Intra-satellite communication based on Bluetooth Low Energy

Christoph Hagen

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 30.7.2017

Thesis supervisor:

Prof. Jaan Praks

Thesis advisor:

M.Sc. Nemanja Jovanovic

Aalto University
School of Electrical
Engineering

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Christoph Hagen

Title: Intra-satellite communication based on Bluetooth Low Energy

Date: 30.7.2017 Language: English Number of pages: 6+62

Department of Electronics and Nanoengineering

Professorship: Circuit theory

Supervisor: Prof. Jaan Praks

Advisor: M.Sc. Nemanja Jovanovic

Wireless communication between satellite subsystems can have many benefits regarding the design and operation of satellites. A wireless communication system for intra-satellite communication based on Bluetooth Low Energy technology is presented, including hardware and software integration details, as well as the results of several tests regarding the most important design criteria for its application in small satellites. All tested aspects, including power consumption, signal strength, packet error rates, latency and achievable data rates, suggest that the presented system is suitable for in-orbit deployment. The system will be further validated with the Aalto-3 satellite, currently being developed by Aalto university.

Keywords: Wireless satellite network, Bluetooth Low Energy, Intra-satellite communication, wired bus replacement

Preface

This thesis is dedicated to my parents, who, through their unconditional love and financial support, allow me to follow my dreams. I could not be more grateful.

Many people have contributed to this work, either directly through suggestions and advice, or in other ways though motivation and support. My professor Jaan Praks deserves special appreciation, as he always helped me with everything I needed, from advice and guidance to providing me access to office space, laboratories and hardware. I could not have been taken better care of. Nemanja Jovanovic helped me often with his knowledge regarding the Aalto-2 satellite and with problematic aspects of my thesis. Additional thanks are also in order for everyone involved in the Aalto-3 satellite project, which gave me a great deal of motivation to deliver quality work. I would also like to thank my friend Aman, for his support in so many ways, and Charlotte, for her unique ability to motivate and inspire me.

And how could I forget all my friends from the spaceMaster programme, who made the last two years this incredible journey. You will always have a special place in my heart.

Special thanks go to the Education, Audiovisual and Culture Executive Agency (EACEA), the European Commission, the Julius-Maximilians-Universität in Würzburg, Luleå University of Technology and Aalto University, for managing and participating in the Erasmus+ programme, and for organising the spaceMaster studies. International cooperation and free education for everyone are the foundation upon which we can build a brighter future.

Otaniemi, 10.7.2017

Christoph Hagen

Contents

Symbols and abbreviations

Abbreviations

ADC	Analog-digital Converter
ADCS	Attitude Determination and Control System
AOCS	Attitude and Orbit Control Subsystem
API	Application Programming Interface
ATT	Attribute Profile (for Bluetooth Low Energy)
BER	Bit Error Rate
BLE	Bluetooth Low Energy
CAN	Controller Area Network (bus system)
CMOS	Complementary Metal-Oxide-Semiconductor
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
DC	Direct Current
DLE	Data Length Extension
DLR	Deutsches Institut für Luft- und Raumfahrt (German Aerospace Center)
EMI	Electromagnetic Interference
EPS	Electric Power System
ESA	European Space Agency
ESTEC	European Space Research and Technology Centre
GAP	Generic Access Profile
GATT	Generic Attribute Profile
GFSK	Gaussian Frequency Shift Keying
GPIO	General Purpose Input/Output
GPS	Global Positioning System
I^2C	Inter-Integrated Circuit
I^2S	Inter-IC Sound
IC	Integrated circuit
IEEE	Institute of Electrical and Electronics Engineers
INTA	National Institute for Aerospace Technique of Spain
IoT	Internet of Things
IrDA	Infrared Data Association
ISM	Industrial, Scientific and Medical (radio band)
LDO	Low Dropout (Regulator)
LE	Low Energy (referring to the Bluetooth Low Energy protocol)
MAC	Medium Access Control
MTU	Maximum Transmission Unit
NASA	National Aeronautics and Space Administration
NTNU	Norwegian University of Science and Technology
NUTS	NTNU Test Satellite
OBC	Onboard Computer
OPTOS	Optical Satellite, developed by INTA
OWLS	Optical wireless links to intra-spacecraft communications
PER	Packet Error Rate
PCB	Printed Circuit Board
PSR	Packet Success Rate
PWM	Pulse-width Modulation

RAM	Random Access Memory
RF	Radio Frequency
RSSI	Received Signal Strength Indicator
RTOS	Real-time Operating System
RX	Receive
SEL	Single Event Latch-up
SEU	Single Event Upsets
SNR	Signal-to-Noise Ratio
SPI	Serial-Peripheral Interface
SPOF	Single Point of Failure
TM/TC	Telemetry and Telecommand
TRL	Technology Readiness Level
TT&C	Telemetry, Tracking and Command
TWI	Two-wire interface
TX	Transmit
UART	Universal Asynchronous Receiver and Transceiver
UHF	Ultra high frequency (300 MHz to 3 GHz)
UUID	Universally Unique Identifier
UWB	Ultra-wide band
VDD	Power supply pin for integrated circuits
XOR	Exclusive OR (logic operation)

Note: The prefixes for bytes and bits are used according to the decimal standards published by the International Electrotechnical Commission (IEC), i.e. 1 kbit is equivalent to 1000 bit, not 1024 bit.

1 Introduction

Wireless communication has become an integral part of our daily lives, as an increasing number of devices omit cable connections to give the user more mobility, easier setup and increased flexibility. Sophisticated technologies like WiFi and 4G wireless communication are used daily by millions of people and provide high data rates, high range, small chips sizes, low energy consumption, reliable connections and easy setup. While these wireless technologies have a wide range of consumer and industrial applications, wired connections are still used in many areas where reliability, high data rates and low latency are required, and reduced mobility of the participating devices can be tolerated. One area where wireless solutions are almost non-existent is for internal communication in spacecraft and avionics systems. While wireless data transmission is naturally the only means of communication with satellites from the ground, data transmission between individual satellite modules still almost exclusively relies on wired technologies. There are multiple reasons why wireless solutions have seen little development in this sector, the most important one being reliability. There is always some reluctancy to switch technologies, especially when it comes with an associated risk of failure. Additionally, there are not many fully developed solutions available that could be easily adopted.

The argument against wireless systems due to reliability can be attributed to the unique nature of space missions: It is usually infeasible to physically service a spacecraft once it is deployed, and it is difficult to test all environmental conditions prior to launch. In this regard, missions to space differ greatly from other engineering areas, since most satellites are - in a manner of speaking - prototypes. So when a satellite systems engineer his given the choice between something flight-proven and an untested technology, he/she will (and in most cases should) opt for reliability[1]. Ideally, trials of new technologies would be carried out in an environment with no repercussions in case of unsuccessful tests. This is notoriously difficult in the area of space technologies due to the uniqueness of the environment in space. Failures in the space industry usually result in huge losses of both money and time, and any malfunction in the communication bus will likely be critical (a spacecraft without a functioning network will almost certainly be inoperable). This is one of the reasons why this area has not seen significant changes since the early days of spaceflight. However, given the technological advancements in the last decade, wireless technologies have reached a point where they should be considered as a suitable alternative to traditional wired bus systems. Therefore, this thesis aims to provide an argument to move away from a wired bus structure in favour of a wireless intra-satellite communication system. This is done by outlining the benefits of such an approach, and by providing a usable concept with minimised risk of failure, accompanied by tests of all necessary aspects of the application.

A communication network is necessary for all spacecraft, and certain requirements have to be met by any communication solution for satellite applications:

[1]This practice can lead to somewhat unusual phenomena: During the last years of the Space Shuttle operations, NASA increasingly had to buy components from private sellers, as some required parts for ground support hardware were no longer produced by the original manufacturer [1].

– Reliable operation for the duration of the mission

– Timeliness of the data transmission between real-time critical systems

– Sufficiently high data rates for all subsystems

– Low energy consumption (especially in small satellites)

– Small size footprint and lightweight

Wired systems perform reasonably well for these aspects. They usually work reliably, but are not completely failsafe, and a short circuit in one of the connection lines can affect the whole satellite. The transmission is usually fast, and the possible data rates exceed the usual requirements of small satellites. Energy consumption is low, albeit not zero [5]. The actual hardware needed for wired connections is very small, but the necessity to connect subsystem modules usually requires connectors, which can be quite large compared to the rest of the satellite structure. Figure 1

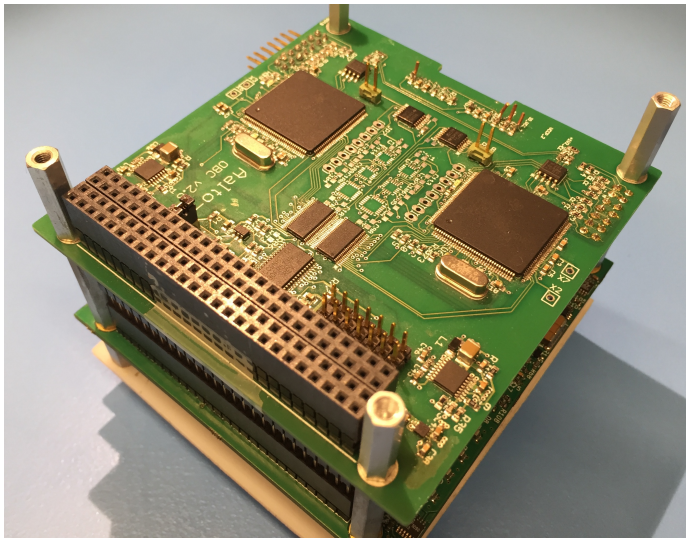


Figure 1: The experimental model of the Aalto-2 satellite, developed at Aalto university. The large connector for the communication between the subsystems can be seen in the front.

shows the experimental model of the Aalto-2 satellite developed at Aalto university, where the connector consumes a significant area on each circuit board. It has been stated that wires and the associated harness usually account for up to 10 % of a satellite's dry mass [2] [33] [39]. This number could likely be reduced by utilising highly integrated wireless chips on every subsystem, thereby removing the need for much of the harness and wires otherwise required to connect the satellite systems. Even though many of the wires and connections can be removed, there is still a need for some wires in order to distribute power to all systems.

There are other disadvantages associated with wired communication systems as well, many of them affecting the design and production phase of the satellite. The most important problems are:

- Complex wire path layout in the satellite structure
- Difficult integration of new modules
- Number of connections increases drastically with number of components
- Mass distribution changes when adding connections
- No/few redundant communication paths
- Time consuming and expensive design iterations
- Difficult testing and debugging

Most of these problems can be solved with appropriate designs, but these sometimes require considerable effort. For the Aalto-2 satellite, the connections need to be assigned to the pins of the connector, and all subsystems must be designed to conform with that assignment. It can be important to physically distance certain pins from one another, as to not introduce interference. Furthermore, the connector has a limited number of pins, which could be insufficient for more complex designs or larger satellites with more modules. Wireless data transmission generally has a higher number of possible concurrent peer-to-peer connections. It is also difficult to add new modules to a wired configuration, since each module in the stack needs to provide the full connector to relay the communication between other modules, even for pins that are not used by the module itself. Again, a wireless system provides easier integration, and each module only needs to concern itself with the data that is intended for it. If multiple modules need to access the same device, then each wireless node can transmit the same data to multiple devices through some form of Multiple Access Control (MAC). A wired network usually needs a physical connection between each pair of modules[2]. This can lead to a high number of connections for highly interconnected satellites, especially when using protocols that need multiple wires (e.g. the Serial-Peripheral Interface (SPI) requires a select line for each slave, in addition to the two communication wires). Cables and their attachments can also change the satellite center of mass, which can affect attitude control. Additionally, it is difficult to create redundant communication paths, especially if the paths should run through different locations in the satellite. As mentioned before, it is also possible that a short circuit in one of the communication lines disables multiple subsystems, or even the whole satellite. In contrast, wireless systems are electrically decoupled from one another. Wireless modules can also be beneficial for movable parts, e.g. deployable solar panels, where additional wires complicate the design.

These mentioned problems of wired bus systems can be addressed by a well designed wireless solution, while special focus has to be placed on the critical requirements, including reliability, power consumption, and latency. The degree to which these factors are substantial largely depends on the requirements of the specific satellite application in which the technology is used. The projects will differ in

[2]This depends on the network topology, as some topologies are more efficient regarding the total number of connections that are necessary.

complexity, funding, mission specification, lifetime, satellite size and weight, available resources, and timeframe, which will change the importance of some of the mentioned aspects. It is important to remember that there is no single optimal solution for all applications, but some general remarks can be made about the characteristics which a wireless communication bus needs to provide. The design presented in this thesis is optimised to provide a replacement or addition to the wired communication systems onboard of small satellites (especially CubeSats), with more rigorous constraints on power consumption, mass, size and cost.

One reason for the slow adoption of wireless hardware is the missing maturity of the technology with regard to space applications. To evaluate this maturity, NASA and ESA use the Technology Readiness Level [55], which is a scale from 1 to 9 in order to measure the advancement of a system in terms of successful test, deployment and demonstrated capabilities. Some wireless systems have already successfully demonstrated their functionality, such as the wireless optical bus in the OPTOS satellite (see section 2.2), which raises its TRL to 9. For solutions based on radio frequency transmission, the TRL ranges somewhere around 4–5, since many concepts have demonstrated that the technology can work in a laboratory environment. There are to date no missions with a reported success of wireless RF based communication network, but single wireless components have been successfully tested.

This thesis attempts to increase the technology readiness level of a wireless intra-satellite network, by testing all necessary aspects of such a system. The described concept is currently being implemented in the Aalto-3 CubeSat developed at Aalto university, which will provide a platform to test and validate the technology.

The following chapter provides an overview of existing wireless technologies and concepts. Commonly used wireless protocols are briefly summarised, and several approaches by other institutions are presented and analysed with regard to their deployment in small satellites.

The third chapter introduces the challenges which have to be considered in such a wireless design, and gives a comparison to an equivalent wired counterpart, the communication architecture of the Aalto-2 satellite. The choice of Bluetooth Low Energy (LE) is justified in this chapter as well, and the results of some initial simulations are given, further validating the decision.

A more detailed description regarding the implementation of Bluetooth LE is provided in chapter 4. First, the selection of the specific chip used for further tests is explained, followed by the general operating principles of Bluetooth LE. Implementation details for the chosen hardware simplify its integration in existing projects, including the most important aspects regarding the hardware and software interface. Additionally, a concept of a redundant system is outlined, which can provide higher reliability for critical communication links.

Chapter 5 focuses on the hardware tests which were carried out with the selected Bluetooth LE chip. Measurements of signal strength, as well as tests with packet error rates, latencies, power consumption and data rates provide the justification for further development.

Finally, the thesis concludes with a summary of the achieved goals, and provides some suggestions on ways in which the concept could be improved in the future.

2 Wireless technologies for intra-satellite networks

Even though wireless intra-satellite networks have not been adopted in many satellite missions, some research and development has been done, ranging from completely theoretical and abstract concepts to sophisticated test on satellite mockup hardware, and even some successful missions, where hardware was tested and used in space. This section provides an overview[3] of potentially applicable technologies for intra-satellite communication as well as short descriptions of similar projects in that area.

2.1 Communication protocols and standards

Some wireless technologies and communication protocols commonly found in household devices can be considered as the technology behind a wireless intra-satellite network, thanks to the advancements in consumer grade hardware over the last decade, and the considerable market for wirelessly connected consumer devices like smartphones, home automation and Internet of things (IoT) technologies. The network technologies discussed in this section offer some qualities that are rarely found in custom hardware in the space sector: High integration of IC components in small packages, low cost parts, extensive documentation, and easy setup and use. Consumer hardware has reached a level of professionalism and quality that warrants its evaluation for space applications.

WiFi

Being one of the most well-known technologies and widely adopted for internet connectivity in private and public areas, WiFi is based on the IEEE 802.11 standard and operates in the 2.4 GHz ISM band. It offers high data rates (up to 1300 Mbit/s with 802.11ac [9]) and typical ranges of up to 100 m. Depending on the protocol used, slower data rates (11 Mbit/s for 802.11b and 54 Mbit/s for 802.11a/g) can decrease power consumption moderately, to the range of below 200 mW. Even though the International Space Station features a WiFi network [61], it is not a good option for other spacecraft, since the power consumption of the WiFi chips is too high for the power budget of most satellites, and the data rates that WiFi provides are usually not needed inside satellites. The necessity of a rather complex router is another drawback that would warrant the use of WiFi only for the biggest satellites.

Bluetooth / Bluetooth Low Energy

The Bluetooth protocol (consisting of the classical version and it's low power counterpart LE) is commonly used for wireless headsets, speakers, keyboards and other peripheral hardware. Similar to WiFi it uses the 2.4 GHz band, but is superior over WiFi in situations where low power consumption is more important than high range and fast data transmission. Bluetooth has been developed since 1994 and modern

[3] The technologies presented here are only a small subset of all existing technologies. Less important ones for the purpose of this thesis have been excluded

chips will soon include Bluetooth 5, the latest specification, which offers ranges of up to 100 m and data rates of up to 3 Mbit/s (2 Mbit/s for Bluetooth LE) [8]. With version 4.0 of the specification, Bluetooth Low Energy (LE) was introduced as an alternative to classical Bluetooth, and is more focused on low power connections, providing longer lifetime to battery powered devices. Bluetooth chips can support both classical Bluetooth and Bluetooth LE, or only one of them. Pure Bluetooth LE chips are not compatible with chips incorporating only classical Bluetooth.
As a technology for intra-satellite communication, especially Bluetooth LE can be suitable in terms of chips sizes, power consumption, data rates, and signal strength.

Zigbee

The Zigbee standard is a low-power, low data rate, close proximity protocol based on the IEEE 802.15.4 standard [10]. It is developed by the Zigbee Alliance and is mainly used for IoT devices, home sensors and light switches. To increase coverage to more than the typical point-to-point range of 10 m to 20 m, the devices are organised in a mesh network. ZigBee defines a data rate of 250 kbit/s in the 2.4 GHz band, which makes it suitable for communication in small satellites without high demands for fast data transmission.

IrDA

IrDA is a standard developed by the Infrared Data Association and uses pulsed infrared light to establish a connection between devices. It requires line-of-sight between the two endpoints, and can achieve data rates of up to 1 Gbit/s with the Giga-IR standard released in 2009 [11]. The technology gained some momentum in the early 2000s with its inclusion in laptops, desktop computers and handheld devices and was believed to play an important role in mobile communications [13], but interest died down with the advancement of Bluetooth and WiFi.
Lower speed versions of IrDA exist, with a power consumption of around 10 mA during operation [12]. Its low power operation, simple protocols and sufficient speed make infrared communication a viable option for intra-satellite networks, with the drawback of the design constraints enforced by the line-of-sight requirement.

Ultra-wide band

Ultra-wide band (UWB) technology is a hypernym for RF technologies in the GHz range with a very wide bandwidth (generally more than 500 MHz). In theory, this enables very high data rates (several hundred Mbit/s) with very low power consumption compared to other narrow-band technologies like Bluetooth or WiFi. Another supposed advantage of UWB is the possibility to measure the time-of-flight of the signals, and thereby provide ranging capabilities. A lot of research has gone into the theoretical aspects of UWB, examining its application in wireless sensor networks [14], [17], identifying some design challenges [15], and comparing it to Zigbee [16]. However, the development of the UWB standard IEEE 802.15.3a came to a halt after two proposals for the standard were backed by several companies on both

sides, which lead to a deadlock and the task group was dissolved [18]. In addition to standardisation problems, hardware development proved to be more challenging than expected, and real-world data rates and power consumption were far below expectations. The chips were also significantly more expensive than comparable WiFi or Bluetooth modules, which lead to the technology being almost completely abandoned by major chip manufacturers.
One of the few more popular UWB chips is the DW1000 from Decawave, which provides ranging with 10 cm precision and data rates of up to 6.8 Mbit/s. Even though this is several times the capacity of Bluetooth, the chip also consumes around 10 times the energy compared to current Bluetooth LE modules[4] [19] [20]. High energy consumption is the main drawback for smaller satellites, though a wireless intra-satellite network for medium-sized satellites based on the DW1000 is worth considering.

2.2 Existing projects

There are a limited number of projects that prove the suitability of wireless technologies for intra-satellite communication, and the adoption of these concepts is slow. The following section gives a quick overview of the progress that has been made in this area, with a focus on the requirements of CubeSat missions.

Wireless Communication Bus For Satellite Applications WI-SAT

The WI-SAT project was an effort funded by ESA/ESTEC and carried out by Control Data Systems SRL as the main contractor. Its main goal was to investigate *"the feasibility of a robust spacecraft communication bus system with reduced or no harness"* [22] while focusing on low power consumption and minimising the overall mass. In the 18 month long project, the ISA100 wireless protocol was ported to a IEEE 802.15.4 physical layer based on the Decawave DW1000 module mentioned in section 2.1. The developed solution to intra-satellite communication is presented in [3], with packet success rate (PSR) tests performed on a Sentinel-3 satellite mockup (a 0.75 m (L) x 0.75 m (W) x 95.5 cm (H) aluminium box equipped with 4 dividing panels inside). The tests show excellent PSRs of 100% on most channels for carbon fibre divider panels, with the module placement not negatively affecting the results. While the paper states that *"This proves that the solution employing VN360 UWB modules is suitable for replacing the intra- spacecraft wired communications"*, and that *"the proposed solution provides high performance for low power UWB transmissions"*, no details are given on latency of the communications, power consumption values or achieved data rates. Some additional information is available in [21], suggesting that latencies were below 10 ms for transmissions at 6.8 Mbit/s, which should be sufficient for real-time critical systems like attitude control. However, the proposed system would most likely not be applicable for smaller satellites, as the power consumption

[4]The pure DW1000 chip has a TX/RX power consumption of 70/30 mA at 3.3 V, while the nRF52832 from Nordic Semiconductor features 5.3/5.4 mA at 3.3 V . Additionally the nrF52 chip features a 48 MHz micro controller, while the DW1000 is a plain module accessible through SPI.

of several DW1000 chips would be too high for the constrained power budget of a CubeSat platform.

Other UWB projects

Another project relying on the DW1000 module is presented in [23], where (similar to WI-SAT) the PER inside a satellite mockup was investigated. The PER was lower than 10^{-3} inside an aluminium box of 1 m x 1 m x 1 m with two aluminium plates between receiver and transmitter. No tests with multiple modules or different constellations were conducted, and power consumption or latency tests are also absent. This work can only be seen as a small indicator for the viability of wireless intra-satellite networks, specifically using ultra-wide band technology.

A theoretical analysis of using UWB impulse radio (UWB-IR) for low latency wireless networks in satellites is presented in [31]. Produced by members of the German Aerospace Center (DLR), it investigates the use of an IEEE 802.15.4e low latency deterministic network (LLDN) to achieve latencies lower than 10 ms for networks with more than 25 devices, mainly focusing on the Attitude and orbit control subsystem (AOCS). From theoretical assumptions and estimations the claim is made that UWB-IR is *"well suited for its use in satellite systems"*. Other important factors for the deployment of wireless technologies in satellites (e.g. low power consumption, small size, sufficient data rate) are not further considered, with the exception of interference and good multi-path fading immunity, which is cited from [32], where the claim is made without being backed up by actual data or sources.

The work in [32] is conducted by Honeywell International and financially supported by ESA. A not specified Decawave RF chip (likely the DW1000 as used in [22], [23] and [3]) is used to test bit error rates (BER) for multiple modules in a Venus Express mockup. The tests indicate the suitability of the Decawave chip for wireless communication inside satellites, with the paper again being vague about other important aspects besides error rates.

NTNU Internal Wireless Bus

The Norwegian University of Science and Technology (NTNU) hosts the NTNU Tests Satellite (NUTS), a student CubeSat platform supported by university staff. For this experimental satellite, a wireless communication system was proposed in [28], as to replace/cooperate with the main I2C bus. Each module is plugged into a backplate, which handles the communication and power supply of the modules. To support and extend the main bus, and to experiment with new technologies, a wireless communication bus is envisioned. Building on top of a commercially available nRF24L01 chip [29] with integrated low-level drivers, it is meant to offer an alternative to the wired bus system. The nRF24L01 chip is a very inexpensive RF module (retail price around 1 €) which uses the 2.4 GHz frequency band and can support data rates of up to 2 Mbit/s. Low power consumption (less than 15 mA in RX/TX) and small footprint (chip size 4 mm × 4 mm) make it suitable for the purpose. It comes complete with a software stack to create point-to-point communications. Early test with packet loss in an environment similar to a small satellite showed minimal

packet loss [27]. The master thesis work presented in [26] did not complete the full implementation of a multi-user wireless network, though a concept is established and the communication between two nodes is shown. Furthermore, the integration of FreeRTOS to manage the network is proposed. Since these transceivers are very popular for low cost home automation applications, a variety of network protocols exist to deal with larger networks. The RF24 protocol [30] organises the nodes in a tree structure, taking advantage of the fact that each device can listen to 6 other nodes at the same time. Many variations of this protocol exist to adapt to different scenarios, and some might be applicable to extent the work done in [26] to a fully functional solution.
The NUTS satellite had a targeted launch date in 2014, but was not yet sent to orbit at the time of this writing (May 2017).

VELOX-I and VELOX-PIII

Constructed and built as part of the Undergraduate Satellite Program at the Nanyang Technological University in Singapore [34], VELOX-I is a nano satellite which will test intra-satellite communication based on the ZigBee platform. After the initial launch phase, the main satellite will establish a wireless connection to the pico satellite VELOX-PIII, which will then be detached. While the satellites drift apart, they will transmit data until out of reach. The measured RSSI is then used to estimate the distance between the modules, in order to determine the suitability of the protocol for inter-satellite communications. Preliminary ground tests have estimated the maximum distance to be around 1 km [33]. While the experiment focuses more on inter-satellite communication as opposed to intra-satellite data transmission, the experiment can still give valuable information about the reliability of wireless consumer-grade chips in space environments. In-flight data was not yet available at the time of this writing.

Optical wireless links for intra-spacecraft communications (OWLS)

Besides the more commonly used RF-based wireless links, optical communication can be used in a similar fashion. Optical wireless as a solution for interconnections between devices was already proposed in 1999 by the National Institute for Aerospace Technique of Spain (INTA) [35]. Since then, the technology has seen much progress, with the first in-orbit test of wireless optical intra-satellite communication onboard NANOSAT-01 [24]. Developed and manufactured by INTA, the medium sized satellite ($\sim 20\,\mathrm{kg}$) was launched in 2004 and included several experiments to test the technology. A redundant wireless link from a 3-axis magnetometer was used to compare the data received from the optical link with the same data from a wired connection, in order to investigate Single Event Transients in the optical detectors produced by proton incidences. The results showed that errors mainly occur over the South Atlantic Anomaly, where the BER is in the range of 10^{-6} [25]. A second experiment was used to determine the BER for light reflected from a satellite wall, and the degradation of the optical detectors and emitters was investigated as well.

All tests and the continued functionality over 4 years since the launch show that optical wireless communication is a well suited technology for intra-satellite communication.

With the positive results from NANOSAT-01, another satellite was developed by INTA to further develop the technology. OPTOS, a 3 unit CubeSat, is equipped with a fully optical bus between all satellite subsystems [36]. Each PCB module features an emitter and a receiver for 950 nm light, which are encapsulated in a module with the dimensions of 25 mm × 14 mm × 14 mm [37]. To facilitate line-of-sight between all modules, each device is located at the back of the satellite, where a small corner of each PCB is cut out. Each module has an average output power of 26 mW, and a reduced implementation of the CAN protocol is used with a data rate of 125 kbit/s. A total of 9 modules is present in OPTOS, each having a power consumption of 81 mW for the receiver, and 13 or 50 mW for the transmitter, depending on the mode [41].

The OPTOS satellite was launched in November 2013, and was operating with no problems regarding the optical communication system for almost 3 years, despite the expected lifetime of only one year [47].

Delft university of technology

An Autonomous Wireless Sun Sensor (AWSS) was successfully tested in the Delft-C3 nanosatellite, which launched in April 2008. The AWSS contains an nRF9E5 System on a Chip (SoC) operating in the 915 MHz band in order to communicate with the onboard computer (OBC). Two identical sensors were integrated in the satellite, with only one functioning properly. Data from the satellite could not determine the cause of the failure, and efforts to pick up the RF signal from the sensor with a large telescope failed, indicating that the sensor itself was not working [40]. The other sensor however worked as expected.

Additional work by R. Schoemaker [42] investigated the use of Bluetooth Low Energy for the communication between submodules. Extensive tests were conducted regarding packet error rates (PER), power consumption, achievable data rates, and connection stability. The results support the conclusion that Bluetooth Low Energy is a viable option for a wireless satellite bus, with certain drawbacks considering the available data rates with the specific chip used (Bluegiga BLE113) [38]. Nevertheless, the tests conducted show that packet error rates are sufficiently low under realistic conditions, and that power consumption is sufficiently low, especially for low rate communication e.g. from sensors. The thesis work proposes the test of the Bluetooth technology in space, by deploying a wireless temperature sensor on the DelFFi nanosatellite. It is currently in the development phase, and a prototype of the OBC board is being tested, including a Bluetooth chip for intra-satellite communication [43].

A broad overview of wireless communication in spacecraft is again given in [39], including design considerations, energy management and comparisons between wired and wireless bus protocols. An energy manager is then proposed to optimise energy consumption for an ADCS system based on a decentralised wireless approach using

the Zigbee protocol. The results of this paper were partly included in the Delfi-C3 satellite.

3 Concept of a wireless intra-satellite network

The design of a wireless network for satellite subsystems depends significantly on the specified mission. This chapter provides an analysis of the requirements that need to be fulfilled by such a communication system. Many of the aspects discussed here can vary in importance for different applications, and a more specific analysis is done with regard to the Aalto-2 CubeSat project.

3.1 General requirements of a wireless bus

A number of different aspects need to be carefully considered in order to design a wireless communication solution as a replacement for a traditional satellite bus. The most important considerations for an RF based[5] wireless bus are provided here in a concise manner[6], extending and unifying the considerations stated in [2], [3], [4], and [39].

Data rate

Similar to a wired systems, any wireless solution will need to provide sufficient transmission speed to exchange all data between the satellite subsystems. The amount of data to be transmitted ranges from a few bytes (e.g. housekeeping data or sensor information) to several megabytes per second (image data between payload and communications module). For some scenarios a hybrid approach might be considered, where a fast wired connection between the payload and the telecommunication subsystem extends the wireless bus to free up some bandwidth.

For a wireless bus, the data rate calculations will need to factor in more than just the raw data between clients, but also account for packet sizes, protocol overhead, error correction, packet loss and retransmission, as well as propagation delay and possibly frequency switching times.

Latency

Some types of data have different timing requirements than others. Measurements or images from the payload could possibly be transmitted to the telecommunication module whenever the transmission medium is free. In contrast, communication between the attitude control subsystem and orientation sensors should be as fast as possible to enable accurate control. The bus system needs to guarantee these requirements even at times where many modules want to transmit simultaneously. The communication protocol needs to account for these aspects, and design parameters like carrier frequency, modulation schemes and medium access control (MAC) have to be selected accordingly.

[5] The decision towards an RF based system is detailed in section 3.3

[6] The order in which the aspects are presented does not relate to their respective importance. The significance of these points strongly depends on the satellite mission.

Size

Especially small satellites will require a compact solution which can be integrated in every module without excessive use of space. This becomes increasingly important when the satellite contains a high number of small modules. In these cases, a hybrid solution might be suitable: A number of smaller subsystems connect to the same wireless chip, which manages the connection to the other components. The operational frequency of the bus determines the necessary antenna size, which is another parameter to consider in the design process. The antenna design also needs to account for signal loss and radiation patterns inside the satellite.

Weight

One of the advantages of wireless systems is that it only requires wires for the power supply of the modules. Some estimates place the weight of bus systems and connectors at up to 10% of the dry mass of a micro-satellite [2] [33] [39][7]. It is only possible to achieve any significant mass reduction if the individual communication nodes are lightweight.

Power consumption

Every satellite has a limited capability to produce energy, typically limited by the solar panel size. This energy should be available to the subsystems without being drastically reduced by the power consumption of the bus system. Wired interfaces like I^2C, SPI or UART are typically operating with RX/TX power consumptions of around 10 mW [5], and wireless replacements should try to operate in similar ranges. There are several design parameters that can be adjusted to reduce the required output power, and the operational frequency and bandwidth can affect the consumed energy. For a suitable selection, the wave propagation inside the satellite structure should be carefully evaluated, to determine the minimum output power for the wireless modules (while still providing a sufficiently low PER). Moreover, the chips should be put in sleep mode when no data is received or transmitted[8].

Wave propagation

Satellites are usually tightly packed, and the paths between the wireless nodes can be heavily obstructed by other modules. These modules usually contain large ground planes, which are difficult to penetrate for some frequencies[9]. The situation is further complicated by the satellite structure, which often consists of aluminium and other metals. This can lead to undesirable multi-path propagation, which can make it difficult to properly detect the signals at the receiver side. In order to prevent these

[7]The reported percentage varies a bit, depending on publication. Some estimates are as low as 6–8 % , other are as high as 15 % [4].

[8]These times are also affected by the speed of the transmission, since a faster connection can transmit the data quicker, and the chip can therefore sleep longer in between transmissions.

[9]Generally lower frequencies can penetrate materials more easily, and bend around obstacles.

effects, the placement of the modules must be considered, as well as antenna radiation patterns, operational frequency, output power, and modulation methods. Waveguides can be considered to improve the propagation properties.

Regulations

All space missions are subject to regulations concerning the frequency bands they operate in, and the output power they produce. For the type of low power, short-range communication featured by a wireless intra-satellite bus, these concerns are likely to be minimal, especially since the satellite structure will prevent most of the radiation from leaving the satellite. However, the concept must still be evaluated to confirm that all regulations are met. It should also be considered that, while the satellite will operate in space, it will still be developed and tested on the ground, and it is necessary to comply with the restrictions of the countries where the satellite is built. Furthermore, the technologies discussed here might be deployed in terrestrial applications as well, such as mobile robotics platforms, where frequency bands are more strictly regulated.

Electromagnetic interference

There are usually some satellite subsystems that use RF communications (e.g the TM/TC module), and the wireless bus needs to be designed to not generate interference with the existing hardware. Overlaps in frequency (including multiples of the operational frequency) can result in a reduced signal-to-noise ratio (SNR) and increased PER and BER. Since longer wires effectively function as monopole antennas, the transmitted signals could introduce oscillations in power lines, potentially resulting in measurement errors or other failures.

Network topology

Fundamental differences exists between wired and wireless network topologies. The physical topology of a wired infrastructure is defined by the cable connections between the nodes, and different types exist: point-to-point, tree, star, mesh, ring, bus, hybrid, and daisy chain. In contrast, wireless systems only have a logical topology, since no physical connection exists between the devices. In theory, all existing physical topology types can be implemented by wireless protocol (except the bus[10]), but most existing solutions rely on point-to-point, star, tree or mesh solutions[11]. By far the most common type is the star layout, which is used by WiFi, Bluetooth and cellular networks, and relies on a central hub to relay data packets between clients. The topology choice will affect hardware complexity, redundancy, achievable data rates, latency, flexibility, and protocol complexity.

[10] A bus refers to all devices being connected by a single cable, therefore a *wireless bus* is nonsensical, even though the term is often used in publications when referring to wireless networks for low-level devices.

[11] Some mesh technologies could alternatively be described as a tree, star-tree or hybrid mesh

Reliability

In-orbit servicing has been done only a few times in history [6], and only for very expensive projects like the Hubble Space Telescope[12]. Small satellite projects will not be attractive for servicing in the foreseeable future, which requires all subsystems to function for the lifetime of the satellite. While failure of some components might only result in reduced functionality, a failure in the communication system can render the whole system inoperable. Reliability is affected by various design parameters, including chosen hardware, its integration, network topology, and redundancy. The network should be able to recover automatically from power loss, dropped connections and failed nodes, and the modules could be made configurable over the wireless link to allow some flexibility in case of unexpected failures.

Redundancy

One way to increase reliability is to add redundancy to the communication paths, in order to decrease the number of possibilities where a single point of failure[13] (SPOF) can occur. Ideally, no such points should exist, which can be attempted in various ways. Depending on the network topology, multiple communication paths can be created to transmit data between two endpoints, so that the data can still be received if one of those paths is lost. Even for a full mesh topology it is possible for each node to fail, which can be mitigated by installing multiple communication chips on a single subsystem, in order to switch between them when one of the chips fails.

Redundancy can be achieved more easily with wireless systems, since modules can be added and removed from the transmission medium at will. In order to create true redundancy in a wired system, additional wires between all components would be needed, possibly routed through different parts of the satellite structure. Ideally, these connections should be electrically decoupled, to avoid the possibility of a single short circuit disabling all communications. In wireless systems, redundancy can be achieved by using multiple chips, which can take over communication in case of one of the modules failing.

Cost

Custom made chips and modules are expensive when produced in low volumes, which is usually the case for satellite projects. To keep production costs low, available consumer or industrial hardware can be considered, which will also reduce the amount of testing required to guarantee proper performance. These products also offer a level of integration, documentation and support which is difficult to reach with any custom solution. Moreover, the use of available technologies and protocols will decrease

[12]One of the servicing missions had to be done to fix spherical aberration of the lens system, since the main mirror had not been manufactured correctly. An account of events that lead to this mistake can be found in [7], which serves as a good reminder on how politics, budgets concerns, and human error can endanger mission success.

[13]This means that one malfunctioning part is able to stop the entire system from working

the time and resources needed to achieve proper integration with existing hardware, thereby decreasing development cost.

Integration effort/speed

Wireless solutions aim to streamline the design process and make the development of the modules more independent. Therefore, the integration of the chips into the individual subsystems should be as easy as possible. Drop-in solutions require multiple interfaces for the module to connect to, and the wireless protocols should require minimal configuration. The documentation should clearly state all necessary preconditions, including chip placements and required connections. The same is true for the capabilities of the system, especially achievable data rates, latencies, and energy consumption in various deployment scenarios.

The use of preexisting technologies can simplify these tasks significantly, since extensive documentation is usually available for these types of products[14].

Flexibility

It often happens that requirements for submodules change during the design process. The wireless bus should therefore be easily adjustable, to account for changing data rates, latency requirements, and module placements. It should also be possible to add more modules to the network without extensive reconfiguration.

The technology of most new satellites is based on existing predecessors. The (partial) reuse of the wireless infrastructure in future projects requires the design to be as modular as possible. This could allow replacements of single components or enable the integration of new features. This includes upgrading the hardware chip to enable higher data rates, which should not affect the interface to the submodules.

Maintainability

Even the most well-designed systems occasionally need maintenance, and it should be performed by someone who knows the system exceedingly well. There are cases when the systems designer is not available at the time when maintenance is needed, which requires extensive documentation for many possible scenarios. Both hardware and software should be well-structured to enable quick response to occurring problems. The utilisation of widely used technologies, which increases the likelihood of finding an expert to diagnose and fix the problems, can improve maintainability immensely. Another good practice is to refrain from using very complex solutions, since this usually results in higher error probabilities and more difficult failure diagnostics[15].

[14] For example, the Bluetooth 5 Core Specification [8] contains more than 2800 pages, and the device specification of the nRF52832 chip is 554 pages long [20]

[15] A good mantra can be: *"As complex as necessary, as simple as possible."*

Extensibility/Reusability

The subsystems of a spacecraft become more modular when wireless communication is used, which creates the possibility to reuse components with only minimal design changes. The communication protocols should support reusability/reconfiguration to simplify this process.

It should also be possible to easily extend existing systems with additional modules. This supports a fast paced development process with significant hardware changes, and gives more flexibility for debugging and testing, since debugging nodes can be added to the internal network to monitor and log data of interest.

Radiation effects

The unique nature of the space environment produces conditions that are difficult to predict and counteract. Depending on the orbit height of the spacecraft, different types of radiation can produce various effects. In addition to accelerated deterioration of the hardware, Single Event Upsets (SEU) can induce errors by temporarily altering the state of a transistor, caused by the impact of highly energetic particle radiation. In more serious cases, a parasitic transistor in the CMOS structure can be triggered, causing a Single Event Latch-up (SEL). This state can result in a thermal burnout, if the device is not powered down to reset the device. The possibilities of these effects (especially for satellites operating in higher orbits) has to be considered in the design, and systems should be in place to mitigate these risks.

Temperature and low pressure resistance

On top of the radiation effects present in space, there are also frequent and substantial temperature changes to consider. These are a result of the at times intense solar radiation combined with the slow heat transfer between the environment and the satellite. The thermal balance of the satellite has to be considered for the communication network as well, since very high or very low temperatures as well as fast temperature changes can negatively affect the performance of integrated circuits. One reason for the slow heat transfer is the low atmospheric pressure in low earth orbits, which can lead to outgassing and therefore result in changes to the sturdiness and composition in materials. The effects of low pressure and temperature changes can be studied by using vacuum and temperature chambers during ground tests.

Security

As an aspect that is often disregarded in the space sector (and elsewhere), security concerns should always be considered. There are multiple ways in which an adversary can disrupt the functionality of a wireless system: By exploiting vulnerabilities to modify the architecture and change the functionality of the system, by eavesdropping to steal sensitive data, or by jamming the signals to limit the communication or even disable it completely. While these aspects might seem highly unlikely or even impossible for satellite missions where the hardware is hundreds of kilometres away from

potential adversaries, and the signal strengths present in an intra-satellite network decrease rapidly with the distance, it is still advisable be aware of these attacks. The developed wireless solution might not always be restricted to space applications, but could find terrestrial use in mobile robotics platforms or similar areas, where the chances of being compromised are much higher. It might therefore be desirable to consider future applications as well when designing a wireless communication system for satellites. And since space missions are usually costly, it is worth examining even small risk factors.

3.2 Requirements of the Aalto-2 CubeSat

While this thesis aims to provide a framework which can be applied to many satellite applications, the technology presented here is intended to be part of the Aalto satellite projects, which currently consist of three CubeSats. While Aalto-1 and Aalto-2 are already finished and launched in 2017, Aalto-3 is currently in the early stages of development and will provide a platform for an initial test of the wireless intra-satellite communication system presented in this thesis. To estimate the data rates and latency requirements in Aalto-3, the available values from Aalto-2 are used as an indicator.

Aalto-2 consists of seven subsystems: The on-board computer (OBC), the electric power system (EPS), the sun sensors (SS), the attitude determination and control system (ADCS), the UHF module for ground communications, the GPS subsystem, and the payload. All subsystems communicate with (and only with) the OBC through either I2C, SPI or CAN interfaces. Table 1 shows the data rate and latency requirements between each pair of modules. Since all transmissions happen at

Subsystem	**Packet**			**Data rate** [bit/s]	
	Size [B]	**Frequency**	**Latency**	**Burst**	**Average**
to UHF	4	1 Hz	1 s	4	4
to EPS	74	1 / min	1 s	74	2
to SS	12	1 Hz	0.1 s	120	12
to Payload	522	1 / day	1 s	522	1
to ADCS	94	1 Hz	1 s	94	94
from UHF	10	1 Hz	1 s	10	10
from EPS	90	1 / min	1 s	90	2
from SS	70	1 Hz	0.1 s	700	70
from Payload	348	1 Hz	1 s	348	348
from GPS	68	1 Hz	0.03 s	2267	68
from ADCS	38	1 Hz	0.1 s	380	38
Total				**2267**	**649**

Table 1: Data rate and latency requirements for the Aalto-2 satellite subsystems. All data is sent to/from the onboard computer (OBC). The total burst data rate assumes that no two systems are transmitting at the same time.

predefined intervals, the subsystems can be configured so that no two modules send at the same time. This implies that the minimum required data rate (excluding protocol overhead) has to be greater than 2.3 kB/s to transfer the GPS data to the main module within the specified time period of 30 ms, which is possible through the UART interface. The timeliness is necessary since the GPS data includes the exact time, and accuracy is lost for high latencies. Constant round trip times are preferable and can provide a means to calculate the offset and correct the time to achieve higher accuracy.

All other communications require no more than 0.7 kB/s at any given moment. These values don't include any data from the payload to the UHF system, which can be in the range of a few bytes to several hundred bytes, but usually with no strict timing requirements. These transmissions need to be considered as well when designing the communication system.

However, some of the data transmitted between the subsystems does not necessarily need to be transmitted over the wireless link. This includes e.g. I2C addresses, which are transmitted to the sun sensors before the actual values are retrieved. This data does not change, and could be pre-programmed into the nodes to reduce transmission overhead and latencies. Additionally, periodic data can be transmitted through the use of timers on the wireless modules without requesting it over the wireless link, thereby decreasing the latency.

3.3 Bluetooth Low Energy applicability

Wireless intra-satellite communication can be achieved in numerous ways, some of which are mentioned in chapter 2. In general, there are multiple ways to transmit data over the air: radio frequency (RF), optical, and acoustic. While it would theoretically be possible to use some form of acoustic data transmission even in the vacuum of space by using the satellite structure as a transmission medium, an approach like this faces many challenges: The difficulty of procuring or manufacturing appropriate hardware, the possibility of interfering with other sensors such as accelerometers or gyros, and the problem of multi path signals are just a few concerns associated with such a system. It is therefore advisable to only consider optical or RF systems for an intra-satellite network. Optical transmission systems have already been proven to work reliably for this purpose, as the OPTUS satellite project has shown (see section 2.2). It does not create any interference with other subsystems from an EMI point of view, and the modulation is almost as simple as for wired systems. There are however some drawbacks to this technology, mainly the requirement of line of sight between the emitters and receivers, which requires all of them to be arranged at the same side of the satellite structure. This places some constraints on the design of the satellite similar to employing a wired bus backbone. Other drawbacks of the solution present in OPTUS are the module size/weight and the power consumption, all considerably higher than possible comparable RF based solutions. The biggest challenge for a system based on radio frequency communication is the possible interference with other subsystems such as the ground link or through inducing oscillations that influence sensor measurements.

In small CubeSats such as the Aalto-3 satellite (a first testbed platform for the technology developed in this work), the benefits of RF solutions in terms of lower power consumption and mass, as well as less restrictions in module placement are likely to outweigh the drawbacks of potential electromagnetic interference. This decision is based on the comparison of the OPTOS satellite (section 2.2) with commercially available RF chips, and EM interference was thought to be unlikely, since the output power of the RF chips is low (no more than 1 mW), and the ground link operates on a different frequency band in UHF. Therefore, optical systems were considered less suitable for this specific implementation, even though they have applications in other areas, especially larger satellites.

With RF as the most suitable technology, an appropriate frequency and protocol needs to be chosen. There are a number of existing solutions (some detailed in section 2.1), and it is also possible to develop a completely new solution in a freely chosen frequency band. However, starting from scratch is associated with considerable development effort, since hardware would need to be designed, manufactured and tested. The process of developing such a new system is expensive, time consuming and requires significant work (far outside the scope of a single master thesis). The end product could admittedly be tailored specifically to the intended use case, but would most certainly still lack behind existing products in terms of miniaturisation, efficiency, and reliability. Due to these reasons, the possibility of designing a completely custom solution for this purpose is considered infeasible, and existing technologies are evaluated for their applicability.

There are an incredible number of different RF technologies available, varying slightly in terms of frequency, achievable range, data rates, power consumption, availability, chip size and cost, as well as available documentation. Considering all these factors, Bluetooth Low Energy is considered the best choice for the scope of this thesis, since it is more power efficient than WiFi, faster than ZigBee, and cheaper than UWB. It offers a large selection of available chips, should have sufficient range (detailed in section 3.4), and offers some flexibility due to various possible operation modes, transmit power settings, and variable topology. Most bluetooth chips are shipped with a complete protocol stack, simplifying the adaption for the specific use case and reducing the amount of low level programming. The achievable raw data rate of 1 Mbit/s (2 Mbit/s for Bluetooth 5) should be high enough to cope with the demands of most CubeSat missions. A comparison between Bluetooth LE, UWB, ZigBee, and WiFi is shown in figure 2. The theoretical maximum for the data rate of UWB communication is considerably higher than stated here, but no actual chips exist yet that can achieve these data rates[16]. The choice for Bluetooth LE is mainly justified by the low power consumption compared to UWB and WiFi, and the significantly higher data rate than ZigBee.

The range of typical Bluetooth devices is around ten meters when the signal path is not heavily obstructed, which might be the case inside small satellites. Therefore, tests have to be conducted to validate the choice, which is done through simulations in the next section as well as later in chapter 5 by measurements with test hardware

[16] with a comparable form factor and energy consumption

Protocol	Bluetooth LE	UWB	ZigBee	WiFi
Size	small	medium	small	medium
Data rate	1-2 Mbit/s	6.8 Mbit/s	250 kbit/s	11-54 Mbit/s
TX power	≤1 mW	−41.3 dBm/MHz	≤1 mW	≤100 mW
Dev. effort	low	medium	low	medium
Typ. range	10 m	<100 m	10-100 m	100 m
Frequency	2.4 GHz	3.1-10.6 GHz	0.9/2.4 GHz	2.4/5.0 GHz

Table 2: Comparison between possible technologies with regard to their applicability as a CubeSat communication system.

and a satellite mockup.

3.4 Signal strength simulations

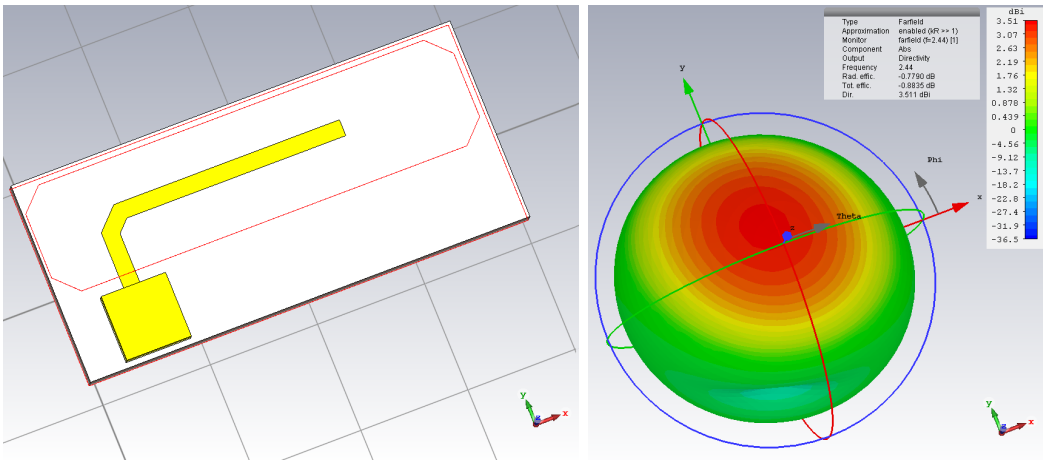
To validate the chosen technology and to assess the constraints on the module placement in satellites, the results of wave propagation simulations are presented, conducted with *CST MICROWAVE STUDIO*, a commercial software suite developed by *CST - Computer Simulation Technology AG* for 3D electromagnetic simulations.

Antenna model

Many Bluetooth Low Energy chips do not use external antennas, but Printed Circuit Board antennas (PCB antennas). They are chosen since they are easy to manufacture and therefore cheap, and provide sufficient gain for most applications. Some designs allow manual adjustments of the antenna resonance frequency by breaking or cutting the end of the antenna to the optimal length, which is helpful during the early development phase and for prototype designs. The antenna used for the simulations in this section is designed in CST Microwave Studio, with the parameters experimentally determined. It has a total length (x-direction) of 22 mm and a width of 4.7 mm. The trace width is 15 mm, and the height 0.2 mm. The antenna is placed on a standard 1.6 mm FR-4 circuit board with a ground plane of 0.035 mm thickness. In order to allow proper wave propagation in all directions the ground plane has a cutout of 40 mm × 10 mm with chamfered corners. The module (including the antenna, the chip and the cutout) is shown in figure 2a.

Printed PCB antenna radiation pattern

Depending on the chosen type and shape of the antenna, the radiation pattern can be fairly uniform in all directions, or heavily directed for more complex layouts. The far-field pattern of the simple L-shaped PCB antenna described in the previous paragraph is given in figure 2b, and is somewhat directed towards the z-axis. This is a suitable configuration for vertically stacked modules, corresponding to the design of most small satellites. The frequency of the simulation is chosen as 2.44 GHz, since the Bluetooth Low Energy communication frequency ranges from 2.402 GHz to



(a) The Bluetooth module. (b) The far-field radiation pattern

Figure 2: Simulation model and results for a Bluetooth chip (6 mm × 6 mm) with an L-shaped PCB antenna (6 mm × 6 mm), designed in CST Microwave Studio. The red lines indicate the ground plane cutout (10 mm × 40 mm). The far-field radiation pattern of the antenna is shown for a frequency of 2.44 GHz.

2.480 GHz. The return loss of the antenna is given in figure 3, with the resonance frequency at around 2.45 GHz and a bandwidth (return loss lower than −10 dB) of approximately 110 MHz.

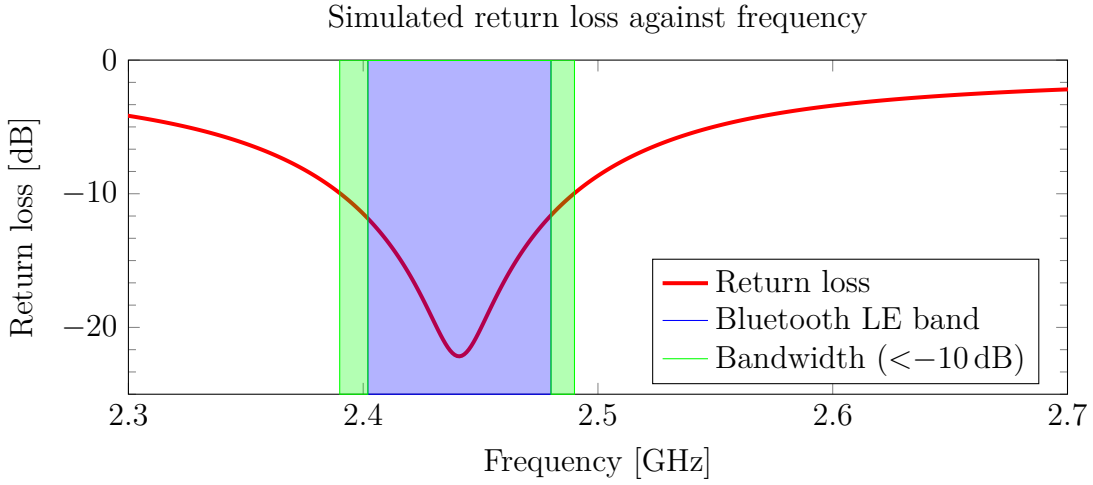
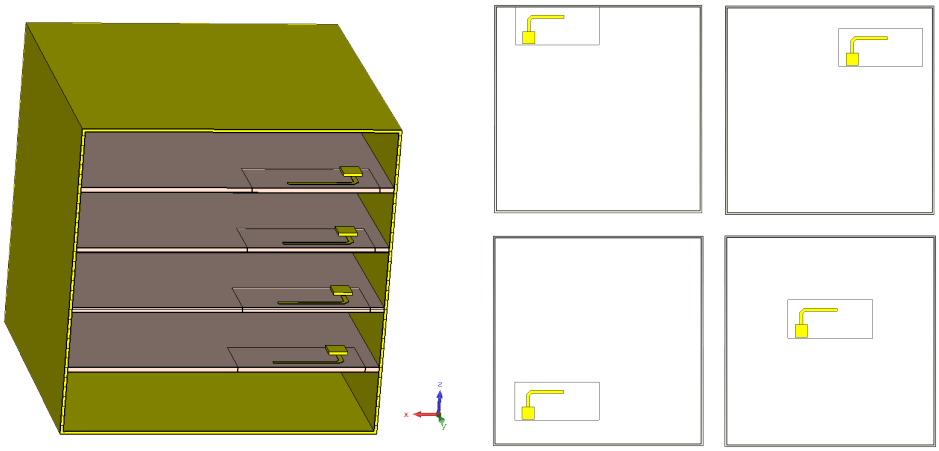


Figure 3: Simulated return loss of an L-shaped printed PCB antenna. The bandwidth is greater than the Bluetooth Low Energy band.

Antenna design is a complex topic, and many optimisations can be done to improve both radiation pattern and gain. The antenna used for this simulation is likely far from ideal, although design considerations were followed from [45] and [46]. For the purpose of estimating the usability of Bluetooth Low Energy for CubeSat missions this design is sufficient, but should be improved for deployment in production hardware. It is possible to use other antenna types as well, including meandered

antennas to conserve space on the PCB, helix antennas for improved performance, or wired monopoles for high efficiencies. The evaluation and design of an appropriate antenna is outside the scope of this thesis.



(a) Vertically stacked. (b) Randomly placed.

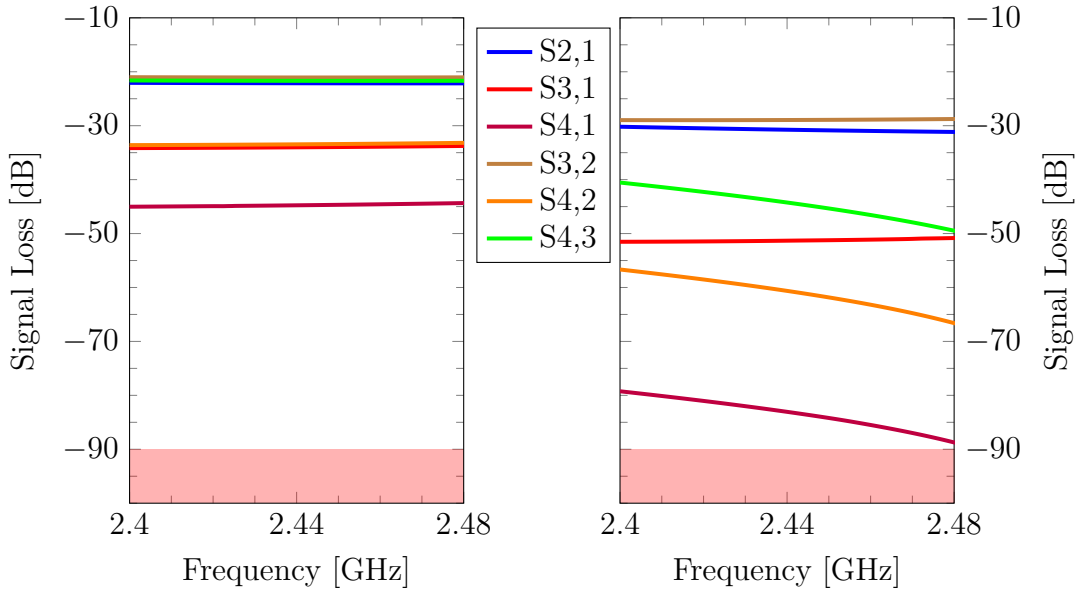
Figure 4: RF simulation model of a small satellite with 4 modules (a) vertically stacked and (b) in random locations, both inside an aluminium box. Shown is the top view of the modules, from lowest module 1 (upper left) to highest module 4 (lower right).

Vertically stacked modules

In order to get a first impression on the signal strength in an environment that resembles a small satellite, a simple model is used. Inside an aluminium cube (2 mm wall thickness, 10 cm edge length) four boards are placed, each with one module in the same location, resulting in vertically aligned antennas, as depicted in figure 4a. The boards are modelled as plain PCBs, similar in substrate and ground plane thickness to the modules themselves. The simulation of the losses between all modules (in the relevant frequency range 2.40 GHz to 2.48 GHz are shown in figure 5a. With a link budget for a typical Bluetooth Low Energy receiver of −90 dBm, the reception on the receiver side is excellent for all modules in this configuration (with the highest loss being −45 dB between modules 1 and 4). The signal loss is fairly constant over the full spectrum of the Bluetooth Low Energy band.

Randomly placed modules

Since the main reason for choosing an RF based technology for a wireless satellite network is the more variable placement of modules, an additional simulation was performed with the modules placed in more random locations. The test included again four stacked PCBs, this time with the modules placed in different locations, as shown in figure 4b. The signal loss simulations in figure 5b indicate that the communication should be possible for all modules, with the communication between



(a) Vertically stacked (b) Random placement

Figure 5: Simulation of the signal loss for different module placements inside an aluminium box. In (a) the modules are stacked vertically (compare figure 4a), while in (b) the modules are placed randomly (compare figure 4b). For the random arrangement the signal loss is in critical range between the farthest modules (1 and 4).

module 1 (at the bottom) and module 4 (at the top) being in a critical region for the higher frequencies of the Bluetooth Low Energy spectrum. The signal loss is at almost −89 dB for a frequency of 2.48 GHz, coming close to the typical link budget of Bluetooth Low Energy chips between −90 dB to −100 dB. These results indicate that the placement of the modules can not be completely random under all circumstances. However, the models used in these simulations are very simple, and more sophisticated tests should be conducted to get better estimations of the constraints on the module placement imposed by the signal strength at the receivers.

Conclusions from the simulation

The results of the conducted simulations support two main conclusions: First, the idea of a fully wireless intra-satellite network deserves to be further investigated, and second, the placement of the modules need to be considered during the design process, since unfortunate positions of the modules might result in high signal loss and consequently poor signals at the receivers. Since the simulations show excellent results for modules placed above each other, it should be sufficient to place all modules on the same side of the satellite. While this somewhat limits the flexibility in the design of each subsystem, it is still superior to a wired solution, which usually consumes significantly more space and requires strict placements for the connectors. It is likely that the results on actual hardware differ to some extent from

the simulations described here, since they contain only simplified models without the circuits, components and materials commonly found in satellites. The proper design of antennas (either PCB antennas or external ones) is another area that could significantly improve real-world results, since antennas with higher gain or increased directivity will result in less signal loss and therefore better performance.

4 Implementation of the concept

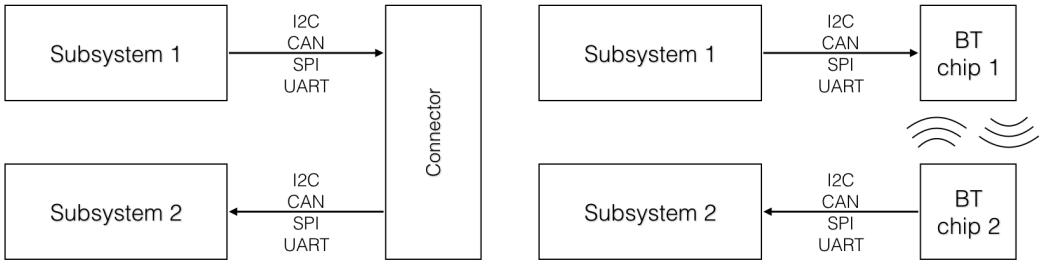
Building on the positive results from the simulations in the previous chapter, a complete concept was developed to use Bluetooth Low Energy as the technology for a wireless intra-satellite network. This involves the selection of appropriate hardware, the development of the software running on the chips, protocols for interfacing with the satellite subsystems as well as integration details to properly incorporate the hardware into the subsystems of the satellite.

4.1 BLE Chip selection

The considerations discussed in section 3.1 provide the criteria for the selection of a suitable Bluetooth Low Energy module. There are two different types regarding their operation: Plain Bluetooth LE chips contain the BLE stack, but no other logic that allows custom features or code to be implemented. Those chips are usually interfaced with through a Serial Peripheral Interface (SPI), and require some other controller for the high level functions (e.g. which device to connect to, how the data is encoded,...). The other type of chip includes a micro controller, which interfaces with the BLE stack and can be programmed to suit the needs of the application. It usually offers a higher number of possible interfaces for data exchange, such as SPI, I2C, TWI, and UART. For the purpose of this thesis the latter is preferred, since it creates the possibility to develop more sophisticated error handling, fault recovery, and higher abstraction, since the Bluetooth protocol can be completely encapsulated in the chip. Such a modular approach has several preferable characteristics regarding the design process of a satellite: The engineers designing the different subsystems do not to be concerned with the actual Bluetooth protocols, only the interface with the chips. Additionally, the Bluetooth chip makes it possible for one subsystem with only an SPI interface to exchange data with another system using UART. This flexibility offers new possibilities for more rapid development and prototyping. It also means that subsystems do not need to be significantly altered when switching to the wireless architecture: If a system did communicate with another through the connector over I2C, it can then be changed to instead communicate with the Bluetooth module over I2C (compare figure 6), with only minimal changes required in terms of software as well as hardware integration. This approach also provides the possibility to introduce redundancy in the design through the use of multiple chips on a single subsystem. An example of such a design is presented in section 4.5. Since the Bluetooth chip includes a micro controller, it is also possible to perform operations that would otherwise require a dedicated chip, for example to retrieve sensor data or measure analog values through the integrated ADC[17]. Due to the advances in miniaturisation, most BLE chips include these micro controllers without a significant increase in chip size or power consumption.

A number of Bluetooth LE chips are available from different manufacturers,

[17]This approach somewhat contradicts the modular design, but might have its value under the right circumstances. There might be subsystems where space or power limitations don't allow for another chip besides the Bluetooth module.



(a) wired (b) wireless

Figure 6: Replacement of the connector with a wireless link. Minimal change has to be made to the existing subsystems.

and many differ only slightly in terms of memory, clock speed, and physical size. With a focus on low power consumption, maximum achievable data rates, receiver sensitivity, available documentation, and availability, the nRF52832 chip produced by Nordic Semiconductor was chosen as a suitable candidate for further testing. Other alternatives include the CC2640R2F from Texas Instruments (out of stock at the time), the MKW41Z256VHT4(R) from NXP, and the BlueNRG-1 from STMicroelectronics. The nRF52 comes with a full Bluetooth 4.2 stack and is compatible with the new Bluetooth 5 specification[18]. A Cortex-M0 CPU running at 48 MHz manages the application as well as the BLE stack, which occupies around 120 kB of the 512 kB flash memory (depending on the configuration). At least 5 kB of the available 64 kB of RAM are also used by the Bluetooth stack. The micro controller offers the most popular bus standards and interfaces, including UART, SPI, TWI, I²C, I²S, ADC, and PWM. Any of the 32 general purpose input/output (GPIO) pins can be used for these buses. The chip contains a total of 48 pins, on a footprint of 6 mm×6 mm[19]. According to the data sheet [20] the chip consumes approximately 5.4 mA when receiving data, and 5.3 mA when transmitting at 0 dBm. The operating voltage is in the range of 1.7 V to 3.6 V. The output power of the signal can be adjusted in the range of −40 dBm to 4 dBm, and the received signal strength indicator (RSSI) can be measured for each connection. For additional power saving a sleep mode with a power consumption of 1.9 µA can be entered. The Bluetooth stack is provided by Nordic Semiconductor as a complied resource (called a SoftDevice) with an API written in C, and is fully compatible with the Bluetooth 4.2 specification. This includes the ability to establish multiple connections as both a master and a slave, allowing the possibility of building Bluetooth mesh networks[20]. Extensive documentation and example code is available through the Nordic Semiconductor info center [48]. The features described in this section make the nRF52832 a good choice as a module for an intra-satellite network. A summary of the nRF52832 is included in table 3.

[18]Only experimental support at the time of this writing, not yet suitable for production use

[19]For the QFN48 package. A WLCSP version with a size of 3.0 mm×3.2 mm exists as well.

[20]The master and slave roles in labeled Central and Peripheral for the Bluetooth protocol, which is described in more detail in section 4.2

Chip	**Name**	nRF52832
	Manufacturer	Nordic Semiconductor
	Processor type	Cortex-M0, 48 MHz
	Flash/RAM memory	512 KB/64 KB
	Size	6 mm×6 mm
	Pin/GPIO count	48/32
	Interfaces	UART, SPI, TWI, I^2C, I^2S, ADC, PWM
Bluetooth LE	**Receiver sensitivity**	−96 dB
	Output power	4 dB to −20 dB
	Bluetooth 4.2	Yes, 1 Mbit/s
	Bluetooth 5	Experimental, 2 Mbit/s
Electrical characteristics	**Standby current**	1.9 µA
	TX current	5.3 mA (at 0 dBm)
	RX current	5.4 mA
	Operating voltage	1.7 V to 3.6 V
	Operating temp.	−40 °C to 85 °C

Table 3: Specification of the nRF52832 chip used for the wireless communication system.

4.2 Overview of Bluetooth Low Energy

The Bluetooth Low Energy protocol is similar to classic Bluetooth in terms of operation, but both technologies are not compatible to each other. A pure BLE device can not connect to a Classic Bluetooth device, although chips exist with support for both technologies. This section will explain the most important aspects of the Bluetooth Low Energy protocol, based on [8] and [49].

Physical Layer

Bluetooth LE uses the 2.4 GHz ISM band from 2.4000 GHz to 2.4835 GHz, which it divides into 40 channels. Three of these channels (37–39, compare figure 7) are used for advertising purposes, which includes broadcasting data and connection setup. The remaining 37 channels are used for transmitting data to connected devices, through a technique called *frequency hopping spread spectrum*. The transmission changes to a different frequency band for each connection event, according to the equation:

$$channel_{n+1} = (channel_n + k) \; mod \; 37.$$

The parameter k is determined during connection setup and changes for each connection. Frequency hopping is used to counteract any radio interference caused by other transmissions on the same frequency band, either by other Bluetooth devices, WiFi communication (also in the 2.4 GHz band), or other sources. The hopping pattern can also be adjusted to exclude channels where high interference is detected.

To modulate the signals and transmit individual bits, Gaussian Frequency Shift

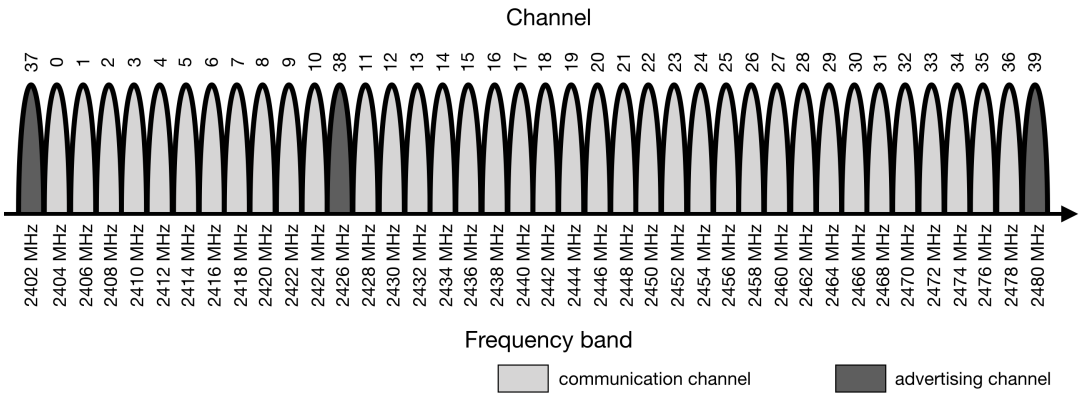


Figure 7: Distribution of the 40 physical channels of Bluetooth Low Energy.

Keying (GFSK) is used, with a modulation rate of 1 Mbit/s[21]. Gaussian FSK is a variant of normal FSK, with an added Gaussian filter to make the frequency transitions smoother, thereby decreasing the out-of band spectrum, and causing less channel interference [50].

Device roles

The Bluetooth specification defines four different roles at the link layer: advertiser, scanner, master, and slave[22] (see table 4). Each device can operate in all modes simultaneously, e.g. be a master in one connection and a slave in another. The advertiser and scanner roles are not associated with another device, but are used for broadcasting (advertising) and to initiate connections (scanning for advertising devices). The master and slave roles refer to a connection between two devices: The master initiates the connection and manages its parameters, while the slave plays the passive part and can only request changes from the master. Usually a device first acts as an advertiser to signal its presence, and then acts as a slave once it is being connected to. In a similar fashion, a master will first scan for a suitable device before establishing a connection. It is however also possible to connect to a device without advertising/scanning, if the unique 48-bit device address is known prior to connection.

Role	**Description**
Advertiser	Broadcast data (e.g. device name, available data, location,...)
Scanner	Scan for advertising packets and devices
Master	Establish a connection and manage its parameters
Slave	Receive connection, request changes to connection parameters

Table 4: The different roles available at the link layer for Bluetooth LE devices

[21]For Bluetooth 4.2. Bluetooth 5 increases the raw data rate to 2 Mbit/s.

[22]The master and slave roles are also called central and peripheral. These terms will not be used in this work, to keep the terminology consistent with other communication protocols.

Advertising and connection establishment

Advertising packets can be used to broadcast small amounts of data to multiple devices, or to send data to a single device without the overhead of a full connection setup. Even though this is useful for beacons, the main use of advertising packets is to notify other devices that a connection can be established with a particular device, and to inform them about the types of data offered by it. Advertising is done on the channels 37–39, in intervals that can be different for each device. At each interval, one packet is sent on each advertising channel in sequence. Some devices only advertise themselves for a certain period of time, for example after the push of a button, others as long as no connection was established, or indefinitely. It is possible to send targeted advertising packets, to signal a particular device that it can establish a connection.

A scanner in turn looks for these packets at defined *scan intervals* for a certain *scan window*. Only one advertising channel can be scanned for each scan window, so three scan intervals are needed to cover all advertising channels. Under normal conditions without heavy channel interference only one scan interval should be needed to detect a device. The three advertising channels are spread across the BLE spectrum to minimise the effects of narrow-band interference.

Once a scanner has obtained the device address of another device (either through receiving an advertising packet or some other means), it can send a connection request to the device (this includes several parameters that are important to the connection, including the frequency hop increment mentioned in section 4.2). To limit unwanted connections, each slave can employ a whitelist with device addresses it will connect to, and only respond to connection request from this list. If the slave wants to accept the connection request, it responds with the connection parameters it supports. The master can then decide if it wants to establish the connection with the parameters offered by the slave. Table 5 describes the three important characteristics that can be negotiated between a master and a slave: The *connection interval* determines at which times the devices will enable their radios to receive or transmit data (connection events). The interval can range from as low as 7.5 ms to up to 4 s, which affects the possible throughput as well as the power consumption[23]. The *slave latency* specifies the number of connection events that a slave can choose to skip before a diconnect occurs. The maximum time between two valid data packets before the connection is severed is described by the *connection supervision timeout*. These parameters depend on the capabilities of the devices (available power, computational resources) as well as external effects like high channel interference. While the connection is established, the connection parameters can be changed by the master, and the slave can request changes (which the master can choose to ignore).

[23]Longer connection intervals allow the slave to sleep for extended periods, since data is only transmitted and received at the beginning of each connection event.

Parameter	Description
Connection interval	Time between two successive connection events (7.5 ms to 4 s)
Slave latency	Number of missed connection events before the connection is terminated
Connection supervision timeout	Time between two valid packets before a disconnect occurs

Table 5: The connection parameters negotiated for each Bluetooth LE connection.

Data structure and transmission

Bluetooth Low Energy is commonly used in consumer products from different manufacturers. To enable interoperability between these devices, the available functionality of each device must be communicated. This is done through protocols, profiles, services, and characteristics. All devices that conform with the Bluetooth specification implement several *protocols*, which describe encoding and decoding, routing and other important aspects for the communication between peers. *Profiles* describe basic operation modes of the devices that are achieved by using the protocols. These can be generic in nature, like the mandatory Generic Access Profile (GAP, handles connections, security, discovery), or describe specific applications (e.g. the Running Speed and Cadence Profile). Many different profiles are defined by the Bluetooth Special Interest Group (SIG), covering the common use cases of Bluetooth LE devices. These profiles consist of one or more *services*, which encapsulate conceptually related data. The Location and Navigation Service, for example, contains the data fields *Location and Speed*, *Position Quality*, and *Navigation*, among others. These data packets are called *Characteristics*, and are the smallest building blocks for data transmission. Each characteristic has its own access permissions (read-only/write), security requirements, and other attributes (e.g. notify on change). All user data transmitted between Bluetooth LE devices is organised in characteristics and services, and 16-bit handles are assigned to them to identify the transmitted data.

Upon connection to a device, a database discovery is initiated, which returns all services and characteristics at the peer. The handles included in this database can then be used to write or read characteristics, or enable notifications for changed values. These notifications and requests are used to transmit data between devices (see table 6). Every device offering a service is called a Generic Attribute Profile (GATT) server, and a device that connects to a GATT server is in turn a GATT client. A client can request data from one or more characteristics, and the server will respond with the data. It is also possible for the server to initiate a data transfer, either as a *notification* without acknowledgement by the client, or as an *indication*, where the client will confirm the successful reception of the packet. It is possible for both the master and the slave to simultaneously be a GATT server and a GATT client.

Type	Initiator	Description
Write	GATT client	The client writes new data to a characteristic on the server (for write enabled characteristics).
Request	GATT client	Request data from a specific characteristic. The server will respond with the most recent data.
Notification	GATT server	Push new data for a specific characteristic to the client. Notifications can be disabled.
Indication	GATT server	Push new data to the client and wait for an acknowledgement packet or a timeout.

Table 6: Description of the different ways to exchange data in Bluetooth LE.

Network topology

Bluetooth LE devices adhere to the star topology, where one master is connected to several slaves. In contrast to WiFi, which is also based on a star topology with a central access point, the slaves connected to the same master are not able to communicate directly with each other. Because Bluetooth was designed with the intention to create wireless communication between two devices, slaves are also not aware of other slaves connected to the same master. Any communication between slaves has to be handled by additional protocols on top of the Bluetooth specification.

The Bluetooth 4.1 specification first defined the possibility for a device to handle multiple roles (master and slave) simultaneously. This creates the possibility to create *scatternets*, where a master can be connected to several slaves, and at the same time act as a slave connected to another master. This allows building multi-hop mesh networks to relay data through multiple nodes, possibly increasing the range of a network beyond the range of a single device. These possibilities are however not the focus of Bluetooth LE, and the protocol is not designed to be efficient in these scenarios.

The Bluetooth specification imposes no strict limits on the number of slaves or masters that a device can simultaneously connect to, but the limited amount of memory and processing power available in most Bluetooth modules creates an upper boundary, as well as timing limits with regard to possibly overlapping connection events. The nRF52832 chip used in this thesis has a limit of 20 concurrent connections (both master and slave roles combined).

4.3 Implementation details and code usage

A code base is made available [52], which implements the core features of the Bluetooth LE protocols in order to provide a simplified API for establishing and communicating over Bluetooth connections. It is written in pure C language to comply with the API provided by Nordic Semiconductor. The code handles automatic device discovery and connection, disconnects, reconnects, and other Bluetooth related events. An easy to use API is provided to send and receive arbitrary user data of

variable length.

General remarks about the code

All functionality of the provided code base is provided through the single header file `manager.h`. Including this file gives access to a variety of functions, which can be used to start or end connections, as well as receive and transmit data. The code is structured and formatted in a way that is found acceptable by the author, without adhering strictly to any particular coding convention.

Testing has been done to make the code stable, but a complete code revision is recommended before deployment. There is work to be done handling some errors, and more functionality could be added, especially in terms of security (currently the Bluetooth connections are not encrypted).

Specific constants and logging

There are a number of constants that need to be fixed at compile time in order for the software to work, and for the devices to interoperate successfully. These constants can either be found in the configuration file `config.h` or in the header and source files of specific modules. A number of constants determine the memory requirements of the SoftDevice, such as the maximum number of possible connections, determined by `MAX_CONNECTIONS_AS_CENTRAL` and `MAX_CONNECTIONS_AS_PERIPHERAL`. These values can be adjusted to suit the needs of the specific use case, and the size and start address of the RAM sections must be adjusted accordingly in the file `multi.ld`.

In order to communicate with one another, all devices must provide services and characteristics with the same UUIDs. These identifiers are declared as macros in `manager.c`, and the 128 bit vendor specific base UUID is chosen randomly as `BB4AFF4F-AD03415D-A96C9D6C-DDDA8304`. The service UUID is set to the also randomly chosen value `0x1523`, and the characteristic UUID for receiving and sending data is `0x1524`.

A number of other values can be configured, such as the advertising and scanning intervals, as well as the output power setting of the radio. Currently these values are hardcoded and therefore constant at compile time, but it is possible to change these values during normal operations in code.

Another important aspect concerns the usage of logging output. The application uses a modified version of the nRF logging library, which can print debugging output to a UART terminal. This is useful during the development phase, as it can give information about specific errors in the submodules. There are 4 severity levels available (`ERROR`, `WARNING`, `INFO`, and `DEBUG`), and the macro `NRF_LOG_DEFAULT_LEVEL` controls which of these are shown in the console. For production code, `NRF_LOG_DEFAULT_LEVEL` and `NRF_LOG_ENABLED` should be set to `0`, since this will decrease code size and lower power consumption. Logging should also be disabled for any measurements regarding power consumption. Future versions of the code can include additional error handling in code, as well as the ability to retrieve logs over an active bluetooth connection for remote debugging.

Example use of the code

The code usage is best explained through an example. The full code sample for establishing a connection and sending/receiving data is included in Appendix A, and the typical procedure will be explained in this section. The connection process is detailed for the devices *Alice* (the master) and *Bob* (the slave)[24].

The first thing that should be done in the main function of both devices is a call to the function `initialiseManager(const char* deviceName)`, which will initialise the Bluetooth stack and other core functionality that is necessary for proper operation (including logging). The provided `deviceName` is a three character long string that will be included in the advertising packets (only slaves advertise), and has to be set accordingly on any other device (*Alice* in this scenario) that wants to establish a connection. *Alice* will therefore call `initialiseManager("ALI")` and *Bob* will call `initialiseManager("BOB")`.

Setting up a connection

Both devices are registering the intent for a mutual connection by calling the function `addConnection(Connection newConnection)`. The function argument is a `Connection` structure, which includes all necessary parameters for setup. The structure has the fields shown in table 7. The field `asCentral` determines if the device acts as a central (master), which will make it responsible for the connection setup and parameter updates. *Alice* therefore sets `asCentral` to `true`, while *Bob* sets it to `false`. The parameter `name` is only necessary for the central, since it will look for advertising packets containing this name. *Alice* sets `name = "BOB"`, while *Bob* can leave this field empty (or set it to `"ALI"` for clarity). In addition to the name, a 48-bit device address is provided, which will ensure that the right device will be connected to. This device address can either be the unique physical device address assigned by the chip manufacturer, or a random address (useful to limit device tracking in certain scenarios). Here, the physical device address is used, so *Alice* will put *Bob*'s address as `mainAddress`, while *Bob* registers *Alice*'s. The `redAddress` is a second device address, which can be used to register a redundant device to connect to, should the main device not be functional. This option exists in preparation for the system discussed in section 4.5. For now, this field is left empty. The `interval` is again only necessary for the master side, and corresponds to the desired connection interval in milliseconds. Values from 7.5 ms to 4 s in increments of 1.25 ms are possible. If the flag `confirmData` is set, then a confirmation is expected every time data is sent to the peer, which serves to guarantee that the data has been successfully arrived. This flag is unidirectional, so both master and slave can enable confirmations for their own data. This is useful in situations where data from one side (e.g. sensor data) does not require guaranteed delivery, but configuration packets to the device should be acknowledged.
Finally, two callback functions can be provided (both can optionally be set to `NULL`), which notify the application about received data and other events related to the

[24]These names are commonly used in computer science, especially related to cryptography

Field	Description
`bool` asCentral	Set to `true`, if the device acts as a master, and is responsible for connection setup.
`const char*` name[1]	Device name to connect to (3 characters long[2]).
`uint8_t` mainAddress[6]	The 48-bit Bluetooth device address of the main device to connect to.
`uint8_t` redAddress[6]	The 48-bit device address of the device to connect to if the main device is damaged/unavailable.
`float` interval[1]	The connection interval for this connection (in the range from 7.5 ms to 4 s in steps of 1.25 ms).
`bool` confirmData	Set to `true`, if a confirmation should be requested each time data is sent.
`DataHandler`[4] received[3]	Function pointer to receive data from the peer.
`EventHandler`[5] eventHandler[3]	Function pointer to receive connection events.

[1] Only necessary for the master role.
[2] Defined by the macro `DEVICE_SHORT_NAME_LENGTH` in the file `config.h`.
[3] Optional, can be set to `NULL` if not needed.
[4] Defined as `typedef void (*DataHandler) (ConstantData data)`
[5] Defined as `typedef void (*EventHandler) (ManagerEvent event)`

Table 7: The data fields of the `Connection` structure that is used to call the function `addConnection(Connection newConnection)`, which will establish a connection to another device.

connection. The `DataHandler` function is called every time data is received from the connected device, and the `ConstantData data` argument of the function will contain the received bytes. In situations where handling received data is not necessary, the handler can be set to `NULL`.

Handling events

The `EventHandler` will be called when important events occur for this connection. At this time, there are five different events, summarised in table 8. The `CONNECTED`

Field	Description
`CONNECTED`	The device was found and a connection established.
`CONNECTION_READY`	The connection is ready to receive/send data.
`DISCONNECTED`	The connection was disconnected.
`DATA_SENT_TO_PEER`	All data is sent and connection is ready.
`DATA_ARRIVED_AT_PEER`	The data was successfully received by the peer.

Table 8: The possible events that the application is notified about in the `EventHandler`.

event will occur when the remote device was found and a connection is established, which does not yet mean that the connection is ready to send or receive data. This state is signalled by `CONNECTION_READY`, notifying the application that the remote database has been discovered, and the connection parameters are set. In case of a disconnect (either initiated by the application or because of a timeout), a `DISCONNECTED` event will occur. If a previous try was made to send data to the remote device but the connection was busy, then a `DATA_SENT_TO_PEER` event will be provided once all data has been sent and the connection is ready to transmit new data. The handler function will be called with a `DATA_ARRIVED_AT_PEER` event if confirmations are enabled and data was successfully transmitted to the remote device.

Connecting to the device

After a connection was successfully registered, it can be started by calling `uint8_t connect(uint8_t ID)`. This will make the device either start advertising (if the connection is to a central) or start scanning (if connecting to a peripheral). Assuming that *Alice* and *Bob* set up their connections correctly and called `connect`, the devices will then automatically establish a connection. *Bob* will advertise itself with the name `"BOB"`, and *Alice* will scan for that name. After *Bob* is found the connection will be established, and all necessary procedures will be handled, including service discovery and parameter negotiation. As soon as both devices are ready, a `CONNECTION_READY` event will be sent to both applications.

Sending and receiving data

It is now possible to send, provide, request, and receive data, by using the functions `sendData`, `provideData`, and `requestData`, as well as the callback `received` from the connection setup. The number of bytes that can be transmitted/received depends on the packet size, and is determined by the macro `NRF_BLE_GATT_MAX_MTU_SIZE` in the configuration file `config.h`. The actual (usable packet size) is 3 bytes less, and is available in code by using the macro `PAYLOAD_LENGTH`. The maximum payload length is 244 bytes. There is no difference (from the perspective of the application) between a connection as a central and a connection as a peripheral once the connection has been established and both devices are ready. This means that the different functions (receive, request, send, provide) work similar for both *Alice* and *Bob*.

`ManagerStatus sendData(uint8_t ID, Data data)` is used to send data to the connected device, where `ID` is the connection identifier returned from `addConnection`, and `data` contains the pointer to the data as well as its length. The return value will be from of the `ManagerStatus` enumeration, which is defined in `manager.h`. A complete documentation of the functions and possible return values is provided with the code base.

`void received(ConstantData data)` is a callback that is triggered when data is received by the device (with or without request). The `data` argument of the function contains a pointer to the received data as well as its length. If confirmations are enabled at the sender, then the confirmation is sent to the peer before the callback is executed.

`ManagerStatus provideData(uint8_t ID, Data data)` does not actually send any data over the connection, but saves the data in case a request for it is issued by the connected device.

`ManagerStatus requestData(uint8_t ID)` sends a request for the last data set provided by the peer through `provideData`. Once the data has arrived from the peer, the `received` callback is triggered with the data. If no data has been provided (yet), the length of the received data will be 0. The return value of this function is `MANAGER_SUCCESS` if the request was sent, or one of the error codes defined in `manager.h`.

It is important to note that the Link Layer of Bluetooth LE already has functionality to notice unsuccessful transmissions of individual packets, and automatically retransmits missing packets. The acknowledgements offered by Indications and Write commands (used by `sendData`) concern higher layers of the stack, where buffer overflows or other problems can occur. The confirmations used here mean that the data has been received by the application. Confirmations will decrease throughput, and should be unnecessary in most situations[25].

Other functions

There is other functionality provided by the code, with the most important functions briefly described here.

`bool isReady(uint8_t ID)` can be used to determine if the connection with the peer is ready to use. The argument `ID` is the identifier returned by the function `addConnection`.

`bool isConnected(uint8_t ID)` can be used to determine if the connection is established with the peer. This again does not mean that data can be sent or received, for this function use `isReady(uint8_t ID)`.

`bool disconnect(uint8_t ID)` can be used to disconnect a registered connection, and put it on hold. This will result in a `DISCONNECTED` event once the connection is terminated. This will reset the connection to the state after `addConnection` is called. Note that this will not stop the other side from trying to reestablish the connection by scanning or advertising.

[25]The tests results in section 5.5 show that packet loss is practically non-existent.

`int8_t getRSSI(uint8_t ID)` will return the current RSSI value for the connection, if it is connected. If the signal strength can not be measured, 0 will be returned.

`float getTemperature()` will return the current temperature of the chip in degrees Celsius. The temperature is measured in steps of 0.25 °C. Note that this function call takes around 50 µs to complete.

`bool startTimer(uint16_t interval, void (*function) (void* context))` will register a function to be called in the specified interval (in milliseconds). It can be used to send periodic data over a connection, or provide new measurement data for requests. At this time, only one function can be registered, and multiple calls to this function should be avoided.

4.4 Hardware integration details

To include the proposed design into an actual satellite, either as a replacement for or an addition to the wired bus, the nRF52832 chip has to be integrated into the subsystems. This section outlines the necessary steps to properly integrate the chip, add the antenna, and connect the communication lines to the rest of the system.

Antenna design and module placement

As previously described the placement of the chips in the satellite is subject to some restrictions regarding the signal strength. For best behaviour and maximised reliability the modules should ideally be placed on the same side of the stacked PCBs. Other configurations will likely work, but require extensive testing under realistic conditions (similar hardware and environment) to guarantee the intended functionality. An additional benefit of vertically stacking the modules is the possibility to decrease the output power of the chips, while maintaining an acceptable signal strength (refer to section 5.2). This will result in marginally decreased power consumption (see section 5.6). Generally, the output power should only be as high as necessary, in order to decrease the possibility of unwanted interference with other equipment or multi-path effects.

The nRF52 Development Kit features the nRF52832 with a printed PCB antenna, and the simulations in section 3.4 are also focused on that type of antenna. It is however possible to use other antennas, for example a Dielectric Resonator Antenna (DRA), also referred to as a ceramic chip antenna. These antennas are a lot smaller in size, are made out of ceramic material on top of a ground plane, and radiate the radio power through semi-transparent walls. They can be very small in size (The Linx ANT-2.45-CHP antenna is only 6.5 mm×2.2 mm×1.0 mm [53]) and have similar gains and radiation patterns as magnetic dipoles [54], but often smaller bandwidth [46]. They are marginally more expensive than a PCB antenna (in the single digit euro range), but can be used where conserving space is paramount. A copper-cleared area and a ground plane is still required, similar to printed PCB antennas.

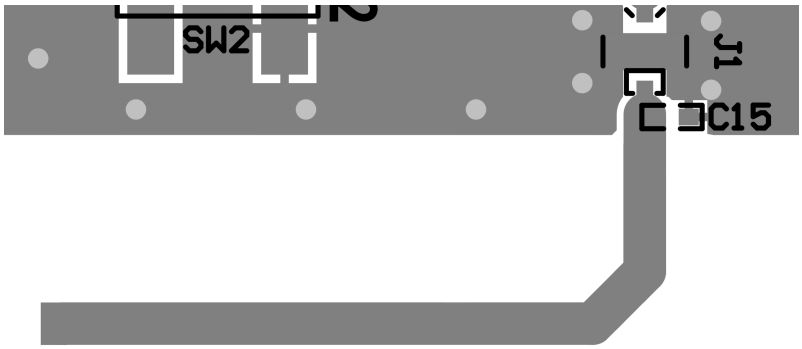


Figure 8: The antenna of the rNF52 development board. It has a size of approx. 18.8 mm×6.4 mm and a trace width of approx. 1.3 mm (exact measurements are not provided by Nordic semiconductor). Image adapted from [56].

The antenna of the nRF52 development board is shown in figure 8. The antenna has a size of about 18.8 mm×6.4 mm[26], a trace width of approximately 1.3 mm, and a chamfered corner to improve the bandwidth of the antenna. The grey area at the top shows the ground plane of the board. The antenna includes a matching network to adjust the impedance of the antenna to 50 Ω, which is done by using an inductance of 3.9 nH in series and a capacitance of 1.2 pF in parallel with ground.

The impedance of the antenna is highly dependent on the surrounding hardware and the environment in which it is used, so retuning will be necessary in the final protoype and deployment hardware. Some additional information regarding the PCB design [57] and antenna tuning [58] is available. The final design of the antenna and matching network and fine-tuning of the system for optimal performance should be done by a radio communications expert, and testing and validation is needed to confirm that the system works as intended.

Necessary components

Nordic Semiconductor provides a reference layout for the successful integration of the nRF52832 chip into production hardware, which is shown in figure 9. The power supply pins 13, 36, and 48 (VDD) each use a capacitance connected to ground as a low pass filter for the input voltage. The input voltage level should be in the range from 1.7 V to 3.6 V. The pins 1, 32, 33, and 46 (DEC1-4) are used for decoupling and are connected to ground via capacitors.
The added impedances L2 and L3 on pin 48 form an LC filter together with the capacitance C10. This filter is needed to power the chip with the internal DC/DC converter (instead of the internal low dropout (LDO) regulator), which decreases the power consumption of the chip under certain conditions. The DC/DC converter reduces the power consumption by more than 50% when the CPU is under load, but requires some current to function, so it is less efficient than the LDO regulator when the CPU is in sleep mode. The nRF52832 can automatically switch between the two

[26] around 0.5 mm measurement precision with a calliper, since no official dimensions are available.

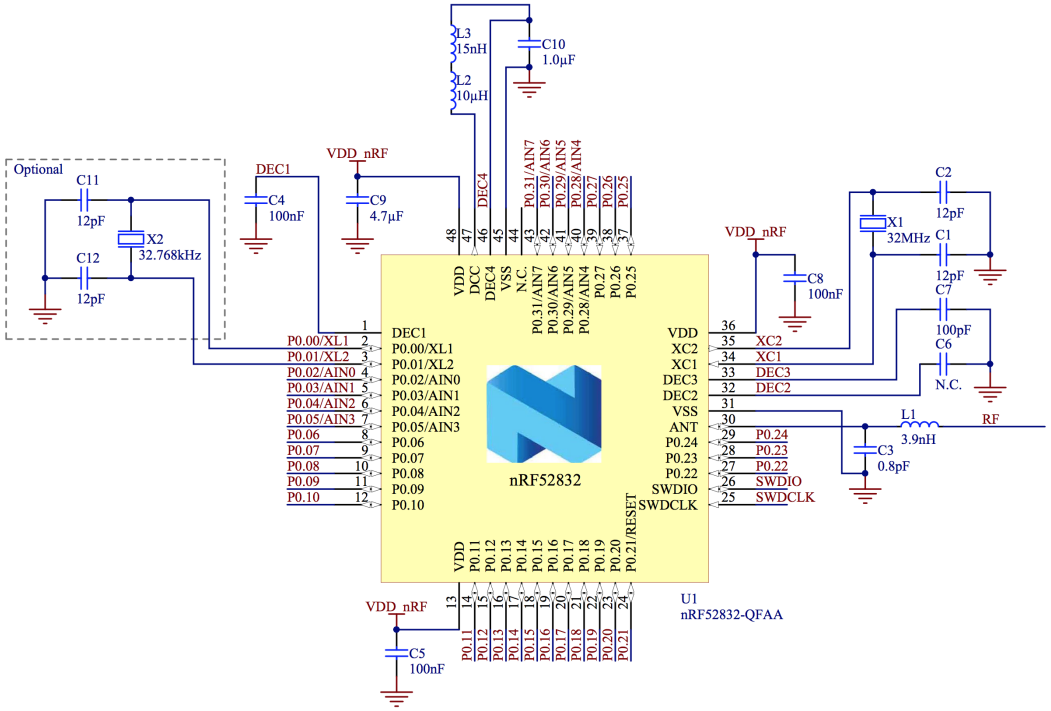


Figure 9: The reference layout for successful chip integration provided by Nordic Semiconductor. A 32 MHz crystal is needed for the micro controller, and an external LC filter is required to use the internal DC/DC regulator (lowers power consumption). Image source: [20]

regulators to optimise power consumption.
The mandatory 32 MHz crystal X1 (together with the capacitances C1 and C2) connected to the pins 34 and 35 is needed to provide a clock signal for the internal synthesised 64 MHz clock of the micro controller. The optional 32.768 MHz crystal X2 can be used as a clock source, and will decrease the power consumption of the device in situations where a clock timer is needed, since the data form the synthesised 64 MHz clock source does not need to be requested from the system.
Finally, the inductance L1 and the capacitance C3 provide the matching network for the antenna pin, to correct its impedance to 50 Ω. Additional information about the hardware integration can be found in the device specification [20] and the reference implementation hardware files [56].

Interfacing with the chip

The previous sections only describe the data transfer between the Bluetooth chips themselves, but not how the chips communicate with the actual subsystems. This task is left mainly to the system designer, since there are many existing technologies that can be used for this purpose, such as UART, SPI, TWI, I^2C, or I^2S. It is even possible to use the ADC or PWM output to directly interface with analog components. All these protocols can be used to communicate with the subsystems

through any of the pins P0.02 to P0.21. While it is theoretically possible to also use the pins P0.22 to P0.31, the usage of those pins is discouraged due to their physical closeness to the radio. The necessary pins that are needed for each protocol can be mapped by setting the associated register values accordingly. A detailed description of the process for each protocol can be found in the device specification [20]. The exact process is specific for each subsystem that connects to the Bluetooth chip, and is therefore not detailed here. The functions described in section 4.3 can be used to set up the necessary interfaces in code. The exact structure of the data transfer (including identifiers and on which connection to send) is also left to the subsystem architect. Very simple protocols with one byte identifying the connection at the beginning of each data packet are possible, or more complex protocols, depending heavily on the exact use case of the system. Example code for the most common protocols is available through the Nordic Semiconductor Infocenter [48]. The details of such interfaces with the satellite subsystems is outside the scope of this thesis.

4.5 A redundant system for increased reliability

The main argument against wireless systems in space applications is their purported unreliability. All integrated circuits can be subject to radiation effects, and each additional chip increases the possibility of a failure during the lifetime of the satellite. In order to provide additional insurance regarding the guaranteed operability of the communication system, redundant hardware can be employed. Redundancy in wireless systems is easier to achieve than for wired connections, since it only requires additional chips, but no added cables or harness. The satellite mass is only increased by the actual chip weight, without the number of active connections between the modules having any impact. It is also easier to switch between the redundant devices in case of a failure, since they all operate on the same communication medium. The Bluetooth LE protocols provide an easy way to detect the failure of a connected device, and measures can then be taken to connect to an alternative.

A concept of a redundant system is shown in figure 10, with two equivalent chips providing mutual monitoring. Both chips are connected to the subsystem through communication lines, but only one connection is active at any given time. Which side is connected depends on the output of a logic XOR, which receives input from both chips. This configuration allows both chips to switch the active chip that communicates with the subsystem, regardless of the state of the other chip. Both devices also have the ability to sense which of the chips is currently active. At all times, one of the chips is in the active role (i.e. communicating with the subsystem and the wirelessly connected peer), while the other module monitors the proper functioning of the active chip. This monitoring is done through a periodic heartbeat signal from the other chip, in order to indicate that it is still operational. Additional wireless sensing can be used to scan for advertising packets from the active chip, assuring that the Bluetooth module is working.

If the active chip ceases to function due to some external or internal failure, then the heartbeat signal and/or the advertising packets will stop, causing the monitoring chip to notice the error. The chip can then either try to reset the active chip or

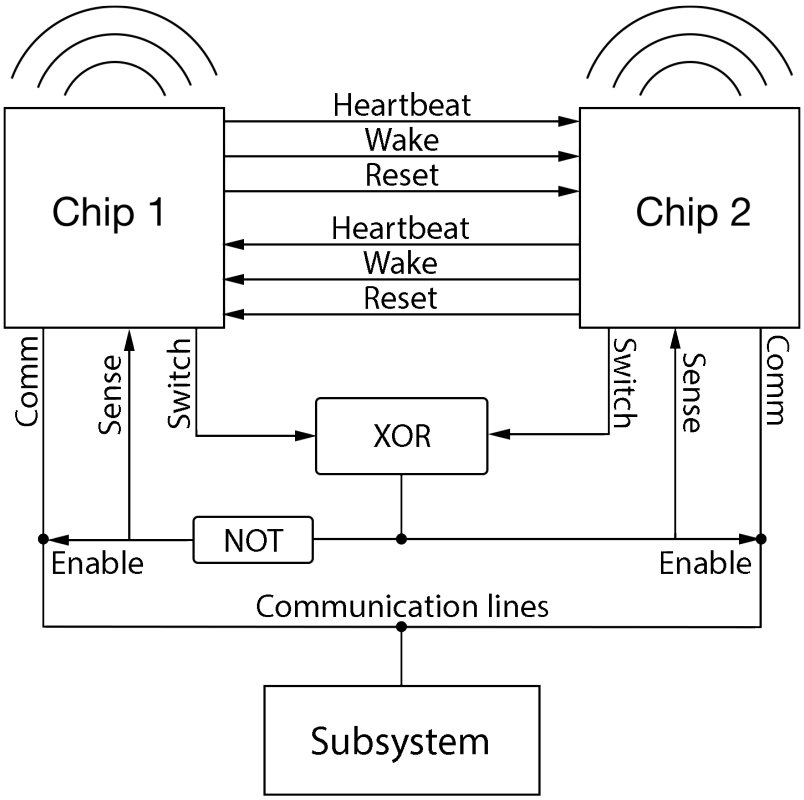


Figure 10: Hardware layout of a self-managing, redundant Bluetooth LE communication system. The chips monitor each other through a heartbeat signal, and can wake up and reset each other when an error is detected. The communication bus with the subsystem can be switched between the chips by either chip, and each chip can sense if it is connected to the subsystem. Additionally, wireless monitoring is used to determine that the connection is working.

transfer the communication with the subsystem to itself by toggling the XOR input signal. It will then send its own heartbeat to the previously active chip, and also start advertising itself.

It is also possible that the currently active chip notices an error that does not immediately cause it to stop the heartbeat signal. If the chip is unable to correct the error in a reasonable time window, then it can send a wake signal to the other chip, and transfer the control over the communication to it by toggling the XOR input pin.

This setup can provide additional confidence that faults in a single chip can be detected and compensated, and that the communication between subsystems suffers only minimal interruptions. This setup is not guaranteed to be devoid of single points of failure, but it provides a way to decrease the likelihood of the communication system failing completely, without the possibility to recover itself.

The Bluetooth protocol supports this sort of architecture nicely, since both chips can advertise themselves with the same name. This allows the remote device to

always connect to the currently active device. It is still possible to notice a change in the configuration, since the device specific peer address of the connected chip will change. Once such a change is detected, debug data or error logs can be requested from the remote devices and be transmitted to the ground station, to help with error diagnostics and future improvements.

As for the connected subsystem, the operation with the redundant system will be similar to a single chip setup. Should an error occur in the redundant system, then the subsystem will temporarily lose the connection, until the active device is switched and the bus is enabled again.

This architecture (while requiring only a few additional components) needs additional space on the PCB it is used on. The chips need separate antennas, since one chip will be transmitting while the other will monitor the connection. The other necessary components are quite small, so the redundant setup will consume approximately twice the space compared to the single chip version. It should therefore only be used in modules that can offer the additional space, and modules that absolutely need the additional reliability, such as the onboard computer. In addition to space concerns, the power consumption of this system will be higher, since the monitoring chip needs to scan for advertising packets, and frequently wake up to check for the heartbeat signal from the active device. The actual power consumption of the system will depend on the scanning interval and the frequency of the heartbeat signal. Lower frequencies will result in lower power consumption, since the chips will sleep for longer periods, but will also increase the response time in case of a failure. A compromise between these criteria needs to be found, depending on the needs of the subsystem.

5 Performance tests with satellite mockup

Several tests were conducted with the nRF52832 chip to test the potential of the bluetooth connection as a communication link between satellite modules. To be a suitable alternative to a wired connection, the Bluetooth module needs to fulfil several criteria: Reliable connection with low packet loss, high throughout, and low latency. The packet loss rate is mainly determined by the received signal strength at the receiver and possible interference signals or multi-path effects.

5.1 Test setup

A simple satellite mockup was used to measure the real-life signal strength of the bluetooth connections, to provide some reference values for possible applications of intra-satellite communications based on Bluetooth LE. The satellite mockup is shown in figure 11 and consists of an aluminium box representing the satellite structure, and four empty circuit boards with full ground planes to simulate several satellite subsystems. The nRF52 development boards (containing the nRF52832 chips) are attached to the circuit boards, where they would be integrated in a fully developed system. The arrangement displayed in figure 11a offers the possibility to measure the signal strength with a single PCB obstructing the signal path (between middle and bottom module), as well as two (top to middle) and three (top to bottom) ground plane obstructions. In the following text and figures the modules will be numbered from 1 to 3, starting from the top (1) to the bottom module (3).

(a) without front cover (b) with front cover

Figure 11: The satellite mockup for the tests regarding signal strength, throughput, and latency. The nRF52 development boards are attached to empty circuit boards, and are enclosed in an aluminium box. Image 11a shows the mockup without the front cover, and the closed case is displayed in image 11b, representing a simplified model of a CubeSat. The modules are labeled as: 1 - top, 2 - middle, 3 - bottom module.

5.2 Received signal strength between modules

Possibly the biggest concern for the deployment of wireless intra-satellite communication is the wave propagation inside the satellite, since the signal path of the waves can be heavily obstructed by the satellite structure and the circuit boards of the subsystems. To simplify the design process of the satellite as much as possible, it would be desirable if the module placement were unconstrained by signal strength considerations, i.e. all modules can be placed in any location of the satellite and still have a sufficient link budget to communicate reliably with every other module. However, the simulations described in section 3.4 already show that such random placement might not be guaranteed to work in every situation. The ground planes in the circuit boards of the submodules result in significant signal loss, and it is therefore important to consider the placement of the modules as an important design parameter during the development of the satellite platform.

Several tests carried out with the satellite mockup described in section 5.1 give more insight into the requirements concerning the signal strength between the modules. The RSSI is given here in the unit of dBm[27], and it can be measured directly on the nRF53832 chip. Separate measurements are available for each Bluetooth connection that is active. For the following measurements, the RSSI values were measured for connections between each pair of modules. Module 1 (working as a master for both other boards) issues requests to both other devices to retrieve the RSSI values on the other side. The data from module 2 includes the data of the connection to module 3, for which it serves as the master. The connection interval for all three connections is 7.5 ms, with a packet size of 27 B and the Data Length Extension (DLE) is disabled. The interval between requests is 250 ms, and the test duration close to 50 s for each test.

To illustrate the behaviour of the RSSI values for ongoing connections, figure 12 displays the RSSI changes over time for the test setup without the front cover (see figure 11a). The signal strength has a high fluctuation, but is in a very acceptable range to provide reliable transmission for all connections. The connection between module 1 and module 3 shows lower RSSI values than the other connections, which is expected due to the higher number of PCB boards between the modules. The signal strength between board 2 and 3 (with one PCB in between) is slightly better than between 1 and 2, where two boards are hindering the signal propagation. The high variation of the RSSI values is likely due to the frequency hopping discussed in section 4.2, with narrowband interference from other Bluetooth devices or WiFi connections in the test area being the cause of the oscillations. The output power of the BLE chips is 0 dBm for this test, and the resulting values are well above the receiver sensitivity of −96 dBm.

More realistic estimates are achieved when closing the aluminium box with a front plate comprised of 3 empty PCB boards (figure 11b). This resembles a typical CubeSat with the hardware enclosed in an aluminium satellite structure. The result

[27] dBm is referenced to the power of 1 mW, with 0 dBm = 1 mW and the general equation $P_{dBm} = 10 \times \log_1 0(\frac{P_{mW}}{1\,\mathrm{mW}})$. However, sometimes the factor 20 is used, for example when exporting data from CST Microwave studio.

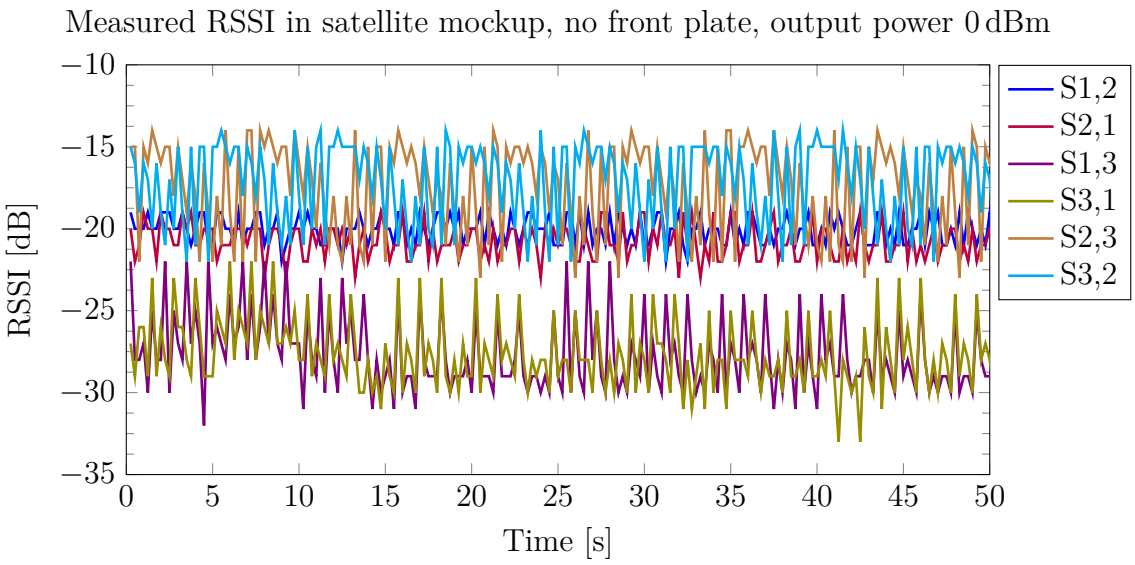


Figure 12: Measured RSSI values for the configuration shown in figure 11a. The legend entries *Sx,y* describe the RSSI values transmitted by y and received by x). The values are well within the −96 dBm link budget (note the scaling of the y-axis)

of multiple tests with varying output power is presented in figure 13. For each possible output power setting of the chip[28] the mean RSSI values are shown separately for each connection. The error bars represent the absolute minimum values measured during the experiment. The signal strength at the receiver falls fairly linearly with decreasing output power, but the fluctuations increase for lower power settings. Even for the lowest output power setting, with RSSI values as low as −74 dBm, the connection is still stable with no packet loss. These measurements show a maximum attenuation of 40 dBm, which is comparable to the results from the simulations in section 3.4, where the attenuation reaches up to 45 dBm for a similar setup. It is however surprising that the satellite mockup shows better results than the simulation, considering that the simulation models feature integrated antennas with a ground plane cutout, which should improve the radiation patterns and lead to less attenuation for the simulation model. This can have multiple possible causes: Higher gain from the PCB antenna of the development board compared to the simulated one, small gaps in the satellite mockup allowing better wave propagation, or other differences between the heavily simplified model and the constructed test environment.

Overall the tests show that reliable communication between the modules is possible for the test setup, and it is reasonable to assume that intra-satellite communication based on Bluetooth LE can work reliably for certain hardware configurations. More test with different configurations, more realistic satellite components, and LE chips integrated into the subsystems can give further details on the feasibility of the concept.

[28] The output power setting of 4 dBm was omitted.

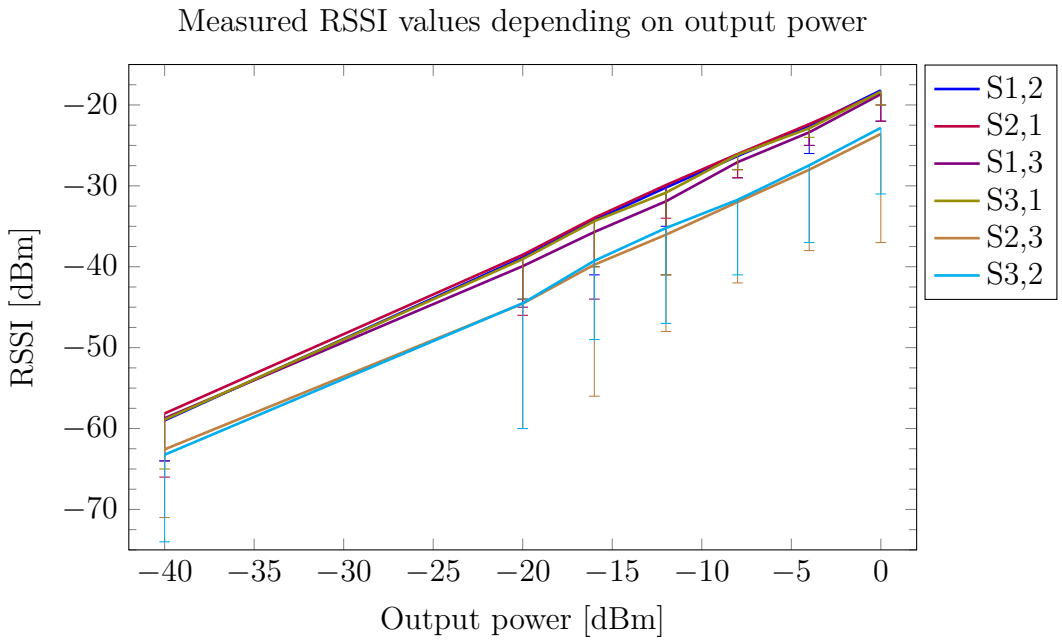


Figure 13: Received signal strength for the connections between the modules as a function of output power. The connection between module 1 and module 3 has on average around 5 dBm lower signal strength than the other connections, and the signal strength fluctuates increasingly for lower output levels and higher attenuation (as evident through the error bars depicting the absolute minimum values).

5.3 Latency of data transmission

The timeliness of the data exchanged between satellite subsystems is one of the major concerns for real time critical systems, such as the Attitude Determination and Control System (ADCS). In order to accurately estimate the orientation of the spacecraft and then control it, data from the sensors has to be gathered without significant delays, and the actuators (e.g. reaction wheels, magnet torquers) need to receive new input in a timely fashion, as to guarantee proper system dynamics and enable accurate pointing of the satellite. A wireless network onboard a satellite needs to fulfil the timing requirements by these systems, if it should be used as a replacement of the traditional wired bus structure.

Generally the latency of individual packets can increase with higher utilisation of the transmission medium, especially in multi-peer scenarios. For Bluetooth LE, a module can only receive from (or transmit to) one other chip at a time, which requires other devices to wait before they can transmit or receive. This is however only true for modules connected to each other, since the frequency hopping technique employed by Bluetooth devices allows several simultaneous transmissions between pairs of devices.

The connection interval negotiated upon connection establishment is the major parameter that influences the latency. To conserve energy, both devices only receive and transmit at the start of each connection event (until no more data needs to

be transmitted) and then sleep for the rest of the connection interval. This means that data requests or transmits occurring during that time window will be delayed until the next connection event. Since the connection interval can be chosen in steps of 1.25 ms from 7.5 ms to 4 s, the desired latency can be chosen individually for each connection. Shorter intervals require the device to wake up more often, and the timeliness of the packets is therefore a tradeoff with power consumption. Additionally, shorter connection intervals result in slightly higher protocol overhead, which decreases the possible throughput of the connection.

Output	$t_{1,2}$ [ms]			$t_{1,3}$ [ms]			$t_{2,3}$ [ms]		
power	**Mean**	**Max**	**SD**	**Mean**	**Max**	**SD**	**Mean**	**Max**	**SD**
0 dBm	22.0	27	4.1	18.0	18	0.0	22.0	27	4.1
−4 dBm	23.0	28	4.1	26.0	26	0.0	21.0	26	4.1
−8 dBm	27.0	28	4.1	18.0	26	0.0	30.0	26	4.1
−12 dBm	21.1	30	4.8	25.3	26	2.3	22.8	81	7.5
−16 dBm	23.0	28	4.1	18.0	18	0.0	25.0	30	4.1
−20 dBm	23.0	28	4.1	18.0	18	0.0	22.0	27	4.1
−40 dBm	22.0	27	4.1	26.0	26	0.0	27.6	60	8.0

Table 9: The round trip latencies between the modules for different output powers. *SD* denotes the standard deviation of the measurements to give an indication on the spread of the measured values.

The latency for a time interval of 7.5 ms is shown in table 9, collected during the RSSI measurements in section 5.2. The latencies are round trip times, with one module requesting new data and then receiving it. It is evident from the collected data that the output power does not alter the latency much, since the connections are stable for all output power settings, and no packets are lost. Another observation can be made about the standard deviations of the round trip times, which are mostly either 4.1 ms or 0.0 ms (with a few exceptions due to delayed packets). These values are due to the timing of the requests in intervals of 250 ms. Since 250 is not a multiple of 7.5 the requests are issued at different times in relation to each connection event. This is true only for the connections between the pairs 1 and 2 as well as 2 and 3, since the request for data on the connection between 1 and 3 is always issued after the data on the other connection is received. This eliminates the time differences and results in constant round trip times for this particular connection. When the connection events are known the round trip times can be reduced, and the variance between the times can be eliminated almost entirely. This creates the ability to use the connection interval as a design parameter to limit the latency effects for data transmissions at predefined intervals.

The packets sent in this measurement include 20 bytes of user data, with a total packet length of 27 bytes. Larger packets can increase the latencies slightly, and multiple simultaneous connections can affect the timeliness (and its variation) as well. It is however possible to achieve point-to-point latencies of less than 30 ms for packets with a size of up to 244 bytes of payload. Figure 14 shows the round trip times of

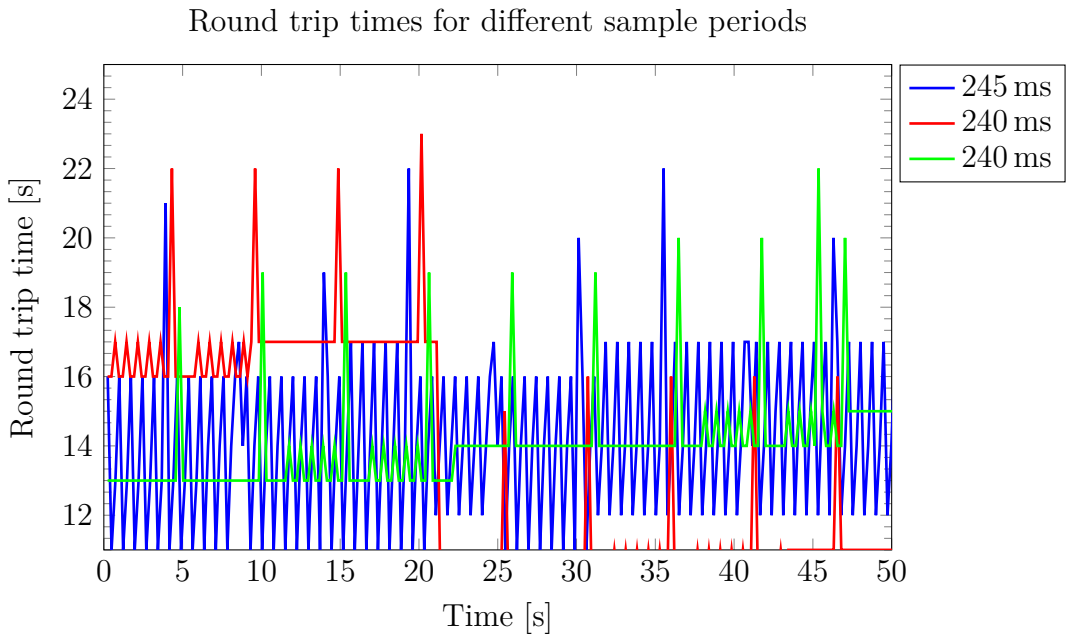


Figure 14: Round trip times for different request periods. When using multiples of the connection interval the round trip times can be kept close to invariant

such a connection (connection interval 7.5 ms). The mismatch between connection interval and sampling period is evident for the connection with 245 ms sample time. The round trip times (while below 20 ms) vary from 11 ms to 16 ms between each request, whereas the latencies for the other two measurements (240 ms periods) are constant for multiple successive requests. There is a slight drift of the latencies in all measurements due to the inaccuracy of the timer library of the nRF52832 chip[29]. When the mismatch from this drift becomes greater than the connection interval, the round trip times drop again to their optimal value.

The additional spikes in all three measurements are due to interferences of other wireless devices in the area (most likely also Bluetooth devices considering the regularity of the peaks).

The results from the tests in this section indicate that communication between real-time critical systems is possible with Bluetooth LE, especially if the connection intervals are matched to the periodic data requests over the Bluetooth link. When properly designed, such a system will even allow time synchronisation between the Bluetooth modules with millisecond accuracy.

5.4 Achievable data rates

While much of the communication between satellite subsystems is infrequent and requires only a few bytes per packet (compare table 1), there often is a need for one connection with higher throughput: The communication between the payload and the

[29]The sample intervals are multiples of the micro controller clock frequency and not exactly 240 ms and 245 ms, respectively.

telemetry, tracking and command system (TT&C) which handles the communication with the ground station. Wether it be images, radar observations or other scientific data, a wireless intra-satellite network must be able to transmit these larger chunks of data fast enough to not create a bottleneck during the times when the satellite is in contact with the ground station and data is retrieved from the payload.

The Bluetooth LE standard (version 4.2) offers a raw data rate of 1 kbps, but the actual data rates with user data can be significantly lower. One important parameter for the throughput of a Bluetooth LE connection is the packet size, called the Generic Attribute Profile Maximum Transmission Unit (ATT MTU), which can be up to 247 bytes. Another factor is the connection interval, since short intervals create more protocol overhead. It is also possible to enable the LE Data Packet Length Extension (DLE), which allows link layer packets of 251 bytes. An overview of the measured throughput for a point to point connection for various settings is given in table 10. The data transfer is done by sending notifications from the GATT server, and the throughput is calculated by measuring the time needed to transmit 1 MB of data. Each test was repeated at least three times to limit the effect of random variations. The measured values come close to the theoretical values reported by the BLE stack specification from Nordic Semiconductor [51]. It is evident that using the DLE is only beneficial for large packet sizes. Moreover, the throughput for a connection interval of 400 ms can be about three times as high as for an interval of 7.5 ms. This represents a tradeoff between throughput and latency, which depends on the needs of the application. The specification reports that data rates of 700 kbps can be achieved for connection intervals of 50 ms, however this configuration has not been tested.

Connection parameters			**Throughput**	
Conn. interval	**ATT MTU**	**DLE**	**Mean**	**SD**
7.5 ms	23 Byte	No	193.03 kbps	0.52 kbps
7.5 ms	23 Byte	Yes	97.68 kbps	0.07 kbps
7.5 ms	158 Byte	No	243.37 kbps	0.05 kbps
7.5 ms	158 Byte	Yes	287.71 kbps	0.04 kbps
7.5 ms	247 Byte	No	235.82 kbps	0.29 kbps
7.5 ms	247 Byte	Yes	262.20 kbps	0.12 kbps
400 ms	247 Byte	No	285.95 kbps	12.48 kbps
400 ms	247 Byte	Yes	753.11 kbps	16.05 kbps

Table 10: Differences in achieved throughput for various connection settings. Short connection intervals limit the possible throughput, and Data Length Extension (DLE) is only useful for larger packet sizes.

The throughput of multiple simultaneous connections has not been tested here, but some guidelines for appropriate settings and timing requirements are available [51]. Another design factor for this scenario is the *connection event length* which determines the time window that is available for each connection to transmit and receive data at each connection event. Giving larger event lengths to certain links will

increase the throughput on that link, while limiting the data rate of the remaining connections. A mechanism called *connection length extension* is available, to allow other connections to transmit and receive when a connection does not need its allocated time slot. Fine-tuning these factors will improve the overall performance of the network, but it should only be necessary for devices with a high utilisation rate.

5.5 Packet error and packet loss

The performance of any communication channel is measured in part by the packet error rate (PER) that can be achieved over it. In the case of Bluetooth Low Energy, packet loss and packet errors are resolved by cyclic redundancy checks (CRC) at the link layer, in combination with infinitely tried retransmits of failed packets. A link layer acknowledgement is sent for each packet[30], and retransmits are requested if the packet CRC is invalid. These mechanisms result either in no packet loss, or the termination of the connection, if the transmission of a packet can not be completed during the supervision timeout (which can be set individually for each connection). This theoretical behaviour is consistent with the results obtained from all previous tests, where all packets are correctly received, and no packet errors or packet loss are evident. Frequent retransmissions due to an unreliable channel will result in lower overall data rates, as well as increased latency for the communication. The signal strength between the modules should be kept at sufficient levels in order to keep these parameters in an acceptable range. According to the tests in section 5.2 RSSI values of $-60\,\text{dBm}$ to $-70\,\text{dBm}$ are sufficient to provide a reliable connection without excessive drops in throughput or increases in latency.

5.6 Power consumption

Energy efficiency is an integral requirement of all hardware components used in satellites, since the power budget of spacecraft is naturally constrained by their ability to harvest energy from their environment, usually through solar panels. CubeSats especially face very stringent constraints on the amount of power that is available to the subsystems, since they have a very limited surface that can be used for solar arrays, small batteries due to weight restrictions, and often periods without sunlight due to their low orbit passing through earth's shadow. Therefore the individual subsystems should be as power efficient as possible, which is especially true for the communication network, since it is one of the systems with the highest utilisation rates. The power consumption of the network depends on a variety of factors: The number of wireless chips used in the satellite, the amount and frequency of the data that is transferred between each system, the transmit power of each chip, as well as the connection parameters for each individual link.

While all these parameters are application dependent and power consumption will therefore vary with the deployment scenario, the power consumption of a single chip

[30]The link layer acknowledgements are independent of the confirmations that are sent for indications, which provide acknowledgements from higher layers.

in various states can give some estimates regarding the overall power requirements of a complete system.

According to the data sheet of the nRF52832 chip, the power consumption during transmit and receive operations is 5.3 mA (at 0 dBm) and 5.4 mA respectively. During times when the module is not actively sending or listening for packets, the micro controller can enter a sleep mode, where it only consumes around 2 µA (depending on several factors, such as RAM retention and the types of wake events possible).

The nRF52 development kit offers the possibility to measure the current of the nRF52832 chip while it is operating. The measurements in this section were performed with the HP 3478A desktop multimeter according to the instructions in the development kit user guide [60]. Logging through the UART connection is disabled during the tests, as it would significantly increase power consumption. The power supply for the chip is held at a constant voltage of 3.3 V.

Operation	**Mean**	**SD**
Sleep mode, no connection	15 µA	1 µA
Advertising, 50 ms	0.34 mA	0.10 mA
Scanning, 175 ms interval, 100 ms window	7.18 mA	0.42 mA
Connected, no transfer, CI 7.5 ms	1.21 mA	0.01 mA
Connected, no transfer, CI 400 ms	0.51 mA	0.02 mA
Transmit, CI 7.5 ms, MTU 23 B	9.29 mA	0.07 mA
Transmit, CI 7.5 ms, MTU 23 B, DLE	4.59 mA	0.05 mA
Transmit, CI 7.5 ms, MTU 158 B	8.33 mA	0.03 mA
Transmit, CI 7.5 ms, MTU 158 B, DLE	5.45 mA	0.05 mA
Transmit, CI 7.5 ms, MTU 247 B	8.13 mA	0.03 mA
Transmit, CI 7.5 ms, MTU 247 B, DLE	5.52 mA	0.06 mA
Transmit, CI 400 ms, MTU 247 B	9.90 mA	0.40 mA
Transmit, CI 400 ms, MTU 247 B, DLE	11.93 mA	0.15 mA

Table 11: Current measurements for various operations, at 3.3 V. Acronyms: CI: connection interval, MTU: maximum transmission unit, DLE: data length extension.

The measured currents for different operations are displayed in table 11. Each measurement was taken over several seconds, with the mean and standard deviation calculated from the samples that were taken at approx. 0.5 s intervals. When the device is in sleep mode, performing no tasks and not connected to another device, it consumes only 15 µA of current, or less than 50 µW of power. The chip can enter sleep mode whenever it is not actively sending or listening on the RF channel. In advertising state, i.e. when trying to establish a connection as a slave, then the power consumption is around 1.12 mW, when advertising every 50 ms. Scanning for a device to connect as a master consumes significantly more energy, since the radio has to be active for longer periods. With a scan lasting 100 ms performed every 175 ms, the average current is 7.18 mA (∼24 mW). Data is only transmitted on Bluetooth LE connections at the start of each connection event, which means that the radio has to be turned on only at these times, and as long as data is

being transmitted. This results in low power consumption for connections with limited data throughput, which is one of the goals of Bluetooth LE. For a device connected to a single peer, the average current is only 1.21 mA with a connection interval of 7.5 ms, and 0.51 mA for a 400 ms interval. Consequently, the currents for devices transmitting and receiving large data packets is significantly higher, and the connection interval, maximum packet size, and the use of the data length extension (DLE) affect the power consumption as well as the throughput. The data in table 11 shows that enabling the data length extension reduces the overall power consumption, in some cases by almost 50 %. This is true only for the measurements for connection intervals of 7.5 ms, for a 400 ms interval the average current actually increases by 20 %. However, table 10 shows that the throughput in this case is more than 2.6 times higher, which implies that the required energy per byte is significantly lower. This measure is more suitable to evaluate connection settings for applications where large amounts of data have to be transferred in an energy efficient manner. The different settings are compared in table 12. It can be seen that enabling DLE increases the

Interval	**MTU**	**DLE**	**Throughput**	**Current**	**Efficiency**
7.5 ms	23 Byte	No	193.03 kbps	9.29 mA	1.27 µJ/B
7.5 ms	23 Byte	Yes	97.68 kbps	4.59 mA	1.24 µJ/B
7.5 ms	158 Byte	No	243.37 kbps	8.33 mA	0.90 µJ/B
7.5 ms	158 Byte	Yes	287.71 kbps	5.45 mA	0.50 µJ/B
7.5 ms	247 Byte	No	235.82 kbps	8.13 mA	0.91 µJ/B
7.5 ms	247 Byte	Yes	262.20 kbps	8.13 mA	0.56 µJ/B
400 ms	247 Byte	No	285.95 kbps	9.90 mA	0.91 µJ/B
400 ms	247 Byte	Yes	753.11 kbps	11.93 mA	0.42 µJ/B

Table 12: Throughput, average current and energy efficiency per byte for different connection settings.

energy efficiency, and the most efficient connection setting is the one with the highest power consumption. While the average current is high for this type of connection, the data is transferred faster. This results in less time needed for the transfer, and consequently less overall energy required per byte (0.42 µJ/B). It might be useful to consider these results when choosing the connection settings for links between subsystems with high data rates, e.g. between the payload and the TT&C module. It should be considered that the results in table 12 do not account for the power consumption of the modules when no data is transmitted, i.e. the test duration is not constant. This limits the comparability of the efficiencies between connections with different connection intervals, since the power consumption in idle mode (no data transmitted) is different in these scenarios (see table 11).

6 Conclusion

Using wireless technologies to connect satellite subsystems can have many benefits, including mass and space savings, more modular and simplified designs, as well as easier extensibility, reusability and testing. While some concepts and implementations exist and have already shown that wireless intra-satellite networks can be operated successfully in space, there is still some concern regarding the reliability of such systems. More testing and validation is needed to overcome these doubts, and easily usable systems need to be developed and standardised to accelerate the adoption of wireless technologies as a viable option for the communication between subsystems in future satellite missions. This thesis provides a comprehensive overview of many design aspects of such systems, and offers a working concept of an intra-satellite network based on the Bluetooth Low Energy protocol. The most important aspects regarding successful deployment of the concept in an actual mission are covered, including the selected hardware, its integration with the subsystems, as well as a software framework to simplify the configuration of the network.

Test with the hardware, the nRF52832 chip from Nordic Semiconductor, show that the provided design performs well in all of the tested areas, including sufficient signal strengths between the modules. Low latency and high throughput data transmission, combined with low power operation and a small integration footprint, show that this concept can successfully be deployed in small satellites, where these requirements are key design aspects. The low-cost modules are a good solution to keep production costs low, and a proposed concept for redundant operation with two chips can provide additional reliability for mission critical communication links.

The results presented in this work justify further tests under realistic conditions, and the Aalto-3 satellite, currently under development at Aalto university, will provide a testbed for the validation of the developed technology. For this purpose, the Bluetooth LE based communication chips will be integrated into several subsystems alongside the traditional wired connections. This configuration will allow comprehensive tests regarding the reliability of the wireless modules, combined with in-situ measurements of signal strength, latency, throughput and power consumption. Additionally, the redundancy concept discussed in section 4.5 will be implemented in at least one subsystem to verify its performance. These tests will hopefully further establish wireless communication links between satellite subsystems as a suitable alternative to the traditional approach, and open up new possibilities regarding the design process of future satellites.

In addition to the integration with the upcoming Aalto-3 satellite, other aspects of this work can be improved in the future. New Bluetooth 5 capabilities, such as the higher data rate of 2 Mbit/s, can be added once the protocol stack moves from an experimental version to a stable release. The variance in the round trip times (section 5.3) can be further investigated, and algorithms can be developed to provide time synchronisation between the modules. Additionally the development of the redundant concept can be completed, with theoretical and practical analysis of the possible points of failure, as well as its performance. Other possibilities to increase the reliability and provide redundancy can be investigated as well. The

provided code can be extended to provide more features, including encrypted links, over-the-air firmware updates and reconfiguration, and improved logging. More tests with a higher number of modules inside a more realistic satellite demonstrator with simulated satellite data can provide additional information and certainty that the system performs reliably for extended periods of time. Solving these challenges, combined with the deployment of the concept onboard of the Aalto-3 satellite, will further establish wireless intra-satellite communication as a technology for future satellite missions.

References

[1] B. Sullivan, CNN, *Shuttle program seeks PC parts on eBay*, Available: `http://edition.cnn.com/2002/TECH/industry/05/16/shuttle.parts.idg/index.html`, accessed April 15, 2017.

[2] G. Xu, S. Li, G. Fan. *Application of wireless satellite bus in micro-satellite design.* IEEE International Conference on Mechatronics and Automation, Changchun, China, August 2009.

[3] O. Ratiu, A. Rusu, A. Pastrav, T. Palade, E. Puschita, *Implementation of an UWB-based Module Designed for Wireless Intra-Spacecraft Communications*, IEEE International Conference on Wireless for Space and Extreme Environments, Aachen, Germany, 2016.

[4] V. Lappas, G. Prassinos, A. Baker, and R. Magnuss, *Wireless Sensor Motes for Small Satellite Applications*, IEEE Antennas and Propagation Magazine, Volume: 48, Issue: 5, October 2006.

[5] J. Tervonen, K. Mikhaylov. *Evaluation of power efficiency for digital serial interfaces of microcontrollers* Univ. of Oulu, Ylivieska, Finland, 2002.

[6] National Aeronautics and Space Administration, *On-Orbit Satellite Servicing Study: Project Report*, Goddard Space Flight Center, U.S., October 2010.

[7] R S. Capers, E. Lipton, *Hubble Error: Time, Money and Millionths of an inch*, printed in Hartford Courant, Connecticut, March 31-April 3, 1991, Available: `http://people.tamu.edu/~v-buenger/658/Hubble.pdf`, accessed March 20, 2017.

[8] Bluetooth SIG Proprietary, *Bluetooth Core Specification v5.0*, December 2016, Available: `https://www.bluetooth.com/specifications/adopted-specifications`, accessed March 23, 2017.

[9] Cisco Systems, Inc., *802.11ac: The Fifth Generation of Wi-Fi Technical White Paper*, May 2017, Available: `http://www.cisco.com/c/en/us/products/collateral/wireless/aironet-3600-series/white_paper_c11-713103.pdf`

[10] N. Baker, *ZigBee and Bluetooth: Strengths and weaknesses for industrial applications*, IEE Computing & Control Engineering, vol.16, no.2, pp 20-25, April/May 2005.

[11] D. Borah, A. Boucouvalas, C. Davis, S. Hranilovic, K. Yiannopoulos, *A review of communication-oriented optical wireless systems*, EURASIP Journal on Wireless Communications and Networking, Vol. 91, March 2012.

[12] M. Habbal, *Bluetooth Low Energy – Assessment within a Competing Wireless World*, Wireless Congress 2012: Systems & Applications, ICM Munich, Germany, November 2012.

[13] C. Knutson, D. Joos, R. Woodings, *Infrared Data Communications in Wireless Personal Area Networks*, Proceedings of the 5th World Multiconference on Systemics, Cybernetics and Informatics, Orlando, Florida, July 22-25, 2001.

[14] J. Zhang, P. Orlik, Z. Sahinoglu, A. Molisch, P. Kinney, *UWB Systems for Wireless Sensor Networks*, Proceedings of the IEEE, Volume 97, Issue 2, February 2009.

[15] S. Roy, J.R. Foerster, V.S. Somayazulu, D.G. Leeper, *Ultrawideband radio design: the promise of high-speed, short-range wireless connectivity*, Proceedings of the IEEE, Volume 92, Issue 2, February 2004.

[16] A. Darif, R. Saadane, D. Aboutajdine, *Performance Evaluation of IR-UWB Compared to Zigbee in Real time Applications for Wireless Sensor Networks*, Mohammed V-Agdal University, Morocco, 2013.

[17] A. Darif, R. Saadane, D. Aboutajdine, *An efficient short range wireless communication technology for wireless sensor network*, Third IEEE International Colloquium in Information Science and Technology (CIST), October 2014.

[18] IEEE 802.15.3a Project Authorization Request, February 2006, Available: `http://standards.ieee.org/board/nes/projects/802-15-3a.pdf`, accessed May 23, 2017.

[19] Decawave Ltd, *Decawave DW1000 Data Sheet*, 2016, Available: `http://www.decawave.com/sites/default/files/resources/DW1000-Datasheet-V2.12.pdf`, accessed May 23, 2017.

[20] Nordic Semiconductor, *nRF52832 Product Specification v1.3*, February 2017, Available: `http://infocenter.nordicsemi.com/pdf/nRF52832_PS_v1.3.pdf`, accessed May 23, 2017.

[21] O. Ratiu, *Wireless Communication Bus For Satellite Applications WI-SAT*, The Software Systems Division (TEC-SW) and Data Systems Division (TEC-ED) Final Presentation Days, ESA/ESTEC Noordwijk, The Netherlands, June 2016.

[22] European Space Agency, *WiSat Project*, Available: `https://indico.esa.int/indico/event/145/contribution/5/material/0/0.pdf`, accessed May 23, 2017.

[23] B.J. Gu, S.-A. Ji, *A study on wireless intra-satellite bus system using UWB technology*, Proceedings of The 2015 World Congress on Aeronautics, Nano, Bio, Robotics, and Energy, Incheon, Korea, August 2015.

[24] H. Guerrero, I. Arruero, M. Álvarez, A. Álvarez, S. Rodriguez, J. Torres, *Optical Wireless Links for Intra-Satellite Communications (OWLS): The Merger of Optoelectronic and Micro/Nano-Technologies*, NanoTech 2002 - At the Edge of Revolution, Houston, Texas, USA, September 2002.

[25] I. Arruego, H. Guerrero, S. Rodríguez, J. Martínez-Oter, J. J. Jiménez, J. A. Domínguez, A. Martín-Ortega, J. R de Mingo, J. Rivas, V. Apéstigue, J. Sánchez, J. Iglesias, M. T. Álvarez, P. Gallego, J. Azcue, C. Ruiz de Galarreta, B. Martín, A. Álvarez-Herrero, M. Díaz-Michelena, I. Martín, F. R. Tamayo, M. Reina, M. J. Gutierrez, L. Sabau, J. Torres, *OWLS: A Ten-Year History in Optical Wireless Links for Intra-Satellite Communications*, IEEE Journal on Selected Areas in Communications, Volume 27, Issue 9, December 2009.

[26] J. Frances, *Internal Wireless Bus for a CubeSat*, Master thesis, Department of Electronics and Telecommunication, Norwegian University of Science and Technology, 2013.

[27] E. Meland, *Internal wireless bus in student satellite experiment*, Project report, Department of Electronics and Telecommunication, Norwegian University of Science and Technology, December 2011.

[28] M. L. Volstad, *Internal Data Bus of a Small Student Satellite*, Master thesis, Department of Electrical Engineering and Telecommunication, Norwegian University of Science and Technology, 2011.

[29] Nordic Semiconductor, *nrf24L01 product specification*, 2007, Available: `http://www.nordicsemi.com/eng/nordic/download_resource/8041/1/52371689`, accessed May 25, 2017.

[30] maniacbug, *Rf24network - network layer for rf24 radios*, 2011, Available: `https://maniacbug.github.io/RF24Network/`, accessed May 25, 2017.

[31] M. Drobczyk, H. Martens, *A study on low-latency wireless sensing in time-critical satellite applications*, IEEE Sensors Conference, Orlando, Florida, USA, October 2016.

[32] P. Moravek, V. Stencel, *UWB network demonstrator for space applications*, Wireless for Space and Extreme Environments (WiSEE), Nordwijk, The Netherlands, October 2014.

[33] S. Xie, G. X. Lee, K.-S. Low, E. Gunawan, *Wireless Sensor Network for Satellite Applications: A Survey and Case Study*, Unmanned Systems, Volume 2, Number 3, 2014.

[34] Nanyang Technological University, *Velox-1 satellite mission*, 2014, Available: `https://directory.eoportal.org/web/eoportal/satellite-missions/v-w-x-y-z/velox-1`, accessed Mai 27, 2017.

[35] H. Guerrero, *Trends and evolution of microsensors: Towards the 21st century transducing principles*, 2nd International Conference on Integrated Micro/Nano-Technologies for Space, Pasadena, USA, April 1999.

[36] P. Cabo, I. Lora, *"OPTOS: A pocket-size giant" (MISION [sic], OPERATION & EVOLUTION)*, AIAA/USU Conference on Small Satellites, 2009.

[37] I. Arruego, 1. Rivas, 1. Martínez, A. Martín-Ortega, V. Apéstigue, J.R. de Mingo, J.J. Jiménez, F. J. Álvarez, M. González-Guerrero, J.A. Domínguez, *Practical application of the Optical Wireless communication technology (OWLS) in extreme environments*, IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE), Orlando, Florida, USA, December 2015.

[38] R. Schoemaker, J. Bouwmeester, *Evaluation of Bluetooth Low Energy Wireless Internal Data Communication for Nanosatellites*, Proceedings of the symposium on Small Satellites Systems and Services, Porto Petro, Majorca, Spain, May 2014.

[39] R. Amini, *Wireless Communication onboard Spacecraft*, Doctoral thesis, Delft University of Technology, The Netherlands, 2016.

[40] M. Bentum, J. Leijtens, C. Verhoeven, H. van der Marel, *Measurements on an autonomous wireless payload at 635 km distance using a sensitive radio telescope*, 33rd ESA Antenna Workshop on Challenges for Space Antenna Systems, Noordwijk, the Netherlands, October 2011.

[41] J. Rivas. 1, J. Martinez-Oter, I. Arruego, A. Martin-Ortega, J. R de Mingo, J.J. Jimenez, B. Martin, *OWLS as platform technology in OPTOS satellite*, International Conference on Space Optics, Biarritz, France, October 2016.

[42] R. Schoemaker, *Robust and Flexible Command & Data handling on Board the Delffi Formation Flying Mission*, Master thesis, Delft University of Technology, The Netherlands, July 2014.

[43] Delft University of Technology, *DelFFi Command and Data Handling*, `http://www.delfispace.nl/delffi/command-and-data-handling`, accessed May 29, 2017.

[44] R. Amini, G. T. Aalbers, R. J. Hamann, W. Jongkind, *New Generations of Spacecraft Data Handling systems: Less Harness, more Reliability*, 57th International Astronautical Congress, Valencia, Spain, 2006.

[45] Nordic Semiconductor, *$\lambda/4$ printed monopole antenna for* 2.45 GHz, White paper, January 2005, Available: `http://infocenter.nordicsemi.com/pdf/nwp_008.pdf`, accessed May 31, 2017.

[46] Texas Instruments, *AN-1811 Bluetooth Antenna Design*, Application Report, May 2013, Available: `http://www.ti.com/lit/an/snoa519b/snoa519b.pdf`, accessed May 31, 2017.

[47] César Martínez Fernández, OPTOS Project Manager, eMail correspondence, May 29, 2017.

[48] Nordic Semiconductor, *Nordic Semiconductor Infocenter*, June 2017, Available: `http://infocenter.nordicsemi.com/index.jsp`, accessed June 2, 2017.

[49] K. Townsend, C. Cufí, Akiba, R. Davidson, *Getting Started with Bluetooth Low Energy: Tools and Techniques for Low-Power Networking*, O'Reilly Media, April 2014.

[50] R. Schiphorst, F. Hoeksema, K. Slump, *Bluetooth demodulation algorithms and their performance*, IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, January 2002.

[51] Nordic Semiconductor, *SoftDevice Specification S132 SoftDevice v4.1*, May 2017, Available: `http://infocenter.nordicsemi.com/pdf/S132_SDS_v4.1.pdf`, accessed June 9, 2017.

[52] Christoph Hagen, *Application code for a wireless intra-satellite network based on Bluetooth Low Energy*, July 2017, Available: `https://github.com/christophhagen/BluetoothSatelliteNetwork`.

[53] Linx Technologies, *Ultra Compact Chip Antenna Data Guide*, 2012, Available: `https://linxtechnologies.com/wp/wp-content/uploads/ant-fff-chp-x.pdf`, accessed June 11, 2017.

[54] Kao-Cheng Huang, D. J. Edwards, *Millimetre Wave Antennas for Gigabit Wireless Communications: A Practical Guide to Design and Analysis in a System Context*, John Wiley & Sons, October 2008.

[55] ESA future missions office, *Technology Readiness Level (TRL)*, June 2015, Available: `http://sci.esa.int/sci-ft/50124-technology-readiness-level/`, accessed June 11, 2017.

[56] Nordic Semiconductor, *nRF52 DK Hardware Files*, April 2016, Available: `https://www.nordicsemi.com/eng/nordic/download_resource/50980/4/25273303/93935`, accessed June 12, 2017.

[57] Martin Børs-Lind, *General PCB design guidelines for nRF52*, Nordic Semiconductor Developer Zone, March 2016, Available: `https://devzone.nordicsemi.com/blogs/870/general-pcb-design-guidelines-for-nrf52/`, accessed June 12, 2017.

[58] Nordic Semiconductor, *Antenna tuning*, White paper nWP-017 v1.0, 2012, Available: `http://infocenter.nordicsemi.com/pdf/nwp_017.pdf`, accessed June 12, 2017.

[59] J.-S. Lee, Y.-W. Su, C.-C. Shen, *A Comparative Study of Wireless Protocols: Bluetooth, UWB, ZigBee, and Wi-Fi*, 33rd Annual Conference of the IEEE Industrial Electronics Society (IECON), Taipei, Taiwan, November 2007.

[60] Nordic Semiconductor, *nRF52832 Development Kit v1.1.x User Guide v1.2*, February 2017, Available: `http://infocenter.nordicsemi.com/pdf/nRF52_DK_User_Guide_v1.2.pdf`, accessed June 23, 2017.

[61] National Aeronautics and Space Administration, *ISS On-Orbit Status Reports 2008*, December 2008, Available: `https://www.nasa.gov/pdf/318312main_reports2008.pdf`, accessed June 25, 2017.

A Appendix A: Example code

Listing 1: Example code for connection as a master

```
#include "manager.h"

uint8_t peerID = MANAGER_CONNECTION_INVALID;

void receivedData(ConstantData data) { /** Do something */ }

void handleEvents(ManagerEvent event) {
  switch (event) {
    case CONNECTED:            /** Do something */ break;
    case CONNECTION_READY:     /** Do something */ break;
    case DISCONNECTED:         /** Do something */ break;
    case DATA_ARRIVED_AT_PEER: /** Do something */ break;
    case DATA_SENT_TO_PEER:    /** Do something */ break;
  }
}

void setupConnection() {
  ConnectionInit toPeer = {
    .name         = "BOB",
    .asCentral    = true,
    .mainAddress  = {0x48,0x69,0x59,0x92,0x5D,0xE6},
    .interval     = 7.5,
    .confirmData  = false,
    .received     = receivedData,
    .eventHandler = handleEvents,
  };
  peerID = addConnection(toPeer);
}

void sendPeriodicData(void* context) {
  uint8_t data[1];
  Data dataToSend = {
    .data = data,
    .length = 1,
  };
  if (isConnected(peerID)) {
    sendData(peerID, dataToSend); /* OR */
    provideData(peerID, dataToSend); /* OR */
    requestData(peerID);
  }
}

int main(void) {
  initialiseManager("ALI");
  setupConnection();
  startTimer(1000, sendPeriodicData);
  loopAndLog();
}
```