Exploiting Wireless Sensors

A gateway for 868 MHz sensors

ALBERT LÓPEZ
and
FRANCISCO JAVIER SÁNCHEZ

KTH Information and
Communication Technology

Degree project in
Communication Systems
Second level, 30.0 HEC
Stockholm, Sweden

Exploiting Wireless Sensors

A gateway for 868 MHz sensors

Albert López
albertlg@kth.se

Francisco Javier Sánchez
fjsg@kth.se

2012-06-20

Supervisor and Examiner: Gerald Q. Maguire Jr.

School of Information and Communication Technology
KTH Royal Institute of Technology
Stockholm, Sweden

Abstract

The great interest in monitoring everything around us has increased the number of sensors that we utilize in our daily lives. Furthermore, the evolution of wireless technologies has facilitated their ubiquity. Moreover, is in locations such as homes and offices where exploitation of the data from these sensors has been more important. For example, we want to know if the temperature in our home is adequate, otherwise we want to turn on the heating (or cooling) system automatically and we want to be able to monitor the environment of the home or office remotely. The knowledge from these sensors and the ability to actuate devices, summon human assistance, and adjust contracts for electrical power, heating, cooling, etc. can facilitate a myriad of ways to improve the quality of our life and potentially even reduce resource consumption.

This master's thesis project created a gateway that sniffs wireless sensor traffic in order to collect data from existing sensors and to provide this data as input to various services. These sensors work in the 868 MHz band. Although these wireless sensors are frequently installed in homes and offices, they are generally not connected to any network. We designed a gateway capable of identifying these wireless sensors and decoding the received messages, despite the fact that these messages may use a vendor's proprietary protocol. This gateway consists of a microcontroller, a radio transceiver (868-915 MHz), and an Ethernet controller.

This gateway enables us to take advantage of all the data that can be captured. Thinking about these possibilities, simultaneously acquiring data from these various sensors could open a wide range of alternatives in different fields, such as home automation, industrial controlling… Not only can the received data be interesting by itself; but when different sensors are located in the same environment we can exploit this data using sensor fusion. For example, time differences in arrival and differences in signal strength as measured t multiple receivers could be used to locate objects.

The final aim of this thesis project is to support diverse applications that could be developed using the new gateway. This gateway creates a bridge between the information that is already around us and our ability to realize many new potential services. A wide range of opportunities could be realized by exploiting the wireless sensors we already have close to us.

Sammanfattning

Det stora intresset för att övervaka allt omkring oss har ökat antalet sensorer som vi använder i vårt dagliga liv. Dessutom har utvecklingen av trådlösa tekniker underlättat de har stor utbredning. Dessutom är på platser som hem och kontor där utnyttjandet av data från dessa sensorer har varit viktigare. Till exempel vill vi veta om temperaturen i vårt hem är tillräcklig, annars vill vi slå på värmen (eller kyla) automatiskt och vi vill kunna övervaka miljön i hemmet eller på kontoret på distans. Data från dessa sensorer och förmåga att aktivera enheter kalla mänsklig assistans och justera avtal för el, värme, kyla, osv. kan underlätta en myriad av olika sätt att förbättra kvaliteten på våra liv och potentiellt även minska resursförbrukningen.

Detta examensarbete syftar till att sniffa trådlösa sensorn trafik i syfte att samla in data från befintliga sensorer och att tillhandahålla sådan information som underlag till olika tjänster. Dessa sensorer arbetar i 868 MHz-bandet. Även om dessa ofta installeras i hem och kontor, är de i allmänhet inte ansluten till något nätverk. För att förverkliga vårt mål kommer vi att utforma en gateway som kan identifiera dessa trådlösa sensorer och avkoda den mottagna meddelanden, trots att dessa meddelanden kan använda en leverantör egenutvecklade protokoll. Denna brygga består av en mikrokontroller, en sändtagare (868 till 915 MHz), och en Ethernet-styrenhet.

Gateway bör göra det möjligt för oss att dra nytta av alla de uppgifter som möjligen kan fångas. Funderar om dessa möjligheter, samtidigt insamling av data från dessa olika sensorer kan öppna ett brett spektrum av alternativ i olika områden, såsom hem automation, industriell kontrollerande ... Inte bara kan de mottagna uppgifterna vara intressant i sig självt, men när olika sensorer finns i samma miljö kan vi utnyttja detta data med hjälp av sensor fusion. Till exempel skulle tidsskillnader i ankomst och skillnader i signalstyrka uppmätt med flera mottagare användas för att lokalisera föremål.

Det slutliga målet med denna avhandling är att stödja olika applikationer som skulle kunna utvecklas med hjälp av utformade gateway. Denna gateway kommer att skapa en initial brygga mellan all information omkring oss och vår förmåga att förverkliga många tjänsteleverantörer möjligheter. Ett brett utbud av möjligheter kan realiseras genom att utnyttja de trådlösa sensorerna vi redan har nära till oss.

Table of Contents

List of Figures

List of Tables

List of Acronyms and Abbreviations

AC	Alternating Current
ADC	Analog-to-Digital Converter
AES	Advanced Encryption Standard
ASK	Amplitude Shift Keying
BCD	Binary Coded Decimal
CCS	Code Composer Studio
CEPT	European Conference of Postal and Telecommunications
CRC	Cyclic Redundancy Check
CRT	Cathode Ray Tube
CSMA/CA	Carrier Sense Multiple Access/Collision Avoidance
CSMA/CD	Carrier Sense Multiple Access/Collision Detection
DAC	Digital-to-Analog Converter
DC	Direct Current
DCF77	Deutschland Long Wave Frankfurt 77.5 kHz
DCO	Digitally Controlled Oscillator
DMA	Direct Memory Access
DSSS	Direct Sequence Spread Spectrum
DVD	Digital Versatile Disc
EIR	Ethernet Interrupt Request
ERP	Equivalent Radiated Power
ETSI	European Telecommunications Standards Institute
FIFO	First In First Out
FM	Frequency Modulation
FPGA	Field Programmable Gate Array
FSK	Frequency Shift Keying
FSM	Finite State Machine
GFSK	Gaussian Frequency Shift Keying
GNU	GNU's Not Unix
GSM	Global System for Mobile Communications
HART	Highway Addressable Remote Transducer Protocol
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
ISA	International Society of Automation
ISM	Industrial, Scientist, and Medical
ITU-R	International Telecommunication Union (Radiocommunication)
IT+	Instant Transmission
LCD	Liquid Crystal Display
LMRS	Land Mobile Radio System
LR-WPAN	Low-Rate Wireless Personal Area Network
MAC	Medium Access Control
MCU	Microcontroller Unit
MSK	Minimum Shift Keying
NFC	Near Field Communication
OOK	On Off Keying
PC	Personal Computer
PD	Powered Device
PHY	Physical Layer

PoE	Power over Ethernet
PSE	Power Sourcing Equipment
PSTN	Public Switched Telephone Network
RAM	Random Access Memory
RF	Radio Frequency
RFID	Radio Frequency IDentification
RISC	Reduced Instruction Set Computing
RoHS	Restriction of Use of Hazardous Substances
RX	Receive
SDR	Software Defined Radio
SNMP	Simple Network Management Protocol
SoC	System on Chip
SPI	Serial Peripheral Interface
SRD	Short Range Device
TCP	Transmission Control Protocol
TFTP	Trivial File Transfer Protocol
TI	Texas Instruments
TX	Transmit
UART	Universal Asynchronous Receiver Transmitter
UDP	User Datagram Protocol
UHF	Ultra High Frequency
UPS	Uninterruptible Power Supply
USB	Universal Serial Bus
USCI	Universal Serial Communication Interface
USRP	Universal Software Radio Peripheral
UWB	Ultra-wideband
Wi-Fi	Wireless Fidelity (a branding effort for IEEE 802.11 WLANs)
WiMAX	Worldwide Interoperability for Microwave Access
WLAN	Wireless Local Area Network
WPAN	Wireless Personal Area Network
WSN	Wireless Sensor Network

1 Introduction

In this first chapter we will present our work exposing the environment where it will be located. We will also clarify what is the big problem we have to face to as well as the initial goals to achieve. Finally we will expose the organization of the chapters of this thesis.

1.1 General introduction to the area

In the last twenty years, networks have changed the way in which people and organizations exchange information and coordinate their activities. In the next several years we will witness another revolution; as new technology increasingly observes and controls the physical world. The latest technological advances have enabled the development of distributed processing, using tiny, low cost, and low-power processor that are able to process information and transmit it wirelessly. The availability of microsensors and wireless communications will enable the development of sensor networks for a wide range of applications, rather than the limited applications of sensor networks today.

Nowadays consumers want to know what is happening around them, especially in specific areas, for example, in their home or office. The number of commonly used electronic devices is increasing and these devices are increasingly connected to some network or communicating across point to point wireless links. Technology is making smart environments possible, i.e., the conditions and the state of ones surroundings is monitored and controlled by utilizing a smart device. It is precisely in such areas where wireless sensor networks are most meaningful.

A very common sensor in homes and offices is a temperature sensor. Today there are many devices that show not only the temperature, but also the time, date, humidity, etc. Many of these sensors use the Industrial, Scientific, and Medical (ISM) band to transmit information from the sensor to another device which frequently is equipped with a display. This display can be placed inside the building, while the sensor(s) might be located both inside and outside the building. A person can read the information on the display but generally there is no way to neither interact with the sensor nor that this data can be easily used by other applications.

1.2 Problem statement

Because some of these wireless sensors use a proprietary protocol to transmit their data we may need to decode this proprietary protocol in order to extract the relevant information before we can send this data to another computer, where an application could use this data (for example, temperature sensors could provide input for a home automation system).

We are starting with an existing wireless sensor, hence we must first reverse engineer the protocol being used by this proprietary sensor(s) and then with this information we will “recycling” (or repurpose) the data that already deployed sensors are transmitting.

1.3 Goals

The ultimate goal is to take advantage of the “closed” wireless point-to-point links by connecting them to the potentially global network, specifically a home or office internet. To achieve this is necessary to build a gateway that listens for data transmitted by sensors transmitting in the ISM band. This gateway will receive and if necessary

decrypt the information that the various wireless sensors are transmitting and then pass this information on to other computers for further processing. Because some sensors use a proprietary protocol to transmit their data we may need to decode this proprietary protocol in order to extract the relevant information before we can send this data to another computer, where an application could use this data (for example, temperature sensors could provide input for a home automation system). The many potential applications which could take advantage of such a gateway are outside the scope of this project – although for testing purposes we will create a sink application to receive this data and store time stamped records into a file. The overall context of this gateway is illustrated in Figure 1-1.

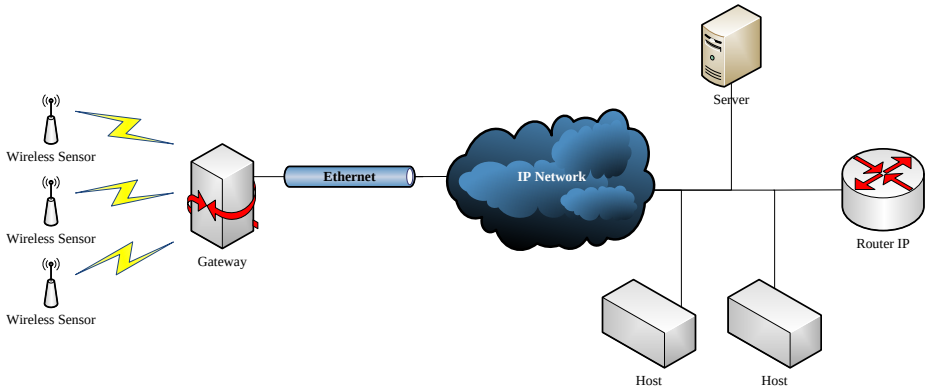


Figure 1-1: An overall view of how the sensor gateway might fit into a networked solution

This thesis project begins by considering only one specific temperature sensor which transmits in the 868 MHz ISM band, but the goal is to be able to identify all kinds of sensors, thus creating a global gateway (which operates within the frequency band(s) of the radio receiver in the gateway). This means that we have to consider how to recognize new sensors and how (and where) to decode the received messages. The gateway presented in this thesis is initially used to connect only a specific type of temperature sensor that could be used in a home or office automation application. The range of this sensor's wireless link is less than 100 meters (hence it is a Short Range Revice (SRD)).

We have divided this project in two functional parts:

1. Sniff and decode the information from one temperature sensor using a commercial radio receiver and appropriate software (The details of this sensor are given in section 3.1.1.). This temperature sensor contains a transmitter and comes together with a receiver built into a clock that displays the measured temperature. We assumed that the information transmitted by these sensors follows the typical link layer frame structure: header, data, and trailer. We will verify our decoding of the received temperature messages by comparing our results with the temperature shown on the receiver's display.
2. Design a gateway with a radio receiver, a microcontroller, and an Ethernet interface. The data will be presented in a format suitable for distributed to other services. Note that some of the processing may be done in the gateway and some in another computer connected to the network, but at some other physical location. The most appropriate partitioning of this processing will be examined in this part of the project. Since the gateway will be permanently connected to the network, we can consider other alternatives for powering such as Power over Ethernet (PoE).

1.4 Structure of this thesis

This thesis is divided into five chapters following a logical sequential order. The Chapter 1, “Introduction”, describes the area within which the problem is addressed and defines the goals to be achieved by this thesis. Chapter 2, “Background”, provides general overview of most of the protocols, concepts, and previous work related to or relevant to the subsequent chapters. Chapter 3, “Method (Implementation)”, examines the requirements of our application and offers a detailed specification for the operation of the gateway. We will describe the needed tools and the different parts of the gateway. Chapter 4, “Applying the Method (Operation)”, explains how we are going to evaluate all what we have done. The process of decoding the proprietary protocol is explained here as well as the gateway has been configured to perform its work. Chapter 5, “Analysis and Evaluation”, evaluates the appropriateness of its performance. We will do some experiments and tests to evaluate if the gateway executes its task as we expect. The last chapter (Chapter 6), “Conclusions”, analyses the results obtained in Chapter 5 and summarizes the conclusions reached as result of the work performed during this thesis project. Finally, some future work is recommended.

2 Background

This chapter will provide some background about both wired and wireless sensors network (see section 2.1). Following this, in section 2.2 we will introduce a number of wireless technologies that are relevant to this thesis project. A majority of the wireless sensors that we will be concerned with utilize one of the ISM bands, so we will review what the ISM bands are in section 2.3. Although any reader with basic knowledge about computer networks can successfully understand this report, we will explain Internet Protocol Suite and the most important protocols in the next sections (sections 2.4, 2.5, 2.6, 2.7, and 2.8) .The gateway will be connected to an Ethernet and the gateway needs some source of electrical power, hence to simply the installation of the gateway we will utilize power over Ethernet technology (see section 2.9). Finally, in section 2.10 we will describe some related work and what others have already done.

2.1 Wireless and Wired Sensor Networks

A Wireless Sensor Network (WSN) is a system with numerous spatially distributed devices. Each node of such a sensor network contains a sensor and we will use these sensors to monitor various conditions at different locations. The conditions that may be monitored include temperature, sound, vibration, pressure, motion, and pollutants. These sensor nodes are part of a network with many other nodes. There is at least one and typically more than one gateway sensor node that is used to connect the WSN to other networks or computers (as shown in Figure 2-1). These sensor nodes typically consist of a microcontroller, a power source (usually a battery), a radio transmitter/receiver (transceiver), and one or more sensors. An example of such a node is shown in Figure 2-2. The individual nodes are also called *motes* because they are tiny and light [1].

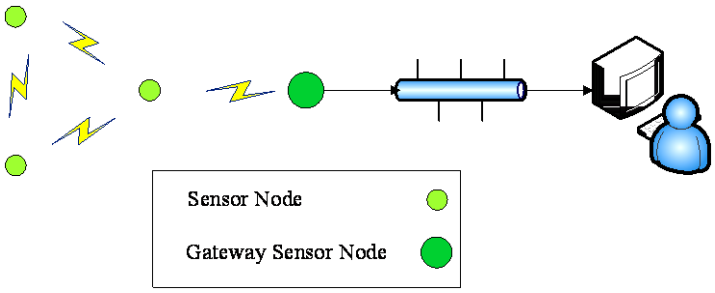


Figure 2-1: WSN structure

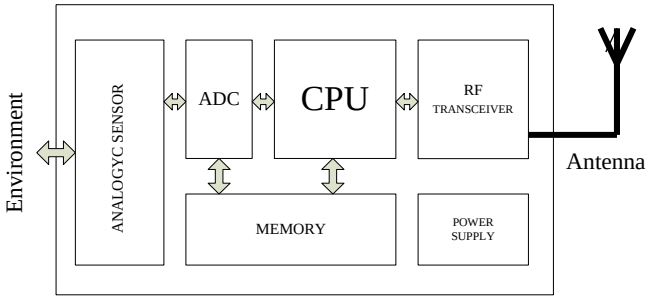


Figure 2-2: Structure of smart sensor

These sensors nodes are capable of processing a limited amount of data. However, when a large number of nodes are coordinated, they have the ability to measure a given physical environment in great detail. Thus, a sensor network application can be described as the coordinated use of a collection of motes that to perform a specific application. Sensor networks perform their tasks more accurately with a high density of node deployment and careful coordination. The coordinated use of sensor data enables **sensor fusion**. For example, using two cameras in a coordinated fashion to perform stereo imaging is based upon combining the data from the two two-dimensional image sensors to determine the three dimensional location of objects.

Sensor networks can consist of a small number of nodes connected by cable to a central data processing unit or they may be distributed WSNs. When the location of a physical phenomenon is unknown in advance, distributing the sensors means that some of these sensors will be close to the event. The data from multiple sensors can often be used to estimate the sensor value that would be measured if there were a sensor at the location of the event. Furthermore, in many cases sensors need to be distributed in order to avoid physical obstacles that would block of the communication links or when it is not possible to locate a single sensor at the desired measurement location.

Another requirement for sensor networks is distributed processing. This distributed processing is necessary because communication is the major consumer of energy, hence performing some local processing reduces the need for communication leading to lower total power consumption. Generally it is a good idea to process data locally as much possible in order to minimize the number of bits transmitted – especially when the sensor nodes are battery powered. However, if the sensor node has a ready supply of power, then it may be better to transmit all of the sensor data to a remote node for processing.

The deployment of sensor networks is increasing day by day. The number of applications that can be built using such sensor network is almost endless, limited only by our imagination and the data itself. Technology is making possible smart environments, where the conditions and the state of surroundings is monitored and controlled in an intelligent way. Example of applications are home and office automation, increased energy efficiency of building, remote patient monitoring, public safety monitoring, the monitoring of physical structures (such as buildings and bridges), the monitoring of equipment (such as motors and turbines), etc.

Today we are seeing lots of wired and wireless sensors networks being deployed. One of the important trends is that networks are no longer being deployed simply for a single application, but instead the sensor networks are being used by multiple applications (either in a serial time sharing fashion or in parallel). A key motivation for this thesis project is to exploit the data *already being generated* by wireless sensor nodes to enable new applications that the original designer of the sensor node had not thought of. The first step in this process is to receive and decode the data sent over the wireless link. The next section will describe some of the technology underlying these wireless links. The actual processes of receiving and decoding the transmissions from an example wireless sensor will be described in the next chapter.

2.2 Wireless technologies

In addition to the processor, battery, and sensor technologies, the development of WSNs also relies on wireless networking technologies. The Institute of Electrical and Electronics Engineers (IEEE) 802.11 standard [2] was the first standard for Wireless Local Area Networks (WLANs). This standard was first introduced in 1997. This

standard is based on a Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) Medium Access Control (MAC) protocol [3]. The initial version of the standard was subsequently revised leading to the IEEE 802.11b standard which allows increased data rates. Although the IEEE 802.11 family of standards were designed to create WLANs, i.e., to connect laptop computers and PDAs, the IEEE 802.11 family of interfaces has also been used in WSNs. However, the high power consumption and high data rates are unsuited to the needs of many WSNs.

Since each wireless application may have its own constraints (in terms of power consumption, desired data rate, number of nodes, communication range, reliability, security, etc.), several different types of wireless links have been developed in order to fulfill these requirements and expectations (see Table 2-1). The characteristics of these technologies affect the design of systems, devices, and applications that may be built. Figure 2-3 shows the differences between these solutions in terms of their maximum range versus peak data rate.

Table 2-1: Comparison of several wireless networking technologies

	ZigBee	Bluetooth	UWB	Wi-Fi	Proprietary
Standard	IEEE 802.15.4 [4]	IEEE 802.15.1 [5]	IEEE 802.15.3a	IEEE 802.11a,b,g,n	Proprietary
Industry organizations	ZigBee Alliance	Bluetooth SIG	UWB Forum and WiMedia Alliance	Wi-Fi Alliance	N/A
Topology	Mesh, star, tree	Star	Star	Star	P2P, star, mesh
RF frequency	868/915 MHz, 2.4 GHz	2.4 GHz	3.1 to 10.6 GHz (U.S.)	2.4 GHz, 5.8 GHz	433/868/900 MHz, 2/4 GHz
Data rate	250 kbits/s	723 kbits/s	110 Mbits/s to 1.6 Gbits/s	11 to 105 Mbits/s	10 to 250 kbits/s
Range	10 to 300 m	10 m	4 to 20 m	10 to 100 m	10 to 100 m
Power	Very low	Low	Low	High	Very low to low
Nodes	65,000	8	128	32	100 to 1000

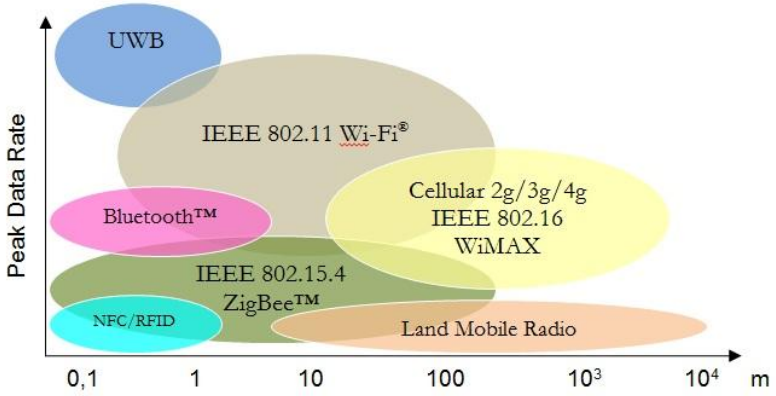


Figure 2-3: Wireless technologies compared according to data rate and range

The sensors that we are concerned within this thesis project operate at a short range (less than 100 meters) and do not require high bandwidth. These characteristics are typical of a Wireless Personal Area Network (WPAN). WPANs were conceived to integrate and interconnect nearby devices or "objects". Today WPANs are increasingly interconnected with other networks leading to the generalized concept of "The Internet of things" [6]. These "things" can be placed anywhere in a building, factory, the human body itself, etc. Because there is a need for flexible deployment wireless links are essential. Additionally, because there are multiple types of networks involved there is a need for one or more types of gateways to interconnect these networks.

In terms of low power WPANs, the ZigBee™ platform developed by the ZigBee Alliance [7] has become quite popular. ZigBee makes use of the IEEE 802.15.4 standard [4]. The IEEE 802.15.4 standard defines the MAC and physical layer of a WPAN. ZigBee defines a set of protocols (at layer 3 and higher) to manage communication among ZigBee nodes. ZigBee is sufficiently flexible that it can handle applications in numerous markets: industrial, health care, positioning, surveillance systems, etc. [8]. ZigBee is supported by several commercial sensor node products, including MICAz [9], TelosB [10], and IRIS [11].

Nevertheless, ZigBee is not the only option when it comes to WPANs. Currently, there are many other proprietary solutions[*]:

- SimpliciTI [12] is an open-source software network protocol developed by Texas Instruments. It is a low-power protocol aimed at simple and small RF networks. Texas Instruments provides some hardware that together with this software enables an easy implementation of a WPAN. Figure 2-4 shows a typical network structure using this technology. As we can see, the communication range can be extended through repeaters. Alarm controls, smoke detectors, and automatic meter reading are the main applications that currently utilize this protocol.

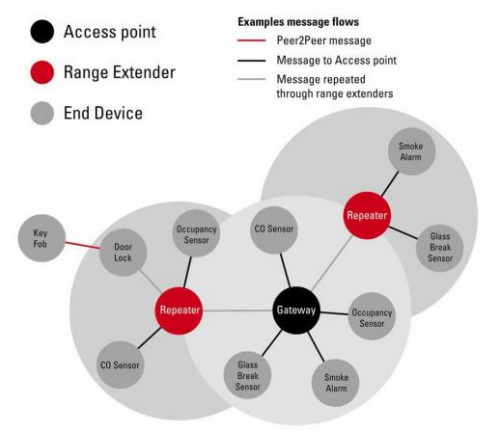


Figure 2-4: Typical network structure when using SimpliciTI [12]

- MiWi [13] was designed by Microchip Technologies. MiWi is based on IEEE 802.15.4. MiWi is designed for Low-Rate Wireless Personal Area Networks (LR-WPANs). The main advantage of MiWi is the ease in developing wireless applications and the ease in portability of these applications across different

[*] Note that ZigBee is a proprietary solution controlled and licensed by the ZigBee Alliance.

Microchip RF transceivers and different wireless protocols depending on the application's requirements, *without* having to change the application firmware. However, this protocol does not support large networks (as the maximum number of nodes is only 1,000 while ZigBee supports 65,000 nodes). While MiWi is an open specification, the microcontrollers and transceivers have to be from Microchip.

- Synkro [14] is a networking protocol software stack running on top of the IEEE 802.15.4 standard. Synkro is designed for use in home entertainment products, such as digital televisions, Digital Versatile Disc (DVD) players, audio/video receivers, etc. This solution is owned by Freescale, although it has been opened up since 2008. Freescale provides a software and hardware migration path for future product line extensions designed to revolutionize the way consumers control their home entertainment devices.
- PopNet™ [15] is a networking protocol and operating environment designed for use with IEEE 802.15.4 transceiver in low-power sensor and control applications. PopNet is owned by San Juan Software. It is a very flexible protocol, so it suits nearly any wireless application. This protocol utilizes Advanced Encryption Standard (AES) in 128-bit model to protect the transmitted data and to make communications more robust.
- Z-Wave [16] is a proprietary wireless communications protocol designed for home automation, specifically remote control applications in residential and light commercial environments. The technology uses a low-power RF radio embedded into home electronics devices and systems, such as lighting, home access control, entertainment systems, and household appliances. Z-Wave operates in the sub-gigahertz frequency range, typically around 900 MHz[†]. The Z-Wave Alliance is an international consortium of manufacturers that provide interoperable Z-Wave enabled devices.
- ONE-NET [17] is an open-source standard for wireless networking. ONE-NET was designed for low-cost, low-power (battery-operated) control networks for applications such as home automation, security & monitoring, device control, and sensor networks. ONE-NET is not tied to any proprietary hardware or software, and can be implemented with a variety of low-cost off-the-shelf radio transceivers and microcontrollers from a number of different manufacturers. It uses Ultra High Frequency (UHF) ISM radio transceivers and currently operates in the 868 MHz and 915 MHz bands. The ONE-NET standard allows for implementation in other frequency bands, and some work is being done to implement it in the 433 MHz and 2.4 GHz frequency ranges.
- ANT and ANT+ [18] are proprietary wireless sensor network technologies featuring a wireless communications protocol stack that enables radios operating in the 2.4 GHz (ISM band) to communicate by establishing standard rules for coexistence, data representation, signaling, authentication, and error detection.

[†] In Europe the operating frequency is 868.42 MHz.

The ANT and ANT+ protocols were designed and are currently marketed by Dynastream Innovations Inc.

- DASH7 [19] is an open source wireless sensor networking standard for wireless sensor networking, which operates in the 433 MHz unlicensed ISM band. DASH7 provides multi-year battery life, range of up to 2 km, low latency for connecting with moving things, a very small open source protocol stack, 128-bit AES encryption support, and data transfer at up to 200 kbit/s. DASH7 is the name of the technology promoted by the non-profit consortium called the DASH7 Alliance.
- WirelessHART [20] is a wireless sensor networking technology based on the Highway Addressable Remote Transducer Protocol (HART). The protocol supports operation in the 2.4 GHz ISM band using IEEE 802.15.4 standard radios.
- ISA100.11a [21] is an open wireless networking technology standard developed by the International Society of Automation (ISA). The official description is "Wireless Systems for Industrial Automation: Process Control and Related Applications".

In addition to wireless technologies there are several operating systems or sets of libraries for building applications that can run in wireless sensors nodes. Some examples of these are:

- TinyOS [22] is an open source operating system designed for low-power wireless devices such as WPANs. Written in nesC [23] (a dialect of C), it provides interfaces, modules, and specific configurations, which allow developers to build programs as a series of modules that do specific tasks.
- Contiki [24] is a small, open source, highly portable multitasking computer operating system developed for use on a number of memory-constrained networked systems ranging from 8-bit computers to embedded systems on microcontrollers, including sensor network motes. It is often called “The Operating System for the Internet of Things”. The name *Contiki* comes from Thor Heyerdahl's famous Kon-Tiki raft.

All of these technologies establish a framework, i.e., they define network topologies, ranges, sensors, compatibilities, and the rest of the information and documentation needed to build a complete WPAN from scratch. However, this thesis starts by going in exactly in the opposite direction in the meaning that we must first reverse engineer the protocol being used by this proprietary sensor(s).

The study of WSNs involves an enormous variety of disciplines. When a developer designs a WSN they have to think about the topology, routing, hierarchy, type of nodes, protocol, multiple access, etc. While designing a WSN is not our goal, since we are faced with an existing sensor node using an unknown communication protocol, we will need knowledge of many of these areas in order to exploit the existing transmission of wireless sensor nodes. In the rest of this chapter we will focus on the background areas related to our thesis project, specifically those technologies that could be useful as we carry out our work.

2.3 ISM band

The ISM bands are defined by the International Telecommunication Union Radiocommunication (ITU-R) [25] in their 5.138, 5.150, and 5.280 Radio Regulations (see Table 2-2). Individual countries' use of the bands designated in these sections may differ due to variations in national radio regulations. Because communication devices using the ISM bands must tolerate any interference from ISM equipment, unlicensed operations are typically permitted to use these bands hence unlicensed operation needs to be tolerant of interference from other devices. The ISM bands do have licensed operations; however, due to the high likelihood of harmful interference, licensed use of the bands is typically low or uses much higher power than unlicensed use.

In Europe the 900 MHz frequencies are part of the Global System for Mobile communications (GSM) allocation [26]. This implies that 900 MHz ISM equipment (illegally) imported from the U.S., Asia, or South Africa cause and suffers substantial interference. Instead, Europe uses the 868-870 MHz band (see section 2.3.1). Similarly, the use of the 433-435 MHz in the USA is replaced by the 340-354 MHz band.

A drawback of the ISM band is the lack of any protection against interference. To ensure some coexistence between new communication users and users already occupying the band, spread-spectrum transmission is mandatory, except for extremely low power applications. Spread-spectrum techniques offer some protection both for the licensed narrowband users of the bands (since the average spectral power density of the new users is low in their existing channels) and also to protect new users (since the processing gain of spread-spectrum systems mitigates interference from existing intentional and non-intentional radiators).

Table 2-2: ISM frequency band

Frequency range		Center frequency	Availability
6.765 MHz	6.795 MHz	6.780 MHz	Subject to local acceptance
13.553 MHz	13.567 MHz	13.560 MHz	
26.957 MHz	27.283 MHz	27.120 MHz	
40.660 MHz	40.700 MHz	40.680 MHz	
314.000 MHz	317.000 MHz	315.500 MHz	Japan
340.000 MHz	354.000 MHz	347.000 MHz	Region 2[‡] only and subject to local acceptance
433.050 MHz	434.790 MHz	433.920 MHz	Region 1[§] only and subject to local acceptance
868.000 MHz	870.000 MHz	869.000 MHz	Region 1
902.000 MHz	928.000 MHz	915.000 MHz	Region 2
2.400 GHz	2.500 GHz	2.450 GHz	
5.725 GHz	5.875 GHz	5.800 GHz	
24.000 GHz	24.250 GHz	24.125 GHz	
61.000 GHz	61.500 GHz	61.250 GHz	Subject to local acceptance
122.000 GHz	123.000 GHz	122.500 GHz	Subject to local acceptance
244.000 GHz	246.000 GHz	245.000 GHz	Subject to local acceptance

[‡] North and South America and Pacific (East of the International Date Line)
[§] Europe, Middle East, Africa, the former Soviet Union, including Siberia; and Mongolia and China

Some critics argue that technically it is a harder problem to protect a wanted signal from only a few interferers than to separate it from many weak interferers. Direct Sequence Spread Spectrum (DSSS) transmission typically spreads all signals over a wide bandwidth, so it also makes it likely that more users experience interfere than in narrowband scenarios – however, because the signal is spread over a wide bandwidth the power in any narrow frequency band is low.

2.3.1 Short Range Devices operating at 868 MHz

The term "Short Range Device" (SRD) covers radio transmitters which provide either unidirectional or bidirectional communications at low power, hence they are unlikely to cause interference to other radio equipment. SRDs use either integral, dedicated, or external antennas.

The 868 MHz radio spectrum was recommended and adopted by the European Conference of Postal and Telecommunications Administrations (CEPT) [27], especially by the Frequency Management, Regulatory Affairs and Spectrum Engineering Working Groups [28]. This recommendation sets out the general position on common spectrum allocation for "Short Range Devices" for countries within CEPT.

The European Telecommunications Standards Institute (ETSI) [29][30] developed standards for the majority of these devices and governs their use. The respective radio spectrum is specific to the European market and falls within the 868.000 - 870.000 MHz frequencies and is separated into four sections: G1-G4. The effective radiated power or Equivalent Radiated Power (ERP) in Watts is the radio frequency energy radiated by a device after taking into consideration all sources of losses and gains.

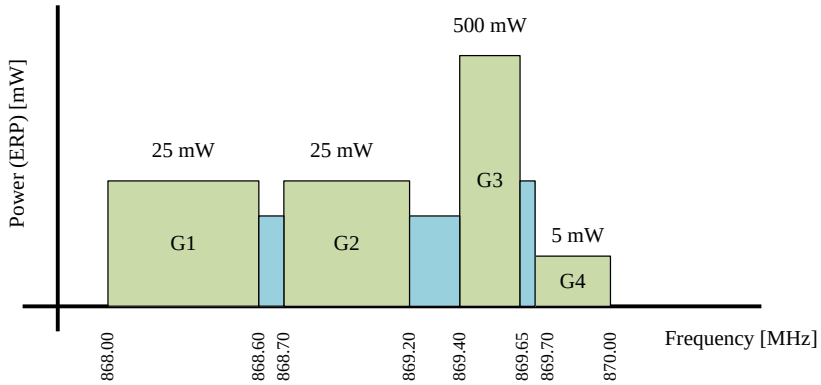


Figure 2-5: 868-870 MHz band. Blue bands are reserved for particular occupancies

2.4 The Internet Protocol Suite

The Internet Protocol Suite is a term used to describe the set of communication protocols, developed individually by the IT community, that implement the protocol stack on which the Internet runs. It is often called the TCP/IP protocol suite, which refers to the most important protocols in it: the Transmission Control Protocol (TCP) and the Internet Protocol (IP). These were also the first two protocols in the suite to be developed.

A protocol stack is a complete set of protocols layers that work together to provide networking capabilities. It is called a stack because it is typically designed as a hierarchy of layers, each supporting the one above it and using those below it. Each of these layers is designed to solve a specific issue affecting the transmission of data. Higher layers are closer to the user and deal with more abstract data, relying on lower layers to convert data into forms that can be physically manipulated for transmission.

According to RFC 1122 [31], the Internet Protocol Suite is divided into four abstraction layers, in contrast with the seven layers of the Open Systems Interconnect (OSI) reference model [32]. Table 2-3 illustrates the TCP/IP protocol stack.

Table 2-3: The TCP/IP stack

Network layer	Description
Application Layer	The set protocols involved in the process-to-process communication
Transport Layer	Responsible for dictating format of data sent, exactly where it is sent and maintaining data integrity
Internet Layer	Delivery data packets across network boundaries
Link Layer	Used to interconnect host or nodes in a network

Sometimes due to the usual mapping of the TCP/IP stack onto the OSI model, it is also common to refer to the Physical Layer as a hardware layer at the lowest part of the Link Layer.

2.5 Ethernet and IEEE 802.3

Ethernet is the predominant form of local area network (LAN) technology used with TCP/IP today. It uses an access method called Carrier Sense Multiple Access with Collision Detection (CSMA/CD). Moreover, it uses 48-bit addresses and originally operated at 10 Mbits/sec.

A few years later the IEEE 802 Committee published a standards, the 802.3 [33], which covers an entire set of CSMA/CD networks and defines the physical layer and data link layer's MAC of wired Ethernet. The MAC layer consists on the channel-access portion of the link layer used by Ethernet, but does not define a logical link control protocol. As for the physical layer, it supports several media, such as different types of coaxial cable, shielded and unshielded twisted pair, or Fiber-Optics. The supported transmission data rates range from 10 Mbits/s to 100 Gbits/s. Some media support half or full-duplex transmission.

The standard defines the mapping between IEEE 802.3 and Ethernet. In particular, Ethernet's data link-layer protocol can be encapsulated within the MAC Client Data field of IEEE 802.3 packets. Figure 2-6 illustrates this Ethernet data link-layer into IEEE 802.3 MAC field encapsulation.

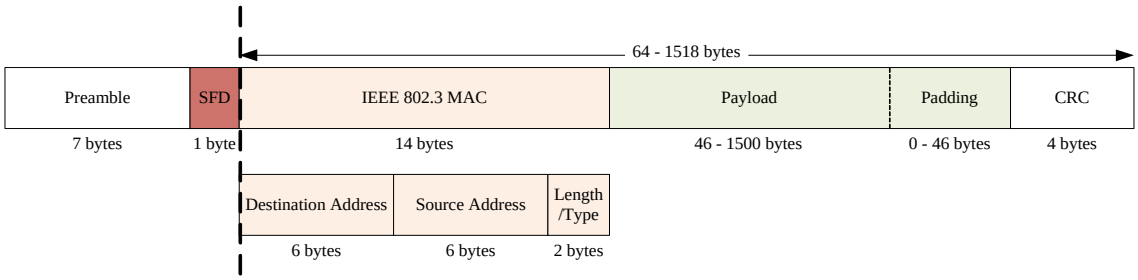


Figure 2-6: IEEE 802.3 and Ethernet encapsulation

- Preamble: Indicates that the frame is about to begin.
- Start Frame Delimiter: Indicates the end of the preamble and the start of the packet data. Always set to 0xAB.

- Destination Address: 48-bit IEEE 802.3 MAC address of the destination node. This may be a unicast, multicast or broadcast address.
- Source Address: The unicast 48-bit IEEE 802.3 MAC address of the source node.
- Length/Type: If the field value is less than 1500, it indicates the length of the payload. If its value is greater than 1500, then it indicates the type of the next higher protocol. Some of the most used values are 0x0800 for IP and 0x0806 for ARP.
- Payload: The data being transmitted.
- Padding: Required if the payload is less than 46 bytes.
- CRC: Cyclic Redundancy Check for integrity verification. This is also called the Frame Check Sequence (FCS).

2.6 Internet Protocol

The Internet Protocol (IP) is the workhorse protocol of the TCP/IP protocol suite. It provides an unreliable, connectionless datagram delivery service. There are no guarantees that an IP datagram successfully gets to its destination. However, IP provides a best effort service (through ICMP messages). Moreover, IP does not maintain any state information about successive datagrams. Each datagram is handled independently from all other datagrams. This also means that IP datagrams can get delivered out of order.

Figure 2-7: IP encapsulation

The IP was first defined on IEEE journal paper entitled "A Protocol for Packet Network Interconnection" [34]. The protocol was later revised and updated to its fourth version (IPv4) and its sixth version (IPv6).

2.6.1 IPv4

Internet Protocol version 4 (IPv4) is defined in RFC 791 [35] and it is the most used version of IP. It uses 32-bit addresses which restricts the total number of addresses to 2^{32}. Its header is illustrated in Figure 2-8. The normal size is 20 bytes, unless options are present.

The most significant bit is numbered 0 at the left, and the least significant bit of a 32-bit value is numbered 31 on the right. The 4 bytes in the 32-bit value are transmitted in the order: bits 0-7 first, then bits 8-15, then 16-23, and bits 24-31 last. This is called *big endian* byte ordering, which is the byte ordering required for all binary integers in the TCP/IP headers as they traverse a network.

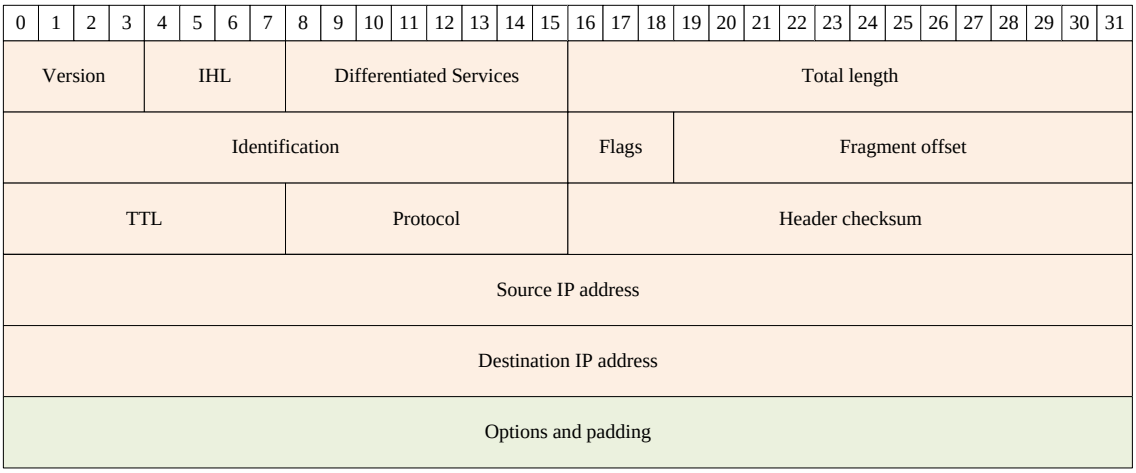


Figure 2-8: IPv4 header

- Version: Specifies the version of the IP. It has a value of 4 for IPv4.
- IHL: Internet Header Length in 32-bit words. The minimum value is 5 (20 bytes header, with no options).
- Differentiated Services: RFC 2474 [36] defines this field.
- Total length: Contains the length of the datagram including the header.
- Identification: Used to identify the fragments of one datagram from those of another.
- Flags: Used for fragmentation control to indicate whether a fragment is the last fragment or not of a datagram or if fragmentation is allowed for a datagram.
- Fragment offset: Used to direct the reassembly of a fragmented datagram.
- TTL: Time to Live. A timer field used to track the lifetime of the datagram. When the TTL field is decremented down to zero, the datagram is discarded.
- Protocol: Specifies the next encapsulated protocol.
- Header checksum: A 16-bit one's complement checksum of the IP header and IP options.
- Source IP address: 32-bit IP address of the sender.
- Destination IP address: 32-bit IP address of the intended receiver.
- Options: Optional field with variable length.
- Padding: Needed to ensure that the header contains an integral number of 32-bit words.

2.7 User Datagram Protocol

The User Datagram Protocol (UDP) is a simple, datagram-orientated, transport layer protocol defined in RFC 768 [37]. It offers a minimal transport service, with no guaranteed datagram delivery. It gives applications direct access to the datagram service of the IP.

Figure 2-9 shows the encapsulation of a UDP datagram as an IP datagram.

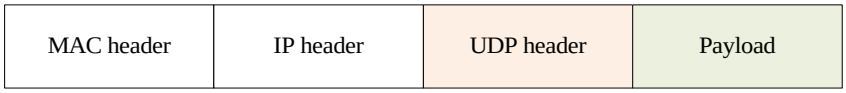


Figure 2-9: UDP encapsulation

UDP provides no reliability: it sends the datagrams that the application writes to the IP layer, but there is no guarantee that they ever reach their destination. Sometimes it is considered almost a null protocol; the only services it provides over IP are *checksumming* of data and multiplexing by port number.

Figure 2-10 illustrates the fields in the UDP header.

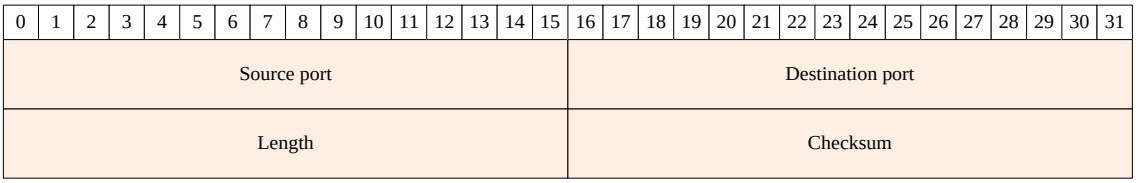


Figure 2-10: UDP header

- Source port: The port number of the sender. Cleared to zero if not used.
- Destination port: The port this packet is addressed to.
- Length: The length in bytes of the UDP header and the UDP data. The minimum value is 8.

The next field is a checksum computed as the 16-bit one's complement of the one's complement sum of a pseudo-header of information from the IP header, the UDP header, and the data, padded as needed with zero bytes at the end to make a multiple of two bytes [38]. Figure 2-11 shows this pseudo-header.

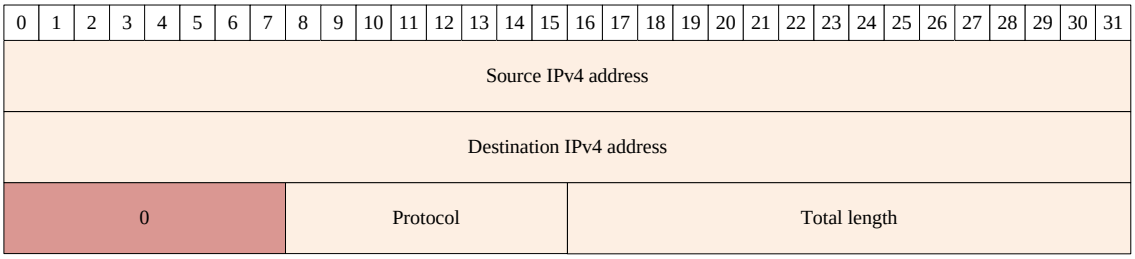


Figure 2-11: Pseudo-header for UDP checksum computation (IPv4)

The *checksum* field in UDP is not mandatory for IPv4. If this field is cleared to zero, then *checksumming* is disabled. If the computed checksum is zero, this field must be set to 0xFFFF. However, it is mandatory when transported by IPv6.

2.8 Other protocols

In this section we introduce other important protocols of the TCP/IP protocol suite that we have worked with along our thesis.

2.8.1 Address Resolution Protocol

The Address Resolution Protocol (ARP) provides a dynamic mapping between network layer addresses and link-layer addresses. In our case, it is used for resolution of IPv4 addresses into IEEE 802.3 MAC addresses. Its description is in the RFC 826 [39].

Figure 2-12: ARP encapsulation

Essential to the efficient operation of ARP is the maintenance of an ARP table on each host. This table maintains the recent mappings from Internet addresses to hardware addresses and it is updated frequently.

Figure 2-13 shows the format of an ARP packet.

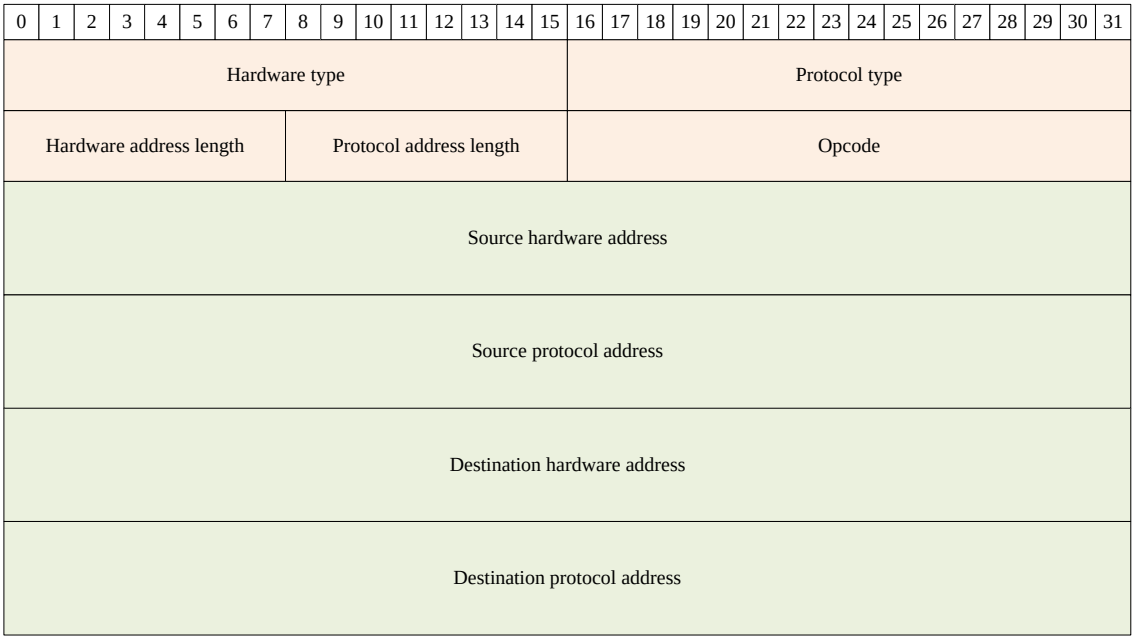


Figure 2-13: ARP header

- Hardware type: Specifies the type of hardware address. Its value is 1 for Ethernet.
- Protocol type: Specifies the type of protocol address being mapped. Its value is 0x0800 for IP addresses.
- Hardware address length: Length in bytes of the hardware address. Its value is 6 for MAC addresses.
- Protocol address length: Length of the protocol address in bytes. Its value is 4 for IP addresses.
- Opcode: Specifies the operation to realize. Its value is 1 for ARP request and 2 for ARP reply.

The next four fields that follow are the sender's hardware address, the sender's protocol address, the target hardware address, and the target protocol address. Note that in our case we refer the hardware address as the MAC address (48-bit address) and the protocol address as the IPv4 address (32-bit address).

Below we expose the mechanism used by the ARP.

1. Node *A* (with IP address *x*) wants to send a packet to node *B* (with IP address *y*.
2. A look for *B*'s MAC address in its ARP table.

3. If found, the packet can be sent directly (go to step 8). If not, then *A* sends an ARP request packet to the MAC broadcast address.
4. *B* receives the ARP request and identifies its IP address (*y*) in the packet's target address field.
5. *B* stores the pair <*A*, *x*> in its ARP table and responds to *A* with an ARP reply packet including its MAC address.
6. *A* receives the ARP reply from *B* and stores the pair <*B*, *y*> in its ARP table.
7. Receiver and sender know each other's MAC address. *A* goes back to step 1.
8. *A* sends it packet to *B*.

Address Resolution Protocol is not used in IPv6. Instead of, it is used the Neighbor Discovery protocol.

2.8.2 Internet Control Message Protocol

The Internet Control Message Protocol (ICMP) is defined in the RFC 792 [40]. It is often considered part of the IP layer. It communicates error messages and other conditions that require attention over IP datagrams.

Figure 2-14: ICMP encapsulated within an IP datagram

The purpose of these control messages is to provide feedback about problems in the communication environment, not to make IP reliable. The higher levels protocols that use IP must implement their own reliability procedures if reliable communication is required. Figure 2-15 shows the header of an ICMP message.

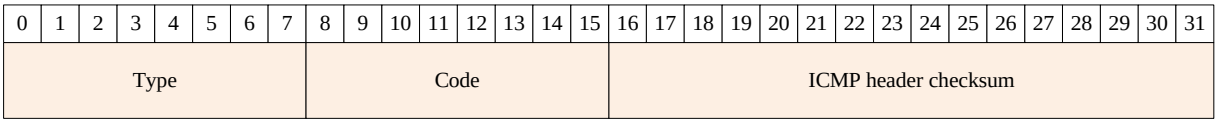


Figure 2-15: ICMP header

- Type: Specifies the format of the ICMP message. The reader can get a large list of ICMP messages in the RFC 792 [40].
- Code: Further qualifies the ICMP message.
- ICMP header checksum: Checksum that covers the entire ICMP message. The algorithm used is the same as we described for the IP header checksum in section 2.6.1.

The *Payload* field contains the data specific to the message type indicated by the *Type* and *Code* fields.

2.8.3 Dynamic Host Configuration Protocol

The Dynamic Host Configuration Protocol (DHCP) is a network configuration protocol for hosts on IPv4 networks. Originally it was first defined in RFC 951 [41] as part of the Bootstrap Protocol (BOOTP) but later was redefined as protocol itself in RFC 2131 [42]. Hosts that are connected to TCP/IP networks must be configured before they can communicate with other hosts. DHCP eliminates the manual task by a network

administrator. There has to be almost one server in the TCP/IP network to be able to realize DHCP tasks.

DHCP uses UDP as its transport layer protocol. The port 67 is reserved for sending data to the server, whereas the port 68 is reserved for data to the client. Due to the features of UDP, DCHP communications are connectionless in nature.

Figure 2-16: DHCP encapsulation

Its operation involves interchanging four packets between the host being configured and a DHCP server: DHCP Discovery, DHCP Offer, DHCP Request, and DHCP Acknowledgement.

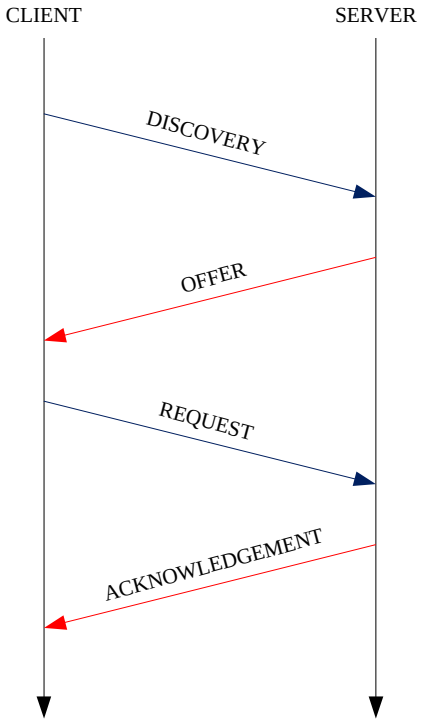


Figure 2-17: DHCP algorithm

After this procedure, the host may acquire a variable set of network parameters like its IPv4 address, subnet mask, and router's IPv4 address. This address assignment is for a limited lease time. When half of the time has expired, the host initiates the DHCP renewal process by sending a new DHCP Request to renew its lease. Figure 2-18 illustrates the DHCP header.

- Opcode: Indicates if it is a *Boot Request* (1) or *Boot Reply* (2).
- Hardware type: The same as for ARP header. For Ethernet, its value is 1.
- Hardware address length: The same as for ARP header. Its value is 6 for Ethernet addresses.
- Hop count: Used by relay agents.

- Transaction ID: It contains a random number chosen by the client, used by the client and server to associate messages and responses between a client and a server.
- Number of seconds: The elapsed time in seconds since the client began an address acquisition or renewal process.
- Flags: Defined in RFC 1542 [43].

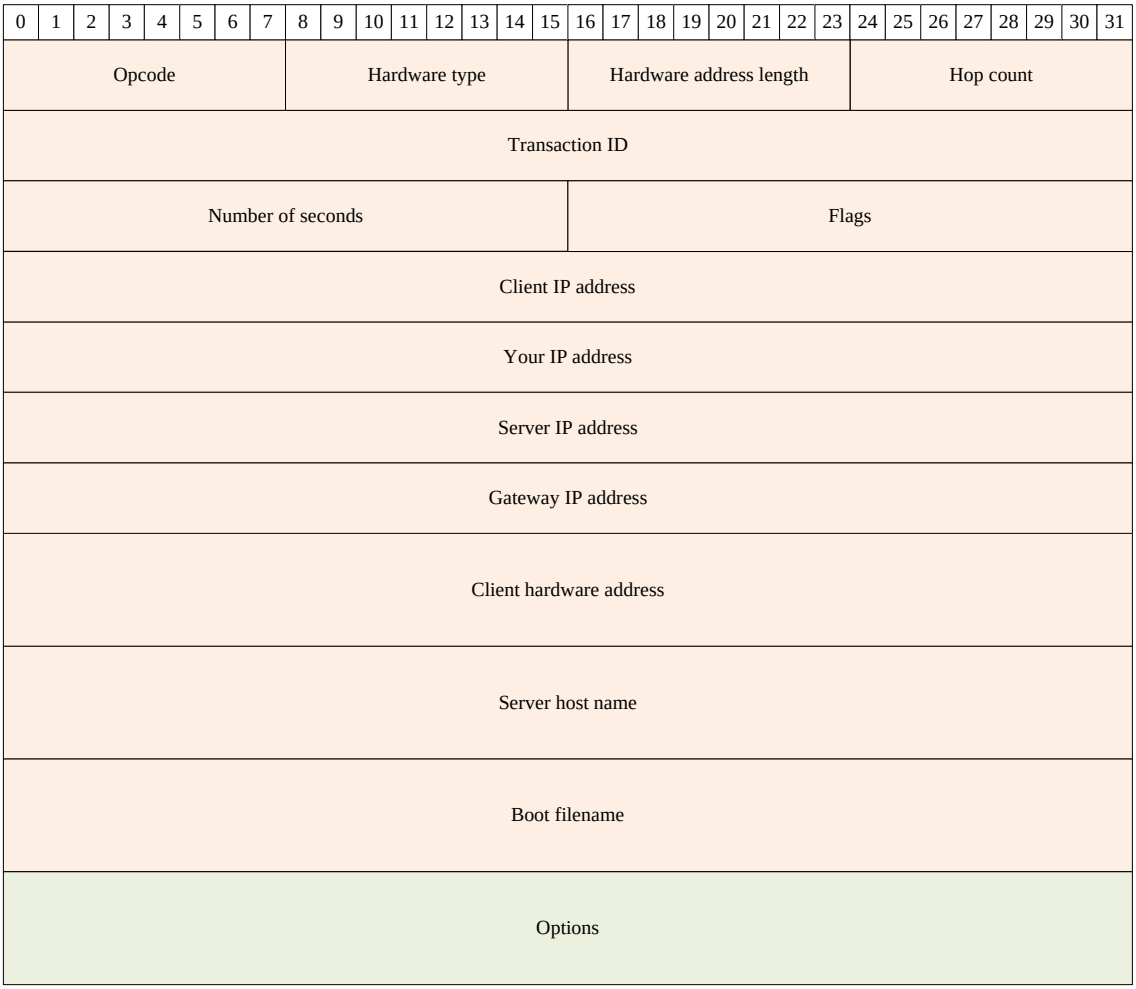


Figure 2-18: DHCP header

The next four fields are 32-bit addresses and indicate the IP address that will be assigned to the client, the temporary IP address of the host, the IP address of the server as well as the IP address of the gateway. The follow field is the *Client hardware address* expressed in 16 bytes. The *Server host name* and the *Boot filename* fields normally are filled with zeros. Finally, the *Options* field has a variable length depending on the DHCP message that is being transmitted.

2.9 Power over Ethernet

Power over Ethernet (PoE) is a technology that integrates power into a standard LAN infrastructure. PoE enables power to be provided to a network device, such as an IP phone or a network camera, using the same cable as that used for network

connectivity. Power is supplied in common mode over two or more of the differential pairs found in Ethernet cables.

PoE technology was standardized in 2003 by the IEEE into a standard called IEEE 802.3af [44]. The mechanism for delivering power via PoE is similar to the way in which the Public Switched Telephone Network (PSTN) [45] provides power to telephone handsets. PoE delivers Direct Current (DC) power of up to a theoretical maximum of about 15 W per cable. However, in practice a maximum power of 12.95 W is available, due to losses in the cable. In 2009, IEEE 802.3at updated the PoE standard (also known as PoE+ or PoE plus) to provide up to 25.5 W of power.

The PoE requires a standard network cable of category 5 (CAT5) or higher for high power levels, but can operate with category 3 (CAT3) cable for low power levels. A category 5 cable has four twisted pairs, but only two of these pairs are used for data. The IEEE 802.3af specification allows either of the spare pairs to be used (i.e., pins 4 and 5, or 7 and 8) or the data pairs (pins 1 and 2, or 3 and 6) to carry the power.

Typically a PoE system consists of both Power Sourcing Equipment (PSE) and a Powered Device (PD). The PSE may either be an Endspan (i.e. PoE Switch) or a Midspan (i.e. PoE hub). The PD is a PoE-enabled terminal such as an IP phone. PoE systems are deployed in a "star topology", so each PD is connected to a separate channel of the central PSE. The configuration when using a switch is shown in Figure 2-19.

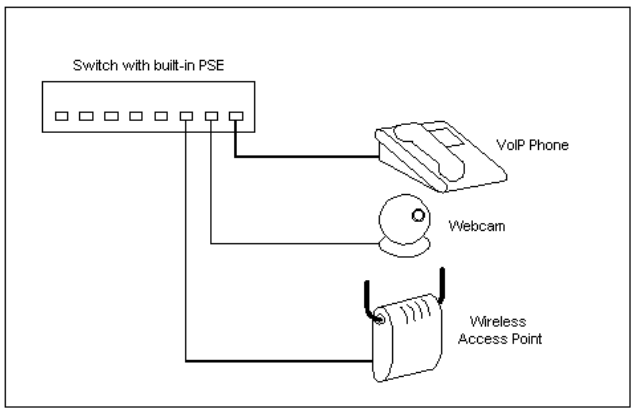


Figure 2-19: Power over Ethernet connection

PoE technology supports a point-to-multipoint power distribution architecture, parallel to the data network. This star topology enables use of a single Uninterruptible Power Supply (UPS) at the network's core to power multiple devices attached to the LAN. The LAN also provides remote access and management via Simple Network Message Protocol (SNMP) [46]. SNMP is used in many cases to monitor and control the UPS device in addition to the LAN switch.

The IEEE 802.3af specification defines a process for safely powering a PD over a cable, and then removing power if a PD is disconnected. The process proceeds through three operational states: detection, classification and operation. The intent behind the process is to leave a non-terminated cable unpowered while the PSE periodically checks for a plugged-in device; this is referred to as detection. The low power levels used during detection are unlikely to cause damage to devices not designed for PoE. If a valid PD signature is present, then the PSE may optionally inquire how much power the PD requires; this is referred to as classification (see Table 2-4). The PD may return a default full-power signature, or one of four other choices. Knowing the power demand of each

PD allows the PSE to intelligently allocate power between PDs, and also to protect itself against overload. The PSE powers up a valid PD, and then monitors its output for overloads. The maintain power signature (MPS) is presented by the powered PD to assure the PSE that it is there. The PSE monitors its output for the MPS to see if the PD is removed, and turns the port off, if it loses the MPS. Loss of MPS returns the PSE to the initial state of detection.

Table 2-4: Power over Ethernet: classification

CLASS	PD POWER(W)	NOTE
0	0.44 – 12.95	Default class
1	0.44 – 3.84	
2	3.84 – 6.49	
3	6.49 – 12.95	
4	-	Reserved for future use

2.9.1 Advantages of PoE

Sometimes, devices requires more power than Universal Serial Bus (USB) offers and very often the device must be powered over longer runs of cable than USB permits. Field-spliced outdoor category 5 Ethernet cable can power radios and other low-power devices, for instance, through over 100 m of cable, an order of magnitude further than USB's theoretical maximum.

In addition, PoE uses only one type of connector, a RJ45, whereas there are numerous types of USB connectors and each new USB standard has added additional types of connectors. PoE is presently deployed in applications where USB is unsuitable and where Alternating Current (AC) power would be inconvenient, expensive, or infeasible to supply.

However, even where USB or AC power could be used, PoE has several benefits over either, including:

- Cheaper cabling — even high quality outdoor category 5 cable is much cheaper than USB repeaters or mains voltage AC wiring (note that the major cost of AC mains wiring is not the wire, but rather the requirement in many locations to have this wiring installed by an electrician).
- Global organizations can deploy PoE everywhere without concern for any local variance in AC power standards, outlets, plugs, or reliability.
- While USB devices require a host computer or router to control the bus, and still require switches or routes for Internet connections, a powered Ethernet device requires only a switch, which can be unmanaged and can provide both power and network connectivity.
- Symmetric distribution is possible. Unlike USB and AC outlets, power can be supplied at either end of the cable or outlet. This means the location of the power source can be determined *after* cables and outlets are installed.

The arrival of PoE changes the network design possibilities. This technology is especially useful for powering IP telephones, networks routers, WLAN access points,

remote cameras, Ethernet switches, industrial devices, embedded computers, and Liquid Crystal Displays (LCDs). In the scope of this project we will use it to power our sniffing gateway.

2.10 Related work

A variety of previous research is related to our master thesis project. In this section we will summarize some of the most relevant work.

There are many papers that describe the decryption of unknown wired and wireless link protocols. Some of these papers focus exclusively on Internet protocols [47][48], although the studies reported in [49] and [53] focused on wireless sensor networks. For example, Rubén Martín Sánchez in [51] discusses sniffing frames in a ZigBee sensor network.

Yu Zhuanghui and others [52] define a system, called *Catcher*, which is capable of extracting information from unknown protocols. The process has 3 steps. First, simple packets are captured, and the program lists them. Then, the fields of the frames are located and the length of each field is calculated. In the final step, *Catcher* identifies static and dynamic fields, in order to extract data that changes for each frame, i.e., the higher layer information or the message itself. This article suggested the procedure that we followed in our work.

The research described in [53] and [54] uses the same RF transceiver and microcontroller unit (MCU) that we will use, although their aims are quite different than ours. These two articles give us information about the communication between MSP430 and CC1101.

The topic of paper [55] is not related to our thesis, however it gives us useful information because the author uses the same Ethernet controller that we plan to use, but applies it to a networked three-phase electric power meter. In particular this article shows the interface between a Texas Instruments microprocessor, the Ethernet controller, and a RJ45 jack. In addition, it explains the procedures necessary to initialize the ENC28J60 chip (initialization of receiving and sending buffers and the initialization of MAC and Physical Layer (PHY) modules), the steps in receiving and transmitting data, and the flow chart of their network software (see Figure 2-20). According to this figure, when the main procedure processes an interrupt from the Ethernet controller, the following steps are determined by reading the Ethernet Interrupt Request register (EIR). This flow chart might be interesting for us when developing our gateway´s software.

Zhou and Shen [56] describe a realization of a ZigBee – Wireless Fidelity (Wi-Fi) wireless gateway. The hardware platform of the gateway is made up of two parts: a 2.4 GHz radio frequency (RF) chip (that comprises incorporates an IEEE 802.15.4 transceiver and a microprocessor) and a Wi-Fi module. The communications between this chip and the Wi-Fi module is achieved through a Universal Asynchronous Receiver Transmitter (UART) interface. The software architecture includes system control, software design of the RF chip, software design of Wi-Fi module, and a wireless gateway application layer protocol. The paper explains the interaction procedures between the ZigBee network and the WLAN (and in the reverse direction) using this gateway. This design has many similarities with ours. Although we do not use Wi-Fi and we only implement the communication from the sensor to the LAN, we can utilize this paper as a good pattern for our work.

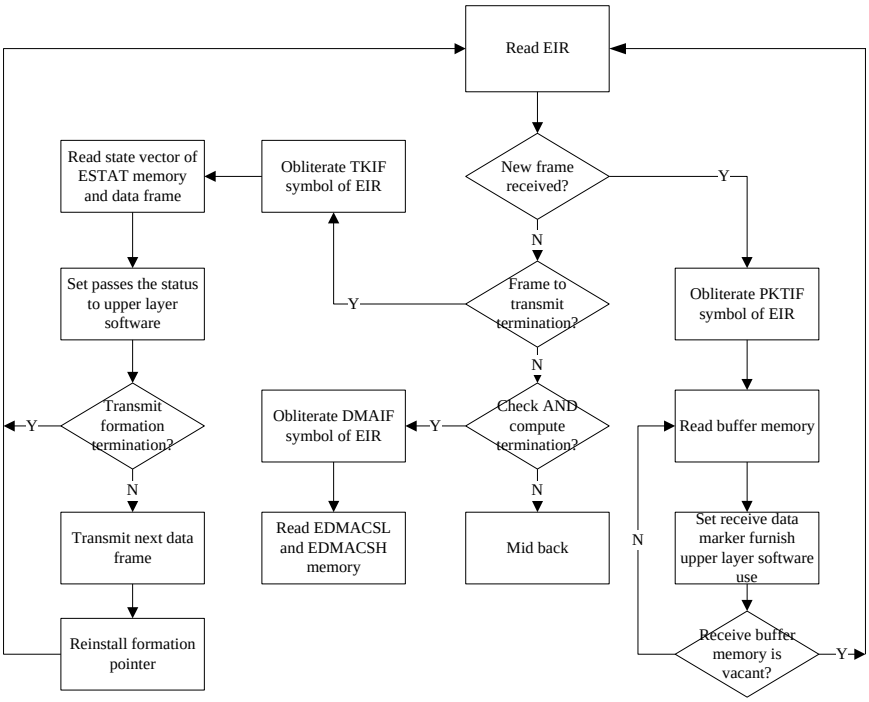


Figure 2-20: ENC28J60 RX and TX flowchart

There is a really interesting discussion in the forum of JeeLabs [57]. The topic is the use of the radio module RFM12B [58] to sniff data from weather station transmitters distributed by La Crosse Technology [59]. As well as our sensor under study, they are catalogued as Instant Transmission (IT+) transmitters. This indicates that they could work similarly. Furthermore, some users have found out how the messages are encoded so we can benefit of this information to our project.

Finally, there are also more general works that address the above (a frame analyzer and design of a specific receiver) [60]. However, in the case of these papers the communication protocol is well known.

3 Method (Implementation)

In this chapter we will explain how to achieve our proposed goal. Basically, we need to do two things: design the gateway and decode the communication protocol of the wireless sensor network. The sort of doing these tasks is irrelevant. It is possible to design firstly the gateway and after that study the wireless sensor network. However, we will follow the opposite sort. We will first study the wireless sensor network in order to figure out how the communication protocol is implemented (developed in section 4.1). The hardware and the tools used to achieve this goal will be described in section 3.1. After that in section 3.2 we will describe our gateway clarifying briefly all of its integrated parts.

3.1 Hardware and Software tools

In order to carry out this thesis project, some hardware and software are required. Professor Maguire provided all the necessary hardware. The first device was the target wireless sensor (see section 3.1.1). To determine the sensor's transmission frequency we used a spectrum analyzer (see section 3.1.2). The next piece of equipment was a software radio to enable us to receive and decode the sensor's transmission (see section 3.1.3). In order to use this software radio we made use of the GNU Radio software (see section 3.1.4).

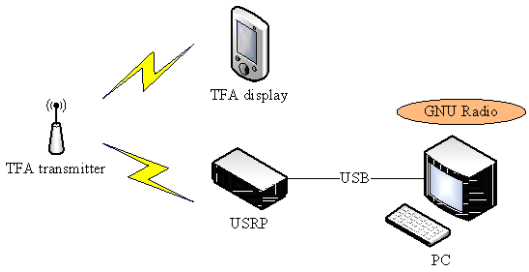


Figure 3-1: Initial hardware configuration to decode the temperature messages distributed by a TFA temperature sensor.

3.1.1 TFA radio-controlled clock and wireless temperature transmitter

The TFA Dostmann GmbH & Co. KG (TFA) 'Wave' wireless thermometer with radio-controlled clock (catalog part number 30.3016.54.IT) is a wireless thermometer solution [61] that operates in the 868 MHz band. The radio controlled clock (shown in Figure 3-2 on the left) consists of a 868 MHz receiver, a 77.5 kHz radio receiver to receive timing information from the DCF77 (D=Deutschland, C=long wave signal, F=Frankfurt, 77=frequency: 77.5 kHz) transmitter in Germany [62], a microprocessor, battery, internal temperature sensor, and a display that shows the date, time, and outside and inside temperature. The outside temperature is provided by an associated temperature-transmitter 868 MHz/IT module with no human-machine interface (catalog part number 30.3164.IT) (shown in Figure 3-2 on the right). This temperature transmitter module will be the initial sensor that we will consider.

Figure 3-2: TFA radio-controlled clock (left) with a wireless temperature transmitter (right)

The gateway will sniff the traffic sent by the wireless temperature transmitter in order to detect frames and decode the contents of these frames. Within these frames we must locate where the data field is and decode it. The visual interface on the receiver will be used in our analysis as it displays the temperature reported by the sensor. As alternative we could study the signal present on the chips of the transmitter but these chips cannot be identified, as the picture in Figure 3-3 shows. There are two chips on this board underneath large drops of epoxy. Decoding the messages by doing that seems not to be feasible.

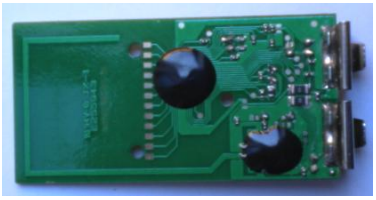


Figure 3-3: Circuit board from within the temperature transmitter

3.1.2 Spectrum Analyzer

We need a spectrum analyzer to determine the frequency of the signal sent by the transmitter. Specifically, we will a HP Agilent model 8595E Portable spectrum analyzer [63]. Although today this model is obsolete, this spectrum analyzer was located in the same laboratory where we are carrying out our thesis project.

This spectrum analyzer is a very sensitive receiver that can work with signals whose frequency is between 9 kHz and 6.5 GHz. It works on the principle of a "super-heterodyne receiver" to convert higher frequencies to measurable quantities. The received frequency spectrum is slowly swept through a range of pre-selected frequencies, converting the selected frequency to a measurable DC level (usually logarithmic scale), and displaying this quantity on the Cathode Ray Tube (CRT) screen. The CRT displays received signal strength (y-axis) as a function of frequency (x-axis).

3.1.3 The Universal Software Radio Peripheral

For development purposes an Ettus Research LLC Universal Software Radio Peripheral (USRP) [64] will be used to receive the data from the TFA temperature transmitter. This function will later be performed by the sniffer gateway that we will design. The particular model of the USRP that we will use is the USRP1 [65]. The USRP1 is a very flexible device that allows a developer to quickly design and implement flexible radio systems. It requires an external 6 volt DC power supply. The USRP can be connected to a computer via a USB port.

The motherboard, shown in Figure 3-4, basically consists of:

- An Altera Cyclone Field Programmable Gate Array (FPGA) to perform high sample rate processing
- 4 high-speed Analog-digital Converters (ADC) connected to the FPGA
- 4 high-speed Digital-Analog Converters (DAC) connected to the FPGA
- A USB 2.0 interface chip

On the motherboard there are 4 slots to plug in daughterboards: 2 receivers (RXA and RXB) and 2 transmitters (TXA and TXB). Each daughterboard has access to 2 of the 4 high-speed converters, specifically DACs for transmitters and ADCs for receivers.

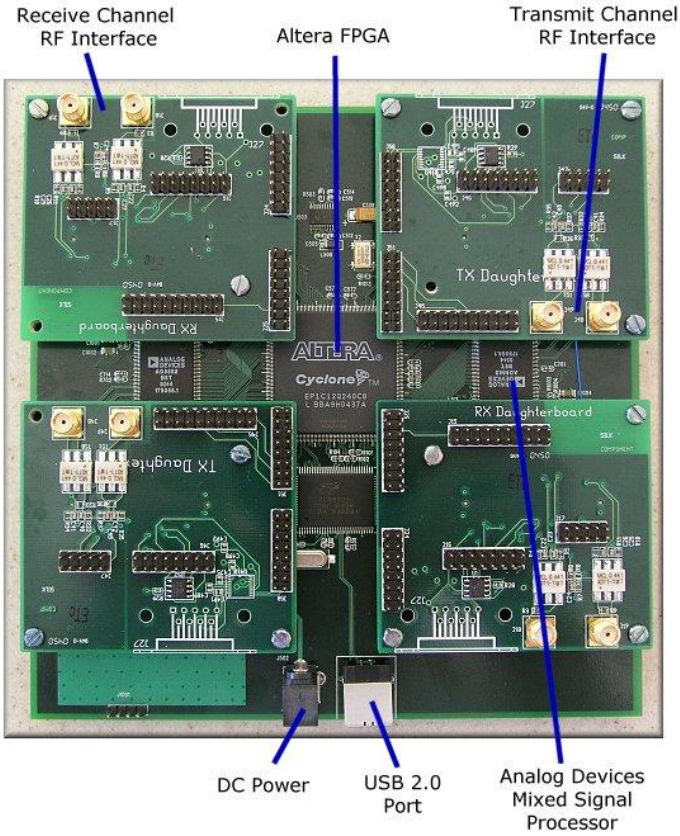


Figure 3-4: USRP Motherboard

There are several different kinds of daughterboards [66], but in this thesis we have used the "DBSRX" daughterboard. The DBSRX is a complete receiver system for 800 MHz to 2.4 GHz, and the noise figure is between 3 and 5. As the initial sensor utilizes a frequency of 868 MHz, this daughterboard was appropriate.

3.1.4 GNU Radio

GNU Radio [67] is a free open-source software development toolkit that provides signal processing blocks to implement software radios. It can be used together external RF hardware to create software-defined radios, or without hardware, as a simulation environment.

A Software Defined Radio (SDR) is a radio system which performs the required signal processing in software instead of using dedicated integrated circuits to do the processing in hardware. The advantage of SDR is that you can create different kinds of radio using the same hardware simply by modifying the software.

GNU radio applications are written mainly in Python [68]. Python is a high-level multi-paradigm programming language (which is free and open-source) first implemented in late 1980s. It uses indentation to delimit blocks, making Python a very readable programming language.

The GNU Radio software includes a number of different tools to facilitate the work of a developer. One of them is GNU Radio Companion [69], a graphical tool for creating signal flow graphs and generating flow-graph source code. There are many pre-existing blocks that you can use when developing an application. Using the GNU Radio Companion development of an application is very intuitive and it does not require much knowledge to create efficient signal flow graphs.

Combining GNU Radio with the USRP (described in section 3.1.3) a powerful SDR is created. For example, with the suitable hardware, you can receive TV signals and view TV channels, or you can listen to various Frequency Modulation (FM) radio channels, receive satellite signals, etc. With a suitable RF front end almost any kind of radio transceiver that you can think of can be created with this combination of hardware and software. The USRP hardware provides the input and output to a signal flow graph and the GNU Radio tools can be used to augment the default functions of the USRP.

3.1.5 Code Composer Studio

Code Composer Studio (CCS) is an integrated development environment for Texas Instruments digital signal processors, microcontrollers, and application processors. It includes a suite of tools used to develop and debug embedded applications. It also includes compilers for each of Texas Instruments (TI) device families, a source code editor, a project build environment, a debugger, a profiler, simulators, and many other features.

There are different versions of CCS in use today. In this thesis project, we are going to work with version v5 of this tool. This tool can be downloaded for free from TI's webpage [70]. Information, examples, guides, and support can be also found from this webpage.

We will use CCS to program the MCU. Using CCS, we will develop the application that the microprocessor has to run to control the receiver and the Ethernet controller, and to send the decoded data that has been receiver to a remote computer. All of our software will be written using the C language. We will also leverage the earlier work done with this processor and Ethernet controller by Luis Maqueda Ara [71].

3.1.6 Easily Applicable Graphical Layout Editor

The Easily Applicable Graphical Layout Editor (EAGLE) is a user-friendly, powerful and affordable software for efficient printed circuit board (PCB) design and combines the modules schematic editor, layout editor and *autorouter* on one single interface. It is very easy to use and includes a set of libraries with the schematics and the footprints of a lot of commercial devices. A freeware version of EAGLE is available on the CadSoft webpage [72].

With this tool we will design the PCB of our gateway and let it ready to be manufactured.

3.2 Gateway specifications

The sniffer gateway that we will design, make, and evaluate consists of a microcontroller, a transceiver or a receiver, and an Ethernet controller. Figure 3-5 illustrates an overall view of the gateway.

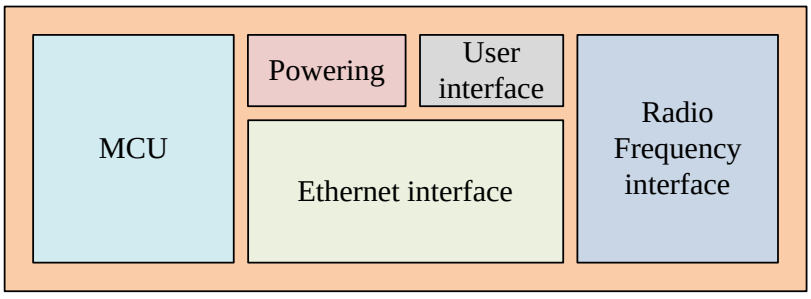


Figure 3-5: General overview of the gateway

3.2.1 MSP430 Microcontroller

The MCU is the core of our gateway. We chose one from the Texas Instruments MSP430 family of microcontroller. The manufacturer's website states the following as the main specifications of this family of microcontrollers:

- Ultra low-power consumption
- Several peripherals
- 16 bit Reduce Instruction Set Computer (RISC) architecture
- Five low-power modes
- Digitally Controlled Oscillator (DCO)
- Fast wake up from low-power modes

We decided to use the MPS430F54xx [73] based upon a previous thesis project done by Joaquin Toledo [74] within the same department where we are doing our thesis. The MPS430F54xx can be clocked at up to 25 MHz and includes three 16-bit timers, a high performance 12-bit ADC with 16 analog channels, hardware multiplier 32x32, Direct Memory Access (DMA), a real-time clock module with alarm capabilities, and various Universal Serial Communication Interfaces (USCI).

Today Texas Instruments offers various MPS430F54xx MCUs. Table 3-1 shows the main differences between them.

Table 3-1: Comparison of MPS430F54xx

MSP430F54XX	Flash (KB)	SRAM (B)	GPIO	Pin/Package	USCI_A	USCI_B
					UART/LIN/IrDA/SPI	I2C & SPI
MSP430F5418A	128	16384	67	80LQFP	2	2
MSP430F5419A	128	16384	87	100LQFP	4	4
MSP430F5435A	192	16384	67	80LQFP	2	2
MSP430F5436A	192	16384	87	100LQFP	4	4
MSP430F5437A	256	16384	67	80LQFP	2	2
MSP430F5438A	256	16384	87	100LQFP	4	4

For our gateway we needed serial communication between the MCU and the transceiver and the Ethernet controller. So only two independent serial interfaces were necessary. Furthermore, no additional signals will be required. For these reasons, we chose to work with the MSP430F5437A [75].

A major feature of this microcontroller is that it provides two Serial Peripheral Interface (SPI) buses. These will be used to simultaneously connect to the Ethernet controller and the 868 MHz transceiver. Additional features that make it particularly suitable for use in the sniffer gateway are its ultralow power consumption (enabling it to be powered using PoE) and the ease of developing and debugging programming using Texas Instruments' Code Compose Studio tool.

3.2.2 Powering

As we explained in section 1.3, we have considered powering the gateway through PoE (see section 2.9). The main advantages are exposed in section 2.9.1.

In order to carry out this we need incorporate a device responsible of applying the correct voltage according with the IEEE 802.3af [44]. The selected device was the TPS2375 of Texas Instruments [76] and it contains all the features needed to envelop an IEEE 802.3af compliant PD. We chose working with this device because it is from Texas Instruments, as the microcontroller is.

In addition to use PoE for supply power to the gateway we also included in our design an auxiliary DC connector for use with an external DC Power Supply. These typical suppliers usually provide 5V, 9V or 12V.

To select among these two different ways for powering, we added a 3-position jumper. Thus, the user can choose with only changing the jumper position.

3.2.3 Ethernet controller

The Ethernet controller module is responsible for connecting the gateway with other network attached devices. The physical Ethernet connector will also be used supply power to the sniffer gateway using PoE technology. The Microchip ENC28J60 [77] has been chosen due to its SPI which will be used to communicate with the MCU. To control and monitor this Ethernet controller we read and write to the control registers via SPI. This Ethernet controller also integrates a dual port Random Access Memory (RAM) buffer for received and transmitted data packets avoiding the need for external memory for packets. Additionally, this Ethernet controller includes all the necessary MAC and PHY modules.

3.2.4 User interface

The function of this part is to ease the communication between the user and the gateway. It is composed of a couple of buttons and light emission devices (LEDs). One button is programmed to reset the MCU while as the other one has a general purpose and it is configurable by the user. The LEDs are also user programmable.

This interface also incorporates the part related to the programming the MCU. We inspired in The Wasa Board Project [78], created by Professor Mark T. Smith (professor of the same department where we developed this work). We used the MSP-FET430UIF to load our software onto the MCU. This tool was borrowed from Professor Smith. It is a powerful flash emulation tool that includes USB debugging interface used to program and debug the MSP430 in-system through the JTAG interface. It works in conjunction with the CCS and supports development with all MSP430 flash devices. We can erase and program the flash memory in seconds.

3.2.5 RF interface

This radio frequency interface consists of the transceiver (or only receiver in our initial case) and all the auxiliary circuitry (i.e. an antenna).

There are a lot of commercial solutions that can be used to design and implement the RF receiver. To simplify the design process, we will use a transceiver or receiver from the same manufacturer as the MCU. Texas Instruments offers a set of transceivers and receivers in the sub-gigahertz band that could suit our design. After studying each of these receivers, we created a table (see Table 3-2) that summarizes the main features of each of them in order to decide which is the most suitable for sniffer gateway.

Based upon the analysis that is described in the next chapter, we choose the C1101 [79]. This transceiver has high receiver sensitivity, supports high data rates, and can work with several types of modulation. This chip can also work as a transmitter, but initially in this thesis we will design a receiver only gateway. (Note that the Ethernet interface is bi-direction and it is only the radio that we are limiting to being a receiver.)

Texas Instruments offers another solution for the receiver based on a System On Chip (SoC). For example, the CC430F6137 [80] integrates a MPS430 microcontroller and a RF transceiver based on the CC1101. This is an interesting option because it avoids the need for external wiring between the MCU and the RF transceiver. However, in order to build directly on the work of Joaquin Toledo we will use separate devices for the MCU and RF transceiver for the first version of our sniffer gateway.

Table 3-2: Comparison of potential Texas Instruments RF receivers

	Features						
	Frequencies	**Modulations**	**Function**	**Maximum data rate**	**Maximum output power**	**Maximum signal to noise ratio**	**Other features**
CC110L	315/433/868/ 915 MHz ISM/SRD	2-FSK, 4-FSK, GFSK, OOK	Value Line Transceiver	600 kbps	+12 dBm	-116 dBm	64-byte RX and TX FIFO Bands: 300-348, 387-464, 779-928 MHz
CC113L	315/433/868/ 915 MHz ISM/SRD	2-FSK, 4-FSK, GFSK, OOK	Value Line Receiver	600 kbps		-116 dBm	64-byte RX FIFO Bands: 300-348, 387-464, 779-928 MHz
CC1000	315/433/868/ 915 MHz ISM/SRD	FSK	Single-Chip Very Low Power RF Transceiver	76.8 kBaud	+10 dBm	-100 dBm	Programmable frequencies: 300-1000 MHz
CC1050	315/433/868/ 915 MHz ISM/SRD	FSK	Single-Chip Very Low Power RF Transmitter	76.8 kBaud	+10 dBm		Programmable frequencies: 300-1000 MHz
CC1101	315/433/868/ 915 MHz ISM/SRD	2-FSK, 4-FSK, GFSK, MSK, OOK, ASK	Low-Power Sub-1 GHz RF Transceiver	600 kbps	+12 dBm	-116 dBm	64-byte RX and TX FIFO Bands: 300-348, 387-464, 779-928 MHz
CC1120	170/433/868/ 915/950 MHz ISM/SRD	2-FSK, 4-FSK, 2-GFSK, 4-GFSK, MSK, OOK	High Performance RF Transceiver for Narrowband Systems	200 kbps	+16 dBm	-123 dBm	Channel spacing down to 12.5 kHz 128-byte RX and TX FIFO Bands: 164-192, 410-480, 820-960 MHz
CC1121	170/433/868/ 915/950 MHz ISM/SRD	2-FSK, 4-FSK, 2-GFSK, 4-GFSK, MSK, OOK, analog FM	High Performance Low Power RF Transceiver	200 kbps	+16 dBm	-117 dBm	128-byte RX and TX FIFO Bands: 410-480, 820-960 MHz

3.3 Gateway's final look

The gateway has been built over two PCB. The main board (motherboard) incorporates the MCU, the powering circuitry, the user interface, and the Ethernet part. The RF part is embedded on another smaller board. We chose build two boards instead of only one because thus the gateway can be used within a different WSN with only replacing the RF board. This determination opens a wide range of opportunities because of the different wireless technologies (as we explained in section 2.2). For example, if the user wants to sniff sensors following the IEEE 802.15.1 [5] he only has to redesign the RF board in order to work within the correct frequency (2.4 GHz instead of 868 MHz).

The design of the gateway is symbolized with its schematic (see Appendix D on page 75) and its layout (see Appendix E on page 79). The motherboard's front and back sides are shown in Figure 3-6 and Figure 3-7 (respectively). The front and back of daughterboard are shown in Figure 3-8 and Figure 3-9.

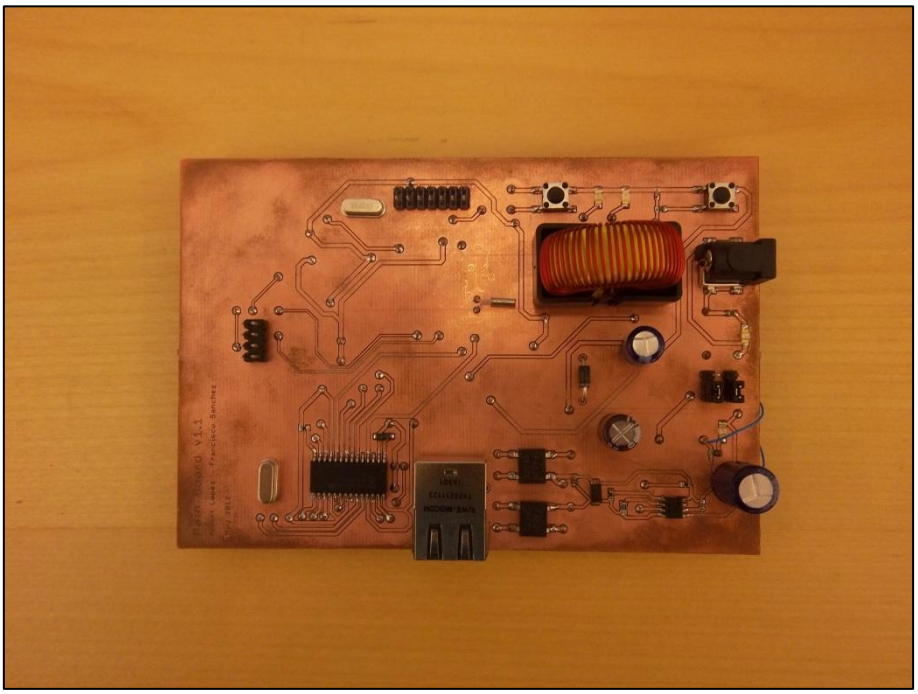


Figure 3-6: The motherboard's front

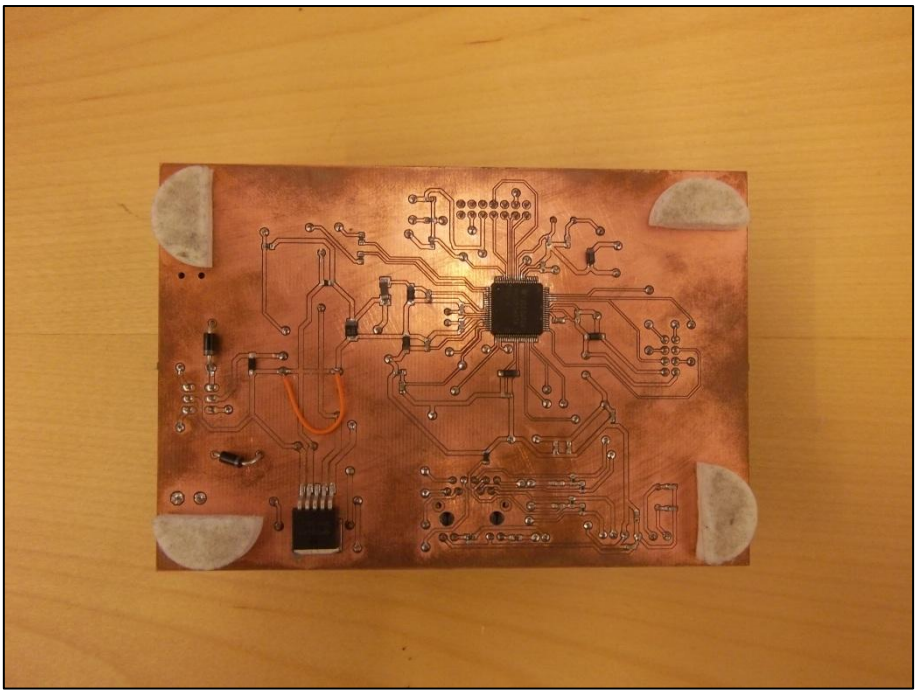


Figure 3-7: The motherboard's back

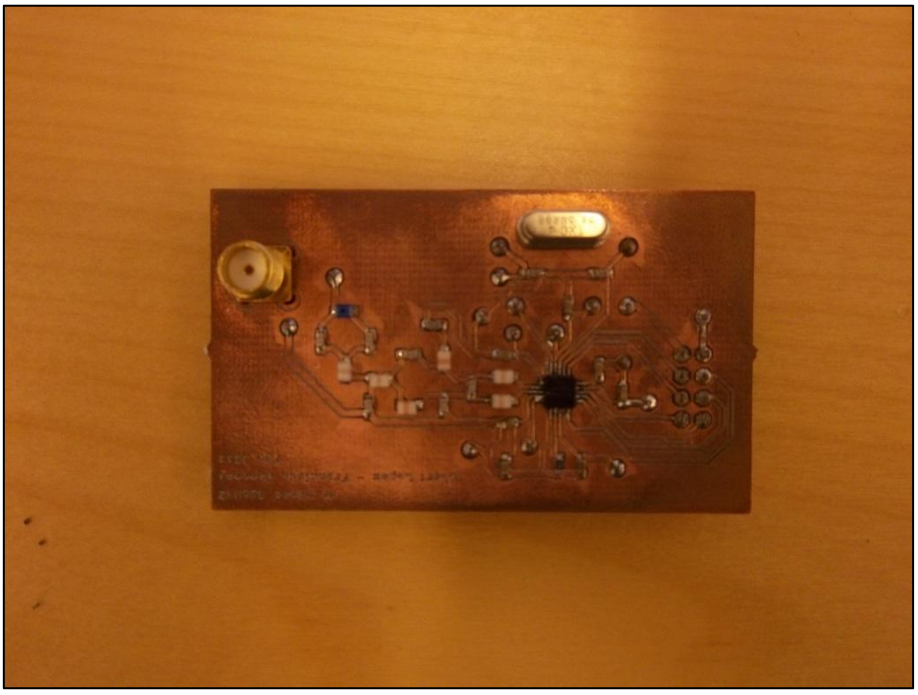


Figure 3-8: The daughterboard's front

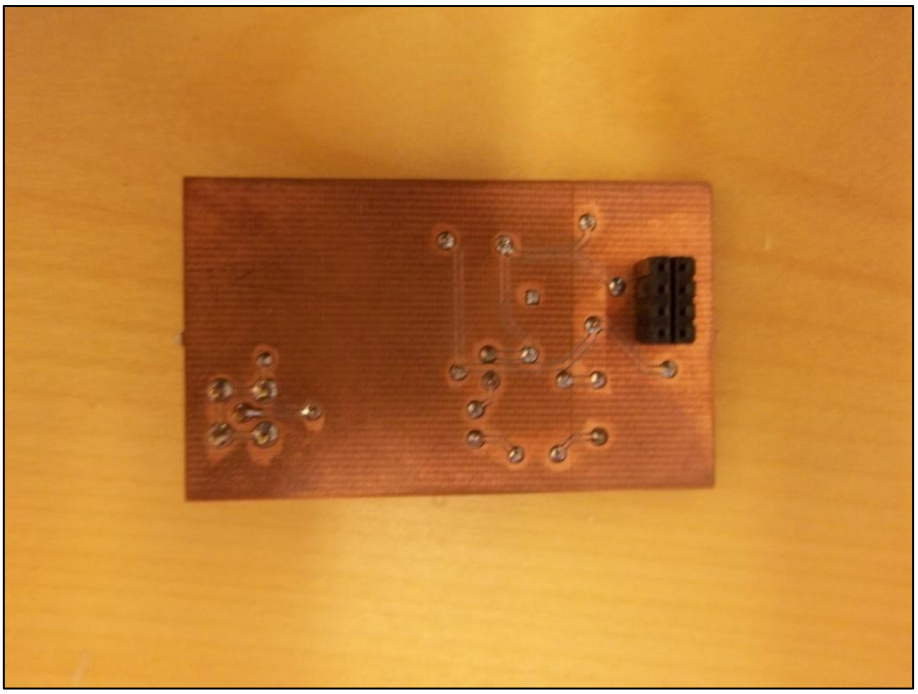


Figure 3-9: The daughterboard's back

Finally the entire gateway is shown in Figure 3-10. The antenna (a 10 cm dipole) placed on the daughterboard was borrowed from one of the wireless routers available in the lab.

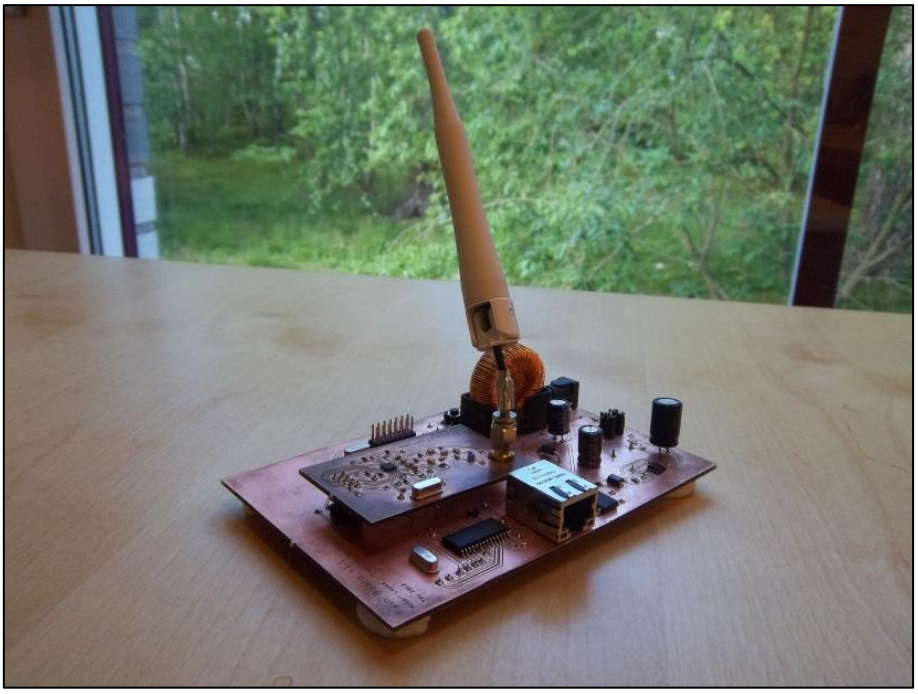


Figure 3-10: The gateway's front

4 Applying the Method (Operation)

This chapter describes the implementation details of the application specified in the previous chapter (Chapter 3). First, in section 4.1, we will explain how what we have done to decrypt the proprietary protocol used in the wireless communication of the sensor under study (Chapter 3, section 3.1.1). After that we will describe how the gateway works. An evaluation test will be described in Chapter 5.

4.1 Breaking the proprietary protocol

In this section we are going to explain the procedure we have followed to decode the proprietary protocol of our sensor under study and extract the information sent by the transmitter. We will illustrate this with various figures (some of them are output from MATLAB[**]) to facilitate our explanation of the procedure we followed to decode the received frames.

Section 4.1.1 describes how we sniffed the transmissions of the temperature transmitter with the USRP. To do this, we used software written in Python to visualize the spectrum of the signal and to store the digitized signal in a file that we analyzed later.

In section 4.1.2 this file is read and the digitize signal was analyzed with MATLAB. Once we had the appropriate signal samples, we analyzed this data in order to distinguish bits and to determine the data rate and the length of each transmission (how many bits were transmitted in each transmission by the temperature transmitter). We wrote some scripts to determine the time intervals at which the data is transmitted by the transmitter and removed all of the other samples from the digitized signal (i.e., we kept only the samples during the times when the transmitter was actively transmitting). In the following sections we describe the details of how we determined where the data fields were in each transmission and how we identified the purpose of each of these data fields.

4.1.1 Initially capturing data from the sensor

The first problem to overcome is to determine the operating frequency of this wireless sensor. This first problem is relatively easy to solve with a spectrum analyzer. The next (and more major) problem is to receive the transmitted signal and to decode the proprietary protocol. Each such protocol may have its own frames, fields, and even encryption.

Before capturing data, it is necessary to understand roughly how the sensor works. Initially, based upon the documentation that came with the transmitter we only knew that the transmission frequency was at roughly 868 MHz and that the transmitter only transmits every 4 seconds, in order to save battery power. Therefore we did not expect to find a continuous transmission, but simply a fast burst of data every 4 seconds.

We used the spectrum analyzer to visualize the frequency components and to learn the exact frequency of the device's transmissions. As it was expected, nearly all of the time only noise was detected by the spectrum analyzer, but sometimes a peak appeared at approximately 868.265 MHz. Most of these peaks were missed or hard to see because of their short duration. However, a longer peak was displayed when the sensor was

[**] © 2012 The MathWorks, Inc. MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See www.mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

initialized, that is, when the batteries were first put into the transmitter. Thus it was possible to determine the approximate center frequency and the bandwidth of our signal. However, the long sweep time (i.e., the time it takes the spectrum analyzer to sweep through a range of frequencies) and the lack of a digital interface to the analyzer made this analysis quite complicated.

To overcome the difficulties of collecting the data that we wanted using the spectrum analyzer we used the USRP with the DBSRX daughterboard, together with the GNU Radio software to receive transmissions between 800 MHz and 2.4 GHz. Since this wide range contained the temperature transmitter's transmission frequency we tried to visualize the frequency components of the signal in real-time with the GNU Radio file *'usrp_fft.py'* (see Appendix A starting on page 69). This script works as a spectrum analyzer, and it showed the RF frequencies that the USRP received. Figure 4-1 shows an example of the output from this script when the center frequency of the receiver was set to 868.265 MHz. Note that this figure does not show the output for the whole spectrum from 800 MHz to 2.4 GHz, although the received power in this range could have been computed and displayed to estimate the center frequency of the transmitter - we used the knowledge learned from using the HP spectrum analyzer to skip this step. Additionally, we might have used another input to the USRP to know when the microcontroller in the temperature transmitter actually sent data to the transmitter chip and used this to trigger our data capture – thus increasing the easy of collecting samples during the device's transmission.

Given what we have learned, the next step was record the signal. This objective was achieved with the python file "*usrp_rx_cfile.py*" (see Appendix A). This script takes several arguments that specify: the frequency, the number of samples you wanted to collect, the gain of the amplifier, the data type of the samples, and the decimation (this value fixes the bandwidth of the capture). Executing this script in a Linux terminal creates a file containing the data collected from our temperature sensor.

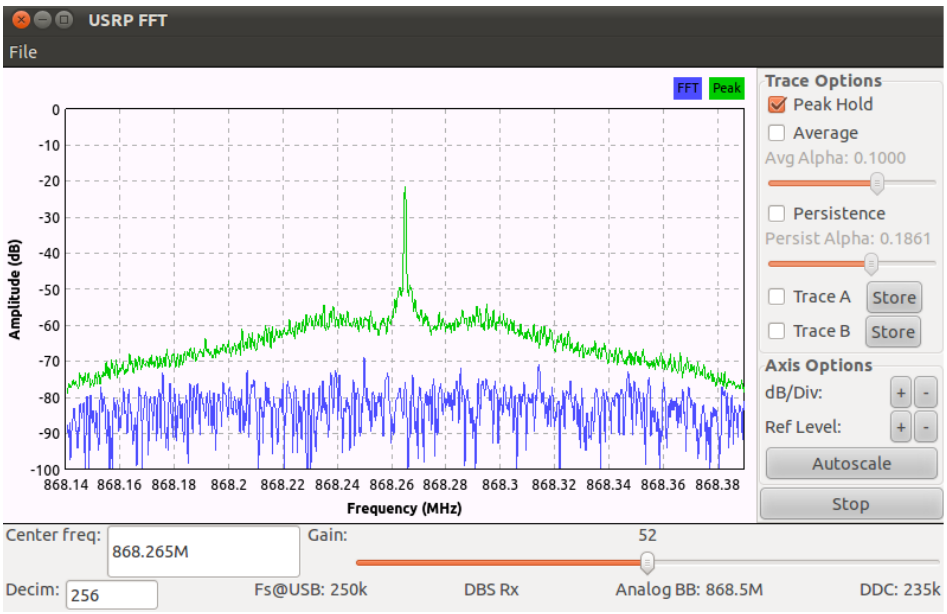


Figure 4-1: Spectrum analyzer designed with the script "*usrp_fft.py*"

4.1.2 Decoding the received signal

A software tool to deal with the captured samples of the signal was required. Although there were a number of different possibilities, we chose MATLAB as it is a very powerful tool that offers to the user many possibilities for signal processing. It allowed us to read the information recorded with the USRP and GNU Radio software, and to easily analyze this data.

We first attempted to view the signal. The MATLAB function "*read_complex_binary.m*" converts the binary samples into a complex vector. As the data type is complex, we will display in Figure 4-2 the real value of the vector's elements. As we expected, a short transmission is visible above the noise level every 4 seconds.

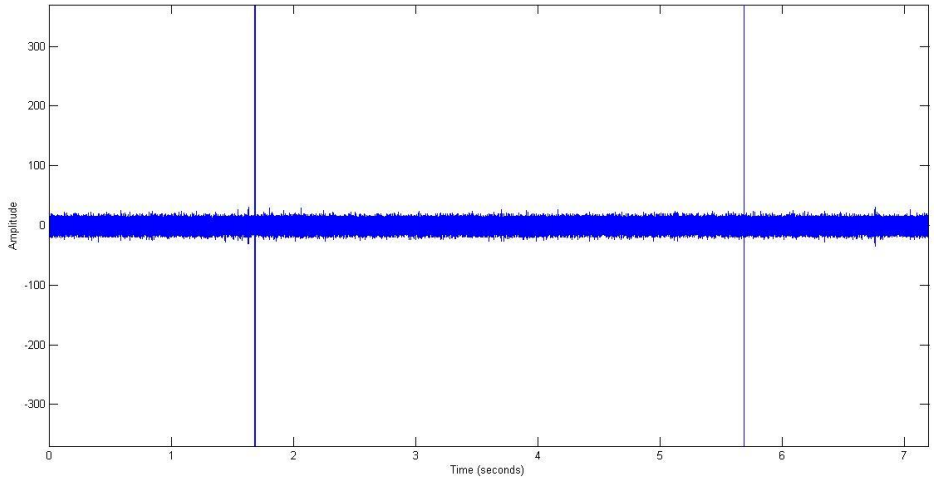


Figure 4-2: Two different transmissions separated 4 seconds

Figure 4-3 shows the details of part of one such transmission. This picture shows only a fraction of one of the transmissions in order to give a better view of how the signal changes with time.

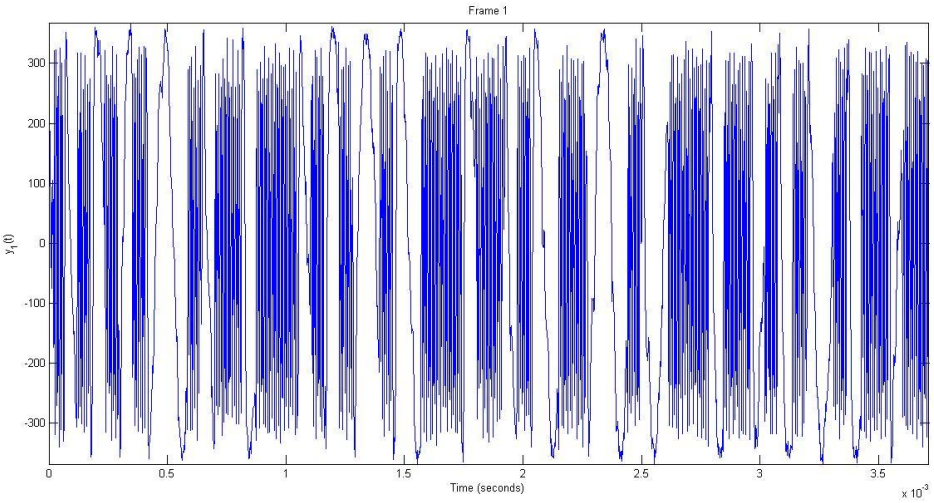


Figure 4-3: One piece of frame

As can be seen in this figure, there are two different frequencies of oscillations, which lead us to believe that the transmitter is using Frequency Shift Keying modulation (FSK). If so, one of those frequencies corresponds with a logical '1' and the other with a logical '0'. As the transmitter has only a very limited amount of information to communicate to the intended receiver, we expect that the modulation and coding used will be simple (otherwise the required computational power and the battery power consumption at the transmitter would both be larger). Furthermore, the transmission time can be determined at this point. A frame lasts about 3.7 milliseconds. We can also count 64 different fields so we assumed that each frame is composed of 64 bits. This gives duration for each bit of 58 microseconds, which corresponds to a bitrate of 17241 bps.

Figure 4-4 shows the spectrum of the signal. We can see two main frequency contributions (f_1 and f_2) and some harmonics. It is possible to compute the frequency deviation as a (f_2 - f_1)/2.

$$f_{dev} = \frac{f_2 - f_1}{2} = \frac{113.7 - 7.004}{2} kHz = 53.3\ kHz$$

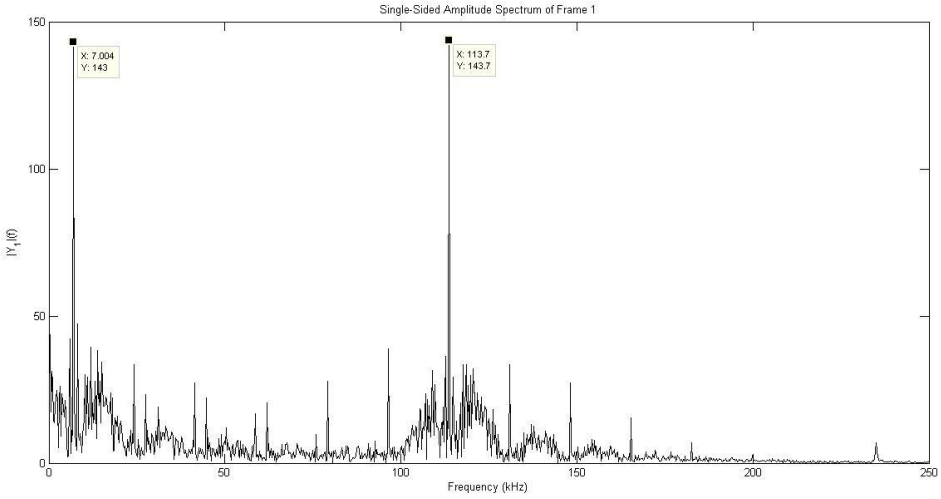


Figure 4-4: Spectrum of one frame

Once the modulation is known to be FSK, a considerable number of samples of different temperature were recorded in order to compare the received signals, hence enabling us to extract meaningful data from the received signal. To do this automatically, it is first necessary to automatically extract ones and zeros from the recorded signal. We developed a MATLAB script to perform this task (see Appendix B starting on page 71). The main idea is first, to detect the different frames (appearing each 4 seconds approximately), and after that, to process each bit separately, based upon the oscillation frequency determine if the bit is a '1' or '0'.

Figure 4-5 illustrates the same part of a frame showed in the previous figure (the signal is shown in red) and the extracted bits (shown in the figure as blue balls). We assumed that high frequency corresponds to '1' and the low frequency, to '0', but when analyzing the data we will check if it this assumption is correct or not.

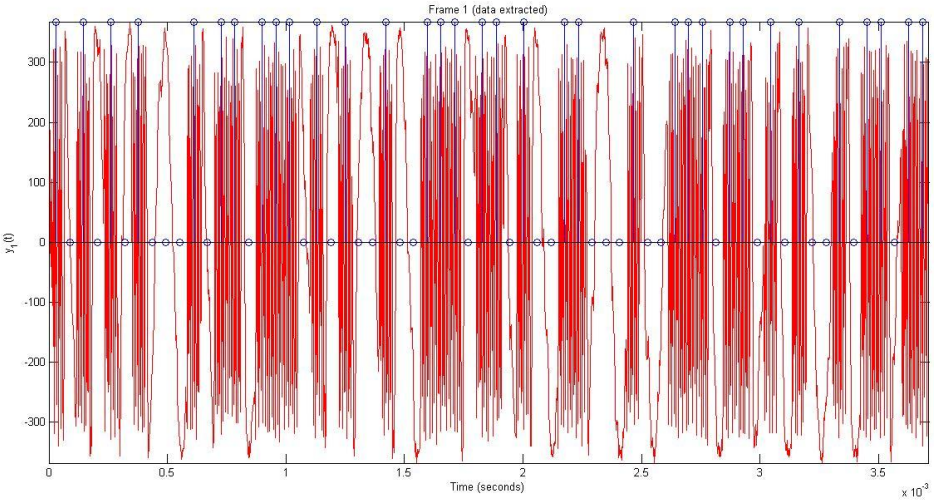


Figure 4-5: Part of one frame (data once extracted)

4.1.3 Analyzing data

In this point, we can capture the signal and extract the bit information. Now, some questions emerge. What do these ones and zeros represent? Which field of the frame contains the temperature data? How is the message encoded and how is the data coded? All of these questions and more are analyzed and answered in this section.

Before this analysis, we spent a little time thinking about what could be in the frame. At the beginning of the frame we expect to find a preamble. Such a preamble is employed to synchronize the receiver, that is, when the station receives the preamble, it knows that a frame is being sent and the receiver can synchronic its clock to the transmitter. The frame might also contain an identifier of the transmitter, in order to distinguish between various transmitters sending information simultaneously in the same environment. The frame also needs to contain the temperature. Finally at the end of the frame might be a field to be used to detect errors that may have been introduced during the frame's transmission. This final field might be a checksum, hash sum, or a CRC. All of these fields were expected to be found in the frame.

To be able to decode the frame a number of frames were captured. As the range of temperature that this temperature transmitter was supposed to be able to measure ranged from – 39.9º to 59.9º Celsius we carried out a set of measurements to cover as much of this range as possible. Some temperatures were not easy to obtain naturally, so the sensor needed to be heated and cooled artificially (using light bulbs and a refrigerator, respectively) to reach the desired temperature for the measurement.

While exposing the temperature sensor to a range of temperatures we recorded its transmissions using the USRP and stored this data in a file – while simultaneously noting the temperature displayed on the TFA receiver. Then, we loaded the resulting file is into MATLAB and applied the scripts that we had developed. The MATLAB output (i.e., the bit corresponding to each of the frames) was stored in a spreadsheet, enabling us to easily organize this information. This spreadsheet (shown in Figure 4-6) helps us to carefully examine these frames.

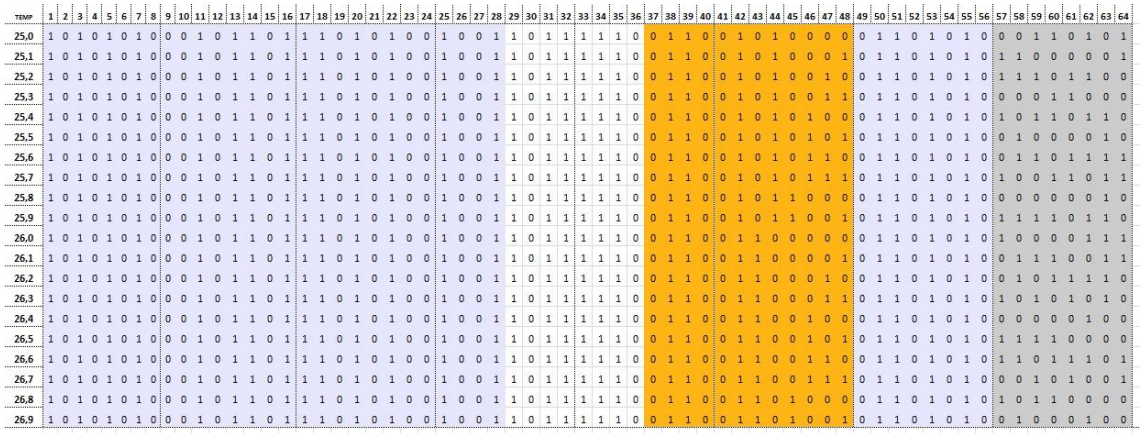


Figure 4-6: Some known positive temperature values - the fields where the bits varied is highlighted in orange

Figure 4-6 shows only a portion of our spreadsheet. The complete set of bit fields from the file 'Temperatures.xls' are shown in Appendix C starting on page 73. The bytes and different fields have been colored to facilitate examining the frames. The first thing to note is changing and unchanging fields. When different measurements are made when changing the temperature between the different measures, we notice the orange colored bits (in the spreadsheet) and the last byte are the only changing fields. At first sight, the last byte does not seem to follow any logical order; hence we assumed that this field corresponds to a checksum or a CRC-8. Now that we have identified the bits that seem to carry the temperature value we need to decode these bits, the procedure to do this is explained in the next subsection.

4.1.4 Temperature field

Our analysis started with the positive temperature values. Focusing on the orange colored bit cells, if we look at the values received when the temperature was reported to range from 25.0 to 25.9, we notice that when temperature increases by a tenth of a degree (Centigrade) that this field increases by 1 when decoded as decimal digits. Further when the temperature increases from 25.9 to 26.0, the last 4 bits (the 4 right most bits, bits in columns 45th to 48th) change from '1001' (in decimal: 9) to '0000' (in decimal code: 0). This indicates that the last 4 bits of the orange cells represent the tenth of the degree encoded as a Binary Coded Decimal (BCD) digit. Applying this hypothesis about the encoding to the rest of the data collected for positive numbers, we determine that this hypothesis is correct.

Following this pattern, when now focus on the bits in columns 41st to 44th. Between 25.0 and 25.9, the unit's digit is 5, and we see a '0101', i.e., a decimal 5 in these bits. A similar pattern occurs for the range between 26.0 and 26.9, but with the decimal digit six. Applying this decoding to all of the positive temperature values in the spreadsheet we see that all the units place of all positive numbers are encoded in BCD.

Following the previous deductions, we might think that the value of the tens digit must be represented in BCD in the 4 first orange cells (bits 37th to 40th). However, between 10.0 and 19.9, that field is '0101' (5 decimal), from 20.0 to 29.9, is '0110' (6 decimal), between 30.0 and 39.9, is '0111' (7 decimal), and so on. When the tens rises one, the value in that field also increases by 1, but the value does not give us the expected decimal value for the corresponding temperature. To solve this difficulty, we examine these bits that are associated with negative temperature values (some of these are shown in Figure 4-7).

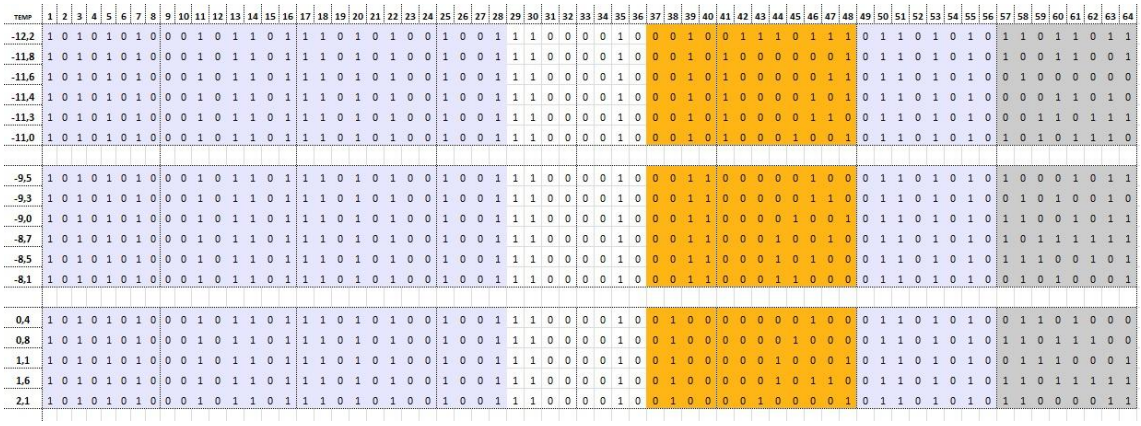


Figure 4-7: Part of the spreadsheet showing the bits for some negative temperatures values

The lowest temperature value that we acquired was -12.2º. However, this was sufficient to deduce how the tens digit of the temperature is codified. By considering the lowest temperature value which can be measured by this sensor: -39.9º, then according to the pattern observed, we realized that this temperature would be codified as '0000 0000 0000'. While the highest temperature, +59.9º, would be '1001 1001 1001'. Based upon this conclude that each group of 4 bits works as a counter mod-10, taking the carry to the next group of 4 bits. The algorithm to extract the temperature value from the bit field is:

- Decode the temperature field as BCD, divide it by 10 and subtract 40 to the result. Example: '0100 0010 0001' (421 in BCD) corresponds with 2.1º. (421/10) – 40 = 2.1.
- If the result of the first step is smaller than 0, add one tenth.

Checking this algorithm against all the data that we recorded, it is always correct; hence we have found out how the temperature field is encoded.

4.1.5 Identifier field

Another changing field is the bits from 29th to 36th. As we have 2 sensors, we collected data from both to examine the differences between their transmissions for the same temperature value. It turned out that some bits in this field vary depending on the transmitter, as was expected. However, when we removed and inserted the batteries in a given sensor, we realized that the identifier field changes, too. Our conclusion is that a random 8 bit identifier is generated when the sensor is initialized. In this way, it is unlikely that two sensors will chose the same identifier. It is for this reason that the temperature transmitter has to be initialized before the TFA receiver and that if the temperature transmitter's battery is changed then the TFA receiver has to be re-initialized.

This sensor identifier field is important for our thesis. As multiple transmitters could be within range of the gateway, the gateway will use this identifier field to differentiate between these sensors. Therefore, theoretically our gateway could listen to up to 255 temperature sensors of this type. However, as a practical matter it is unlikely that there will be this number of such temperature transmitters within range and if there were this large a number of devices, then the probability that a given temperature transmitter would choose an identifier value that is already utilized by another transmitter would be quite high. Since the temperature transmitters do not listen for other transmitters nor

does the temperature transmitter handshake with the TFA receiver, hence the temperature transmitters would not know that they had chosen the same identifier.

4.1.6 Last byte

The last byte appears to always be random, i.e., it does not follow a logical order. As noted earlier we assume that this is some sort of CRC-8 checksum, however the algorithm or the polynomial employed is not (yet) known.

Hence at the present time we have not decoded this field. Of course, it would be useful to find know how this byte is generated. For the purposes of the gateway, the most important fields for us are the temperature and the identifier and we have now figured out how to decode them.

4.1.7 Rest of frame

The rest part of the frame does not change. It is always the same, so we can do not know what it represents. We guess it is a code indicating the company and the family of products, but it is not (yet) possible to determine this. However, we do note that the first byte is always '1010 1010', hence we assume that these alternating bits are a preamble that is used by the receiver to synchronize itself to the transmitter.

Later we may acquire another IT+ device from this same vendor, but with a different type of sensor to see if we can decode more of the rest of the frame. Additionally, we should obtain another IT+ sensor from another vendor to see which fields change.

4.1.8 Comparative with Bossard's work

After finishing decoding the transmissions, we found an interesting related work that could have helped us. Fred Bossard [81] uses the transmitter TX29-IT+ [82] working in the 868 MHz band with the associated weather station WS-7014-IT [83]. These devices belong to La Crosse family.

In his works, Bossard analyzes and decodes the messages received in the weather station. The transmitter works as ours, transmitting burst data every 4 seconds. After some experiments, Bossard deduces all the fields of the received messages.

- The first byte is always a preamble ('1010 1010') and may be used to synchronize the receiver clock.
- The second byte is '0010 1101'and the third is '1101 0100'. These fields are another synchronize pattern and they appear in all the frames.
- Bits from 25^{th} to 28^{th} are always '1001'. This gives information about the message length in number of quartets that follows.
- Bits from 29^{th} to 34^{th} are the sensor identification. These 6 bits allows up to 64 different transmitters and it is random generated when the transmitter is power up.
- Bit 35^{th} is a new battery indicator. When the transmitter is power up, this bit is set to '1' during a little time.
- Bit 36^{th} is unused and it is always '0'.
- Temperature field (bits 37^{th} to 48^{th}) is BCD coded (as we reasoned).

- Bit 49th is a weak battery indicator. We assumed that it is '1' when the batteries are too old. In our experiments, this bit is always '0'. This is because we used new batteries for the transmitters.
- Bits from 50th to 56th are a special field that changes depending on the kind of transmitter. If the sensor can also measure the humidity (i.e. it is an hygrometer), the value in % is shown in this field, hence the maximum value is 0x64 in hexadecimal (100 in decimal). However, if the sensor has not this option, this field does not change and its value is always 0x6A in hexadecimal. That is the case of our transmitter.
- The last byte is a CRC-8 checksum that follows the polynomial $x^8+x^5+x^4+1$. This polynomial it is applied on 40 bits of data, from 25th to 56th. To check the CRC-8 of the recorded frames, we used an online CRC calculator [84].

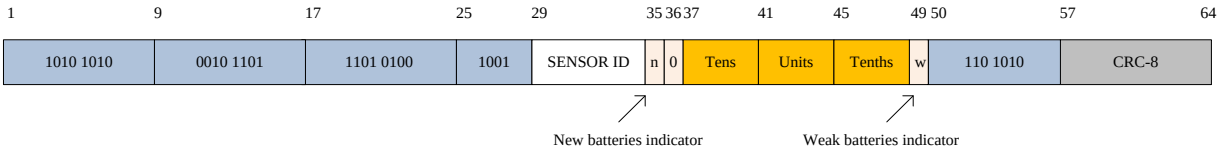


Figure 4-8: All the fields of a frame

We compared the Bossard's work and ours and we totally agree. However, the way we obtained the result was really different than Bossard did. Although some of the fields were not necessary to be identified, we now have a complete knowledge of how the IT+ transmitters work.

4.2 Gateway operation

After decoding the proprietary protocol used by the initial sensor the next step is to let the gateway totally ready to start working.

The gateway basically has to perform two tasks: it has to sniff wireless data from sensors in the environment and transmit it over Ethernet to an IP network. We can choose if we are going to process this data (disassembly the packet and keep only the necessary fields) before transmitting or transmit the raw packet. This last option frees the gateway to do complex operations but force the user to perform this task; hence we decided to do this alternative. Finally, the gateway also has to implement all the functions to handle Ethernet packets according with the Internet Protocol Suite (Chapter 2, section 2.4).

An overall gateway operation is represented in Figure 4-9. It has been thought as a Finite State Machine (FSM). The process changes from one state to another if the right requirement has been accomplished. Designing the gateway tasks as a FSM makes this performance more transparent and easy to understand. The Ethernet and the RF interface become independent.

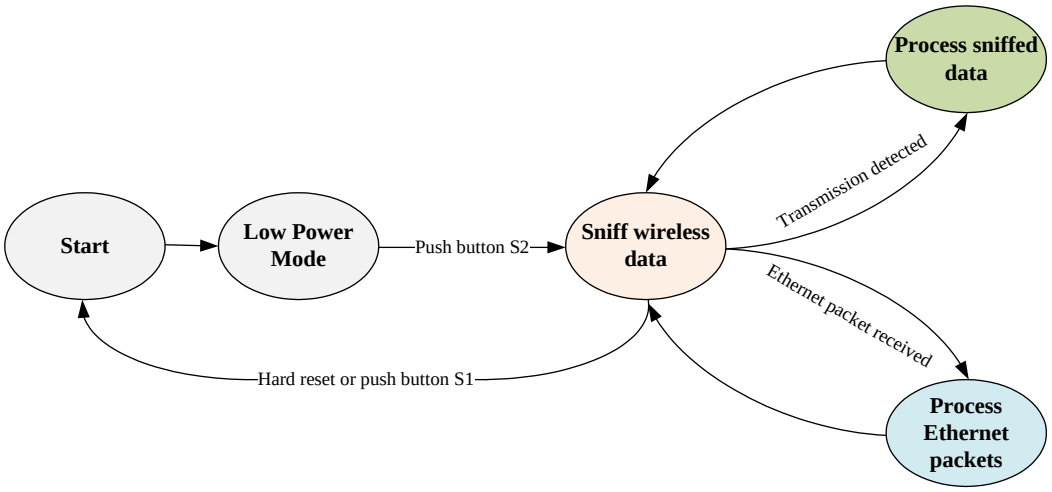


Figure 4-9: Gateway operation represented as a finite state machine (FSM)

The first state, *Start*, is reached when the gateway receives enough power to be able to work. After that, the MCU enters in a low-power state, reducing its consumption. Some of peripherals and functions are disabled in this state. The gateway remains in low-power mode until the button S2 is pressed. It activates all the functionalities and the MCU returns to the default active state. In this phase, the gateway initializes the Ethernet controller as well as the transceiver, writing in the own registers. Below, the MCU waits for some interrupt from or the RF transceiver or the Ethernet controller. Depending on the source of this interrupt, the MCU will process the packet incoming from the corresponding interface. If the button S1 is pressed at any time, a reset will be performed and the gateway will return at the *Start* state.

The microcontroller gets access to the Ethernet controller and the transceiver though a SPI bus. This is a synchronous serial data link standard operating in full duplex mode. It is based in master/slave model where is the master who always initiates the communication. Multiple slaves are allowed with individual slave select (chip select) lines. The SPI bus is composed of four lines: serial clock (SCLK), master output - slave input (MOSI), master input – slave output (MISO), and slave select (SS). To begin a communication, the master first selects the slave device (SS line, active low). After that, the master has to output the clock via SCLK and a full duplex data transmission occurs: the master write a bit on the MOSI line and reads a bit on the MISO line in the same clock cycle. The frequency of the SPI clock is limited by the maximum frequency that each slave device can support. It is possible to select different frequencies for each slave device using *prescaler* with the same clock source or selecting different clock sources.

On the other hand, when the slaves wish to communicate with the microcontroller they have to use some interrupt line. The slaves devices considered in the gateway are the ENC28J60 (Ethernet controller) and the CC1101 (transceiver) and both are provided with interrupt lines that can be programmed to be activated after special events had occurred. Figure 4-10 illustrates these procedures.

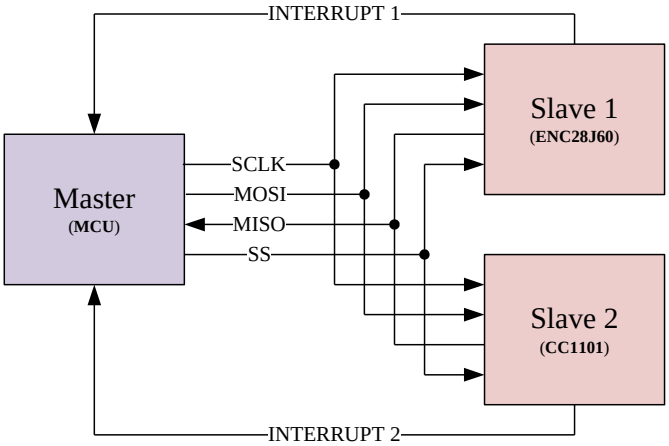


Figure 4-10: SPI bus. Master/Slave model with interrupt lines

In the following sections we are going to explain the different procedures the gateway executes for each interface.

4.2.1 Radio Frequency interface: operation

This interface corresponds to the part of the gateway responsible for sniffing and processing data from the wireless sensors. We are going to explain the steps that the gateway carries out in order to collect data.

The first task to realize is to configure the transceiver properly. Following the manufacturer datasheet and some application notes it is not complicated to do this task. In section 4.1.2 we figure out the modulation as well as the bitrate. Table 4-1 summarizes the most relevant parameters.

Table 4-1: Summary of the parameters of the transmission

Parameters	
Frequency	868 MHz
Modulation	FSK
Bitrate	17241 bps
Frequency deviation	53.3 kHz
Total length	64 bits

The transceiver chose for the gateway (CC1101) has some configuration registers (more details can be found in [79]). They have to be written with the right values for a properly performance. The easiest way to do this is using the SmartRF™ Studio software [85]. It is highly recommended for obtaining optimum register settings.

Moreover, the CC1101 has built-in hardware support for packet oriented radio protocols. In receive mode, the packet handling support will de-construct the data packet implementing the following:

- Preamble detection
- Synchronization word detection
- Packet length check
- One byte address check
- CRC computation and CRC check

The transceiver will wait for a valid preamble and synchronization word. The synchronization word is a two-byte value that provides byte synchronization of the incoming packet. When found, the transceiver has obtained both bit and byte synchronization and will receive the first payload byte. As we saw in section 4.1.8, the sensor uses a synchronization pattern '00101101 11010100' (0x2DD4). So the CC1101 has to be programmed for receiving this two-byte value after the preamble.

The packet length check is useful for discard packets with another length that the expected. In section 4.1.8, the length information was in the 4-bit field following the synchronization pattern ('1001'). The packet length checking used by the CC1101 is designed for one entire byte but it is easy to handle with a bit mask on the upper 4-bit and shifting the bits.

Information about the address is not relevant for our sensor. The transmitter always sends its identification number. But this identification is a random number generated when the batteries are put. So we cannot predict this number and the address check will not be used. Furthermore, CRC check and computation will also not used because the CC1101 only supports CRC-16. As we saw, the sensor uses a CRC-8.

Using this packet handle, receiving a right packet is very easy. Moreover, the CC1101 can be programmed to generate an interrupt on the MCU (through one of its general purpose outputs) when a packet has been received. It prevents the MCU of polling continuously for a new packet incoming.

4.2.2 Ethernet interface: operation

This part of the gateway is responsible for the communication between our gateway and the IP network it is connected to. As we mentioned earlier, hosts that wish to communicate with this gateway can be located on the same network, hence the gateway has to have implemented the Internet Protocol suite. The different protocols that the gateway supports are described in section 2.4 and onwards.

The Ethernet controller (ENC28J60) communicates with the MCU similarly to the way that the transceiver does. The ENC28J60 needs to be properly configured before using it to communicate. It supports packet filtering of incoming packets (based upon a MAC address filter, frame length filter, and CRC filter). The bytes received and the bytes to be transmitted are written into a buffer inside the Ethernet controller. This buffer is divided in two parts: memory reserved for received bytes and memory reserved for bytes to be transmitted. The MCU accesses the appropriate portion of the buffer through read and write pointers (more details can be found in [77]). Once a valid packet has been received, an interrupt request is generated by the ECN28J60 to the microcontroller.

We designed a set of functions which implement a minimal IP stack. This stack is executed periodically by the CPU. Each header is processed individually. These processes follow a layered model through encapsulation and decapsulation. When an interrupt from the ENC28J60 occurs, the MCU starts processing as described in Figure 4-11. If the packet received is an ARP Request, then the gateway will process the packet and will reply with an ARP Reply packet. If the packet is an IP packet, the higher protocol can only be ICMP or UDP. In case of ICMP, only an ICMP Request is expected with its subsequent ICMP Reply is generated by the gateway. Finally, if the transport protocol is UDP a counter will be incremented and the datagram passed to higher layer protocols. The counter will be used by the higher level protocols.

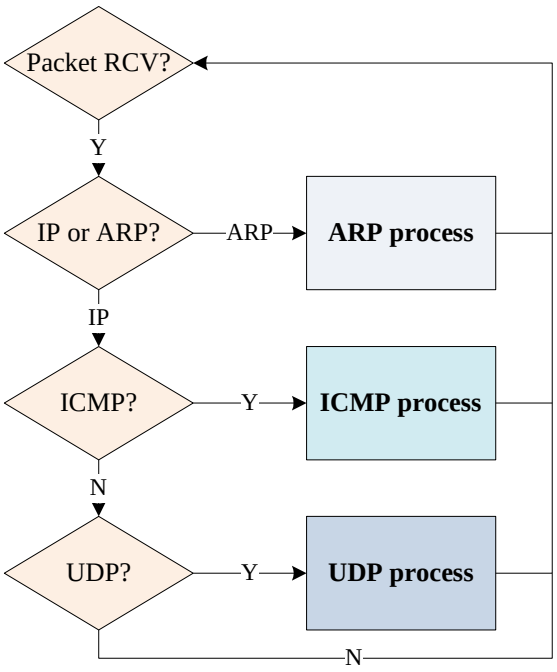


Figure 4-11: Stack process

One of these application layer protocols is DHCP. A DHCP request will be used by the gateway to request a new IP address from a DHCP server. The gateway will make a DHCP when the gateway starts up and when the IP should be renewed according to the lease time given by the DHCP server. We designed a DHCP process (shown in Figure 4-12) that it is executed concurrently with the stack process (previously shown in Figure 4-11).

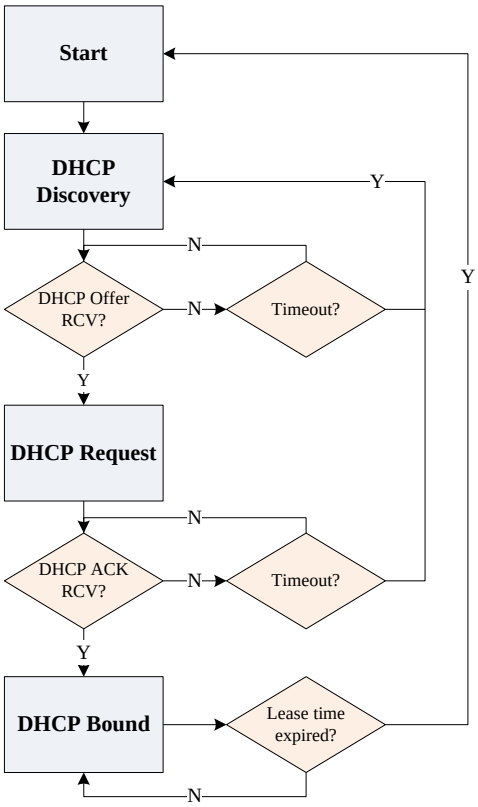


Figure 4-12: DHCP process

Once the gateway gets an IP address it will be able to communicate with other hosts in the network. We created a very simple application protocol called *UDP-SniffedPacket* to enable hosts to request sensor data and for the gateway to send the most recent sensor value. The gateway replies to requests from hosts. In each request the host sends one byte (a command). The gateway will reply with the entire packet sniffed by the CC1101. The procedure for decoding the packet will be performed by the host who made the request. In the next chapter, Chapter 5, we will examine these operations in greater depth.

5 Analysis and Evaluation

This chapter will describe the different procedures that have been used to evaluate the gateway's compliance with its requirements in terms of both correctness and performance. In section 5.1 we will evaluate the RF interface by analyzing the wireless sniffed data. In section 5.2 the network traffic will be examined in order to check how the implemented protocols work. The interoperation of both interfaces will be tested in section 5.3. Finally, in section 5.4 we will evaluate the entire system by powering the gateway by PoE.

5.1 Radio Frequency interface: evaluation

Since we are using only one sensor, for evaluating this part we do not need anything more than the gateway and the sensor (the transmitter and the receiver). The TFA receiver's display will be used as the correct value of the temperature that the transmitter is sending.

We have programmed one of the LEDs on the *motherboard* to blink for each new incoming radio frame. When the transceiver sniffs a wireless frame it generates an interrupt on the MCU. We assumed that the incoming frame follows the structure that we described in section 4.1 (i.e. the preamble, a synchronization word, and the address field are correct). After processing this interrupt the yellow LED on the motherboard blinks once. This LED blinks every 4 seconds as expected from the earlier measurements of the radio's activity with a spectrum analyzer.

After performing a lot of experiments, we saw that sometimes this LED blinked every 8 seconds instead of 4 seconds. In addition, the display of the TFA receiver did not change immediately. Since the CC1101 seemed to work well, we decided to compare the frames received by the gateway with the frame received in the USRP. To our surprise, the USRP also received the frames every 8 seconds. We assume that if the temperature measured by the transmitter does not change for a large period of time, the sensor node increases the transmit interval from 4 seconds to 8 seconds. We validated this assumption by changing the placement of the transmitter, placing it in a new hotter environment caused it to transmit again at 4 second intervals.

Initially we processed the incoming frame in order to extract the temperature field and compared it with the value on the TFA receiver's display. Since the different fields are grouped in quartets of bit, we had to apply bit masks and perform shifts to extract the correct values. Figure 5-1 shows a capture done while executing the program using the debugger. We can observe in the figure the raw incoming frame ("sniffedData") and the processed frame ("sensor"). Once we could decode all the fields properly, we removed this code from the MCU and the incoming frame is no longer processed.

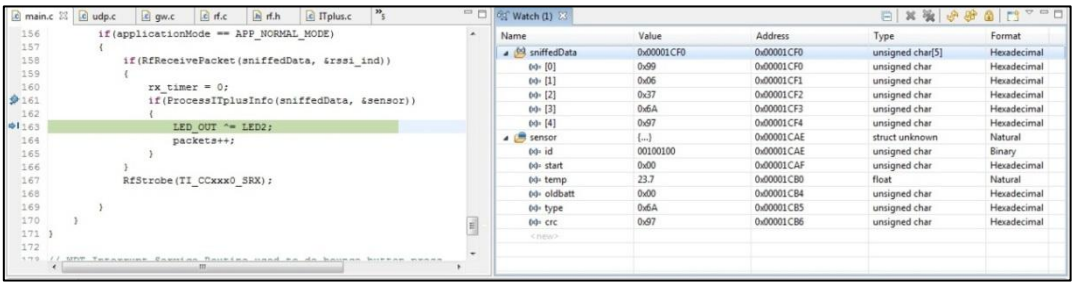


Figure 5-1: Processing of the incoming frame on the wireless interface

5.2 Ethernet interface: evaluation

The scenario in this case involves the gateway and a router. The router will act as a DHCP server. The router can assign a private IP address to the gateway based upon the device's DHCP request. The particular model of the router that was used is the Cisco LinkSys BEFSR41 [86]. To examine all the relevant protocols we used Wireshark [87] to capture and observe these transmissions in real-time. The scenario where we performed this evaluation is composed by the gateway, the router, and a Personal Computer (PC) running Wireshark (see Figure 5-2).

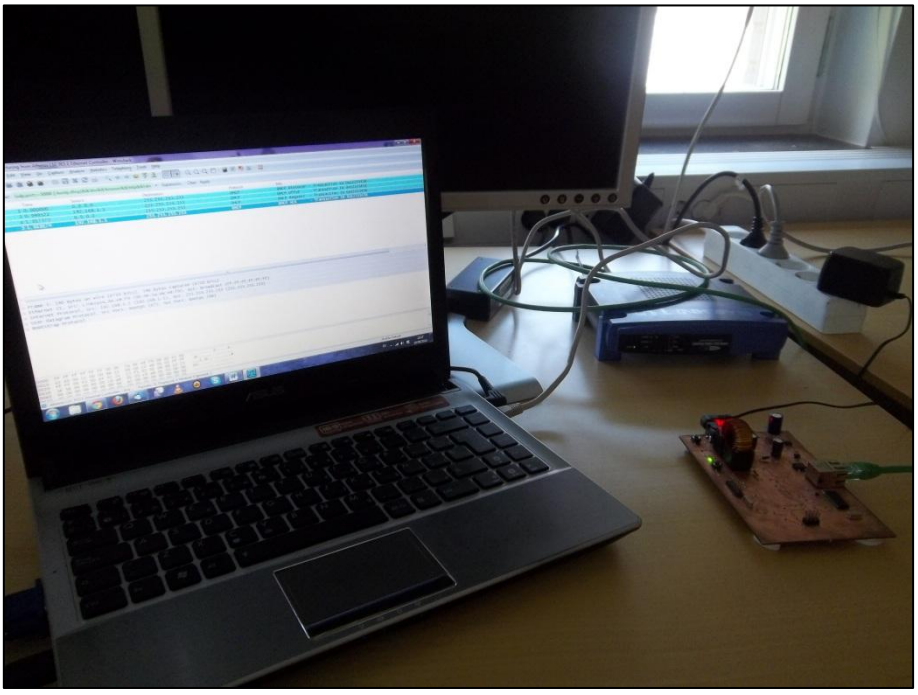


Figure 5-2: Evaluation of the Ethernet interface

This made it very easy to check the proper operation of the DHCP process. The four DHCP packets are shown in Figure 5-3. Initially, the gateway has an IP address of zero. The DHCP Discover request is sent to the link local IP broadcast address (255.255.255.255). For testing purposes we have used the Ethernet MAC address 00:04:a3:00:00:00. This address belongs to the Microchip ENC28J60 Ethernet interface chip – before it has been configured with a specific MAC address. The router, which has an IP address of 192.168.1.1, sends a DHCP Offer packet back to the gateway. In this packet the router offers a private IP address to the gateway. The process continues until the router sends a DHCP ACK. In this case, the IP address offered is the 192.168.1.2. The green LED on the gateway motherboard was programmed to light up after finishing the DHCP process, i.e., when the gateway gets a new IP address. If the LED does not light, this indicates that the gateway does not (yet) has an IP address.

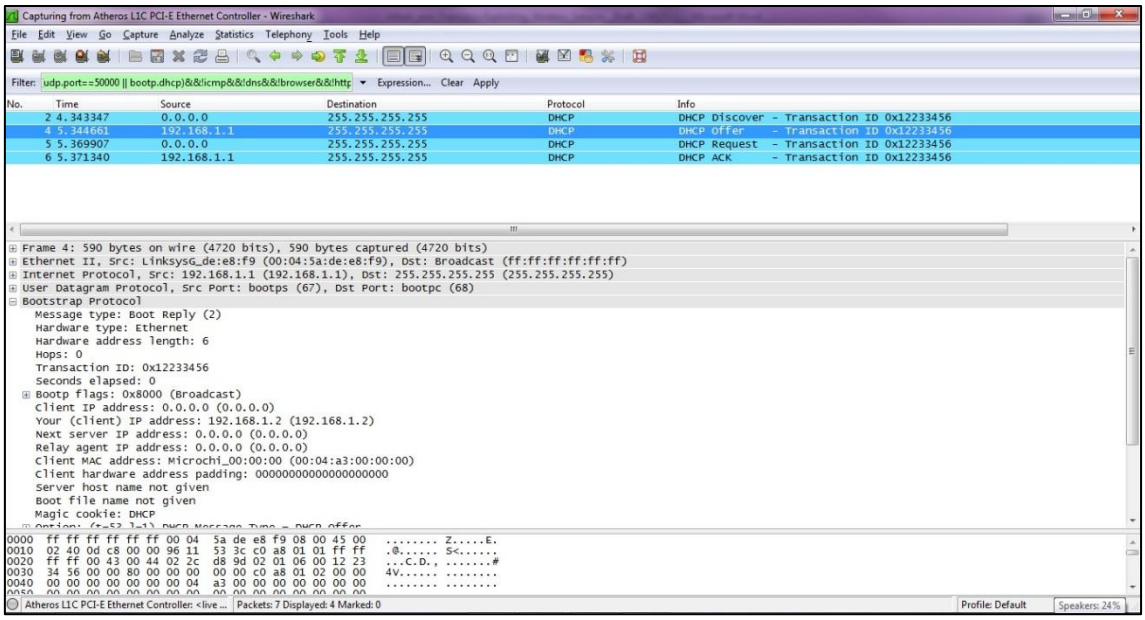


Figure 5-3: Analyzing the DHCP process – highlighting the DHCP ACK from the router

Once the gateway has an IP address we checked how the gateway responds when receives an ICMP Request or an ARP Request. For example, we investigated what happens when we execute a *ping* command on the PC directed to the gateway. By default, in Windows the command *ping* sends four ICMP requests. We observe in Figure 5-4 the ARP Request performed for the PC (which has an Asustek Ethernet MAC address) in order to learn the MAC address of the target. The gateway replies with its MAC address (in this case Microchip_00:00:00). Then the ICMP process occurs with its respective requests and replies.

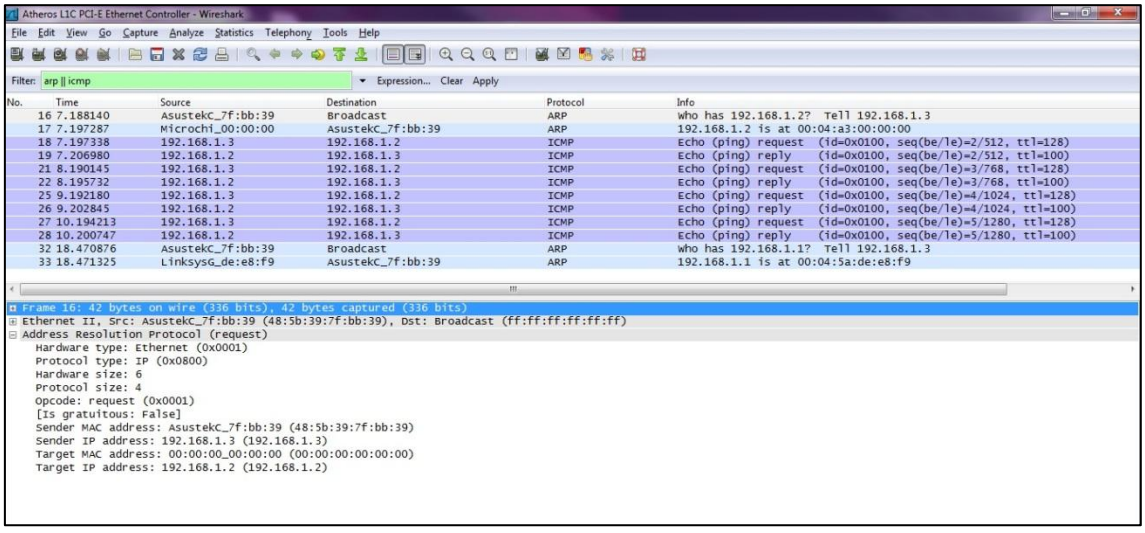


Figure 5-4: Analyzing ICMP and ARP processes

From this simple experiment we can see that the time it takes the gateway to respond to the ARP request is 9.147 ms. The observed median time to respond to a ICMP echo request is 8.088 ms with a standard deviation of 2.431 ms. Figure 5-5 shows an experiment with 100 ICMP request/replies to get a better estimate for these numbers.

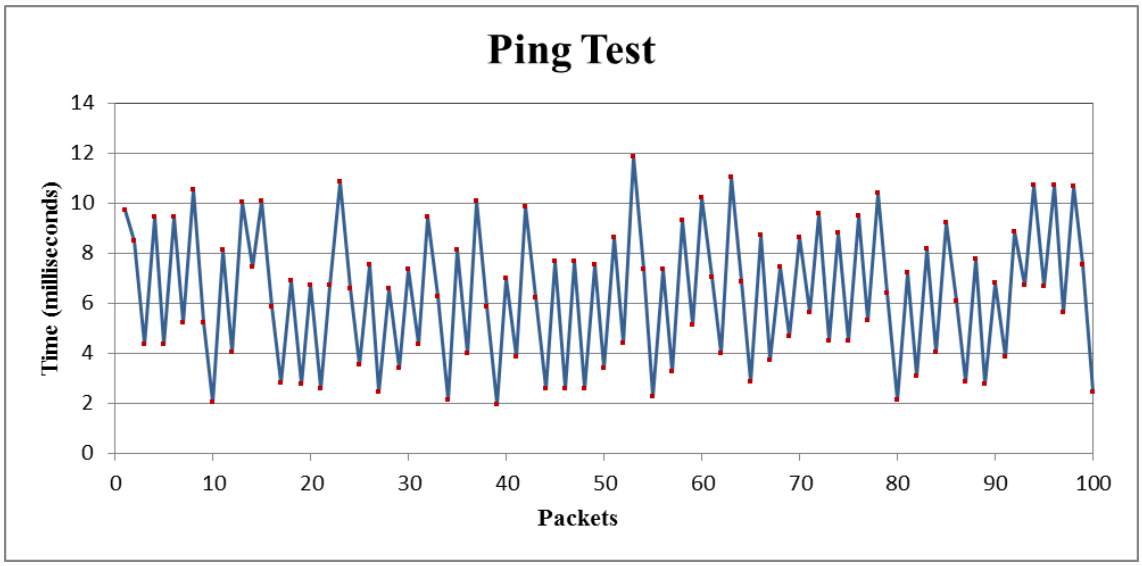


Figure 5-5: Ping test (100 packets)

In Table 5-1 we can observe different statistics of the experiment. The standard deviation measure should not be too high. The time within which the CPU processes a packet should be the same (or nearly the same) for each packet. In our case it is not true. After debugging the microprocessor we figured out the following: during the time within which the CPU is processing a packet it receives several interrupts incoming from the transceiver. These interrupts are generated (and processed) while the CPU is processing an Ethernet packet and they usually are generated in a shorter time (less than the time required for processing an Ethernet packet); hence before finishing processing an Ethernet packet, many interrupts are requested and processed. It should be fixed in order to reduce the deviation in the time of processing an Ethernet packet.

Table 5-1: Statistics of ping test (100 packets)

Minimum time	1.952 ms
Maximum time	11.851 ms
Average time	6.372 ms
Median time	6.669 ms
Standard deviation	2.709 ms

5.3 System: evaluation

Finally we tested all the features analyzed in the previous sections operating at the same time. The most important requirement is that the gateway replies to requests to forward sniffed wireless frames as UDP packets. This is done by the *UDP-SniffedPacket* protocol. It utilizes UDP with the ports 50000 for the server and 50001 for the client respectively. It is based on a request packet being sent from any host in the network to request a reply from the gateway with the latest sensor value. The request packet contains only one byte as the payload of the UDP datagram. This is considered a command byte. Currently we have only implemented one command (with the value 0x01) to request the entire sniffed wireless sensor packet. When the gateway receives this request (incoming on UDP port 50001), it replies with an UDP packet whose payload contains the following:

- 1-byte reply command (value 0x02),
- 1-byte length of the data that follows, and
- n-bytes with the whole sniffed wireless sensor frame.

The sensor we are using transmits 5 bytes of data (not including the preamble or synchronization word), hence the total length of the payload for the gateway's reply will be of 7 bytes. Figure 5-6 shows the request and Figure 5-7 shows the reply containing the sniffed wireless sensor's data. In this case, the payload of the reply is 0x02059906376a97. The sniffed packet is 0x9906376a97. Taking a look to Chapter 4, section 4.1, we can deduce that the temperature value is 23.7 ºC.

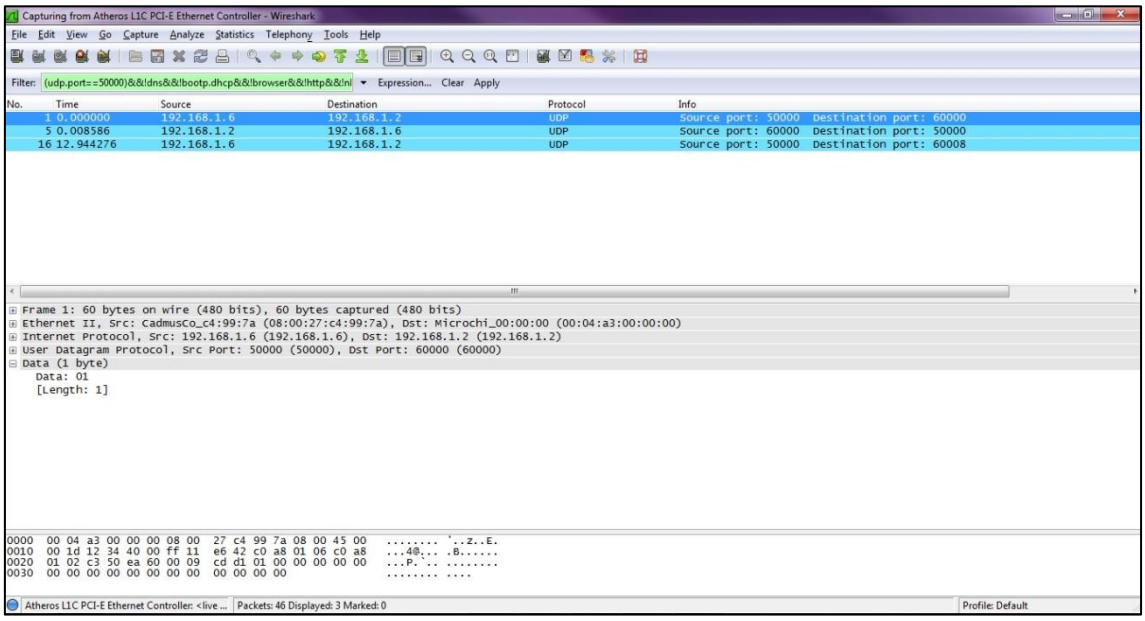


Figure 5-6: Request for sniffed wireless sensor's data

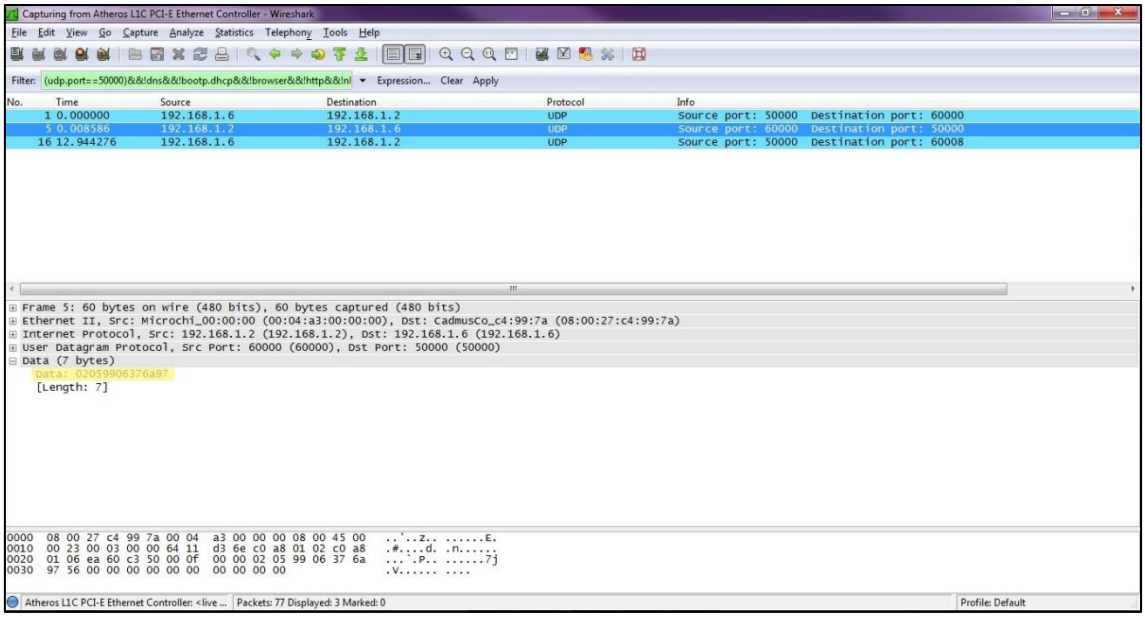


Figure 5-7: Reply with the sniffed wireless sensor's data

In these experiments the UDP packets from the PC to the gateway have been generated using the software PacketETH available from [88]. Figure 5-8 shows the interface where the user can generate the packet.

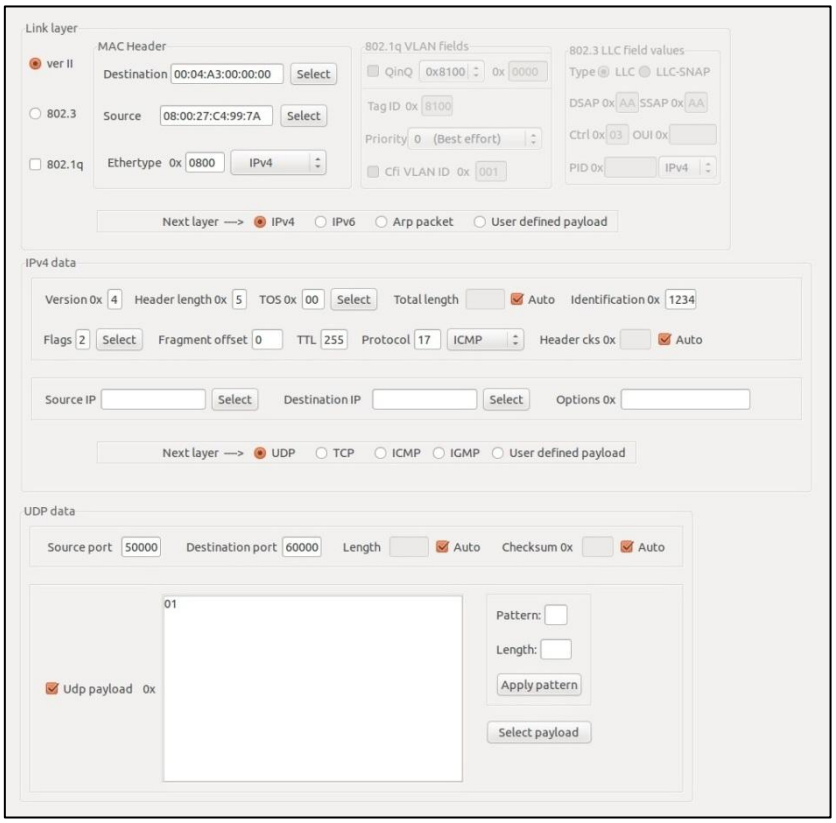


Figure 5-8: PackETH: packet generation

After performing these tests and evaluations, we observe that the gateway properly responds (in 7.127 ms), hence it meets the original project requirements. Multiple requests are showed in Figure 5-9. For each request the gateway replies with the last sniffed packet.

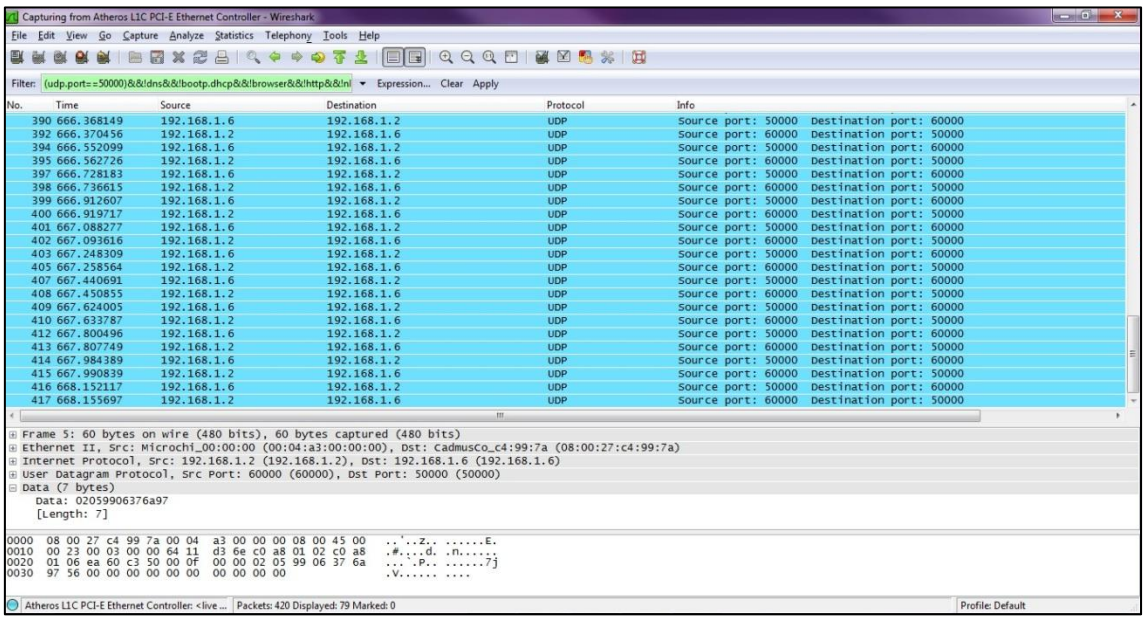


Figure 5-9: Analysis of multiple frames

5.4 Power over Ethernet: evaluation

One of the objectives of this project was that the gateway should be able to be powered using power over Ethernet (PoE) technology. The board was designed so that it would be manually configured to be powered from either PoE or an external DC power supply. The board is indeed capable of being power by either means (as selected by two jumpers on the board). For testing purposes the system has been tested with a TP-LINK model TL-POE150S PoE power injector. The PoE subsystem of the board is capable of powering not only the board and the daughterboard, but could even power other devices – as the current design is able to support a total load of 3.84 W (Class 1).

The final gateway with daughterboard is shown in Figure 5-10 being powered using the PoE power injector.

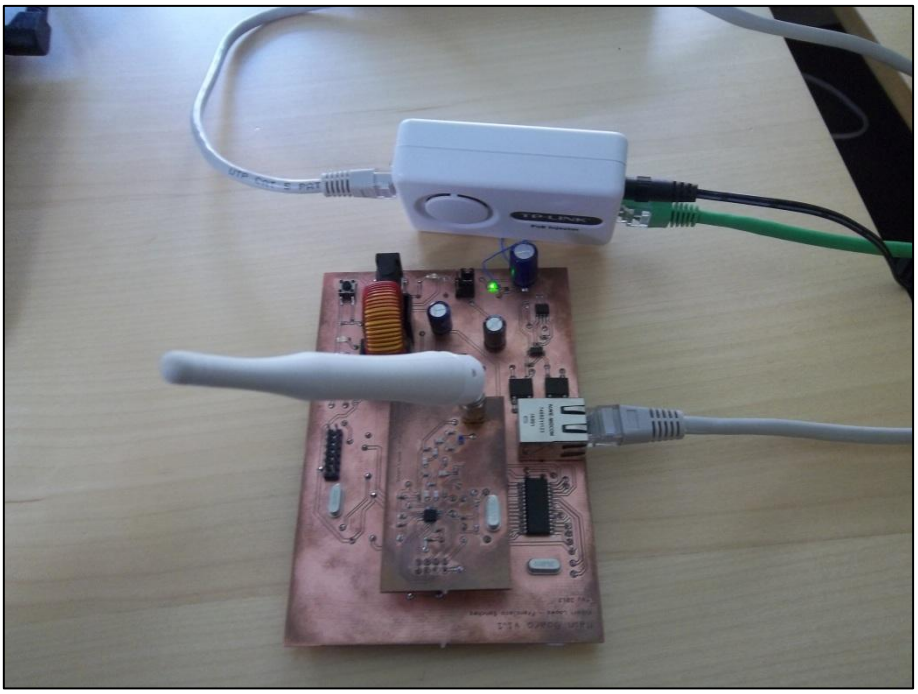


Figure 5-10: PoE injector and gateway

6 Conclusions

This chapter explains the conclusions obtained throughout the design, development and evaluation described in this thesis and proposes a number of improvements or complements that may be of interest in order to continue this work.

6.1 Discussion of the results

In this section we will state the conclusions and we will analyze the achieved goals proposed in the beginning of this thesis (Chapter 1, section 1.3).

We explained that our main goal was to build a gateway capable of sniff wireless sensor traffic in order to collect sensor data from existing sensors. This temperature, humidity, etc. sensors are frequently installed in homes and other locations, but are generally not connected to devices that are connected to the network. The gateway we designed along this project allows working with all kind of wireless sensors working at 868 MHz (915 MHz in USA). Whether you wish to work with sensors running in another frequency range it is only needed redesign the *daughterboard* modifying the external components following the manufacture datasheet.

The initial difficulty and the first big problem was that these sensors typically communicate using proprietary protocols that are not known by users. Although we could sniff wireless transmissions it would be impossible to know exactly the contents. However, as we demonstrated in section 4.1, it is possible to decode the communications. Knowing only the approximate working frequency, we were able to detect transmissions between the temperature transmitter and the TFA display. We assumed that these short range devices did not utilize complex circuitry and they would use simple forms of modulation. After examining the sniffed transmissions we realized that the modulation used was FSK. This suggests that probably most of these types of sensors use the same modulation or another equally simple scheme (i.e., Amplitude Shift Keying (ASK)).

Once we discovered the contents of these frames they could be processed. We did this in two ways. We considered whether this processing should be done in the gateway itself or perhaps in the final host. This later alternative would reduce the amount of work that the MCU has to support, hence reducing the energy consumption of the gateway, but would increase the amount of traffic from this gateway via the fixed network. Because the gateway can be powered by PoE decoding the frames remotely was not necessary. The final choice is reserved to the programmer.

We did some tests as explained in Chapter 5. These experiments demonstrated that the gateway successfully performs all the tasks related with the radio interface **and** the Ethernet interface. This last part requires a special mention. We have been able to build an autonomous system capable of connecting to a network with automatic configuration and operation. Additionally, the PoE subsystem can power the board and daughterboard.

Finally, all the results exposed above reveal that we have succeeded in achieving the main goal of this thesis project.

6.2 Future work

There have been several aspects that have been left out of scope of this thesis. Although sufficiently complete for the purpose of our study there are several

improvements that could be made to the current implementation in order to extend its functionality or applicability. This section enumerates some of these features that have been left out or work that has been left undone, together with some related work that may be of interest for future thesis projects.

- Adapt the gateway for IPv6. Our initial implementation only implements an IPv4 protocol stack. Although this is the most common IP protocol, IPv6 is becoming quite popular in home networks. Furthermore, according with *The Internet of the things* [6], in a short time *things* (objects) are expected to become active and capable of interacting and communicating among themselves and with the environment. Some sources suggest that each object should have its own IPv6 address. Converting our gateway to work via IPv6 will only require a small adjustment of the software in the MCU.
- There is no security implemented for the Ethernet interface. This was adequate for our testing since we performed our experiments in a private network with only our gateway, one router, and one PC. If this gateway is to be connected to a larger network where the authenticity of hosts is not guaranteed, then the gateway should implement IPv6's SEcure Neighbor Discover (SEND) [89] protocol.
- We only worked with one sensor. The gateway is capable of working with different types of sensors working at 868 MHz (if the frequency is different it is necessary to change the *daughterboard*) with the appropriate software. It would be interesting to test the gateway with different sensors in the 868 MHz band.
- We initially designed our gateway to sniff and receive wireless packets, but no transmit. However, during the course of this thesis project we have done some experiments to transmit a fake temperature value to the TFA display. The first packet received by the TFA was properly displayed, but the rest of the frames were lost. We suggest that this problem be investigated in order to enable the gateway to transmit to the TFA (or other) displays. In addition, the transmitter could be used with sensors that can be remotely controlled (this would be a long term objective).
- As we mentioned in section 3.2, our gateway is composed of two PCBs. One of them, called the *daughterboard*, incorporates the radio frequency part of the gateway with the transceiver, the antenna, and some external components. We did that because to facilitate adapting the gateway to different wireless sensor networks that might use a different working frequency. This is facilitated adaptation of the hardware. It would be interesting if the software could be remotely updated. For example, if the software related to the radio frequency part could be downloaded from a server. This server could contain several different programs already compiled and ready to be loaded into the MCU. Each one of these programs would be associated with different wireless sensor s. Once the gateway is connected to the IP network, it would request the appropriate program from the server, download it, and load it onto the flash memory of the MCU. The process of downloading this "program file" could be done using the Trivial Transfer File Protocol (TFTP) [90], by using HTTP, or some other protocol.
- The MSP430, the Ethernet controller, and the CC1101 provide low power modes of operation. We have thus far only implemented low power modes

on the MCU, but not in the other devices. Making use of the low power modes for the other subsystems can be done with slight modifications to the existing software.

6.3 Required reflections

During the course of this project we considered the economic and environmental aspects of this research by designing a low **cost** and low **power** gateway that can exploit **existing** sensors. As a result making these existing sensors more value and extending the applications that can utilize this *existing* data. As noted earlier in the thesis the data from these sensors could potentially be used to reduce the energy consumption of a home or other business, while providing a comfortable working environment when people are present; however, details of implementing such a solution are not part of this thesis – but have been considered in other work on context-awareness. The gateway has been developed with parts that are in compliance with the EU's Restriction of Use of Hazardous Substances (RoHS) regulations, although a small amount of lead-tin solder has been used to attach parts to the circuit boards. The circuit boards were mechanically milled out, rather than being photoetched – so there was no liquid chemical waste produced. The issues associated with mass production of such a gateway are outside the scope of this thesis.

We have not encountered any significant ethical issues when carrying out this thesis project. The data that is being sniffed contains temperature measurements made by wireless sensors, rather than human communication. At present only the capability of providing gateway functions has been demonstrated, the use of such a gateway or multiple gateways together with sensor fusion could have the potential to violate the expected privacy of individuals, but this work lies outside the scope of this thesis project.

References

[1] D. J. Cook and S. K. Das, Smart environments: technologies, protocols, and applications, vol. 43. Wiley-Interscience, 2005.

[2] 'IEEE 802.11, The Working Group Setting the Standards for Wireless LANs'. [Online]. Available: http://www.ieee802.org/11/. [Accessed: 23-November-2011].

[3] 'Federal Standard 1037C: Glossary of Telecommunications Terms'. [Online]. Available: http://www.its.bldrdoc.gov/fs-1037/fs-1037c.htm. [Accessed: 09-March-2012].

[4] 'IEEE 802.15.4'. [Online]. Available: http://www.ieee802.org/15/pub/TG4.html. [Accessed: 21-October-2011].

[5] IEEE Computer Society. LAN/MAN Standards Committee, Institute of Electrical and Electronics Engineers, IEEE-SA Standards Board, and American National Standards Institute, *IEEE standard for information technology telecommunications and information exchange between systems-- local and metropolitan area networks-- specific requirements. Part 15.1, Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications for Wireless Personal Area Networks (WPANs)*. New York, N.Y.: Institute of Electrical and Electronics Engineers, 2005, ISBN: 0738147079 9780738147079 0738147087 9780738147086, Available at http://ieeexplore.ieee.org/servlet/opac?punumber=9980.

[6] J. M. Bohli, C. Sorge, and D. Westhoff, 'Initial observations on economics, pricing, and penetration of the internet of things market', *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 2, pp. 50–55, 2009.

[7] 'ZigBee Alliance > Home'. [Online]. Available: http://www.zigbee.org/. [Accessed: 21-October-2011].

[8] A. Wheeler, 'Commercial Applications of Wireless Sensor Networks Using ZigBee', *IEEE Communications Magazine*, vol. 45, no. 4, pp. 70–77, April 2007, DOI:10.1109/MCOM.2007.343615.

[9] MEMSIC, 'MICAz Wireless Measurment System'. [Online]. Available: http://www.memsic.com/support/documentation/wireless-sensor-networks/category/7-datasheets.html?download=147%3Amica2. [Accessed: 24-November-2011].

[10] MEMSIC, 'TelosB Mote Platform'. [Online]. Available: http://www.memsic.com/support/documentation/wireless-sensor-networks/category/7-datasheets.html?download=152%3Atelosb.

[11] MEMSIC, 'IRIS Wireless Measurement System'. [Online]. Available: www.memsic.com/support/documentation/wireless-sensor-networks/category/7-datasheets.html?download=135%3Airis. [Accessed: 29-November-2011].

[12] ‘SimpliciTI™ - RF software protocol’. [Online]. Available: http://www.ti.com/corp/docs/landing/simpliciTI/index.htm?DCMP=hpa_rf_general&HQS=NotApplicable+OT+simpliciti. [Accessed: 21-October-2011].

[13] ‘Wireless Solutions’. [Online]. Available: http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=2664&param=en520414. [Accessed: 21-October-2011].

[14] ‘SynkroRF’. [Online]. Available: http://www.freescale.com/webapp/sps/site/overview.jsp?code=PROTOCOL_SYNKRO. [Accessed: 21-October-2011].

[15] ‘San Juan Software - PopNet™ The Easy, Economical Wireless Sensor and Control Network’. [Online]. Available: http://www.sanjuansw.com/?p=15. [Accessed: 21-October-2011].

[16] ‘Z-Wave.com - ZwaveStart’. [Online]. Available: http://www.z-wave.com/modules/ZwaveStart/. [Accessed: 24-November-2011].

[17] ‘Everything One-Net: one-net.info’. [Online]. Available: http://www.one-net.info/. [Accessed: 01-November-2011].

[18] ‘This is ANT, the Wireless Sensor Network Solution’. [Online]. Available: http://www.thisisant.com/. [Accessed: 22-November-2011].

[19] ‘Home | dash7.org’. [Online]. Available: http://www.dash7.org/. [Accessed: 24-November-2011].

[20] ‘HART Communication Protocol - Wireless HART Technology’. [Online]. Available: http://www.hartcomm.org/protocol/wihart/wireless_technology.html. [Accessed: 22-November-2011].

[21] ‘Home | ISA’. [Online]. Available: http://www.isa.org//MSTemplate.cfm?MicrositeID=1134&CommitteeID=6891. [Accessed: 22-November-2011].

[22] ‘TinyOS Home Page’. [Online]. Available: http://www.tinyos.net/. [Accessed: 21-October-2011].

[23] ‘nesC: A Programming Language for Deeply Networked Systems’. [Online]. Available: http://nescc.sourceforge.net/. [Accessed: 23-November-2011].

[24] ‘The Contiki OS’. [Online]. Available: http://www.contiki-os.org/. [Accessed: 22-November-2011].

[25] ‘Radiocommunication Sector (ITU-R) - ITU-R Home’. [Online]. Available: http://www.itu.int/ITU-R/index.asp?category=information&rlink=rhome&lang=en. [Accessed: 23-November-2011].

[26] ‘GSM World Coverage Map- GSM Country List by frequency bands’. [Online]. Available: http://worldtimezone.net/gsm.html. [Accessed: 23-November-2011].

[27] ‘CEPT.ORG’. [Online]. Available: http://www.cept.org/. [Accessed: 23-November-2011].

[28] ERC Recommendation 70-03, ‘Relating to the use of short range devices (SRD)’. Available at http://www.erodocdb.dk/docs/doc98/official/pdf/rec7003e.pdf, [accessed November 29, 2011].

[29] ‘ETSI’. [Online]. Available: http://www.etsi.org/WebSite/homepage.aspx. [Accessed: 23-November-2011].

[30] ETSI, ‘ETSI EN 300 220-2 v2.3.1’. Available at http://www.rfm.com/company/etsi.pdf, [accessed November 1, 2011].

[31] R. Braden, ‘Requirements for Internet Hosts - Communication Layers’. [Online]. Available: http://tools.ietf.org/html/rfc1122. [Accessed: 17-May-2012].

[32] I.-I. O. for Standardization, ‘ISO - International Organization for Standardization’, Available at http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=20269, [accessed May 17, 2012].

[33] Institute of Electrical and Electronics Engineers, IEEE standard for information technology telecommunications and information exchange between systems : local and metropolitan area networks : specific requirements. Part 3, Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications : section five. New York: Institute of Electrical and Electronics Engineers, 2008, ISBN: 9730738157979 9730738157962, Available at http://ieeexplore.ieee.org/servlet/opac?punumber=4726157.

[34] V. Cerf and R. Kahn, ‘A Protocol for Packet Network Intercommunication’, *Communications, IEEE Transactions on*, vol. 22, no. 5, pp. 637 – 648, May 1974, DOI:10.1109/TCOM.1974.1092259.

[35] J. Postel, ‘Internet Protocol’. [Online]. Available: http://tools.ietf.org/html/rfc791. [Accessed: 22-May-2012].

[36] K. Nichols, D. L. Black, S. Blake, and F. Baker, ‘Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers’. [Online]. Available: http://tools.ietf.org/html/rfc2474. [Accessed: 22-May-2012].

[37] J. Postel, ‘User Datagram Protocol’. [Online]. Available: http://tools.ietf.org/html/rfc768. [Accessed: 24-May-2012].

[38] W. R. Stevens, *TCP/IP illustrated. 1, The protocols*. Reading, Massachusetts [etc.]: Addison-Wesley, 2000, ISBN: 0201633469 9780201633467.

[39] D. Plummer, ‘Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware’. [Online]. Available: http://tools.ietf.org/html/rfc826. [Accessed: 22-May-2012].

[40] J. Postel, ‘Internet Control Message Protocol’. [Online]. Available: http://tools.ietf.org/html/rfc792. [Accessed: 23-May-2012].

[41] J. Gilmore and W. J. Croft, ‘Bootstrap Protocol’. [Online]. Available: http://tools.ietf.org/html/rfc951. [Accessed: 22-May-2012].

[42] R. Droms, ‘Dynamic Host Configuration Protocol’. [Online]. Available: http://tools.ietf.org/html/rfc2131. [Accessed: 22-May-2012].

[43] W. Wimer, ‘Clarifications and Extensions for the Bootstrap Protocol’. [Online]. Available: http://tools.ietf.org/html/rfc1542. [Accessed: 22-May-2012].

[44] ‘IEEE 802.3 ETHERNET’. [Online]. Available: http://ieee802.org/3/. [Accessed: 24-November-2011].

[45] ‘PSTN’, *About.com Wireless / Networking*, Available at http://compnetworking.about.com/od/voipvoiceoverip/g/bldef_pstn.htm, [accessed March 9, 2012].

[46] D. Harrington, B. Wijnen, and R. Presuhn, ‘An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks’. [Online]. Available: http://tools.ietf.org/html/rfc3411. [Accessed: 09-March-2012].

[47] M. Gebski, A. Penev, and R. K. Wong, ‘Protocol Identification of Encrypted Network Traffic’, in *Web Intelligence, IEEE / WIC / ACM International Conference on*, Los Alamitos, CA, USA, 2006, pp. 957–960, DOI:http://doi.ieeecomputersociety.org/10.1109/WI.2006.139.

[48] K. Gopalratnam, S. Basu, J. Dunagan, and H. J.Wang, ‘Automatically Extracting Fields from Unknown Network Protocols’, Available at http://research.microsoft.com/pubs/69364/sysml_114_cameraready.pdf, [accessed November 1, 2011].

[49] Yong-Sik Choi, Young-Jun Jeon, and Sang-Hyun Park, ‘A study on sensor nodes attestation protocol in a Wireless Sensor Network’, in *2010 The 12th International Conference on Advanced Communication Technology (ICACT)*, 2010, vol. 1, pp. 574–579.

[50] T. Hådén, ‘IPv6 Home Automation’, KTH Royal Institute of Technology, School of Information and Communication Technology (ICT), Department of Communication Systems, TRITA-ICT-EX-2009, 28 June 2009, Available at http://web.it.kth.se/~maguire/DEGREE-PROJECT-REPORTS/090601-Thor_Haaden.pdf, [accessed November 24, 2011].

[51] R. Martín Sánchez, ‘Desarrollo de un sniffer para redes de sensores basadas en ZigBee’, Thesis, Telecommunications Engineering, specializing in Telematics, Universitat Politècnica de Catalunya, UPC. Castelldefels School of Telecommunications and Aerospace Engineering (EETAC), 18 March2011, Available at http://upcommons.upc.edu/pfc/bitstream/2099.1/11946/1/memoria.pdf, [accessed November 24, 2011].

[52] Zhuanghui Yu, Yongzhong Huang, Shaozhong Guo, Bei Zhou, and Hua Ren, ‘Extracting Information from Unknown Protocols On CampusNet’, in *First IEEE International*

Symposium on Information Technologies and Applications in Education, 2007. ISITAE '07, 2007, pp. 535–539, DOI:10.1109/ISITAE.2007.4409343.

[53] Cheng Zhang, Shuran Song, Canxi Li, and Tiansheng Hong, 'Long-distance data communication based on wireless communication technology', 2011, pp. 4045–4048, DOI:10.1109/AIMSEC.2011.6010054, Available at http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6010054.

[54] Hao Haohao and Xiong Junqiao, 'Design of wireless sensor networks for density of natural gas', 2011, pp. 141–143, DOI:10.1109/ICSSEM.2011.6081166, Available at http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6081166.

[55] Chen Xiao-long and Zhou Wen-hua, 'ENC28J60 Ethernet controller applied in the network's three-phase Electric Energy meter', in *2010 6th International Conference on Networked Computing (INC)*, 2010, pp. 1–4.

[56] Chaoli Zhou and Jianhua Shen, 'The design and realization of ZigBee—Wi-Fi wireless gatway', in *2011 International Conference on Electric Information and Control Engineering (ICEICE)*, 2011, pp. 1786–1790, DOI:10.1109/ICEICE.2011.5777502.

[57] 'RFM12B and Weather station transmitters | JeeLabs.net Forum'. [Online]. Available: http://forum.jeelabs.net/node/110. [Accessed: 01-December-2011].

[58] Hope Microelectronics, 'RFM12B Datasheet'. Available at http://www.hoperf.com/upload/rf/RFM12B.pdf, [accessed December 1, 2011].

[59] 'Welcome to La Crosse Technology'. [Online]. Available: http://www.lacrossetechnology.com/. [Accessed: 01-December-2011].

[60] B. da Silva Campos, J. J. P. C. Rodrigues, L. M. L. Oliveira, L. D. P. Mendes, E. F. Nakamura, and C. M. S. Figueiredo, 'Design and construction of a wireless sensor and actuator network gateway based on 6LoWPAN', 2011, pp. 1–4, DOI:10.1109/EUROCON.2011.5929390, Available at http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5929390.

[61] 'Funk-Themometer'. Available at http://tfa-dostmann.de/index.php?id=61, [accessed October 21, 2011].

[62] Linum Software GmbH, 'Alles über das DCF77 Signal'. [Online]. Available: http://www.dcf77.de/. [Accessed: 07-June-2012].

[63] '8595E Portable Spectrum Analyzer, 9 kHz to 6.5 GHz [Obsolete] | Agilent'. [Online]. Available: http://www.home.agilent.com/agilent/product.jspx?nid=-536902970.536881804.00&lc=eng&cc=SE. [Accessed: 29-November-2011].

[64] 'Ettus Research LLC | Home'. [Online]. Available: http://www.ettus.com/. [Accessed: 21-October-2011].

[65] Ettus Research, 'Universal Software Radio Peripheral'. Available at http://www.ettus.com/downloads/ettus_ds_usrp_v7.pdf, [accessed October 21, 2011].

[66] Ettus Research, ‘TX and RX Daughterboards’. Available at http://www.ettus.com/downloads/ettus_daughterboards.pdf, [accessed November 24, 2011].

[67] ‘GNU Radio’. [Online]. Available: http://gnuradio.org. [Accessed: 24-November-2011].

[68] ‘Python Programming Language – Official Website’. [Online]. Available: http://python.org/. [Accessed: 24-November-2011].

[69] ‘GNU Radio - GNURadioCompanion - gnuradio.org’. Available at http://gnuradio.org/redmine/projects/gnuradio/wiki/GNURadioCompanion, [accessed November 24, 2011].

[70] ‘Code Composer Studio (CCStudio) Integrated Development Environment (IDE) v5 - CCSTUDIO - TI Tool Folder’. [Online]. Available: http://www.ti.com/tool/ccstudio. [Accessed: 24-November-2011].

[71] L. Maqueda, ‘Neighbor Discovery Proxy-Gateway for 6LoWPAN-based Wireless Sensor Networks’, KTH Royal Institute of Technology, School of Information and Communication Technology (ICT), Department of Communication Systems, TRITA-ICT-EX-2011:221, December 2011 2012, Available at http://web.it.kth.se/~maguire/DEGREE-PROJECT-REPORTS/111221-Luis_Maqueda-with-cover.pdf.

[72] ‘CadSoft - Home of CadSoft EAGLE PCB Design Software- Best PCB Design Software’. [Online]. Available: http://www.cadsoftusa.com/. [Accessed: 07-June-2012].

[73] ‘MSP430 5xx/6xx 16-bit microcontrollers from Texas Instruments’. [Online]. Available: http://focus.ti.com/paramsearch/docs/parametricsearch.tsp?familyId=1615&sectionId=95&tabId=2229&family=mcu. [Accessed: 29-November-2011].

[74] Joaquín Juan Toledo, ‘Wireless Sensor Architecture for a Home Event Management System’.

[75] Texas Instruments, ‘MSP430f5437 Datasheet’. Available at http://www.ti.com/lit/ds/symlink/msp430f5437a.pdf, [accessed November 29, 2011].

[76] ‘Power Over Ethernet (PoE)/LAN Solutions - Powered Device - TPS2375 - TI.com’. [Online]. Available: http://www.ti.com/product/tps2375. [Accessed: 07-June-2012].

[77] Microchip, ‘ENC28J60 Datasheet’. Available at http://ww1.microchip.com/downloads/en/DeviceDoc/39662c.pdf, [accessed November 24, 2011].

[78] ‘Wasa Board Project’. [Online]. Available: http://web.it.kth.se/~msmith/wasa/wasa_board_project.html. [Accessed: 08-June-2012].

[79] Texas Instruments, ‘CC1101 Datasheet’. Available at http://www.ti.com/lit/ds/symlink/cc1101.pdf, [accessed November 24, 2011].

[80] Texas Instruments, ‘CC430F6137 Datasheet’. Available at http://www.ti.com/lit/ds/symlink/cc430f6137.pdf, [accessed November 29, 2011].

[81] F. Bossard, ‘Wireless Temperature Sensor: CRC’. [Online]. Available: http://fredboboss.free.fr/tx29/tx29_crc.php. [Accessed: 11-December-2011].

[82] La Crosse Technology, ‘TX29U-IT’. [Online]. Available: http://www.lacrossetechnology.com/tx29/index.php. [Accessed: 13-December-2011].

[83] La Crosse Technology, ‘WS-7014’. [Online]. Available: http://www.lacrossetechnology.com.au/WS7014.html. [Accessed: 13-December-2011].

[84] GHS Infotronic, ‘Online CRC Calculation’. [Online]. Available: http://ghsi.de/CRC/. [Accessed: 01-December-2011].

[85] ‘SmartRF Studio - SMARTRFTM-STUDIO - TI Software Folder’. [Online]. Available: http://www.ti.com/tool/smartrftm-studio. [Accessed: 09-June-2012].

[86] ‘Linksys BEFSR41’. [Online]. Available: http://homesupport.cisco.com/en-eu/support/routers/BEFSR41. [Accessed: 11-June-2012].

[87] ‘Wireshark · Go deep.’ [Online]. Available: http://www.wireshark.org/. [Accessed: 11-June-2012].

[88] ‘packETH - ethernet packet generator’, *packETH - Ethernet packet generator*. [Online]. Available: http://packeth.sourceforge.net/. [Accessed: 11-June-2012].

[89] J. Kempf, J. Arkko, P. Nikander, and B. Zill, ‘SEcure Neighbor Discovery (SEND)’. [Online]. Available: http://tools.ietf.org/html/rfc3971. [Accessed: 11-June-2012].

[90] K. Sollins, ‘The TFTP Protocol (Revision 2)’. [Online]. Available: http://tools.ietf.org/html/rfc1350. [Accessed: 08-June-2012].

A. Python scripts

The Python scripts of the application developed as part of this master thesis is publicly available in the following repository:

http://gateway868mhz.googlecode.com/svn/trunk/Python_scripts/

B. MATLAB scripts to decode FSK encoded signal

The MATLAB scripts of the application developed as part of this master thesis is publicly available in the following repository:

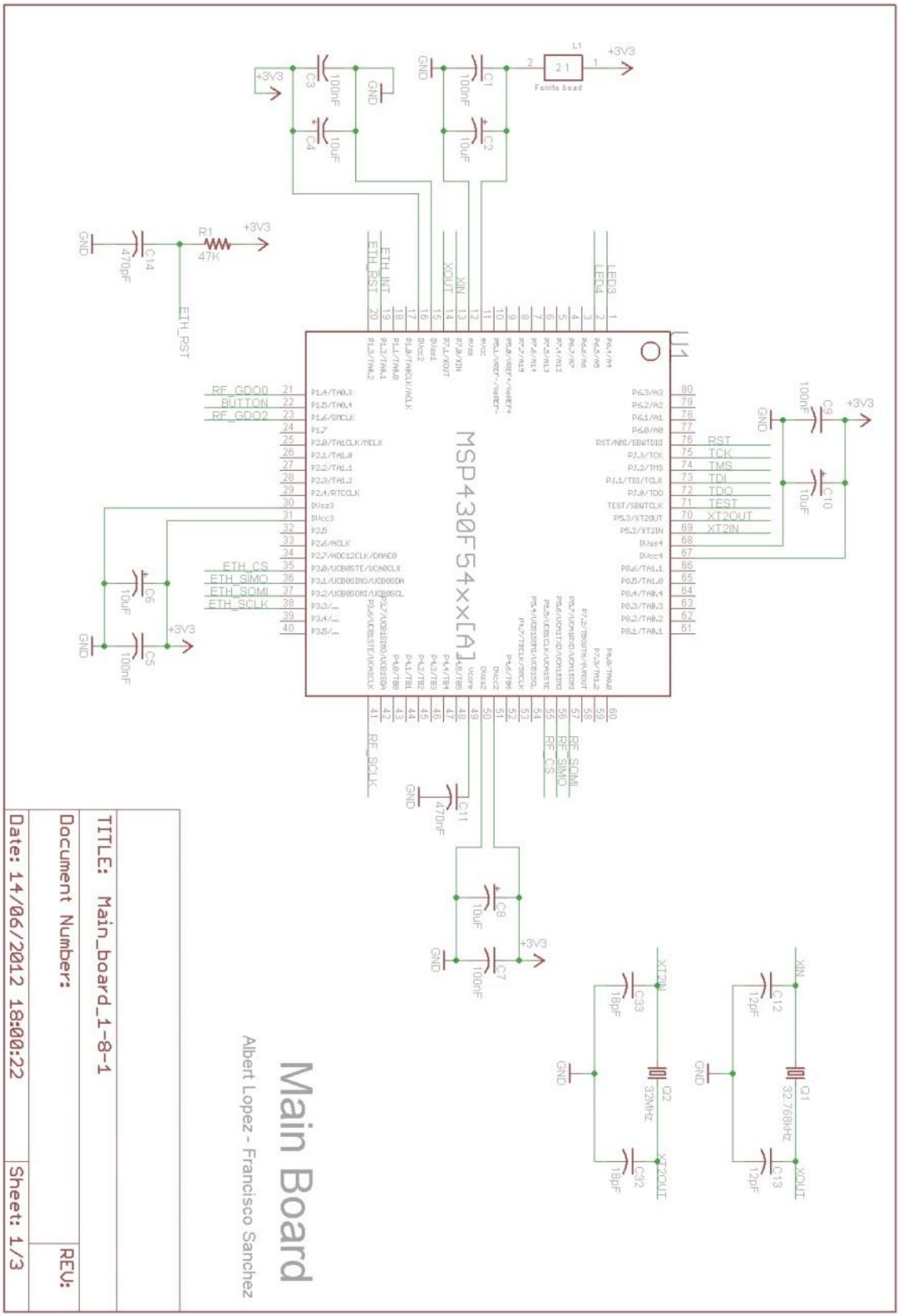
http://gateway868mhz.googlecode.com/svn/trunk/MATLAB_scripts/

C. Bit fields from the spreadsheet 'Temperatures.xls'

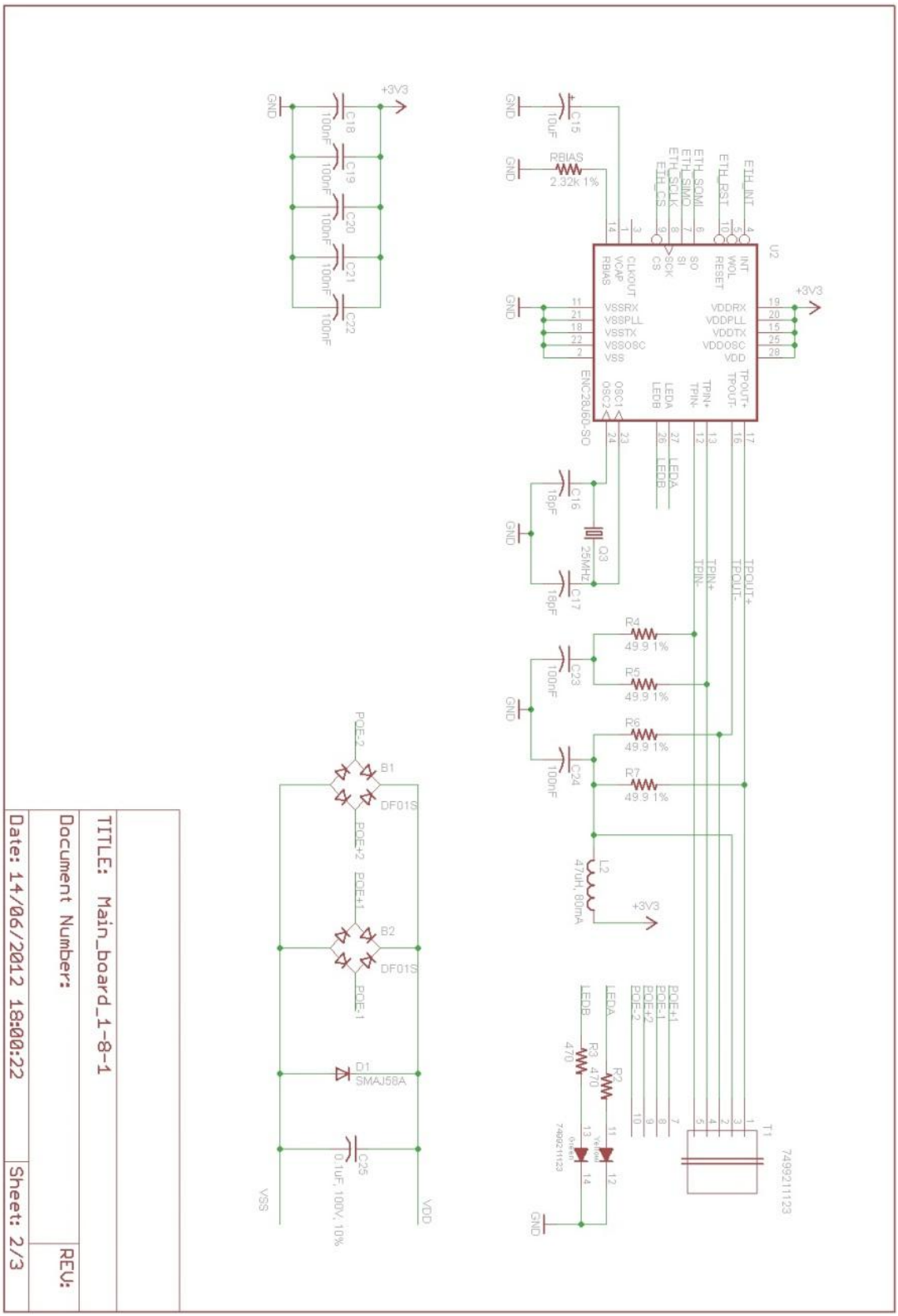
TEMP	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
-12,2	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	1	0	0	0	0	1	0	0	0	1	0	0	1	1	1	0	1	1	1	0	1	1	0	1	0	1	0	1	1	0	1	1	0	1	1
-11,8	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	1	0	0	0	0	1	0	0	0	1	0	1	0	0	0	0	0	0	1	0	1	1	0	1	0	1	0	1	0	0	1	1	0	0	1
-11,6	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	1	0	0	0	0	1	0	0	0	1	0	1	0	0	0	0	0	1	1	0	1	1	0	1	0	1	0	0	1	0	0	0	0	0	0
-11,4	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	1	0	0	0	0	1	0	0	0	1	0	1	0	0	0	0	1	0	1	0	1	1	0	1	0	1	0	0	0	0	1	1	0	1	0
-11,3	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	1	0	0	0	0	1	0	0	0	1	0	1	0	0	0	0	1	1	0	0	1	1	0	1	0	1	0	0	0	1	1	0	1	1	1
-11,0	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	1	0	0	0	0	1	0	0	0	1	0	1	0	0	0	1	0	0	1	0	1	1	0	1	0	1	0	1	0	1	0	1	1	1	0
-9,5	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	1	0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	1	0	0	0	1	1	0	1	0	1	0	1	0	0	0	1	0	1	1
-9,3	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	1	0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	1	1	0	0	1	1	0	1	0	1	0	0	1	0	1	0	0	1	0
-9,0	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	1	0	0	0	0	1	0	0	0	1	1	0	0	0	0	1	0	0	1	0	1	1	0	1	0	1	0	1	1	0	0	1	0	1	1
-8,7	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	1	0	0	0	0	1	0	0	0	1	1	0	0	0	1	0	0	1	0	0	1	1	0	1	0	1	0	1	0	1	1	1	1	1	1
-8,5	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	1	0	0	0	0	1	0	0	0	1	1	0	0	0	1	0	1	0	0	0	1	1	0	1	0	1	0	1	1	1	0	0	1	0	1
-8,1	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	1	0	0	0	0	1	0	0	0	1	1	0	0	0	1	1	0	0	0	0	1	1	0	1	0	1	0	0	1	0	1	0	0	0	1
0,4	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	1	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1	1	0	1	0	1	0	0	1	1	0	1	0	0	0
0,8	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	1	0	0	0	0	1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	1	0	1	0	1	0	1	1	0	1	1	1	0	0
1,1	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	1	0	0	0	0	1	0	0	1	0	0	0	0	0	1	0	0	0	1	0	1	1	0	1	0	1	0	0	1	1	1	0	0	0	1
1,6	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	1	0	0	0	0	1	0	0	1	0	0	0	0	0	1	0	1	1	0	0	1	1	0	1	0	1	0	1	1	0	1	1	1	1	1
2,1	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	1	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0	0	0	1	0	1	1	0	1	0	1	0	1	1	0	0	0	0	1	1
14,2	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	0	1	1	1	1	0	0	0	1	0	1	0	1	0	0	0	0	1	0	0	1	1	0	1	0	1	0	0	0	1	1	0	0	0	1
15,4	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	1	0	0	0	0	1	0	0	1	0	1	0	1	0	1	0	1	0	0	0	1	1	0	1	0	1	0	1	1	0	0	1	0	0	1
23,4	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	0	1	1	1	1	1	0	0	1	1	0	0	0	1	1	0	1	0	0	0	1	1	0	1	0	1	0	1	1	1	0	0	0	1	1
25,0	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	0	1	1	1	1	1	0	0	1	1	0	0	1	0	1	0	0	0	0	0	1	1	0	1	0	1	0	0	0	1	1	0	1	0	1
25,1	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	0	1	1	1	1	1	0	0	1	1	0	0	1	0	1	0	0	0	1	0	1	1	0	1	0	1	0	1	1	0	0	0	0	0	1
25,2	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	0	1	1	1	1	1	0	0	1	1	0	0	1	0	1	0	0	1	0	0	1	1	0	1	0	1	0	1	1	1	0	1	1	0	0
25,3	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	0	1	1	1	1	1	0	0	1	1	0	0	1	0	1	0	0	1	1	0	1	1	0	1	0	1	0	0	0	0	1	1	0	0	0
25,4	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	0	1	1	1	1	1	0	0	1	1	0	0	1	0	1	0	1	0	0	0	1	1	0	1	0	1	0	1	0	1	1	0	1	1	0
25,5	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	0	1	1	1	1	1	0	0	1	1	0	0	1	0	1	0	1	0	1	0	1	1	0	1	0	1	0	0	1	0	0	0	0	1	0
25,6	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	0	1	1	1	1	1	0	0	1	1	0	0	1	0	1	0	1	1	0	0	1	1	0	1	0	1	0	0	1	1	0	1	1	1	1
25,7	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	0	1	1	1	1	1	0	0	1	1	0	0	1	0	1	0	1	1	1	0	1	1	0	1	0	1	0	1	0	0	1	1	0	1	1
25,8	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	0	1	1	1	1	1	0	0	1	1	0	0	1	0	1	1	0	0	0	0	1	1	0	1	0	1	0	0	0	0	0	0	0	1	0
25,9	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	0	1	1	1	1	1	0	0	1	1	0	0	1	0	1	1	0	0	1	0	1	1	0	1	0	1	0	1	1	1	1	0	1	1	0
26,0	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	0	1	1	1	1	1	0	0	1	1	0	0	1	1	0	0	0	0	0	0	1	1	0	1	0	1	0	1	0	0	0	0	1	1	1
26,1	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	0	1	1	1	1	1	0	0	1	1	0	0	1	1	0	0	0	0	1	0	1	1	0	1	0	1	0	0	1	1	1	0	0	1	1
26,2	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	0	1	1	1	1	1	0	0	1	1	0	0	1	1	0	0	0	1	0	0	1	1	0	1	0	1	0	0	1	0	1	1	1	1	0
26,3	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	0	1	1	1	1	1	0	0	1	1	0	0	1	1	0	0	0	1	1	0	1	1	0	1	0	1	0	1	0	1	0	1	0	1	0
26,4	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	0	1	1	1	1	1	0	0	1	1	0	0	1	1	0	0	1	0	0	0	1	1	0	1	0	1	0	0	0	0	0	0	1	0	0
26,5	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	0	1	1	1	1	1	0	0	1	1	0	0	1	1	0	0	1	0	1	0	1	1	0	1	0	1	0	1	1	1	1	0	0	0	0
26,6	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	0	1	1	1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	1	0	1	0	1	1	0	1	1	1	0	1
26,7	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	0	1	1	1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	1	0	1	1	0	1	0	1	0	0	0	1	0	1	0	0	1
26,8	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	0	1	1	1	1	1	0	0	1	1	0	0	1	1	0	1	0	0	0	0	1	1	0	1	0	1	0	1	0	1	1	0	0	0	0
26,9	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	0	1	1	1	1	1	0	0	1	1	0	0	1	1	0	1	0	0	1	0	1	1	0	1	0	1	0	0	1	0	0	0	1	0	0
29,0	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	0	1	1	1	1	1	0	0	1	1	0	1	0	0	1	0	0	0	0	0	1	1	0	1	0	1	0	1	0	0	1	1	1	1	1
30,0	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	0	1	1	1	1	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	1	1	0	1	0	1	0	1	0	0	1	0	1	0	0
31,1	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	0	1	1	1	1	0	0	0	1	1	1	0	0	0	1	0	0	0	1	0	1	1	0	1	0	1	0	0	1	1	1	0	1	1	1
31,3	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	0	1	1	1	1	0	0	0	1	1	1	0	0	0	1	0	0	1	1	0	1	1	0	1	0	1	0	1	0	1	0	1	1	1	0
31,6	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	0	1	1	1	1	0	0	0	1	1	1	0	0	0	1	0	1	1	0	0	1	1	0	1	0	1	0	1	1	0	1	1	0	0	1
31,8	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	0	1	1	1	1	0	0	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	0	1	0	1	0	1	0	1	1	0	1	0	0
31,9	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	0	1	1	1	1	0	0	0	1	1	1	0	0	0	1	1	0	0	1	0	1	1	0	1	0	1	0	0	1	0	0	0	0	0	0
32,1	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	0	1	1	1	1	0	0	0	1	1	1	0	0	1	0	0	0	0	1	0	1	1	0	1	0	1	0	1	1	0	0	0	1	0	1
32,3	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	0	1	1	1	1	0	0	0	1	1	1	0	0	1	0	0	0	1	1	0	1	1	0	1	0	1	0	0	0	0	1	1	1	0	0
32,6	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	0	1	1	1	1	0	0	0	1	1	1	0	0	1	0	0	1	1	0	0	1	1	0	1	0	1	0	0	1	1	0	1	0	1	1
32,8	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	0	1	1	1	1	0	0	0	1	1	1	0	0	1	0	1	0	0	0	0	1	1	0	1	0	1	0	0	0	0	0	0	1	1	0
32,9	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	0	1	1	1	1	0	0	0	1	1	1	0	0	1	0	1	0	0	1	0	1	1	0	1	0	1	0	1	1	1	1	0	0	1	0
33,0	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	0	1	1	1	1	0	0	0	1	1	1	0	0	1	1	0	0	0	0	0	1	1	0	1	0	1	0	0	1	0	1	1	1	1	1
33,1	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	0	1	1	1	1	0	0	0	1	1	1	0	0	1	1	0	0	0	1	0	1	1	0	1	0	1	0	1	0	1	0	1	0	1	1
33,2	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	0	1	1	1	1	0	0	0	1	1	1	0	0	1	1	0	0	1	0	0	1	1	0	1	0	1	0	1	0	0	0	0	1	1	0
33,4	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	0	1	1	1	1	0	0	0	1	1	1	0	0	1	1	0	1	0	0	0	1	1	0	1	0	1	0	1	1	0	1	1	1	0	0
46,0	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	1	0	1	0	1	1	0	1	0	0	0	0	1	1	0	0	0	0	0	0	1	1	0	1	0	1	0	1	0	1	0	1	1	0	0
46,4	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	1	0	0	0	0	1	0	1	0	0	0	0	1	1	0	0	1	0	0	0	1	1	0	1	0	1	0	0	1	0	0	0	1	1	0
46,7	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	1	0	0	0	0	1	0	1	0	0	0	0	1	1	0	0	1	1	1	0	1	1	0	1	0	1	0	0	1	1	0	1	0	1	1
47,0	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	1	0	0	0	0	1	0	1	0	0	0	0	1	1	1	0	0	0	0	0	1	1	0	1	0	1	0	1	0	1	0	1	0	1	1
50,4	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	1	0	0	0	0	1	0	1	0	0	1	0	0	0	0	0	1	0	0	0	1	1	0	1	0	1	0	0	1	0	1	0	1	0	1
50,8	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	1	0	0	0	0	1	0	1	0	0	1	0	0	0	0	1	0	0	0	0	1	1	0	1	0	1	0	1	1	1	0	0	0	0	1
51,0	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	1	0	0	0	0	1	0	1	0	0	1	0	0	0	1	0	0	0	0	0	1	1	0	1	0	1	0	1	0	1	1	1	0	0	0
51,3	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	1	0	0	0	0	1	0	1	0	0	1	0	0	0	1	0	0	1	1	0	1	1	0	1	0	1	0	1	0	0	1	0	1	0	1
51,6	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	1	0	0	0	0	1	0	1	0	0	1	0	0	0	1	0	1	1	0	0	1	1	0	1	0	1	0	1	1	1	0	0	0	1	0
51,9	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	1	0	0	0	0	1	0	1	0	0	1	0	0	0	1	1	0	0	1	0	1	1	0	1	0	1	0	0	1	1	1	1	0	1	1
52,2	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	1	0	0	0	0	1	0	1	0	0	1	0	0	1	0	0	0	1	0	0	1	1	0	1	0	1	0	1	1	0	1	0	0	1	1
52,4	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	1	0	0	0	0	1	0	1	0	0	1	0	0	1	0	0	1	0	0	0	1	1	0	1	0	1	0	1	0	0	0	1	0	0	1
52,7	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	1	0	0	0	0	1	0	1	0	0	1	0	0	1	0	0	1	1	1	0	1	1	0	1	0	1	0	1	0	1	0	0	1	0	0
53,0	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	1	0	0	0	0	1	0	1	0	0	1	0	0	1	1	0	0	0	0	0	1	1	0	1	0	1	0	0	1	1	0	0	1	0	0
53,2	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	1	0	0	0	0	1	0	1	0	0	1	0	0	1	1	0	0	1	0	0	1	1	0	1	0	1	0	1	0	1	1	1	1	0	1
53,5	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	1	0	0	0	0	1	0	1	0	0	1	0	0	1	1	0	1	0	1	0	1	1	0	1	0	1	0	0	0	0	1	0	0	1	1
53,7	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	1	0	0	0	0	1	0	1	0	0	1	0	0	1	1	0	1	1	1	0	1	1	0	1	0	1	0	1	1	0	0	1	0	1	0
53,9	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	1	0	0	0	0	1	0	1	0	0	1	0	0	1	1	1	0	0	1	0	1	1	0	1	0	1	0	1	0	1	0	0	1	1	1
54,2	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	1	0	0	0	0	1	0	1	0	0	1	0	1	0	0	0	0	1	0	0	1	1	0	1	0	1	0	1	0	0	0	0	1	1	0
54,4	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	1	0	0	0	0	1	0	1	0	0	1	0	1	0	0	0	1	0	0	0	1	1	0	1	0	1	0	1	1	0	1	1	1	0	0
54,6	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	1	0	0	0	0	1	0	1	0	0	1	0	1	0	0	0	1	1	0	0	1	1	0	1	0	1	0	0	0	0	0	0	1	0	1
54,8	1	0	1	0	1	0	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1	1	0	0	0	0	1	0	1	0	0	1	0	1	0	0	1	0	0	0	0	1	1	0	1	0	1	0	0	1	1	0	1	0	0	0

D. Schematic of the motherboard and the daughterboard

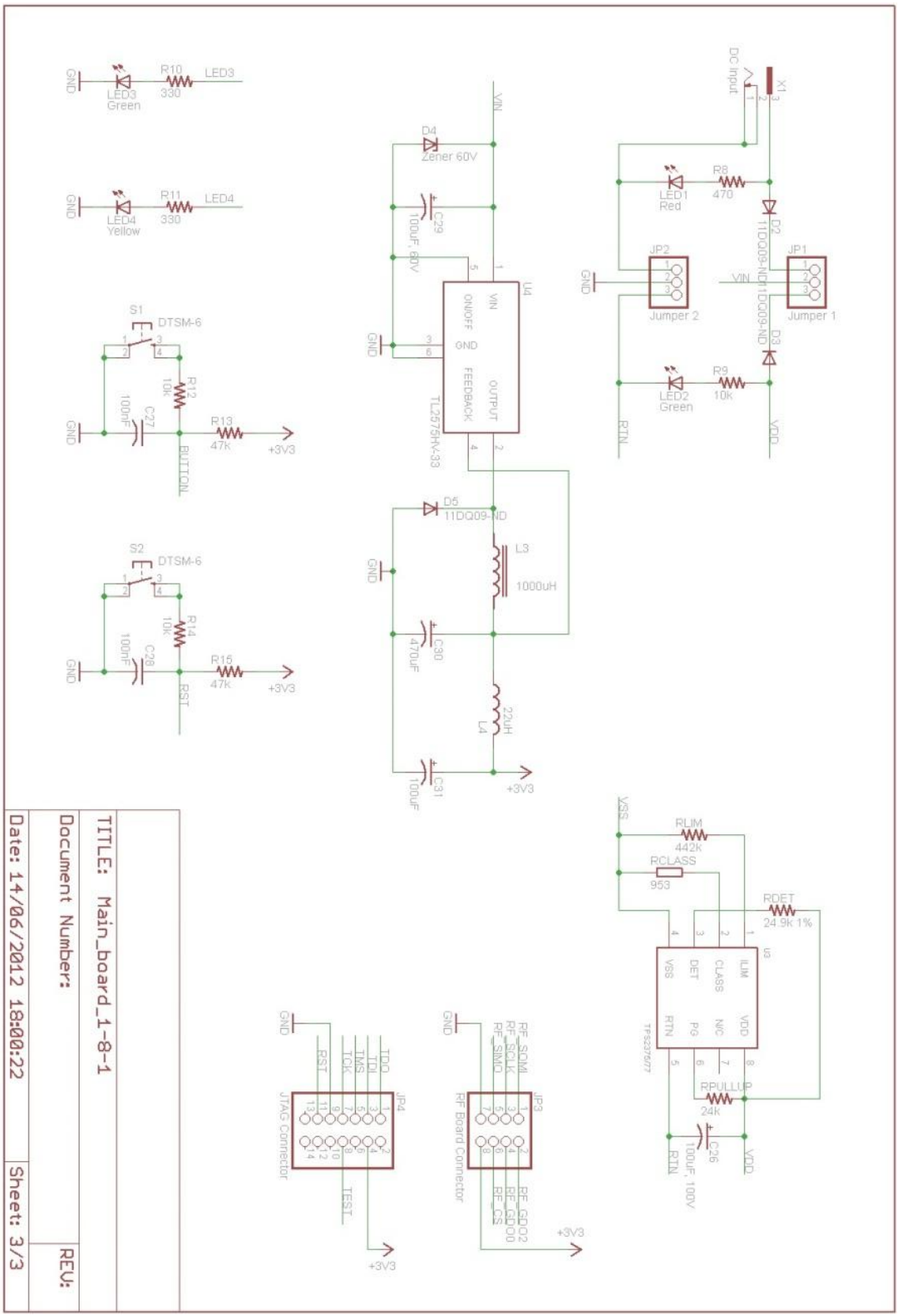
Motherboard (sheet 1):



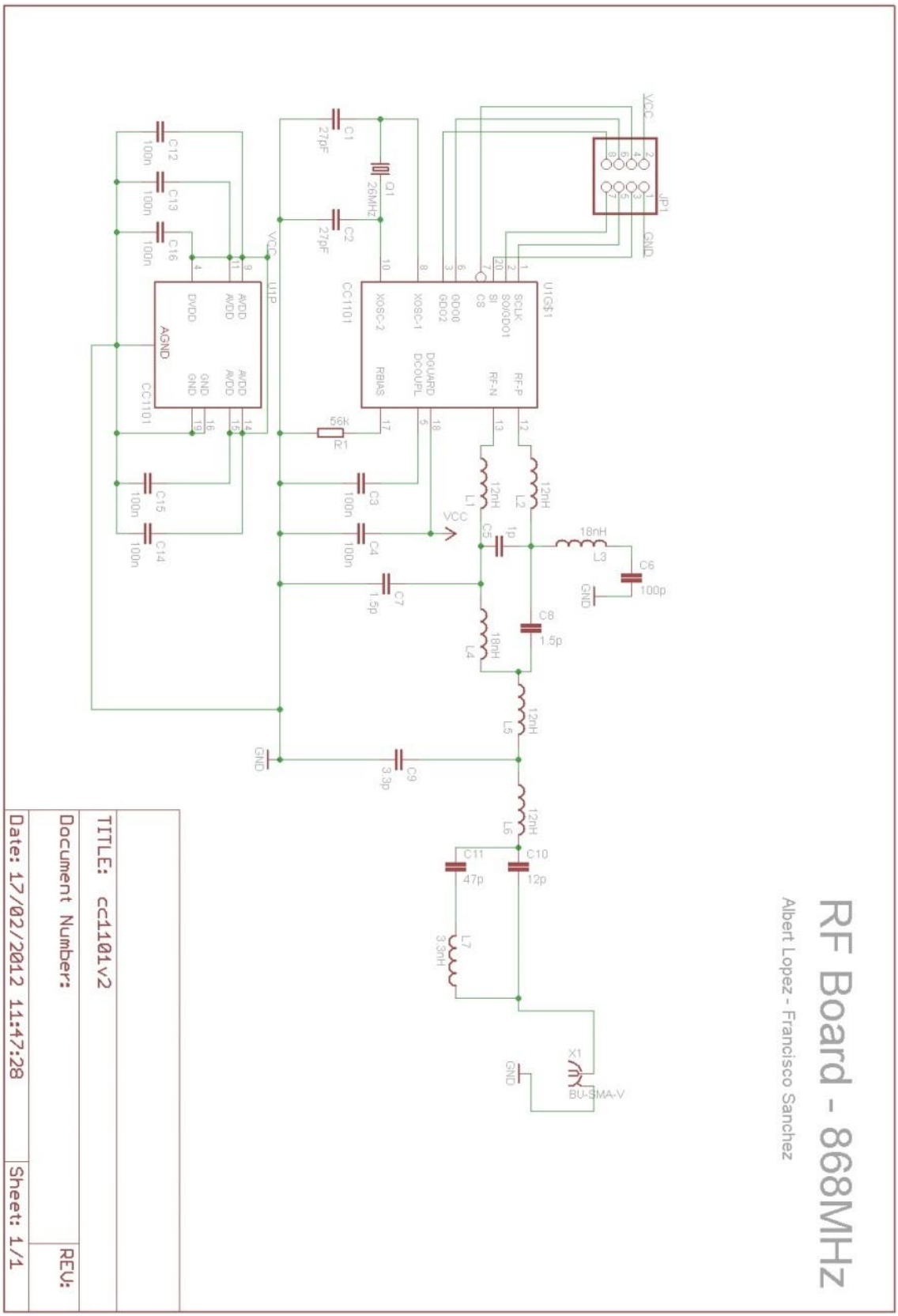
Motherboard (sheet 2):



Motherboard (sheet 3):

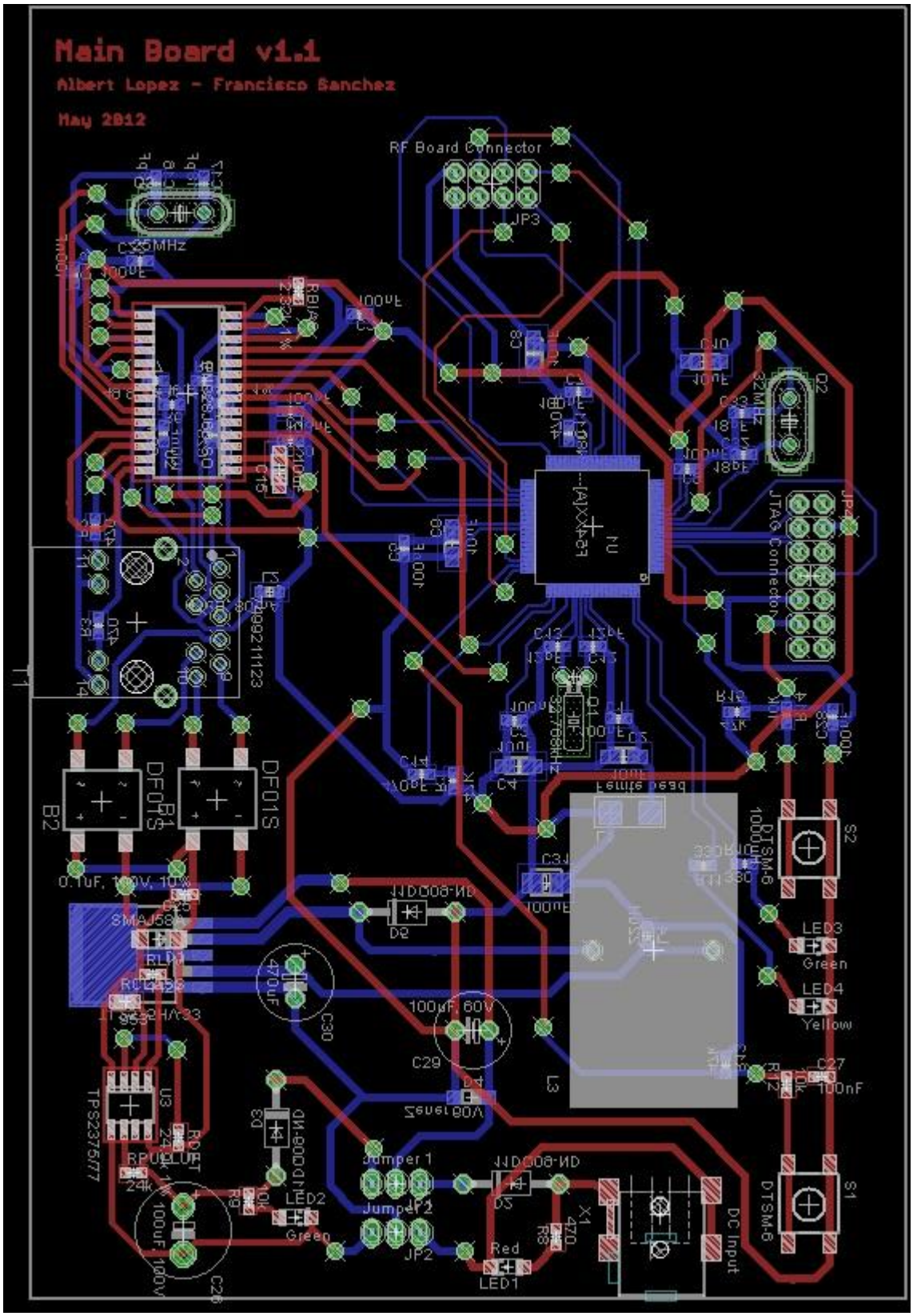


Daughterboard:

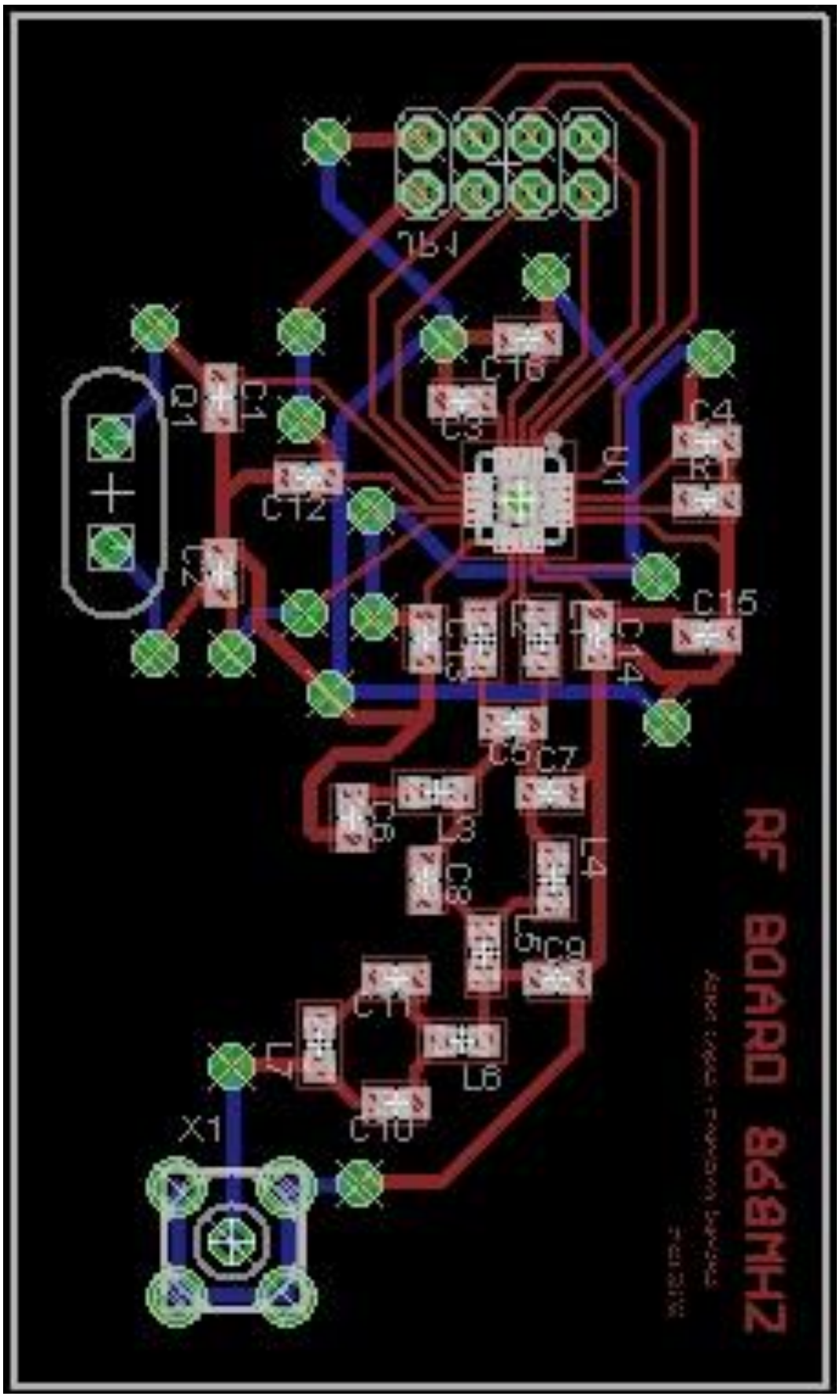


E. Layout of the motherboard and the daughterboard

Motherboard:



Daughterboard:



F. Source code for the gateway

The source code of the application developed as part of this master thesis is publicly available in the following repository:

http://gateway868mhz.googlecode.com/svn/trunk/GW868MHz/

TRITA-ICT-EX-2012:110

www.kth.se