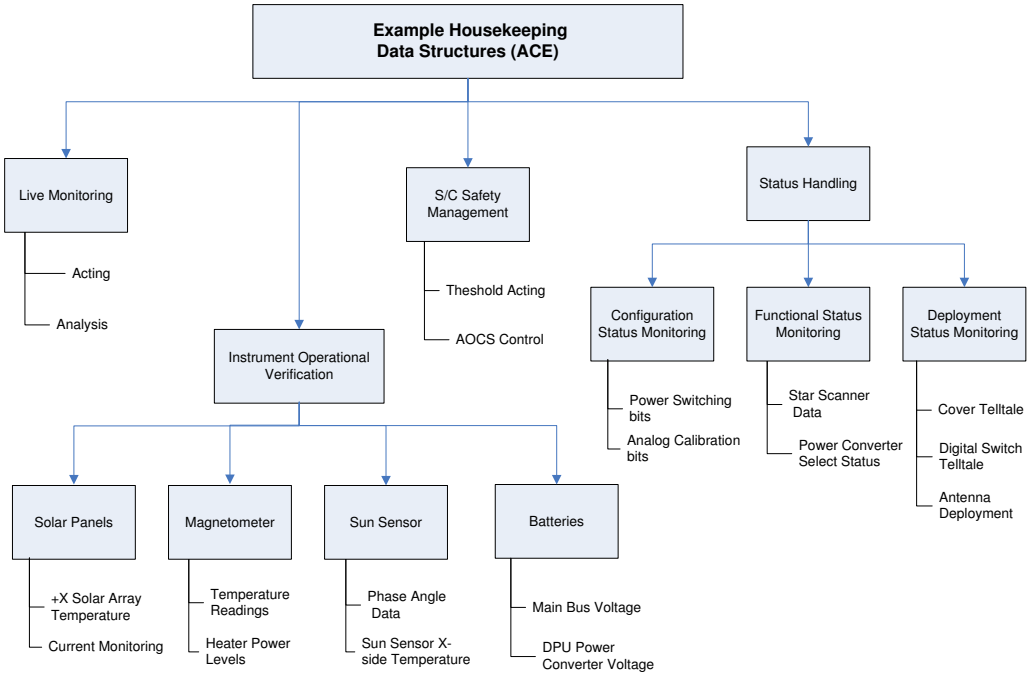


Figure 2.3: Example housekeeping structure inside a typical spacecraft (parts taken from the ACE spacecraft [15])

Figures 2.2 and 2.3 describe some examples of the ACE housekeeping structures [15]. Various one-bit telltale switches are maintained by the on-board computer and serve as indicators of problem conditions. Other sensors are located near the instruments and provide support for verification of instrument operation or problems. Examples are temperature sensors near the sun-sensor, which is a vital component for a spacecraft, or near its solar panels.

During AIT phase

During AIT phase, the spacecraft is typically tested thoroughly by using test-benches, and include various engineering tasks such as thermal tests, shock tests, mechanical tests and Electromagnetic Compatibility and Interference (EMC/EMI) prediction and analysis. Specifically for the wireless systems onboard the spacecraft (such as telemetry and telecommand), undesirable electromagnetic coupling between the subsystems which are closely packed within the spacecraft are a major concern for AIT engineers. For

AIT operations there are already existing solutions called *"Wireless Data Acquisition Equipment"* (NASA) [37], Wireless Data Loggers, etc. Typical testing sensors are based on reusable "sticking" or "bolt-on" to the parts that need monitoring, like strain sensors, pressure sensors [51] and *"lick-n-stick"* leak detection sensors [24]. Other glue compounds also exist for reusing and easy removing of the sensing element. One typical strain gage (wich are used for measuring stresses on outer body) is the Vishay CEA-13-250UW-120, and can be used for crack or fatigue testing. These sensors are used to test spacecraft structures, like the Gamma-Ray Large Area Space Telescope (GLAST) from NASA [14].

The fact is that current wired sensor systems for test purposes have to be robust in many ways to tolerate the intended test, as you do not want the to be tested system give unreliable results during testing. Replacing test wiring might bring advantages when the locations of the sensors are difficult to reach during the setup of the test-bench.

2.2.1 Comparing phases

Table 2.1 illustrates the differences for a wireless system under test conditions and a wireless sensor system to be integrated and operated in space. The main differences show that the wireless sensors when using on the terrestrial environments are designed for a dedicated range that the tested system will operate in. Inside an operating spacecraft, the environmental requirements will be longer lasting than during AIT operations, where test operations consist of accelerated lifetime tests at the exceeded (in-orbit) aerospace requirements. Intra-spacecraft wireless systems (during orbit operation) operate more on an autonomous way: the spacecraft is under remote monitoring, where during AIT spare power (e.g. batteries) and spare parts are easily accessed. Also, the majority of the spacecrafts built are configured statically, such that wireless nodes should be optimized for operation of long lifetimes and operate fully autonomously.

Operation Phase	**Wireless Data Acquisition (AIT phase)**	**Intra spacecraft Wireless Sensing (in-orbit)**
Example application	Structural vibe testing	Spacecraft health monitoring
Operational Duration	Several hours/days	Several years
Operational Environment	Dedicated to (test) environment	Broad range
Maintenance Intervention	Manual	None (autonomous)
Reusability of nodes	Yes	No
Resources: – Power – Spare parts	 – unlimited – unlimited	 – minimal – none
Node Interoperability	High	Low

Table 2.1: Comparison between wireless sensor systems for on-board spacecraft and ground testing phase

Another notable difference is that during operation phase, the intra-spacecraft wireless sensor network is operating more or less continuously (i.e. real-time) and autonomic,

where as during testing (or AIT) phase the data delivery is more or less driven to be *observer initiated*. That means, during the AIT phase, data might be stored intermediately, as the interest for AIT is in the set of the whole test-bench data, during a certain test setup.
The aim for this thesis is that some particular sensors attached to the spacecraft during AIT might "spin off" in the flight qualified operation quickly if the wireless sensors are reliable enough to cope with the extreme testing environments. Possible spin-off targets of the wireless sensors could include, but are not limited to:

- Wireless sensor networks for planetary exploration [9]
- Wireless intelligent sensors inside manned spacecrafts (health prognostics) [27, 28]
- Wireless sensors for rocket health management or for satellite launchers [9, 36]

2.3 Spacecraft Housekeeping Sensors

Housekeeping is considered to be one of the key functions inside a spacecraft. Housekeeping data is the result of the monitoring of the spacecraft's health and operating status.

A generalized overview of housekeeping data is presented in this paragraph, to give a clearer understanding about this kind of data inside the spacecraft and how it is delivered back to the users for further analysis.

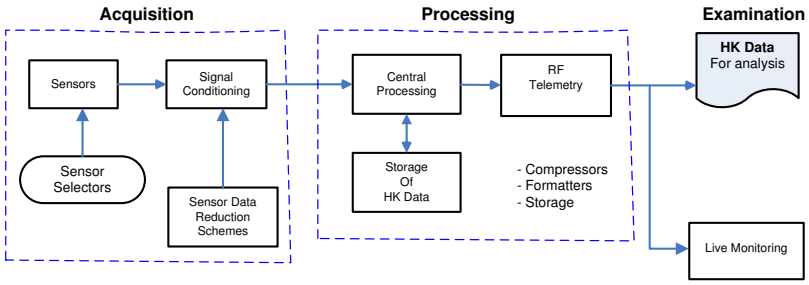


Figure 2.4: Typical housekeeping data-flow inside satellites

Housekeeping data can be physically separated in housekeeping data intrinsic to the satellite framework itself and mission-based housekeeping data (support for the mission). In practice, no distinction is made, as they both go through one link to the ground station. Most of the satellites merge these two data-sources and process them separately afterwards. While the payload also might have support housekeeping data (such as monitoring its imaging-lens temperature), this kind of data is in later stages being processed as mission support housekeeping data.

2.3.1 General data flow

Figure 2.4 explains a typical housekeeping setup used for spacecrafts currently. It can be split into acquisition (the process of sensing and conditioning), processing (the collection, storage and uplink), and examination (typically at mission control on the ground).

Example housekeeping data from two sensors is shown in figure 2.5, where data is plotted for analysis from inside of a typical satellite. Most of the time for in-orbit satellites, housekeeping data is sent back (via downlink) to the ground, and during pre-processing an assessment is made based on the priority of (some of) the housekeeping data. For example, high-priority housekeeping data can be attitude determination data, while low-priority housekeeping data is normally provided by analog devices like thermocouples and other spacecraft hardware [84]. This split of priority will eventually lead to less telemetry data overhead, and limits bandwidth that is available for downlink.

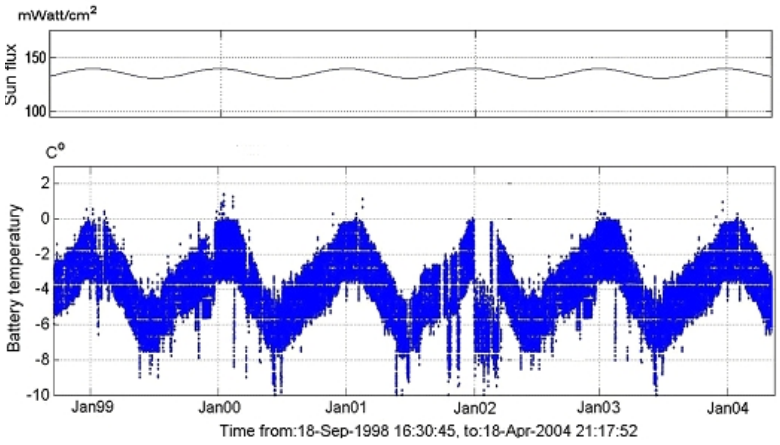


Figure 2.5: In-orbit electric battery temperature measurements and sun flux variation during 1998-2004, from the TechSat Gurwin satellite [26]

Traditionally, the spacecraft health monitoring system relays pressure, temperature, voltage, strain and acceleration data back to the Mission/Launch Control center. Integrated Vehicle Health Monitoring (IVHM) goes a step further by providing onboard processing capability. An IVHM system can often detect spacecraft system anomalies earlier and respond faster than a ground-linked system.

2.3.2 Data acquisition optimizations

Many existing schemes exist to improve the quality of the sensor acquisition. The focus is not only on improving overall data quality (like accuracy or benefiting from larger collections of data), but focuses on *data-reduction*. Data reduction is the problem of reducing data traffic (by using techniques based on data streams) while still assuring a minimum data quality that allows to reduce energy consumption and delay. While, at the end it saves power expenditures, the level of implementation is basically determined by the application, and willingness to give away some quality of the data in order to maintain a fully operational system within specified boundaries. In satellites, data reduction studies are basically created by the utilization of specific models from space engineering, like reducing ADCS sensor streaming intervals in [3]. This is an example of a system level optimization to improve the quality and usability of data. In wireless data acquisition these solutions can be seen as not only optimizing the quality of data, but also a means for

power management to conserve (overall) energy consumption. In many wireless sensor network devices, transmission power dominates over processing power [22], depending on sensors connected and what data goes over the link. The primary goal is to reduce this tranmsission burden by utilizing strategies, algorithms or other methods. All these optimizations refer to a class of sensors that have intelligent behavior, based on the actual application level of the system.

Other examples are *compression techniques* (which reduces bit rates and data storage), *data compaction* (by using run-length encoding schemes, etc), *adaptability*, *sensor-aggregation*, *correlation sensing* (e.g. *compressive sensing* [25]), etc. These solutions are also quite applicable inside the wireless network, as in principle they generate processing cycles in return for transmission power cycles.

As the generic housekeeping data is less sophisticated than other (more intelligent) sensors, data processing might not be effective at the end (e.g. it consumes more power by the processing itself), however data reduction schemes can contribute to more insight of redundant data in the system that can be avoided.

2.3.3 Housekeeping organisation & characteristics

In order to monitor the data inside the spacecraft, an On-Board Data Handling subsystem (or OBDH) is necessary to processes the flow of housekeeping (and science) data. In most of the cases, like BIRD[49], this is done by the central On-Board Computer (or OBC), however in some cases this unit is called an Integrated Housekeeping Unit (IHU). Because the flight-computer is basically executing housekeeping tasks (most of the time), it is evident that such a unit is called a housekeeping computer. Examples are the Suit-Sat satellite[43] and other amateur radio satellites, like OSCAR 9[52], AMSAT[80] and BlueSat[47].

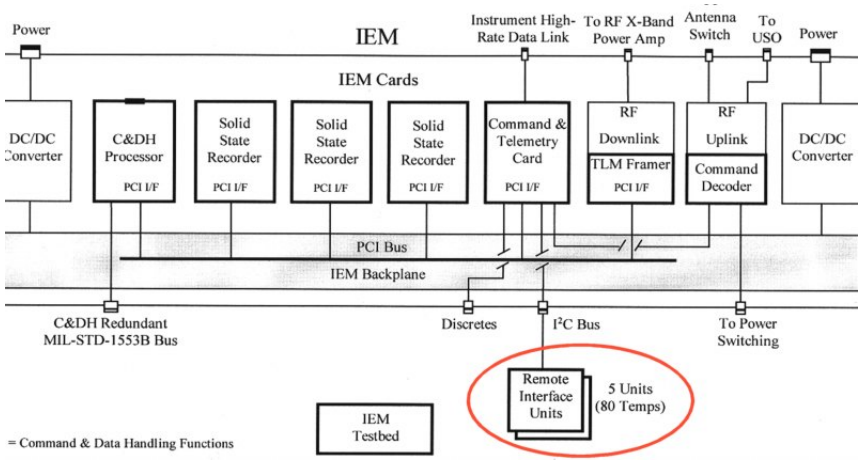


Figure 2.6: NASA STEREO spacecraft integrated electronics module configuration (note the RIUs, in the circle) [66]

Another case of an external dedicated housekeeping unit is called a "RIU", a Remote Interface Unit which collects, converts and buffers spacecraft temperature information,

and is used in various NASA mission satellites like the STEREO spacecraft[66] (see figure 2.6) and the Hubble Space Telescope (HST). The RIUs, as described from the following citation from the pre-phase-A study (from [66], in Chapter 4), gives an idea of how the RIU instruments are conceptually designed and is interconnected:

> *"Spacecraft temperature information will be monitored, collected, converted from analog to digital and buffered by five remote telemetry units, each of which is capable of acquiring data from 16 temperature sensors. A total of 80 temperature thermistors can be monitored in this fashion. The five units will communicate with the C&T subsystem in the IEM via a serial digital Inter-Integrated Circuit (I2C) bus. Each of the five units are daisy chained together via the bus and connected to the IEM. Precautions are required to mitigate the likelihood of a failure in a single RIU that disables the entire I2C bus. The baseline STEREO Remote Interface Unit (RIU) design is an existing TIMED design which can be replicated with no required changes... (p4-3—4-4)"*

The RIUs in the STEREO spacecraft are all identical and replicatable, a clear advantage for fast design and assembly in the spacecraft without modification. Downsides are the precautions that are required for each of the units to not disable the I2C bus, in case of failures.

In (most) other satellite systems, like CHAMP [64], both housekeeping and scientific data are combined into one subsystem (the OBDH), since most instruments provide scientific as well as housekeeping-data.

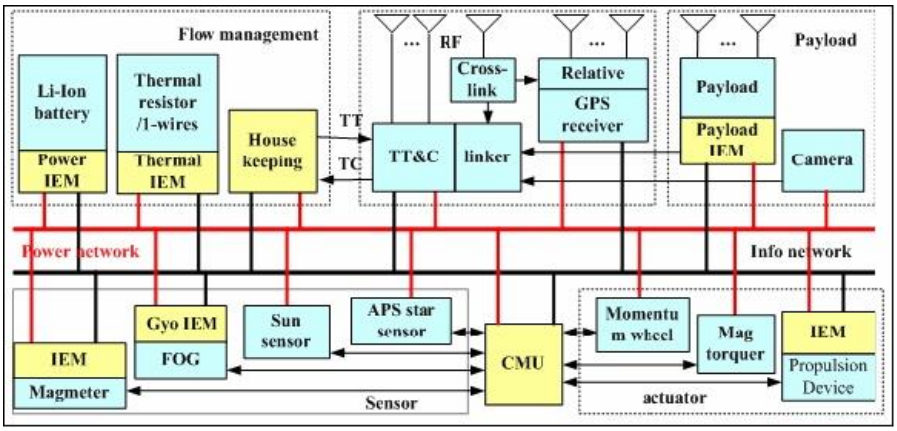


Figure 2.7: Overview of the HummerSat-1 SC onboard network architecture[42]

Figure 2.7 shows an example *mini satellite* architecture with multiple subsystems having housekeeping, all interconnected through a networked bus. The on-board network consists of a Controller Area Network (CAN) as a backbone of the satellite and is a dual-redundant bus. All spacecraft instruments share a common power bus and data bus. Furthermore, each module has its own characteristic housekeeping data, usually only a small percent of the actual payload instrument data, like shown in table 2.2 below, taken from the Imager for Magnetopause-to-Aurora Global Exploration (IMAGE) mission [60].

Combined instrument data rate	26 kb/sec
Housekeeping data rate	2 kb/sec
Total data rate	28 kb/sec

Table 2.2: Separated data rates of the EPS unit of the IMAGE Mission (1997) [60]

Data-rates for small satellites should never exceed 100kbps, but for (large) communication satellites, 10,000s of sensors could sum up towards Mbytes/second links at the bus-side. The sum of all data-rates for all sensors means that the bus will have to handle a couple of megabytes per seconds for many nodes. Aggregating sensors inside nodes will help to improve link-bandwidth (i.e. connecting multiple sensors with one wireless link), however aggregating sensors might not always be physically possible, due to (inherently) distributed locations of the physical sensors. An example is shown in figure 2.8 where the construction of the Large Area Telescope (LAT), part of the Gamma-ray Large Area Space Telescope and its housekeeping sensors are shown [67].

Figure 2.8: The GLAST project and its cabling harness [18]

Real-time housekeeping data

The typical data rates for housekeeping are in the order of tens of kbps[4], this housekeeping data is generally stored on-board and sent down to earth in batches. Sometimes housekeeping data can be send instantly to earth, as the instruments make their observations. Then, the notion of high priority housekeeping is common (see data flow observations, section 2.3.1), like real-time attitude information. This is also called real-time housekeeping data. In general, we cannot clearly distinguish between housekeeping data for a dedicated function, but this term is used for a general measure as each instrument has both housekeeping data and its typical instrument-data.

Locations of housekeeping sensors

Most of the HK sensors are located in the vicinity of active components from distinct subsystems of the satellite:

Internal: Inside active components, e.g. each power supply have temperature sensors inside for monitoring temperature of power cells, and pressure sensors near propulsion systems such as inside tanks or valves.

PCB: (Printed Circuit Board) sensors on-board temperature sensors on the circuit boards are ubiquitous, they are mostly already integrated (like microprocessors and FPGAs) and are "there" to be used directly.

External: Located on the surface of the spacecraft, e.g., antenna booms, under (deployable) solar panels, etc.

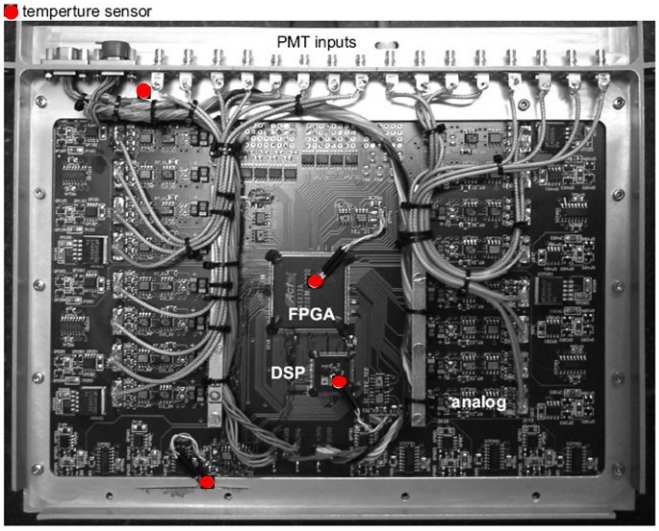


Figure 2.9: Inside of the PAMELA PCB-board, containing several AD590 temperature sensors spread over the components [33]

Current satellites use for example temperature sensors like AD590 (by Analog Devices Inc.) and are spread over vital components (such as FPGAs) or within the vicinity of important locations. An example is shown in figure 2.9, inside the PAMELA research board used in the Russian Resurs-DK1 satellite.

Sensor variations

The typical sensor data consist of temperature sensors, voltage/current sensors, 1-bit switches (e.g. deployment) and tell-tale switches.

We can conclude that the information the sensors transmit is most of the time low-quantities and single-valued information like:

- Physical phenomena readings
  - Temperatures
  - Voltages
  - Pressures
- Operational status bits
  - Functional operation status (e.g. sun sensor operation)
  - Deployment status (e.g. boom/array deployment)
  - Configuration status (e.g. redundancy branch status, which is in use)

2.4 Other Related Sensors and Applications

This section focuses on example existing sensor systems used in spacecrafts and some related (wireless) sensor systems that are used by NASA today.

2.4.1 Temperature Sensors

Temperature sensors are the most commonly used sensors for both testing and flight applications. Several types of temperature sensors have been examined and they are based on the level of application requirements, such as accuracy, repeatability, operating range and overall reliability.

The ISS forward technology solar cell experiment

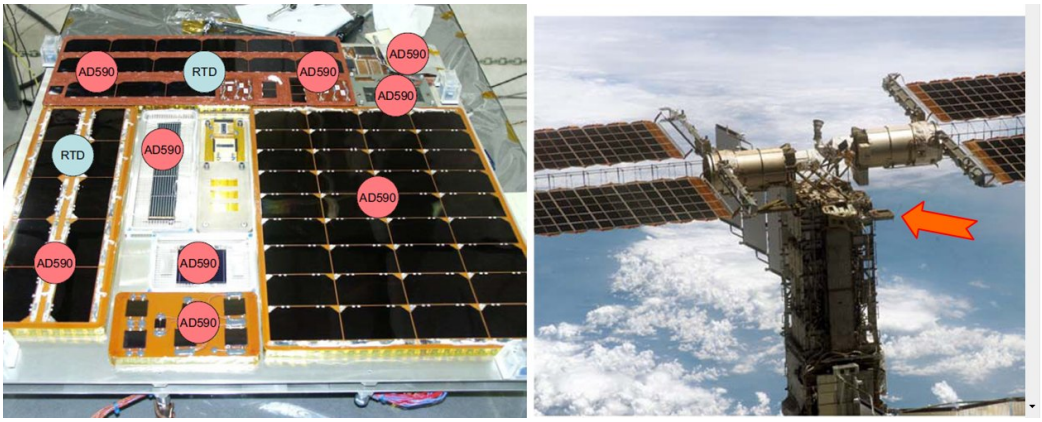


Figure 2.10: The ISS Forward Technology Solar Cell Experiment, using AD590 Temperature Sensors [82]

The AD590 (Analog Devices Inc.) is widely used in space systems, as well as other space-flight hardware like in the test setup for ISS (figure 2.10) [82]. It is a well-proven, space qualified and thoroughly tested thermocouple for accurate temperature read out. The AD590 is particularly used in many spaceflight missions, due to its capability of using long wire lengths with almost zero loss in signal. The AD590 requires a voltage source, and has a current output proportional to absolute temperature. Furthermore,

BIRD [49], ACE [15] and other spacecrafts invariably use AD590 temperature sensors for thermal housekeeping.

RTD Sensors

An Resistive Temperature Detector (RTD) is a temperature sensitive resistor. RTDs are mostly used for their wider range and higher accuracy than conventional thermocouple temperature sensors. As RTD sensors are becoming more and more available, support for RTD sensors might also be incorporated. A typical space-qualified wide range RTD sensor is the #29230 from RdF Corporation [13], which has a (wide) operating range of –200°C...+260°C.

2.4.2 NASA's Wing Leading Edge Impact Detection System

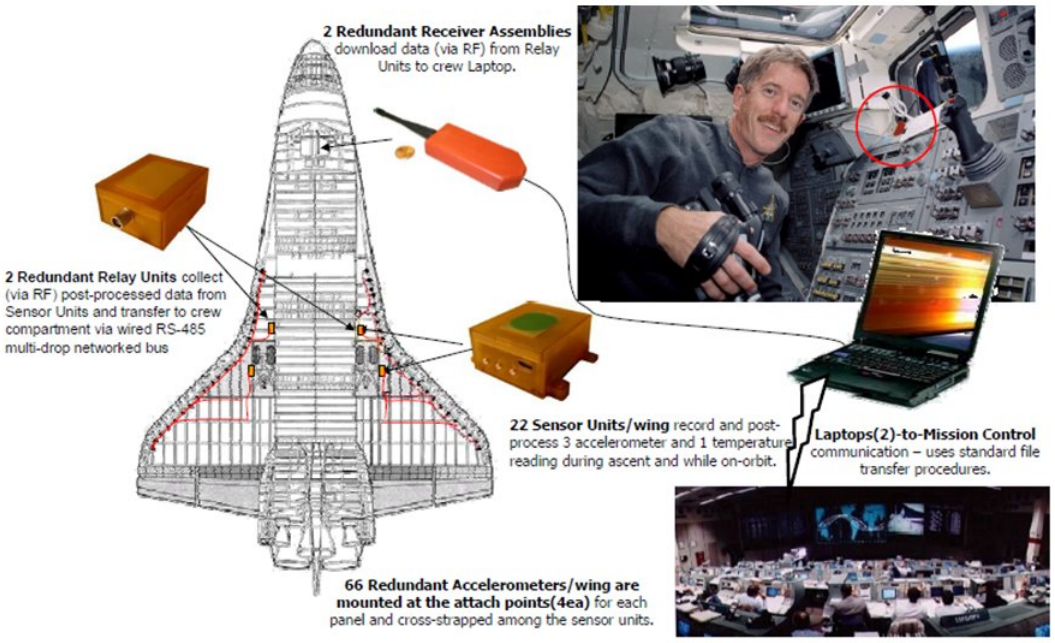


Figure 2.11: Wing-Leading Edge Impact Detection System (WLEIDS) [73]

An example use case of accelerometers is the Wing Leading Edge Impact Detection System (WLEIDS), depicted in figure 2.11 [73]. After the tragic loss of Columbia, this system was installed and detects any foam impact during ascent or micrometeorite impacts during orbit. In this scenario multiple accelerometers are used for impact detection of debris. Wireless was found as a flexible way to be used (during integration) for retrofitting the space shuttle with additional sensors.

The system consists of 132 single axis accelerometers mounted along the length the orbiter leading edge wings (see Figure 2.11). During launch, the accelerometers collect data at a rate of 20kHz and stores data onboard (through the use of wireless relay units) for subsequent downlink to Mission Control where data is analyzed by specialists.

Accelerometers can be considered as high data rate sensors. As these sensors also require additional components near the sensing elements like a Digital Signal Processor (DSP) and intermediate storage (logging), the design of a low-power sensor design is very different than the (low-rate) discrete sensors like thermistors and pressure sensors. Even after the signal processing, data rates for this type of sensors are quite high.

A common solution for high data rate sensors is to use a (large) logging memory, to store data and download it before each new acquisition. To conclude, the most viable solutions when using accelerometers is to use either high-speed RF links (point to point) or use a data logger, which requires manual intervention (by cabled receiving of data) afterwards.

2.5 Attitude Determination Sensors

Attitude determination is the technique where the orientation and location of the spacecraft or satellite in space is determined. The spacecraft has to be correctly oriented because of the solar panels have to be correctly pointed towards the sun, antenna's to earth and other sensors (e.g. scientific instruments on-board the satellite) have to correctly be oriented for proper operations.

2.5.1 Sun sensors

The sun sensor is one of the most common sensors used for attitude determination in spacecrafts. It senses if the sun is present on the sensor and digital sun sensors can provide the direction (i.e. angles) of the sun. Sun sensors could be read out using for a typical sensor network, because of the low sampling rates (2-10Hz), and low data bandwidth. A study of micro sun-sensors [57] revealed that bandwidth depends on complexity after processing sensor data. Without any processing, RAW images will likely to be in the order of numerous kilobytes to megabytes, per sample.

Functionality	**Sensor Bandwidth**	**Sensor Complexity**
Raw images	2 Mbyte/sec	Low
Bright Pixels	2.6 Kbyte/sec	Medium
Windows	2.0 Kbyte/sec	Medium
Centroids	128 byte/sec	High
Angles	40byte/sec	Very High

Table 2.3: JPL Sun sensor design complexity study (taken from [57])

An overview of a typical micro sun-sensor (JPL, [57]) functional complexity is shown in table 2.3. The JPL Micro Sun Sensor (MSS) has been developed and tested in a student nanosatellite, KUTESat-2 [72]. In order to support the sun-sensor, at least some level of processing has to be done, before wireless transmission is possible. The JPL Sun sensor is designed for 8Hz sampling rates, and will result in 2MByte data of RAW images that is reduced to 128 bytes/s centroids using an FPGA. The processing can

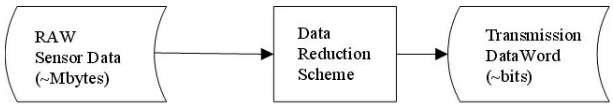


Figure 2.12: The use of data reduction techniques to reduce transmission bandwidth

be categorized as data-reduction, and must be performed at the sensor side, or directly at the processing side (e.g. FPGA, microcontroller or a microprocessor).

Another (demonstrator prototype) sun-sensor, the TNO Autonomous Wireless Sun Sensor [16], utilizes 60bps for transmissions (and 20bps for network-link information), and angles are transmitted as four 16-bit ADC values.

As for coarse Sun sensors and sun presence sensors, they will have lower datarates even and can certainly be candidates for integration in low power wireless sensor networks.

2.5.2 Magnetometers

The earth's magnetic field can also be used for to find out the orientation of the spacecraft. Magnetometers are typically deployed and sit on the outer rim of the spacecraft, free from any form of (magnetic) actuation provided by the satellite and provides the spacecraft with angle (direction vector) and magnitude of the earth. Magnetometer data is typically low rate and operates at low quantities of data (only a couple of bytes). Like sun-sensors, the magnetometer can be considered as a possible use-case for a wireless sensor network.

For these, and other wireless attitude determination sensors, the most important issues are high-reliability of the attitude sensor data, including a high availability at several cases. Another parameter, end-to-end delays of the transmission, determines the actual applicability of a magnetometer, as data must be delivered on time and cannot tolerate huge delays. Also, interaction patterns (i.e. the sensor re-sampling time) has to be coordinated together with the actuators, as actuators (e.g. a magnetorquer) have effect on sampling the magnetometer. One solution is to use a predefined scheduling and based on these transmission schedule, e.g. actuation can be performed when not sampling the magnetometer at the same time!

In the end, time-to-time failure (of an entire sample) might not be that stringent, but still has some limited effect, as when no sample is received, no actuation can be performed, so delaying actual satellite controls.

The wireless free-flying magnetometers

A successful (student) microsatellite project, is constructed by picosatellites wich are *wireless* magnetometers. These picosatellites (10×2cm) working as a remote magnetometer, see figure 2.13, where jettisoned from a mother ship, Orbiting Picosat Automatic Launcher (OPAL) [12]. The picosatellite transmits wirelessly (through FM bands) to the ground station.

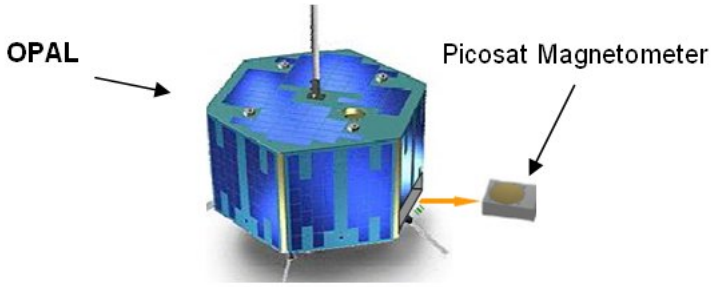


Figure 2.13: OPAL (Orbiting Picosatellite Automated Launcher) ejecting its magnetometer test bed

It consists of a test bed of hockey-puck sized free-flying magnetometers, independently capturing information about the magnetosphere. The picosatellites, in turn, time-stamp their data and transmit them back to the mothership as a collective set. This project indicates the possibilities of distributed sensing and demonstrated the use of mothership-daughtership communications utilizing tiny sensor nodes.

2.6 Sensor Candidates

For intra-spacecraft sensors, many sensors can be accommodated with wireless links (as seen in the previous sections), however the most probable sensors are temperature sensors and the sun-sensor. Other sensors are located on PCBs and can easily be connected through (short) wires. Sensors that cannot be wireless can be one of the current sensors and voltage sensors, as they might be in the circuit loop, and therefore there is no point of making these sensors wireless.

Sensor Type	**Temperature**	**Pressure**	**Strain**	**Tell-tale switches**
Typical # of sensors	100s	100s	100s	10s
Typical Data Rates	1–10Hz	1–10Hz	1–10Hz	Aperiodic
Typical Data Sizes	Tens of bits	Tens of bits	Tens of bits	Bits
Interaction pattern	Periodic	Periodic	Periodic	Aperiodic

Table 2.4: Typical Low Data-Rate Housekeeping Sensor Candidates

Other sensors are not generally housekeeping sensors by definition, examples are:

- *Tell-tale switches* (e.g. boom switches): they only provide low-level data (e.g. 1 bit information) at very low data rates and (single) event based.
- *Magnetometer sensors* – provide measurements of the earths magnetic field only when the magnetorquer (or interfering actuators) is not actuated
- *Sun sensors* – provides sun incidence angles only when the sun is present on the sensor

While the latter two sensors fall in a different reliability class as they are considered to be ADCS sensors, these sensors are still probable candidates for the system by their *low data rate* nature. Some sensors only send once needed (aperiodic) information out, and might be required for only once in a periodic event (like ascent or once during deployment). Monitoring messages may be a combination of periodic and/or event-driven. For example, temperature may be reported every 20 seconds on a regular schedule, but also reported immediately when a temperature threshold is exceeded. For other (non critical) sensors or during a test bench setup, sensors could be read out on request also (i.e. observer initiated, see section 2.2.1).

In the OBC, housekeeping parameters are characterized in software by code that can be reused at certain intervals. Also separated boards contain dedicated software routines that handle local-instrument based housekeeping data. The accelerometer is one of the

Sensor Type	**Magnetometer**	**Sun Sensor**
Location	Boom/ inside s/c	On s/c hull
Typical # of sensors	1–10	2–6
Typical Data Rates	10–20Hz	8–20Hz
Typical Data Sizes	Tens of bytes	Tens of bytes
Interaction pattern	Periodic (when required)	Periodic (when sun)

Table 2.5: Typical Low Data-Rate ADCS Sensor Candidates

high-data rate sensors (like other *modal testing*[2] sensors, such as specific strain sensors and pressure sensors), will not be addressed in this thesis as the hardware architecture will be profoundly different than for low-rate sensing only. Additional DSPs, memory and high-speed processing power is required for these kind of sensors. However, for strain sensors and (e.g. barometric) pressure sensors, there is a different case, as these sensors can be sampled both at low frequencies (order of Hz) and high frequencies (kHz to MHz), varying from application to application.

2.7 Modular Sensing

One of the goals of the wireless sensor network is the faster assembly, integration and testing at ground level. For this, a modular setup helps speeding up this AIT process. Utilizing the proposed reconfigurable and plug-and-play functionality in the platform, the sensing system and energy source can be upgraded or updated with minimal effort and errors. However, for some parts power consumption has to be considered, as well as "dead mass" during operation phase.

Introduction to plug-and-play interfacing

The presence of an electronic datasheet helps speeding up selecting and testing sensors on different sensor boards. IEEE 1451.(0-X) [29] provides means to interface sensors to

[2]Modal Testing is based on the estimation of a set of Frequency Response Functions relating the applied force and the corresponding response at several points along the (spacecraft) structure.

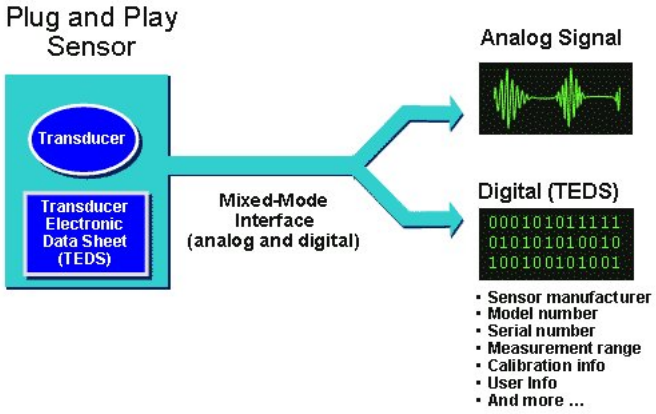


Figure 2.14: Basic TEDS setup, conform IEEE1451 [32]

networks with the goal of achieving plug-and-play (PnP) and interoperability (see figure 2.14). IEEE 1451 consists of two main components, the Network Capable Application Processor (NCAP) and the Transducer Interface Module (TIM), and an additional component, the Transducer Electronic Datasheets (TEDS) that describes the interaction. Figure 2.15 illustrates the general overview of a IEEE 1451 system and external interfaces.
To summarize, the TEDS have many benefits:

- Enable self-identification – no need for manual configuration/ or manual storage of calibration information or ID tagging;
- Ease field installation, upgrade, and maintenance of sensors – this helps to reduce life cycle costs because a less skilled person is required to perform the operations which results in less labour-intensive activities;
- Reduces human error: automatic transfer of TEDS data to the network or system eliminates the manual entering of sensor parameters which could induce errors due to various conditions.

Drawbacks are however:

- Additional power consumption when active/reconfiguring: the power will go up when the system retrieves or programs the TEDS onto the board;
- Additional ROM/RAM (Memories) are required, which have to be space-qualified;
- Additional design complexity: additional wiring is required in the sensor node and commands have to be implemented;
- Dead mass — when the system is operating, manual reconfiguration is no longer possible and TEDS looses its functionality.

To mitigate these problems, an alternative (memoryless) solution might be to use a hardware-coded ID on the modular sensor node and leave the TEDS on the microcontroller. Then, the unique hardware ID correlates with the on-chip (e.g. flash) memory, and the need for additional (external) ROM is removed.

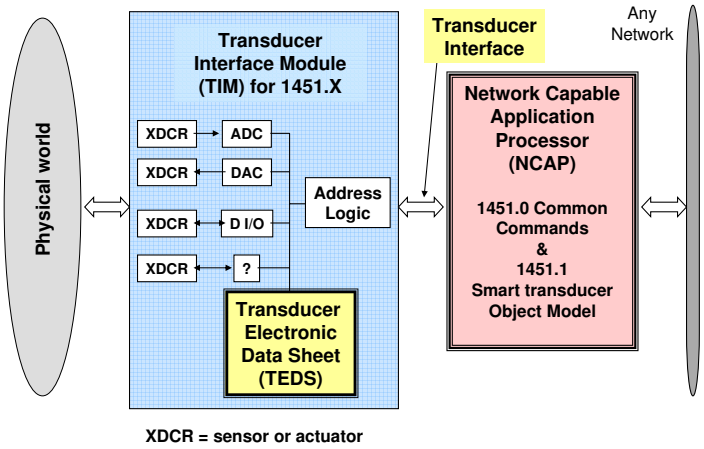


Figure 2.15: IEEE 1451 Smart Transducers Interface System Diagram

TEDS locations

The identification and data formats need to be known by the sensor node microcontroller module, however since multiple sensors are used, each with multiple data formats, TEDS need to be located somewhere in the system. Additionally, a TEDS pool (or TEDS Cache[3]) needs to be placed at the sink node required for the upper layers. The problem is that TEDS readout consumes unnecessary power, that might be used for the initialization operation only, as TEDS is solely used in the setup phase and configuration will be static afterwards. Therefore, 3 options are available to locate the sensory specific data (TEDS) onto:

1. Sensor EDS at the sensor node;
2. Sensor EDS at the processing module (e.g. microcontroller);
3. Sensor EDS at the central network node (e.g. sink node, gateway or access point).

As the number of nodes becomes really high, like temperature sensors (10s to 100s), it should be more useful to have simplistic hardware in nodes and put all complexity in the sink node. However, locating TEDS inside the sink implies an "*over-the-air*" transmission of TEDS, which costs power and bandwith of the network.

It is difficult to select a location of sensor data, because 1 specifies more hardware (multiple EEPROMS) in sensors (there are always more sensors than gateways), and 2 specifies a single, centralized location for a single hardware, and keeps single sensor-hardware more simplistic. Downside is more software (and communication) overhead during the setup phase, however once TEDS is fully downloaded in sensor modules, reconfiguration is not likely to be done again within a long period of time. The next section will continue on the discussion about modularity, and current implementations of modular spacecraft interfaces.

[3]in IEEE TEDS, the TEDS Cache is typically located in the NCAP, see [8] for details.

2.8 Interfacing

This section forms a basic study on the interfacing of the intra-spacecraft wireless sensor network with its spacecraft other instruments, located on the spacecrafts bus.

2.8.1 Current spacecraft interfacing taxonomy

Today, satellites and spacecrafts are constructed using custom based interfaces, and sensor interfaces are typically developed by the sensor's custom interfacing part. I.e. traditional systems engineering is component centric, relying upon a detailed component interface control document (ICD) to enable a system configuration (figure 2.16).

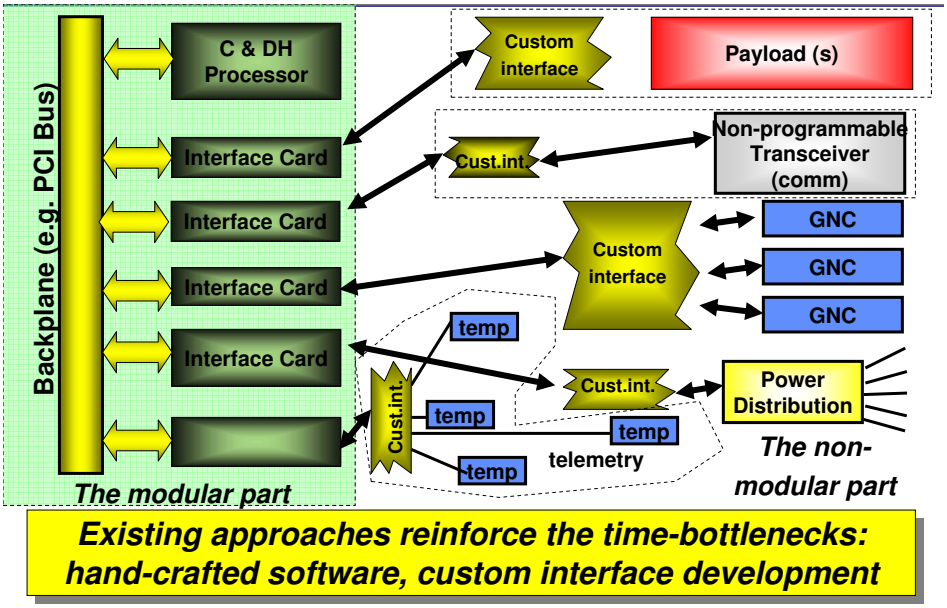


Figure 2.16: Current Spacecraft Interfaces Taxonomy

2.8.2 Universal interfacing

Existing research and applications focus on a universal interface module; the notion of a "Remote Terminal Unit" is common in spacecraft systems. A definition in spacecraft and avionics is explained in the quote below:

An RTU acts as intermediate and as a fixed interface component between the Spacecraft's bus and Sensor Appliqué. Figure 2.17 shows an example RTU, the ÅAC Aerospace Remote Terminal Unit (RTU-100-CS), developed in Sweden [1]. The developer also states that this device can act as a general sensor/actuator bridge (SpaceWire, CAN, I2C, SPI) and serve as an OBC for nanosatellites and microsatellites. In this way, hardware or sensors can be connected to the common spacecrafts bus.

"A Remote Terminal Unit (RTU) is a device that allows fast and easy integration of payloads, sensors and subsystems on complex systems like spacecraft and unmanned aerial vehicles (UAV). Other common names for RTUs are Remote Data Concentrator (RDC) in aviation and Appliqué Sensor Interface Module (ASIM) in the US. RTUs or ASIMs are vital elements of the U.S. AFRL derived Space Plug and Play Avionics (SPA) standard." (Source: ÅAC Microspace, Sweden [1])

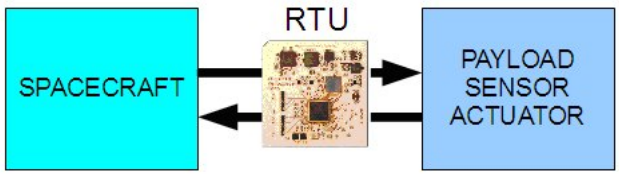


Figure 2.17: ÅAC Aerospace RTU-100-CS Remote Terminal Unit [1]

Aggregated sensors inside spacecraft instruments

The main setup of each spacecraft is that each instrument (e.g. Payload, Radio, ADCS) is connected separately to the spacecraft bus, the sensors are not directly separated from the bus. The sensors are mostly connected to its corresponding instrument system. In current research, a universal modular approach is chosen for each instrument only. Figure 2.18 illustrates the Space Plug-and-play Avionics (SPA) concept, and incorporates of one Appliqué Sensor Interface Module (ASIM) per instrument. It is envisioned that an ASIM can be constructed using a single chip microcontroller (figure 2.19).

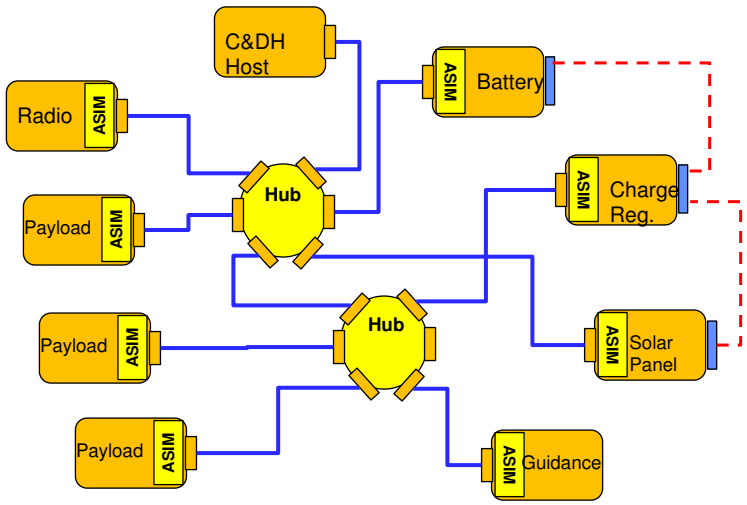


Figure 2.18: Individual Instrument Modularity, the SPA SDM (Satellite Development Model) supporting self-organizing networks [50]

SPA uses a run-time parsable ICD that has been labeled as an extended trans-

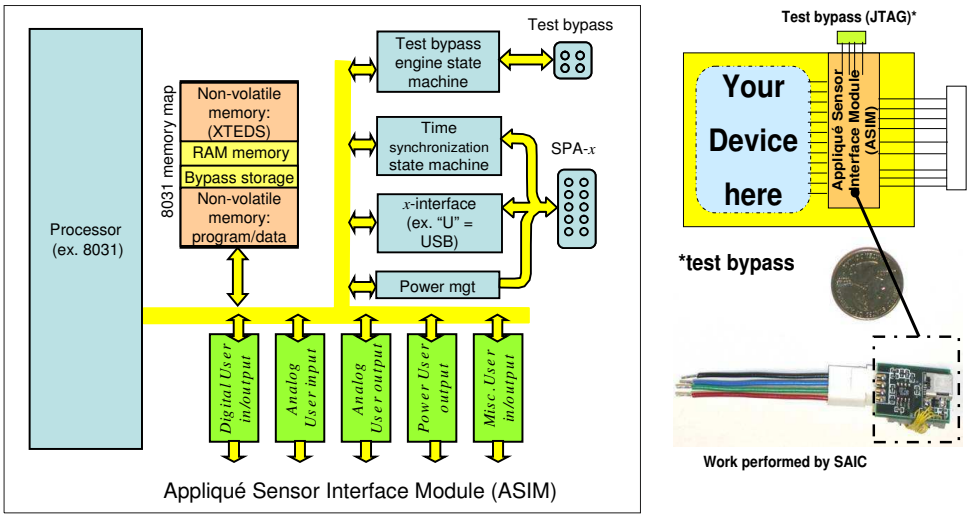


Figure 2.19: An example one-microcontroller ASIM interface, work done by SAIC [50]

ducer electronics data sheet (XTEDS) for each ASIM. These "self-describing" components can be used to automatically construct networks dynamically. After connection to the "Hubs" in the network, the sensor manager (displayed in figure 2.20) handles I/O dataflow and detects if new devices are being connected.

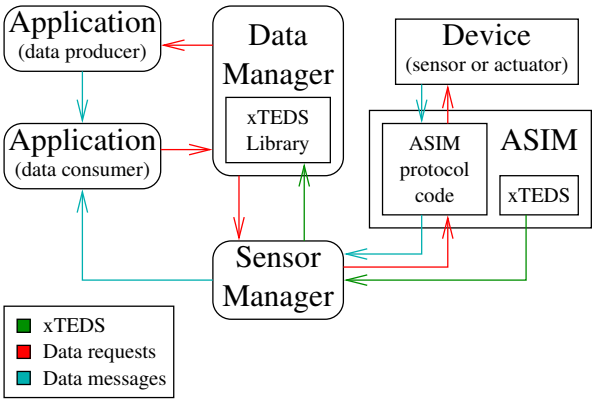


Figure 2.20: Dataflow of the SDM using SPA-compliant devices

The goal of a wireless sensor network will require decoupling of this ASIM model even more. However, there are quite a lot of commonalities between the components of the SPA system model and the wireless sensor network. To list some examples:

- Sensor Manager: this manager is already present in wireless by default, as wireless is interoperable and plug and play by definition in the wirelessly way (i.e. without 'plugs');
- ASIM: the actual interface between sensor nodes and spacecraft bus systems,

specifying a universal protocol.

In the wireless system these components are *de facto* present and act as a gateway or bridge between the wireless network.

Beacuse a lot of effort already exists in the wired domain for PnP devices, the extension of PnP to the wireless sensor network is considered as feasible, but has limitations due to overhead and additional complexity. Additionally, the gained fault tolerance is the common by-product of PnP functionality, such as:

- Sensor network commissioning: Wireless sensors have to annouce its presence by joining to the network and noticing leaving of the network (*device enumeration*);
- Discovery of other networks: Adapt to changes in the network by selectively connecting to another network.

Current work by AFRL

A lot of effort already is spent into the SPA concept and its components. Recently a joint venture of the U.S. Air Force Research Laboratory (AFRL) and Sweden focuses on the SPA-1 concept which is based on I2C. Under consideration already are the xTEDS drafts in general: currently ESA is investigating all the possible electronic datasheets and ESA advocates to harmonize and decide the best way of storing EDS in any form. A current research project includes the design of a wireless (SPA compliant) modular data bus by Northrop Grumman [41] based on the existing SPA draft standards.

2.8.3 IEEE 1451 extensions

A more detailed and universal accepted standard that is related and worth noting is the use of the IEEE 1451 interfaces, for smart transducers (Figure 2.21). Smart transducers are actually capable of sending out their sensed data in a packetized way, and including some sort of higher-level interface. The protocol specifies not only data formatting (TEDS), but also a higher level network-independent communication interface. In principle, the requirement for a modular exchangeable sensor interface also requires a way of sensor identification and correct readout using its datasheet. The 1451.0 standard is a (possible) candidate for a possible universal transducer interface, however certain functionality might be overkill (such as service discovery and some streaming data modes).

Another possibility is the adoption of IEEE 1451.5 (the wireless version of the standard transducer interface) [71]. This certified standard (in 2007) has included several known wireless network physical (PHY) layer and Medium Access Control (MAC) standards like Zigbee and Bluetooth.

Key problems on the IEEE1451.5 is that the standard is not be customizable enough to fit the application into it, such as low power consumption and long operating lifetime. Also the inherent complexity of the IEEE 1451 standards bottlenecks possible implementations. The tendency to develop a custom sensor interface is highly supported by

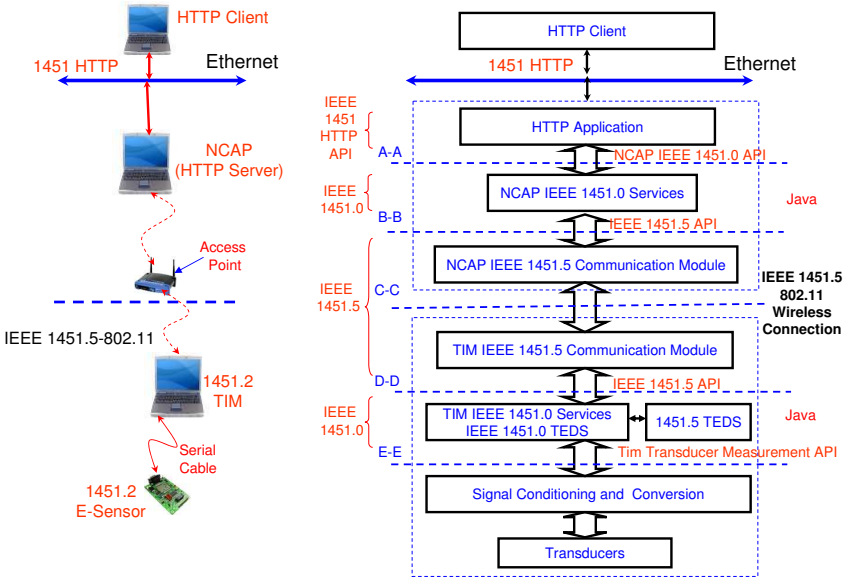


Figure 2.21: An application for a wireless self-describing web based sensor, implemented using the IEEE1451.5 protocol [37]

the fact that the system might have too much functionality not neccesary during the spacecraft mission, as it is a one-time configurable sensor system in the sense that it can only be one-time hardware configured and supposed that it will not be not changed during the mission. Moreover, the IEEE 1451 standards seems to be focused on ground-based (test) equipment, where power is in abundance, and manual power replenishment for sensor nodes is always a possibility.

The use of standard formatting over a standardized protocol is an attractive idea. The TEDS system is already successfully used in many Honeywell systems, for the last 8 years [30]. The idea of plug-n-play sensor modularity is well used in the concept and many benefits are worth paying attention to like the advantage of having self-documenting hard– and software and the utilization of existing wireless PHY interfaces.

2.8.4 Energy source interfacing

Interfacing the sensor nodes energy source is another issue. Several energy sources today exist (primary batteries, rechargeables, solar cells, etc.) and are selectable for reliable operation during the lifetime of a sensor node.

Wired power vs. batteries

A case study funded by NASA[6] shows that *wired power* in wireless sensor nodes might be more effective than batteries (and other power harvesting devices), as these energy sources itself also carry additional mass. The study was to implement a wireless Thermal Protection System (TPS) and was actually developed as a prototype to reduce harness mass. Also, in some cases wireless data with wired power can have other advantages

such as improved operating reliability and support for power-bursts, required for bursty– and more high-speed transmissions. Many typical applications for wireless sensors in a spacecraft are found at inaccessible locations where batteries are not a good solution and also temperature extremes (within some areas of the spacecraft) often exclude the use of batteries.

Energy harvesting

Energy harvesting is the use of thermal properties, solar irradiance or other harvesting methods to capture and store energy. Typically, a secondary power source (i.e. rechargeable) is used for storing this energy.

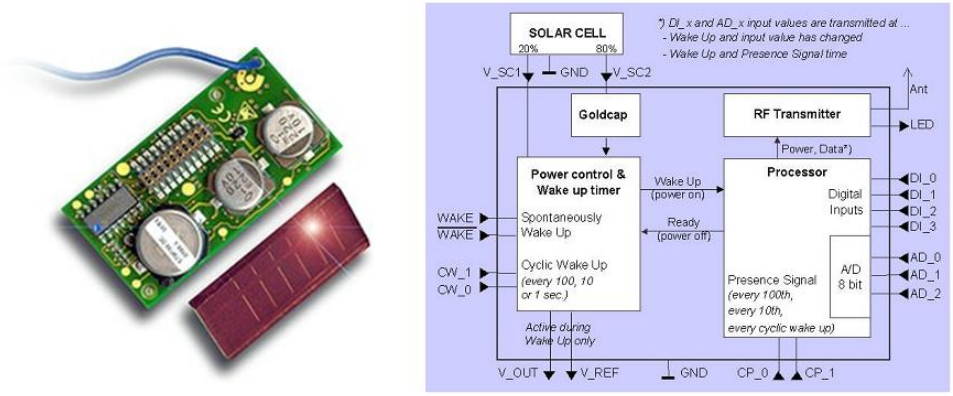


Figure 2.22: The EnOcean Radio Sensor powered by a mini solar cell, and its block diagram [21]

Examples solutions which utilize power harvesting properties are already beginning to start on the consumer market. An example commercial solution is the STM 100 [21], figure 2.22. This solution incorporates a miniature device like a sensor mote (a 'remote' node), and operates with 3V at only 500μW. The modules can be utilized to monitor temperatures in buildings for purposes of efficient air conditioning and harvest their power from the 50Hz light bulbs in the surrounded space. A lifetime of 10 years is also a property of the radio nodes. In Figure 2.22, at the right side the circuit components of the solar powered transmitter module STM 100 is shown. An analog timer circuitry completely deactivates all components during the sleep phase of the node. This timer draws only a current of about 20nA. In short, interfacing from and to the power module is an important way to enable power management. In-fact, a separate timer or control module for powering nodes is not uncommon. In the Waspmote[45], a ready-available sensor node platform, a Real-Time Clock (RTC) can be used to disconnect the mote from its main battery, decreasing current consumption from 62μA (deep sleep) to 0,7μA (hibernate mode). In this way the node uses these real-time clocks for event-based applications, e.g. to wakeup every 1.5 hours or wakeup once a day.

Super capacitors

The STM 100 is powered by an additional high-capacity capacitor, sometimes called a "gold cap" or "super cap", as seen in the figure 2.22. The main advantage of this supercapacitor is the wide operating ranges compared to (chemical) batteries (the super caps allow for improved survivability). Thus, at first glance the super caps seems to have most suitable characteristics to fit into a spacecraft, however not many mature solutions exist today, and missing algorithms bottleneck quick development and a valid proof-of-concepts over long durations of time. The (charge/discharge) characteristics of gold-caps are not the same as battery-based power sources [56], so practical applications super caps have not yet demonstrated their full potential yet, particularly for wireless sensor nodes.

Energy awareness

Energy awareness for the sensor node is key to enable energy conservation. Because the energy sources could be modular like transducers, a form of Energy Electronic Data Sheet (EEDS) [83] could be applied to see what kind of power is currently connected and its accompanied characteristics. A one-wire (1-wire) communication line with a 1kbyte EPROM connected inside the power module might be a solution, however 1-wire consumes current, even when no communication is on the line. For the sensor node, sensing and possibly the extrapolation of the remaining energy, battery voltage and harvesting delays could be useful to incorporate for more reliable and prolonged operation.

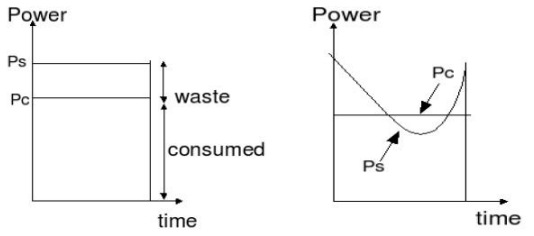
Time-to-time failure might be permitted in certain cases (housekeeping data), as long as there is no continuous system failure, so the energy source should be able to be self-replenishing and able to signal this status and control information to the sensor node controller. Another solution incorporates the energy replenishment rate into the cost metric when computing the routes [46].

Energy harvesting operation

When considering energy harvesting, different schemes for operation are presented, depending on lifetime and energy available during harvesting. Although energy harvesting has been successfully applied in some applications, the technology used in combination with wireless sensors is still is not quite mature [69]. Thus, powering and replenishing wireless nodes remains a major issue. On board power management should be required for each element in the sensor node, as this will increase sensor operation life [59]. This includes software, RF devices and other hardware connected to the node.

As of today, solar cells typically provide the best possible power with minimum wasted mass and area for a wireless sensor. Other harvesting systems (such as thermal scavenging) typically are heavy and bulky [69]. In combination with a (re-chargeable) battery, long lifetime of the sensor could be achieved. Two possible energy management modes for this scenario are illustrated in figure 2.23.

Possible modes could include:



(a) Harvest-Use (b) Harvest-Store-Use

Figure 2.23: Example energy-harvesting operational modes [74]

- Harvest and use directly: the power harvested is directly (and continuously) used by the sensor node. If the harvested energy is more than that consumed by the node, it simply gets wasted. On the other hand if harvested energy is less than required, the node does not operate;
- Harvest-store-use systems are harvesting energy whenever possible and stored for future use.

2.8.5 The wireless gateway

Gateway, or sink nodes are often intrinsic parts of sensor networks and are required to bridge data between sensor networks and servers/devices in other networks. Without gateway nodes, data collected by a sensor network are only local to that sensor network and therefore no remote monitoring or interactions can be achieved. A separate gateway

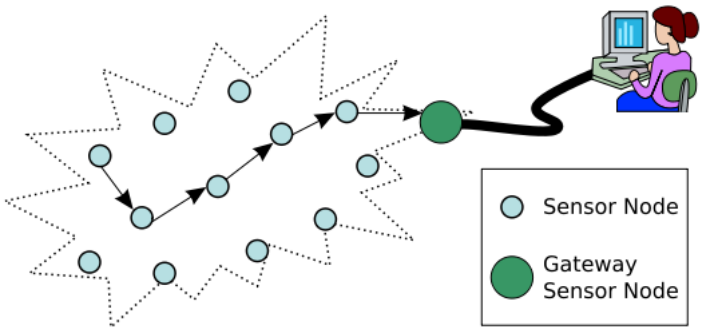


Figure 2.24: A Wireless Sensor Gateway Node [Wikipedia]

node might be considered as one node is the bridge between the spacecraft's bus, and another node might be the interface for remote configuration and testing.

Gateway functionality

In practice, the gateway node essentially is a wireless node, with more dedicated hardware resources and centralized functionality inside, like data storage and communication forwarding/bridging commands. For the application, data centric exchange and identity centric exchange are key parts of the network. Another note is that this functionality

also depends on the level of intelligence of the network, i.e. are some nodes considered as routers, for hop-to-hop data flows? A router-node on the other hand can have its own sensor integrated, as the wireless gateway node only listens packets send out by other nodes, and performs protocol conversion when necessary. The gateway node never relays packets.

To conclude, the gateway node essentially consists of the basic RF functionality and processing capabilities without the sensor parts. The gateway node is an anchor-node which acts like a general sink of the data from the sensor nodes.

Gateway recommendations

Some recommendations for a wireless system can be seen back from observations of current wired sensor systems, and will have implications on the design phase of the wireless sensor network:

- A wireless gateway node is required for universal interfacing with the spacecrafts bus, or ground equipment during AIT phase;
- Gateway node will not relay any (wireless) packets, it only listens to the other nodes;
- Positioning of the gateway node must be chosen such that all (necessary) nodes can communicate correctly with the gateway node;
- Protocol conversion rules need to be defined, in order for the sensor-network to cope with another bus, e.g. IEEE MIL-STD-1553, CAN, I2C, Advanced Full Duplex Switched Ethernet (AFDX[4]), etc.;
- Gateway nodes, or access points are often the components in the network who control and delegate commands to the sensor nodes, if faults at the gateway occur, it has profound impacts on the entire network operation.

2.8.6 Summary and conclusions

Individual sensor interfacing is a recognized problem in both the transducer and spacecraft community. Various universal solutions today have been proposed to have a generic plug-and-play transducer interface. The main solutions mentioned here basically form a viewpoint to make tradeoff studies and to perform tests with. As for the best suitable interface, a low level and simple interface is best suited, like I2C. While IEEE 1451 is standardized for (modular) transducer systems, the protocol is considered fairly complex and might be overkill in terms of functionality. Limitations such as the need for a compiler (because of non-readability and binary storage) are the main drawbacks for the IEEE 1451 transducers standards. Furthermore, the (current) SPA draft standard is currently only focusing on only a limited number of devices (dedicated flight instruments and common buses) and might not be suitable for the many (low-data rate) housekeeping sensors. The goal of this thesis, however is focused on the many sensors in the system, and focuses on the first logical step: low-data rate sensors like housekeeping, and integration-test sensors.

[4]AFDX is used in multiple aircraft systems, and was considered as the (modular) network for the Orion Crew Exploration Vehicle (in 2004), the successor of the spaceshuttle [58]

Source/system	IEEE 1451.0	IEEE 1451.5	xTEDS	EEDS [83]
Purpose EDS	Transducers	Wireless transducers	Spacecraft instruments	Energy sources
Stage	Standardized	Standardized	Draft standard	Research
Data Format	Binary	Binary	XML	Binary
Wireless	No	Yes – Wi-Fi – Bluetooth – Zigbee	Possibly	Possibly
Extendability	Limited	Limited	Full	N/A

Table 2.6: Comparison of existing work on electronic datasheets

As for the sensor nodes' power source interfacing, various signals have to be included in the interface, in order to have full advantage of a exchangeable power-system, and possibly additional signals allow the sensor-node to have some form of power-awareness and control of its power source. To have a basic overview of the reviewed standards or sources that support electronic datasheets, table 2.6 is setup to compare the differences for each EDS device.

The interfacing of the sensor nodes to the spacecraft is another essential part of the network. Various solutions are available to have both an interface and a test-interface for connecting the network to the end application. As the sensor network operates wirelessly, various components might be selected on incorporation of sleep-modes or other low-power functionality that can be utilized to save power.

A wireless gateway is the most prominent solution as this functionality addresses most of the systems needs and also might double as a test-node during AIT.

2.9 Dependability and Fault-Tolerance

In a spacecraft or satellite, the computing systems that control the spacecraft must be reliable. This property is most important for vital components in the spacecraft for correct operation. As an example, the (wireless) sun sensor should have an acceptable low failure rate in order to correctly determine the spacecraft orientation.

Failures and faults

A failure is an event inside the system that causes unwanted behaviour or degradation in performance. The fault is considered as an event and should be detected to determine if an action should be taken. Furthermore, it should be determined what action should be taken and what probably caused the fault. Usually, due to the long operation lifetime in space of the systems without maintenance, aging is one of the fault causes. Other faults are appearing immediately during launch or after launch, because of the excessive forces during this phase of the mission. That is why the most vital components of the satellite are equipped with redundant parts.

The failure appearance has been empirically observed and is related to the quality and durability of the component (e.g. materials [65], software, etc.) used [78]. It is observed that well treated and prepared parts show a stable performance until the actual material is weared-off to a certain degree by the harsh environment in space. Radiation is the major source of interference typically seen in space missions. Radiation caused faults typically result in Single-Event Upsets (SEUs). Radiation hardening (by process or by design) is one way for components to operate without faults the space environment [7].

Fault tolerant design

There exist three methods to improve or maintain a systems normal performance where the system is subjected to failures from the environment. These can be classified as fault avoidance, fault masking and fault tolerance.

Fault avoidance techniques are used to prevent the occurrence of bugs in the first place. Several techniques include wide scale testing and making design reviews are examples of fault avoidance in the first place. *Fault masking* is the process that prevents bugs in a system from introducing errors into the informational structure of that system. The aim of *fault tolerance* is to make applications resilient to various faults and crashes of hardware or software.

In all fault tolerance approaches the common used methodology is the use of redundancy. Redundancy is simply the addition of information or resources (e.g. additional execution time) beyond what is needed for normal system operation. Fault-tolerance could be achieved in several layers of a system. It may be introduced at hardware level, at software level or at data level. A subset of these techniques used are outlined in the following sub-sections.

2.9.1 Hardware fault tolerance

Spacecraft systems must function with very high availability even under hardware fault conditions. This section covers several techniques that are used to minimize the impact of hardware faults. In hardware fault tolerance, the hardware itself is responsible of providing fault tolerance in the system.

Topological fault tolerance

Fault tolerance can also be addressed as system-level approach. For example by decomposing the system into a certain sub-systems and therefore it can be designed to reliably operate as a whole better. Most wireless sensor networks tend to include spatial redundancy as a way to encompass physical damage in a certain zone. Care should be taken to not propagate faults into the network and still be able to communicate to a centralized computer or remote terminal unit. Therefore it is best to physically separate redundant network components to prevent outages due to loss of electrical power and other physical damage caused by the harsh environment in space.

Another way of incorporating redundancy is the use of electronically cross strapping sensors, computers or other hardware. Figure 2.25 illustrates an example of a fully

cross-strapped architecture for guidance, navigation and control [17]. In this topology the output of each component is physically connected to the input of each immediate element in the control loop. In this way every sensor output is physically connected to every computer, and every computer is connected to every actuator.

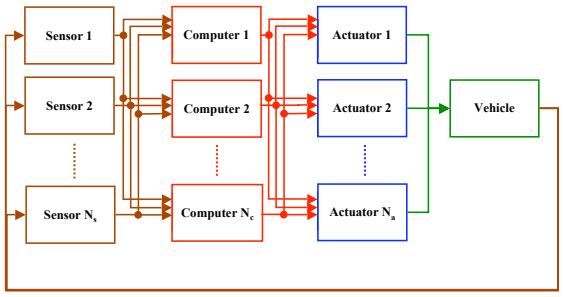


Figure 2.25: A cross strapped architecture for guidance, navigation and control [17]

Cold spares

In a certain topology, cold spares can be used for a fault-tolerant system in preference to hot spares. In cold (standby) spare only one copy of each component is active at a certain time. The active component should be designed to be stopped after failure detection (i.e. if the active hardware has failed the system will switch off and replace with a spare hardware unit).

As an example, the Large Area Telescope (LAT) subsystems incorporates the use of cold spares, which are essentially duplicate systems. The LAT housekeeping unit (called the Spacecraft Interface Unit, SIU for short) have a designated cold spare unit. The SIU is responsible for both monitoring (it is a housekeeping master unit) and control of the telescope [75]. Figure 2.26 illustrates the use of cold spare parts inside the LAT data acquisition system. On top the redundant SIUs are shown and three Event Processor Units (EPUs) are shown (one is cold spare) and used for processing and filtering data from the telescope.

Hardware diversity

As discussed before, topologies may incorporate redundancy in the form of multiple copies of the same hardware unit. Diversity can be introduced by employing hardware units from multiple manufacturers in a topology. Hardware produced by independent manufacturers can be effective at detecting and tolerating physical faults. By using voting between independent hardware faults can be compared

2.9.2 Wireless network fault-tolerance

For wireless networks, reliability can be defined as correctly receiving every transmitted message, every time during operation. The addition of basic network functionality already makes the system more reliable in many ways. Robust link communications

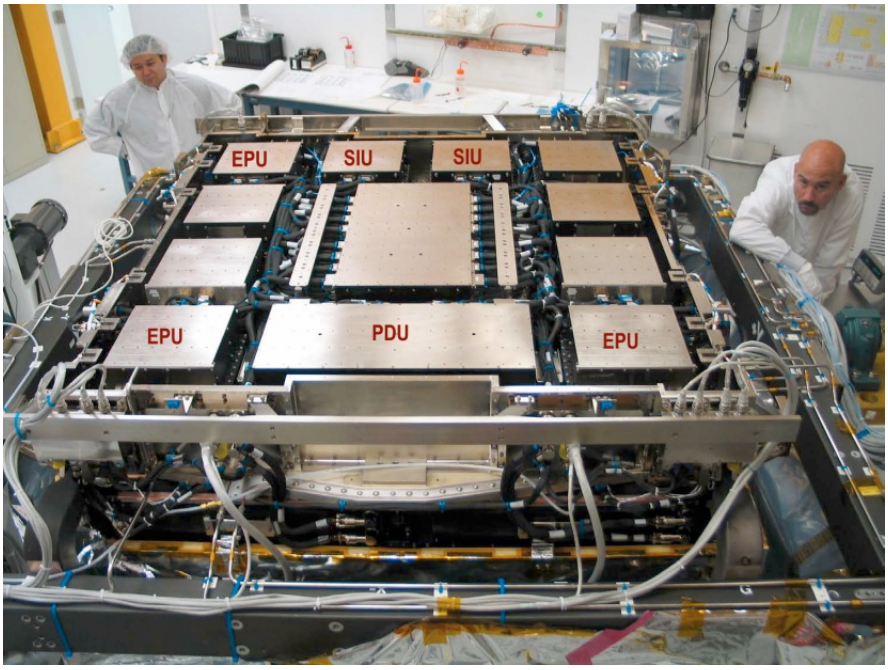


Figure 2.26: The LAT data acquisition system electronics and cold spares [62]

schemes ensure high reliability (e.g. frequency hopping, and other adaptability to interfering communications).

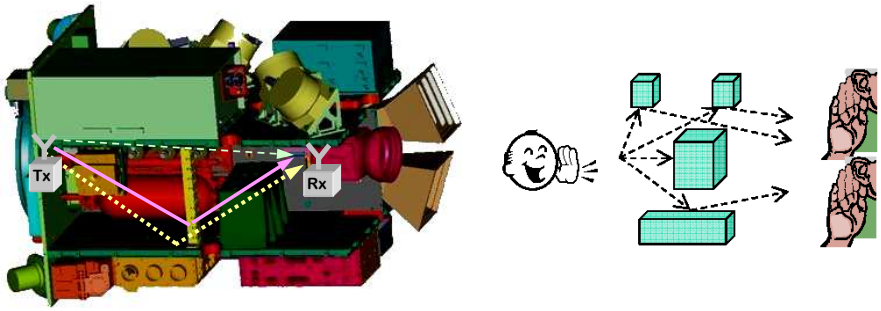


Figure 2.27: Mutipath effects inside a typical spacecraft

The number of retries is another issue for reliable wireless transmissions. Eventually, the lack of an acknowledgement back will trigger a retry. Collisions may occur, multiple signals can be send back reflected (i.e. by multipath effects) or the receiving end can be temporary out of range. Multi-path effects are caused by signal reflections of an object and the receiver sees the incoming signal a little later. This is known as multipath delay. The delay spread is different for each material the signal bounces on. Figure 2.27 explains this phenomena for a typical spacecraft with its instruments and walls as obstructions.

The range of the networking devices is also another point that determines the reliability of the devices. When transmitting on a wider range less retries are required, and

even less communication errors can be seen, resulting in an overall more stable wireless link.
Solutions to increase reliability as discussed above will have significant impact on the power consumption of the wireless nodes and should *all* be optimized for wireless mobile communications.

3 The Wireless Platform

Based on the initial strategies and conclusions listed in the previous chapter a choice of the required wireless hardware has to be made. This chapter deals with the survey and selection of the particular wireless hardware and protocols to conduct our experiments. The selected hardware platform will also act as a testbed that allows for rapid development of applications and middleware.

Goals and targets are first discussed in section 3.1. A review of current (standards-based) COTS wireless protocols and transceiver systems is given in section 3.2. Included is the result of a survey of standards-based low-data rate wireless network architectures and the IEEE 802.15.4 wireless standard is described and reviewed in section 3.2.2. A discussion on benefits and drawbacks of general network topologies is presented in section 3.3. Lastly, the hardware platform selection is described in section 3.4.

3.1 Goals and requirements

The end goal of this thesis is to have a general architecture (engineering model) and fault-tolerant platform that can be used during AIT operations. This platform may have a progressive introduction into on-board systems, giving initially the preference to support low criticality sensors such as (engineering) housekeeping monitoring.

The following goals where identified and decided on:

Sensor selection: Regular 'low-rate' sensors such as temperature, humidity, pressure strain, etc.

Sampling rate: Maximum (transmission) sampling rate of 20Hz for ADCS sensors and 10Hz at most for non-critical housekeeping sensors.

Modularity: The sensor node hardware has been chosen such that it consists of separate modules: a power module, core module and a sensor module. The sensors will directly interface with the core module via an Universal Transducer Interface (UTI).

Criticality: Non-essential systems, such as engineering housekeeping monitoring.

Failure rate: Housekeeping sensors allow for time-to-time failure, as long as it is not systematic.

Node Powering: Primary battery power (non-rechargeable) is selected as one source, and secondly, a combination of secondary batteries with solar cells as an in-orbit solution. A third (modular) source could be wire powered.

During the first phases of the thesis, the accelerometer was ruled out the scope of this thesis, as it required high-datarates (when used realtime) and eventually digital-signal processing capability. Therefore, the goal was to use low data bandwidth for sensors also.

The first steps will be hands-on exploration with common commercially available components, that will result in flight-suitable components and subsystems.

3.2 Radio Transceiver Systems

This section forms a basic study about the RF transceiver aspect of the intra-spacecraft WSN. As the transceiver part of any wireless sensor network is the main part of the node and considered to be the most power-hungry part of the node. Special requirements from the application side, e.g. a lifespan of at least 10 years of operation and qualification for (maintenance free) harsh conditions, give directions of power requirements for the sensor nodes. Currently, many wireless sensor network systems are based on extreme low-duty cycles (since power is the biggest issue for all wireless devices).

3.2.1 Standards-Based RF Radios Overview

Today, many standards are available for low-data rate and low-power networks. Our focus will be short-distance (because spacecraft dimensions are typically in the order of few meters) and low bit rates.

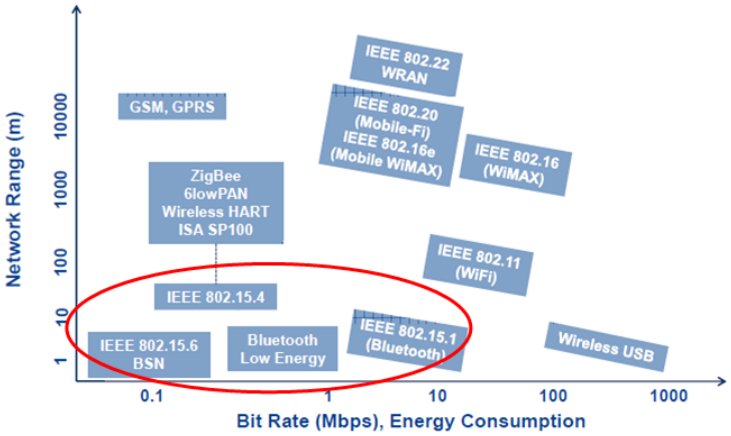


Figure 3.1: Standards based wireless networks and network range

In Figure 3.1, the circle highlights the applicable RF standards that can be used, with primary focus on low-data rates and close proximity communication. For the three highlighted standards (IEEE 802.15.6, 802.15.1 and *Bluetooth Low Energy*), the most mature (and widely accepted standard) is 802.15.4. The *Body Sensor Networks* (BSN) and *Bluetooth Low Energy* specifications are not yet certified. Bluetooth (802.15.1) is considered for medium data rate applications, such as file transfers or as a wireless bus replacement for the spacecraft, such as e.g. the CAN over Bluetooth system from [53].

Sensor Data Size

IEEE 802.15.4 already has a low data packet (payload) size, in the order of 1 to 114 bytes. Other wireless systems, such as WiFi and bluetooth offer higher datarates (e.g., up to a couple of megabytes) but require high power consumption. As an example, thermal information will be typically digitized by 10-12 bits, for a typical spacecraft [23].

Coverage

A large coverage can contribute to a better link utility in obstructed environments, such as large spacecrafts like the Herschel satellite with spans about 7.5m high and 4.0m wide. For RF transmissions, the problem is that at high transmission output rates other components suffer from the generated electromagnetic-field interference generated by these wireless devices. The other problem is that RF transmission output must be high enough to have a stable wireless link located inside the spacecraft, thus requiring more power.

3.2.2 The IEEE 802.15.4 low-rate standard

From recent studies like TU Delft [3], ESA [54] and [77] provided already that one of the most applicable wireless candidate is 802.15.4. This standard is also known as Zigbee, however Zigbee is a protocol stack over the physical RF transceiver implementation standard 802.15.4. This section provides some basic background information of 802.15.4 and possibilities of current wireless stacks based on this standard. 802.15.4 has some key features and benefits that allow the system to operate for many years without manual intervention:

- Simpler stack than bluetooth;
- Lower power consumption than bluetooth;
- The "Beacon-enabled mode" fits delay-sensitive sensor applications;
- A wide range of COTS components and platforms are readily available;
- It is interoperable (modular);
- Approved already for home-monitoring and (various) industrial applications.

Zigbee

Zigbee is a specification for the higher protocol layer, and builds upon the physical (PHY) and medium-access control (MAC) layers of the 802.15.4 specification. Zigbee defines three classes of devices: Zigbee Coordinators (ZC), Zigbee Routers (ZR), and Zigbee End Devices (ZED). Each network has one ZC, which is responsible for network formation and which can also aid in message routing. ZR's participate in routing and can run a sensing/ actuation application as well. ZED's only run applications and cannot participate in message routing. Zigbee can operate in both beaconed and nonbeaconed mode. In beaconed mode, the nodes are to some extent synchronized and the superframe is divided into 16 slots. The slots in the frame are generally contention-based, usingCarrier Sense Multiple Access/Collision Avoidance (CSMA/CA). There is an option to use up to seven of these as dedicated slots to specific nodes to increase determinism, a so-called guaranteed time slot (GTS), each spaced 10ms apart.

CSMA/CA is the main channel access mechanism used by Zigbee and most wireless LANs in the ISM bands. A channel access mechanism is the part of the protocol which specifies how the node uses the medium, e.g. when to listen and when to transmit. The basic principles of CSMA/CA are listen before talk (Carrier Sense) and Clear Channel Assessment (CCA) to determine if the medium is idle. This is an asynchronous message passing mechanism (connection-less), delivering a best effort service, but no bandwidth and latency guarantee. Its general advantage are that it is suited for network protocols such as TCP/IP. Also, this mechanism adapts quite well with the variable condition of traffic and is quite robust against interferences.

WirelessHART

WirelessHART [85] is a different MAC layer for 802.15.4, it is more reliable (uses contention-free MAC). Previous work shows that this form of MAC protocol should be more applicable for control & real-time sensing like in industrial applications [63].

Disadvantage is that WirelessHART uses encryption (security) as a default, and cannot be turned off completely. In space-applications security is less a problem, therefore it is best to have the option to turn it off completely to save on pre-processing power. However, since preprocessing takes place at the microcontroller side, (e.g. where the WirelessHART protocol stack is located), power expenditure might be insignificant as processing power might not dominate global sensor node power utilization. Other downsides are that this protocol only supports mesh-based networks (routing is enabled by default, on *each* device). Advantages of WirelessHART are the (real-time) short transmission times (up to 4ms delays in packet arrivals) and its Time Synchronized Mesh-Protocol (TSMP). TSMP uses frequency diversity, time diversity, and spatial diversity for reliability improvement. Another advantage is that WirelessHART is a Time Division Multiple Access (TDMA) based network. All devices are time synchronized and communicates in pre-scheduled fixed length time-slots. TDMA minimizes collisions and reduces the power consumption of the devices. The downside again is that WirelessHART does not support CSMA/CA operation modes, like in Zigbee. Unlike in Zigbee where you can have end-nodes and routers, each node in WirelessHART is a full router, and the precise timing requirements of TSMP allow all nodes to be duty-cycled to low power states more than 99% of the time.

At the time of writing this thesis, NASA is conducting first tests with WirelessHART sensor nodes in the Lunar Habitat Wireless Testbed (LHWT), which is focused on interoperability with other wireless devices [81].

ISA100

Currently, the emerging ISA100 standardization committee has activities underway with WirelessHART Convergence (ISA100.12), which is targeted for plant automation. The ISA100 committee is part of ISA and was formed in 2005 to establish standards and related information that will define procedures for implementing wireless systems in the automation and control environment. The ISA100.11a standard is intended to provide reliable and secure wireless operation for non-critical monitoring, alerting, supervisory control, open loop control, and closed loop control applications. ISA100.11a technology

enables real-time control of devices and collection of sensor data at 100ms intervals. The ISA100.11a Industrial Wireless Networking standard was approved in September 2009 and is the first in the ISA100 family of standards.

Power consumption

The adoption of a full custom design (or an Application Specific IC, ASIC) will always be optimized for power consumption; however modularity of sensors could introduce an increased power-consumption for particular sensors. This power consumption might be reduced by selecting the right interface and its components (e.g., low-power buses, transmission lines, external components, etc.).

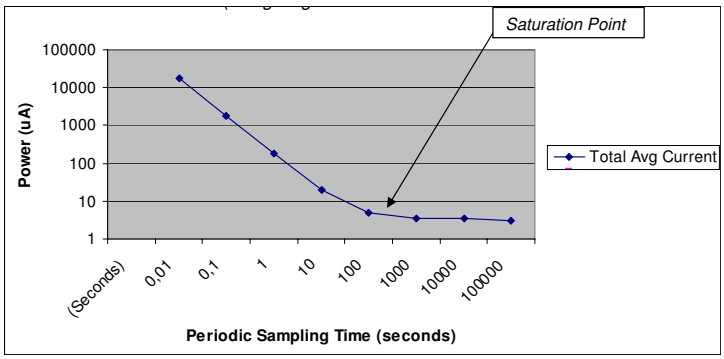


Figure 3.2: Average power consumption of the 2.4GHz 802.15.4 Jennic node vs. sampling time [38]

Also as the wireless sensor node components often have a "sleep mode", the sensor interface may require a "sleep mode" to save power consumption.

Interaction pattern vs. power consumption

Logging (i.e. storing some previous samples before transmission) might not have any benefits for power-reduction, as overheads in processing and compression time will take eventually more power. An example calculation, with the help of the Jennic datasheets [38], the power consumption can be calculated, depending on the sample (and transmission) duty cycles. The Jennic JN5121 device used is a fully IEEE 802.15.4 compliant device. It is shown that near 10-100 seconds of interval between samples (and transmissions), the saturation of the sleep current dominates over the consumed current. So the best area of the sensor node sampling interval is in this area, or on the left side of this area (see figure 3.2).

Many other factors at all system levels of the sensor node affect power consumption, but typical power consumption of the sensor node generally depends on:

- Duty cycle: low data rates will have sleep power dominating the system;
- Packet length: Long packet length will have radio transmission power dominating, short packet lengths will have calibration power dominating the system (figure 3.3).

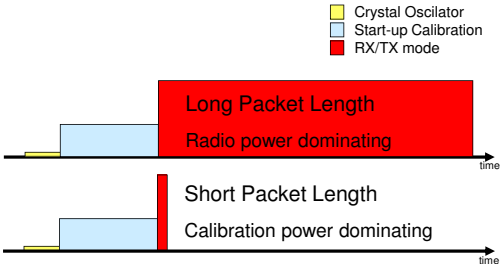


Figure 3.3: Packet length vs. power consumption

Latency

For a 1-byte transmission, the average latency is about 10ms for one packet to arrive at the next node. For larger packet sizes, 802.15.4 has a latency determinism of about 90ms, for a packet size of 114 bytes. The lower the packet size, the lower the end-to-end latency has been shown on calculations for 802.15.4 on 2.4Ghz[44]. It should be noted that when using sub-GHz bands (e.g. 868MHz) latency will also be higher than using the same platform at 2.4Ghz. That means that for a 2 byte package, we will have a latency of about 40ms[1], instead of 15ms at 2.4GHz.

3.2.3 Alternative hardware and systems

Many alternative hardware solutions of wireless sensors exists already in the form of commercially available systems and custom protocols, related to the IEEE 802.15.4 standard. This subsection gives a short overview of some examples.

Hybrid RF and microcontroller systems

Completely integrated solutions can be found on the market nowadays, integrating a microcontroller and baseband transceiver. These solutions however have limitations in the flexibility and number of I/Os available. An example SoC (System on Chip) from Texas Instruments is the CC430F5133. This system-on-chip incorporates both an RF part and MSP430 16-bit ultra low power microcontroller. Drawbacks of these SoCs are the limited flexibility of selecting an appropriate architecture and limited connection (e.g. limited I/O pins) to connect the sensor module.

Mote-on-Chip systems

Systems-on-chip that integrates both RF-part and microcontroller unit, and additionally, integrated radio circuitry components (mostly proprietary solutions) also exist. An example of a commercially available product is the Dust Networks DN2510 [61], which is wirelessHART compliant and only requires an external antenna for operation. Another (research) example of high integration, worth mentioning, is the ChipSat system [7],

[1]Note that this latency is excluding sensor conversion time, and includes network connection setup (CCA).

that can be considered as a single satellite-on-a-chip. It includes solar cells, a payload, antenna's and more. This combination of all elements on a single chip saves power even more, as the system is fine-tuned to work with each other and consists of a single SoC. The Mote on Chip (MoC), requiring zero external components, is a perfect example of complete miniaturisation and the ultimate goal of a wireless sensor node.

Simplified protocol stacks

Some vendor-specific proprietary protocols also exist and based on the 802.15.4 MAC, that are actually simplified versions of the (comprehensive) Zigbee stack, with a reduced set of communication instructions to speed up development (flexibility) and operate at lower power consumption. Examples are Microchip MiWi Stack [31], and TI SimpliciTI [34]. One major drawback of SimpliciTI is that the stack only supports to 30 nodes, so it is not applicable for larger networks.

Downscaling to 868/915MHz

Downgrading to a sub-GHz PHY interface could be a good option for various reasons. To name a few:

- Silicon is more power efficient at sub-GHz level;
- There is less transmission/antenna loss at sub-GHz (i.e. 8dB less loss at 915MHz versus 2.4GHz systems);
- Simpler antennas can be used;
- Generally, a higher link robustness is gained when operating at sub-GHz;
- Less interference than in the 2.4GHz bands due to less crowded bands.

3.3 Network Topologies

The topology will be generally highly dependent on the configuration of the spacecraft (i.e. complexity, and number of installed instruments) as well as the size of the spacecraft. For some instruments and sensors in the spacecraft, single wireless point-to-point links (e.g. wireless bus-based solutions) are be more suitable, while in other sensor systems a complete network are the correct solution.

When a number of wireless nodes is growing size to the order of 10s to 100s, a wireless sensor network might be the best option to guarantee more reliable systems. Traditionally, several problems exist, when using wireless point-to-point links:

- RF interference: There is no way to predict what interferers will be present at a given time and frequency;
- Node Loss, due to some external environmental (irradiation effects), hardware might fail, or semiconductor material will wear out after long operating lifetimes;
- Power failure, power brownout or blackouts.

Any of these problems will bring a point-to-point network down.

3.3.1 IEEE 802.15.4 standard topologies

The 802.15.4 standard supports two kinds of network topologies, star, and peer-to-peer, as shown in figure 3.4. In the star topology, all data exchanges are controlled by a coordinator that operates as a network master, while devices operate as slaves and communicate only with the coordinator. This single-hop network is most suitable for delay critical applications.

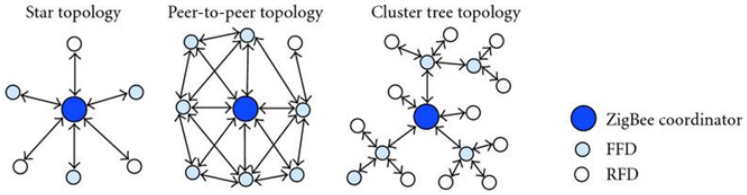


Figure 3.4: Star, peer2peer and cluster-tree based topologies [40]

A peer-to-peer topology allows *mesh* type networks, where any coordinator may communicate with any other coordinator within its range, and have messages multihop routed to coordinators outside its range. This topology enables the formation of self-organizing network topologies. An advantage of this peer-to-peer network is the ability to route around the loss of a single node.

A special type of peer-to-peer topology is a *cluster-tree network*, defined by Zigbee only. The network consists of clusters, each having a coordinator as a cluster head and multiple devices as leaf nodes. A coordinator initiates the network and serves as the root of it. This structure based on clusters, has some advantages as nodes in the cluster may save energy in a sleep mode. In peer-to-peer topology, nodes need to receive continuously to be able to receive data from other nodes in the network.

3.3.2 Topology proposal

Things that need to be considered prior selection of a topology are the reliability (node loss due to environmental factors) and possible power failure of individual nodes. As the network operates at fairly low data rates and extremely low duty-cycles, network saturation and throughput rates are less important. As a result, latency due to network saturation effects should be hardly possible.

In the considered scenario, the advantages are that the physical topology is fixed (stationary) and hence can be carefully selected at the development phase, before actual deployment. This advantage minimizes RF problems, like RF interference can be verified thoroughly before final deployment. Other advantages include that there is no need for online-topology control, like on-the fly assigning power levels to each node and the problem of (complex) ad-hoc based routing schemes.

Location centric topology

The basis for separation by hierarchy might also be more focused on location details, e.g. boom antenna's and long distances between S/C solar-panels and its flight-instruments

systems (i.e. deployables). For example, temperature sensors placed on the solar panels might use a relay node, in between the spacecrafts bus and remote locations that allow for routing of adjacent temperature sensor node information.

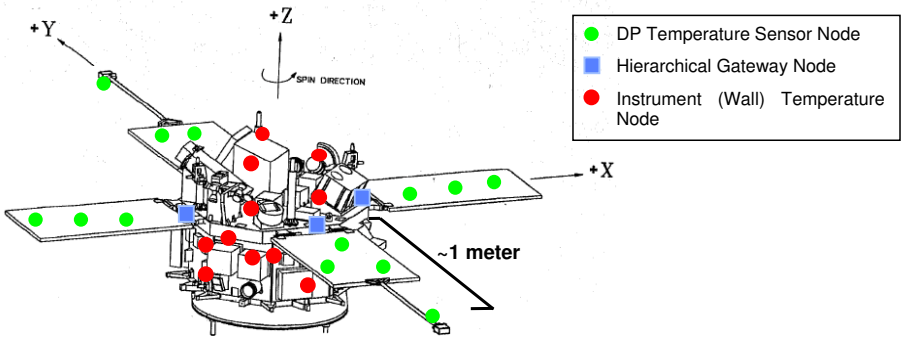


Figure 3.5: Conceptual Spacecraft WSN topology based on locations of nodes

Figures 3.5 depicts the locations of these sensor nodes and gateways on the ACE satellite. Figures 3.6 and 3.7 exemplify this conceptual topology in an abstract form, where figure 3.6 uses mesh for the sensor clusters.
By utilizing this location-oriented topology we have several advantages:

- Allows for routing & mesh-based sub network topologies;
- Allows for lower power consumption (for RF transmission) when transmitting to adjacent nodes (this, in term, leads to reduced EMI interference).

The topology is also known as a cluster-tree topology (see section 3.3.1 for comparison), but in this case with a wired backbone in between the clusters. One of the disad-

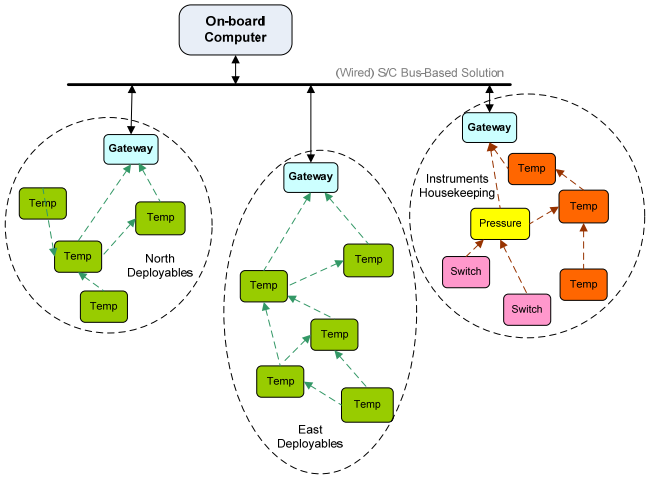


Figure 3.6: Conceptual WSN topology based on locations and using mesh for nodes

vantages of this topology are the routing overheads that are power-consuming due to the

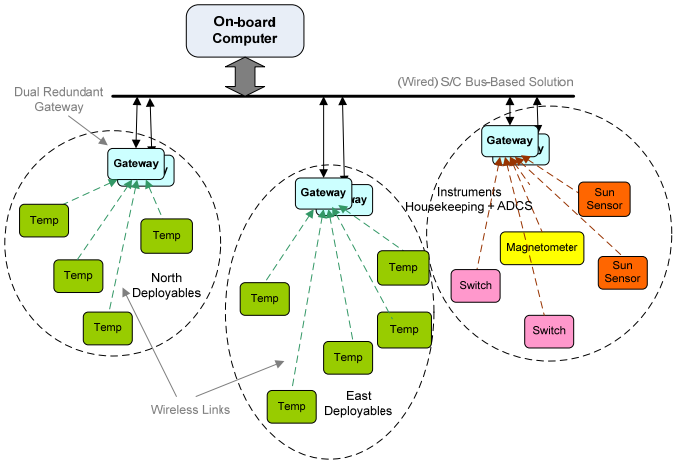


Figure 3.7: A conceptual WSN topology based on locations and using star topologies and incorporating gateway redundancy

listening, receiving and relaying of packets. Another disavantage is the single-point-of-failure at the gateway nodes. Additionally, routing significantly increases protocol complexity required in each sensor node and consumes additional wireless bandwidth. This approach with routing is considered not necessary for intra-spacecraft sensor networks using 802.15.4, however could become necessary for applications involving long-distances where transmission power limits the range of communication. To avoid overcome failures at the gateways, the gateway nodes should be wire-powered continuously. A drawback of the proposed single-hop topology is that the DP (Deployable) temperature sensor nodes might require an ability to transmit at higher power outputs to reach its (nearest) gateway. Typical range of 802.15.4 networks is about 10–20 meters[2], which should be enough for the current satellites.

Extend gateway functionality

Because the network already consists of multiple gateway nodes, increased functionality could be deliberately put on the gateway nodes, and will result in less complexity for the sensor nodes. This is specially the case for real-time sensor nodes, where simple sensor node hardware can be used without processing or storage. Configuration of the nodes (during AIT), might go through the gateway nodes. Also time-stamps, and the migration and delegation of the higher level commands/interaction patterns from the OBC should be located onto the gateway node. Therefore, it makes sense to put all higher level functionality in one dedicated gateway, and use simpler, wireless reconfigurable nodes as sensing nodes.

It is seen that the gateway is considered a single-point of failure in the network, and therefore (hardware) redundancy was considered as an option. Gateway redundancy is also typically required for multiple scenarios where high availability is required, such as

[2]depending on antenna configuration, transmission output power and other factors

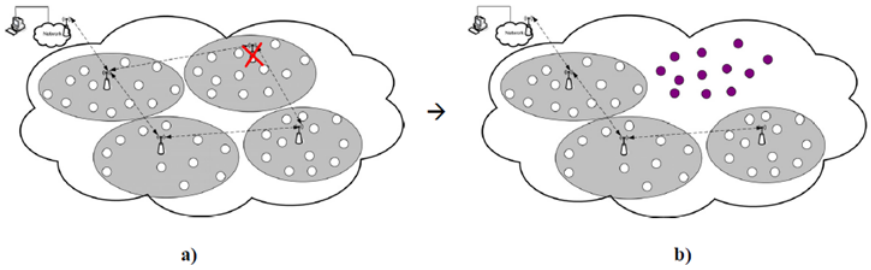


Figure 3.8: A multi-gateway topology (a) and access point failure, resulting in a quarter loss of sensor nodes (b)

for during launch, for mission control the availability (of sensors and system) is very important for the telemetry system. Figure 3.8 illustrates the common problem with sensor nodes not in range of the network. While the proposed topology in figure 3.7 utilizes closely coupled gateways, gateway switchover can only be beneficial for those sensors in range of both gateways. If, for larger spacecrafts, or in whatever (out of reach) gateway a fault appears, the network still fails. Therefore, wired (or dual) gateway hardware redundancy is still considered in the network.

Distributed sensors

Another possible topology is the use of multiple sensors per node. This leverages the number of global nodes, thus limiting total weight and power consumption for the system (see figure 3.9). An additional benefit is the reduction of the global (RF) bandwidth if

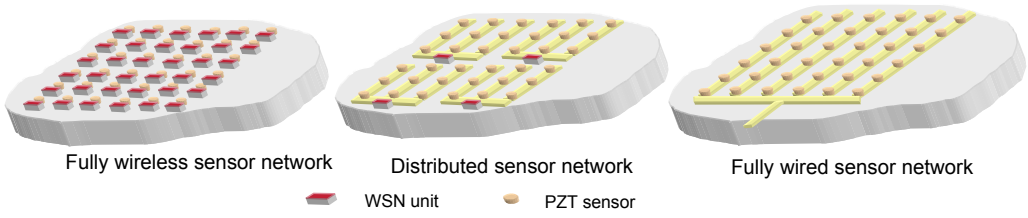


Figure 3.9: Comparison of sensor level distribution & wireless sensor networks [10]

multiple sensor readings being transmitted in one data packet. This eliminates communication overhead (such as CCA), and in the end allows for even more globally connected sensors.

Disadvantages of combined sensors within nodes is the level of systematic fault tolerance: when the RF device fails, several temperature sensors fail completely, instead of only one sensor. Therefore, this alternative is feasible only when power constraints are dominant and time-to-time failure of multiple nodes is accepted.

3.3.3 Bandwidth allocation

The data rates for 802.15.4 are considered to be slower (yet more robust) than common higher data rate wireless networks such as Wi-Fi. Therefore, a general study was performed of how many transmissions can be supported in 802.15.4 and its corresponding number of nodes that can be used in the network.
For applications, network bandwidth of the sensors can be allocated for 2.4GHz and sub-GHz, which corresponds with low latency and more channels to be allocated for transmission of individual sensors. For 2.4GHz, 802.15.4 supports 16 channels that may be allocated for bandwidth and 10 channels for 915MHz (see figure 3.10).

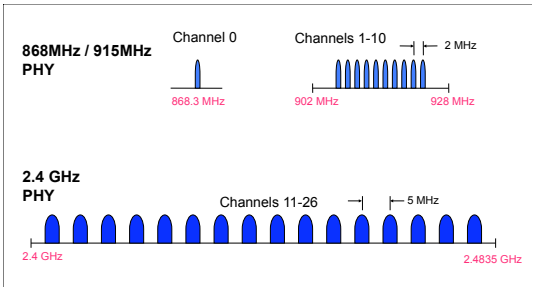


Figure 3.10: IEEE 802.15.4 PHY operating frequency bands

The results of the study (not included) have shown that there is limited bandwidth available for higher duty cycles than 1Hz. Since the majority of sensors only require tens of bits to be transmitted, other sensors, like the ADCS sensors require at most 20Hz and are already consuming almost 50% bandwidth in one 802.15.4 channel.

Traffic and flow diversity

Inside the spacecraft, the wireless network is in charge of handling different data traffic types. They include payload data, house-keeping data, ADCS sensors and actuators data traffic. The different data traffic types impose various requirements on the data rate and data handling system. House-keeping data are intended for monitoring the spacecraft's health. Transmission on best-effort basis can be utilized for these sensors. Yet, care has to be taken of that we then should be designing in the likelihood of failures (even if these failures are momentarily). This requires additional planning and fault-tolerant design at data-handling level.

Typically, the attitude data requires higher data rate and higher update frequency to guarantee almost real time attitude determination (see section 2.6). Additionally, increased criticality and availability for these sensors is also required. Selecting separate network hardware or frequency band for these sensors could be a better way than using the same network and use different type of quality of services on a single wireless network.

3.4 Wireless Platform Selection

A comprehensive tradeoff study has been performed during the thesis time. This section concludes the tradeoff with the current selected platform.

3.4.1 Protocol comparison

Hardware components seem to compete all in the same power consumption region, but have different solutions in software protocols (stacks) and operating systems. A summary of possible candidate stacks to be implemented on 802.15.4 is shown in table 3.1.

Stack	**Zigbee PRO**	**WirelessHART**	**Proprietary**
Standardized	Yes	Yes	No
PHY	802.15.4-2006	802.15.4	802.15.4/propr.
Stack Size	40K-100KB	6.5KB	1K-17KB
Real Time	Yes	Yes	varies
Link Robustness	Medium	High	varies
Operation Mode	Best-effort & GTS	Scheduled	varies

Table 3.1: Summarized differences in COTS wireless sensor software protocols

Zigbee PRO over IEEE 802.15.4 has been been chosen, as it has the following advantages:

- Improved maturity of standard compared to the other (relative new) standards
- Interoperability between different vendors allows for easy development and manufacturing
- Possibility of an open-standard Zigbee is underway
- Possibility to use beacon-enabled mode to allow scheduled transmissions
- Possibility of a sub-gigahertz implementations to separate network traffic for different kinds of critical sensor nodes.

Also, with the appropriate stack ontop of 802.15.4, there is no need to "reinvent" all the basic network functionality that is required for proper operation. Fact is that Zigbee PRO is capable of all basic functionality including various autonomy functionality such as self-configuration, self-healing and self-optimizing. The platform should support autonomous nodes as well as gateways that should organize network configuration by themselves, to ease the integration-test engineers and increases reliability for in-orbit solutions.

The advantages of WirelessHART where critically analyzed and it is observed that it currently has limitations in flexibility and availability (there are only a few implementations available). The use of routers (and thus mesh or tree based networks) in the network further decreases network bandwidth and fact is that there is already limited bandwidth available within 802.15.4.

3.4.2 The Atmel ZigBit development environment

The Atmel ZigBit development kit has been chosen as the development plaform for the wireless sensor network, mainly because of its flexibility and rich functionality. The following section detail the development platform characterestics and discuss some of the requirements that fit within the development system.
Advantages of the Atmel selected Zigbee development environment:

- Support for sub-GHz Zigbee PRO implementations (prototyping)
- Support for multiple operating systems (Contiki, TinyOS, etc.)
- Support for over the air upgrade/configurability
- Configuration server: allows for simple single-file configuration parameter changing (ideally suitable for TEDS or other configuration settings)
- Persistent Data Server: allows for quick fail recovery after power failure

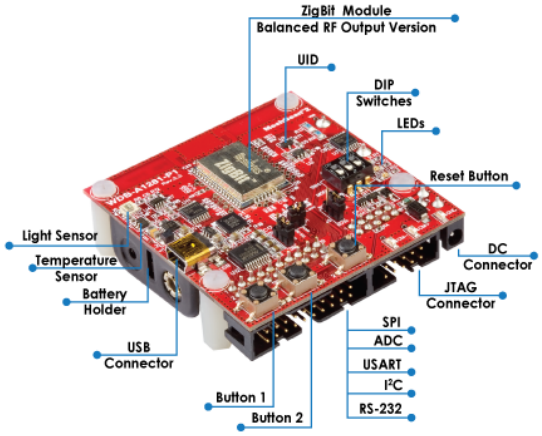


Figure 3.11: The Atmel Zigbit Wireless Sensor Development Kit

The ZigBit integrates an Atmel RISC microcontroller and a 802.15.4 compliant RF radio. The Zigbit module occupies less than a square inch of space. The device consists of an Atmel's ATmega1281V Microcontroller and AT86RF230 RF Transceiver. Fig. 3.12 illustrates the ZDM-A1281-B0 block diagram. The ZigBit device already contains a complete RF/MCU design with all the necessary passive components included.
Some of the key features of the ZigBit hardware device are:

- support for operation in 783/868/915MHz and 2.4GHz band;
- High receiving sensitivity;
- Low power consumption (<6 μA in sleep mode);
- Ample memory resources (128 kBytes of flash memory, 8 kBytes RAM, 4 kBytes EEPROM);
- Wide range of interfaces (both analog and digital);
- Support for contemporary protocols (e.g. Zigbee, WirelessHART and ISA100).

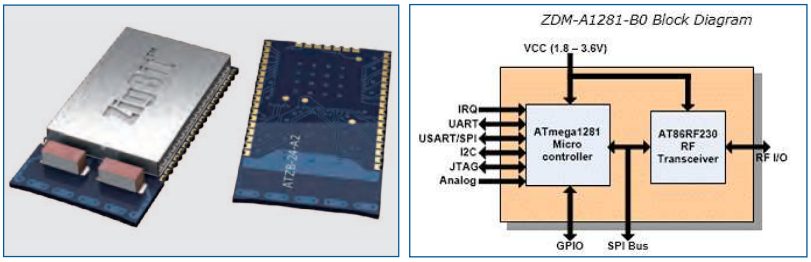


Figure 3.12: The Atmel ZigBit SoC and internals

3.4.3 Reconfigurability

For the application, reconfiguration of individual nodes is one of the important requirements (see sections 2.7 and 2.8). The level of reconfigurability demonstrates the amount that can be reconfigured properly in the system, at a certain time. Temporary parameters such as intermediate sensor values or the on-the-fly changing of the node interaction patterns with its parent node can be seen as examples of reconfiguration parameters. In contrast to predefined sensor nodes where configuration is static, reconfigurable architectures enable a reduction manual reconfiguration and intervention because its architecture can be adapted to the target application. This is ideal for AIT operations where there is much time and money spent in the constant reconfiguration of test support equipment for a particular setup or testbench. Additionally, in case of failure, the sensor node should quickly reconfigure itself and adapt to changes in the network however since the network is located at a fixed position inside the spacecraft during operation, reconfiguration of the network topology is of less concern.

We can distinguish different levels of reconfiguration like dynamic or static. Live reconfiguration (i.e. dynamic, on the fly) could be the the mode of operation for a particular sensor (e.g. interaction patterns) that might be changed from time to time during operation, when specific events occur. Offline (or static) reconfiguration could be TEDS for example and therefore could be stored in flash of the microcontroller.

Over the air reconfiguration

The Zigbit modules do support over the air reconfiguration, however this functionality is still in beta phase. For the ZigBit device, this functionality is called OTAU (Over-the-air-upgrade), but requires an additional EEPROM storage for the firmware. OTAU is using a special software program (written in Java) that allows for wireless firmware upgrade.

For static reconfiguration, one might choose to reprogram the entire flash of the mote (wired), however alternatives exist. For TinyOS, XNP [39] exists. TinyOS itself is not reconfigurable, however XNP that resides on top of TinyOS allows to install a new image on the mote and requires a reboot.

3.5 Conclusions

This chapter provided the functional requirements and goals for the wireless sensor network platform. Various standards-based wireless systems, research and related solutions have been discussed. Focus is on low-data rate sensor networks, and low duty-cycle network systems. IEEE 802.15.4 wireless hardware is considered a fault-tolerant, yet limited for applications where many sensors are deployed with higher duty-cycles than 1Hz. A protocol, that resides ontop of 802.15.4 should provide additional fault-tolerance and a selection of protocols have been discussed. Traffic for low-priority houskeeping sensors and the more critical attitude sensors should be split accordingly.

For this thesis, we mainly focused on the engineering housekeeping sensors in a spacecraft, and best-effort transmission based protocols and development software have been selected. A fault-tolerant topology, mapped for on-board a typical spacecraft has been proposed. Based on multiple gateways, the topology can utilize bandwith more efficient, and is single-point fail-safe.

4 System Design

The previous chapter discussed the wireless hardware, protocols and network topologies of the sensor network, this chapter presents the design of the architecture and components of the gateway and sensor node devices. While gateway and sensor nodes are both very similar by its underlying wireless architecture, its operation and hardware is quite different. Therefore, components for each wireless device are described separately in sections 4.1.1 and 4.1.2. Sensor node modularity is addressed by the proposal of specific hardware and middleware concepts in section 4.2. The last section concludes with a possible provision of the use of the sensor network during AIT operations.

4.1 Architecture

This section will cover the design details and the various aspects considered for the development of the wireless system development platform.

4.1.1 Gateway architecture

Figure 4.1 presents the envisioned gateway architecture. Hardware consists of a RISC microcontroller, a Zigbee radio and an Real-Time-Clock (RTC). An OS-based Application Programming Interface (API) provides access to the Zigbee stack operations and microcontroller resources. The stack has support of (initial) network parameter storage functionality, and custom area for other reconfiguration parameters, both programmable in EEPROM or Flash. Also, the TEDS pool is located and saved by this functionality.

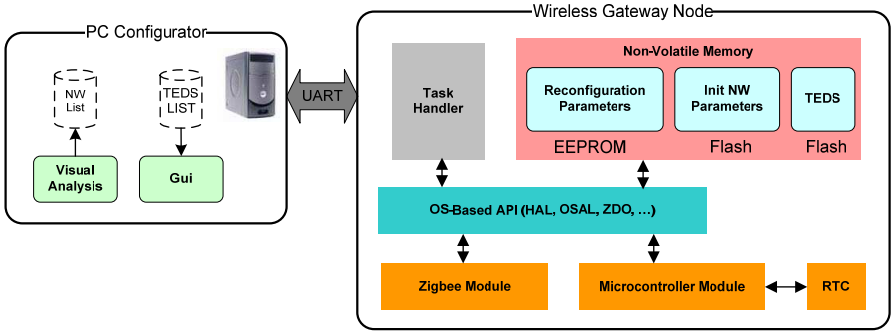


Figure 4.1: The envisioned gateway end-application software Architecture

The lower layers of the OS comprise of an Hardware Abstraction Layer (HAL) and various APIs for the network stack (i.e. ZDO, the Zigbee Device Object APIs). The highest level is an event-based task handler that can execute user and application code.

Gateway redundancy

In the fist phases of the thesis, gateway redundancy was investigated and recommended (see section 3.3.2) to provide a general higher availability (and global network reliability increase) during operation phase. This would also allow the system to accomodate dependable sensors.

The real time clock provides a time stamp as well as a non-volatile timestamp (by a backup battery) during switchover for the redundancy design. A task handler in the OS executes the appplication program, in this case the redundancy management and general Zigbee tasks.

Gateway operation

Typical tasks the gateway module has to perform:

- Time stamping of received sensor data.
- Redundancy management, when a second gateway is installed next to the primary gateway.
- Wireless collection and (possibly) dissemination of TEDS and (re)configuration parameters.
- Protocol conversion for communication with a PC/ on-board computer.

The initial network parameters are required for operation after a reset, to return to a "nominal" (or default) network state, that sensor nodes can join to. In the reconfiguration parameters area, various sensor interaction pattern data is stored for the network (including the nodes connected to that gateway), as well as network oriented parameters, like beacon enabled mode, CSMA mode, GTS mode, etc.

While reconfiguration of network settings might not be possible with Zigbee, over-the-air reconfiguration of the entire firmware is an considered as an option. At the PC side, a GUI (Graphical User interface) takes care of network setup, application setup (e.g. TEDS) and analyzing of network status during operation. In our setup, the PC is also used for debugging by means of a virtual com port (UART) over USB.

4.1.2 Sensor node architecture

The sensor node typically consists of the same hardware as the gateway (see the previous section), excluding the RTC and PC connection (see figure 4.2). The sensors are interconnected by a conditioning circuitry (typically an ADC or OPAMP) and a digital Universal Tranducer Interface, specified by one of the standards based interfacing systems described in section 2.7, e.g. IEEE 1451.5.

An interchangeable power module with power control is interconnected directly to the microcontroller and provides power to the sensors, conditioning circuitry, Zigbee module and microcontroller. Not shown, but typically present in common 802.15.4 compliant devices is an Unique Identification Chip (an UID). This chip provides an unique 64-bit MAC address for each node. Section 4.2 provides details and schematics for the general modular sensor node architecture. TEDS is typically stored in the Flash area of the microcontroller, e.g. received by the gateway and correlated by an UID on the sensor module (see section 2.7).

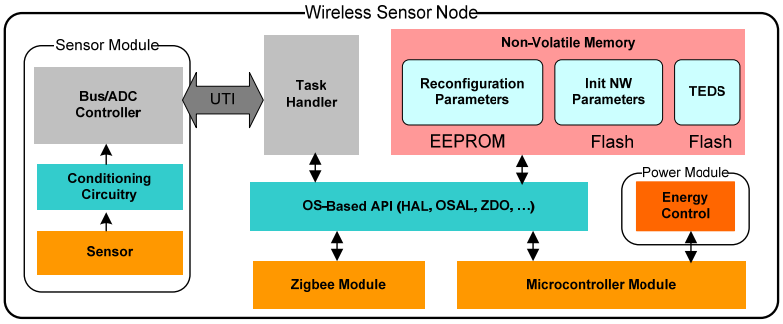


Figure 4.2: The envisioned architecture of the sensor node side.

Typical tasks the controller on the sensor node has to perform are:

- Sensor readout & RF Transmission
- Applying sensor data reduction schemes
- Interaction pattern setup (sample rates, events, etc.)
- Scheduling/Configuration setup
- Configuration of TEDS for its sensor module
- Energy control for the connected power source, like shutdown, sleeping modes, etc.

The sensor controller has to perform correct conversion of sensor data, and distribution of sensor data to the sensor node gateway or access point.

4.2 Modularity

The following proposal (figure 4.3) uses a pin-powered, I2C based modular approach but still needs a lot of refinement though. Using the microcontroller pin as power source, additional quiescent current can be saved and the sensor can be turned on only when sensing is required. For simplicity a unique address (using an UID) can be used, however in final design, a more custom or EEPROM based address might be used. For powering the sensor, different methods where considered, including solar cells, secondary battery cells, a hard wired power bus and power scavenging. As an example, four 1×1cm solar cells (SCs) are shown with corresponding BCRs (Battery Charge Regulators) and some limited power conditioning.

4.2.1 Conditioning circuitry

A programmable conditioning circuitry for different sensor types is the best option selected. Various integrated circuit devices are already COTS available, and it is time to make a comparison of functionality between them. The advantages of an external sigma-delta conditioning device are obvious, it allows for 256 times the resolution of the internal microcontroller's ADC (10 bits) and has additional support for multiple sensors and low power conditioning circuitry, ideal for the wireless sensor nodes that have to be ultra low power.

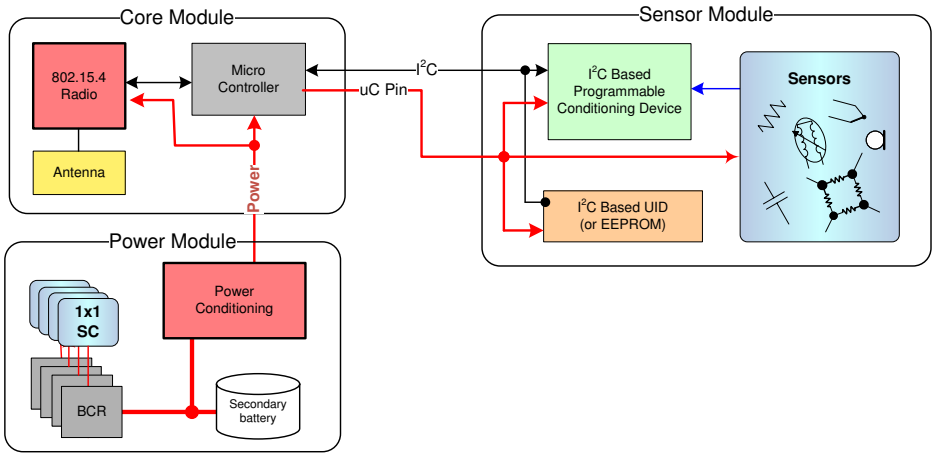


Figure 4.3: Proposal for a modular wireless sensor node (red lines indicate power lines)

Conditioning IC	**MCP3421**	**MCP3221**	**MCP3551**
ADC Type	Sigma-Delta	Traditional SAR	Sigma-Delta
ADC Accuracy	18-bits	12 bits	22 bits
Supported Sensors	1 ch.	1 ch.	1 ch.
ADC Settling time	4ms+1ms powerup	10μs	50ms
Bus Interface	I2C	I2C	SPI
On-board VREF	Yes	No	No
Power consumption	At 3V: – 145 μA typ. (continuous) – 0.39 μA (1Hz) – 0.1 μA (sleep)	At 5V: – 175 μA typ. (conversion) – 5nA (sleep)	At 2.7V: – 100 μA typ. (conversion) – 10 μA (sleep)
Operating margin	–55 to +125°C	–40 to +125°C	–40 to +125°C

Table 4.1: Comparison between different (low-power) sensor signal conditioning circuits

During the course of the thesis, the AD590 was considered as one of the reliable temperature sensors, as it is both military and aerospace-qualified [65].

Problems did occur, as it required at least 4V of operation (8V typically), and since the sensor node hardware operates at 1.8–3V only, additional power conditioning is required which might cause in a reduced reliability. It was difficult to find ultra-low power aerospace compliant components, or even low power components made for aerospace at all (i.e. components in which draw a few microamps).

There are also integrated (silicon based) temperature sensors, however all of these specify their own protocol and are less flexible by accuracy, repeatability and reliability than regular analog sensors. To be fully plug and play, an additional controller with its own protocol conversion has to be implemented to allow communication with the core

module. For simplicity, prototyping could benefit from an integrated conditioning device with most parts integrated, such as ADC, on-board voltage reference and built-in I2C controller. The choice of this integrated device speeds up development, however has some drawbacks:

- Lack of support of custom sampling frequencies
- Additional inclusion for cold-junction compensation is required for thermocouples

Advantages of the selected components:

- Support for high-resolution sensors, such as platinum based RTDs, high accurate pressure sensors, etc.
- Limited external components required for prototyping
- Uniform, standardized I2C interfacing
- Low power consumption and power down support for the sensor node
- Wide operating margin

The typical setup of the system will be thus based on (families of) these devices, which have minimal components and allow for a I2C communication protocol to be specified by TEDS. Figure 4.4 illustrates this conceptual setup.

Still, I2C is preferred above SPI as you can have multiple sensor conditioning devices (maximum of 7) connected through only 2 wires (one I2C device holds the TEDS), where SPI requires 3 wires per device (one is chipselect), so you need additional logic and hardware to select devices. The Sigma-Delta conditioning circuits support 256× the accuracy of the common Successive approximation Register (SAR) based ADCs, and support on-board voltage references. Problems with this kind of ADC devices is the settling time of the AD conversion. In Sigma-delta ADC devices, it can go up to 125ms, depending on accuracy required. For the sensor nodes, we still require a limited 'on' time and need to reduce power consumption during active periods. The best way to go is to have:

1) Either sleep the microcontroller device, during conversion.
2) Use a faster conversion time, thus a faster ADC device.

Option (1) seems possible with some devices, however require the microcontroller to wakeup again, requiring again some startup time (a couple milliseconds, oscillator startup, program start), however seems complex to implement using the Zigbit functionality due to timeouts and the inaccuracies of the sleep timers (they are rounded to 10ms) as they are basically running on TinyOS. Therefore, the other option (2) has preference for long life operation (many years), as is does not require multiple shutdowns and boot-ups of the microcontroller to and from sleep and performs acquisition in less than one millisecond. The other option was to choose a fast Sigma-Delta ADC (e.g. AD7798), which samples about 470Hz (2ms). Even at this rate, sampling accuracy is exactly the same as a SAR ADC, 12bits. The alternative is to use a high-speed sigma delta ADC like the AD7190, however increased (continuous) power consumption is the bottleneck here. Sampling for these kind of ADCS already draw couple of milliamperes versus the few microamperes for the 'low speed' sigma-delta ADCs. At the end, the application determines the accuracy, sampling frequency and other parameters. With the use of an modular interface, the ADC devices can be selected freely and TEDS specify the required

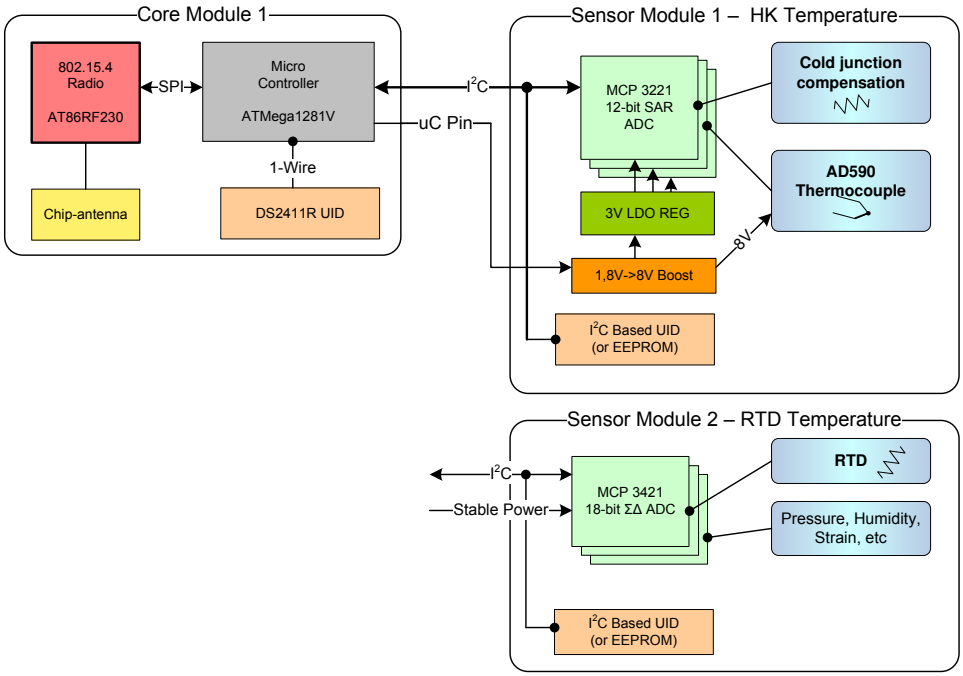


Figure 4.4: ZigBit based I2C modular sensor system proposals

and supported operating parameters. Connecting a sensor directly to the Zigbee microcontroller is not possible, you still need an OPAMP or other conditioning circuitry. Most lower power SAR ADCs have no built-in power reference and need additional external circuitry for this. Considering that the continuous supply current for micropower voltage references has to be added with the continuous currents and overall power consumption is therefore higher than by utilizing a regular all-in-one highly integrated ADC with all necessary components. Also, the RTD/Thermocouple designs utilize a differential input and require differential sensing opamps for better readout with limited noise and ground problems. Figure 4.4 shows the current proposed results for a low-power modular sensor system, based on decisions described above and during the thesis.

4.2.2 TEDS templates/ Application information

An example template for simple TEDS is created for testing functionality of a modular and self-describing approach is shown in figure 4.5. To keep things simple, only basic functionality of simple housekeeping sensors are included. The actual test implementation is structure wise, however due to limited time, tests could not be performed as the actual node hardware and sensor module hardware is still not yet completed. Appendix-D lists an example strain sensor TEDS configuration. Included also are signal conditioning specific parameters, such as commands supported by the ADC device, like setup and configuration (I2C read speed, cold start delays, etc.). Basic geo-TEDS for the nodes is also included, such as general sensor locations and node locations (e.g. if the actual sensors are not at the same locations as the node itself).

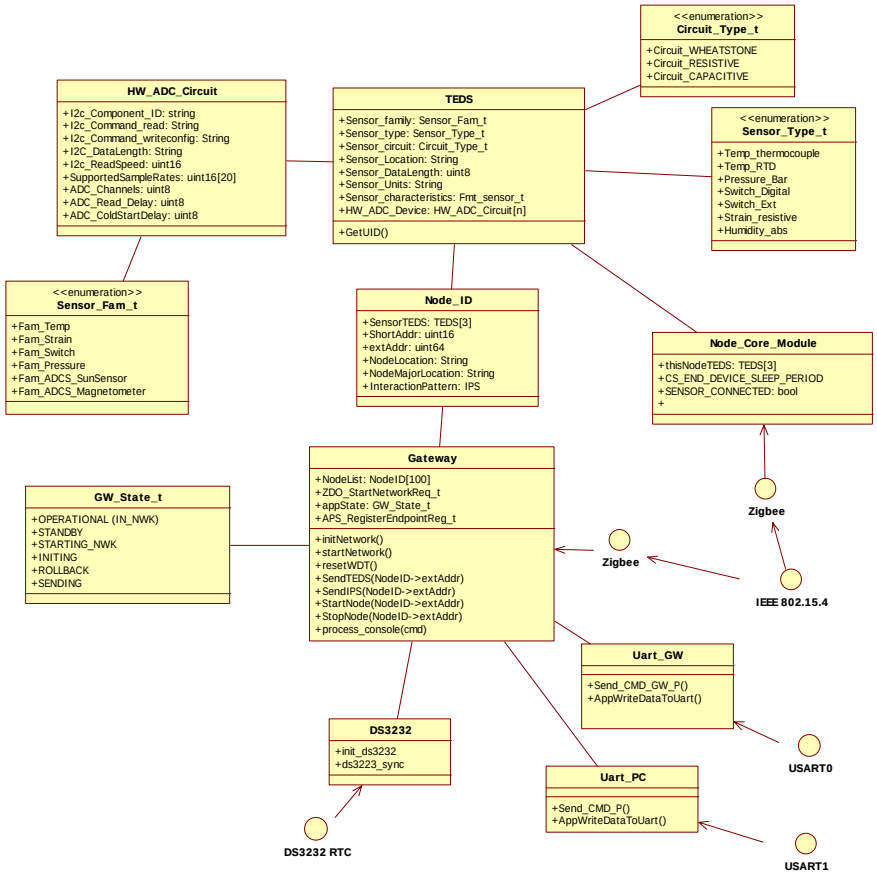


Figure 4.5: An example overview of application structures, sensor node TEDS and gateway dependencies

The gateway redundancy functionality is also shown in the figure 4.5, and some (example) future application-based operations and commands that are used for delegating and controlling the sensor nodes. Examples are starting/stopping nodes and transmission of interaction patterns (IPS).

4.2.3 TEDS alternatives

As some basic TEDS functionality is recommended, a standards-based implementation like IEEE 1451 is considered fairly complex, however allow for of a proof of a concept of the TEDS. There are alternatives, but not yet approved, or used in industry widely. xTEDS is considered as a viable alternative (used in the SPA PnP concepts), but still considering that this is in development, limited information about the concept was gained.

Figure 4.6 exemplifies the extension of the platform for a xTEDS pool, considering that multiple xTEDS have to be stored in the gateway node. Additional software is yet still required for sending and correlating the TEDS with node MAC address and

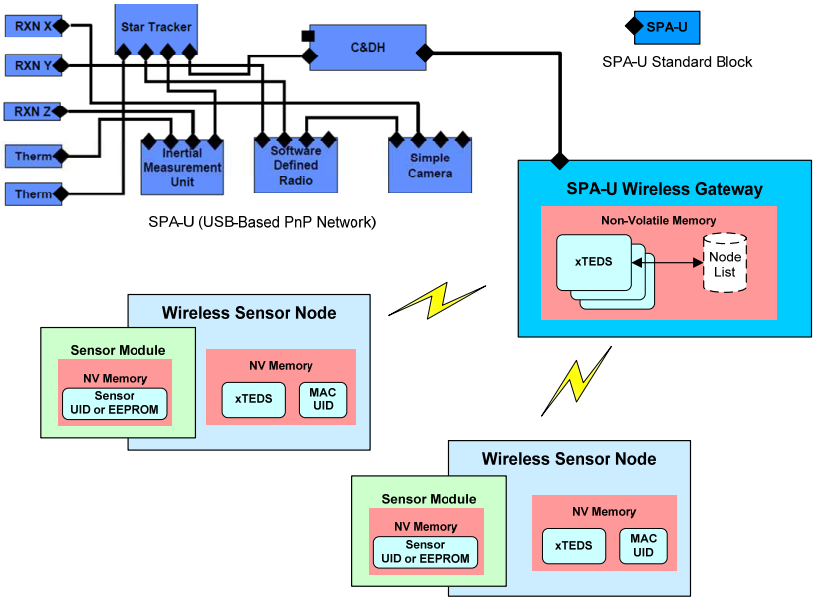


Figure 4.6: A wireless SPA-U sensor network concept, using xTEDS based on SPA.

its connected sensor module. Again, due to overheads TEDS might be located in the gateway nodes only, thus multiple TEDS has to be stored in there, this is never the case with SPA: each TEDS is separately hard coded into the RTU device or SPA-compliant end device. The problem is the faced efficiency of IEEE1451. It only supports TEDS, no commands or flexible ICD information are supported by TEDS (like I2C commands, variables, etc.) In the eyes of the author of this thesis, the SPA xTEDS is a wonderful alternative to TEDS as it is much more flexible, contemporary (XML based), etc.

The problem with collecting the xTEDS (and forwarding to the upper layers) is that it still needs to be correlated with the number of connected wireless nodes, the position where the sensors are located on. This can be done by using the IEEE 802.15.4 MAC Address, retrieved from a static 64-bit UID chip. The overall advantage is that the SPA-1 (I2C variant of SPA) can also be directly used in the sensor nodes, and SPA-U based xTEDS directly at the gateway.
For samples SPA-U xTEDS used in current (prototyped) spacecrafts, surf to:

http://www.datadesigncorp.net/xteds/app

Other examples utilize TCP-based self-describing languages (RESTful) over 8 bit micro-controllers, like in [68]. This approach is considered to be only valid when large amounts of resources are available both the gateway and sensor node.

JSON
The use of JavaScript Object Notation (JSON) eliminates the need of complex parsers, but still is pretty much a higher level platform-independent parser. The need for a universal, lightweight embedded parser is key in order to enable interchangeability of sensors. Advantages of these higher level parsers are improved readability than other

byte-code parsers.

4.3 Visibility During AIT

In current default wireless systems, all nodes are connected to a centralized sink. During AIT *all* nodes should be visible, therefore requiring a hook on the centralized sink, or alternatively, the use of (wireless) eavesdropping node traffic by listening to all nodes in the network using a sniffer. The latter option seems quickly realizable in hardware as IEEE 802.15.4 sniffers are abundantly available in any form and run on any platform. The problem here is data collection (e.g. centralized access) and data storage. Another point is the requirement of the implementation of the detection and handling of duplicate message and protocol formats (at both the gateways in the satellite and AIT gateways) required for offline sensory analysis. (e.g. one message is from ADCS sensor, using WirelessHART and one is stress test, using Zigbee with TEDS). Figure 4.7 shows a conceptual satellite framework using IEEE 802.15.4 during AIT operations, called a "wireless" space dock.

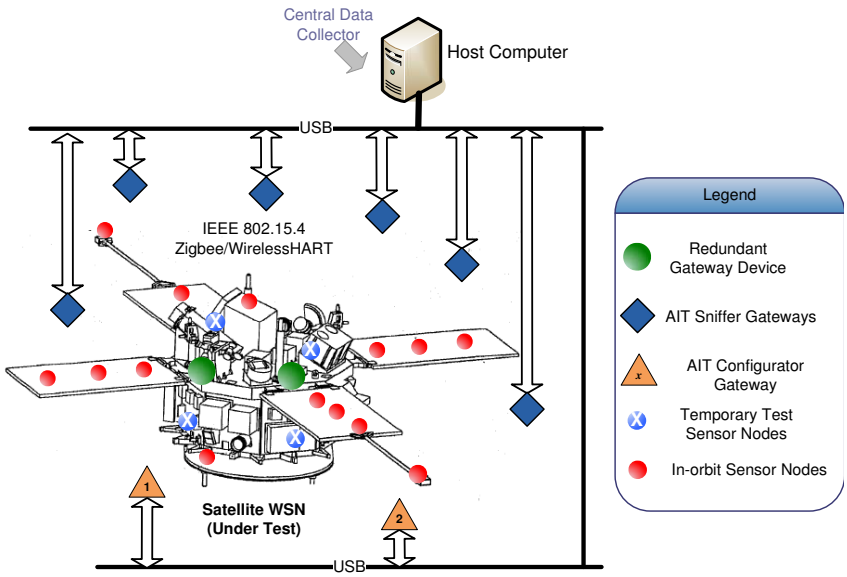


Figure 4.7: Combined sensor node setup of in-orbit and additional (test) sensors using sniffer gateways during AIT phase

Due to the limited number of available channels in 2.4GHz Zigbee, there should be a separate allocation of channels for testing sensors made before deployment. Another option is to use Software Defined Radios (SDR) to be set up as (fast) multi-channel sniffers by a fast (multi-core) PC like in [11]. The sniffers will feed data to an AIT PC which acts as the configuration and logging PC which will store the sensor data and other wireless transactions to perform analysis with. Drawbacks are the needs of additional software (or frameworks) that perform reformatting and filtering of sensor data for inspection.

The AIT configurator gateways shall be used by the PC to configure the redundant gateways inside the spacecraft. Configuration settings like TEDS listings, sensor node interaction-patterns and gateway configuration settings (such as the maximum number of allowed nodes per gateway or node allowance lists) can be send wirelessly over to the gateways. One downside is that again additional reconfiguration routines should be located on the gateways, like in the wired setups. The advantages however could be a dramatic simplification during AIT operations.

4.4 Conclusions

During the preliminary design phase, different components were identified and a set of proposals were made. The sensor node hardware should be simple, and yet fault tolerant, by using a combination of low-power hardware components and reliable sensors.

Prototyping with the modular hardware could be implemented by connecting a simple breakout board to the development kit and programming the lookup of TEDS afterwards should be simple, when already having a working Zigbee prototype. TEDS functionality should be addressed limited; however basic parts from xTEDS will also be recommended to be used for a basic template. TEDS data can be stored into (local) EEPROM or UID devices can be used to correlate for TEDS stored centrally, however implies over-the-air transmission of TEDS.

During AIT operations, additional wireless equipment and middleware can act as wireless fault-injection, testing and verification of the spacecraft. Software defined radio can be used as a fast alternative to real hardware for ground test equipment.

5 Experimental Setup & Results

This chapter presents the test results of envisioned the sensor network platform. Secondly, first tests of gateway redundancy are . This chapter is organized as follows. First, the wireless platform details are discussed in section 5.1.1. After a simple performance measurement to test transmission duty cycle, first tests with hot and cold redundancy were conducted and results are presented in section 5.2. Implementation details are presented in section 5.2.1. Discussion for the implemented redundancy scheme is presented in section 5.3.1 in the next sections. Finally, in section 5.4 the complete platform architecture concept is proposed.

5.1 Platform Software

5.1.1 The Atmel BitCloud software architecture

The Zigbee stack used is the BitCloud Zigbee development suite (version 1.9.0) downloadable from the Atmel website. It includes a Software Development Kit with specific support for the ZigBit kit. Moreover it comes with application examples, which make the development process much faster. The development software used was AVR Studio 4. This software includes all the tools needed to develop, debug and download the firmware to the microcontrollers in the circuit boards. Uploading the firmware can be easily done by a USB programmer tool by Atmel.

The stack is composed of multiple layers which have entities responsible for data transmission services and management services. A detailed overview of the stack is shown in Figure 5.1 below.

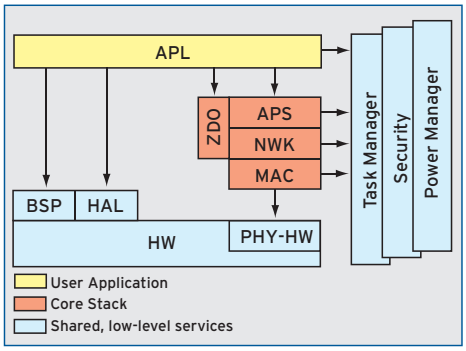


Figure 5.1: Atmel BitCloud software architecture

The Zigbee Device Object (ZDO), which is responsible for the network management functions. They use the features of the network through a set of services provided by the

Application Support Sublayer (APS). The Hardware Abstraction Layer (HAL) includes a complete set of APIs for using on-module hardware resources (EEPROM, app, sleep, and watchdog timers) as well as the reference drivers for rapid design and smooth integration with a range of external peripherals. The *task manager* is where the actual application is running, it supports scheduling tasks in a priority queued way. The Board Support Package (BSP) includes a complete set of drivers for managing standard peripherals for the selected meshbean development board such as sensors, a Unique Identification chip (UID), etc.

The applications on the BitCloud SDK are written in an event-driven programming style. Event-driven programming or event-based programming is a programming paradigm in which the flow of the program is determined by events such as sensor outputs, key presses or messages from other peripherals. In fact, all internal stack interfaces are defined in terms of forward calls and corresponding callbacks. In this way, a reasonable robust way of applications can be written ontop of this paradigm. Its programming model takes a little getting used to, and computationally-intensive applications can be difficult to write.

Hardware interfacing

The BitCloud API also provides an extensive support of common general-purpose interfaces. In order to enable communication over UART interface, application first configures corresponding UART port using static global variable of `HAL_UartDescriptor_t` type. Second, data reception over UART is configured for operation in callback mode. Moreover, UART settings is applied using `HAL_OpenUart()` function with argument pointing to global variable of `HAL_UartDescriptor_t` type with desired port configuration. Returned value indicates whether port is opened successfully and can be used for data exchange.

Config Server (CS)

The BitCloud stack provides an extensive set of network and system configuration parameters which determine different aspects of network and node behaviour. These parameters are accessible for application via Configuration Server (CS) interface. In order to perform parameter read/write procedure at run-time, the API functions, `CS_ReadParameter()` and `CS_WriteParameter()` are used. Both functions require parameter ID and a pointer to parameter value as arguments.

Persistent Data Server (PDS)

Another feature of the BitCloud stack is a built-in *check pointing algorithm*, which periodically stores data to EEPROM. All Config Server parameters can be divided into two categories: persistent and non persistent. Persistent parameters are stored in power independent EEPROM memory and their values are accessible for application and the stack after node hardware-reset. Non persistent parameters are stored in RAM and upon hardware-reset are reinitialized with their default values. The PDS also detects any CRC errors by automatically storing a checksum alongside every parameter stored

in EEPROM. When a parameter is read, the checksum is computed and compared to the one stored in EEPROM. The PDS can also be used at the application layer to correctly store user-data to EEPROM with corresponding CRC by using `PDS_WriteUserData()`.

5.2 Gateway Redundancy Implementation

The first tests conducted with the gateway were based on cold-redundancy. It was assumed that a second gateway was powered off and only kicks in after fault detection and afterwards restores back data to the primary gateway and goes back to "cold" standby, performing check pointing and error detection. While the first tests seemed an correct switchover operation, it was found that (cold) booting the gateway took almost 2 seconds, thus a lot of "missed" sensor data can occur during this period. As an alternative approach, a "hot" redundant scheme was proposed and tested out. In this redundancy scheme, one gateway is master and the second one is slave. The master is executing the full network stack operations, the slave is in standby mode (though initialization parameters where loaded into memory and OS are booted). Fault detection communication is in between master and slave gateways.

5.2.1 Experimental setup

In the lab, and evaluation was made what was required for the redundancy communication scheme. Figure. 5.2 illustrates a simple laboratory setup were tests are performed with, which consists of two sensor nodes configured as Zigbee coordinators (i.e. gateways) and one as an end-device (i.e. sensor node), which is capable of sleeping.

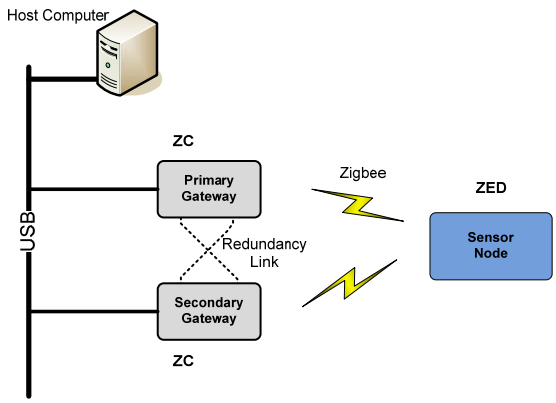


Figure 5.2: Experimental evaluation of redundancy and switchover functionality

The gateway system states are depicted in figure 5.3. With the help of check pointing (i.e. the periodic storage of the device state), a switchover could be transparent to the sensor nodes. Note that rollback (the restoration of a previous check point) is shown here, but used in a different way, and no communication-link in between coordinators was implemented (see section 5.2.3). The actual tested detailed switchover timings with rollbacks are depicted in the figure in Appendix B.

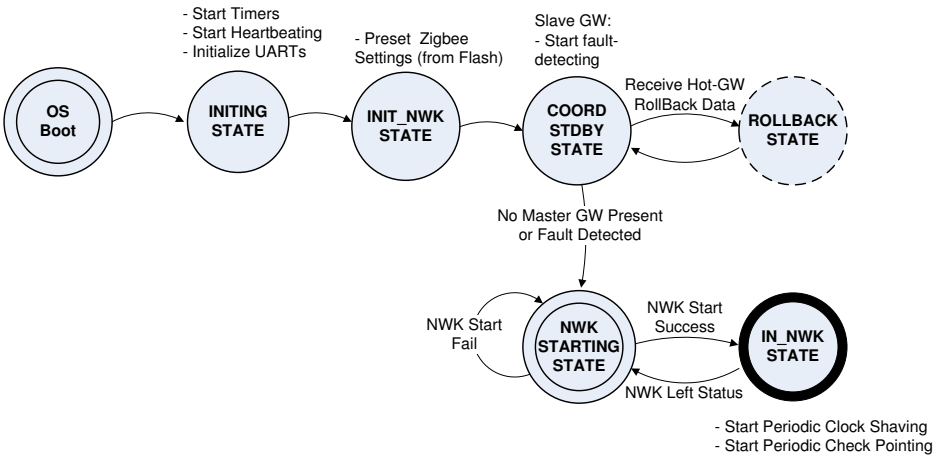


Figure 5.3: Simplified state diagram of transparent redundancy operations

5.2.2 Zigbee node poll rates

Using Zigbee, the amputation of nodes by leaving and rejoining is quite tedious and timeconsuming due to the additional network commands and delays. However Zigbee supports an additional functionality that allows for a graceful removal of connected network devices. Zigbee incorporates low indirect poll rates, i.e. the poll rates of a message waiting to be received for the sensor node is adjustable to a very low frequency. For example, poll rates may be send out by every 4 data transmissions (e.g. consider data transmission to take place every second, the poll rate is then 0.25Hz). See figure 5.4 for a sequence diagram where periodic polling is shown. When the child device (in our case the

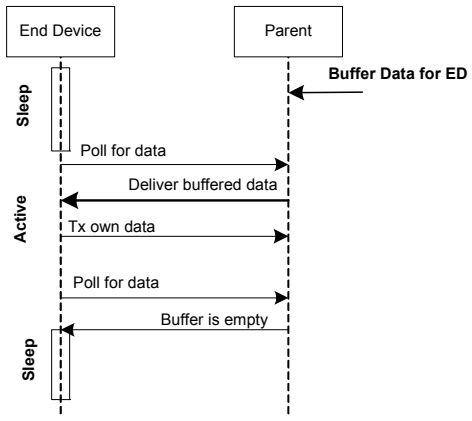


Figure 5.4: Sequence diagram depicting two (indirect) polls during active times of the end-device (sensor node)

sensor node) does not receive a poll acknowledge 3 times, it assumes that the coordinator or router is offline (i.e. network loss will be detected $3 \times$`CS_INDIRECT_POLL_RATE` $= 12$ seconds). Figure 5.5 shows an example redundant switchover timing sequence, based on

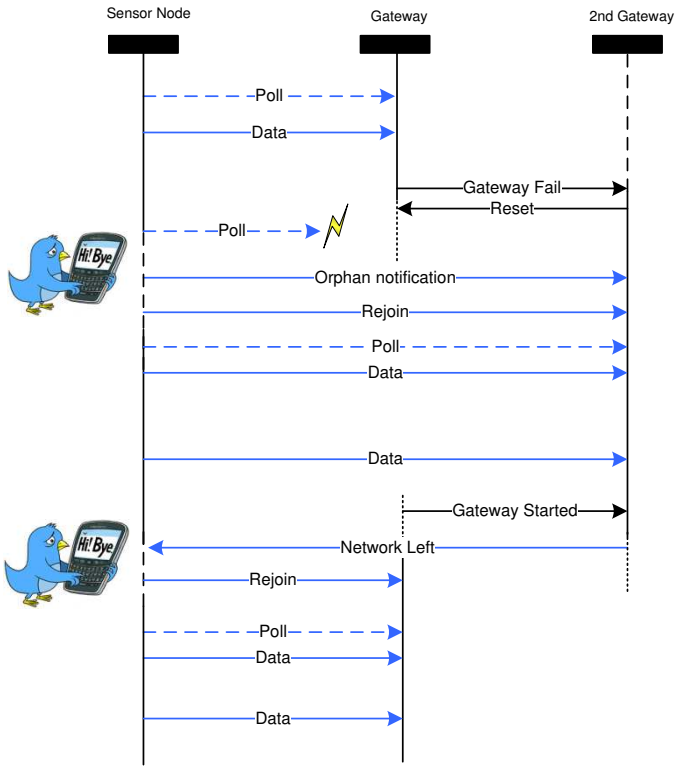


Figure 5.5: (Simplified) timing diagram of an example switchover functionality using Zigbee

the inherent Zigbee poll timeouts and poll rates.

5.2.3 Implementation issues

The first thing was to strip off unused functionality from the Zigbee stack (using BitCloud). Basic debug functionality is still kept, such as UART ports, stdio libraries and some of the board support packages (buttons, LEDs, etc.). C-Code was split for Zigbee coordinator and end-devices.

Since the worst-case application requires data rates of 10–20Hz (see section 3.1), a performance test was conducted to see if this was achievable on a single node. Problems did occur on end-devices because of additional latency of going to sleep and waking up the stack. It was found that at most a 9Hz duty cycle can be setup with the used ZigBit devices and the delivered Zigbee stack. For test purposes, an 8Hz sample rate (125ms duty cycle) was found reasonably accurate by conducting initial tests (with no other interfering nodes or frequency disturbances in the channel). This frequency also was taken to have a representative bandwidth utilization (for many nodes operating at lower frequencies), to test gateway throughput during switchover.

RTC

For time stamping, the RTC is directly connected to the gateway's I2C interface, and provides an interrupt pin for periodic "shaving" of the internal system clock (controlled by the OS) with an offset. The internal clock source runs on a 32kHz crystal, so it should be accurate, however due to temperature variations this clock will drift easily. The DS3232 has a high-accuracy, temperature compensated, built in clock source and stores both date and time. Together with the (OS) system clock a resolution of ±1ms is achieved for timestamps[1].

Check pointing data

For check pointing data, there are two different classes of information that needs to be check pointed. First there is initialization check pointing data, more or less static parameters like selected topology (maximum network depth and buffer size amounts) and dynamic parameters that could change frequently during operation of the network, such as network operating channels within Zigbee PRO.
Full Flash and EEPROM based check pointing are already supported by some of the (wireless) sensor operating systems like Contiki (Coffee File System [76]) and TinyOS (Capsule Storage System [55]) as an additional functionality. These are based on OS-like filesystem functionality and leave check pointing invisible to the application task. Also, these algorithms are meant to run on a single device (e.g. acting as a single node failure recovery) and do not support other hardware devices than the tested ones. As the main goal is increasing availability and thus increase robustness of the network, so adding hardware redundancy requires external communication between the gateways.

Redundant gateway hardware

The redundant gateway is single-point fail tolerant and should therefore have built in redundancy functionality that is implementable by using by the stack and OS. A proposed hardware lab setup for the Zigbit devices is shown in Figure 5.6.

Redundancy functionality is controlled by each gateway and its supervisory circuits. In addition, external watchdog timers should provide additional robustness to OS failures if the standby gateway also fails. A shared UID over 1-wire is proposed as UID has to be the same when utilizing transparent switchover. The primary purpose of the UID is to supply a unique MAC address to the BitCloud software stack. As the UID always includes an unique address (it is actually factory lasered), a single UID must be used for both gateways to have the same MAC addresses for switchover operation. If the UID chip is not present, the stack will look at a predefined memory address in EEPROM to get the MAC address. For convenience, during the tests a hardcoded (arbitrary) MAC address was used. In the end-design multiple differences must be taken into account, such as duplicate RTCs (which require additional duplicate synchronized time setting logic). A shared RTC might be an option, however additional logic has to come in as I2C

[1]The next generation of ATMega128 (ATXMEGA128A1) microcontrollers provide a built-in RTC with support of virtually the same functionality as the DS3232

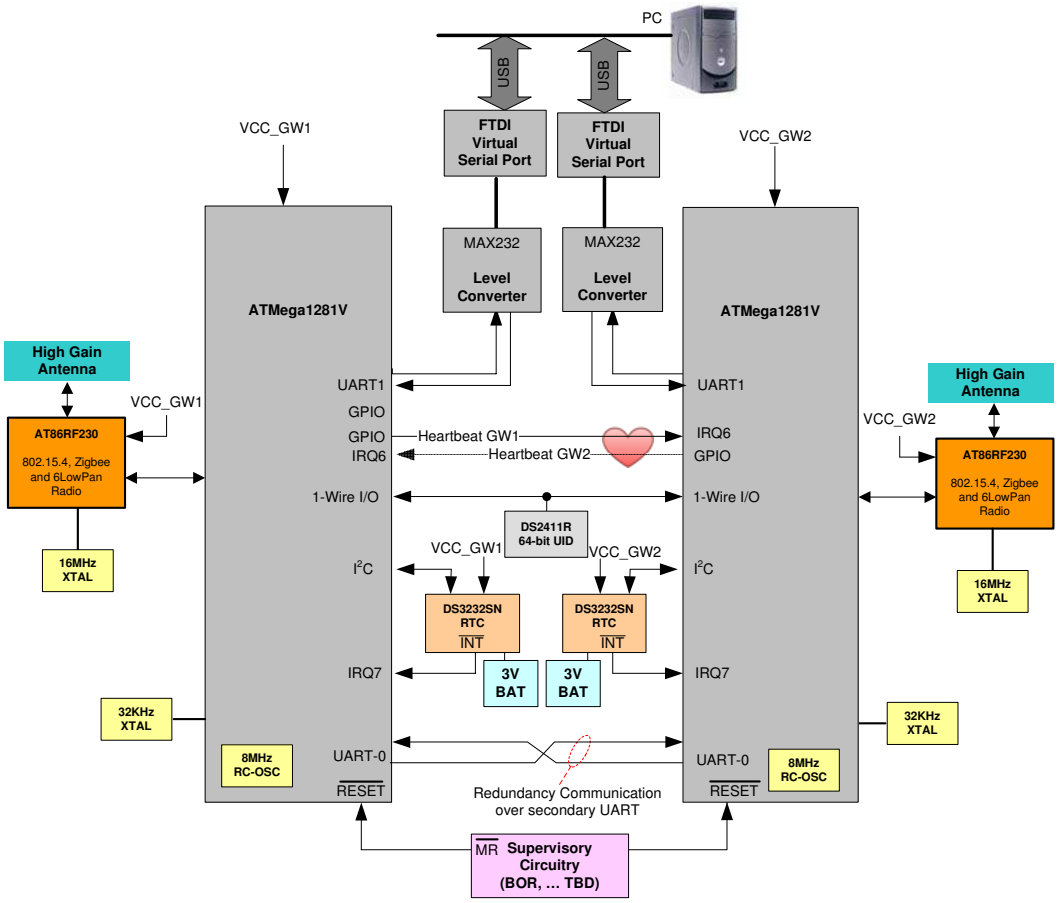


Figure 5.6: The envisioned ZigBit based dual redundant gateway hardware architecture

is master-/slave based. The source code for the gateway can be seen back in Appendix A. Appendix C lists a complete configuration settings file used by the BitCloud environment.

Clock sources in the ZigBit devices

The ZigBit device has many clock sources in use for operating both the 802.15.4 PHY chip (AT86RF230) and the ATMEGA1281V microcontroller. In the test setup, the BitCloud stack and operating system clock was set to 8MHz, which comes from an onboard internal RC-oscillator. By using clock division, the SPI clock is running at 4MHz. The start-up time therefore is 6 CK + 65 ms. Also there is an 32KHz crystal attached to the MCU, which is used for application timers in BitCloud.
The AT86RF230 has a dedicated 16MHz external oscillator and uses SPI for communication to the microcontroller.

Heart beat as node fail detection

The standby gateway uses the heartbeat pulse to monitor the health of the active gateway. In the scheme above, simple heartbeat pulses of 10ms have been used to detect a failure of the master device.
Connecting this heartbeat directly to an interrupt pin on the microcontroller and the gateway will be able to detect a malfunction of the gateway with simple software counters. This also allows for *hardware based* heartbeats like PWM pulses. These heartbeat links could also be redundant on itself, however due to limited available external interrupt pins on the Atmega1281V this is not implemented, instead a dedicated serial interface (UART0) has been provided to support redundancy communication. As this serial interface currently uses only network-state health information and might be used for check pointing, additional heartbeats as a backup for software fault detection can be implemented, but is left open. Together with PWM hardware based error detection this scheme shall be a fully dissimilar system in error detecting. Figure 5.7 displays the

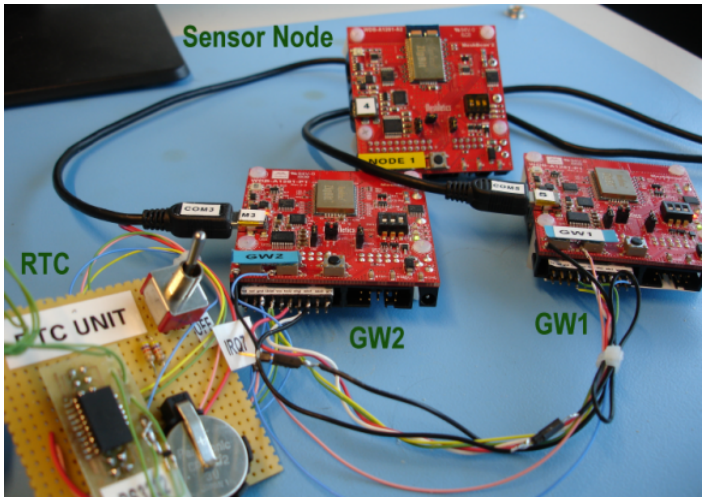


Figure 5.7: The ZigBit hardware lab prototype setup

hardware in the laboratory setup, including the interface between the two gateways and RTC device.

Check pointing operation

During operation of the gateways, a check pointing algorithm periodically stores current network parameter data, such as current operating channel, the nodes connected to the network (e.g. child nodes), etc. This can be done once every minute, day or just whenever any change of the network-state is encountered. A check point will set the flag "`restore_awaiting`" to true, to identify that a restore of network parameters is possible for the hot gateway.

Alternative use of checkpoint operations

Since there was limited time to complete a parser that serializes chunks of the data and implementation of restore at the other gateway, an alternative checkpoint feature is used, the "Persistence Data Server" (see also section 5.1.1). Complete PDS sourcecode was given by Atmel.
In Zigbee, it is obligatory to store data on the nodes by means of EEPROM, Flash, other persistence data. This is called Persistent Data (see Zigbee datasheets, [2]). Since the PDS can be activated manually, it then automatically stores frequently used parameters (e.g. neighbour tables) to EEPROM every 5 minutes. The latest BitCloud (10/2010, version 1.10.0) supports check pointing on an event-basis and can be manually disabled or enabled during runtime.

Several parameters are saved in EEPROM if `CS_POWER_FAILURE` is enabled, and it also stores all network parameters and restores them back when you try to start network after reset. If `CS_POWER_FAILURE` is enabled on the gateway, network availability increases as node information (i.e. `NEIB_TABLE`, see listing 5.1) are located in the gateway after reset. Tests shown that a minimum of 3 packets are lost (at optimal conditions), because it takes about 340–500ms[2] to start the network, depending whether `CS_POWER_FAILURE` is enabled or disabled.

```
#if defined(_POWER_FAILURE_)
  uint8_t CS_NEIB_TABLE_ID[CS_NEIB_TABLE_SIZE * sizeof(Neib_t)];
  ...
#endif //_POWER_FAILURE_
```

Listing 5.1: Detail of C-Code from pdsWriteData.c containing check pointing (connected) neighboring nodes to EEPROM

5.3 Experimental Results

To measure performance of the wireless redundancy implementation, an number of tests were conducted and different switchover times where gained. Additionally, the goal was to measure the impact on the robustness of the wireless network stack and its connected sensor nodes. An example debug listing (with and without check pointing) from the gateway nodes is shown in figure 5.8.

Each gateway lists all incoming data, by printing it to the UART and was saved for further analysis. The data consists of a command or event (denoted by the dollar sign) from the sensor node, and its timestamp, from the RTC. The second gateway (denoted 'Gateway 2') was already initialized to and acts as the fail-standby gateway. After power has been cut on gateway 1, the second gateway stack is started and after a while it gets its timestamp from the RTC.

Figure 5.9 shows the results of the switchover time, that is the time between the last seen data from the node and the newly seen data from the node at the second gateway (i.e. standby gateway). The left side represents the test results without check pointing

[2]This is empirically observed

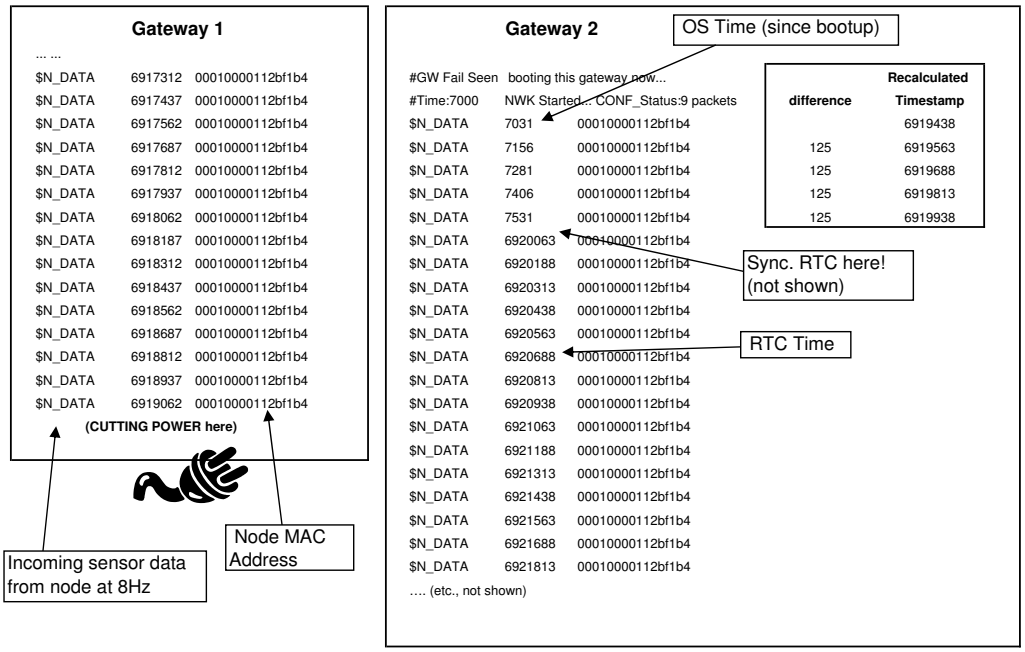


Figure 5.8: Debug listing of gateway redundancy, with check pointing enabled (enabling CS_POWER_FAILURE)

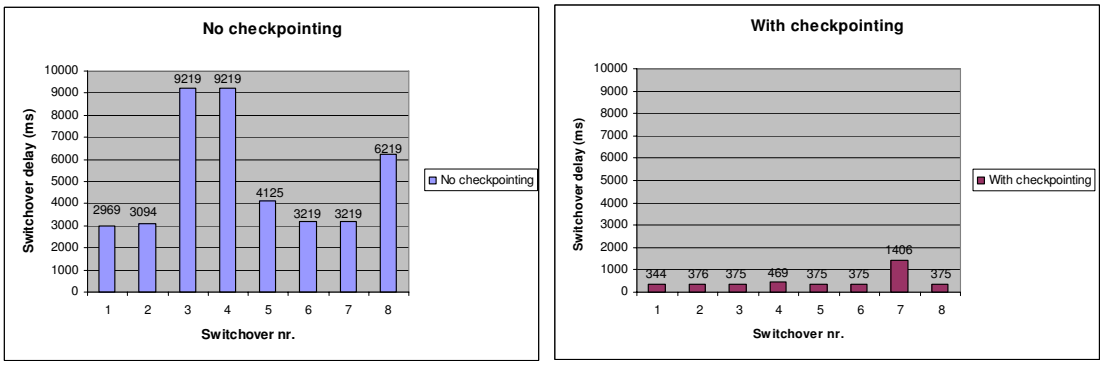


Figure 5.9: Gateway redundancy switchover times without check pointing (left) and with check pointing (right) enabled.

and right with check pointing enabled. A number of 16 tests where performed to have a general spreading of possible switchover times. The settings which were used by the BitCloud stack and node are listed in table 5.1. During all tests, the `APS_ACK` (application layer acknowledge) was disabled because of additional latency (the application ACK can only be received the next time the end device polls) and transmission bandwidth costs. The MAC layer, which provides per-hop acknowledgments, is reliable enough for single-hop networks because additional ACKs for this type of network are considered unnecessary.

Setup Parameters	No check pointing	With check pointing
CS_POWER_FAILURE	false	true
CS_AUTO_POLL	true	
CS_INDIRECT_POLL_RATE	1000ms	
CS_NEIB_TABLE_SIZE	1	
Node send rate	8Hz	
APS_ACK	Disabled	
Stack op. speed	8MHz	

Table 5.1: BitCloud settings during redundancy tests

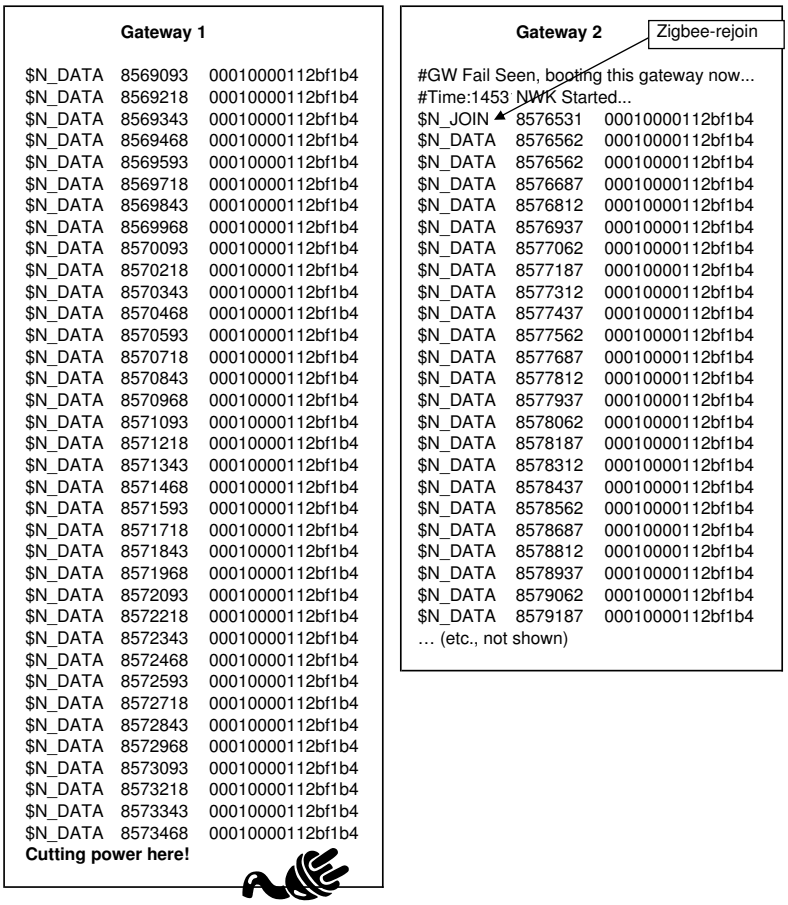


Figure 5.10: Debug listing of gateway redundancy, with no check pointing (using ZigBee-switchovers)

5.3.1 Comparing performance

From Figure 5.9 it can be made clear that switchover time without check pointing depends on the following parameters:

- Node indirect poll rates: the poll rates determines that the node stops sending data and rejoins with the parent (gateway). In our setup a default poll rate of 1000ms was used. At the gateway, the absence of (at least) 3 polls from a node will result in an amputated node from the neighboar list.
- The number of nodes connected: as the nodes require realignment to the new gateway, additional bandwidth (from the gateway to the node and node to gateway) is required. Since there is already limited bandwidth in 802.15.4 available, if multiple nodes start polling for a network at one instance, the network is flooded quickly and therefore network formation for a quite large number of nodes may take many minutes.

5.3.2 Encountered problems

During the implementation, a number of problems occurred that where already foreseen because of the use of Zigbee. The first problem found was that packet arrival latencies are not exactly deterministic, due to the CSMA/CA algorithm. It was observed that occasional late sensor-data arrival rates of 30ms are common, when multiple sensor nodes are used in the network at high transmission dutycycles. This can be avoided by implementing time-synchronized protocols such as WirelessHART or the usage of channel agility (implemented in Zigbee PRO), or the emerging ISA100.11a and other standards families.

Another problem was related to power consumption of the end-devices. After a poll request, end devices stay a long time (up to 120ms) in wake time, to receive back the frame from the coordinators. The latest "Atmel" based ZigBits, based on the AT86RF230B RF chip claimed this problem has been solved, but was not tested.

During tests, many printfs where used and it was clear that execution of these resource hogs where problematic. Additionally, there was a lot of Zigbee-communication (e.g. orphaning messages), running transparent to the user code. Another way to see what was going over the network is to use a wireless 802.15.4 sniffer and can save a lot of extra debugging.

5.3.3 Further improvement

It is noted that even with check pointing enabled, the stack has to be started before the alignment of the neighbour nodes is complete, as a consequence there is always some "blackout" delay. By a number of provisions and finetuning this delay can be decreased:

- **Increasing heartbeat frequencies**
  and therefore decreasing latency that error is detected (now the timeout is set at 3×80ms and heartbeats are pulsed out every 50ms). By this way you are increasing fault-isolation and detection speed.
- **Using faster hardware**
  The 32-bit Atmel based ARM7 Microcontroller (e.g. the AT91SAM7S256) could increase stack execution speed (by utilizing a clock frequency of 48MHz, a speedup of at least 6 times can be achieved compared to the current microcontroller).

– **Support for quick enable/disable functionality in the stack**
Some stacks offer the possibility of an periodic disabling and re-enabling of the stack. The Microchip Zigbee stack encorporates for example `APLDisable()` and `APLEnable()`, however these commands are only intended for (sleepy) end-devices and not coordinators which are supposed to be always active.

5.3.4 Discussion

One should ask the question, when to check point and if check pointing might have less benefits than the regular (over-the-air) network based switchovers. It is shown that with simple check pointing, fast switchovers can be guaranteed, thus allowing the network to be self-healing, transparently to the nodes. While it is shown that switchovers are also practical at Zigbee layers, there is additional latency and infrastructure communication. Distributed check pointing can result in major advantages in reducing these delays and bandwidth, however additional (unwanted) side-effects may cause other faults. One example is *checkpoint contamination*: in the current experiments, every 5 minutes a check point is made. It could be, for whatever reason, that the checkpoints become contaminated (e.g. erroneous MAC addresses, etc.). Then, there is a possibility this may propagate to the standby gateway as well and in the end could destabilize the entire gateway operation by iterative checkpoints. Ways to circumvent these problems are possible, yet again implies complex implementations. One solution might be the periodic "flushing" of the check point and begin a clean rescan of the neighbouring nodes by using stack operations again. This will result in a fresh, clean and recent checkpoint.

The latest BitCloud (version 1.10.0) supports a minor improvement in the PDS: it supports manual check pointing to EEPROM (or by events).

5.4 Conclusions

Throughout the design phase, it was concluded that the gateway is the more complex device of the network, as this device is in charge of commissioning the network, delegating commands and should be implemented as single-point fail tolerant. Two switchover mechanisms where tested out and it can be said that fault tolerance is readily graduating inside COTS components too, and can be utilized effectively for aerospace systems. The proposed gateway architecture (shown in figure 5.11) illustrates the dependencies inside the gateway. Still not everything is implemented yet (missing are the specifications of interaction patterns, a working TEDS implementation and timestamp formatting combined with synchronization methods).
Sensor nodes should have a wireless heartbeat as well to indicate correct node operation (like the poll rates in Zigbee). Simple timesynchronization on gateways is easy to implement however (wired) inter-gateway links are still required for synchronizing all gateway clock sources to a single OBC. This should be addressed as an additional command from the OBC.

Also, Failure Detection Isolation and Recovery (FDIR) algorithms at the OBC which use the housekeeping information should be able to process diffracted housekeeping sensor data in time (as the wireless network is consuming data in a time-dispersed way).

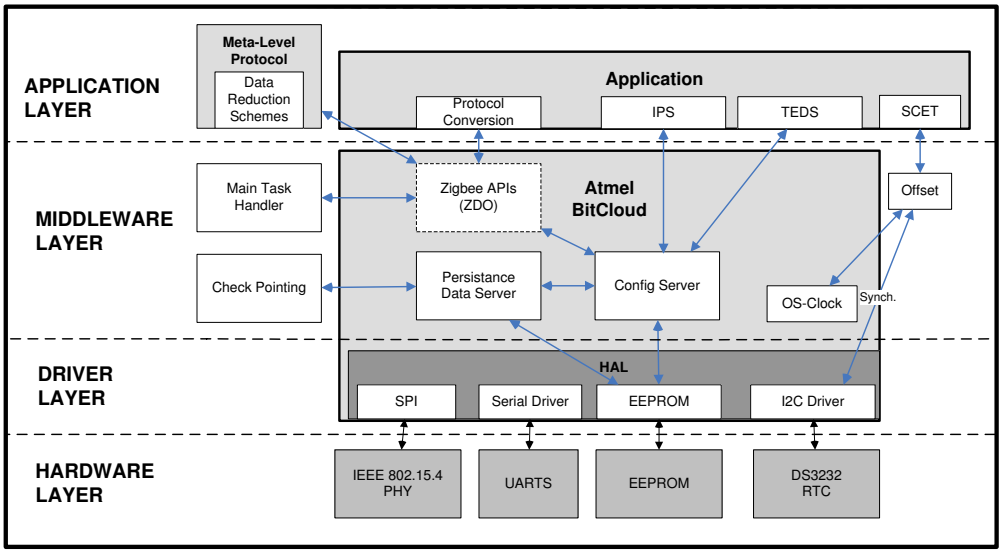


Figure 5.11: The envisioned gateway architecture for the intra-spacecraft wireless sensor network

The overall gateway system architecture can be mapped into four distinct layers: application layer (high-level functionality and user interface), middleware layer (e.g. where the communication protocol resides and check pointing services), driver layer (i.e. handled by the operating system) and hardware layer.
The architecture of the proposed platform (depicted in figure 5.11) is arisen from this thesis and consists of:

- Atmels Bitcloud Suite (Zigbee PRO) used as a demonstrator protocol.
- An IEEE 802.15.4 standards-based RF Device.
- DS3232 Real Time Clock with I2C interface
- Hardware Abstraction Layer (HAL), hardware driver source files given by Atmel
- Config Server (CS): handles parameters for the ZigBee stack
- Persistence Data Server (PDS): handles check pointing to EEPROM for the Zigbee stack.
- Data Reduction Schemes (DRS): reduces network traffic generated by the nodes.
- Interaction Pattern Setups: (IPS): Node information about the way it interacts with the gateway.
- Transducer Electronic Datasheets (TEDS): basic I2C based information sheets for specific sensors.
- Spacecraft Elapsed Time (SCET): ESA compliant time stamps.

6 Conclusions & Future Work

This chapter discusses the conclusions and relevant future work for implementation and roll out of a test-bed of wireless sensor networks for a spacecraft. Further improvements are highlighted which require further attention and middleware to be implemented inside the sensor network. Recommendations and directions are given for an increased reliability for the sensor network components in section 6.2, such as radiation tolerance and the further exploitation of wireless cross-strapped links. The use of data-reduction schemes reduces network bandwidth, but can also be beneficial for lowering power consumption of the sensor nodes. One example scheme is highlighted and discussed in section 6.2.3. The thesis is concluded with a conceptual future spacecraft wireless network for flight– and AIT operations.

6.1 Conclusions

In this thesis, a platform is proposed that can be used as testbed of low data-rate wireless sensors inside a spacecraft. Initial adoption of wireless devices during in-orbit missions for spacecrafts can be made possible by development, production, demonstration and utilization of ground-based wireless equipment for use during AIT operations. Not only will the result be faster integration and checkout, but also significantly simplified testing due to improved visibility of wireless hardware.

This thesis addressed the use of standards-based systems, as a baseline to enable a fault-tolerant wireless sensor network infrastructure for common spacecraft sensors. An evaluation of the sensors that are applicable and most applicable to be wireless has been made, and a survey on applicable COTS based sensor network platforms are compared. The use of standards-based protocols and platforms can significantly speed up development and demonstration of real applications in the aerospace domain. Spacecraft housekeeping sensors are considered the most applicable sensors near-term adoption for wireless applications, however some sensors can benefit more from wireless than others. Examples are those sensors located externally on the spacecraft such as on deployables.

After the study of the concept of modularity, we compared standards and recent research and adapted parts to a template which allows for a self-describing wireless modular sensor node. In wireless networks, device enumeration is already present and increased self-programming capable systems will be more fault tolerant. The inclusion of TEDS (Transducer Electronic Datasheets) and other various higher-level autonomy functionality will eventually make the sensor network more fault-tolerant and will make engineering (during AIT phases) faster, better, easier and at its highest extend, a wireless sensor network does no longer need maintenance, configuration or whatever manual intervention at all.

In this thesis, fault-tolerance methods of wireless COTS hardware and software is examined. However, wireless (proximity) technology adoption for space applications is still under discussion and there are many issues and disadvantages when choosing wireless. Further analysis concludes that centralized wireless devices, such as access points, gateways and routers, are a source of a single-point of failure and has effects on the entire network. These *"choke-points"* in the network, where wireless sensor control and data is collected, relayed or processed, are the most vulnerable components in the system.
As a workaround, a multi-gateway topology is proposed and outlined, which is based on multiple locations of sensors that will allow a many sensors to be used in dense areas and should result in an efficient utilization of the available wireless bandwidth. In this context, we have been using the IEEE 802.15.4 and Zigbee communication protocols for WSNs. Zigbee supports several network topologies (star, mesh and cluster-tree), and a general reliable topology is proposed for use in spacecraft which make use of multiple clusters of gateways.

The presented and tested universal gateway redundancy scheme, which provides fast recovery in case of a gateway failure of the sensor network, illustrate that wireless offers a simple, yet powerful solution for redundant operations with (almost) minimal design effort. Wireless is cross-strapped by itself, and further algorithms and methods should make more advantage out of this (see section 6.2.2), not only to gain fault tolerance at the gateway level, but to increase the overall health of the network.

6.1.1 Concluding remarks

IEEE 802.15.4 wireless links tend to suffer from limited bandwidth compared to its higher bandwidth links. It turned out that network saturation is playing roles at higher frequencies than 1Hz. Zigbee is meant for extremely low data rates (e.g. 30 second update rates) and more spatial distributed networks, therefore saturation of the network is not an issue. Intra-spacecraft applications, on the other hand, require densely populated sensors spread over the instruments. It is therefore assumed is that the high (spatial) density of the sensor nodes inside the central satellite is a significant problem for a wireless network. Emerging standards like WirelessHART, ISA100 and others tackle these problems only partially by fast diversity switching mechanisms and time synchronization, yet limited bandwidth is available at the same time. The actual application determines whether the level of link reliability is enough, e.g. if time to time failure is allowed, a best-effort based system like Zigbee can be applied and for high-criticality sensing, such as attitude sensors, highly scheduled systems with strict latency guarantees are best suitable.
Still, even with the emerging standards, wireless communication success is a probability issue, as even the best protocols and RF links make mistakes. One should focus on the advantages wireless offers, and use wireless as the application suits for it accordingly.

The current review of selected COTS devices shows that there is a growing trend in consumer devices in more and more autonomous systems. Intelligent software and high-level self-describing languages (such as XML) allow for plug and play systems. Self-healing, self configuring and other "self-x" techniques are applied more and more frequently and the underlying hardware adapts to this by increasing demands in processing speeds and

flash and RAM resources of embedded devices. Over and over, data reduction schemes are the key in operating and managing future sensor networks. Yet, care must be taken to not diminish or deteriorate the amount or quality of information contained in the returned data.

Still, there are many challenges in the applications of wireless sensor networks which must be investigated before wireless sensor networks can be applied in aerospace applications. Power consumption, volume, sensor bandwidth, and mass must be reduced. Issues such as electromagnetic interference (EMI), ionizing radiation, vibration and extreme harsh environments must be addressed. Despite all the challenges, wireless technology offers many great benefits that cannot be ignored.

6.2 Future Work

As the platform is considered only a first step towards a flight-ready system, it is still good to look at some future implementation aspects, like on future generations of satellite systems. Future work for a successful manufacturing and deployment of the wireless sensor nodes includes:

- **Radiation tolerant** design of key components of the sensor nodes and gateways;
- Adoption of reliable **data-reduction techniques** to enable lower power consumption and better network utilization;
- The exploitation of **wireless** cross strapped links at network level to increase reliability and availability of the network;
- The survey and empirical evaluation of the **interference** of other wireless systems and **interferers** of other spacecraft components (e.g. obstacles).

Basic guidelines and principles for the first three goals are discussed in the following sub sections. The latter one is left out for future work as it requires in-depth knowledge of the physical layers of wireless components and details of its operation (by protocol).

6.2.1 Radiation tolerant implementation

The ability to withstand radiation could help the sensor nodes (and gateways) for improved survivability, and endured lifetime in long space missions. Vulnerable parts (such as the protocol stack and the OS) should be made radiation tolerant. First of all, there is no single IEEE802.15.4 radiation tolerant version available (to date). Thus, for now it is best to use ultra-low power COTS components here and have a way of general fault tolerance mechanisms such that the modules degrade gracefully. The best way for a new sensor node design is to use a combination of (aerospace qualified) high-power components and optimize for power consumption here with the use of ultra-low-power COTS devices.

6.2.2 Exploiting cross strapped links

In traditional wired systems, cross strapping is commonly used in multi-string redundant systems, such as with flight computers (see section 2.9.1). In case of failure of one of the computers, another computer takes over processing. This architecture requires multiple

(cross-strapped) wires in between the computers, and is most of the time fairly expensive. Cross strapped links come almost for 'free' with wireless: wireless devices can see and communicate with any neighbouring devices they like by means of channel diversity, frequency diversity, etc. The problem however is that wireless communications cannot transmit concurrently without interference while wired communications can do this. RF devices have to send their values in a simultaneous way and on a non-interference basis. This communications bottlenecks limits the sensor node sample rates and overall bandwidth of the network.

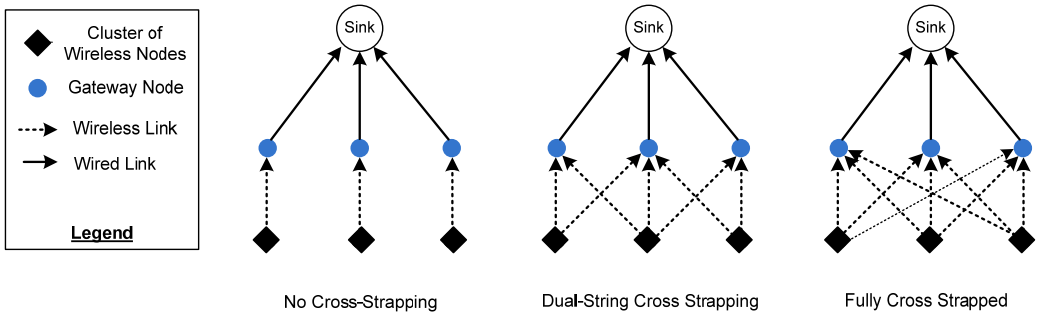


Figure 6.1: Cross strapped wireless sensor networks

The solutions for this problem might be found in multiple domains. A common solution is to use data-reduction at the sensor nodes to reduce overall communication volume (or bandwidth) and reduce the loads at the gateways. If correctly used, the data reduction should not result in significant lost data. Another solution might be the use of inter-gateway load balancing, and requires that the gateways should have an (additional) communications link between them.

To conclude, dual string-cross strapping might be a feasible alternative for dual-redundant gateway architectures however has some drawbacks. The main advantages are that this is compatible with the location-centric topology, as neighboring gateways can be in the vicinity of the neighboring sensor nodes.

Inter-gateway load balancing

When the network is setup using multiple gateways (which is always the case when many sensors are used at high transmission intervals), the gateways should be able to communicate with each other in order to have a more reliable network (i.e. a more global network awareness). This communication allows for the assessment of the entire network state and therefore allows for some balancing and tuning of the network to have an overall better operating network in the end. A following proposal is given that concludes that at leas one-inter-gateway communication link is required to allow for a robust network state. Since a wireless inter-gateway link will not be efficient for the communication bandwidth for the sensors, an alternative wired link might be required and might be controlled by the sink to have control of the entire network state.

An example with the dual-string cross strapping, only 50% load of the should be allocated to the gateways in order to cope with a single gateway failure, and serve two of

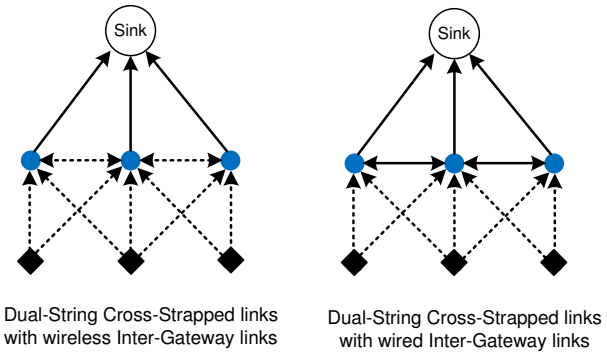


Figure 6.2: Cross strapped wireless sensors networks with inter-gateway links

the sensor clusters at the same time. The alternative of full cross strapped gateways is also an option, however this requires that the gateways should balance out the number of connected sensor nodes for optimal operation.

A way to improve the switchover delays again is to implement distributed recovery blocks, and the gateways should selectively accept the blocks, like for example 50% of the neighbour lists are used in a dual-string cross strapped system. Another option is to use (wired or wireless) heartbeat pulses to indicate a gateway failure, and then let the secondary gateway take over the part of the recovery block from the failed gateway. Assumed is that all gateways should again have full access to all (next nearest, i.e. in 'RF sight') nodes.

6.2.3 Data reduction schemes

As shown by many problems during development, data reductions schemes can be used to reduce communication, and can even be useful for the low-rate 802.15.4 networks. A sample approach used in [4] can be used to have a reduction of about 90% in communication (see figure 6.3).

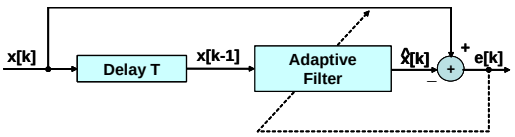


Figure 6.3: An adaptive predictor used for data reduction [4]

Using adaptive filtering, this algorithm can be used to have about 90% less communication, with a corresponding minimum accuracy of 0.5C with a temperature sensor on a sensor mote. With this configuration it is allowed to have less battery capacity for a longer period of time that the system operates. However full power consumption then is determined by the time difference the sensor is operating at. At low rate frequencies, depending on environment, the sensor node will operate longer while at higher frequency changes of the sensor readings, more transmissions are required to have a more accurate

reading. Hence the name of the filter is an adaptive one. Adaptive filters are more used in non-stationary environments and are typically used in mobile sensor networks. To conclude, the adoption of this kind of filtering may depend on the environment model and stationarity of the sensors.

Implementation of this filtering requires two algorithms implemented on both the sensor node and gateway and coordination have to be set on both ends to assure correct data reduction reconstruction at the gateway. Sampling has to be performed still at desired frequencies, so the node has to wakeup sense, perform data-reduction and store to memory or send over the air. As Zigbit gateways are always powering up the stack, alternatives must be found such that we can only sample with limited currents and not wake up the stack. Also if transmission is required after sensing, additional wait time is required to boot the stack after all. So power reduction might not be exactly 90%, due to these additional routines and delays for the microcontroller. It is generally said that computation is 'free' and communication is costly so only experiments with new hardware will reveal true power consumption at the end.

6.3 Expected future evolutions

A future wireless architecture for spacecrafts should be split by the level of criticality of the sensors. This split is considered purely hypothetical in point of view. In this section a prediction for a future evolution is made.

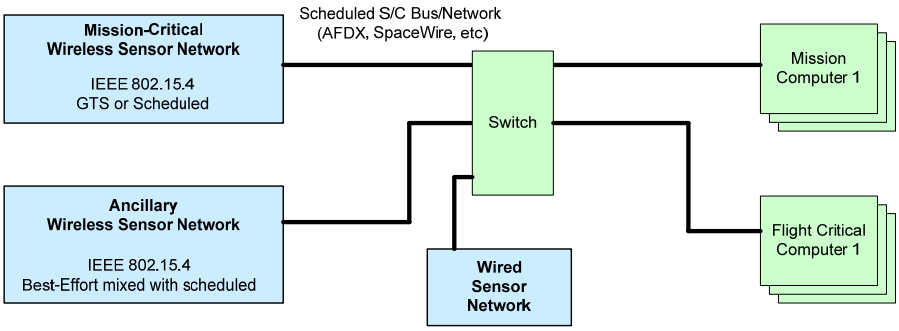


Figure 6.4: Intra-spacecraft mission computer network with low-data rate wireless sensor/actuator networks

There are basically two different wireless domains: mission critical sensor networks and ancillary sensor networks. Each of the two can handle low-rate sensors. While the mission-critical sensor network is largely deterministic and preconfigured, the ancillary sensor network is used to be adaptable and quickly reconfigurable, and transmission should be based on *best-effort* or other more deterministic services. Mainly attitude sensors are used for mission-critical sensor networks and (engineering) housekeeping sensors and sensors for mission support for the ancillary sensor network. During AIT it is considered that adding sensors is required for flight tests. It is considered that smart wireless sensors with built-in electronic datasheets (such as TEDS and EEDS) should be supported in this kind of networks. Ground support interoperable wireless test equipment

for integration-test personnel enables fast integration and the use of a "*wireless toolkit*" should provide easy access to the spacecraft structure during any test bench. Figures 6.4 and 6.5 show an example of this expected future wireless sensor network for spacecrafts, based on IEEE 802.15.4 as a low datarate sensor network. Scheduled transmissions are recommended for time sensitive applications and where high reliability and criticality is required such as attitude sensors or control (actuators).

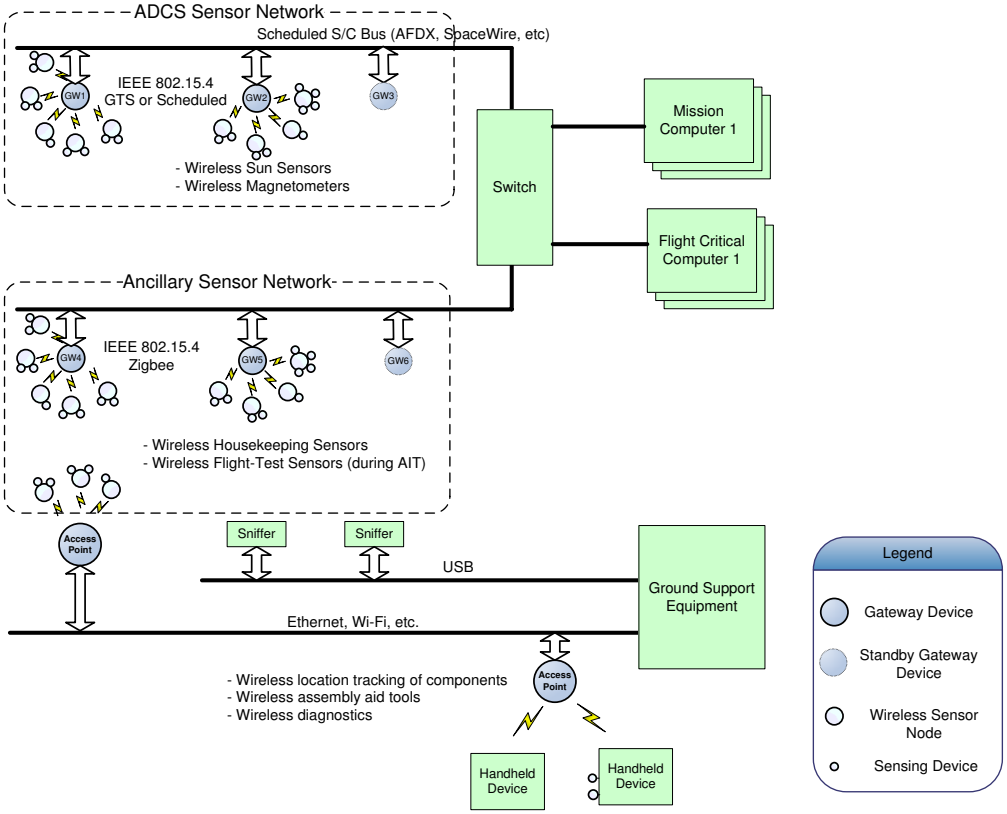


Figure 6.5: Detailed intra-spacecraft wireless network with example wireless sensors and wireless ground support equipment

Future generations of on-board wireless sensor network systems (e.g. ISA100) allow for a robust, interoperable coexistence with other wireless (sensor) networks, where a mixed case of best-effort and scheduled transmissions are used for effective data delivery. The advent of other new wireless technologies, such as the use of Ultra-Wide Band (UWB) in the 802.15.4a standard will have an impact on current narrowband wireless problems related to interference as with UWB this interference is below the noise level of the spacecraft.

In pararell, the ground support equipment will benefit the most from other wireless technology such as handhelds (e.g. iPads) and tools that will increase overall productivity during assembly and integration. Examples are real time monitoring of test equipment (or other components) inside the spacecraft [41], electronic inspection checklisting, etc.

The ultimate goal will be a highly integrated portable sensing device capable of scanning every phenomena, wirelessly, and incorporates fully functional processing, assessment and awareness of the environment. Such a device, called a tricorder in *"Star Trek"* allows for a portable read out of sensors used in spacecraft (engineering), environmental, biological and medical areas. The design of this scanner is highly integrated, multi-sensor and handheld readout with a single interface. While this device is science fiction, it allows us to open up ideas about future generation spacecraft interfacing, sensing and processing which sounds promising [19].

Bibliography

[1] ÅAC Microtec AB., *The NanoRTU*, Sweden, 2010.

[2] Z.B. Alliance, *Zigbee specification*, ZigBee Document 053474r06, Version (2005), 1.

[3] R. Amini, G. Gaydadjiev, and E. Gill, *Smart Power Management for an Onboard Wireless Sensors and Actuators Network*, 2009.

[4] R. Amini, E. Gill, and G. Gaydadjiev, *The Challenges of Intra-Spacecraft Wireless Data Interfacing*, 57th International Astronautical Congress, September, Citeseer, 2007, pp. 24–28.

[5] A. Arts, *The ESA Herschel/Planck Satellite*, Planck album (http://www.satellite-planck.it), 2009, p. photo album (2009).

[6] D. Atkinson, G. Swanson, and J. Schlee, *Implementation of Wireless TPS Sensors*, Georgia Institute of Technology (2008), Sixth International Planetary Probe Workshop, (Contributor: University of Idaho. Dept. of Electrical and Computer Engineering).

[7] D.J. Barnhart, T. Vladimirova, and M.N. Sweeting, *Design of self-powered wireless system-on-a-chip sensor nodes for hostile environments*, Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on, IEEE, 2008, pp. 824–827.

[8] J. Burch, *Proposed IEEE 1451.0 to P1451.5 Interface*, 2004.

[9] CCSDS, *CCSDS REPORT CONCERNING INTEROPERABLE WIRELESS NETWORK COMMUNICATIONS*, CCSDS 880.0-G-0.195 (2010), 149 pages.

[10] F-K. Chang, *SHM Design for Space Vehicles*, Dept. of Aeronautics and Astronautics) (2004), Stanford University.

[11] L. Choong, *Multi-channel IEEE 802.15.4 Packet Capture using Software Defined Radio*, Tech. report, Technical report, 2009.

[12] D.S. Clarke, M.T. Hicks, A.M. Fitzgerald, J.J. Suchman, R. Twiggs, TW Kenny, and J. Randolf, *Picosat Free Flying Magnetometer Experiment*, Proceedings of the 10th Annual AIAA/USU Small Satellite Conference, 1996.

[13] RdF Corp., *"These low cost, sealed platinum surface RTDs are the worlds toughest." – SURFACE PLATINUM RTDs*, http://www.rdfcorp.com/products/capsule/r-scap_pf.html, 2010.

[14] D. Kaufman, *Flight Force Measurements (FFMs) of the Gamma-Ray Large Area Space Telescope (GLAST) / Delta II Flight*, NASA Engineering and Safety Center Technical Assessment Report, NESC (NASA Engineering and Safety Center) Loads and Dynamics Deputy, 2009.

[15] A. Davis, *The ACE science center*, http://www.srl.caltech.edu/ACE/ASC/, California Institute of Technology, 1998.

[16] CW. de Boom, JAP. Leijtens, and N. van der Heiden, *Micro digital sun sensor: System in a package*, MEMS, NANO and Smart Systems, 2004. ICMENS 2004. Proceedings. 2004 International Conference on, IEEE, 2005, pp. 322–328.

[17] A.D. Dominguez-Garcia, G.Z. Hanuschak, S.R. Hall, and E.F. Crawley, *A Comparison of GN&C Architectural Approaches for Robotic and Human-Rated Spacecraft*, Massachusetts Institute of Technology (2007), 1–15.

[18] S. Dong, K. Allen, P. Bauer, B. Bethke, A. Brzezinski, T. Coffee, R.D. Chambers, M. Flores, A. Gallagher-Rodgers, J. Head, et al., *Self-assembling wireless autonomously reconfigurable module design concept*, Acta Astronautica **62** (2008), no. 2-3, 246–256.

[19] G.A. Dorais and Y. Gawdiak, *The personal satellite assistant: an internal spacecraft autonomous mobile monitor*, Aerospace Conference, 2003. Proceedings. 2003 IEEE, vol. 1, IEEE, 2005, pp. 1–348.

[20] J-F. Dufour, *Flight-tested technologies, Taking the wires out of satellites*, ESA (2009), http://www.esa.int/esaMI/Technology/SEMF5G1P0WF_2.html.

[21] A. Anders EnOcean GmbH, *RF Sensor Transmitter Module, STM 100 – User Manual V1.51*, 2006.

[22] L. Ferrigno, S. Marano, V. Paciello, and A. Pietrosanto, *Balancing computational and transmission power consumption in wireless image sensor networks*, Virtual Environments, Human-Computer Interfaces and Measurement Systems, 2005. VECIMS 2005. Proceedings of the 2005 IEEE International Conference on, IEEE, 2006, p. 6.

[23] G. Artaud, P. Plancke, G. Furano, R. Magness, C. Plummer, *IEEE 1451: transducer networking*, DASIA 2004, Conference, Nice, France (2004), ESA–ESTEC.

[24] G. W. Hunter, G. A.Ruff, and J. C. Xu and G.C.Steele, *Multiparameter Fire-Detection System Miniaturized and Tested for Possible Use on Crew Exploration Vehicle*, Constellation Project Office, NASA, 2008.

[25] D. Gao, D. Liu, Y. Feng, Q. An, and F. Yu, *A Robust Image Transmission Scheme for Wireless Channels Based on Compressive Sensing*, Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence (2010), 334–341.

[26] M. Guelman et al., *The Gurwin-Techsat Microsatellite: Six Years Successful Operation in Space*, Small Satellites, Systems and Services, vol. 571, 2004, p. 62.

[27] GW Hunter, LG Oberle, G. Baakalini, J. Perotti, N.J.F.K.S. Center, K.S. Center, and FLT Hong, *Intelligent Sensor Systems for Integrated System Health Management in Exploration Applications*, electrochem.org, First International Forum on

Integrated System Health Engineering and Management in Aerospace, Napa, CA, 2005.

[28] GW Huntera, JC Xu, BJ Wardb, DB Makelb, P. Duttac, CC Liud, and RA Dweike, *Smart Sensor Systems for Aerospace and Biomedical Applications*, NASA John F. Kennedy Space Center, NASA Glenn Research Center, ECS, 2009.

[29] IEEE, *IEEE Standard for a Smart Transducer Interface for Sensors and Actuators - Common Functions, Communication Protocols, and Transducer Electronic Data Sheet (TEDS) Formats (IEEE1451.0.2007)*, IEEE Instrumentation and Measurement Society (2007), ieee.org.

[30] Honeywell International Inc., *"Honeywell Sensotec – TEDS, plug and play sensor configuration*, www.content.honeywell.com/sensing/sensotec, 2008.

[31] Microchip Technology Inc., *MiWi Wireless Networking Protocol Stack*, 2007.

[32] National Instruments Inc., *Sensors Plug&Play The New Standard for Automated Sensor Measurements, IEEE 1451.4*, 2010.

[33] F. Institutionen, *The Anticoincidence System of the PAMELA Satellite Experiment*, AIAA (2005), 66.

[34] T. Instruments, *SimpliciTI Compliant Protocol Stack*, 2009.

[35] J.R. Irons and J.G. Masek, *Requirements for a Landsat data continuity mission*, Photogrammetric engineering and remote sensing **72** (2006), no. 10, 1102.

[36] J. O'Keeffe, *Wireless Vehicle Health Monitoring*, 2007, www.jtokeeffe.com.

[37] J. Perottid, A. Lucena, P. Medelius, C. Mata, A. Eckhoff, and N. Blalock, *Modular Wireless Data-Acquisition and Control System*, John F. Kennedy Space Center, KSC-12386, 2004.

[38] Jennic, *Calculating JN5121 Power Consumption.* , Product Brief (2008), JN–AN–1001.

[39] J. Jeong, S. Kim, and A. Broad, *Network reprogramming. TinyOS documentation. 2003*, 2003.

[40] M. Kohvakka, J. Suhonen, M. Kuorilehto, M. Hännikäinen, and D. Hämäläinen, *Network signaling channel for improving ZigBee performance in dynamic cluster-tree networks*, EURASIP Journal on Wireless Communications and Networking **2008** (2008), 1–15.

[41] S. Koris, *Northrop Grumman to develop wireless satellite avionics data bus technology*, 2009.

[42] H.J. Kramer, *HummerSat-1*, "Observation of the Earth and Its Environment: Survey of Missions and Sensors", http://directory.eoportal.org/presentations/6146/15541.html, 2008, pp. 1–9.

[43] W5DID L. McFadin, *SuitSat 2*, 2007.

[44] X. Liang and I. Balasingham, *Performance analysis of the IEEE 802.15.4 based ECG monitoring network*, Proc. The Seventh IASTED International Conferences on Wireless and Optical Communications (WOC07) (2007), 99–104.

[45] Libelium, *Waspmote Power Programming Guide*, libelium.com, Libelium Comunicaciones Distribuidas S.L., 2009.

[46] L. Lin, N.B. Shroff, and R. Srikant, *Asymptotically optimal energy-aware routing for multihop wireless networks with renewable energy sources*, Networking, IEEE/ACM Transactions on **15** (2007), no. 5, 1021–1034.

[47] K. Lo, *ULSSD BLUEsat Project Annual Report*, 2002.

[48] D.A. Lohr, L.J. Zanetti, B.J. Anderson, T.A. Potemra, and M.H. Acuña, *The NEAR Magnetic Field Instrument*, Johns Hopkins APL technical digest **19** (1998), no. 2, 137.

[49] F. Lura, B. Biering, H.G. L
"otzke, H. Studemund, and V. Baturkin, *BIRD Microsatellite Thermal Control System–5 Years of Operation in Space*, Small Satellites for Earth Observation: Selected Contributions (2008), 277.

[50] J. Lyke, S. Cannon, D. Fronterhouse, D. Lanza, and T. Byers, *A Plug-and-play System for Spacecraft Components Based on the USB Standard*, proceedings of the 19th Annual AIAA/USU Conference on Small Satellites, Logan, UT, 2005, pp. 8–11.

[51] M. Monaghan, *Composite Crew Module, HyperSizer both pass NASA tests*, SAE International, http://www.sae.org/mags/aem/simul/7414, 2010.

[52] M. Wade, *UoSAT-OSCAR9, (UoSAT-1)*, http://www.qsl.net/bg4ji/satellite/sathist.htm, The Radio Amateur Satellite Corporation 2004, 1997.

[53] R. Magness, *A Comparison of CAN and Bluetooth Protocols A Study for Application of CAN Over Bluetooth for Wireless On-Board Data Handling for a Spacecraft Sensor Network*, DASIA 2003, vol. 532, 2003, p. 52.

[54] R. Magness, *Short-range RF Wireless (Proximity) Networks Technology Assessment for Space Applications*, TEC-E WIRELESS TECHNOLOGY DOSSIER (2006), ESA.

[55] G. Mathur, P. Desnoyers, D. Ganesan, and P. Shenoy, *Capsule: An energy-optimized object storage system for memory-constrained sensor devices*, Proceedings of the 4th international conference on Embedded networked sensor systems, ACM, 2006, pp. 195–208.

[56] G.V. Merrett, A.S. Weddell, A.P. Lewis, N.R. Harris, B.M. Al-Hashimi, and N.M. White, *An empirical energy model for supercapacitor powered wireless sensor nodes*, Computer Communications and Networks, 2008. ICCCN'08. Proceedings of 17th International Conference on, IEEE, 2008, pp. 1–6.

[57] S. Mobasser and C.C. Liebe, *Micro sun sensor for spacecraft attitude control*, JPL (2004), 6.

[58] Y. MontenegroMendez, *Modeling/Evaluation of modular spacecraft avionics network architectures*, 2009.

[59] J. L. Munoz, *Power Aware Computing and Communications (PAC/C)*, Tech. report, DARPA PAC/C, 2002.

[60] NASA, *The Mission and Instruments of IMAGE (Imager for Magnetopause-to-Aurora Global Exploration) — TECHNICAL APPROACH*, http://image.gsfc.nasa.gov, 2000.

[61] Dust Networks., *DN2510*, 2007.

[62] P. F. Michelson, *LAT Status – Data Acquisition System Electronics*, GLAST Users Committee, Stanford University, 2006.

[63] R. Jäntti, *Towards Reliable Wireless Sensor and Actuator Networks*, Department of Communications and Networking (2008), 2008.

[64] Ch. Reigber, *Satellite & Systems — The CHAMP Satellite*, http://op.gfz-potsdam.de/champ/systems/satellite_CHAMP.html, 2000.

[65] R.M. Rivas, A.H. Johnston, T.F. Miyahira, B.G. Rax, and M.D. Wiedeman, *Test results of total ionizing dose conducted at the Jet Propulsion Laboratory [bipolar and CMOS ICs]*, Radiation Effects Data Workshop, 2004 IEEE, IEEE, 2004, pp. 36–41.

[66] Daniel E. Rodriguez, *Solar Terrestrial Relations Observatory (STEREO) — Pre-Phase - A Requirements Review*, 1999.

[67] S. Maldonado, *LAT Housekeeping – Data Acquisition/Flight Software*, LAT-TD-02905, GLAST – LAT Flight Software, 2005.

[68] L. Schor, P. Sommer, and R. Wattenhofer, *Towards a zero-configuration wireless sensor network architecture for smart buildings*, Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings, ACM, 2009, pp. 31–36.

[69] C. Shaw, *EADS reveals developments in aerospace energy harvesting* , Tech. report, http://www.newelectronics.co.uk/article/26986/EADS-reveals-developments-in-aerospace-energy-harvesting-.aspx, 2010.

[70] J. Short and J. Sykes (SSTL), *New disaster monitoring satellite fit for space*, the space test chamber and the UK-DMC-2 satellite (J. Short, ed.), Science and Technology Facilities Council (STFC) and Surrey Satellite Technology Ltd. (SSTL) UK, 2008.

[71] E.Y. Song and K.B. Lee, *IEEE 1451.5 Standard-Based Wireless Sensor Networks*, Advances in Wireless Sensors and Sensor Networks (2010), 243–271, National Institute of Standards and Technology (NIST).

[72] T. Sorensen, G. Prescott, M. Villa, D. Brown, J. Hicks, A. Edwards, J. Lyke, T. George, S. Mobasser, K. Yee, et al., *KUTESAT-2, A Student Nanosatellite Mission for Testing Rapid-Response Small Satellite Technologies in Low Earth Orbit*, AIAA 3rd Responsive Space Conference, Citeseer, 2005.

[73] G. Studor, *Lessons Learned JSC Micro-Wireless Instrumentation Systems on Space Shuttle and International Space Station*, NASA Johnson Space Center (2006), CANEUS.

[74] S. Sudevalayam and P. Kulkarni, *Energy Harvesting Sensor Nodes: Survey and Implications*, Department of Computer Science and Engineering (CSE), Indian Institute of Technology Bombay (IITB), Tech. Rep. IITB/CSE/2008/December/19, TRCSE-2008, Dec (2008), 2008.

[75] JG Thayer, *High Reliability System Design Experience With the Gamma Ray Large Area Space Telescope (GLAST)*, Tech. report, Stanford Linear Accelerator Center (SLAC), 2007.

[76] N. Tsiftes, A. Dunkels, et al., *Enabling large-scale storage in sensor networks with the Coffee file system*, Proceedings of the 2009 International Conference on Information Processing in Sensor Networks, IEEE Computer Society, 2009, pp. 349–360.

[77] G. Prassinos V. Lappas and T. Vladimirova, *ESA Wireless Sensor Motes Study*, "ESA Wireless Sensor Motes Study", University of Surrey, 2007, p. Surrey Space Centre.

[78] T. Vladimirova, C.P. Bridges, G. Prassinos, X. Wu, K. Sidibeh, D.J. Barnhart, A.H. Jallad, J.R. Paul, V. Lappas, A. Baker, et al., *Characterising wireless sensor motes for space applications*, Adaptive Hardware and Systems, 2007. AHS 2007. Second NASA/ESA Conference on, IEEE, 2007, pp. 43–50.

[79] T. Vladimirova, X. Wu, K. Sidibeh, D. Barnhart, and A.H. Jallad, *Enabling technologies for distributed picosatellite missions in leo*, Adaptive Hardware and Systems, 2006. AHS 2006. First NASA/ESA Conference on, IEEE, 2006, pp. 330–337.

[80] M. Wade, *Amsat*, 1997.

[81] R.S. Wagner, *Standards-based wireless sensor networking protocols for spaceflight applications*, Aerospace Conference, 2010 IEEE, IEEE, 2010, pp. 1–7.

[82] R.J Walters, J. Garner, S. Lam, J. Vazquez, W. Braun, R. Ruth, J. Warne, J. Lorentzen, S. Messenger, R. Bruninga, et al., *Forward Technology Solar Cell Experiment First On-Orbit Data*, Proceedings of the 19th Space Photovoltaic Research and Technology Conference, NASA CP-2007-214494, 2007, pp. 79–94.

[83] AS Weddell, NJ Grabham, NR Harris, and NM White, *Modular plug-and-play power resources for energy-aware wireless sensor nodes*, Sensor, Mesh and Ad Hoc Communications and Networks, 2009. SECON'09. 6th Annual IEEE Communications Society Conference on, IEEE, 2009, p. 1.

[84] J.R. Wertz, *Spacecraft attitude determination and control*, Kluwer Academic Pub, 1978.

[85] WirelessHART, *HART Communication Foundation, Main page*, 2009, www.hartcomm2.org/index.html.

[86] X. Wu and T. Vladimirova, *Hardware-in-Loop Simulation of a Satellite Sensor Network for Distributed Space Applications*, Adaptive Hardware and Systems, 2008. AHS'08. NASA/ESA Conference on, IEEE, 2008, pp. 424–431.

Source Listings

A

A.1 Gateway Source Code

```
(Confidential)
```

Listing A.1: Redundant Coordinator Test Listing

A.2 DS3232 RTC Source Code

```
(Confidential)
```

Listing A.2: DS3232 RTC Listing

Redundancy Swichover Timings

B

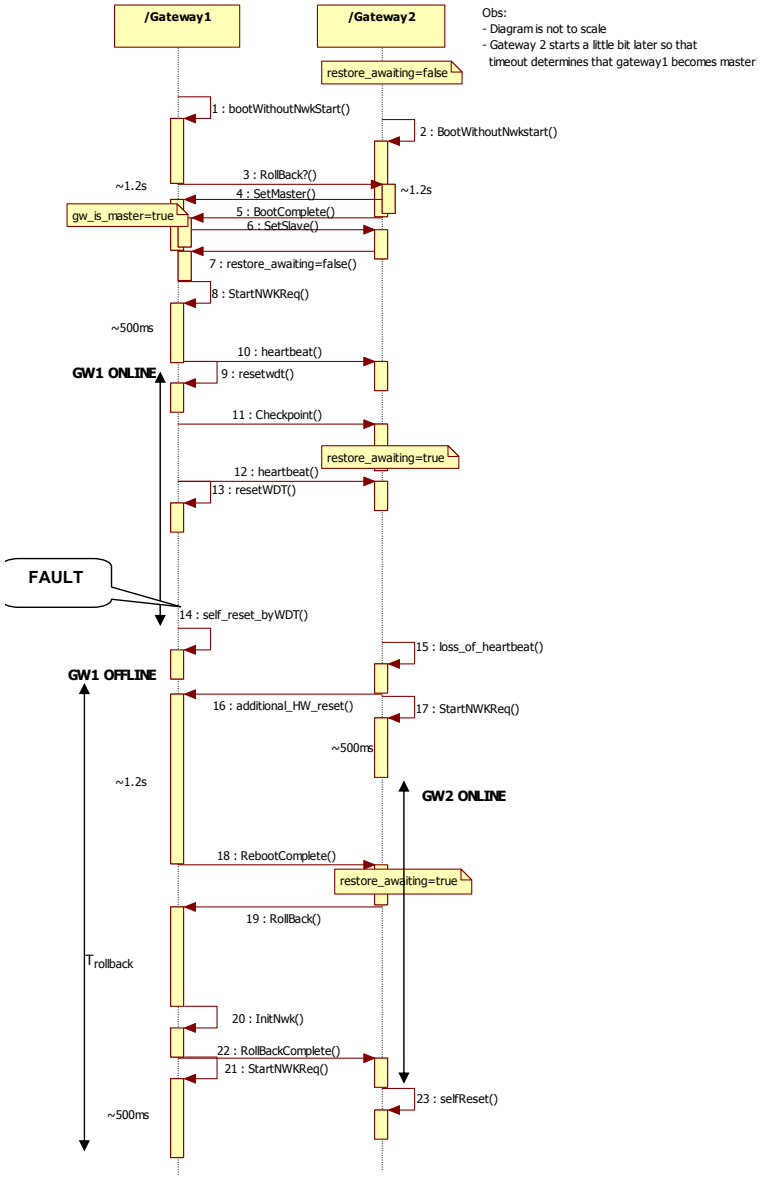


Figure B.1: Sequence diagram of transparent redundancy operation during switchover

Configuration Settings Listing

C

C.1 ZigBit BitCloud Configuration File Settings

```
#-----------------------------------------------------
# Components path definition
#-----------------------------------------------------
COMPONENTS_PATH = ../../BitCloud/Components

#-----------------------------------------------------
# Application path
#-----------------------------------------------------
APP_PATH = .

#-----------------------------------------------------
# Project name
#-----------------------------------------------------
PROJNAME = Isis_WSN1

#-----------------------------------------------------
# Compiler type
#-----------------------------------------------------
COMPILER_TYPE = GCC
#COMPILER_TYPE = IAR

#-----------------------------------------------------
# Boards selection
#-----------------------------------------------------
BOARD = BOARD_MESHBEAN

#-----------------------------------------------------
# Board-specific options
#-----------------------------------------------------
ifeq ($(BOARD), BOARD_MESHBEAN)
  HAL = ATMEGA1281

  #HAL_FREQUENCY = HAL_4MHz
  HAL_FREQUENCY = HAL_8MHz
  RFCHIP = AT86RF230

  APP_INTERFACE = APP_INTERFACE_USART
  APP_USART_CHANNEL = USART_CHANNEL_1
endif #BOARD_MESHBEAN

APP_INTERFACE_USART = 0x01
APP_INTERFACE_VCP = 0x02
#-----------------------------------------------------
```

```
# Avilable values for AT86RF212 chip: -17..11, for others: -17..3
#-------------------------------------------------
CS_RF_TX_POWER = 3


#-------------------------------------------------
# BitCloud security options
#-------------------------------------------------
SECURITY_MODE = NO_SECURITY_MODE
#SECURITY_MODE = STANDARD_SECURITY_MODE
ifeq ($(SECURITY_MODE), STANDARD_SECURITY_MODE)
  # Preconfigured network key
  CS_ZDO_SECURITY_STATUS = 0
  # Not preconfigured
  #CS_ZDO_SECURITY_STATUS = 3

  CS_NETWORK_KEY = "{0xCC,0xCC,0xCC,0xCC,0xCC,0xCC,0xCC,0xCC,0xCC,0
      xCC,0xCC,0xCC,0xCC,0xCC,0xCC}"
  CS_APS_TRUST_CENTER_ADDRESS = 0xAAAAAAAAAAAAAAAALL
  #-------------------------------------------------
  # APS security timeout period
  #-------------------------------------------------
  CS_APS_SECURITY_TIMEOUT_PERIOD = 10000

  #-------------------------------------------------
  # APS security buffers amount
  #-------------------------------------------------
  CS_APS_SECURITY_BUFFERS_AMOUNT = 4

endif #STANDARD_SECURITY_MODE

#-------------------------------------------------------------------------------

# BitCloud stack parameters
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------

# Stack library type detection  ** CHANGE BOTH FOR Change of node
    function! **
#-------------------------------------------------------------------------------

# Stack library supports all device types (Coordinator, Router and
    EndDevice)
#STACK_TYPE = ALL_DEVICES_TYPES
# Stack library supports Coordinator device type
STACK_TYPE = COORDINATOR
# Stack library supports Router device type
#STACK_TYPE = ROUTER
# Stack library supports End Device device type
#STACK_TYPE = ENDDEVICE

#-------------------------------------------------------------
# Device type detection (for the boards without DIP switches)
#-------------------------------------------------------------
```

```
# Device is Coordinator
APP_DEVICE_TYPE = DEV_TYPE_COORDINATOR
#APP_DEVICE_TYPE = DEVICE_TYPE_END_DEVICE
#APP_DEVICE_TYPE = DEV_TYPE_ROUTER
#*******************************************************************************


#-------------------------------------------------
# Channel mask and page to be used to run network
#-------------------------------------------------
CS_CHANNEL_MASK = "(1l<<0x0f)"

#-------------------------------------------------
# Whether the stack is to enable its receiver during idle periods
#-------------------------------------------------
CS_RX_ON_WHEN_IDLE = true
#CS_RX_ON_WHEN_IDLE = false

#-------------------------------------------------
# Extended PAN ID of the network to start or to join to
#-------------------------------------------------
CS_EXT_PANID = 0xAAAAAAAAAAAAAAAALL

#-------------------------------------------------
# Device UID (for UID use 0x0LL, other static)
#-------------------------------------------------
#CS_UID = 0x0LL
CS_UID = 0x00FF00FF00000000

#-------------------------------------------------
# End Device sleep period, ms
#-------------------------------------------------
CS_END_DEVICE_SLEEP_PERIOD = 125

#-------------------------------------------------
# Device short address
#-------------------------------------------------
#CS_NWK_ADDR = 0x0000

#-------------------------------------------------
# Maximum number of children that a given device (coordinator or router)
    may have
#-------------------------------------------------
CS_MAX_CHILDREN_AMOUNT = 6

#-------------------------------------------------
# Maximum number of routers among the children of one device
#-------------------------------------------------
CS_MAX_CHILDREN_ROUTER_AMOUNT = 0

#-------------------------------------------------
# The size of neighbor table
#-------------------------------------------------
CS_NEIB_TABLE_SIZE = 1
```

```
#-------------------------------------------------
# Maximum amount of records in the NWK Route Table
#-------------------------------------------------
CS_ROUTE_TABLE_SIZE = 8

#-------------------------------------------------
# The maximum number of hops that a packet may travel is twice that large
#-------------------------------------------------
CS_MAX_NETWORK_DEPTH = 1

#-------------------------------------------------
# Enabales or disables the power failure feature
#-------------------------------------------------
CS_POWER_FAILURE = true
#CS_POWER_FAILURE = false

#-------------------------------------------------
# Auto polling by end device
#-------------------------------------------------
CS_AUTO_POLL = true
#CS_AUTO_POLL = false

#---------------------------------------
#Poll rate for acknowledge by end-device
#-------------------------------------------------
CS_INDIRECT_POLL_RATE = 1000


#-------------------------------------------------
# Application parameters
#-------------------------------------------------
#-------------------------------------------------
# Application debug mode
#-------------------------------------------------
DEBUG = 0
#DEBUG = 1

#-------------------------------------------------
# Specifies APS Fragmentation usage (on/off)
#-------------------------------------------------
#APP_FRAGMENTATION = 1
APP_FRAGMENTATION = 0

ifeq ($(APP_FRAGMENTATION), 1)
  #-------------------------------------------------
  # The maximum blocks amount the asdu could be splitted into
  #-------------------------------------------------
  CS_APS_MAX_BLOCKS_AMOUNT = 4

endif #1
```

Listing C.1: ZigBit BitCloud Configuration File Settings

TEDS Listings

D

D.1 Example of a Wireless Strain Sensor TEDS

```
Dim Node_1 As New Node
Node_1.Node_ID       = "Wireless Node 1"
Node_1.Node_MAC      = "00010000112BF1B4"
Node_1.NodeLocation = "SC_TC11.770"
Node_1.NodeMajorLocation = "Solar panel array bulkhead 11"
Node_1.Sensor_UID    = "09000FFA000F1008" //64-bit sensor UID
Node_1.ShortAddr     = "0002" //Short Zigbee (MAC) address

//Strain Sensor connected by I2C (MCP3425)
Dim Strain_Sensor1 As New TEDS

Strain_Sensor1.HW_ADC_Device.I2C_Component_ID = "MCP3425"

//Continous ADC conversion here...
Strain_Sensor1.HW_ADC_Device.I2C_Command_WriteConfig = "I2CSTART,
    I2CWRTBYT,02,D0,98,I2CSTOP"
Strain_Sensor1.HW_ADC_Device.I2C_Command_Read = "I2CSTART,I2CWRTBYT,01,D1
    ,I2CRDBYTNLB,I2CSTOP"

Strain_Sensor1.HW_ADC_Device.I2C_DataLength  = 16  //16-bit ADC mode
Strain_Sensor1.HW_ADC_Device.I2CReadSpeed    = 400 //400KHz I2C modus
Strain_Sensor1.HW_ADC_Device.SuppliedVoltage = 3   //e.g. 3V power source
Strain_Sensor1.HW_ADC_Device.SupportedSampleRates = {15, 13, 18}
Strain_Sensor1.HW_ADC_Device.ADC_Channels = 1
Strain_Sensor1.HW_ADC_Device.ADC_ColdStart_Delay = 1000  //1000us delay

//*** Next, the descriptive parameters of the sensor here... *///
Strain_Sensor1.Sensor_characteristics.Accuracy = 2  //e.g. 2% accuracy
Strain_Sensor1.Sensor_characteristics.Max = 100
Strain_Sensor1.Sensor_Family    = Sensor_Fam_t.Fam_Strain
Strain_Sensor1.Sensor_Type      = Sensor_Type_t.Strain_resistive
Strain_Sensor1.Sensor_Circuit    = Circuit_Type_t.Circuit_Resistive
Strain_Sensor1.Sensor_location   = "Bulkhead 23"
Strain_Sensor1.Sensor_Units      = "Newton"  //Strain force
Strain_Sensor1.Sensor_DataLength = 16  //16 bits

Node_1.Sensor = Strain_Sensor1

Node_1.InteractionPattern.SampleFreq = 10       //10Hz Sampling Rate
Node_1.InteractionPattern.TransmissionFreq = 10 //10Hz Transmission Rate
```

Listing D.1: Simple Wireless Strain Sensor TEDS (pseudocode)