UNIVERSITY OF VAASA

FACULTY OF TECHNOLOGY

TELECOMMUNICATION ENGINEERING

Peilin Zhang

WIRELESS SENSOR SYSTEM FOR MONITORING AND CONTROL

Master´s thesis for the degree of Master of Science in Technology submitted for inspection, Vaasa, March 20, 2014.

Supervisor Timo Mantere

Instructor Reino Virrankoski

ACKNOWLEDGEMENTS

First and foremost, I would like to thank Professor Timo Mantere and Senior Researcher Reino Virrankoski for their elaborate guidance and constant support throughout the whole work in this thesis.

Secondly, I would like to thank Principal Lecturer Heikki Palomäki, Researcher Petri Hänninen and Laboratory Engineer Veli-Matti Eskonen who provided me a lot of essential and valuable support for my work.

Besides, I would like to thank everyone in the Communications and Systems Engineering Group, University of Vaasa: Caner Cuhac, Matti Tuomaala, Ruifeng Duan, Tobias Glocker, Tomi Voltti, Markus Madetoja and other people who had contributed and helped in any manner leading to my completion of this thesis.

I also would like to give my sincere thanks to my family who has been giving me a great support in the material and spiritual all the time.

Peilin Zhang
Vaasa, Finland, March 20, 2014.

TABLE OF CONTENTS

SYMBOLS AND ABBREVIATIONS

3D	Three-Dimensional
3G	Third Generation
ACK	Acknowledge Character
ADC	Analog-to-Digital Converter
AES	Advanced Encryption Standard
CAN	Controller Area Network
CRC	Cyclic Redundancy Check
CS	Carrier Sense
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
CSN	Chip Select NOT
DAC	Digital-to-Analog converter
DMA	Direct Memory Access
FIFO	First In First Out
GPIO	General Purpose Input/Output
I2C	Inter-Integrated Circuit
I2S	Inter-IC Sound
IDEs	Integrated Development Environments
IEEE	Institute of Electrical and Electronics Engineers
ISM	Industrial, Scientific and Medical
JTAG	Joint Test Action Group
LR-WPANs	Low-Rate Wireless Personal Area Networks
MAC	Medium Access Control
MC	Main Controller
MCU	Microprocessor Control Unit

MISO	Master In Slave Out
MLME	MAC Layer Management Entity
MOSI	Master Out Slave In
MPDUs	MAC Protocol Data Units
MUTEXES	Mutual Exclusions
OS	Operating System
OSI	Open Systems Interconnection
PHY	Physical
PID	Packet Identification
PRX	Primary Receiver
PTX	Primary Transmitter
PWM	Pulse-Width Modulation
QFN	Quad-Flat No-Lead
RF	Radio Frequency
RFC	Radio Frequency Controller
RPD	Received Power Detector
RTC	Real-Time Clock
RTOS	Real-Time Operation System
RX	Receive/Receiver
SAP	Service Access Point
SCK	Serial Clock
SDCC	Small Device C Compiler
SPI	Serial Peripheral Interface
SOC	System-on-Chip
SRAM	Static Random Access Memory

SSP	Synchronous Serial Port
SYNC	Synchronization
TX	Transmit/Transmitter
UART	Universal Asynchronous Receiver/Transmitter
UAS	University of Applied Science
USB	Universal Serial Bus
WDT	Watchdog Timer
WSN	Wireless Sensor Network

UNIVERSITY OF VAASA

Faculty of Technology

Author:	Peilin Zhang
Topic of the Thesis:	Wireless Sensor System for Monitoring and Control
Supervisor:	Timo Mantere
Instructor:	Reino Virrankoski
Degree:	Master of Science in Technology
Department:	Department of Computer Science
Degree Programme:	Degree Programme in Information Technology
Major of Subject:	Telecommunication Engineering
Year of Entering the University:	2010
Year of Completing the Thesis:	2014 **Pages:** 101

ABSTRACT

With the fast development of wireless sensor network (WSN) technology, a large number of applications have been widely used over the past few years. As a matter of fact, wireless monitoring and control system is unavoidable one of the applications that consist of WSN nodes. A generic, modular and stackable WSN node, named UWASA Node has been developed by the University of Vaasa and Aalto University lately. Besides, SurfNet node, developed by Seinäjoki University of Applied Science, is designed as low-power consumption, high-data rate, small and powerful sensor node that is suitable to implement the monitoring and control tasks under multiple conditions.

In this work, a wireless sensor system for monitoring and control is integrated and developed by one UWASA Node, one Linux board, and SurfNet nodes. Firstly, the basics of WSN including IEEE 802.15.4 and ZigBee standard are introduced. Secondly, a new design and development of the hardware and software for the wireless sensor system is explained in detail. After that, several experiments are performed to verify the system performance due to the limited computational and power source of the sensor nodes in the WSN. In one word, this developed wireless sensor system provides a wireless solution for remote monitoring and control of the deployed environment.

KEYWORDS: WSN, UWASA Node, SurfNet, IEEE 802.15.4, ZigBee, Remote Environmental Monitoring and Control

1. INTRODUCTION

Wireless sensor network (WSN) has been developing rapidly during the latest decade. Computer science, automation technologies, radio frequency (RF) technology, electronics and other related techniques have contributed extensively to the development of WSN technology. Generally, a wireless sensor node is a device equipped with at least microprocessor, radio transceiver, memory, power source, analog-to-digital converter (ADC) and one or multiple sensors. WSN means a wireless network organized by a large number of wireless sensor nodes disposed in the environment, which can process data, gather information and communicate with each other directly or over multi-hop paths within the network. In a WSN, wireless sensor nodes equipped with processors enable advanced distributed data processing, controlling and other intelligent operations in the network. Thus, WSN enables access to harsh places or environments where are impossible to set up the cables. Consequently, it is quite suitable to apply a wireless sensor system to obtain reliable data from the wireless network for the purpose of monitoring and control.

Recently, Communications and Systems Engineering Group in University of Vaasa, jointly with Aalto University, has created a generic software and hardware architecture for wireless automation, namely, UWASA Node, whose feasibility has been verified by building five different pilot applications in different areas: industrial environment, wind turbines, distributed energy production, greenhouse and cattle house. Additionally, the business potential of the system and the ways to commercialization were also highly considered in the research work. (Virrankoski 2012: 1.)

In this work, we utilize UWASA Nodes and SurfNet nodes. By using this architecture, we build a wireless sensor system to monitor circumstances inside a building. There are two main challenges to overcome in the system development: the first is how to construct the WSN of the system for the purpose of environmental monitoring. The second is how to manage the communication between WSN and the user end. SurfNet nodes are performing the measurements and transmitting the data to UWASA Node, which acts as a gateway to the embedded Linux server. These nodes are equipped with humidity and temperature sensors for collecting data. Since the sensor nodes have the scarce power resources, the energy efficiency plays an important role in the system development. Secondly, the compatibility between UWASA Node and SurfNet nodes is enabled. The UWASA Node communicates with Fox G20 embedded Linux server, and acts as a gateway between the WSN and the user end. Consequently, the wireless sensor system for the monitoring of environmental circumstances inside a building can be established.

This thesis is divided into six chapters. In Chapter 2, WSN basics are introduced. It includes the basics of WSN and wireless sensor nodes, and some discussion about wireless automation applications, WSN protocols, IEEE 802.15.4 standard and ZigBee protocol stack. The hardware architecture of the developed monitoring system is presented in Chapter 3 and the software architecture in Chapter 4. The performance of the developed system is evaluated through experiments in Chapter 5. Finally, conclusions and some directions to the future work are presented in Chapter 6.

2. WIRELESS SENSOR NETWORKS

This chapter introduces the background, basics and principles of WSN firstly. Some discussions about applying WSNs to wireless automation are also presented. Then, the network protocol stack, IEEE 802.15.4 standard and ZigBee protocol stack, which have been proposed for WSN are explained briefly. Finally, it focuses on a number of issues about wireless sensor node that also relates to the following chapter, where a general explanation of the sensor nodes in this research will be given.

2.1. Wireless Sensor Networks

In the last decades, the progress of microelectronics, computing, and wireless communication technologies has facilitated a rapid development of multi-functional sensor nodes. It enabled the functions of, for example, data collection, processing and wireless communications, to be implemented well in the tiny-size sensor nodes. WSN could be a multi-hop and self-organized network, which consists of a large amount of spatially distributed sensor nodes. It aims to monitor physical conditions by cooperatively collecting data, possibly processing it in the network and then transmitting it to the users. In this general concept of WSN, it implements the data flow by collecting, processing and transmitting.

Usually the WSN consists of wireless sensor nodes, a sink which acts as a network gateway, and a user end, as shown in Figure 1. In the monitoring area, the sensor nodes are able to collect and transmit data to sink node by a single

hop or multi-hop manner. The sink node, as a gateway of the WSN, communicates with the user end via some other communication architecture, such as the Internet or satellite communication.

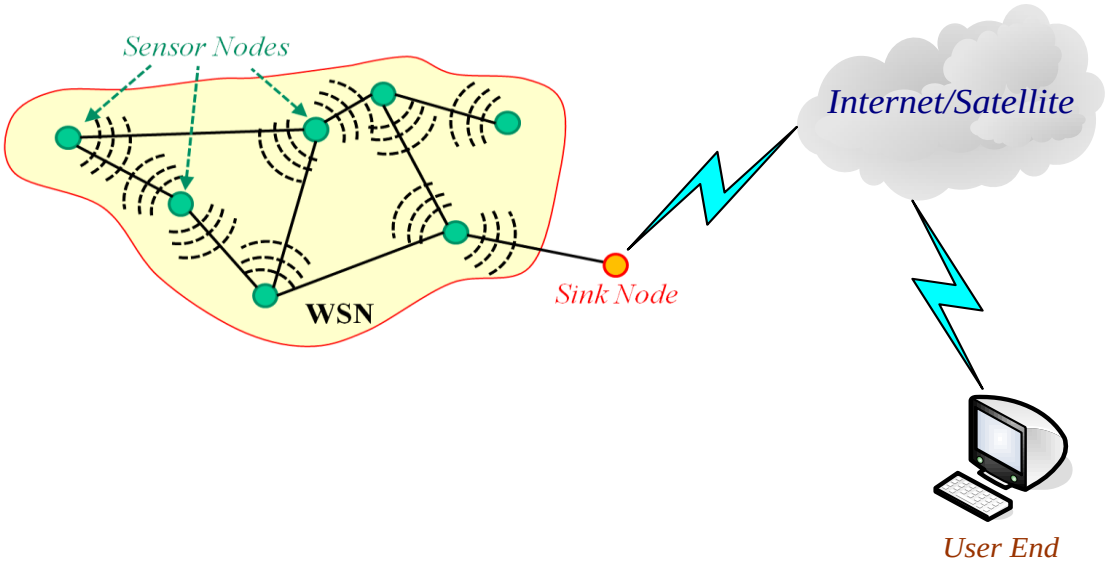


Figure 1. WSN network architecture (Wikipedia 2012).

According to the essence of a WSN, the main characteristics of a WSN include (Wikipedia 2012) the following contents:

- Power consumption constraints for nodes using batteries or energy harvesting;
- Ability to cope with node failures;
- Mobility of nodes (might be either mobile or static);
- Heterogeneity of nodes;
- Scalability to large scale of deployment;
- Ability to withstand harsh environmental conditions;
- Ease of use.

To implement these characteristics, there is a quantity of challenges for WSNs nowadays. On the one hand, the energy supply for the WSNs node is a significant constraint which directly determines the lifetime of the sensor node and even the whole system. Usually one node can only be equipped with a limited power resource because of its tiny size. The way to maximize the energy efficiency and the development of innovative energy supply methods are some of the key challenges in the WSN development.

On the other hand, designing innovative mechanisms, new architectures, as well as protocol concepts for a wireless network is also challenging because of the scarce resources of the wireless sensor nodes.

Besides, distributed operations are carried out quite commonly in WSNs, for example, in weather monitoring applications, the data on weather condition should be obtained effectively and in time through the server via Internet. Thus, the development of distributed algorithms feasible for WSNs is a continuous challenge.

2.2. Wireless Automation

Wireless automation is one of the main developing areas in which WSN applications have a great business potential. If WSN operates as a part of the wireless automation system, it must fill the system performance requirements. The system performance can be assessed in terms of data transmission rate, sample rate, communication reliability and power efficiency. In wireless automation, the sensor nodes must be able to fill the performance requirements

which are rising from the specifications of the automation system. It usually means that the same level of energy efficiency as the one achieved in some monitoring applications cannot be achieved (Virrankoski 2012: 3).

2.3. Network Protocol Stack

Based on the intensive research, several network protocol stacks have been proposed within a couple of years. As Figure 2 shows, the network protocol stack of WSN includes physical layer, data link layer, network layer, transport layer and application layer. The physical layer provides the signal modulation, wireless transmission and receiving techniques. The data link layer is responsible for data framing, frame detection, medium access control and error control. The network layer takes charge of the generation and selection of routes. The transport layer accomplishes the transmission control of data flow. In the application layer, various kinds of application software can be implemented. Besides, the protocol stack has also three management platforms with power, mobility and task, which correspondingly manage the power consumption, movement and task scheduling of the nodes in WSN. (Akyildiz, Su, Sankarasubramaniam & Cayirci 2002: 104–105.)

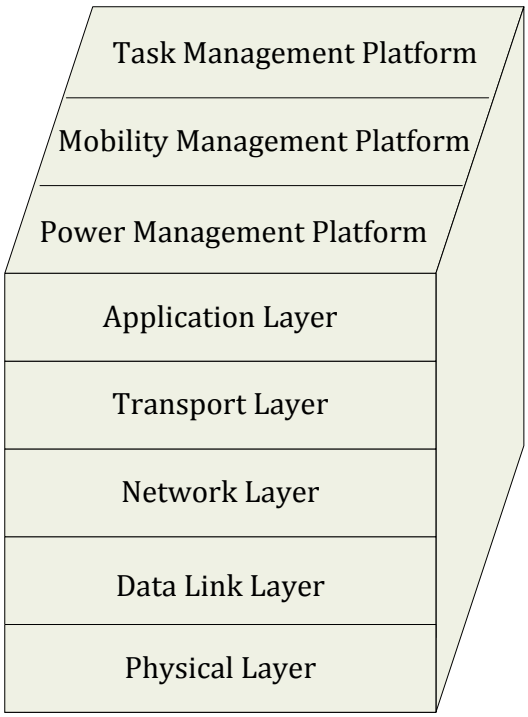


Figure 2. Network protocol stack (Akyildiz, Su, Sankarasubramaniam & Cayirci 2002: 105).

2.4. IEEE 802.15.4 and ZigBee

2.4.1. IEEE 802.15.4

There are some common standards used in WSN communications such as ZigBee, 6LoWPAN, WirelessHART, all of which are based on the same physical (PHY) layer and medium access control (MAC) standard IEEE 802.15.4.

IEEE 802.15.4 is the standard that specifies the physical layer and media access control for low-rate wireless personal area networks (LR-WPANs). It

emphasizes the low cost communication of short-range devices with little underlying infrastructure, intending to reduce power consumption.

In terms of transmission, the network topology in IEEE 802.15.4 is defined as Star or Peer-to-Peer Topology as illustrated in Figure 3. The sensor nodes can form a star topology when the transmission ranges are large enough so that they can transmit their data directly to the sink node. However, a multi-hop communication is a more common case of sensor network transmission. In mesh topology, sensor nodes must not only not only measure, process and transmit their own data, but also serve as relays for other sensor nodes. (IEEE Standard 802.15.4 2011: 8–9.)

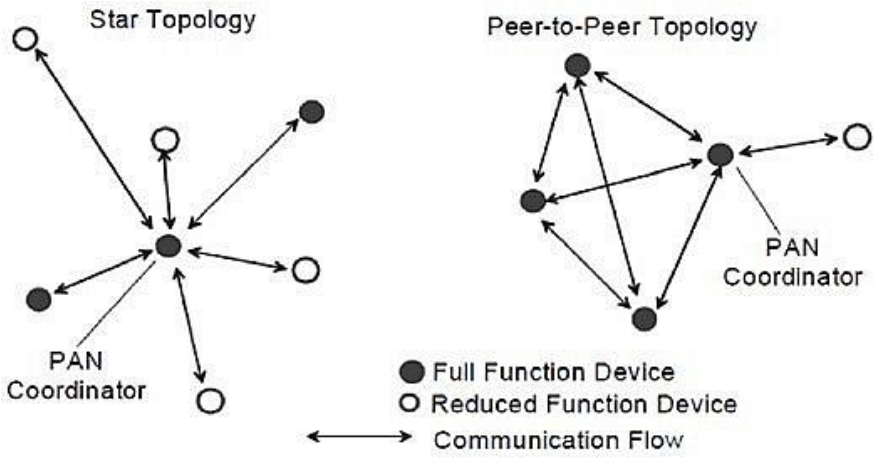


Figure 3. IEEE 802.15.4 network topologies (IEEE Standard 802.15.4 2011: 7).

The IEEE 802.15.4 architecture is defined in terms of PHY layer, MAC layer, and upper layers. The definition of the network layers is based on the open systems interconnection (OSI) model, which is presented in Figure 4. The physical layer

has two services: data service and management service. The data service focuses on the transmission and reception of physical layer protocol data units across the physical radio channel. The MAC layer also provides these two services. The data service enables the transmission and reception of MAC protocol data units (MPDUs). The management service connects to the MAC layer management entity (MLME) service access point (SAP) across the PHY data service. (IEEE Standard 802.15.4 2011: 10.)

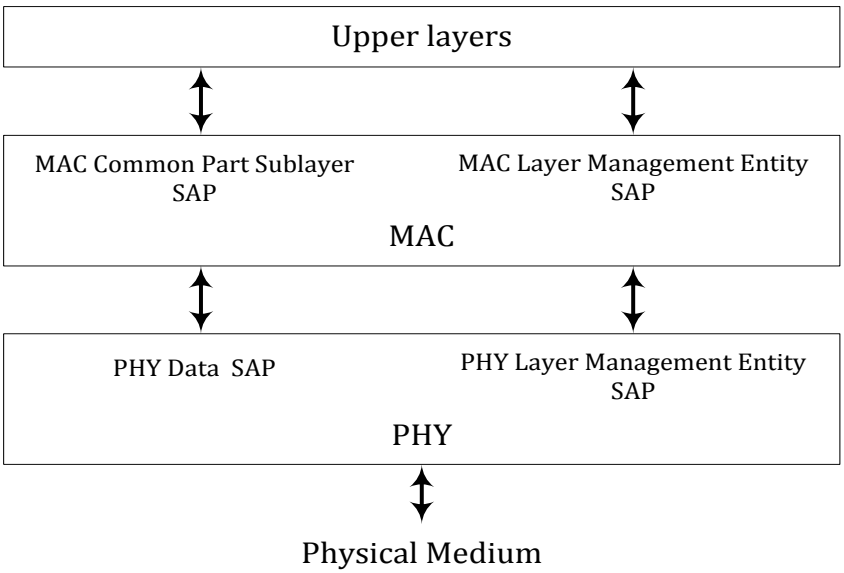


Figure 4. IEEE 802.15.4 network architecture (IEEE Standard 802.15.4 2011: 10).

2.4.2. ZigBee

Based on the IEEE 802.15.4 standard for personal area networks, ZigBee, a rising specification, is featured as low-complexity, low-power consumption, low-rate, and low-cost standard for a suite of wireless network communication

protocols. It has been recognized as a most common network protocol used in WSN nowadays. It is comprised by the physical layer, MAC layer based upon IEEE 802.15.4 standard, network layer and application layer. Figure 5 shows the ZigBee protocol stack.

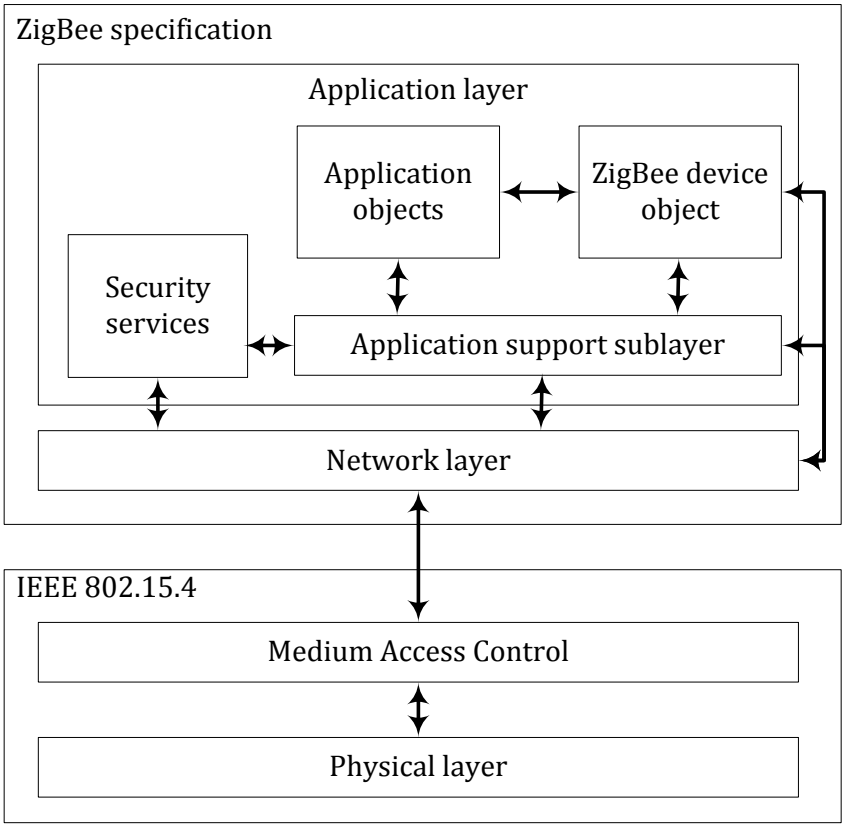


Figure 5. ZigBee protocol stack (ZigBee™ Alliance 2008: 2).

Based on IEEE 802.15.4 standard, ZigBee operates in three different industrial, scientific and medical (ISM) radio bands, which specifically are 868 MHz in Europe, 915 MHz in the USA and Australia and 2.4 GHz in most countries worldwide (ZigBee™ Alliance 2008: 111). Data transmission rates vary from 20 Kbps in the 868 MHz frequency band and 40 Kbps in the 915 MHz frequency

band to 250 Kbps in the 2.4 GHz frequency band. (IEEE Standard 802.15.4 2011: 14.)

The ZigBee network layer typically supports star and tree networks, and also generic mesh networks. In a ZigBee wireless network, the link address can be short address (16-bit) or long address (64-bit). Accordingly, the network has a capacity of 2^{16} and 2^{64} of devices, respectively in maximum number. (ZigBee™ Alliance 2008: 3.)

ZigBee utilizes the carrier sense multiple access with collision avoidance (CSMA/CA) in order to avoid the collision between radio carriers. In addition, to ensure the reliability of data transmission, ZigBee also establishes a complete communication protocol.

For the purpose of ensuring the security of the data communication between ZigBee devices, ZigBee uses advanced encryption standard (AES) encryption algorithm with a key size of 128, to process the encryption of the transmitting data information.

2.5. Wireless Sensor Nodes

A wireless sensor node is an embedded device with, at least, a microprocessor, limited power source, some memory and one or several sensors. Sensor nodes are not only responsible for the data collection and processing, but also for the storage, management and fusion of the data from other sensor nodes. Therefore, sensor node can be treated as either a terminal or a router in a WSN.

In a WSN, there can be a great many of wireless sensor nodes, which can be all similar. Alternatively, there can be some different specified types of wireless sensor nodes or actuators, which have more resources than other ones, depending on the particular needs. Different tasks can be defined for different nodes. If it only requires measurement and data transmission, it is easier to make smaller and more energy efficient nodes. In addition, the software architecture does not need to be that complicated either. However, if it needs to make nodes that are more efficiently capable of data processing and data storing, then the size tends to be bigger as well as the energy consumption, and the software architecture becomes more complicated.

Generally, one of the sensor nodes acts as a sink node, which is a gateway to other parts of the communication system. Then the further storing and processing of the measured data is performed in other parts of the system. For example, in this wireless system, the UWASA Node, which is connected to the embedded Linux server, acts as a gateway to the WSN, in other words, as a sink. Then the Linux server stores the data, which can be processed further and transmitted to other parts of the system, from the embedded server, for example to the user end.

In this research, the generic UWASA Node, which acts as a gateway between the Linux server and the WSN that consists of SurfNet nodes, is integrated to work with the FOX board G20 and the SurfNet nodes. The related hardware architecture and the software development are about to be explained further in the following chapters.

3. HARDWARE ARCHITECTURE

System hardware design must be capable to fill the application requirements. It must also stay at a reasonable price level. In this chapter, the designed system hardware architecture is described. The hardware components and their constitutions functioning in a WSN node, for example the UWASA Node or SurfNet node, are further introduced.

3.1. The UWASA Node

The UWASA Node is a modular and stackable wireless sensor platform. Its generic, modular architecture enables fast adaptation for development of different wireless automation applications by providing a means of stacking relatively simple slave boards (Virrankoski 2012: 32). It mainly consists of a power module, a main module and a number of slave modules. Figure 6 shows more details about the generic node stack.

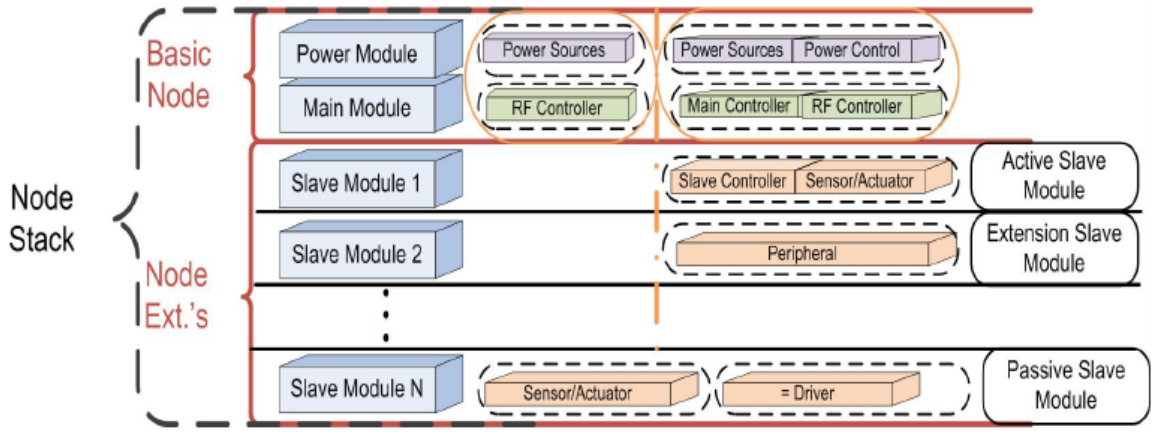


Figure 6. One possible stackable node architecture (Virrankoski 2012: 33).

The power module provides management and distribution of power for the node. The main module, which contains a basic processing unit, memory and radio frequency hardware, takes charge of processing, computing and communication. Basically, it is composed of main controller (MC), RF controller (RFC), and main controller MC-RFC interface circuit. Depending on the target of the application design, the slave modules can be single or multiple, which ensures the UWASA Node to maintain the good adaption and extension interfaces for both hardware and software (Yigitler et al. 2010).

As shown in the Figure 7, the slave module can be maintained in the main module through the connectors. Therefore, this kind of generic platform, designed to support a number of wireless automation applications, can be re-applied in various deployments. These intrinsic factors support the UWASA Node to be a proper sink node of the WSN in this thesis work as well.

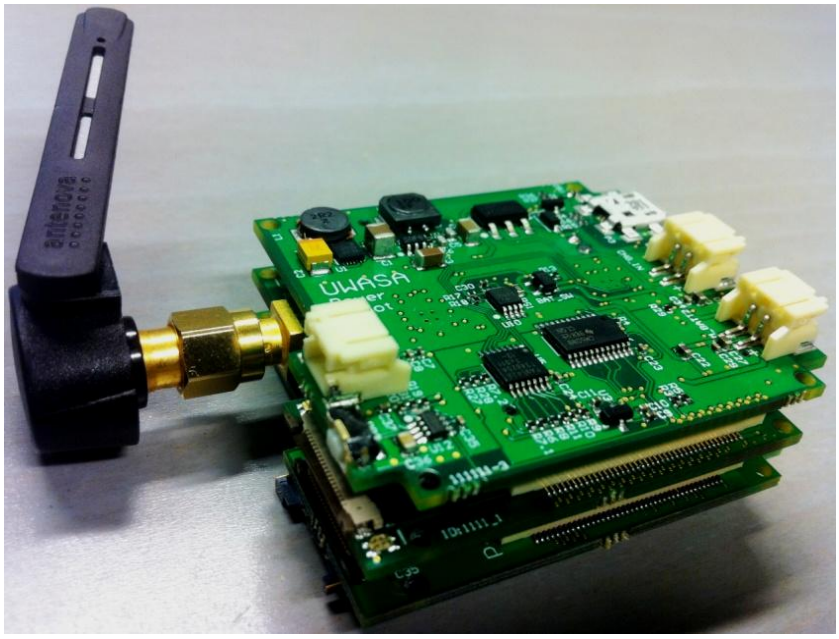


Figure 7. UWASA Node power module, main module and slave module.

3.1.1. Main Controller

Table 1. Features of LPC2378 (NXP Semiconductors 2011: 2–3).

Component	Features
Processor	· ARM7TDMI-S processor; · Running at up to 72 MHz.
Memory	· Up to 512 KB on-chip flash program memory; · 32 KB of static random access memory (SRAM) on the ARM local bus; · 16 KB SRAM for Ethernet interface; · 8 KB SRAM for general-purpose direct memory access (DMA) use.
Interfaces and Peripherals	· Ethernet MAC with associated DMA controller; · Universal serial bus (USB) 2.0 full-speed device; · Four Universal asynchronous receiver/transmitter (UART)s with fractional baud rate generation; · Controller area network (CAN) controller with two channels; · Serial peripheral interface (SPI) controller; · Two synchronous serial port (SSP) controllers; · Three inter-integrated circuit (I2C)-bus interfaces; · Inter-IC Sound (I2S); · SD/MMC memory card interface; · 104 general purpose I/O (GPIO) pins; · 10-bit ADC with input multiplexing among 8 pins; · 10-bit digital-to-analog converter (DAC); · Four general purpose timers/counters; · One pulse-width modulation (PWM)/timer block; · Real-Time Clock (RTC); · 2 KB SRAM powered from the RTC power pin; · Watchdog Timer (WDT).
Power	· Single 3.3 V power supply (3.0 V to 3.6 V); · Modes: idle, sleep, power-down, and deep power-down; · Peripheral Clock Control; · Consumption: 125 mA(max) - 0.4125 W.
Debug Interface	· Embedded ICE: Joint Test Action Group (JTAG); · Embedded Trace.

Based on the ARM7TDMI-S processor with a real-time emulation that combines the microcontroller with 512 KB of high-speed flash memory, LPC2378 from NXP Semiconductors is proper to be chosen as the main controller (NXP Semiconductors 2011: 1). Most main features of the MC can be fully depicted in Table 1.

3.1.2. RF Controller

The CC2431 from Texas Instruments, a system-on-chip (SOC) for ZigBee/IEEE 802.15.4 solutions, is the RF controller of the UWASA Node responsible for wireless communication. The key features of the microcontroller are illustrated in Table 2.

Table 2. Features of CC2431 (Texas Instruments 2009: 1).

Component	Features
Processor	・ 8051 MCU; ・ 32 MHz.
Memory	・ 128 KB in-system programmable flash; ・ 8 KB RAM.
Interfaces and Peripherals	・ 2.4 GHz IEEE 802.15.4 compliant RF transceiver; ZigBee® protocol stack; ・ Location Engine; ・ 21 GPIO pins; ・ Two powerful USARTs; ・ ADC with up to eight inputs and configurable resolution; ・ One IEEE 802.15.4 MAC Timer; ・ One general 16-bit timer and two 8-bit timers.
Power	・ Supply voltage (2.0 V to 3.6 V); ・ Modes: idle, sleep, power-down; ・ Consumption: 40 mA(max) - 0.12 W.
Debug Interface	・ Wire Debug Interface.

3.1.3. MC-RFC Interface

Since the MC and the RFC both have the different power supply level, an interface is required to convert the level smoothly between each controller. As shown in Table 3, the features of the MC-RFC interface can be observed clearly. Because of its small form factor, supporting for low power consumption and high data rate, Texas Instrument TXB0108 is used as the bi-directional voltage level shifter with auto-direction sensing.

Table 3. Features of TXB0108 (Yigitler 2010: 47).

Component	Features
Port 1 Supply Voltage	2.5 V
Port 2 Supply Voltage	3.3 V
Maximal Data Rate	60 Mbps
Power Consumption	4 μA(max)
Form Factor	QFN20 Package: 4.65x3.65x1 mm

3.2. SurfNet

SurfNet, as Figure 8 shows, is the name of the WSN architecture developed by Seinäjoki University of Applied Science (UAS). Additionally, it is improved further in GENSEN-project. It consists of hardware platforms - SurfNet node, network protocol and the corresponding application development environment. The SurfNet node has the size of 23x14x5 mm, and is equipped with a single-chip nRF24LE1D microcontroller by Nordic Semiconductors. Several

sensors such as temperature sensor, humidity sensor and three-dimension (3D) acceleration sensor can be simultaneously mounted to the node. The node is powered by batteries that are usually around 3.0 V in total. SurfNet node is also quite simple and easy to deploy in a wide variety of environments. Therefore, it is selected to be used in the WSN architecture of this thesis work.

Figure 8. SurfNet node (Palomäki 2010a).

As mentioned before, nRF24LE1D is used as the MC of SurfNet node. The nRF24LE1D is a member of the ultra-low power and high-performance family of intelligent 2.4 GHz SOC RF transceivers with embedded microcontrollers (Nordic Semiconductor 2010: 10). Namely, it mainly contains an enhanced 8051 microprocessor control unit (MCU) and an nRF24L01 2.4G RF transceiver. These two parts communicate with each other by SPI bus. Since the SurfNet node is assembled by nRF24LE1D microcontroller, the nRF24LE1D and the nRF24L01 RF transceiver will be discussed in the following part specifically.

3.2.1. NRF24LE1D Microcontroller

Besides processor and memory, the main features of nRF24LE1 are presented in Table 4.

Table 4. Features of nRF24LE1 (Nordic Semiconductor 2010: 11–12).

Component	Features
Processor	· Fast 8-bit; · Intel MCS 51 compliant instruction set; · Reduced instruction cycle time; · 32 bit multiplication–division unit
Memory	· 16 KB of Flash memory with security features; · 1 KB of on-chip RAM memory; · 1 KB Non-volatile data memory
Interfaces and Peripherals	· GPIO; · SPI master; · SPI slave; · 2-Wire master/slave; · Full duplex serial port; · PWM; · External interrupts; · Timer inputs; · 32.768 kHz crystal oscillator; · Debug interface; · High performance 2.4 GHz RF-transceiver; · A/D converter; · Analog comparator; · Encryption/decryption accelerator; · Random number generator;
Power Management	· Single 3.0 V power supply (1.9 V to 3.6 V); · System reset and power supply monitoring; · Low power design supporting fully static stop/ standby; · MCU clock frequency from 125 kHz to 16 MHz; · Voltage regulators supporting low power mode; · Watchdog and wakeup functionality running in low power mode

As it shows, the microcontroller maintains a number of specifications, which are quite suitable to be implemented to be used in the sensor node. For example, the low power design supports sleep mode, standby mode, and deep sleep mode. That is, these properties can be utilized to reduce the power consumption by combining multiple working modes based on the requirements of the application system. According to Palomäki (2010a: 6), the lifetime of sensor node can be further extended by taking advantage of nodes' switching modes and synchronizations (SYNCs).

The nRF24LE1D is an ultra-compact 4×4 mm, 24 pin quad-flat no-leads (QFN) package (7 generic I/O pins). Its top view of pin assignment for the QFN24 4×4 mm package, and the pin number of the SurfNet node can be found in Figure 9.

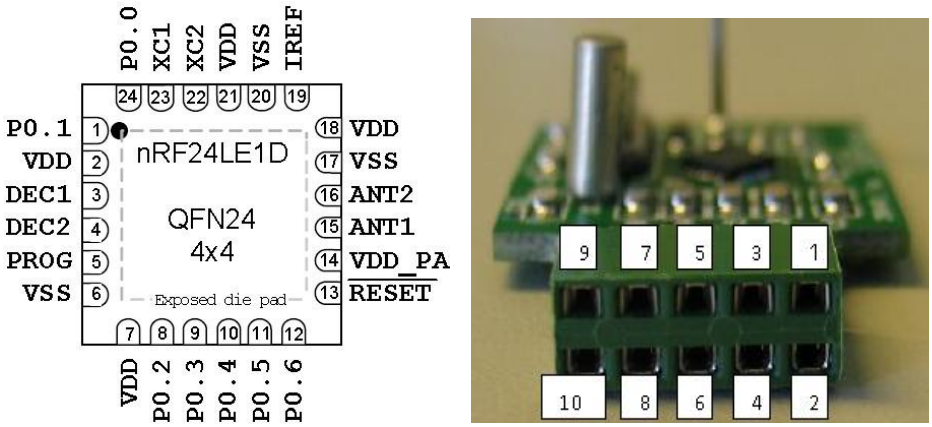


Figure 9. Pin assignment and pin number of SurfNet node (Nordic Semiconductor 2010: 14; Palomäki 2010a).

Moreover, the basic descriptions of the corresponding pin functions can be found in Table 5. It includes the function of power supply, GPIO and so on,

which are corresponding to the pin number of SurfNet node shown in Figure 9.

Table 5. Pin functions of nRF24LE1D.

No.	Pin	Name	Type	Description
1	P0.3	Ain3	Digital/Analog I/O	GPIO pins
2		VDD	Power	Power supply (+1.9V to +3.6V DC)
3	P0.6	Ain6	Digital/Analog I/O	GPIO pins
4		PROG	Digital input	Input to enable flash programming
5		RESET	Digital input	Reset system, low is active
6	P0.5	Ain5	Digital/Analog I/O	GPIO pins
7	P0.2	Ain2	Digital/Analog I/O	GPIO pins
8		VSS	Power	Ground (0V)
9	P0.4	Ain4	Digital/Analog I/O	GPIO pins
10		VSS	Power	Negative supply series to ground (0V)

3.2.2. NRF24L01 RF Transceiver

The 2.4 GHz RF transceiver is an integrated radio frequency unit. The ISM radio band of 2.400–2.4835 GHz is used by the microcontroller. The RF transceiver can be configured by the register, and the register can be accessed by MCU through the SPI bus in every mode of operation. Figure 10 presents the internal structure of the transceiver.

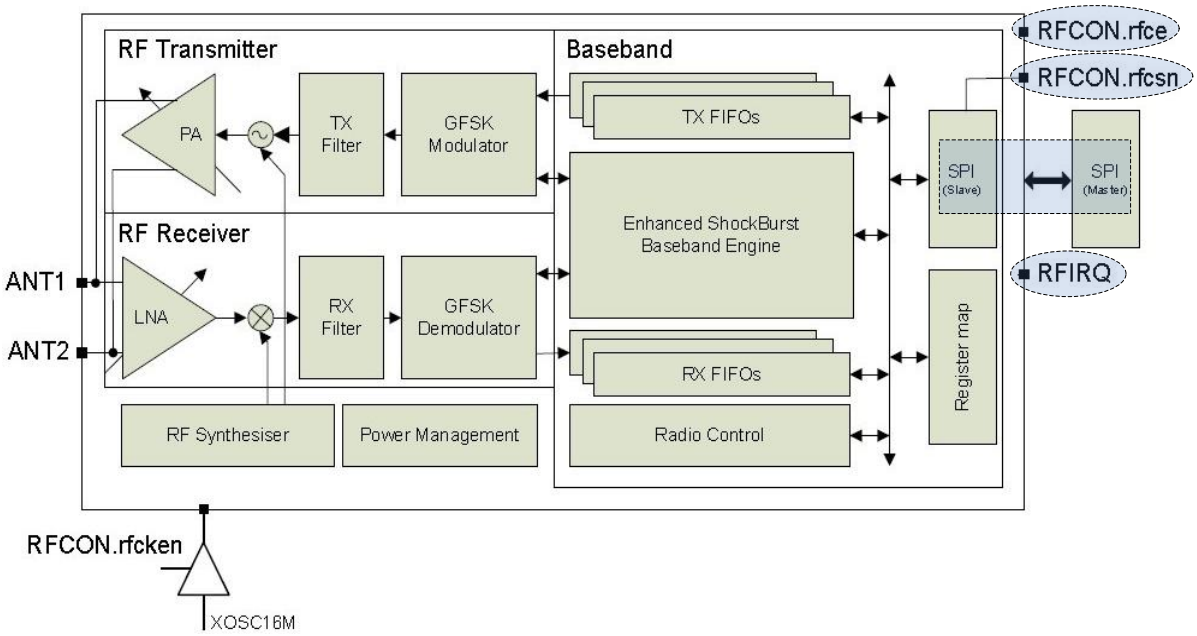


Figure 10. RF transceiver block diagram (Nordic Semiconductor 2010: 17).

As shown in Figure 10, it is by SPI bus that the transceiver communicates with the MCU. Namely, MCU controls the transceiver by three interfaces: RFCON.rfce, RFCON.rfcsn and RFIRQ. The register map backs up for the register. That is, it stores the configurations for the transceiver from MCU. Transmit (TX) first-in-first-out (FIFO) and receive (RX) FIFO are FIFO buffers which are used for the storage of transmitting and receiving data. There are four modes of operation for the RF transceiver: power down mode, standby mode, RX mode, and TX mode. As presented in Table 6, the operating mode can be altered by configuring the bytes of the responding registers.

Table 6. States of RF transceiver and related registers (Nordic Semiconductor 2010: 20).

Mode	PWR_UP register	PRIM_RX register	rfce	FIFO state
RX mode	1	1	1	-
TX mode	1	0	1	Data in TX FIFO. Will empty all levels in TX FIFO
TX mode	1	0	Minimum 10 μs high pulse	Data in TX FIFO. Will empty one level in TX FIFO
Standby-II	1	0	1	TX FIFO empty
Standby-I	1	-	0	No ongoing packet transmission
Power Down	0	-	-	-

The nRF24L01 supports 250 Kbps, 1 Mbps and 2 Mbps data transmission rate. They can be configured by setting up the RF_DR of the RF_SETUP register. Using a higher rate decreases the possibility of collision on air. However, by using a lower rate achieves the higher sensitivity of the data reception. In any case, the RF transmitter and receiver must maintain the same rate to be able to communicate with each other. At the rate of 250 Kbps or 1 Mbps, the nRF24L01 occupies 1 MHz bandwidth in the 2.400–2.4835 GHz ISM band. While in the rate of 2 Mbps, it uses 2 MHz bandwidth in the 2.400–2.4835 ISM band. The radio frequency (F0) is determined by the RF_CH register, and can be calculated as:

$$F_0 = (2400 + RF\_CH)\ MHz \quad (1)$$

To ensure a reliable wireless communication, transmitter and receiver must maintain the same radio frequency channel at the same time, for example, 2440 MHz.

The nRF24L01 received power detector (RPD) is only one bit, which equal to 1 when the received power level is higher than -64 dBm. It means -64 dBm is the minimum received power level receiver can detect in the RF channel. If the received power is less than -64 dBm, RPD equals to 0, which means the receiver has nothing detected. In RX mode, the value of RPD can be read out at any time. Whenever a package is received or RFCON.rfce is set to 0 by MCU, RPD is latched. In this way, the function of carrier sense (CS) can be implemented by checking the value of RPD.

Enhanced ShockBurst™, developed by Nordic Semiconductor, is a packet based data link layer. It includes such features as automatic packet assembly and timing, automatic acknowledgement and package retransmission, if needed. It improves the power efficiency for unidirectional and bi-directional systems, without adding complexity on the controller. (Nordic Semiconductor 2010: 22–23.)

Moreover, the Enhanced ShockBurst™ makes the bi-directional data link communication much easier to achieve. Actually, the packet processing in this mode means the packet exchange between RF transceivers. That is, one transceiver is considered as a primary receiver (PRX) while the other one is acting as a primary transmitter (PTX). The procedure of the automatic packet assembly proceeds as follows:

- 1. PTX transmits a packet to PRX, after which PTX is set to receive mode and

waits for the acknowledgement character (ACK) packet from PRX;

- 2. Once the data packet is received by PRX, the Enhanced ShockBurst™ function automatically assembles and sends an ACK packet to PTX. Then, PRX returns to the receive mode again;
- 3. If PTX does not receive any ACK packet immediately, Enhanced ShockBurst™ will automatically retransmit the packet again after a programmable time interval. Then, the PTX is set to receive mode and waits for an ACK packet.

The parameters of retransmission, for example, retransmission delay time and times of retransmission, can be configured in the Enhanced ShockBurst™ mode. After that, all the operations will be completed automatically without any intervention of the MCU.

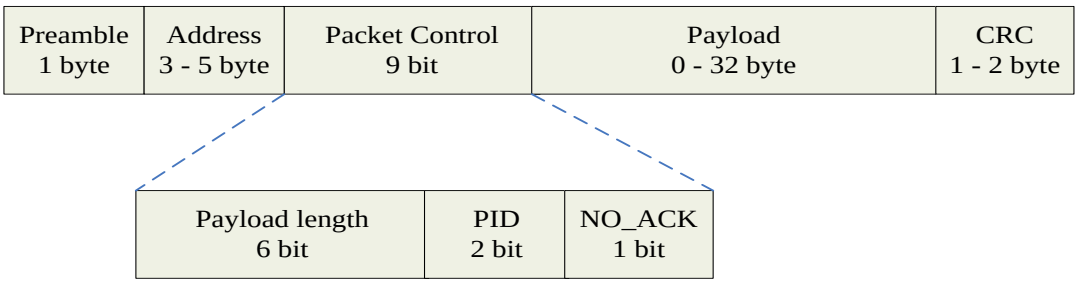


Figure 11. An Enhanced ShockBurst™ packet (Nordic Semiconductor 2010: 23).

The format of the Enhanced ShockBurst™ packet is shown in Figure 11. It contains a preamble field, address field, packet control field, payload field and a cyclic redundancy check (CRC) field. The preamble field is to ensure that the receiver has enough time for the processing. The address field contains the address of the receiver. In addition, the packet control field contains nine bits,

which consist of six bits of the data payload length, two bits of the packet identification (PID), and one bit of no acknowledgment flag. The payload field contains the data defined by the user. CRC field is used for the packet error detection.

MultiCeiver™ by Nordic Semiconductor is a feature used in RX mode. It contains a set of six parallel data pipes with unique addresses, as shown in Figure 12. A data pipe is a logical channel in the physical RF channel. Each one of them has its own physical address that is configured in the RF transceiver. Up to six RF transceivers configured as PTX can communicate with one RF transceiver configured as PRX. In PRX, only one data pipe can receive a packet at a time. Only after one data pipe receives a complete packet, the other data pipes can begin to receive. When multiple PTXs are transmitting to a PRX, the auto retransmission delay function can be used to skew the auto retransmission so that they only block each other once. (Nordic Semiconductor 2010: 33.)

As shown in Figure 12, PRX and PTX 1, for example, assign the same physical address of the data pipe, for example, Pipe 1, so that they can communicate with each other successfully with auto retransmission. The address of PTX can be configured in the TX_ADDR register while the address configuration of PRX is stored from RX_ADDR_P0 up to RX_ADDR_P6, which depends on the data pipe to be used.

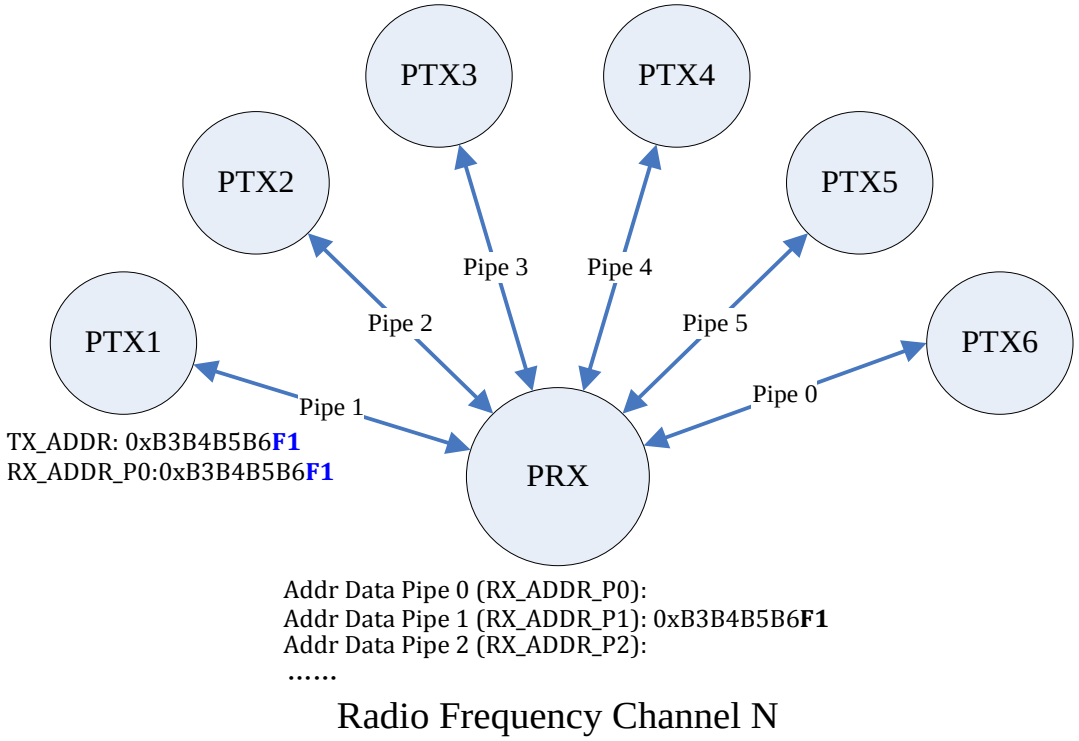


Figure 12. MultiCeiver™ used by PRX (Nordic Semiconductor 2010: 35).

3.2.3. SPI Bus

Generally, there are three types of interfaces provided by nRF24LE1D for data exchange: SPI, 2-Wire and UART. In this research, SPI bus, which is a synchronous serial data connection between master and slave, is mostly used for the inter-microcontroller communication of the SurfNet node. Table 7.a illustrates the pin assignment for the slave SPI bus, which is also applied to the SurfNet bridge node. The SurfNet bridge node can act as a sink and its hardware is compatible with all other SurfNet nodes. In our architecture, the SurfNet bridge connects SurfNet nodes to UWASA Node by operating as a slave SPI device. Table 7.b shows the pin assignment for the Master SPI bus.

Table 7. SPI bus pin assignment.

a. Slave SPI bus			
Pin	Signal	Description	Direction
P0.2	SCK	Serial Clock	Input
P0.3	MOSI	Master Out Slave In	Input
P0.4	MISO	Master In Slave Out	Output
P0.5	CSN	Chip select NOT	Input
P0.6	SYNC	Synchronization	Output
b. Master SPI bus			
Pin	Signal	Description	Direction
P0.2	SCK	Serial Clock	Output
P0.3	MOSI	Master Out Slave In	Output
P0.4	MISO	Master In Slave Out	Input
P0.6	SYNC	Synchronization	Output

The SPI communication is full-duplex and the data are transferred byte by byte of 8 bits. After the synchronization between slave and master, the communication starts. Figure 13 shows the principle of the transmission of one byte by the SPI mode.

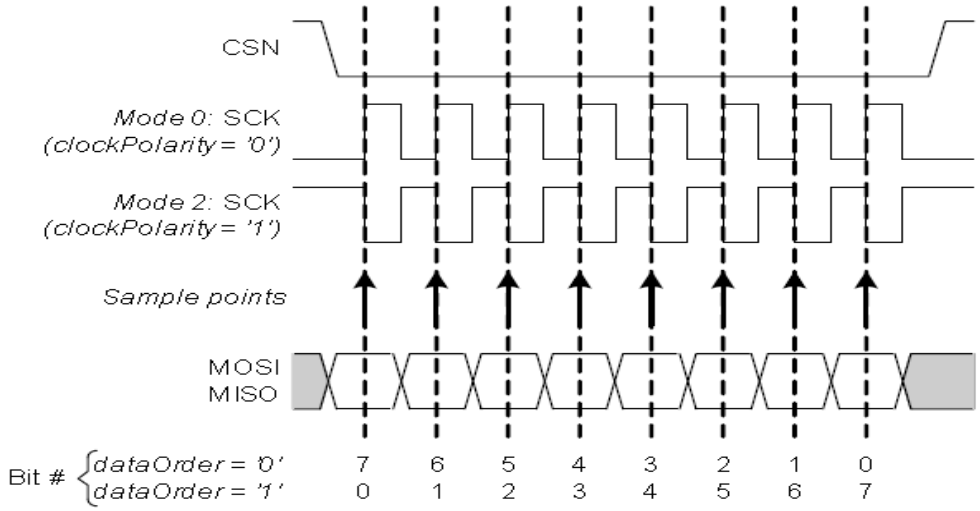


Figure 13. One-byte transmission by SPI (Nordic Semiconductor 2010: 152).

As the figure explains, the message frame starts from the CSN lowing edge. Once the edge rises, the frame ends. There are eight circulations during the exchanging time, correspondingly for eight bits output. Furthermore, there is one bit output to master-out-slave-in (MOSI) in each circulation. Then, the clock signal is set to high state so that the current master-in-slave-out (MISO) bit from the master device could be captured. After that, the clock signal is set to low state again. Thus, the circulation for one bit exchanging is done. Afterwards, the next bit is shifted to be transferred. The timing diagram is given in Figure 14, and its parameters are presented in Table 8. The proper SPI timing must be set according to the table so that the SPI device will achieve enough time to react.

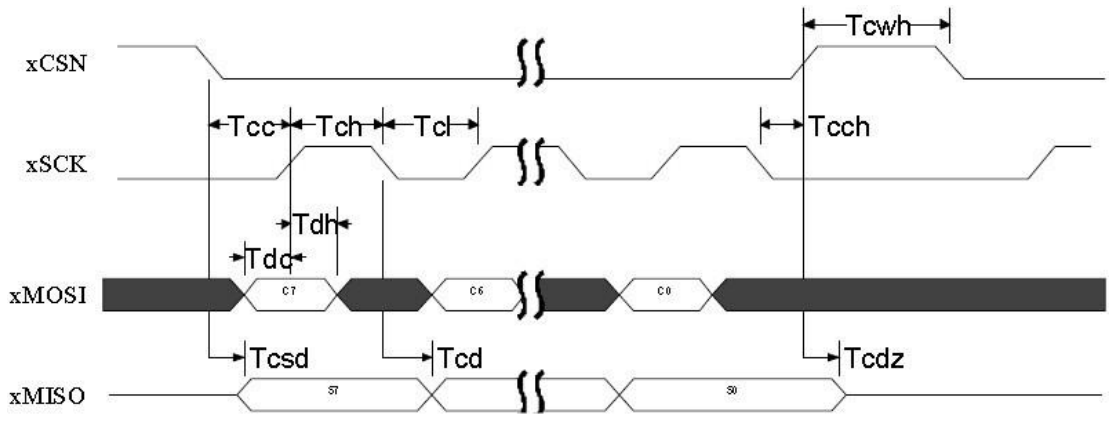


Figure 14. SPI timing diagram: one-byte transmission (Nordic Semiconductor 2010: 152).

Table 8. SPI timing parameters (C_{Load} = 5 pF) (Nordic Semiconductor 2010: 153).

Parameters	Symbol	Min	Max	Units
Data to SCK Setup	Tdc	2		ns
SCK to Data Hold	Tdh	2		ns
CSN to Data Valid	Tcsd		38	ns
SCK to Data Valid	Tcd		55	ns
SCK Low Time	Tcl	40		ns
SCK High Time	Tch	40		ns
SCK Frequency	Fsck	0	8	MHz
SCK Rise and Fall	Tr,Tf		100	ns
CSN to SCK Setup	Tcc	2		ns
SCK to CSN Hold	Tcch	2		ns
CSN Inactive time	Tcwh	50		ns
CSN to Output High Z	Tcdz		38	ns

3.3. UWASA Node Development Board

UWASA Node development board is designed to provide a set of fast development interfaces for the MC. It includes the MC-JTAG connector, power selection block, MC/RFC USB-UART converters, LCD connector, sensor board connector, SurfNet node connector, extension connector and so forth. The main interfaces of UWASA Node development board are shown in Figure 15.

The MC-JTAG connector is working with J-Link Debugger from SEGGER, which is a USB powered JTAG emulator and supported by most Integrated Development Environments (IDEs), for example, IAR EWARM.

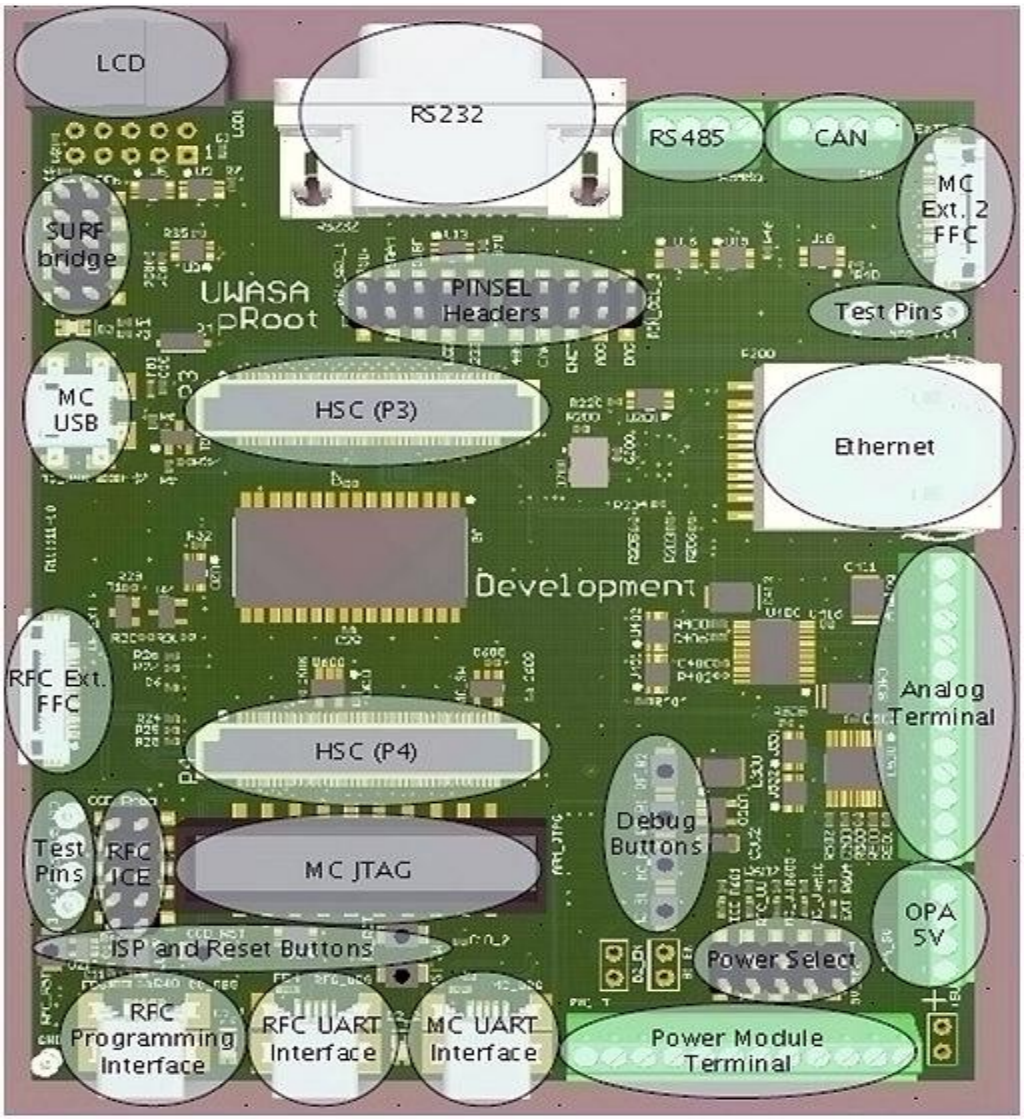


Figure 15. Interfaces of UWASA development board in third revision (Cuhac 2012: 18).

The development board has a SurfNet connector, which is designed to connect the board to SurfNet node. The schematic of the connector is illustrated in Figure 16. Through the connector, the communication between MC and SurfNet node can be done smoothly.

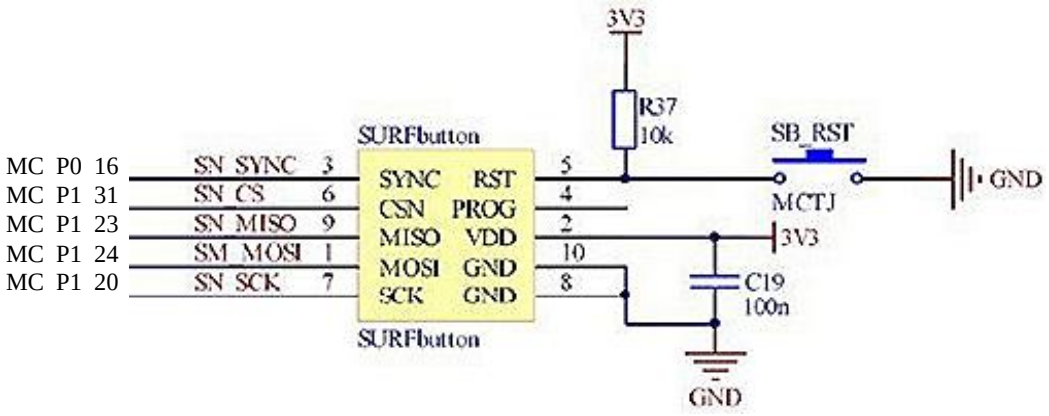


Figure 16. SurfNet connector (Yigitler 2010: 64).

Extension connector is implemented for additional access to peripherals of MC. In this work it is used for UART communication between UWASA Node and Linux FOX Board G20, and also for SPI communication between UWASA Node and SurfNet node. The pin assignment of MC extensions is presented in Figure 17. On the left of the figure, Pin 1 is the power supply of 3.3 V, while Pin 10, 14, 20 are the pins connected to the ground once used. In this work, Pin 18 is assigned to receive data and Pin 19 is used to transmit data, when UWASA Node activates the UART communication with Linux board. In EXT2_F, all the assigned pins in the shown figure are used for connection to SurfNet node. The connector is the same as the one in the development board.

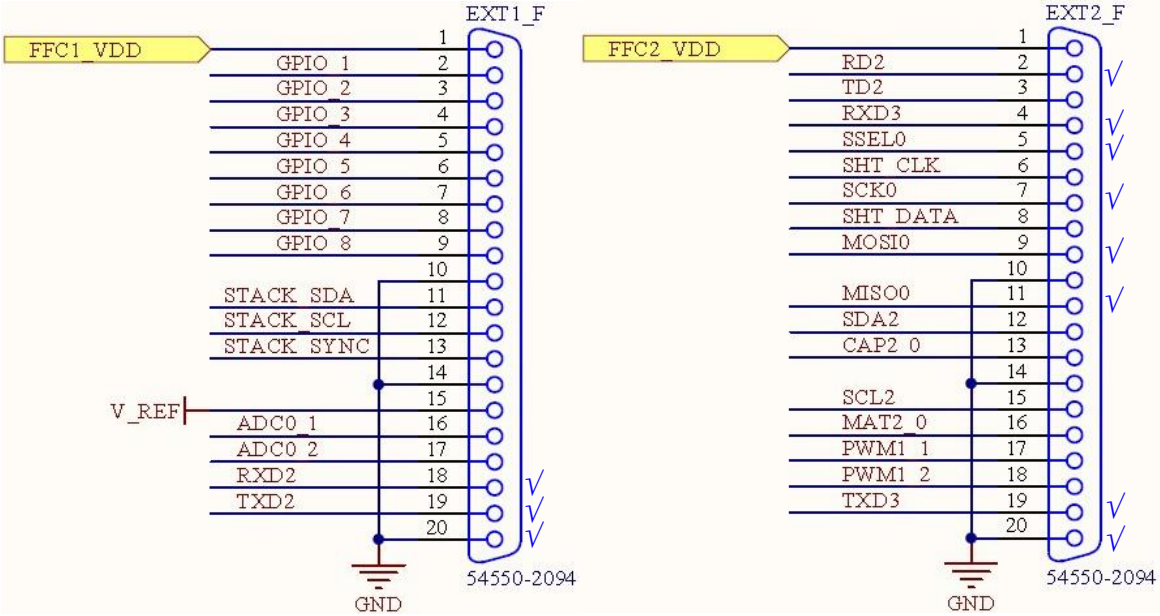


Figure 17. Extension connectors (Yigitler 2011: 4–5) and the used pins.

3.4. Embedded Linux Server with 3G Module

3.4.1. FOX Board G20

The FOX Board G20 is a low-cost embedded Linux board built of ARM9@400Mhz Atmel CPU AT91SAM9G20. The hardware specifications of the FOX Board G20, are presented in Figure 18. Its main features are the following ones (ACME System 2013):

- Built on the Atmel ARM9@400Mhz CPU module Netus G20-L (included);
- 256KB of FLASH memory for the boot loader;
- Two USB 2.0 host ports (12 Mbits);
- One Ethernet 10/100 port;
- One USB device port (12 Mbits);
- One debug serial port (3.3 V);

- Two serial ports (3.3 V);
- One serial port for 4DSystems oLed displays;
- 5VDC power supply input (compatible with PS5V1A);
- RTC with on-board backup battery;
- GPIO lines (3.3 V);
- 4 A/D converter lines, I2C bus, SPI bus;
- Built-in quad power supply Netus PS1 module;
- Average power consumption: 80 mA@5V (0.4 Watt) without micro-SD, Ethernet link, USB devices or other peripherals.

In this work, the embedded Linux server takes care of data processing, third generation (3G) communication, and records the sensors and the camera data. The operations are done in the following sequence. In every cycle, the sensor measurements are first collected from the wireless sensor nodes. Then a picture is taken by the web camera, which is also connected to the server. At the end of each cycle, the embedded server updates the information on the remote server (website) over 3G connection. In this thesis work, the developed system is used for remote monitoring of environmental circumstances inside the building. Since a bidirectional data transmission is enabled, it can be used also for remote control.

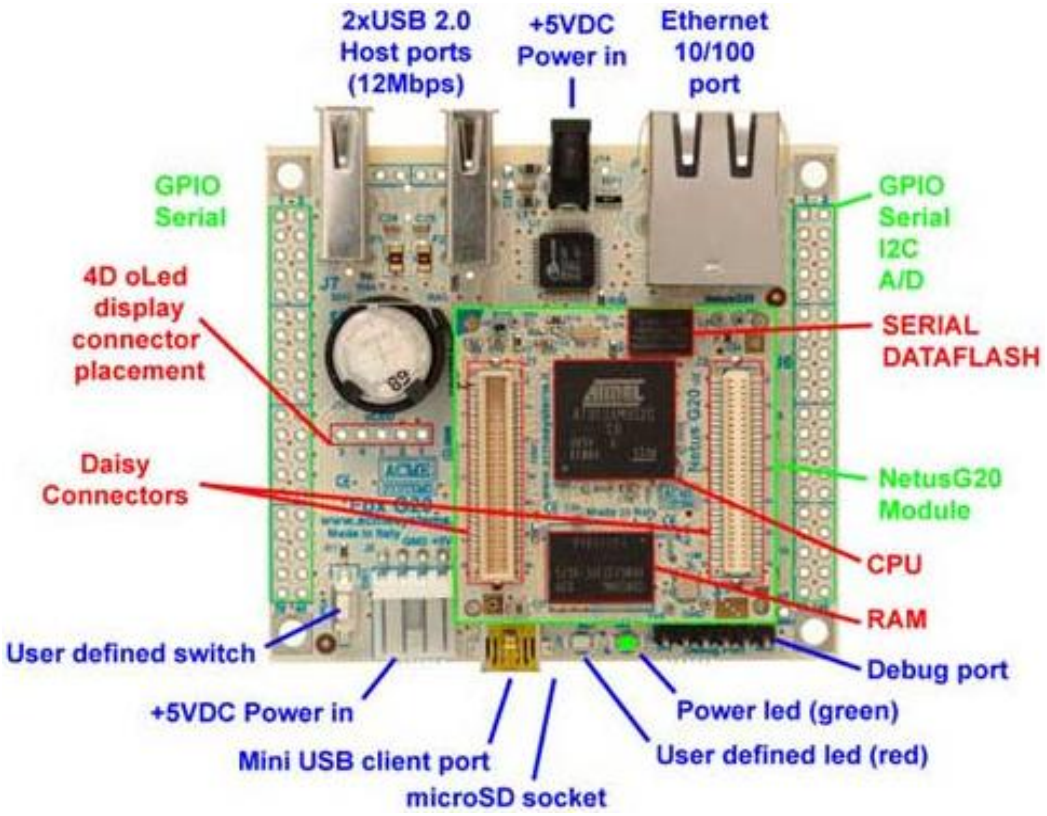


Figure 18. Overview of the embedded Linux FOX Board G20 (ACME System 2013).

3.4.2. 3G Module

The type of the 3G module in use is ZTE MF699. It is a 3G HSPA+USB modem, which is shown in Figure 19. The data transmission speed of ZTE MF669 can theoretically achieve up to 5.76 Mbps upload and 21.6 Mbps download. It works compatibly with most types of operating systems (OS): Windows, Mac, and Linux. Our embedded Linux server connects with cabled Internet server over the 3G connection provided by ZTE MF669.

Figure 19. 3G module: ZTE MF669 HSPA + 3G USB modem.

3.5. Complete Hardware Architecture

The complete hardware architecture and its packaging are presented in Figure 20. In the developed architecture, SurfNet nodes are equipped with temperature and humidity sensors. They transmit their measurements to the SurfNet bridge node connected to UWASA Node that acts as a gateway to the embedded Linux server.

In this system, every SurfNet node is set to sleep for a certain period of time and they are woken up by an internal timer in each cycle. After that, they collect the measured data and transmit the whole data in the TX buffer. Then they wait for the ACK from the receiver, in this case the SurfNet bridge node. Otherwise, the nodes repeat to transmit the previous measurement. When the times of repeating equal to the default value that is set in SurfNet node, the node stops transmitting and goes to sleep. Correspondingly, the SurfNet bridge gives an

ACK once it detects the data from the transmitter. Then it sends the data to UWASA Node by SPI.

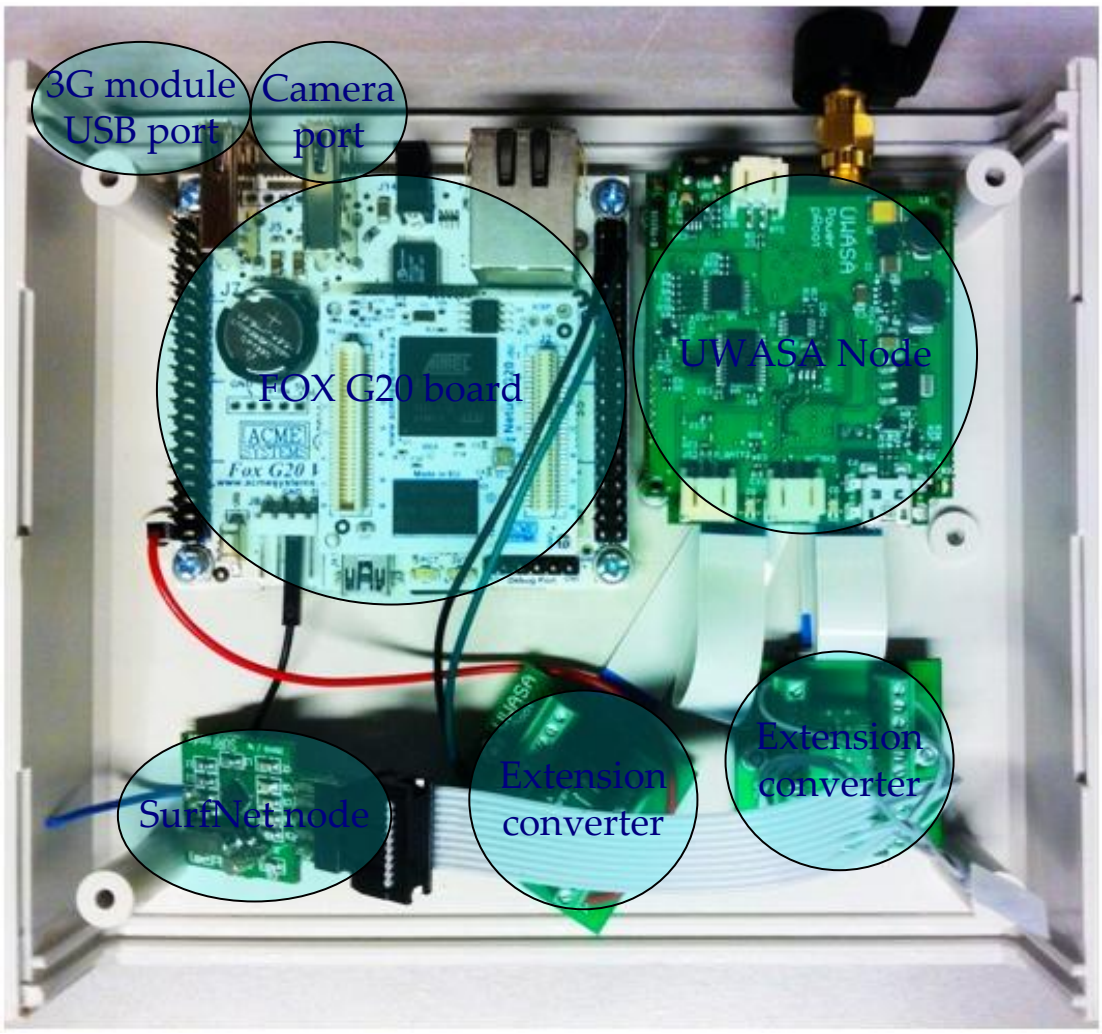


Figure 20. System hardware architecture and packaging.

Subsequently, after receiving data, UWASA Node updates, and stores the data packages in different specified buffers. Once it has a request from the Linux server, UWASA Node forwards the corresponding data to the server through

UART. After the embedded Linux server receives the data from UWASA Node, it logs the time stamp, analyses the package and transfers the data from binary format to the real value. Then the server transmits the data to the web server by 3G link. Finally, the end user can access to the measurements through the server.

4. SOFTWARE DEVELOPMENT AND IMPLEMENTATION

In this chapter, we will describe the system software architecture. At the beginning, the OS is discussed in detail, as well as the protocols and mechanisms, which are implemented in the system. After that, the software developments related to each part of the system are presented.

4.1. Applied Operating System

FreeRTOS is an increasingly popular real-time operating system (RTOS) for embedded devices. Since it is also applied to UWASA Node, FreeRTOS is specifically considered in this thesis. It provides three APIs: task management, queue management and semaphore/mutual exclusion. Only the most important APIs exploited in the development will be introduced briefly, which are task management and semaphore/mutex mechanism.

A task is a set of activities to perform a special function. Each task should have its own memory resources, and the OS needs an appropriate scheduler to control the executing of variant tasks. This scheduler allows the tasks to take or release control of the processor depending on their priorities. (Yigitler 2010: 93–94.)

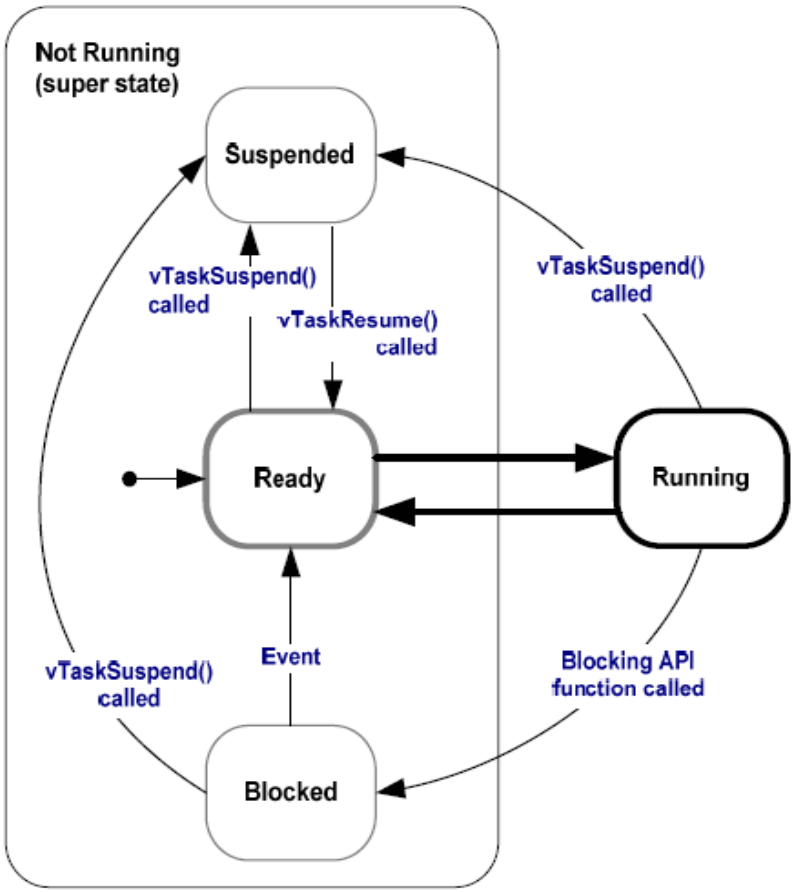


Figure 21. Task state transition (Barry 2009: 20).

As illustrated in Figure 21, there are generally four states of a task: ready, running, suspended and blocked. Each state can be transited to another by the assigned functions, which is pointed out in the figure.

Semaphores and mutual exclusions (Mutexes) are both used for the purposes of resource guarding and inter-task synchronization. Semaphore is a variable that provides a simple but useful abstraction for controlling access to a resource. When a task finishes occupying the resource, the resource is released for other usage. Therefore, a semaphore avoids changing a resource whenever it is taken, no matter what the priority of the current owner task is. (Yigitler 2010: 99.)

Mutexes are binary semaphores that employ a priority inheritance mechanism. In this mechanism, if a high priority task blocks while attempting to obtain a mutex, which is currently held by a lower priority task, then the priority of the task with the mutex is temporarily raised to the same of the blocking task. Therefore, higher priority task is kept in the blocked state for as short time as possible, thus minimizing the problem of priority inversion. (Barry 2009: 105.)

4.2. Applied Protocol Software

As for the aspect of software, there are two different development kits regarding to the different hardware of subsystems, which are used in the system. IAR Embedded Workbench IDE is used for developing the software for UWASA Node, while SURFprogrammer, a programming environment for nRF24LE1 radio controller platform from Seinäjoki UAS, is used for developing the programs of SurfNet. The details of the mentioned IDEs can be seen in the Appendix 9 and Appendix 10. We apply two types of sensor nodes in our WSN: the UWASA Node and SurfNet. The basic operations in the nodes are executed as presented in Figure 22.

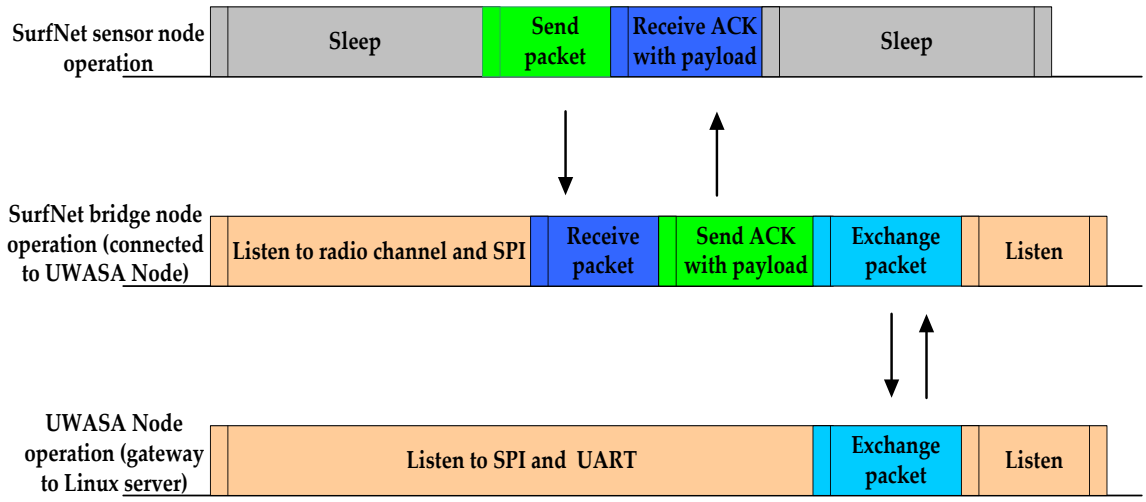


Figure 22. Data packet transaction.

Every time when UWASA Node is powered on, it firstly initializes the processors, including main processor and RF processor. Then, it handles the tasks in which it is continuously in listening mode for the requests sent by the embedded Linux server via UART, or for the data packets transmitted through SPI by the SurfNet receiver. Otherwise, if there is no transmission request detected, a timeout occurs and it breaks out of the listening mode. Once the node gets a request from the Linux server, it checks the data packet, and then it copies and passes the related buffer to the Linux server, that contains the humidity and temperature measurements at a certain time collected from one single sensor node. In addition to acting as a gateway between the WSN and Linux server, the UWASA Node also powers up the SurfNet node that is attached into it.

Particularly, there are two operating modes for the receiver: low-latency mode and low-power mode. In low-latency mode, the receiver keeps listening to the radio channel continuously. In that case, the receiver obtains a lower time delay,

but consumes more power. In low-power mode, the receiver alternates between sleep mode and listening mode according to pre-set timer. The receiver only stays in listening mode for a limited time. In this mode, it consumes lower power. However, it might miss the incoming packets while sleeping. In the system developed in this work, the low-latency mode is applied in the SurfNet receiver. The power consumption of the gateway node is not a quite critical issue, because the SurfNet node, which is connected to UWASA Node in the gateway, is also powered up by the UWASA Node. If it has data detected in the RX buffer, the node copies the buffers, transmits them through SPI to UWASA Node immediately, and then enters back to the listening mode.

The SurfNet sensor node is developed to be used as a low-power sensor node. It mainly runs in three different modes: sleeping, transmitting and listening. In every period of time preset by a timer, the node wakes up, gets sensors powered on. Then the node goes to low-power mode again for several milliseconds to settle the humidity and temperature sensors. Then it wakes up, makes the measurements and stores the data in TX buffer, and transmits them. After that, the node listens to the channel for a small period, for example, 100 μs. If the node gets an ACK from the receiver, it ends the listening mode and enters to sleep mode. If there is an error or collision happening during the wireless transmission, so that the transmitter cannot get the ACK from the receiver. In this case, the node retransmits the packet after an assigned delay time, for example, 250 μs. This mechanism is designed to reduce the power consumption of the sensor nodes. By doing so, it also extends the operating time (lifetime) of the system.

To prevent the transmission collisions, the MultiCeiver™ technique by Nordic

Semiconductor is applied in SurfNet nodes. To improve the reliability of the wireless communication, there is a retransmission function working in the transmitter if there is any packet collision in the receiver.

Furthermore, while the sensor node is in listening mode for the ACK after one transmission, CSMA is applied to check whether the radio channel is occupied by another node or not. That is, the node to listen to the channel by using a received power detector. If the received power is higher than -64 dBm (3.98×10^{-7} mW), the channel is detected to be occupied, and the transmission from each sensor node to receiver is delayed based on the ID number of the sensor node. CSMA is quite simple and easy to implement. It can enhance the development efficiency because it does not need centralized control or pre-defined priorities.

According to the MultiCeiver™ technique, there are up to six communication data pipes from PTXs to PRX. It only allows one data pipe functioning in one time. In other words, only after the complete transmission is done in one data pipe, can other data pipes be enabled. When multiple sensor nodes are transmitting to the receiver through different data pipes, the data pipe, in which the first packet reaches to the receiver, is enabled prior to the others. The other sensor nodes listen to the channel for a while, but fail to get an ACK. In this case, they have to make an auto retransmission delay, the value of which is set based on the ID number of each sensor node.

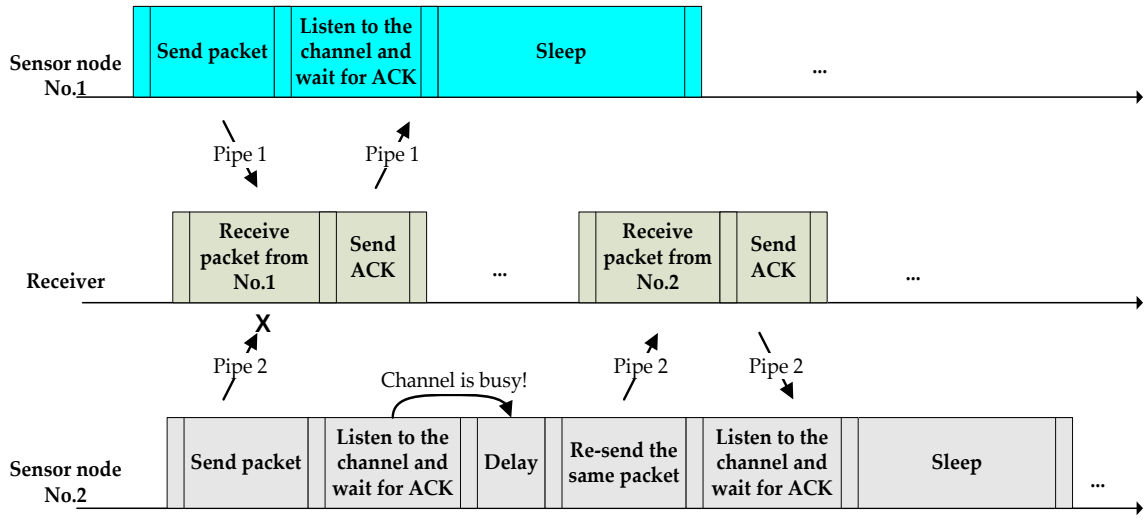


Figure 23. Packet collision avoidance by MultiCeiver™ technique.

For example, there are two nodes respectively with ID 1 and ID 2. In addition, there is a preset time delay of one and two milliseconds respectively in sensor Node 1 and Node 2. As shown in Figure 23, assuming they are not transmitting at the same time (which is the worst case explained in next paragraph), the nodes are trying to transmit one data packet to the receiver. Node 1 first manages the transmission with the receiver in pipe 1. Before the complete packet is received, other data pipes are not enabled. In this way, even though Node 2 transmits the data packet, but it is not able to receive the ACK from the receiver, because the receiver does not send any at all. It keeps waiting for two milliseconds. After that, it tries to transmit the data again until the limited times of retransmission, for example, four times of retransmission.

However, in a worst case, multiple sensor nodes can transmit their data packets at the same time. The receiver cannot make a judgment that which packet arrives firstly, and it does not send an ACK back to the sensor nodes. Sensor nodes cannot detect any ACK and they delay a period of time based on their

node ID number, for example, one millisecond for Node 1 and two milliseconds for Node 2. Then sensor nodes retransmit the same previous data packet when the delay time is due. In this worst case, there is only one packet collision in the receiver at the beginning, and the packets in different data pipes will not block each other again.

By using these techniques, packet collisions from the transmitters can be effectively prevented. At least, even in the worst case, which means the nodes transmit the packet absolutely at the same time, the collision of the packet will not happen more than once.

4.3. Message Structure

According to the architecture of the system, there are three different message structures that are necessary to be specified. In the WSN, the message between the SurfNet transmitter and receiver contains eleven bytes, as shown in Figure 24. The message frame starts with hexadecimal value '0x2A' (42 in decimal values) and ends with '0xAB' (171 in decimal) and '0x55' (85 in decimal). Because of the SPI communication protocol between UWASA Node and the SurfNet node attached to it, these three bytes are formatted in the sensor node from the beginning of the data flow and then they remain the same all the time to the Linux server.

Message frame	0x2A	Length	ID of Sensor Node	Data	Counter	Checksum	0xAB	0x55
Number of byte	1	2	3	4-7	8	9	10	11

Figure 24. Message structure between SurfNet nodes.

In the frame, the second byte represents the number of bytes, in other words, the length of the message. The third byte is the ID of the SurfNet sensor node. Then, the following four bytes are the measured data from the sensor: first two bytes together means the temperature value in hexadecimal and the next two bytes means the value of humidity. These hexadecimal values are directly obtained from sensors and converted by the ADC. The next byte is a counter of the data packet in one sensor node, from where the number of packets has been sent by the sensor node can be known. The 9th byte is a checksum byte, which means the CRC is applied to packet error detection. This message frame is structured in the SurfNet sensor node.

After the message is received by the SurNet receiver, the node adds its ID number into the message structure. In addition, it removes the checksum byte because the byte has been already used for CRC function. Therefore, the new message still contains eleven bytes, including the receiver's ID. The receiver node passes the packet to UWASA Node though SPI. The restructured message can be seen in Figure 25.

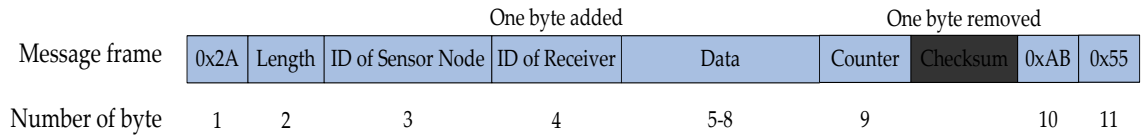


Figure 25. Message structure between SurfNet receiver and UWASA Node.

When UWASA Node receives the packet, it firstly checks if the start and end bytes in the message frame are all correct, to ensure there is no error occurring

during the SPI transmission. Then UWASA Node saves the whole packet in an assigned buffer according to the ID number of the sensor node in the message, for example, packet from SurfNet Node 1 is saved in Buffer 1. In other words, the format of the packet saved in UWASA Node is not restructured or changed. Once the Linux servers requests the data, the UWASA Node passes the corresponding buffer to the server. In this way, the server can obtain the information from each sensor node in the WSN.

Furthermore, the request packet from Linux server to UWASA Node is also a message structure need to be specified, which contains only two bytes: ID number and '0x55' in hexadecimal value. The ID number means that from which sensor node the server requests the information. More specifically, according to this byte, UWASA Node is able to find the corresponding buffer that the server needs. And '0x55' is only an end byte of the message. After UWASA Node gets the request packet from the Linux server, it also needs to check whether the end byte of the packet is correct or not. The message structure is shown in Figure 26.

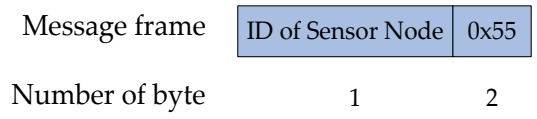


Figure 26. Message structure from Linux server to UWASA Node.

4.4. Sink

A sink consists of UWASA Node and a SurfNet node attached into it by SPI. It acts as a gateway between the wireless network and the Linux server.

4.4.1. Role of the UWASA Node

In the software development and implementation of UWASA Node, IAR Embedded Workbench® IDE for ARM is mainly used. It is a C/C++ compiler and a debugger tool suite supporting a large number of 8-bit, 16-bit and 32-bit microcontrollers. Besides, J-link is working with IAR IDE compatibly, which is a JTAG emulator supporting ARM cores.

When working as a part of the sink, the UWASA Node must guarantee the reliability of the communication from both SPI and UART. Consequently, its main responsibility contains listening for queries from the Linux server, transmitting packets through the UART, receiving and handling data packets from SurfNet node through SPI. Namely, UWASA Node handles the data packets after it receives them from the SurfNet node. According to the ID number in the packet, it stores the packet in the corresponding buffer, for example, packet from SurfNet Node 1 is saved in Buffer 1, which is explained in the previous section. If one new packet is coming from the same sensor node, the related buffer is updated with the latest packet while the previous one is erased. In this way, it ensures that when the Linux server requests data packet, the buffer is always filled with the latest one before transmission. The general flow chart of UWASA Node operation can be seen in Figure 27.

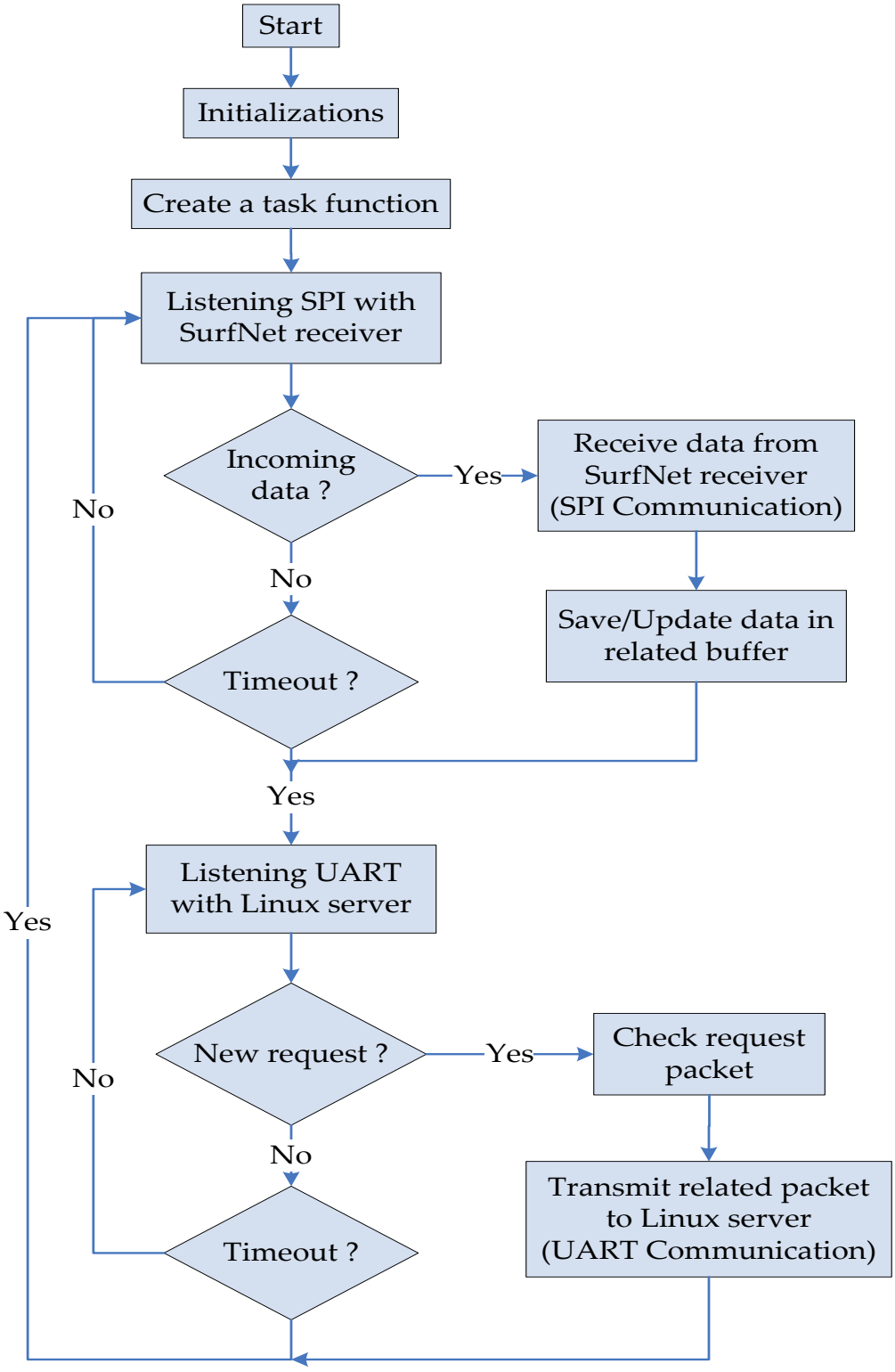


Figure 27. Flow chart of UWASA Node operations in the sink.

As the flow chart presents, UWASA Node periodically listens to the buffers for UART and SPI communications. As soon as UWASA Node obtains data from SurfNet node, it copies and saves the data to the different buffers related to the ID number of the sensor node in the packet. If the packet from the same sensor node is received again, then the related buffer is updated. Once UWASA Node gets a request from the Linux server (FOX G20) through the UART, it firstly checks the packet, and transmit the buffer that the server needs according to the ID number in the packet. After that, UWASA Node goes to listen to SPI with SurfNet nodes again. To ensure the reliable UART transmissions, the same baud rate is initialized and configured to both UWASA Node and the Linux server. When the data packet arrives in the Linux server, the server records a time stamp in the system immediately, which shows the real time to the users.

4.4.2. SurfNet Receiver

SURFprogrammer programming environment, developed by Seinäjoki UAS, is used to program the SurfNet nodes. The suite has the related software and USB-programming stick based on an AT90USB162 controller (see Figure 28). SURFprogrammer is integrated with a small device C compiler (SDCC), which compiles C language program into Intel-hex format and the USB-programming stick transfers the code into the flash memory of the nRF24L1E radio processor, through an SPI channel (Palomäki 2010b: 3).

Figure 28. SURFprogrammer.

In the developed system, some SurfNet nodes are equipped with temperature and humidity sensors. One of them is connected with UWASA Node and it acts as a gateway to the Linux server. The details of the SurfNet node operation in a gateway are illustrated in the general flow chart in Figure 29.

As the figure shows, the SurfNet receiver continuously listens to the radio channel. Once a packet is captured, the node sends an ACK packet back to the transmitter and then adds one byte that is the ID number of the receiver to the packet, as mentioned in the previous section. Then the node passes the restructured packet to the UWASA Node.

In order to guarantee the quality of radio communication with SurfNet nodes, the Enhanced ShockBurst™ technique by Nordic Semiconductor is functioning in the receiver. For example, if the receiver detects a packet for one certain sensor node, it automatically gives an ACK back to the sensor node. Thus, the sensor node is aware of the successful transmission. Additionally, the MultiCeiver™ technique developed by Nordic Semiconductor is also applied at the receiver. It enables the receiver to receive data packets from multiple paths (up to six) based on the different communication pipes of the transmitters.

Besides, the receiver checks the correctness of the received data. Every time the receiver obtains a packet, it determines whether there is an error or not by checking the checksum byte in the data packet. If an error occurs in the packet, the receiver discards the erroneous packet, and receives another one instead.

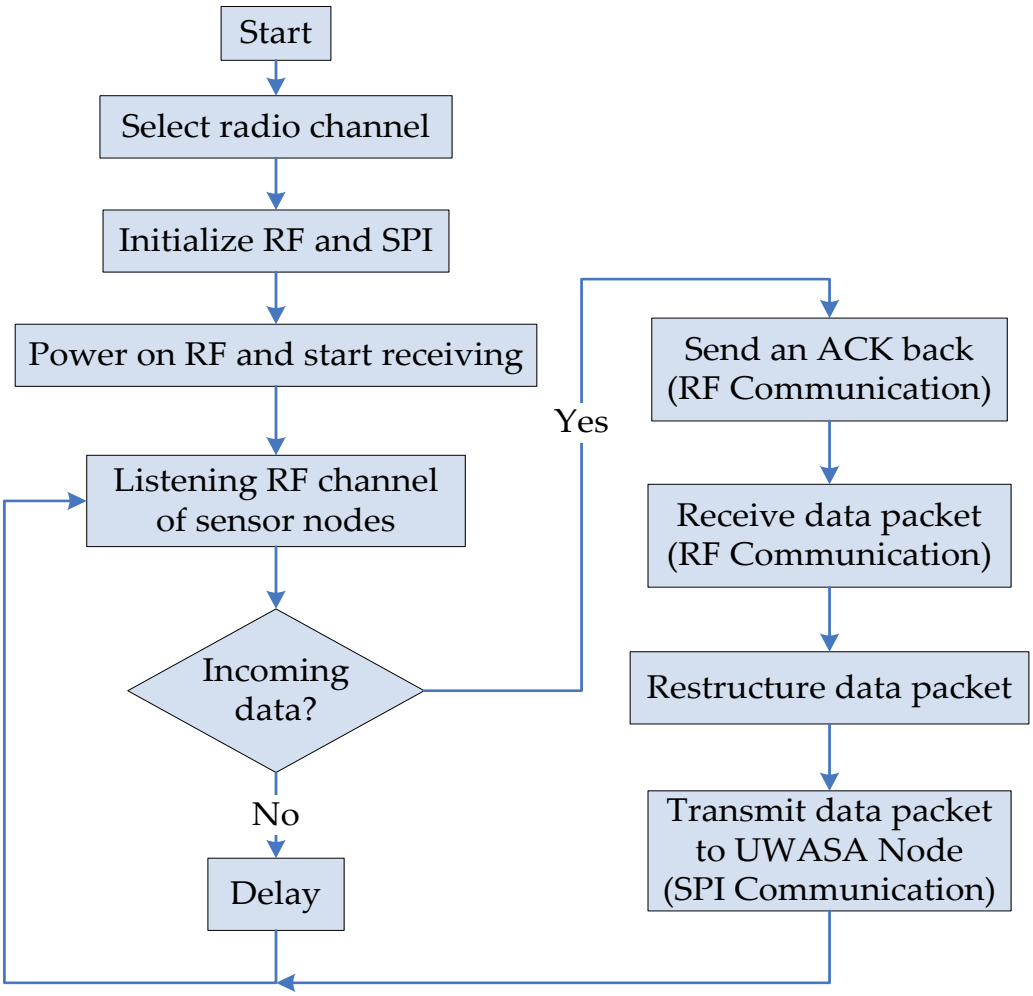


Figure 29. Flow chart of SurfNet receiver's operations in the sink.

4.5. SurfNet Sensor Node

In the system developed in this thesis work, we use SurfNet nodes to collect the temperature and humidity measurements. Their small size and low energy consumption provide the flexibility, which is needed in this sort of monitoring. The SURFprogrammer programming environment is also used for the nodes, as explained in the previous section. Figure 30 shows the SurfNet node equipped with temperature and humidity sensors and one 210 mAh button battery. Since there are many similar sensor nodes in the network, each node must have a unique node ID, for example, ID 0 is assigned to be the receiver's ID.

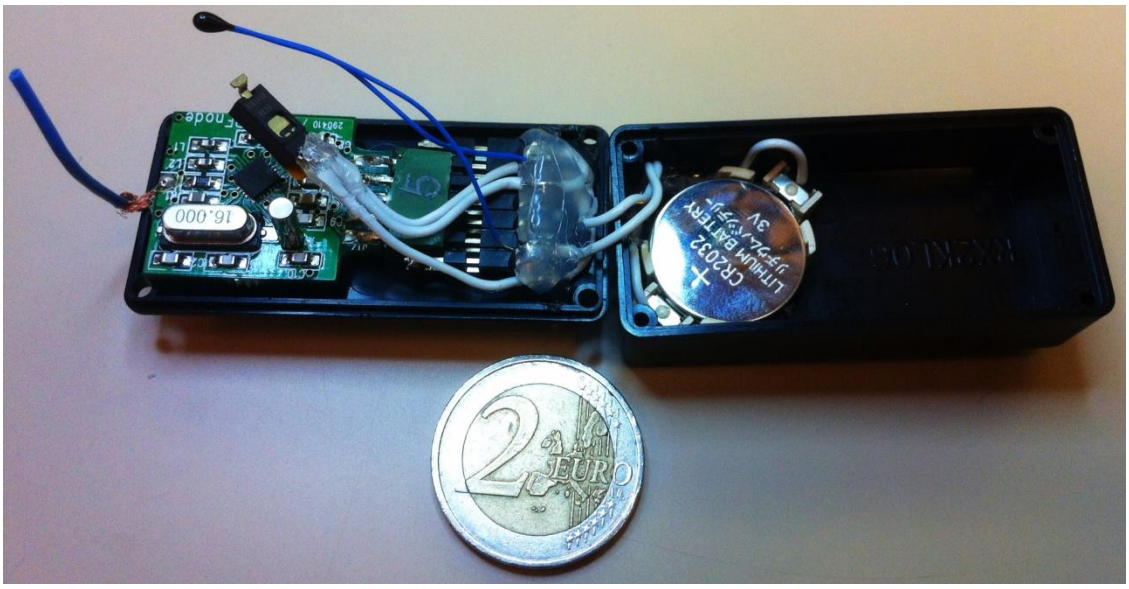


Figure 30. An encasing of the SurfNet node equipped with temperature and humidity sensors and a battery.

Enhanced ShockBurst™, MultiCeiver™ and CSMA techniques are implemented in this system to ensure the reliability of the transmission, which includes the

functions such as auto ACK, auto retransmission and some other communication parameters such as retransmission times, delay time for receiving ACK, RF power and data rate of transmission, and so forth.

Periodic sleeping and waking up modes are applied in the WSN consisting of SurfNet nodes to achieve energy savings. Generally, every SurfNet sensor node has an internal timer inside the node. By using the preset timer, the node wakes up and sleeps periodically. The sleeping time of the SurfNet node in this system is set to eight seconds, which can be changed based on different purposes.

After the node wakes up from the sleep mode, firstly it power up humidity and temperature sensors. There must be at least 2.5 milliseconds for settling down the humidity sensor, otherwise the measurement can be incorrect (see APPENDIX 8 in detail). During this settling time, SurfNet also stays in sleep mode to decrease the power consumption of the node. After settling the sensors, the sensor node powers on ADC to obtain the measurements of the current environment. Afterwards, sensor node formats data packet in TX buffer and prepare to transmit. Then the node powers on the radio and begins to transmit.

After transmission to the receiver, the sensor node stays in listening mode and waits for an ACK packet. If it receives the ACK from the receiver, the communication between both nodes is completed. Otherwise, after a preset time delay, the sensor node tries to retransmit the previous packet. Because the transmission from each node is asynchronous, each node sets the delay time for its retransmission based on its node ID to avoid the collision of the data packets from different nodes to the receiver. Besides, the retransmission times is also preset in the SurfNet node, for example, in the SurfNet node of the developed

system, the retransmission times is set to four. The flow chart presenting the operation and the main functions of the sensor node can be seen in Figure 31.

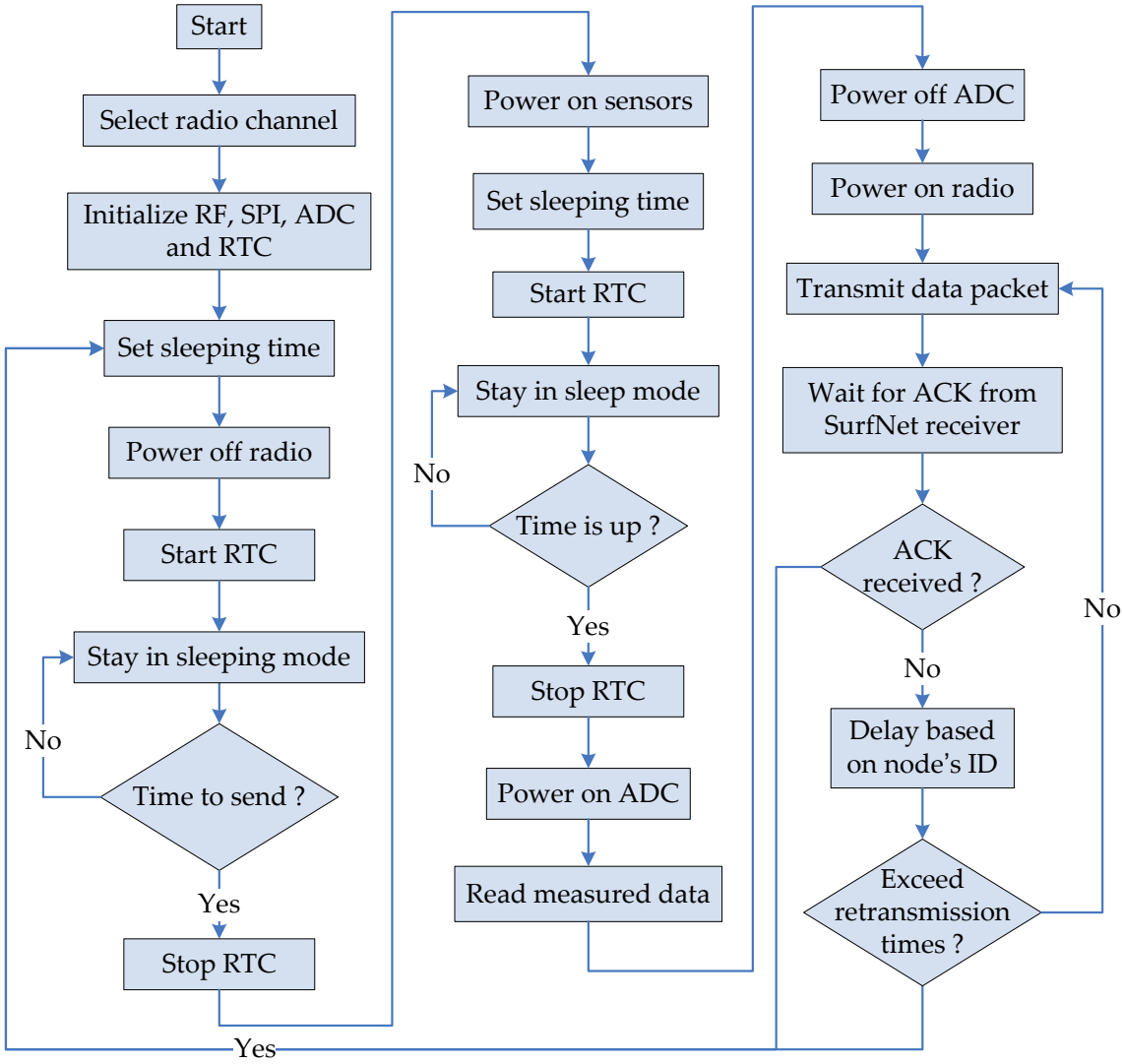


Figure 31. Flow chart of SurfNet sensor node's operation.

5. EXPERIMENTS AND RESULTS

5.1. Experimental Setups

The developed system is illustrated in Figure 32. It consists of SurfNet Nodes, UWASA Node, a Linux server Fox G20 and a 3G module. Its performance is evaluated through following experiments.

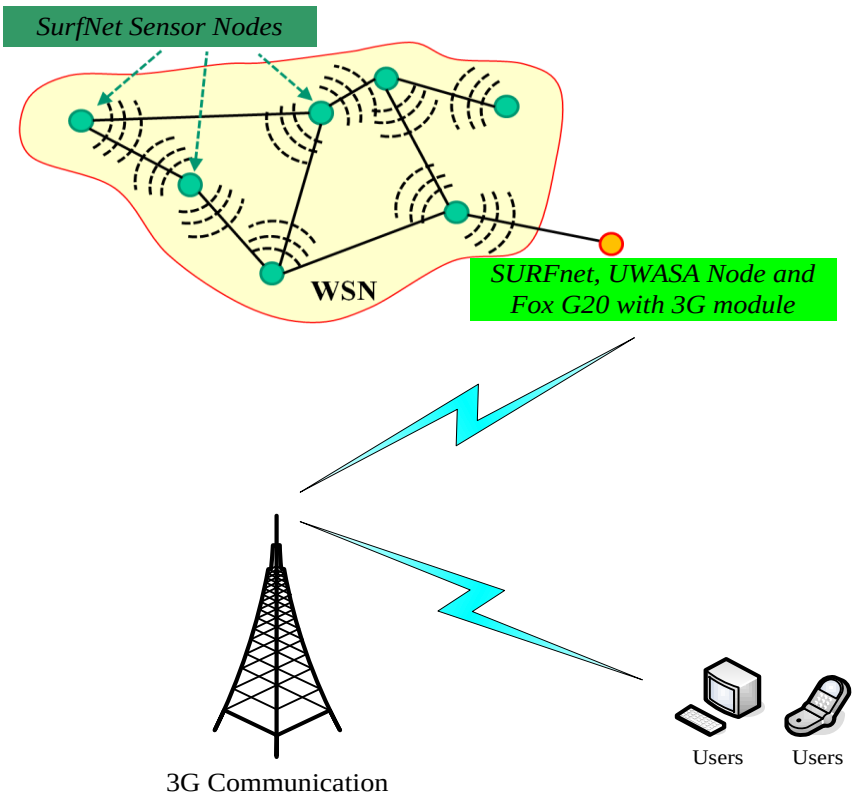


Figure 32. Overview of the deployed system.

The first experiment is to measure the packet loss as a function of distance

between the SurfNet nodes. In the experiment, we vary the distance between one SurfNet node and the sink in indoor and outdoor scenarios. In both scenarios, the distance between the node and the sink was respectively set to 5, 10, 15, 20, 25, 30, 35, 40, 45, and 50 meters. They were placed one meter above the ground level. Five sets of measurements were performed at each location.

In each set of measurements, the transmitter sent 255 packets to the receiver in total and one packet once in every second. That means, the sleeping time of the transmitter is set to one second. And the receiver keeps on listening all the time. Once it detects any packet in its RX buffer, it transmits them to UWASA Node through SPI. In each packet, there is one counter byte that counts the number of the packet sent by the SurfNet node. The counter in the packet starts from 0x00 to 0xFF except 0x55, because 0x55 is already assigned to distinguish the end of each packet. Once the UWASA Node receives the packet, it passes every received packet to PC via USB. Then we can capture the data packets from the port of PC by using a serial terminal. By recording all the received packets in the terminal, the packet loss can be calculated by Formula 2. In this experiment, the number of total packets equals to 255.

$$Packet\ Loss = \frac{Number\ of\ Lost\ Packets}{Number\ of\ Total\ Packets} \cdot 100\% \qquad (2)$$

RealTerm is a terminal software for capturing and controlling data streams. Here, it is used to capture and analyze the packet. One example about using the RealTerm is shown in Figure 33. The message structure contains eleven bytes, which is explained in Chapter 4.3. These bytes are start byte (0x2A), packet length (0x0B), receiver's ID (0x01), transmitter's ID (0x02), humidity (high byte), humidity (low byte), temperature (high byte), temperature (low byte), counter

(from 0x00 to 0xFF, except 0x55), end byte (0xAB), and end byte (0x55).

By calculating the counter byte, the lost packets can be realized. Here is an example shown in Figure 33 to demonstrate the way to obtain the packet loss of the communication. In this example, we sent 23 packets with the counter from 0x07 to 0x1D. As shown in the figure, there are 8 lost packets with the counter from 0x0E to 0x15. In this special case, The observed packet loss as a percentage of transmitted packages can be calculated as 8/23, which is approximately 34.783%.

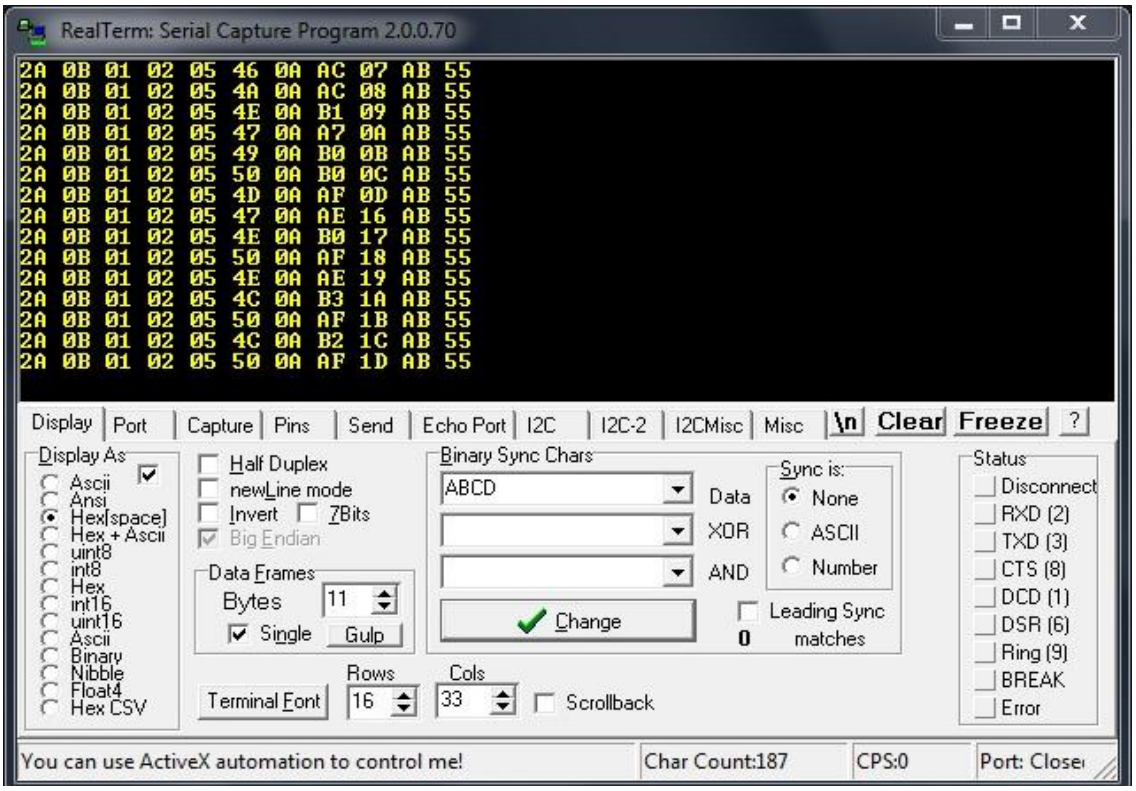


Figure 33. One test example in RealTerm, which is a serial capture program.

In these tests of the first experiment, the SurfNet transmitter is configured with

the following settings:

- RF-channel is set as (2400 + 40) MHz;
- 16-bits CRC is enabled;
- RF data rate: 2 Mbps;
- RF output power in TX mode: 0 dBm (1 mW);
- Auto Acknowledgement function on data pipe is enabled;
- Automatic retransmission is set as:
  - Auto retransmit delay: wait 4000 μs;
  - Auto retransmit count: up to 15 retransmits if fails;
- Wake up from sleep mode in every one second.

Similarly, the SurfNet receiver is configured with the following settings:

- The RF channel is set as (2400 + 40) MHz;
- 16-bits CRC is enabled;
- RF data rate: 2 Mbps;
- Auto Acknowledgement function on data pipe is enabled;
- Automatic retransmission is set to cooperate with the transmitter.

In the second experiment, another significant issue, power consumption of our system is considered. When measuring the power consumption, we observe two entities: the SurfNet nodes equipped with sensors and the sink (UWASA Node and a SurfNet node connected to it by SPI). Since the Linux server is using an external power supply, the power consumption of the server is not discussed in this thesis.

In the sink, the UWASA Node is powered by the Linux server and the SurfNet node attached into it is powered by the UWASA Node. We connected one serial ampere meter in series between the UWASA Node and the power source. In

this way, when the system is operating, the current value of the sink, which contains the UWASA Node and the SurfNet node connected with it, can be read from the ampere meter.

According to Palomäki & Huhta (2010), the node protocol software has a remarkable effect on the node power consumption. The protocol software used in the SurfNet node is explained in more detail in Chapter 4. For example, different operating modes, including sleeping mode and short-term listening mode, are applied for saving power.

The average power consumption of the node is computed by measuring the current in different operating modes of the SurfNet node, which is equipped with sensors. In this test, one digital oscilloscope and one test resistor of 1 ohm are used. The average current taken by SurfNet node is computed by using the measured voltages over the resistor in different operating modes. The applied test hardware setup is presented in Figure 34.

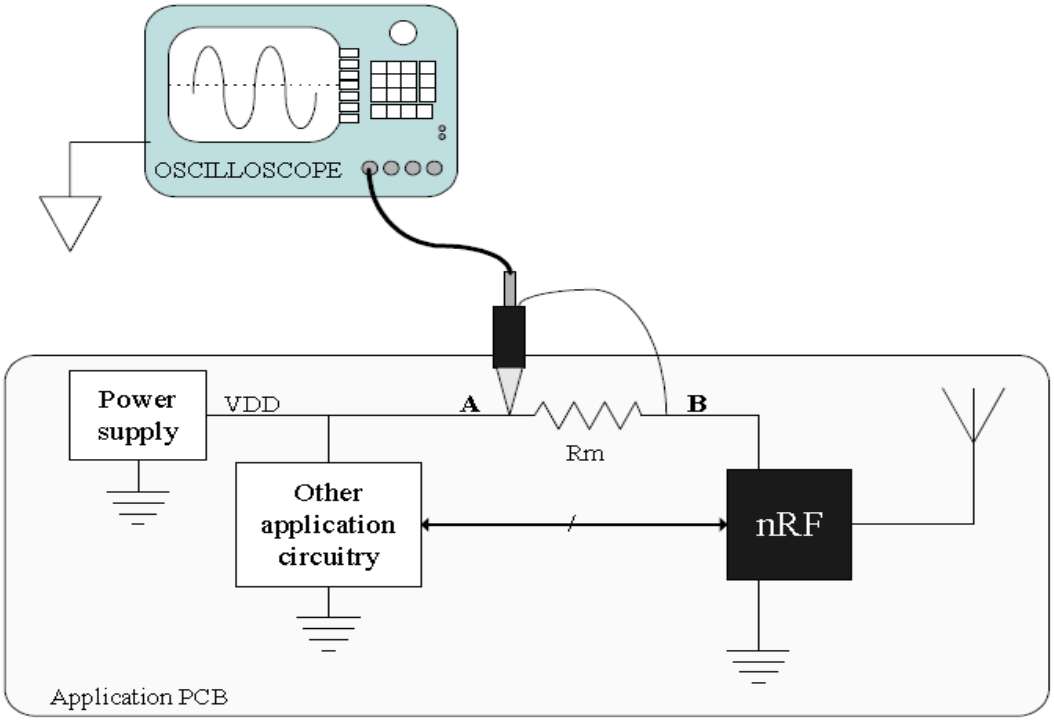


Figure 34. Hardware setup in SurNet node power consumption measurements (Nordic Semiconductor 2008: 39).

Average current means the average of every instantaneous current value from zero to the peak in different phases. To calculate the average current consumption of the SurfNet node equipped with sensors, the following formulas are used. In (3), the peak current multiplies by its corresponding duration time. Then it makes the sum of the results in each phase. After that, it makes the summation divided by the total duration time of different phases. That is the average current in this circuit. Moreover, (3) can be transformed to (4) by turning the peak current to the result of dividing the peak voltage by the resistance. (5) shows the final transformation result.

$$I_{AV} = \frac{\sum_n I_{peak_n} \cdot t_n}{\sum_n t_n} \quad (3)$$

$$= \frac{\sum_n \frac{V_{peak_n}}{R} \cdot t_n}{\sum_n t_n} \quad (4)$$

$$= \frac{\sum_n V_{peak_n} \cdot t_n}{R \cdot \sum_n t_n} \quad (5)$$

By using the oscilloscope shown in Figure 34, we can measure the peak voltage of the test resistor (1 ohm) in different phases of SurfNet node, as well as the corresponding time of each phase in the node. According to (5), we can multiply each peak voltage with the related time in different phases and then make a summation of them. In addition, we make a summation of the time of different phases and then times one, which is the resistance value of the test resistor. Finally, dividing the first summation by the time summation, gives the average current value of the SurfNet node with its sensors.

In this experiment, one SurfNet node equipped with temperature and humidity sensors, is configured with the following settings:

- Sleeping time: 2 seconds;
- The RF channel is set as (2400 + 40) MHz;
- 16-bits CRC is enabled;
- RF data rate: 2 Mbps;
- RF output power in TX mode: 0 dBm;
- Auto Acknowledgement function on data pipe is enabled;
- Automatic retransmission is set as:
  - Auto retransmit delay: wait 250 μs;
  - Auto retransmit count: up to 2 retransmissions if fails.

5.2. Communication Capability

5.2.1. Indoor Scenario

The indoor experiment was done in the passageway in Technobothnia Laboratory. Both sides were metal walls, as presented in Figure 35. There was a line of sight between the transmitter and the receiver in the passageway.

Figure 35. Indoor test environment.

The results of the indoor packet loss experiment are presented in Figure 36. The symbol of black point indicates the percentage of lost packets out of 255 packets in one distance, and the blue stars are the averages of five sets of measurements at each location. The blue line, which connects the average values, shows the packet loss as a function of the communication distance.

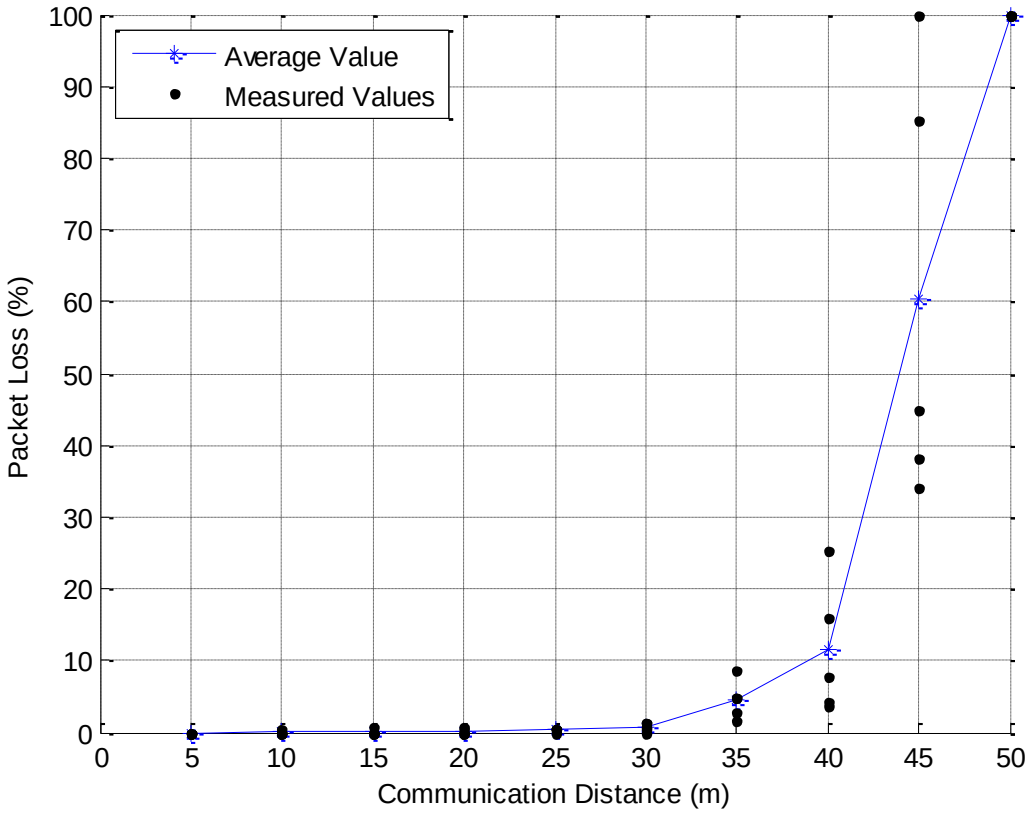


Figure 36. Observed packet loss in the indoor environment (0 dBm transmission power was applied).

As the result shows, when the communication distance is no longer than 25 meters, the packet loss is close to zero. When it reaches to 35 meters, the packet loss increases and its standard deviation also starts to increase, but the average

value is still less than 5%. In this case, the system can be still in use. However, when the communication distance is increased to 45 meters, the communication cannot be considered reliable anymore because the packet loss is over 50%.

5.2.2. Outdoor Scenario

Similarly, the outdoor experiment was carried out in the open space, which the passageway outside Fabriikki building. The test condition can be shown in Figure 37. There were no obstacles in the air between the transmitter and the receiver.

Figure 37. Test condition of outdoor.

The results of the outdoor scenario are presented in Figure 38. The communication reliability remains quite good up to 20 meters distance. Once the distance increases from 20 meters, the communication reliability weakens rapidly, as indicated by the increase of the packet loss and the packet loss standard deviation. Finally, the connection was completely lost after 30 meters.

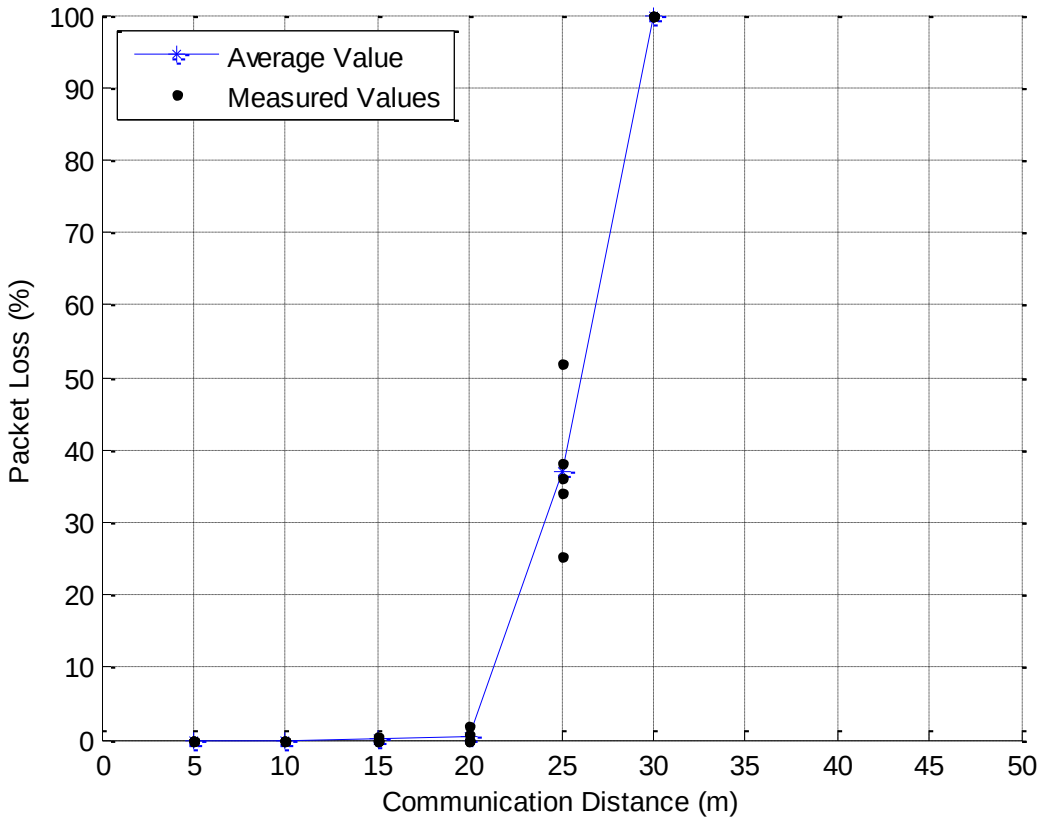


Figure 38. Observed packet loss in the outdoor environment (0 dBm transmission power was applied).

Compared to the indoor tests, the result indicates that the communication reliability is weaker in this outdoor test scenario. The result was unexpected

because in principle there should be more signal attenuation in the indoor environment than in open space and the packet loss of outdoor test should be better. Specifically, the first thing that explains the result is the attenuation caused by the brick walls and bushes in the outdoor environment. The radio signal attenuation might be increased in this environment. The second thing that affected on the result was the metal walls in the indoor environment, which might have improved the radio environment there.

5.3. Power Consumption

The power consumption of two subsystems is measured in this part. These subsystems are the sink, which contains UWASA Node that operates as a gateway and a SurfNet node connected to the node by SPI, and SurfNet nodes equipped with sensors.

5.3.1. UWASA Node and SurfNet Node in the Sink

The average current of the devices can be measured by using a serial ampere meter. The average of the measured current taken by the SurfNet node in the sink was 18.35 mA. This power is supplied by UWASA Node into which the SurfNet node is connected in the sink. The measured joint power consumption of the UWASA Node and the SurfNet node in the sink was 380 mA. However, in this developed system, they can have an external power supply by the Linux server.

5.3.2. SurfNet Node with Sensors

As explained in Chapter 5.1, the average power consumption of the SurfNet node with its sensors, is computed by measuring the current in different operating modes in the node. To obtain the current in the circuit, we can measure the voltage of a test resistor that is connected to the circuit in series.

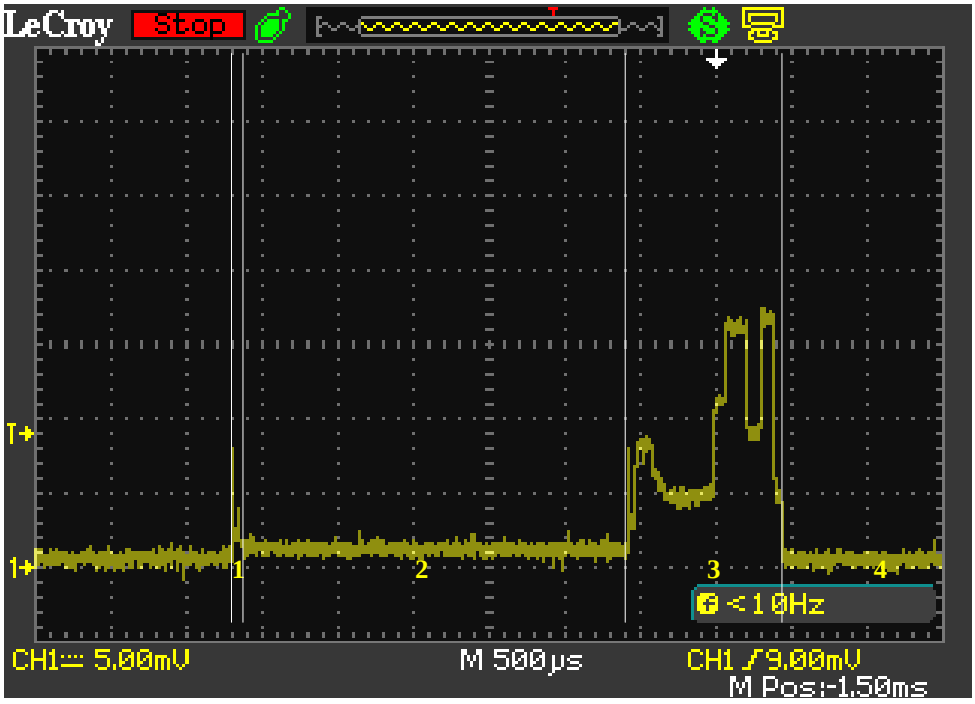


Figure 39. The current consumption of the SurfNet node equipped with temperature and humidity sensors, in different operating modes.

Generally, there are four main operation phases for the SurfNet node with the sensors, which are waking up, settling the sensors, RF active period, and sleeping. Figure 39 presents the result of current consumption measurements from the SurfNet node, which is equipped with temperature and humidity

sensors. The four main different phases are noted by the yellow numbers in the figure, including the sleeping mode that is two seconds here:

- 1. Waking up from sleeping mode;
- 2. Powering and settling down the sensors;
- 3. RF active period;
- 4. Sleeping.

Furthermore, as the figure shows, in the RF active period, there are also several operation phases. The details are shown in Figure 40, which presents the result of current consumption measurements from the SurfNet node with sensors, in one RF active period. In Figure 40, in total, there are nine different operating phases in one RF active period, noted by the yellow numbers:

- 3.1. Waking up from sleeping mode;
- 3.2. Setting RTC;
- 3.3. Starting ADC and getting measurements;
- 3.4. Preparing transmission interrupt;
- 3.5. Starting execution and upload data to TX buffer;
- 3.6. RF transmission;
- 3.7. Settling as a receiver;
- 3.8. RF listening for ACK;
- 3.9. Download data in RX buffer.

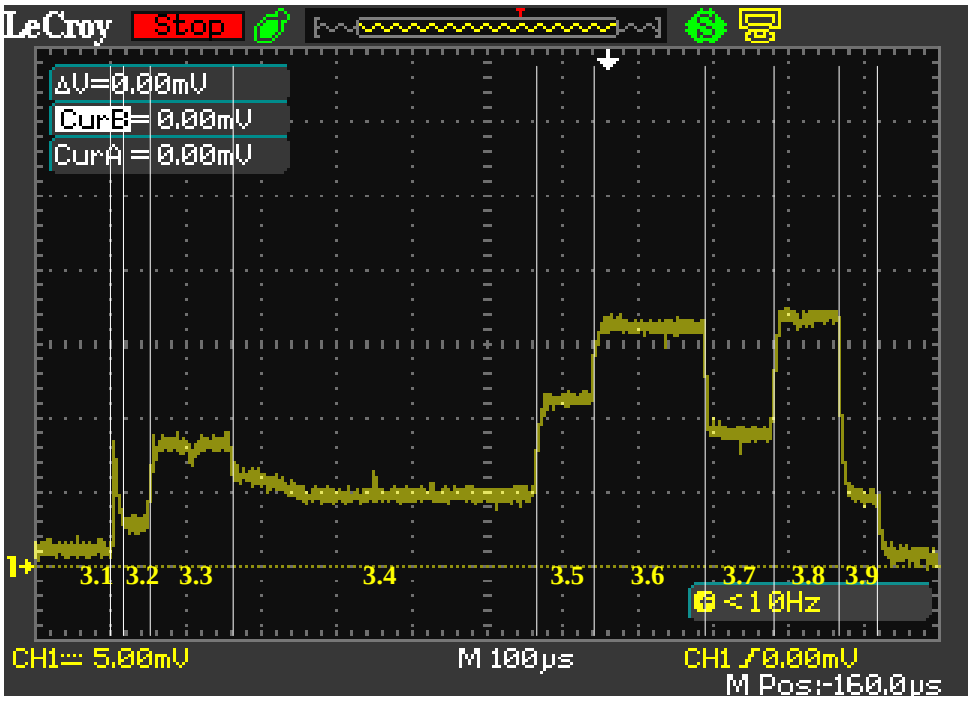


Figure 40. The current consumption of the SurfNet node sensors, in RF active period.

Besides, the SurfNet node consumes 3 μA averagely when it stays in sleeping mode with timers on (Nordic Semiconductor 2010: 183). After measuring the voltage of other phases, the result can be obtained, which is shown in Table 9. With the table, we can obtain the average current consumption of the SurfNet node with the humidity and temperature sensors, by applying the corresponding values into Formula 5.

Table 6. Measured values in each operating mode of SurfNet node with sensors.

Number	Phase Description	Voltage(mV)	Time(μs)
1.1	Waking up from sleeping mode	8.5	12
1.2	Setting RTC, powering on sensors and starting to sleep	2.4	45
2	Settling sensors	0.7	2560
3.1	Waking up from sleeping mode	8.6	12
3.2	Setting RTC	2.4	43
3.3	Starting ADC and getting measurements	8.8	116
3.4	Preparing transmission interrupts	5.2	404
3.5	Execution and upload data to TX buffer	11.8	76
3.6	RF transmission	16.8	148
3.7	Settling as a receiver	9.2	92
3.8	RF listening for an ACK	18.8	91
3.9	Download data in RX buffer	4.4	48
4	Sleeping	0.003	2000000

$$I_{AV} = \frac{(8.5\cdot 12+2.4\cdot 45)\,\text{mV}\cdot\mu\text{s}}{1\ \text{ohm}\cdot(12+45+2560+12+43+116+404+76+148+92+91+48+2000000)\ \mu\text{s}} + \frac{(0.7\cdot 2560)\,\text{mV}\cdot\mu\text{s}}{1\ \text{ohm}\cdot(12+45+2560+12+43+116+404+76+148+92+91+48+2000000)\ \mu\text{s}} + \frac{(8.6\cdot 12+2.4\cdot 43+8.8\cdot 116+5.2\cdot 404+11.8\cdot 76+16.8\cdot 148+9.2\cdot 92+18.8\cdot 91+4.4\cdot 48)\,\text{mV}\cdot\mu\text{s}}{1\ \text{ohm}\cdot(12+45+2560+12+43+116+404+76+148+92+91+48+2000000)\ \mu\text{s}} + \frac{(0.003\cdot 2000000)\,\text{mV}\cdot\mu\text{s}}{1\ \text{ohm}\cdot(12+45+2560+12+43+116+404+76+148+92+91+48+2000000)\ \mu\text{s}} \approx 0.0087\ \text{mA}$$

As a result, once the sleeping time is set to two seconds and the settling down time for the sensors is 2.5 milliseconds, the average power consumption of sensor node with its sensors is 0.0087 approximately. In this situation, if the

node is equipped with a button battery of 210 mAh, the lifetime of the node can be 32 months, which equals two and a half years.

However, because the maximum sleeping time set by RTC is 2 seconds, the node needs to wake up every 2 seconds, even though it is not the node's duty period to transmit. For example, if the sleep time is 8 seconds, the node still wakes up three times and sets the RTC without any other operations. After it wakes up, the node checks whether its sleeping is done. Every time the node wakes up from sleeping mode, it consumes about 8.5 mA and takes 12 μs for waking up and consumes about 2.4 mA and takes 45 μs for setting RTC, no matter it continues to sleep or wakes up to work, as shown in Table 6.

In the developed system, actually, the sleeping time represents the sampling interval. The power consumption of the SurfNet node varies along with the sampling interval, which is the inverse of the sample rate. Figure 41 presents the power consumption as a function of the sampling interval in the SurfNet node with sensors. By setting the different sampling intervals as 0.5, 1, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, and 38 seconds, the corresponding power consumption of the SurfNet node with sensors can be calculated based on the measurements and Formula 5 presented in Chapter 5.1. As shown in the figure, each blue point represents an average current consumption at one certain sampling interval.

As the figure shows, when the interval between two samples is set to less than 8 seconds, there is a sharp decrease of the average current consumption. When the interval increases from 10 seconds to further, the consumption continues to decrease slightly. In addition, the longer sampling interval is set, the more

power the node saves thus the longer lifetime the node obtains. For example, if the sampling interval is set to 30 seconds, then the power consumption can be around 3.48 μA. As a result, by using a button battery of 210 mAh and the sampling interval of 30 seconds, the lifetime of the SurfNet node with temperature and humidity sensors can be about 6 years and 7 months theoretically. In our system, we set the sampling interval between each packet to 8 seconds, which is 0.125 Hz. Because we require the system updates the measurements fast enough, while consumes power as less as possible.

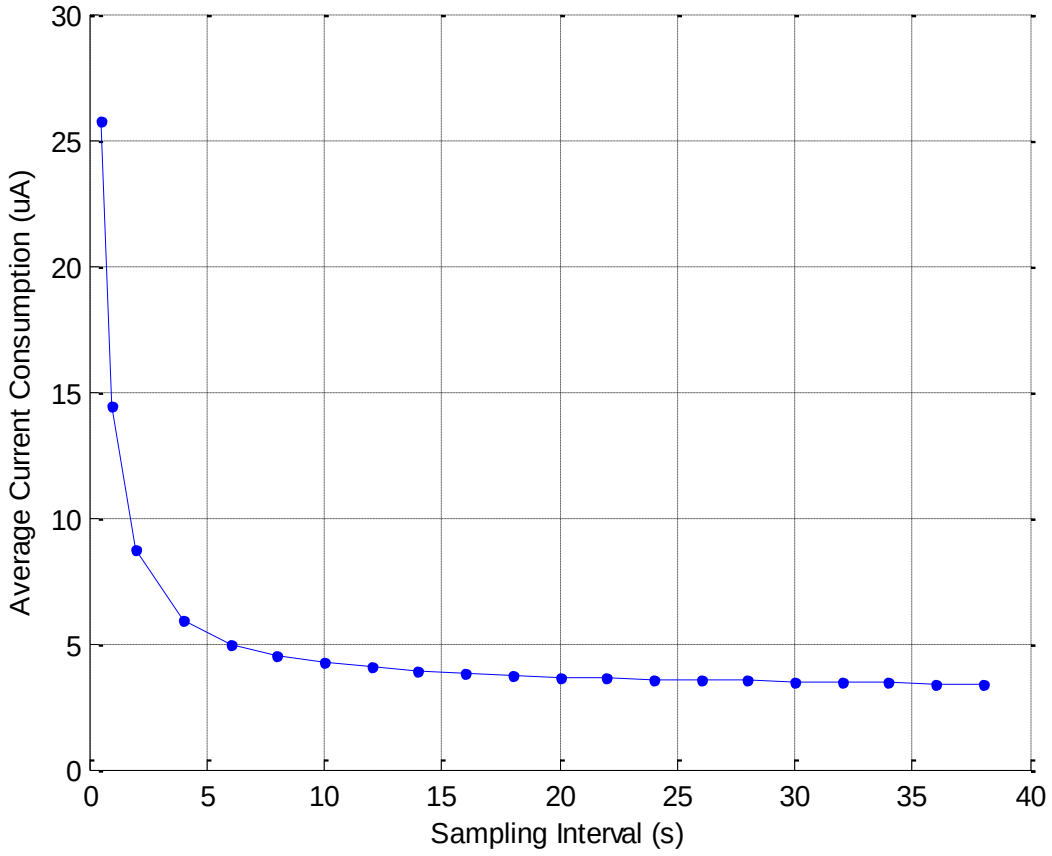


Figure 41. Average current consumption as a function of the sampling interval of SurfNet node.

5.4. System Performance Evaluation

In this section, to guarantee the communication quality and the low power consumption features of the developed system, two experiments were deployed to evaluate the performance. These results are based on the improved software as described in the last chapter.

Based on the first experiment, the observed reliable communication range can be 30 meters in indoor scenario and 20 meters outdoor scenario. It shows that the communication in indoor scenario is more reliable than the one in outdoor. One reason is that the objects in outdoor environment might increase the attenuation of the radio signal. Another reason is that the metal walls in the indoor environment might have improved the radio environment of the wireless communication. Thus, the obtained result of this experiment might change if the test condition is different.

Moreover, the result shows that the communication range is limited in both scenarios. It is because SurfNet node has a limited RF output power that is 0 dBm in TX mode, and a limited receiving sensitivity when in RX mode, which is -82 dBm at the data rate of 2 Mbps. These conditions have an influence on the reliable wireless communication range of SurfNet nodes in WSN. In our system, the reliable communication range between the SurfNet nodes is acceptable as required.

The second experiment shows the SurfNet nodes with sensors can consume lower power by increasing the sampling interval of the packet. In the experiment, we applied to the nodes with 2 seconds sampling interval and 2

Mbps data rate that is the maximum rate in the device. In this case, the average current consumption is approximately 0.0087. That is, if the node is equipped with a button battery of 210 mAh, the lifetime of the node equals two and a half years. Additionally, if the sampling interval is set to 30 seconds and other conditions remain the same, the lifetime of the node can be about six years and seven months in theory. Therefore, except the subsystems that are supplied by the external power source, the SurfNet node with sensors powered by a limited battery can consume lower power by using the developed software. This also ensures the practicality of the developed system.

Increasing the sampling rate might slow down the update rate of the measurements in the Linux server. Also, decreasing the data rate can shorten the communication range. However, for some long-term monitoring applications, which do not require high data update rate and long-range wireless communication, it is acceptable to apply this developed system. Because it can be developed further to improve the power efficiency, so that the system can operate even longer, for example, increasing the sampling interval and/or decreasing the data rate in the node.

In one word, through the experiments carried out in this chapter, it turns out that the developed system satisfies the requirements that we demanded. It can be applied as a wireless solution of smart home remote application to monitor the environment.

6. CONCLUSIONS AND FUTURE WORK

6.1. Conclusions

The performance of the developed wireless sensor system has been evaluated in the last chapter. It can be used to monitor environmental circumstances. This work is completed by developing a WSN and a Linux server, which is connected to a remote site over the 3G connection. Therefore, the measured data in the assigned environment can flow from every single sensor node to the sink node, then to the Linux server and eventually arrive remotely at the user end by 3G communication.

Generally speaking, this study has achieved two main goals. The primary goal is to design a low-power consumption WSN appropriate for the environmental monitoring and control. The first goal has been achieved with the SurfNet node hardware and software architecture, among which there are transmitters and receivers in the star network topology. Moreover, the developed software has been applied to achieve better communication stability and power efficiency of the system.

The second goal is to set up the 3G communication from the WSN to the sink and to the user end. In other words, this goal is to manage a module for the sink so that it is capable of communicating by 3G to the user end. This goal has been achieved by using the Linux board FOX G20 with a 3G module. As a result, a newly designed hardware interface has been successfully produced as a prototype, namely, the joint use of SurfNet nodes, UWASA Node and a Linux

server with a 3G module.

Consequently, this application can be also develop for industry, medical use and some other fields, by expanding the use of the wireless sensor nodes and creating thoroughly-examined wireless monitoring and control system with all the necessary implementations and improvements.

6.2. Future Work

Firstly, there is a need for further study to improve the algorithms and mechanisms of the nodes. To improve the performance of the developed system, we might need to research the time synchronization algorithms for the WSN, which makes the communication between the transmitter and the receiver synchronous. At the same time, the power consumption of the node should be re-evaluated as well in this case, because there can be more time for a node staying in RX mode, which consumes more power.

In addition, to improve the reliability of the wireless communication, the antenna in the node can be developed further so that the transmitting signal gain can be increased. Thus, the reliable communication range can be increased, as well.

Moreover, the data security of the wireless communication can be further improved if it is required. For example, the encryption and decryption function can be enabled. Furthermore, the AES firmware is available in NRF24LED device. It can also be developed for the security of the data packet while

transmission.

Moreover, the SurfNet node equipped with sensors, can be developed for other purposes, by changing the sensors. For example, the node can equip with an air pressure sensor to monitor the air pressure in the environment. Moreover, the node can be further developed for positioning or localization in a wireless network.

For the network architecture, the range of the network could be more extensive, for example, one larger multi-hop WSN. In this case, some advanced algorithms and mechanisms might be focused and developed, for example, the suitable time synchronization, precise mechanism for self-adaption, and so on.

This field of research topics could be compared to a gold mine and it has a lot to deeply dig for, so that it makes increasingly more contributions to the development of the technology.

REFERENCES

Akyildiz, Ian F., Weilian Su, Yogesh Sankarasubramaniam & Erdal Cayirci (2002). *A Survey on Sensor Networks*. IEEE Communications Magazine, Volume: 40, Issue: 8, Aug. 2002, pp. 102–114.

ACME System. *FOX Board G20 - Linux Embedded SBC* [online]. Available from the Internet: <URL: http://www.acmesystems.it/FOXG20>.

Barry, Richard (2009). *Using the FreeRTOS Real Time Kernel: A Practical Guide* [online]. Available from the Internet: <URL: ftp://ftp.cs.sjtu.edu.cn:990/hongzi/embedded%20systems/referece%20books/Using+the+FreeRTOS+Real+Time+Kernel+-+a+Practical+Guide.pdf>.

Cuhac, Caner, Huseyin Yigitler (2012). *The UWASA Node Reference Manual 3.0.0*. Vaasa, Finland: University of Vaasa, 2012.

IEEE Standard 802.15.4 (2011). *IEEE Standard for Part 15.4: Low-Rate Wireless Personal Area Networks (WPANs)*. Sep. 2011.

Nordic Semiconductor (2008). *RF Performance Test Guidelines: White Paper v1.0* [online]. Available from the Internet: <URL: http://www.bdtic.com/DownLoad/NORDIC/RF_Performance_Test_Guidelines_v1_0.pdf>.

Nordic Semiconductor (2010). *Ultra-low Power Wireless System On-Chip Solution: nRF24LE1 Product Specification, v1.6* [online]. Available from the Internet:

<URL:http://www.nordicsemi.com/eng/nordic/download_resource/10875/3/43943441>.

Palomäki, Heikki (2010a). *SurfNet data sheet* [online]. Available from SeAMK embedded systems resource page: <URL: http://lompsa.seamk.fi /sulautetut/>.

Palomäki, Heikki (2010b). *SURFprogrammer data sheet* [online]. Available from SeAMK embedded systems resource page: <URL: http://lompsa.seamk.fi /sulautetut/>.

Palomäki, Heikki & Marko Huhta (2010). *Low Power Synchronization in Wireless Network*. Second Workshop on Wireless Communication and Applications (WoWCA2010), 5–6 May 2010, Vaasa, Finland.

Texas Instruments (2009). *CC2431 Data Sheet* [online] [cited 8 Jun. 2013]. Available from the Internet: <URL: http://www.ti.com/lit/ds/symlink/cc2431.pdf>.

Virrankoski, Reino (2012). *Generic Sensor Network Architecture for Wireless Automation (GENSEN)* [online]. Vaasa, Finland: University of Vaasa, 2012. Available from Internet: <URL: http://www.uva.fi/materiaali/pdf/isbn_978-952-476-387-5.pdf>.

Wikipedia. *Wireless Sensor Network* [online] [cited 19 Sep. 2013]. Available from Internet: <URL: http://en.wikipedia.org/wiki/Wireless_sensor_network>.

Yigitler, Huseyin, Reino Virrankoski & Mohammed Elmusrati (2010). *Stackable Wireless Sensor and Actuator Network Platform for Wireless Automation: The UWASA Node*, In: Aalto University Workshop on Wireless Sensor Systems [online]. Aalto University Wireless Systems Group. Helsinki: Aalto University.

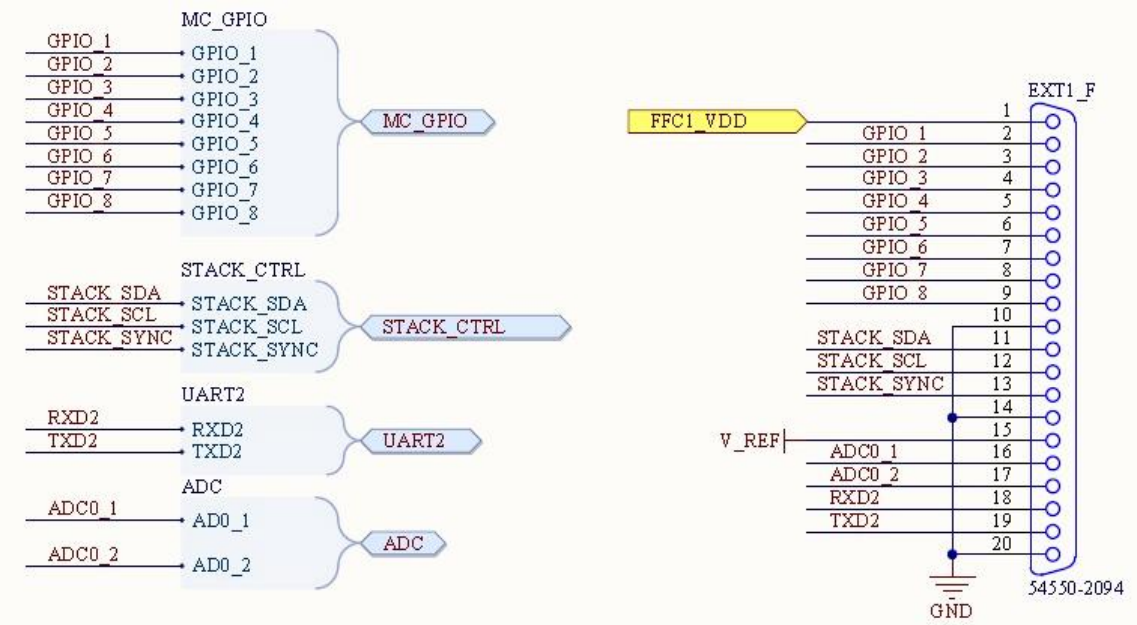
Yigitler, Huseyin (2010). *The UWASA Node Reference Manual*. 1st Ed. Vaasa, Finland: University of Vaasa, 2011.

Yigitler, Huseyin (2011). *PCB of Generic Slave Module*. Aalto University Wireless Systems Group. Helsinki, Finland: Aalto University.

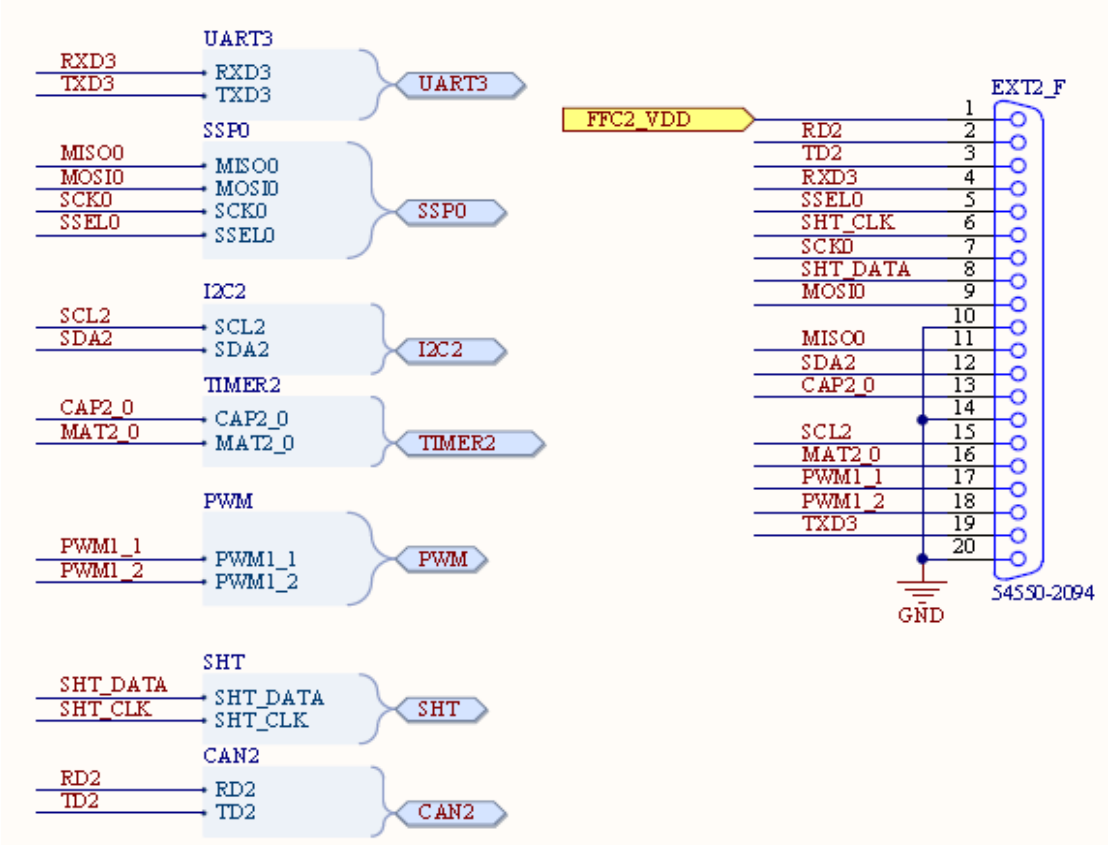
ZigBee™ Alliance (2008). *ZigBee Specification* [online]. Available from the I nternet: <URL: http://people.ece.cornell.edu/land/courses/ece4760/FinalProjects/s2011/kjb79_ajm232/pmeter/ZigBee%20Specification.pdf>.

APPENDICES

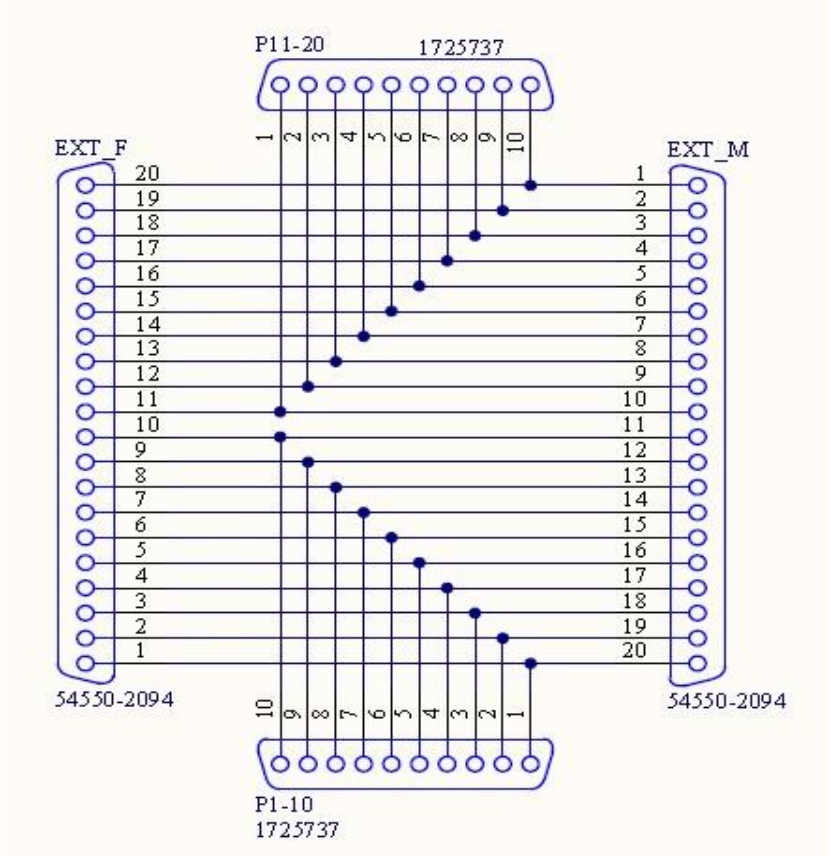
APPENDIX 1. Schematics of generic slave module: MC_EXT_FFC1.



APPENDIX 2. Schematics of generic slave module: MC_EXT_FFC2.

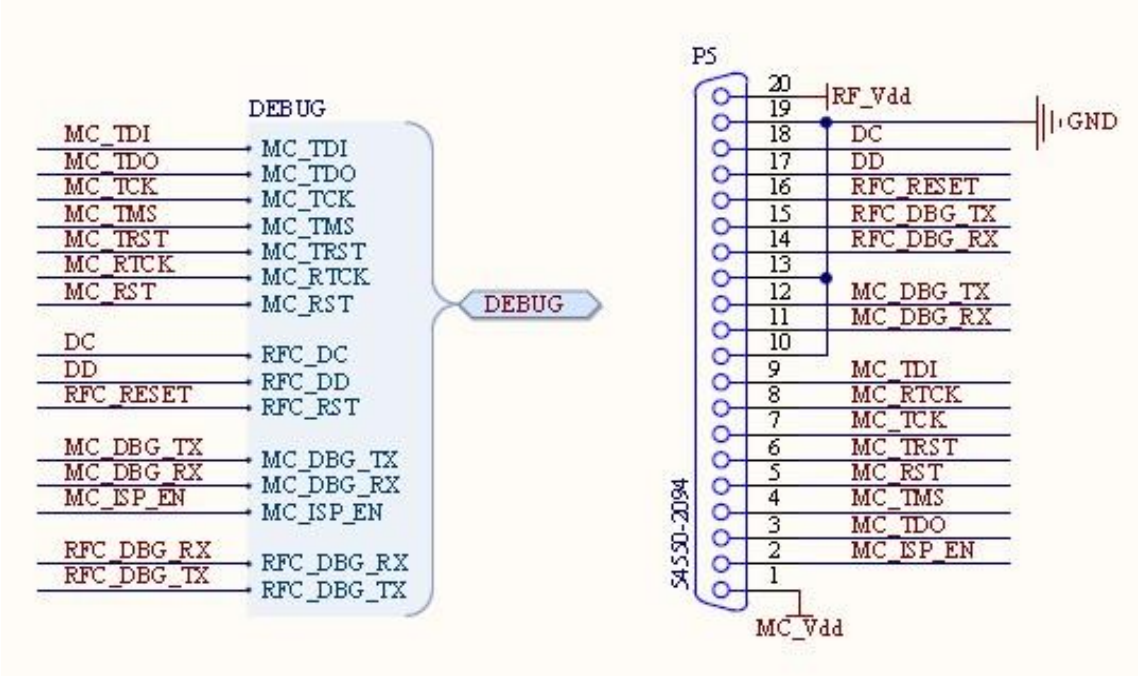


APPENDIX 3. Schematics of FFC convertor of UWASA pRoot.

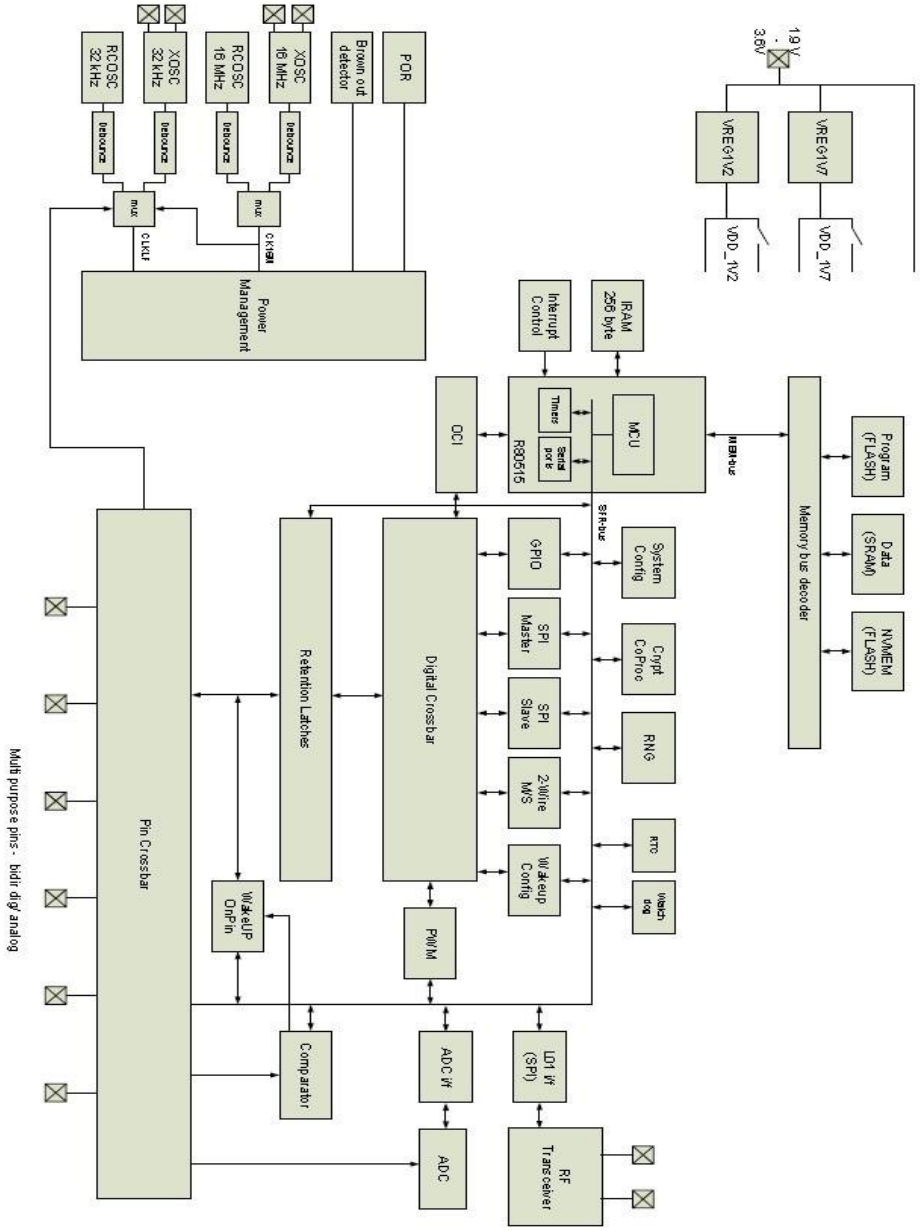


APPENDIX 4. Schematics of generic slave module: DEBUG_FFC.

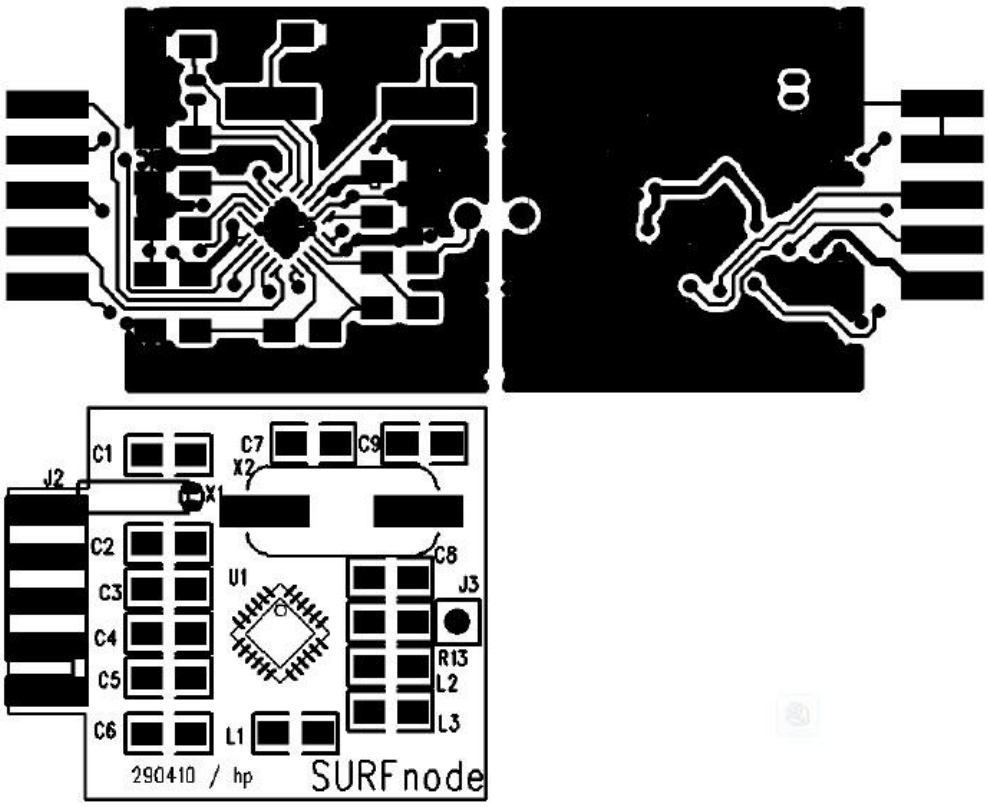
Pin MC_RST is used for resetting the main controller of UWASA Node.



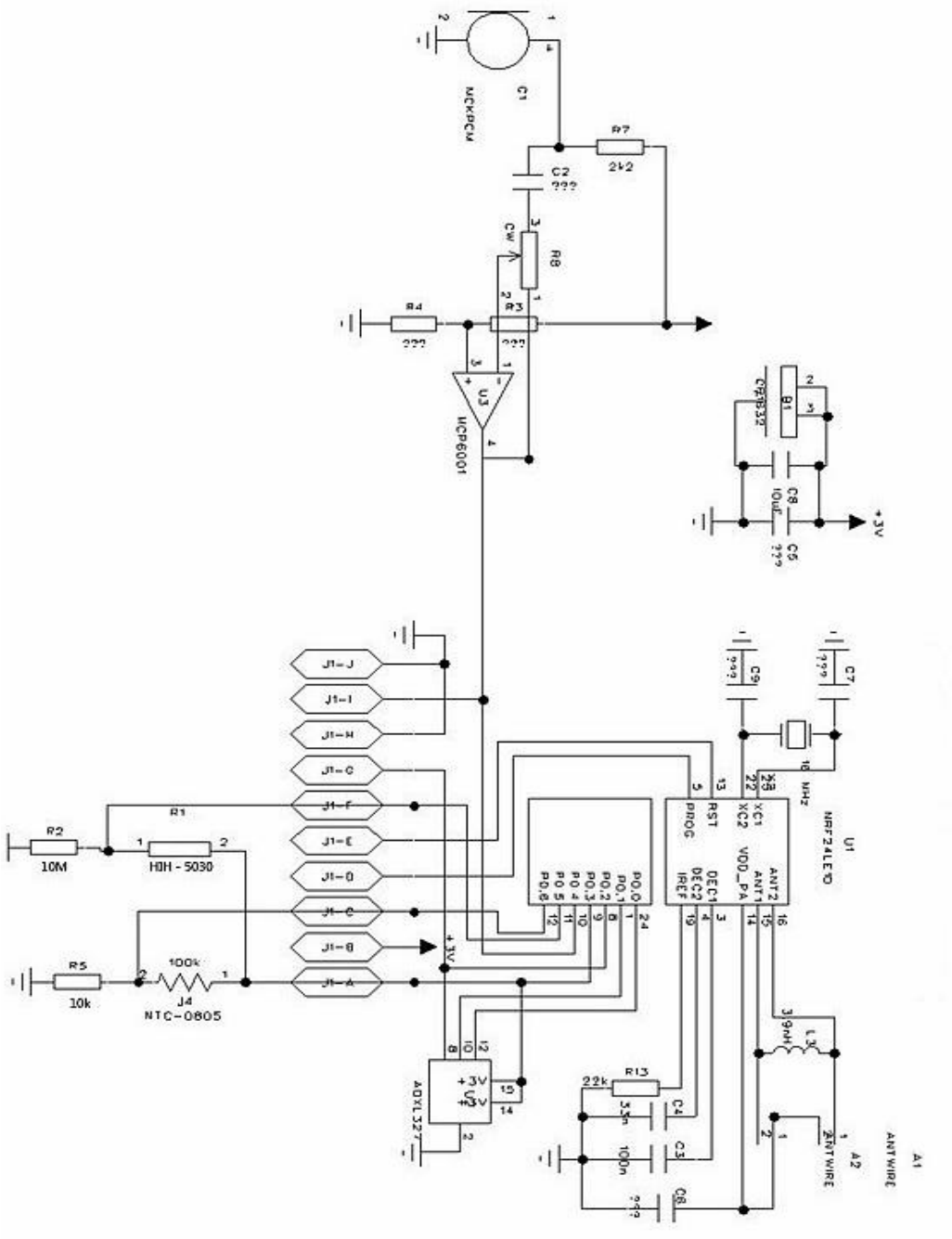
APPENDIX 5. Hardware architecture of nRF24LE1.



APPENDIX 6. PCB layout of SurfNet node.

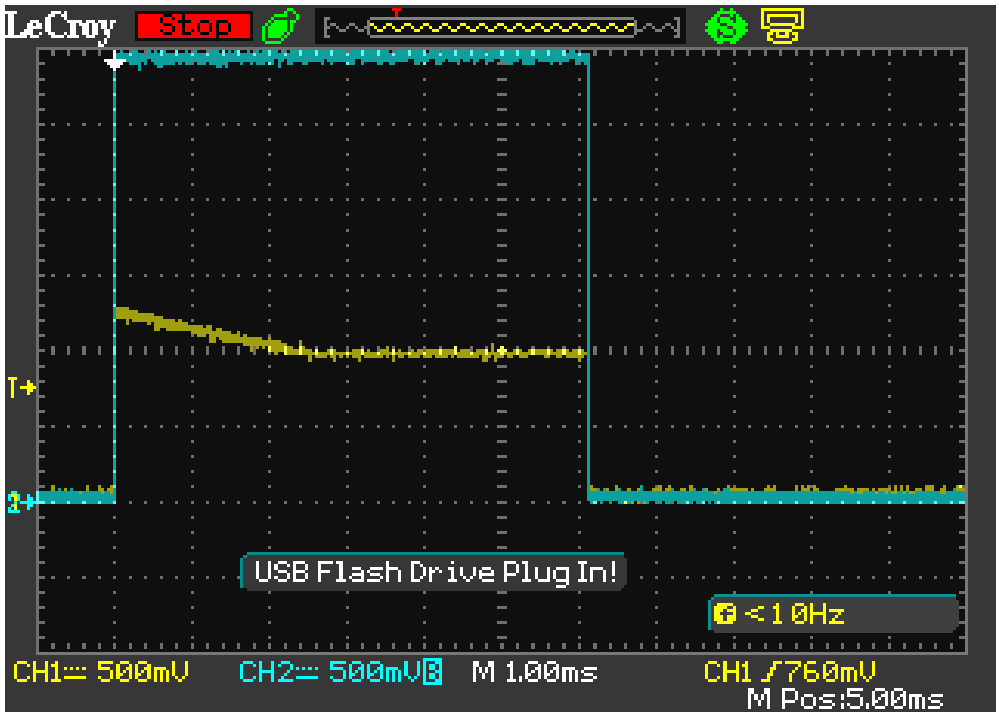


APPENDIX 7. Circuit schematics of SurfNet node with sensors.

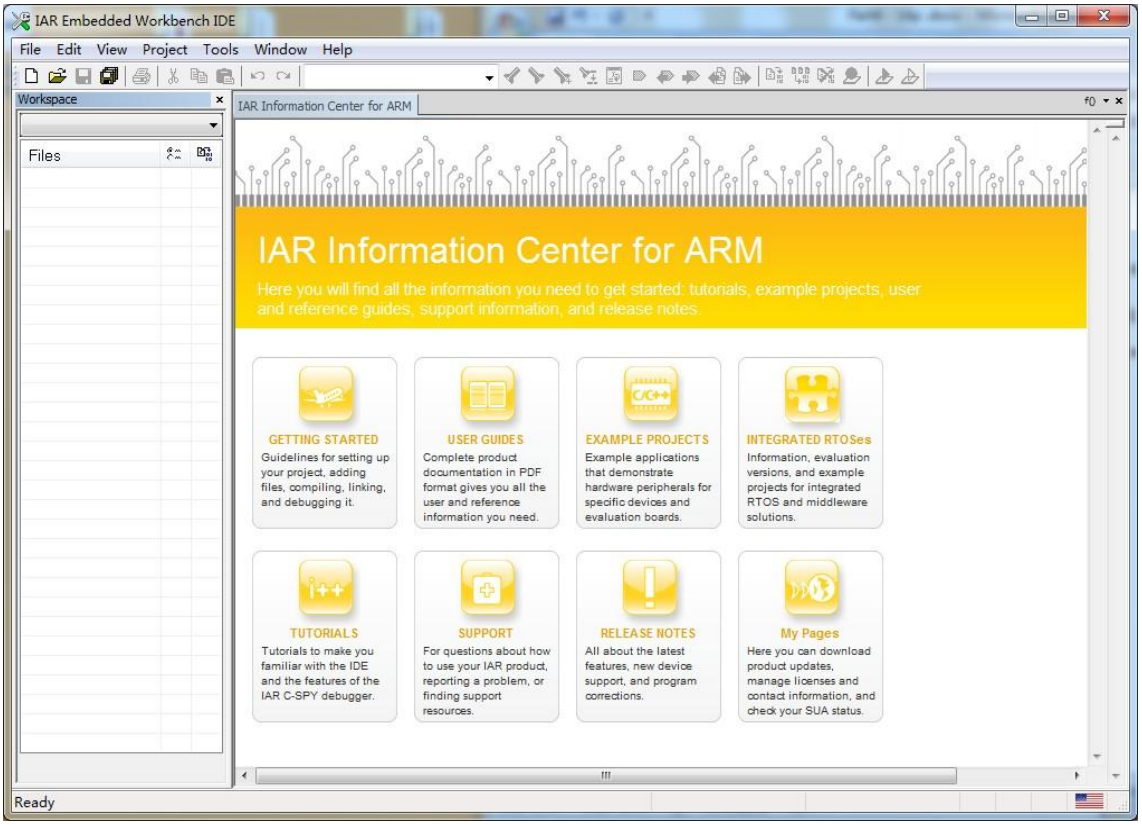


APPENDIX 8. The changing output voltage of the resistor in humidity sensor.

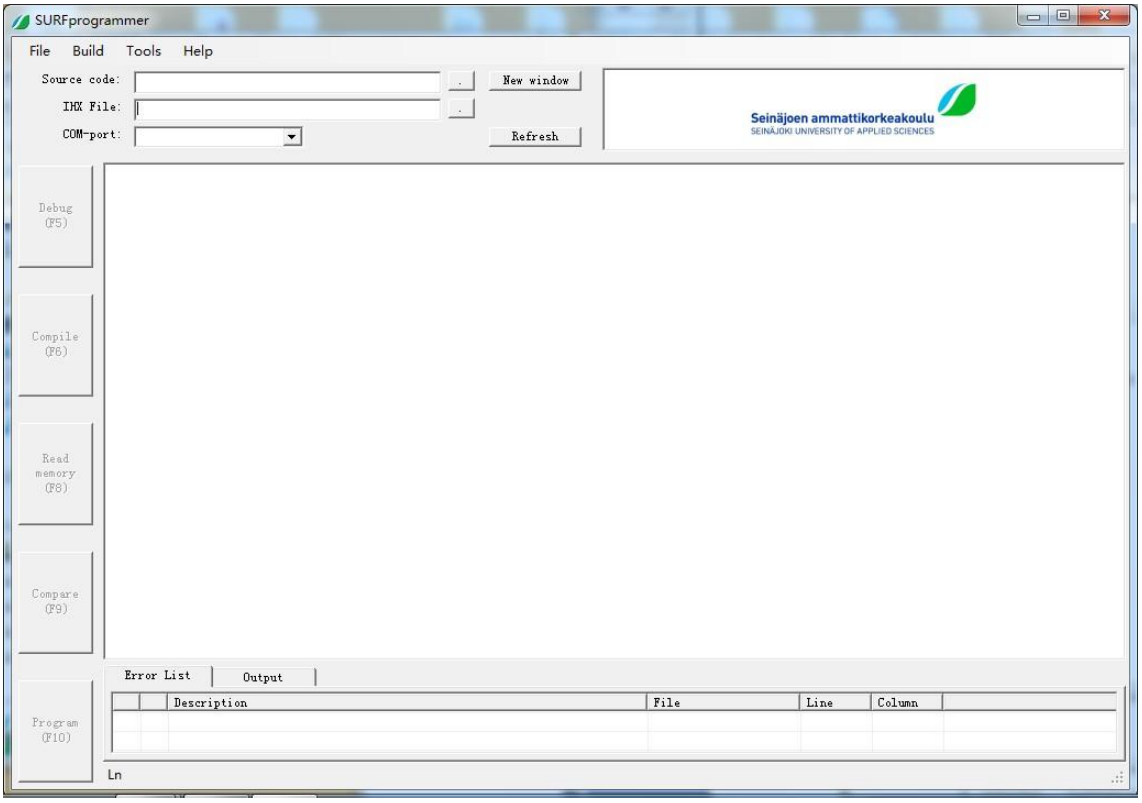
In the figure, the blue line represents the voltage output of the power source for humidity sensor, while the yellow line shows the voltage of the resistor in humidity sensor. As the figure shows, there should be a time delay for humidity sensor stabilizing. Otherwise, the measured humidity values would be incorrect.



APPENDIX 9. IAR embedded workbench for ARM IDE.



APPENDIX 10. Interface of SURFprogrammer by Seinäjoki USA.



APPENDIX 11. Sensor Node 5 and Node 6 calibrate measurements with a digital thermo-hygrometer in environmental monitoring test.