# Development of Robust Wireless Sensor Network Communications for an Embedded Real Time System on Construction Sites

Master's Thesis

**Alexandru Cohal**

Matriculation number: 406284

Handover Date: 31.10.2018

European Master in Embedded Computing Systems (EMECS)
Fachbereich Elektrotechnik und Informationstechnik
Technische Universität Kaiserslautern, Deutschland

Examiners: Prof. Dr. Paul Lukowicz, Prof. Dr.-Ing. Wolfgang Kunz
Supervisor: Marco Hirsch

# Declaration of Academic Honest

Hereby I declare that the present thesis is drawn up after *MPO (Masterprüfungsordnung) Elektrotechnik und Informationstechnik* by myself without help of third parties but the support of my supervisor, that all used sources and tools including the internet are completely and exactly mentioned, and that everything is marked which is taken unchanged, shortened or analogous from other literature.

Kaiserslautern, 31.10.2018

# Acknowledgement

I would like to thank all the staff members of the *Embedded Intelligence* department within *DFKI* (*Deutsches Forschungszentrum für Künstliche Intelligenz* / *German Research Center for Artificial Intelligence*) for the provided help and guidance while developing this master's thesis.

I am very grateful to everyone who supported me, especially to my family.

# Abstract

The construction industry represents one of the most important sectors based on the number of workers and the amount of invested money. In order to take care of the employees, increase the profit and optimize the workflow, an *IoT*-based *Wireless Sensor Network* can be used for tracking the equipments, materials and workers and to monitor the statuses of the machines and tools, the health of the workers and the environment. A large number of *Sensor Nodes* can be deployed on a construction site in order to acquire different sensors' values and to send them to a *Gateway Node* using radio signals. The ultra-low power dual-band (*Sub-1 GHz* and *2.4 GHz*) wireless *CC1350* microcontroller produced by *Texas Instruments* can be used due to its advantages.

The scope of this thesis is to develop a *Communication Protocol Stack* for a *CC1350*-based *WSN* running on a Real-Time Operating System (*TI-RTOS*). Based on the targeted domain, several requirements had to be defined in the beginning, considering the general properties of a network and the capabilities of the used microcontroller, described thoroughly, and compared with the ones of the newer versions. Overviews and comparisons were realized for different communication protocols and standards used in *IoT*, application layer protocols, spectrum access methods and security solutions. A combination between the *IEEE 802.15.4 e/g* standard, *TI 15.4 Stack* and *MQTT-SN* application layer for the *Sub-1 GHz* and *BLE* for the *2.4 GHz* were considered to be the most suited core choices for the stack. Moreover, *Listen Before Talk* coupled with *Adaptive Frequency Analysis* was considered to be the most suited spectrum access method for the *Sub-1 GHz* due to the existing regulations. Encryption/decryption and authentication based on the *AES CCM 128 bit shared key* technique using the dedicated features of the *CC1350* microcontroller were chosen for enhancing the network's security.

Based on these choices, a complete *Communication Protocol Stack* was assembled and described and an architecture of the whole network was proposed. Laboratory and field tests were performed for a basic version of this stack, considering the use case of a highway construction site. The results obtained (i.e. the average number of received, lost and duplicate packets for different configurations) were analyzed, followed by a list of further task to be done in the future.

# Abstract

Die Bauwirtschaft ist eine der wichtigsten Branchen, basierend auf der Arbeiters Anzahl und der Höhe des investierten Geldes. Um sich um die Mitarbeiter zu kümmern, den Gewinn zu steigern und den Arbeitsablauf zu optimieren, kann ein *IoT*-basiertes *Wireless Sensor Network* verwendet werden, um die Ausrüstungen, Materialien und Arbeitskräfte zu verfolgen und um den Status der Maschinen und Werkzeuge sowie den Zustand der Geräte zu überwachen Arbeiter und die Umwelt. Eine große Anzahl von *Sensor Nodes* kann auf einer Baustelle eingesetzt werden, um die verschiedenen Sensorwerte zu erfassen und sie über Funksignale an einen *Gateway Node* zu senden. Der ultra-low power dual-band (*Sub-1 GHz* und *2.4 GHz*) wireless *CC1350 Texas Instruments* Mikrokontroller, kann aufgrund seiner Vorteile verwendet werden.
Das Ziel dieser Arbeit ist es, einen *Communication Protocol Stack* für ein *CC1350*-basiertes *WSN* zu entwickeln, das auf einem Real-Time Operating System (*TI-RTOS*) läuft. Basierend auf der Zieldomäne mussten anfangs mehrere Anforderungen definiert werden, wobei die allgemeinen Eigenschaften eines Netzwerks und die Fähigkeiten des verwendeten Mikrocontrollers sorgfältig beschrieben und mit denen der neueren Versionen verglichen wurden. Es wurden Übersichten und Vergleiche für verschiedene *IoT*-Kommunikationsprotokolle und -Standards, Protokollen der Anwendungsschicht, Spektrum-Zugriffsmethoden und Sicherheitslösungen erstellt. Eine Kombination zwischen den *IEEE 802.15.4 e/g* Standard, *TI 15.4 Stack* und *MQTT-SN* für die Anwendungsschicht für *Sub-1 GHz* und *BLE* für *2.4 GHz* wurden die am besten geeigneten Kernoptionen für den Stack angesehen. Darüber hinaus wurde *Listen Before Talk* in Verbindung mit *Adaptive Frequency Analysis* aufgrund der bestehenden Vorschriften als die am besten geeignete Spektrum-Zugriffsmethoden für *Sub-1 GHz* angesehen. Die Verschlüsselung/Entschlüsselung und Authentifizierung basierend auf der *AES CCM 128 Bit Shared Key* Methode, die die speziellen Funktionen des *CC1350* verwendet, wurde ausgewählt.
Basierend auf diesen Optionen wurde ein vollständiger *Communication Protocol Stack* zusammengestellt und beschrieben, und es wurde eine Architektur des gesamten Netzwerks vorgeschlagen. Labor- und Feldversuche wurden für eine Basisversion dieses Stack unter Berücksichtigung des Anwendungsfalls einer Autobahnbaustelle durchgeführt. Die erhaltenen Ergebnisse (d.h. Paketstatistik für verschiedene Konfigurationen) wurden analysiert, gefolgt von einer Liste weiterer Aufgaben, die in der Zukunft zu erledigen sind.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Motivation

The construction industry represents one of the most important sectors based on the number of workers and the amount of invested money. According to the annual report for the year 2017 of the *European Construction Industry Federation* [24], *6.4%* of Europe's total employment is guaranteed by the construction industry. At the same time, *43.6 million* workers in the European Union depend, directly or indirectly, on the construction sector. From the investments point of view, *1.364 billion Euro* were invested in the construction sector in *2017* in the European Union. Since 2013, when it was the end of the economic crisis' effects, the investments have been increasing constantly and it is expected to follow the same trend in the following years [13].
Even though it is a very profitable sector which ensures a large number of workplaces, it involves also a large number of accidents. According to *Eurostat* [25], more than one fifth of all fatal accidents at work in the European Union in 2015 took place within the construction sector. Another big problem of the construction sector is represented by the equipment theft. According to [74], the value of the construction equipments stolen each year is approximated to be between *200* and *800 million Euro*, not including the building materials. From these thefts, less than *25%* are recovered. Moreover, not always the equipments are brought back to the designated storage space after use. This implies time lost for finding them when the next usage is needed.
So, in the construction industry, in order to take care of the employees, increase the profit and optimize the workflow it is needed to track the equipments, materials and workers and to monitor the statuses of the machines and tools (i.e. to predict when the reparations are needed), the health of the workers and also the environment (e.g. temperature, pressure).

## 1.2 Problem Statement

Multiple solutions for these tracking and monitoring challenges are based on the *Internet of Things (IoT)*. *IoT* represents the creation of a global infrastructure of interconnected physical and virtual things which provide information to be used further on by different advanced services in order to fulfill various tasks. Thus, a *Wireless Sensor Network (WSN)* can be used. This network is composed by a large number of *Sensor Nodes* which are equipped with different types of sensors (e.g. location, temperature, position, motion). The acquired measurement values are send using a radio signal to a *Gateway Node*, thus ensuring the mobility of the *Sensor Nodes*. The values can be sent not only directly but also through multiple *Sensor Nodes*, achieving a better coverage of the network. From the *Gateway*, the information is sent over the *Internet* to the *Cloud* where it is processed by different services and used by the *Clients* which can be anywhere in the world. The structure of such a *WSN* for the construction industry is shown in Fig. 1.1.

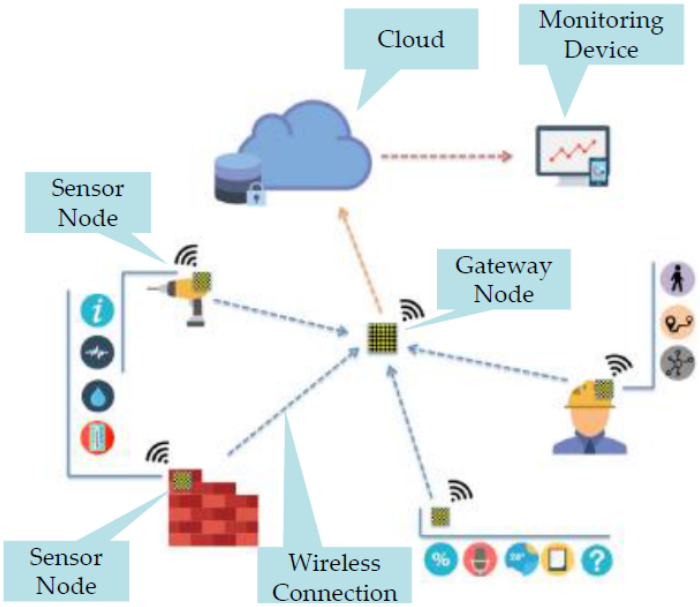


Fig. 1.1: Overview of the *Wireless Sensor Network* for a construction site

Multiple solutions of creating a *WSN* for the construction industry were developed. The main difference of the one on which it is based the current thesis is represented by the usage of the ultra-low power dual-band wireless *CC1350* microcontroller produced by *Texas Instruments*. It uses the *Sub-1 GHz* and *2.4 GHz* frequency bands for wireless communications. The *Sub-1 GHz* frequency band has the advantages of a higher range and a lower degradation when meeting obstacles, whereas the *2.4 GHz* frequency band is used by very popular communication standards (e.g. *Bluetooth*) which can enhance the user experience by using a smartphone

application. Though, due to its capability of handling both frequency bands, the *CC1350* microcontroller represents a bridge between them, combines their advantages and represents a valuable solution for developing *WSN* applications for a large range of fields (e.g. building automation, security systems, health monitoring, smart grid).

## 1.3 State of the Art

Wireless communications represents a popular topic nowadays which is discussed from both theoretical and practical point of views in different sources, alongside with different protocols and technologies ([86], [122]). Some of these sources are focused only on specific categories of wireless networks, for example *Low-Power Wireless Area Networks (LPWAN)* ([52]).

*Wireless Sensor Networks (WSN)*, a particular case of wireless networks where a large number of sensor nodes enable various services by physical and virtual interconnections, represent also the main topic of a large number of publications. The characteristics (e.g. topology, data aggregation, security) and the challenges (e.g. architecture, concurrent access) of a *WSN* can be explained from both theoretical and practical point of view by using a couple of applications (smart grid, smart water network, intelligent transportation, smart homes) ([73]). A high number of protocols, algorithms and technologies for the different layers of a *WSN* have been developed in order to cover different use cases and requirements ([21], [121]). Security represents also a key aspect of a *WSN*, due to the large number of existent vulnerabilities ([103]). Various techniques can improve the security of the current standards and implementations of the layer, which are characterized as being ”inadequate” to provide a safe functioning ([99]).

*Wireless Sensor Networks (WSN)* have been used in multiple applications from different domains, obtaining good performances. The construction industry is one of these domains. A *WSN* can be combined with *BIM (Building Information Modeling)* for visually monitoring the status of a building through gas, temperature and humidity sensors, in order to observe any abnormalities ([15]). The resource tracking and parameters' monitoring problems within building construction sites were discussed in relation with different possible wireless communication technologies ([107], [41], [88]). In one research, *ZigBee* is considered to be the best choice, needing however some implementation adaptations. The *TI 15.4 Stack* developed by Texas Instruments is used as a solution in several application reports as well. These highlight how the *Sub-1 GHz CC1310*, *CC1312* and multi-band *CC1350* and *CC1352* wireless microcontrollers can be used in *WSNs* for different purposes: smoke alarm [72], motion detector [71], door and window sensor [70], GPS tracking [51]. In all of them, a star network topology based on the

*TI 15.4 stack* available in the *CC13xx SDK* is described as the best solution when a concentrator node is used. Otherwise, a proprietary implementation of a network protocol based on the *EasyLink* layer can be implemented. A solution based on the *TI's Sensor to Cloud* reference design is briefly presented for connecting the collector to the internet using a gateway in order to send messages to the security company, the fire department or to the owner, in case of a fire. Due to the dual band capability of *CC1350* and *CC1352*, *Bluetooth Low Energy* technology can be used to enhance the user experience (e.g. configuration of the network, testing the network, updating the software, broadcast alerts through beacons (advertisements)). In the end, the power consumption analysis for a specified test case is shown. Even though not from the construction industry, another application which contains useful information is the one which consideres the usage of WSNs in animal tracking ([75]). Thus, their behaviour can be better understand. Zebras are tracked in a research area using low-power custom collars which contain a *GPS* unit, a *CPU*, a *Flash memory* and a wireless transceiver. Each collar collects data and sends them to the nearby collars when the zebras are gathering together (e.g. when drinking water). A mobile gateway is moved by a researcher after a longer period (i.e. several days) through the research area and receives the data from the nearby collars.

The *TI CC1350* ([44]) is a wireless dual-band ([116]) microcontroller part of the generic family *SimpleLink$^{TM}$ Sub-1 GHz wireless MCUs* ([60]). An open-source free Software Development Kit (SDK) is provided ([59]) for developing easily various applications without taking care of the drivers. A large number of project examples and tutorials are available within the *TI Resource Explorer* ([64]). A development board (*LaunchPad*) based on this microcontroller is also provided by *Texas Instruments* ([54]).

## 1.4 Contributions

The main goal of this thesis is to develop a *Communication Protocol Stack* for a *WSN* based on the dual-band *CC1350* microcontroller. This stack is intended to be implemented on the Real-Time Operating System *TI-RTOS*. The *WSN* is inteded to be used in the construction industry for multiple purposes (e.g. tracking and monitoring tools, vehicles, workers). Like any other network, wired or wireless, the considered one has to fulfill several properties [122]:

- *Predictability* - ability to predict the communication's performances (e.g. packet transmission time)
- *Scalability* - ability to include/exclude multiple nodes while functioning
- *Mobility* - ability to handle the change of nodes' positions

- *Reliability* - ability to guarantee correct transmissions and receptions
- *Robustness* - ability to provide potentially degraded functionalities when errors occur (e.g. when a message cannot be received correctly)
- *Privacy* - ability to prevent joining of unwanted node and unauthorized data transmission, modification and reading

Starting from these general properties, a particular list of requirements and specifications for the considered case of a *CC1350*-based *WSN* used in the construction industry was firstly created (Appendix A). These are defined in connection with the capabilities of the used *CC1350* microcontroller described thoroughly in on of the thesis' sections. A comparison between this microcontroller and the newer versions recently announced is done, concluding that it is better to use the basic version due to its stability and large community experience.
Further on, an overview of the most popular wireless communication solutions used in *IoT* and not only (e.g. *TCP/IP*, *6LoWPAN*, *Bluetooth*, *Zigbee*, *TI 15.4 Stack*, *Wi-Fi*, *SigFox*, *LoRa*, *Cellular*, *RFID*) is presented. These solutions are compared taking into account: used frequency, topology, range, data rate, layers covered and overhead of each message. In multiple literature sources exist comparisons between *IoT* communication protocols and technologies but only for smaller subsets of them ([28], [101], [19], [33], [81]). Thus, by having an analysis of these solutions and by comparing them with the requirements for the considered application, a solid decision can be made: reuse an already existent complete communication solution, combine parts of several existent solutions or implement a solution from scratch. Due to the fact that the *CC1350* microcontroller is dual-band, for the *868 MHz* frequency band a combination between *IEEE 802.15.4 e/g* standard, *TI 15.4 Stack* and *MQTT-SN Application layer* was chosen, whereas for the *2.4 GHz* frequency band, *BLE* was selected.
The restrictions imposed by the *European Commission* through *CEPT* and *ETSI* for the message transmission by the *Short Range Devices (SRD)* over the *868 MHz* frequency band are also taken into account. An overview of different spectrum access methods is presented, specifying for each of them what limitations are introduced by these regulations. Based on the possibility of sending unlimited messages, the solution based on *Listen Before Talk (LBT)* technique combined with *Adaptive Frequency Agility (AFA)* method is chosen. This solution can be used within the selected *Communication Protocol Stack* but it cannot be used by some of the other communication solutions (e.g. *Zigbee*), bringing one more advantage to the decision previously made.
Due to the security features of the *CC1350* microcontroller (e.g. dedicated *128 bit AES hardware accelerator*, implementation of the *ECC (Elliptic Curve Cryptography)* core in the *ROM*, *True Random Number Generator*), different security solutions can be used. A brief overview of these is presented and the most suited one for the chosen communication protocol is decided:

encryption/decryption and authentication based on the *AES CCM 128 bit shared key* technique. Thus, by combining these modules, the most suited *Communication Protocol Stack* for the considered application is obtained. Moreover, an architecture of the whole network based on this stack is proposed.
The use case of a highway construction site is considered. *Sensor Nodes* are desired to be placed on trucks in order to acquire information which will be used to improve the entire process by determining the optimal number of trucks, their routes, the number of loadings and transporting cycles done per day, the number and the length of the breaks. For this, information about the location, the waiting time for being loaded by the excavator and how many buckets of soil are needed in order to be filled have to be acquired. A proprietary already developed board for the *Sensor Nodes* based on the *CC1350* microcontroller is used, alongside the *CC1350 LaunchPad* for the *Gateway*. A basic version of the communication protocol between multiple *Sensor Nodes* and a *Gateway Node* was implemented (i.e. the same *Physical* layer but reduced *MAC*, *Network* and *Application* layers). Modules for sensors' data acquisition were also implemented and combined with the *communication stack*. A laboratory test analysis regarding the numbers of successfully received, lost and duplicated packets for different number of *Sensor Nodes*, sending periods and packet lengths is performed and the results are discussed. A field test was also performed and the results are described.

This thesis is composed of *8* sections. After this introduction into the construction field, containing the motivation, problem statement, state of the art and contributions, the next section includes a description of the hardware and the software blocks linked with the chosen microcontroller. Here are also thoroughly described the two frequency bands used by the *CC1350*, their differences and regulations which have to be respected. A comparison with the recently announced versions of the *CC1350* microcontroller is also included. In the third chapter, different spectrum access methods are explained, being important for the *Sub-1 GHz* frequency band restrictions. Further on, the fourth chapter briefly presents the security area of *WSNs* and the capabilites offered by the *CC1350* microcontroller for fulfilling them. The following section contains the description and the comparison of different communication standards and protocols used in *IoT*. By combining several communication protocols and standards, an application layer, a spectrum access method and a security solution, in section number six it is presented the designed proprietary communication stack which fulfills the described requirements. The seventh chapter contains details about the implementation did for the considered use case, the performed tests, a discussion about the results obtained and a list with further developments. The last chapter summarizes the work done within this thesis.

# Chapter 2

# Texas Instruments CC1350 microcontroller

In this chapter, the *Texas Instruments CC1350* microcontroller, which is used for the considered construction industry *WSN*, is analyzed thoroughly, from both the hardware (e.g. features, *RF Core*, *Sensor controller*) and the software (e.g. *SDK*, *TI-RTOS*, *TI 15.4 Stack*) perspectives. A comparison between the frequency bands supported by it is done, describing also the existent regulations for the *Sub-1 GHz* band. A comparison between this microcontroller and the recently announced versions *CC1352R* and *CC1352P* is done in the end.

## 2.1 Specifications

As described in [44], [45] and [48], *CC1350* is an ultra-low power dual-band wireless microcontroller produced by Texas Instruments (Fig. 2.1). It uses the *Sub-1 GHz* (*315-, 433-, 470-, 500-, 779-, 868-, 915- or 920-MHz*) and *2.4 GHz* frequency bands for wireless communications. It contains the following processing units:

- *ARM Cortex-M3* (clock speed up to *48 MHz*) as the main processing unit for the application and the higher layers of the wireless communication protocol stacks (from the link layer to the application layer)
- *ARM Cortex-M0* as dedicated radio controller (the physical layer of the wireless protocol stack)
- *Sensor Controller* for handling the connected sensors in order to keep as long as possible the main processor in the sleep state

The available memory types are:

- *Flash memory* (*128 KB*): nonvolatile in-system programmable storage for code and data. It is split into erasable pakes of *4 KB*
- *SRAM* (*20 KB*): used to store data end execute code
- *ROM*: contains a bootloader for reprogramming the *MCU* using *SPI* or *UART*, a preprogrammed *TI-RTOS* kernel and the driver library (*Driverlib*) used for accesing the peripherals
- *Flash cache* (*8 KB*): if it is disabled, it can be used as general-purpose *RAM*

*CC1350* contains various peripherals:

- Real-Time Clock, Continuous Time Comparator, 4 General-Purpose Timer Modules (4 *32-Bit* timers or 8 *16-Bit* timers; each of them being capable of handling *PWM*)
- *12-Bit ADC* (8-Channel analog *MUX*)
- Communication modules: *UART*, 2 *SSI* (*SPI*, *Microwire*, *TI*), $I^2C$, *I2S*
- *AES-128* security module
- Integrated temperature sensor
- and others

The main factors which make the *CC1350* microcontroller to be an ultra-low power one are:

- the sensors are handled in a very low-power way by the dedicated sensor controller
- the low-level *RF* protocol is handled by a dedicated RF core
- very low active *RF* and microcontroller current consumption
- flexible low-power modes

The *Sub-1 GHz* frequency band has the advantages of a higher range and a lower degradation when meeting obstacles, whereas the *2.4 GHz* frequency band is used by very popular communication standards (e.g. *Bluetooth*) which can enhance the user experience by using a smartphone application. Though, due to its capability of handling both *Sub-1 GHz* and *2.4 GHz* frequency bands, the *CC1350* microcontroller can be used in various Wireless Sensor Networks applications (e.g. building automation, security systems, health monitoring, smart grid).

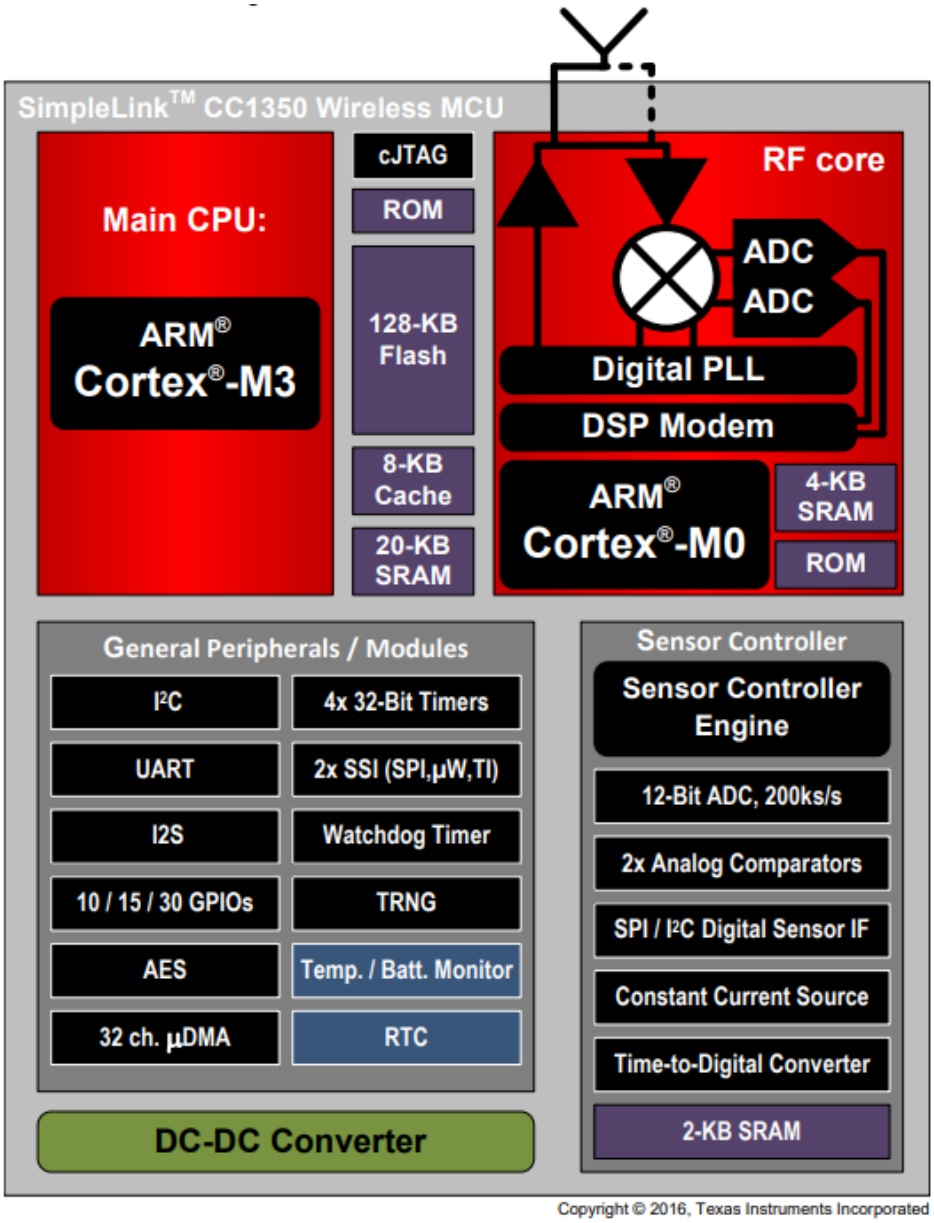


Fig. 2.1: The functional diagram of *CC1350* microcontroller [44]

## 2.2 RF Core

The *CC1350* microcontroller contains an *RF (Radio Frequency) core* based on the *ARM Cortex-M0* processor. The role of this module is to handle the low-level *RF* commands which are stored in *RAM* or *ROM*, leaving the main processor (*ARM Cortex-M3*) to handle the user application or to stay in a sleep-state, in order to ensure a low energy consumption. More precise, as described in [45], the *RF core* assembles the bit sequences representing the information into the selected packet structure, handles data to and from the system side and interfaces the analog *RF* circuits (implements the physical layer of the protocol stack). The main processor communicates with

the RF core processor through a hardware interface called *RF doorbell*.
Some of the characteristics of the *CC1350*'s *RF core* presented in [45] are:

- Supports *Sub-1 GHz* (*315-, 433-, 470-, 500-, 779-, 868-, 915- or 920-MHz*) and *2.4 GHz* frequency bands
- Wide range of modulation formats (multilevel *FSK (Frequency-Shift Keying)* and *MSK (Minimum-Shift Keying)*, *OOK (On-Off Keying)*)
- Wide range of data rates (between *625 bps* and *4 Mbps*)
- Dedicated packet handling accelerators for: Automatic *CRC (Cyclic Redundancy Check)*, *Forward Error Correction (FEC)* and *Data Whitening*
- Automatic *Listen-Before-Talk (LBT)*
- Digital *RSSI (Received Signal Strength Indicator)*
- Configurable channel filtering (supports channel spacing schemes between *40 KHz* and *4 MHz*)
- Receiver's sensitivity of *–124 dBm* using Long-Range Mode and *–110 dBm* at *50 Kbps* for *Sub-1 GHz* and *–87 dBm* for *2.4 GHz*

## 2.3 Sensor Controller

The *CC1350* microcontroller contains a *Sensor Controller* module based on a *16-bit CPU*. It is optimized for low power consumption and can perform independent of the main processor (*ARM Cortex-M3*) peripheral related operations (e.g read and monitor sensors) [50]. It executes code from a dedicated *2 KB ultra-low-leakage (ULL) RAM*.
The peripherals which are included in the *Sensor Controller* module, according to [45], are:

- A *12-bit ADC* with *8 inputs* (based on a multiplexor), capable of handling *200 ksamples/s*. It can be triggered by different sources (e.g. timers, I/O pins, software, the analog comparator, the RTC)
- Analog modules can be connected to up to *8* different *GPIO*s
- Capacitive sensing functionality - realized by combining a constant current source, a time-to-digital converter, and a comparator
- The low-power continuous clocked comparator can be used to wake the device from any state in which the comparator is active.

Using *Sensor Controller Studio IDE*, microcode for choosing which peripherals are controlled and what are the conditions for waking up the main CPU can be developed and programmed to the *Sensor Controller*.

## 2.4 Sub-1 GHz and 2.4 GHz frequency bands

In the last years, the popularity of the *Internet of Things (IoT)* has increased significantly, making more and more devices, sensors and actuators to be connected together and to the Internet. *2.4 GHz* represents the most used frequency band for *wireless local area networking (WLAN)*, having *Wi-Fi* as the main technology (based on the *IEEE 802.11* standard). According to [96], it is predicted that by 2020, more than 20 *Wi-Fi* devices will be in each house, which will determine a drop in the frequency band performance due to the high level of interference. In addition, the limited range (around *53 meters* for the *IEEE 802.11.n* standard) and the poor ability to pass through walls make the *2.4 GHz* band to be an unsuited solution for *IoT*, even though it is capable of high data rates (up to *1.73 Gbps* for the *IEEE 802.11ac wave2* standard).
The *Sub-1 GHz* frequency bands overcome the disadvantages of the *2.4 GHz* band and become more and more used in the IoT field. Thus, carrier frequencies below *1 GHz* in the ISM (Industrial, Scientific, and Medical) band are used (*433/868/915 MHz*). Due to the physical properties of lower frequencies, higher ranges and the ability to pass through walls and to bend around corners are obtained for the same antenna size, in contrast to the *2.4 GHz* band. However, lower data rates are achieved but lower interference level on these frequencies and lower power consumption are obtained. This frequency band is not yet very popular, not many devices (smartphones, tablets) being capable to use it.
Table 2.1 contains a comparison between the *Sub-1 GHz* and *2.4 GHz* frequency bands with respect to their main characteristics.
So, as outlined in [116], the *CC1350* micrcontroller, as a *Sub-1 GHz* and *2.4 GHz* dual band microcontroller, represents a bridge between the two frequency bands and combines their advantages: a very good RF range (using the *Sub-1 GHz* band) and the possibility to offer a pleasant user experience through a smartphone / tablet application (using the *2.4 GHz* band).

### 2.4.1 Sub-1 GHz Band Regulations

In Europe, the *European Commission* through *CEPT (The European Conference of Postal and Telecommunications Administrations)* and *ETSI (The European Telecommunications Standards Institute)* defined a series of regulations for the transmitters and the receivers of *Short Range Devices (SRDs)* using the *ISM* bands [17]. These regulations defines the spectrum management

Table 2.1: Comparison between the *Sub-1 GHz* and *2.4 GHz* frequency bands

| | **Sub-1 GHz** | **2.4 GHz** |
|---|---|---|
| **Range** | High (up to *20 km* for long-range mode (low data rate)) | Small (around *53 meters* for the *IEEE 802.11.n* standard) |
| **Data rate** | Small (up to *5 Mbps* but for long-range transmission it is much smaller) | High (up to *1.53 Gbps* for *IEEE 802.11.ac wave2* standard) |
| **Interference** | Low (fewer devices use this frequency band; it can pass through walls and to bend around corners better) | High (a large number of devices use this frequency band, high attenuation when passing through walls) |
| **Communication protocols** | Just a few and not so popular (e.g. *Sigfox*, *LoraWAN*); Multiple proprietary solutions | Multiple and very popular (e.g. *Wi-Fi*, *Bluetooth*, *Zigbee*) |
| **Antenna size** | Large | Small |

requirements: allocated frequency bands, maximum power levels, channel spacing, modulation and duty cycle.

As described in [83], these regulations are dividing the *SRD* applications in *13* categories (e.g. equipment for detecting avalanche victims, automatic vehicle identification for railways, alarms, non-specific SRDs). Thus, some frequency bands are dedicated for these categories, decreasing the probability of interference. Any application can be defined as *non-specific*, being capable to use any of the frequency bands which are not kept for the specific categories.

The considered application does not fit in any of the specific categories, so the *non-specific SRDs* category will be considered. For this category, the frequency bands based on the *868 MHz* frequency used by the *CC1350* used microcontroller and their restrictions are listed in Table 2.2. The *863 - 870 MHz* frequency band is called *"The 863 MHz band"*, after the smallest frequency, but also *"The 868 MHz band"* after the defining frequency. In this project, *"The 868 MHz band"* name will be used. In order to guarantee a fair usage of the frequency bands, the effective radiated power and the spectrum access restrictions have to be respected. For both considered bands, the maximum effective radiated power is set to *14 dBm*, which is exactly the maximum transmitted power of the CC1350 microcontroller. So, this restriction is fulfilled all the time. These are based either on a specific *Duty Cycle* or on using the *LBT (Listen Before Talk)* and *AFA (Adaptive Frequency Agility)* techniques. The explanations of these spectrum access methods are included in Chapter 3.

Table 2.2: The frequency bands based on the 868 MHz frequency and their regulations for Non-Specific SRDs [17]

| **Frequency Band** | **Effective Radiated Power** | **Spectrum access** |
|---|---|---|
| 863 - 870 MHz | 25 mW = 14 dBm | $\leq$ 0.1 % Duty Cycle or LBT + AFA |
| 868 - 868.6 MHz | 25 mW = 14 dBm | $\leq$ 1 % Duty Cycle or LBT + AFA |

## 2.5 SimpleLink CC1350 wireless microcontroller LaunchPad

*SimpleLink CC1350 wireless microcontroller LaunchPad* is a development board created by *Texas Instruments*. It is part of the *SimpleLink MCU Platform* which provides the complete environment (hardware, software and developing tools) for developing *IoT* applications [61]. As described in [64], it contains:

- The *CC1350 MCU*
- Access to all *I/O* signals with the *BoosterPack plug-in module* connectors
- Separately integrated PCB trace antennae for both *Sub-1 GHz* (*868 MHz*) and *2.4GHz* frequency bands
- *XDS110* debugger.

## 2.6 RF Driver

The *RF Driver* provides the exclusive access to the *RF core* of the *CC1350* microcontroller for the user applications. In [46] it is explained that the driver represents a high-level interface for command execution and for the *Radio Timer (RAT)* (a timer which runs with a speed of *1/4* of the high-frequency oscillator and provides accurate timing for the execution of *RF* commands). This driver ensures the lowest possible power consumption by providing automatic power management that is fully transparent for the application.

## 2.7 EasyLink

*EasyLink* is an abstraction layer on top of the *RF Driver*. It represents a starting point for customers in creating a *Proprietary Sub-1 GHz protocol* or application [49].

## 2.8 Proprietary Sub-1 GHz Protocol

Due to the fact that there is not yet a *Sub-1 GHz* communication protocol widely used (as *Wi-Fi* or *Bluetooth* are for the *2.4 GHz* band), using the *EasyLink* layer for the *RF* communication and the *TI-RTOS* for real-time multitasking services, developers can design their own *Sub-1 GHz* communication protocol. This can be then further used in their own various applications.

## 2.9 IEEE 802.15.4

*IEEE 802.15.4* is a communication standard which defines the physical and the data link layer for Wireless Personal Area Networks (WPAN) having low power consumption and low cost [73]. It is used for the operation of Wireless Sensor Networks. The main characteristics of this standard are:

- It uses the ISM frequency bands at *800/900 MHz* and at *2.4 GHz* and represents the base for some popular standards: *Zigbee*, *WirelessHart* and *Thread*
- It is suitable for real-time applications due to the allocation of time slots
- It ensures collision avoidance through the *CSMA/CA (Carrier-Sense Multiple Access with Collision Avoidance)* network access method
- The topologies defined in this standard are peer-to-peer (all the nodes have the same privileges and are connected directly to some of the oher nodes) and star (one coordinator which is connected directly to all the other nodes)

To the base standard, multiple revisions were made, adding new features in order to overcome its limitations (e.g. in *IEEE 802.15.4e* it is added *Time Slotted Channel Hopping (TSCH)* as a combination between time slotted access with multi-channel and channel hopping capabilities) [36].

## 2.10 TI-RTOS

*TI-RTOS* is an ecosystem of open-source tools developed by Texas Instruments. It can be used by various embedded processors, making it highly configurable. As described in [65], it contains:

- device drivers (e.g. *Ethernet*, *GPIO*, $I^2C$)
- middleware components (e.g. *TCP/IP*, *USB*, *FAT File System*)

- interprocessor communication
- a real-time operating systems kernel (previously known as *SYS/BIOS*). It is designed to be used when real-time scheduling and synchronization are needed in an application. It provides preemptive multi-tasking, memory management and real-time analysis and configuration tools

In [65] and [66] it is specified that the main modules of the TI-RTOS kernel are:

- *Scheduler* - its responsibility is to ensure that the highest priority thread is running if it is capable (i.e. no needed shared resources are occupied by other threads). The four types of threads in *TI-RTOS*, from the one with the lowest priority to the one with the highest priority:
  - *Idle* - it is the lowest priority task (priority 0) and it runs when no other task is ready to run. It performs user defined jobs and background service jobs like system stack checking and CPU load determination. This task allows the *Power Policy Manager* to enter the lowest available power savings.
  - *Regular Task* - represent the common OS threads which can have different priority levels. A task has its own task used to maintain its state. The stack is used when it blocks (when it waits for the necessary resources to be available). A task can be in four different states (Fig. 2.2):
    - *Ready* - the task is able to be executed but a task with a higher priority is occupying the processor (is in the *Running* state)
    - *Running* - the task is utilizing the processor and it is executing
    - *Blocked* - the task is waiting for a temporal (e.g. a delay to be finished) or external event (e.g. wait for a shared resource to be free)
    - *Terminated* - the task finished its execution (i.e. it does not contain any infinite loop)
  - *Software Interrupt* - software events trigger these tasks which are running to completion (they do not block). They do not have their own stack, like in the case of regular tasks, but they share the same one. Although, hardware interrupts can interrupt them and also higher priority software interrupts can interrupt lower priority ones.
  - *Hardware Interrupt* - hardware events trigger these tasks which are running to completion (they do not block). They do not have their own stack, like in the case of regular tasks, but they share the same one, which is identical with the one used by the software interrupts. Although, higher priority hardware interrupts can interrupt lower priority ones.

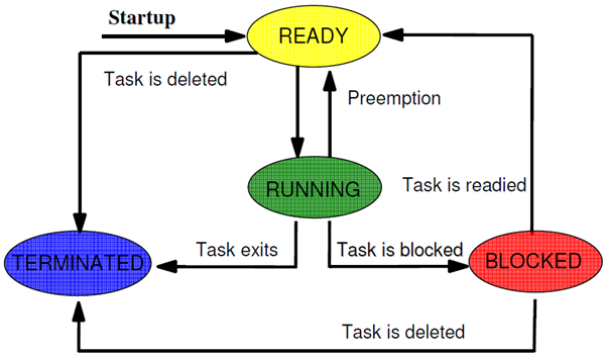


Fig. 2.2: The states of a regular task and the conditions for switching them [66]

- *Thread Communication* - the main ways of ensuring communication between threads are:
  - *Semaphore* - an object used to control the access to a resource shared by multiple tasks. Can be used for task synchronization and mutual exclusion
  - *Mutex* - a *Binary Semaphore* respecting the ownership property (only the task which locked a Mutex can release it)
  - *Mailbox* - used to pass buffers from one task to another on the same processor
  - *Queue* - a double linked list with no synchronization
  - *Gate* - represents a reentrant *Mutex* and it is used to protect concurrent access to critical data structures
  - *Event* - communication and synchronization way between tasks, similar to *Semaphores*. Multiple conditions (i.e. events) can be specified before the waiting task returns
- *Timing Services* - are ensured by:
  - *Clock* - module responsible for generating the system tick that is used by the kernel to keep track of time
  - *Timer* - module for managing the available hardware timers in order to measure time intervals
  - *Seconds* - module for providing a way to set and get the number of seconds elapsed since *1st of January 1970 00:00:00 GMT* (the Unix epoch)
  - *Timestamp* - module for adding timestamps (e.g. adding timestamps to logs)
- *Memory Manager* - provides tools to set up the memory map and also allocate and dealocate memory buffers while the system runs.

## 2.11 POSIX

*POSIX (Portable Operating System Interface for Unix)* is a family of *IEEE* standards and represents an abstraction layer of the *RTOS kernel* functionalities in order to ensure compatibility between different operating systems (Fig. 2.3). Thus, user applications can be ported and reused to different kernels (e.g. *TI-RTOS*, *FreeRTOS*). *POSIX* defines the *API (application programming interface)*.
As explained in [53], the *SimpleLink SDK* (explained later) supports only some of the *POSIX* sets of *APIs*:

- *Pthread* (includes mutexes, barriers, condition variables and read-write locks)
- *Semaphores*
- *Clocks / Timers / Sleep*
- *Message Queue*

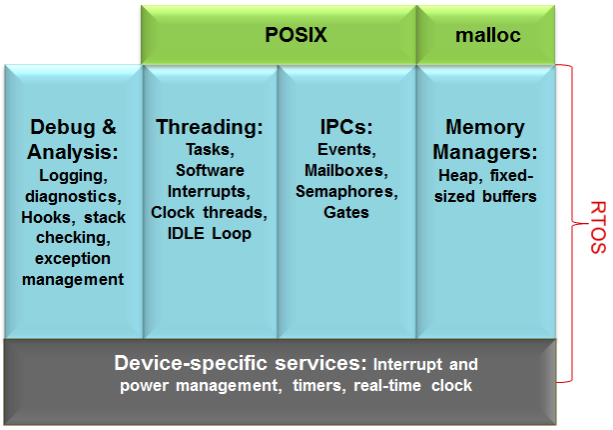


Fig. 2.3: POSIX as an abstraction layer on top of the RTOS kernel [53]

## 2.12 TI 15.4 Stack

*TI 15.4 Stack* is a complete end-to-end wireless application development solution provided by *Texas Instruments*, used for developing directly applications without being needed to focus on the lower layers. It is included in the *SimpleLink CC13x0 Software Development Kit (SDK)* and can be used for multiple wireless microcontrollers from Texas Instruments (e.g. *CC13x0*, *CC13x2*). It is a software stack based on:

- the *IEEE 802.15.4* basic standard (2006): defines the physical and media access control layers of the *OSI (Open System Interconnection)* model of network systems

- the *IEEE 802.15.4e* revision: for industrial applications
- the *IEEE 802.15.4g* revision: for smart-utility networks
- the *Wi-SUN* field area network specifications: for the frequency-hopping scheme used

It is intended to be used for applications which require "extremely low-power, long-range, reliable, robust and secure wireless star-topology-based networking solutions" [47].
The features of the *TI 15.4 stack* are described in [47] and [59]:

- Two architectures can be used, depending o the end product application:
  - the application and the protocol stack are both implemented on the same TI wireless microcontroller (e.g. *CC13x0*)
  - the protocol stack runs on a TI wireless microcontroller (e.g. *CC13x0*) and the application is executed on an external Microcontroller / Microprocessor unit. The connection between the two modules is done using the *Network Protocol Interface (NPI)* over a serial *UART*
- Four network operation modes can be utilized:
  - *Synchronous mode (Beacon mode)*: The coordinator of the network (*PAN*) transmits periodically beacons in order to indicate its presence and to allow the nearby devices to perform PAN discovery and synchronization. The beacons help on one side the device intending to join the network to synchronize timing and network related parameters before starting the join process and on the other side the existing device in the PAN to maintain the network synchronization
  - *Asynchronous mode (Non-Beacon mode)*
  - *Frequency-hopping mode*: This mode has the advantages of sharing the same frequency band with a large number of devices (due to the low narrow-band interference effects) and decrease the interception possibility.
  - *Bluetooth Low Energy (BLE) advertiser*: non-connectable unidirect *BLE advertisements* can be used simultaneously with the *TI 15.4 stack*, based on the *uBLE (micro BLE)* stack subsystem, without using a separate *TI-RTOS stack*. The payload of the *BLE advertisements* is fully modifiable.
- Medium access is performed using *CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance)* in all frequency bands except *863 MHz* where *LBT (Listen Before Talk)* is used

- Security is ensured by the *AES encryption* as defined by the *IEEE 802.15.4* specifications. The application is responsible for the keys' management.
- Up to *150 nodes* can be connected in a network
- *Over-The-Air (OAD)* firmware update can be performed using Sub-1 GHz or BLE
- Network and device management: joining, commissioning, service discovery
- *FCC (Federal Communications Commission) / ETSI (European Telecommunications Standards* Institute) certification-ready with worldwide region support for *915, 863, and 433 MHz* bands

## 2.13 Bluetooth Low Energy (BLE) Stack

Characteristics [43] [84]:

- It is based on the *BLE core stack 4.2*, *version 2.3.2*
- It is not backward compatible to the classic *Bluetooth* (*Bluetooth Basic Rate / Enhanced Data Rate (BR/EDR)*)
- The architecture of the *BLE stack* is shown in Fig. 2.4. It contains two blocks, the *Controller* (the lower layers of the stack, including the radio) and the *Host* (the upper layers of the stack), being similar with the implementation of the classic *Bluetooth*, where these sections are implemented separately. On top of these two blocks is placed the user application.
- The *Controller* block contains:
  - The *Physical Layer* - defines a *1 Mbps*, *40 channels* (the last *3* are used for advertising, broadcast and setting up connections), adaptive frequency hopping, *GFSK* modulation radio which operates on the *2.4 GHz* ISM unlicensed frequency band
  - The *Link Layer* - interfaces the *Physical Layer*, establishes connections, filters advertisement packets, manages the connection interval and defines the *Advertiser* (a device generating and sending advertising packets; through these advertisements, it informs the initiating devices that it is a connectable device), *Scanner* (a device scanning for advertising packets), *Master* (a device which initiates a connection) and *Slave* (a device which accepts a connection request) roles
  - The *Host - Controller Interface* - ensures the communication between the Host and the Controller wither through a software *API* or by a hardware interface (e.g. *UART*, *SPI*, *USB*)

- The *Host* block contains:
  - *Logical Link Control and Adaptation Protocol Layer* - provides data encapsulation, fragmentation and recombination services to the upper layers
  - *Attribute Protocol Layer* - allows a device (the server) to expose a set of attributes and their associated values to a peer device (the client)
  - *Generic Attribute Profile Layer* - it is placed on top of the *ATT*, it adds a data model and a hierarchy and it defines how data is organized and exchanged in between different applications
  - *Generic Access Profile Layer* - interfaces with the application and the profiles in order to provide the discovery and the connection services. Fig. 2.5 shows the *GAP* state diagram of a device according to the role for which it was configured. The states are based on the defined roles from the *Link Layer*: *Standby* (after reset, the device is in the initial idle state), *Advertiser*, *Scanner*, *Initiator* (the device establishes a connection with an advertiser), *Slave* and *Master*
  - *Security Manager Layer* - defines the methods for pairing and key distribution and provides methods for the other layers for connecting and exchanging data with other devices

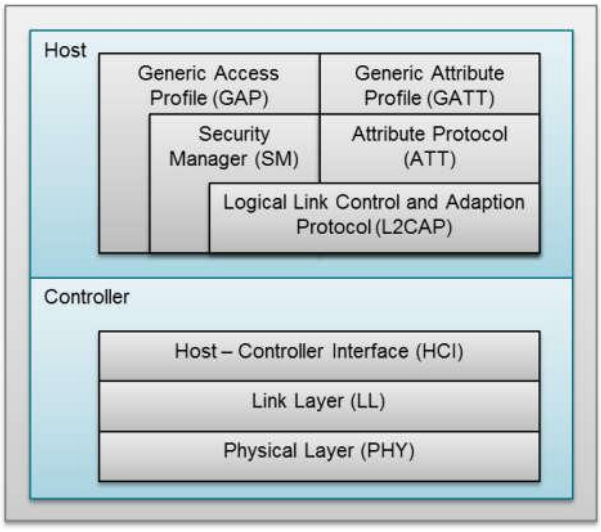


Fig. 2.4: The architecture of the *BLE* Stack [43]

- *BLE* devices can have two roles: *Central Devices* or *Peripheral Devices* (they connect to the *central devices*)
- *BLE* devices can send two types of data: *Advertising Packets* and *Scan Response Data* (sent as response to *advertising packets* in order to request more information and to initiate the connections)

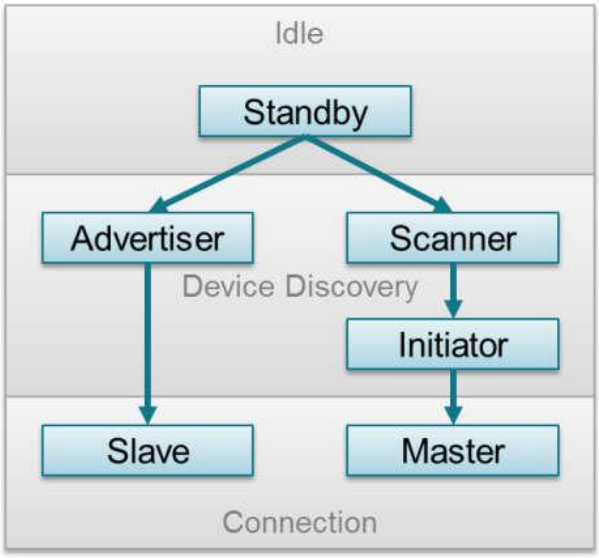


Fig. 2.5: The *GAP* state diagram [43]

- *BLE* devices can transmit packets through *Broadcasting* (sending packets to all the listening devices) or *Connections* (periodical data exchange of packets between two devices)
- Alongside the *Point-to-Point* and the *Broadcast* topolgies, *BLE* supports also the *Mesh* topology in order to create larger networks used especially in monitoring and control

## 2.14 TI SimpleLink MCU Software Development Kit (SDK)

The *TI SimpleLink MCU Software Development Kit (SDK)* is composed from a set of software development tools used for developing applications based on some of the *Texas Instruments* microcontrollers. According to [56], the components of the *SDK* are (Fig. 2.6):

- *Hardware Abstraction Layer (HAL)* - the layers used by the drivers and the OS kernels to access the hardware features
- *OS / Kernel* - the kernel of the operating system, by default, *TI-RTOS*, but can be changed (e.g. FreeRTOS). *POSIX* abstraction layer can be used for ensuring compatibility between different operating systems. The *Driver Porting Layer (DPL)* abstracts driver interfaces.
- *TI Drivers API* - facilitates the usage of the functionalities of the hardware-specific drivers
- *Middleware* - add functionalities on top of the drivers (e.g. communication stacks - BLE, TI 15.4 Stack)

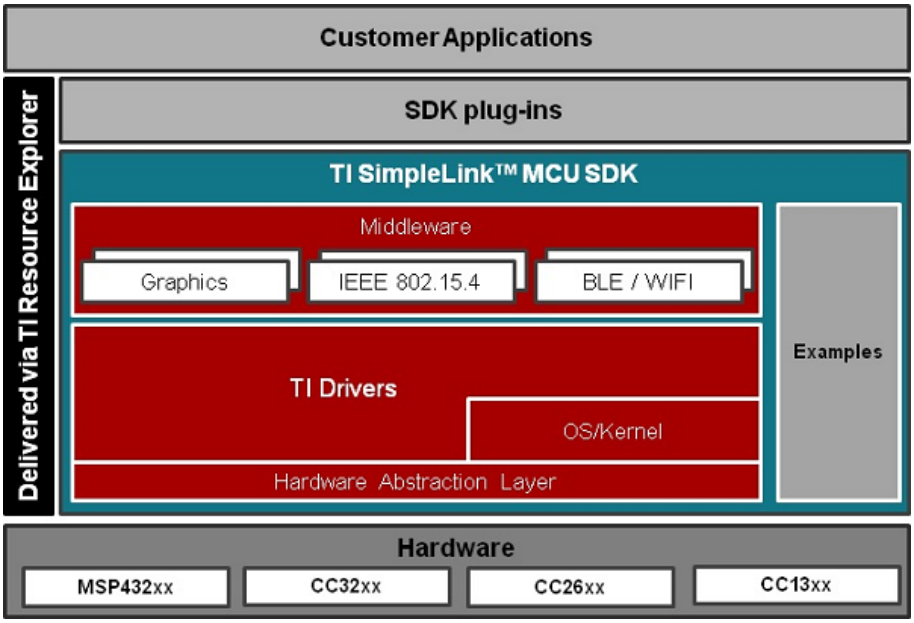


Fig. 2.6: The components and the architecture position of the *TI SimpleLink MCU SDK* [56]

The *TI SimpleLink MCU SDK* does not include an *IDE (Integrated Development Environment)* or code generation tools (e.g. compiler, linker). However, it includes a range of examples in order to make easier to begin a new project.

## 2.15 Application

The applications developed for the *Texas Instruments CC1350* microcontroller are based on the *TI SimpleLink MCU SDK* from which it is offered access to all the drivers, the communication protocols and the real-time operating system. If a *Proprietary Sub-1 GHz Protocol* is desired to be developed and used, *EasyLink* and *TI-RTOS* available in the *CC1350 SDK* should be used. Moreover, the examples included in the *SDK* can be used as a starting point.

## 2.16 Discussions

### 2.16.1 The interconnections between the CC1350's components

The presented hardware and software components related with the *CC1350* microcontroller are shown in Fig. 2.7 together with the interconnections between them. Thus, the entities which are involved in connecting the user application and the the hardware used (the *CC1350 LaunchPad* and the proprietary *CC1350-based Sensor Node*) can be seen more easily.

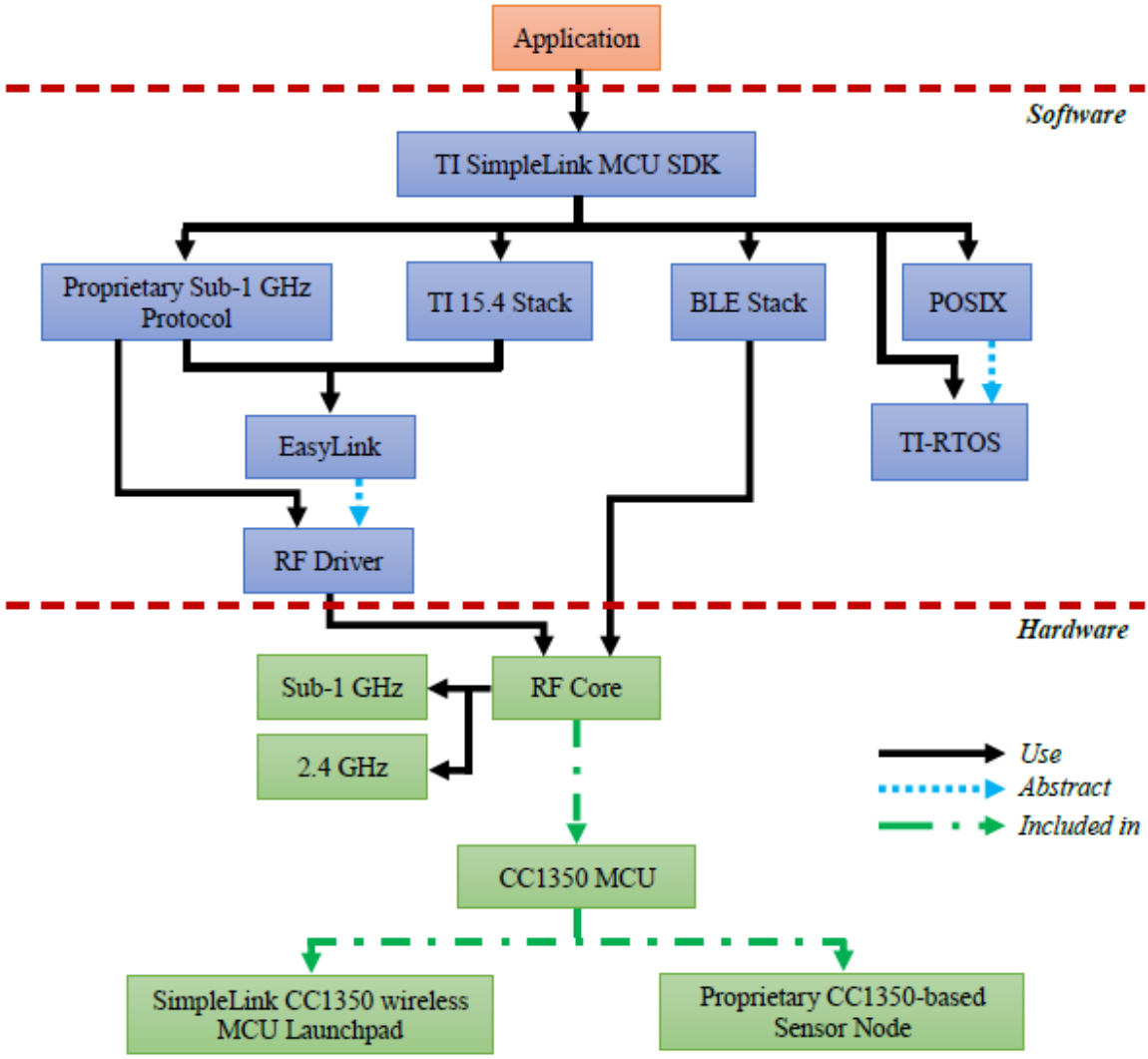


Fig. 2.7: The interconnections between the *CC1350*'s related hardware and software blocks

### 2.16.2 Comparison between *CC1350* and the newer versions *CC1352R* and *CC1352P*

The *CC1350* microcontroller has multiple valuable advantages: it is a dual-band wireless microcontroller (*Sub-1 GHz* and *2.4 GHz* frequency bands), it is ultra-low power, it has separate radio and sensor controllers, it is cheap (approximately *3.4 $* for one chip), it has a complex *Software Development Kit (SDK)* (see Section 2.14) and it has a large community of users which produced a high number of forum discussions and online articles.

However, it has also some disadvantages. The main one is represented by the strongly constrained available available memory. The *ROM* (which has a small size, not specified in the datasheet) is pre-programmed with the embedded *TI-RTOS kernel*, device driver functions,

low-level protocol stack components, and a serial bootloader. The main parts of the communication protocol stacks have to be loaded into the internal *Flash* memory. Due to its small size (*128 KB*), the memory left for the application code is very small. Moreover, if an application requires a concurrent communication over the two available frequency bands (*Sub-1 GHz* and *2.4 GHz*), this cannot be done. Only one communication protocol stack can be loaded in the internal *Flash* memory and used. The second one has to be loaded in the external *Flash* memory. When this stack is needed, it has to be switched with the other one, making the usage of both protocol stacks in the same application heavy and slow. The small *SRAM* size limits the number of connections a node can have at the same time (*50* with the *MAC layer* security, and around *400* without, due to the *RF* link lists which have to be stored during runtime). Moreover, only the *TI 15.4 Stack*, *EasyLink* based *Sub-1 GHz* proprietary stack and the *BLE version 4.2* stacks are supported by this microcontroller and its *SDK*, the range of solutions being limited.

During the development of this project, two new upgraded microcontrollers were announced by *Texas Instruments*: *CC1352R* [58] and *CC1352P* [57]. They are planned to be released at the beginning of *2019*, being in the *Preproduction* stage (i.e. experimental devices without any warranties, for which the testing and the final processing might not have been completed currently) at the moment of this project development. The main differences to the *CC1350* are shown in Table 2.3. One important feature of these two new microcontrollers is the support for the *Zigbee* communication protocol. Even though the *Zigbee* stack (*Z-Stack*) included in their *SDK* uses *Zigbee 3.0*, which is based on the Zigbee PRO 2015 specifications which can use either the *Sub-1 GHz* or the *2.4 GHz* frequency bands (not both at the same time, this feature being added in *Zigbee PRO 2017* specifications), the actual implementation supports only the *2.4 GHz* band. For the considered project, an advantage would have been represented by having the possibility to use the *Zigbee* protocol for the *Sub-1 GHz*, due to the advantages of this communication protocol (see Chapter 5) and of this frequency band (see Section 2.4). The *CC1352P* microcontrollers has an integrated power amplifier which increases the maximum transmitted power to *20 dBm*. Even though this would guarantee a higher range, the restrictions described in Section 2.4.1 limit the maximum transmitted power to *14 dBm*, which would bring this microcontroller to the same level as the *CC1350* microcontroller. However, even though the *CC1352R* and *CC1352P* microcontrollers still have very valuable advantages, they does not represent an option for the considered project at this moment, due to the fact that they are in the *Preproduction* stage, because the community is not yet familiar with them and there are just a few information and discussions about them and because they are not pin to pin compatible with *CC1350* which would imply a new design and manufacturing of the hardware boards.

Table 2.3: Comparison between the *CC1350*, *CC1352R* and *CC1352P* microcontrollers

| | **CC1350** | **CC1352R & CC1352P** |
|---|---|---|
| **ROM** | Includes the embedded TI-RTOS kernel, device driver functions, low-level protocol stack components, a serial bootloader | 256 KB, includes the embedded TI-RTOS kernel, device driver functions, a serial bootloader and significant parts of the communication protocol stacks |
| **Flash** | 128 KB | 352 KB |
| **Communication protocols supported** | TI 15.4 Stack, Sub-1 GHz proprietary stacks, BLE 4.2 | TI 15.4 Stack, Sub-1 GHz proprietary stacks, BLE 5, Zigbee, Thread;<br>Multiprotocol support (based on the Dynamic Multi-Protocol Manager which allows multiple wireless stacks to coexist and operate concurrently (only works with BLE and EasyLink at the moment)) |
| **Data Rate** | Maximum 4000 Kbps | Maximum 5000 Kbps |
| **TX Power** | 14 dBm | 14 dBm for CC1352R; 20 dBm for CC1352P (has an integrated power amplifier) |
| **Main processing unit** | ARM-Cortex M3 | ARM-Cortex M4F |
| **Power consumption** | Lower than CC1352P and CC1352R for Active-Mode RX, Active-Mode TX, Active-Mode MCU, Standby | Lower than CC1350 for Shutdown mode<br>Very high for CC1352P in Active-Mode TX at +20dBm and 868 MHz (65 mA) |
| **UART** | 1 port | 2 ports |
| **Security** | AES 128-bit hardware accelerator | AES 128-bit, AES 256-bit, public key (used for RSA and elliptic curves), SHA2 (with support for SHA224, SHA256, SHA384, SHA512) hardware accelerators |

| | | |
|---|---|---|
| **GPIOs** | 30 (RGZ package) | 28 |
| **Price** | 3.4$ | 3.69$ for CC1352R<br>5.25$ for CC1352P |
| **Availability** | Available | Preproduction<br>Available from 2019 |

# Chapter 3

# Spectrum Access Methods

The *Spectrum Access Methods* (or *Channel Access Methods*) are those techniques which allow multiple devices to communicate simultaneously in the same area by sharing the transmission medium. The main methods discussed in this Chapter are *Duty Cycle*, *Listen Before Talk (LBT)* and *Adaptive Frequency Agility (AFA)*, due to the fact that these are specified in the regulations of the *Sub-1 GHz* frequency band (Chapter 2). Other methods are briefly introduced.

## 3.1 Duty Cycle

Characteristics [32] [83]:

- Represents the maximum total time which can be used for transmitting, expressed as percentage related to a one hour time period
- It represents a solution for multiple users to share the time axis without a central coordinator
- Besides the total transmitting time, it defines also the maximum time of a single transmission and the minimum time between two consecutive transmissions
- Table 3.1 contains these values for the duty cycles of *0.1%* and *1%*, respectively

Table 3.1: The restrictions imposed by two different duty cycle values [83]

| **Duty Cycle** | **Maximum total transmission time** | **Maximum time of a single transmission** | **Minimum off time between two consecutive transmissions** |
|---|---|---|---|
| 0.1% | 3.6 seconds | 0.72 seconds | 0.72 seconds |
| 1% | 36 seconds | 3.6 seconds | 1.8 seconds |

## 3.2 Listen Before Talk (LBT)

Characteristics [32]:

- It is a *Spectrum Access Method* which is based on the verification of the channel whether it is free or not. This verification is done using the standard *Clear-Channel Assessment* methods:
  - *Preamble Detection* - a detector is running continuously in order to sense the preamble of a transmitted packet. The main disadvantages are the high level of power consumption due to the continuous running and the dependency on previously known preamble content
  - *Energy Detection* - it is a robust solution which is based on comparing the *RSSI* level with a threshold value. It does not depend on any previous information (e.g. format of the packets, modulation scheme). However, the reliability decreases for lower signal-to-noise ratios
- The performance of the *LBT* is influenced by:
  - The *Signal-To-Noise Ratio (SNR)* - if it is low then the probability of triggering a false alarm increases; if it is high then the probability of missing an alarm decreases
  - The *Energy Detection Threshold* - if it is high then the probability of missing an alarm increases and the probability of triggering a false alarm decreases; if it is low, the opposite case occurs
  - The *Listening Time ($T_L$)* - the period of time when the channel is listened for any signals (Fig. 3.1)
  - The *Dead Time ($T_D$)* - the period of time between the end of the listening period and the start of the transmission (Fig. 3.1)
  - The *Minimum Interference Detection Interval ($T_R$)* - the minimum period of time the signal must be present in order to be detected (Fig. 3.1)
- *LBT* is affected by the main problems in wireless networks (Fig. 3.2):
  - *The Hidden Node* - a transmitter $T_1$ is ”visible” for a receptor $R$ but not for another transmitter $T_2$ which wants to transmit to the same receptor $R$. Thus, for $T_2$ the channel is free so he is transmitting, leading to collisions at $R$
  - *The Exposed Node* - a transmitter $T_1$ is preventing for transmitting for a receptor $R_1$ because it is too close to a transmitter $T_2$ which transmits to the transmitter $R_2$

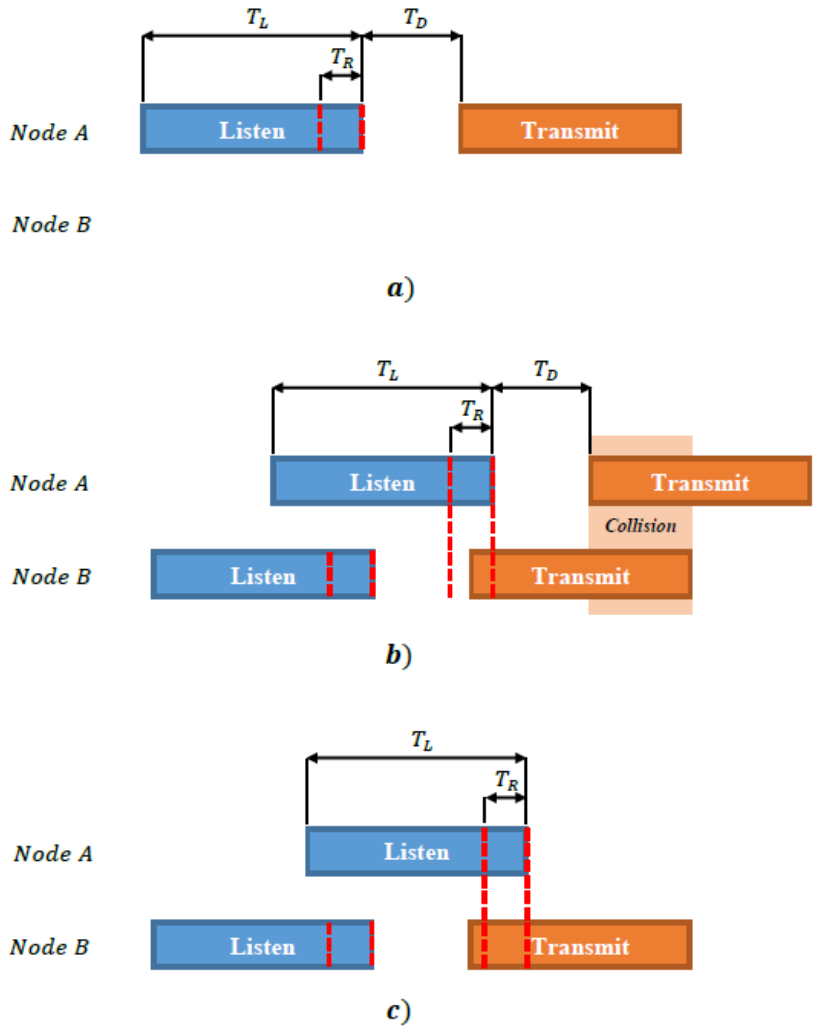


Fig. 3.1: The *Listening Time* ($T_L$), The *Dead Time* ($T_D$) and the *Minimum Interference Detection Interval* ($T_R$) for the *LBT* spectrum access method. In *a)* only one node wants to transmit a packet. In *b)* two nodes want to transmit and due to the fact that the transmission of *Node B* is not present at least for $T_R$ in the listening interval $T_L$ of *Node A*, a collision occurs because *Node A* considers that the channel is free. In *c)* two nodes want to transmit and due to the fact that the transmission of *Node B* is present at least for $T_R$ in the listening interval $T_L$ of *Node A*, a collision does not occur because *Node A* knows that the channel is not free

- If the *LBT* method signals that the channel is not free, the packet which wanted to be transmitted can either be discarded or a retransmission can be attempted after a period of time

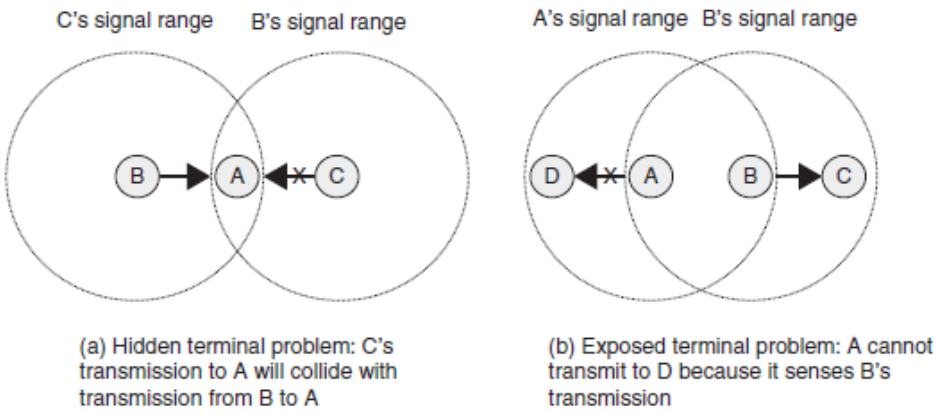


Fig. 3.2: *The Hidden Node* and *The Exposed Node* problems [122]

## 3.3 Adaptive Frequency Agility (AFA)

Characteristics [32] [17] [16]:

- It is a *Spectrum Access Method* based on the frequency selection. *Frequency Agility* means that two devices which want to communicate can select a specific frequency from the used frequency band according to the status of the environment (e.g. occupied channels). In addition, being *Adaptive* means that some of the channels are avoided, typically the ones which are occupied by fixed transmission sources (which does not *hop*)
- In the regulations for the *Sub-1 GHz* frequency band usage it is not specified what preconditions should be fulfilled in order to change the frequency channel. Thus, different solutions can be used:
  - The transmitter keeps the same channel until it detects that the channel is occupied (i.e. contention)
  - Irregular frequency channel changes - if the transmitter had to wait in the past multiple times for the frequency channel to be free or it had to retransmit the packet multiple times, it can decide that the current frequency channel is not reliable anymore and it should be changed
  - Change the frequency channel continuously, at regular time intervals, according to a previously known sequence, no matter of the channels' status - this method is called *Frequency Hopping*

### 3.3.1 Frequency Hopping

- It is also called *Frequency Hopping Spread Spectrum (FHSS)*

- It is a feature which allows the devices to change their frequency used for transmitting or for listening (reception) according to a certain sequence
- It solves the problem of interference on specific channels
- According to the regulations for the *Sub-1 GHz* frequency band usage, by using the *Frequency Hopping*, the usage of a higher transmitting power is allowed, which leads to a higher range. According to [118], an increase of up to *34* times more coverage in rural areas and *13* times more coverage in urban areas was obtained. However, this method increases the complexity of the project
- The *Adaptive* variant of *Frequency Hopping* identifies the fixed sources of interference and excludes them from the list of available channels, never ”hopping” on these ones [39]. It is used in *Bluetooth*

Characteristics of the *Frequency Hopping* implemented in the *SDK* of the *TI 15.4 Stack* starting with the version *2.0.1* [117] [118] [47]:

- It is based on the *Wi-SUN (Wireless Smart Utility Network) Alliance* specifications defined for *FANs (Field Area Networks)*
- It uses *Receiver Directed Transmissions* - the data frame is sent by the transmitter on the receiving node’s current frequency channel. This is possible due to the tracking done by each node of its neighbour’s hopping sequence
- The hopping sequence is based on a *Direct Hash Channel Function (DH1CF)* which generates pseudo random frequency channels using the extended address of the node. Thus, this sequence is unique for each node
- During the initial discovery and joining procedure of a node, the information about the hopping sequence is exchanged. For the joining procedure, four types of asynchronous messages are sent over a specified non-changeable channels list, not being dependent on the hopping sequence of any node. These messages are:
  - *PAN Advertisement Solicit (PAS)* - used for requesting a *PAN Advertisement (PA)* frame from the coordinator or from other already joined nodes
  - *PAN Advertisement (PA)* - used for informing the neighbours about the *PAN ID*, routing cost and *PAN* size by the coordinator or by the already joined nodes
  - *PAN Configuration Solicit (PCS)* - used for requesting a *PAN Configuration (PC)* frame from the *PAN coordinator* or from neighbours by the device which has the *group master key (GMK)* used in the network

  - *PAN Configuration (PC)* - used for transmitting by the coordinator or an already joined node, the *Hopping Sequence* and the hash values of the list of *GMK* keys that are actively used

- The joining procedure defined by *Wi-SUN*, based on the previous explained messages (Fig. 3.3), is silent, meaning that the *PAN coordinator* does not know which devices successfully joined the network. The *TI 15.4 Stack* adds an additional step in order to solve this problem, by using the *MAC Layer* association procedure described in the *IEEE 802.15.4* specifications (Fig. 3.4). Moreover, the coordinator can know if the joining node is *sleepy* or *non-sleepy* (through the *Capability Information* field of the association request message sent by the device to the *PAN coordinator*). Thus, it knows whether it should send the responses as soon as the requests are received or it should buffer them until the node wakes up

- The *863 - 870 MHz* frequency band has:

  - *34 channels* for *50 Kbps* and *5 Kbps* data rates, (*200 KHz* channel spacing, the frequency of *channel 0* is *863.125 MHz*)
  - *17 channels* for *200 Kbps* data rates, (*400 KHz* channel spacing, the frequency of *channel 0* is *863.225 MHz*)

- The number of channels used for hopping can be set in the beginning, but by increasing the number of channels which can be used, the spectrum access becomes more robust (the probability of finding the channel which you want to use already occupied decreases)

- The time spent on each channel is called *Dwell Time* and can be set between *15 ms* and *250 ms*

- A node can send a packet *Unicast* or *Broadcast*. Each node has its own *Unicast Hopping Sequence* (Fig. 3.5). This sequence is used when two nodes communicate to each other (Fig. 3.6). When a broadcast transmission is wanted, a *Broadcast Schedule* is started by the *PAN coordinator*. This schedule contains the values of the *Broadcast Dwell Interval* and of the the *Broadcast Interval* and information about the *Broadcast Hopping Sequence*. All the devices follow the *Broadcast Hopping Sequence* during the *Broadcast Dwell Interval*. After this, until the end of the *Broadcast Interval*, each node follows its own *Unicast Hopping Sequence* (Fig. 3.7).

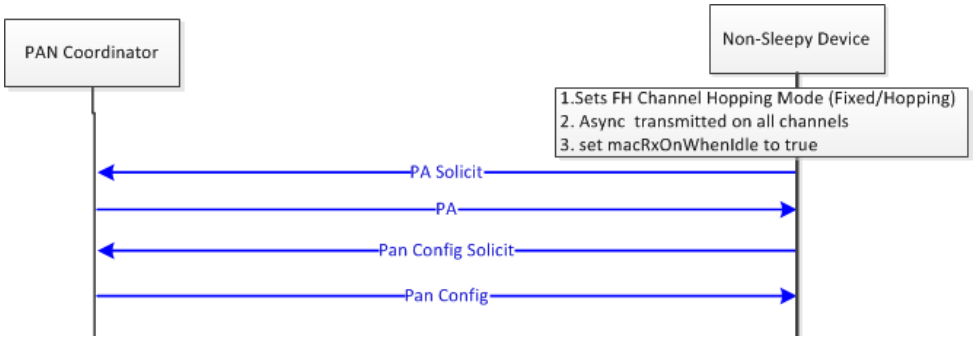


Fig. 3.3: The *Wi-SUN* joining procedure of a *non-sleepy* device for *Frequency Hopping* [47]

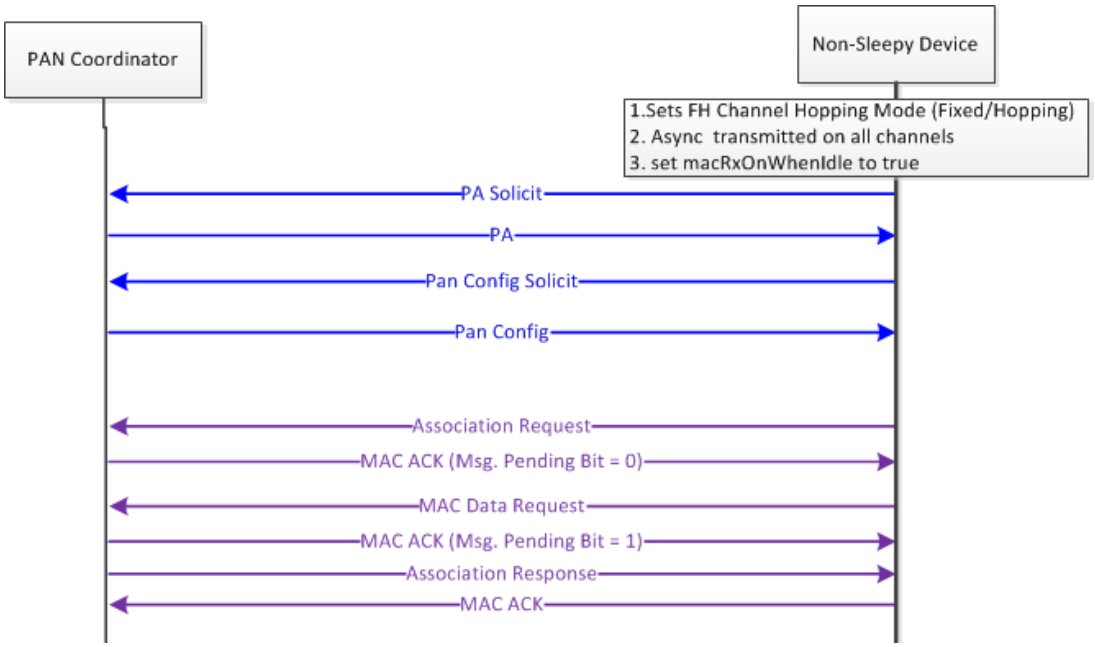


Fig. 3.4: The *Wi-SUN* joining procedure and the *TI 15.4 Stack* additional step (based on the *MAC Layer* association procedure described in the *IEEE 802.15.4* specifications) of a *non-sleepy* device for *Frequency Hopping* [47]

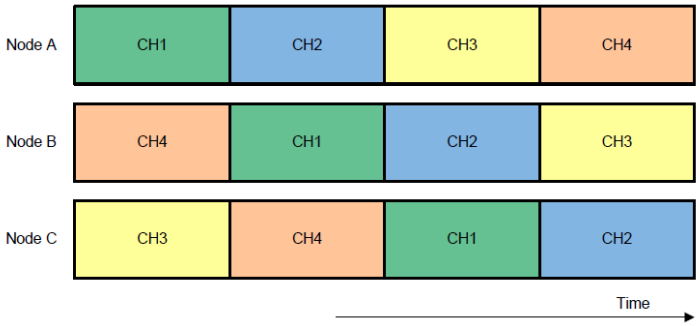


Fig. 3.5: The different *Unicast Hopping Sequences* over time of three Nodes [47]

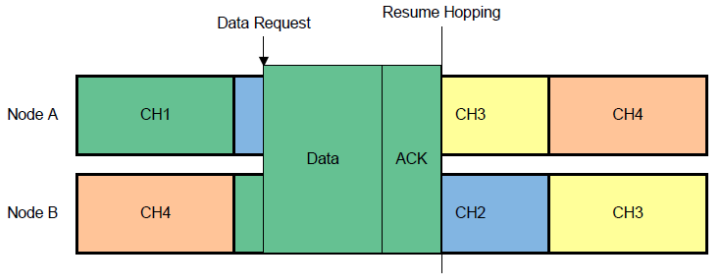


Fig. 3.6: The *Unicast* data exchange between two nodes. It is based on the *Receiver Directed Transmission* concept, when the transmitter node (*Node A*) transmits the data on the receiver's (*Node B*) frequency channel [47]

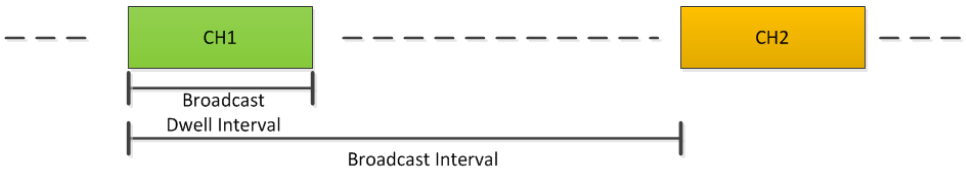


Fig. 3.7: Broadcast Channel Hopping Sequence for one node. During the *Broadcast Dwell Interval*, the node follows the *Broadcast Hopping Sequence*. After this, until the end of the *Broadcast Interval*, each node follows its own *Unicast Hopping Sequence* [47]

## 3.4 Carrier Sense Multiple Access (CSMA)

Characteristics: [86] [122]

- This method uses *LBT* to check if the channel is free. Multiple techniques exist for scheduling the following attempts if the channel is busy:
  - *1-persistent*: the node sends the packet as soon as the channel becomes idle. Though, if more nodes want to send a packet, they will all start to transmit at the same time (when the channel becomes idle), leading to a collision
  - *P-persistent*: the node sends the packet as soon as the channel becomes idle with a probability *P*. Otherwise, it waits for a fixed offset time
  - *Non-persistent*: after the channel is assessed to be busy, the node waits a random *Backoff time* and then verifies again the status of the channel
  - *CSMA/CD (Carrier-Sense Multiple Access with Collision Detection)*: the node sends the packet as soon as the channel becomes idle. While transmitting, the node listens to the channel to verify if any collisions occurred. If a collision was detected, the node stops transmitting, sends a jam signal and then starts again the whole procedure after a random *Backoff time* if the maximum number of attempts was not reached. This method is affected by the *Hidden Node* problem (Fig. 3.2).

- *CSMA/CR (Carrier-Sense Multiple Access with Collision Resolution)*: it is based on *CSMA/CD* with the concept of *Arbitration* added. Instead of stopping the transmission for all the nodes which collided, an arbitration scheme is applied in order to select only one node which can continue to transmit (e.g. bit arbitration used in *CAN (Controller Area Network)* protocol)
- *CSMA/CA (Carrier-Sense Multiple Access with Collision Avoidance)*: it is based on the *Non-persistent* method - after the channel is assessed to be busy, the node waits a random *Backoff time* and then verifies again the status of the channel. If the channel is idle, the node waits for a period of time called *Interframe Space* in order to allow the transmitted signals to reach all the nodes if another node started earlier to transmit. After this, if the channel is still idle, the packet is transmitted. The transmission is confirmed and not repeated if an *Acknowledgement* message is received from the receiver

## 3.5 Frequency Division Multiple Access (FDMA)

Characteristics [86]:

- The used frequency band is split into multiple sub-bands, each of them being assigned to a different data stream (uplink or downlink) from different connections
- It has the advantages of requiring little digital signal processing and simple synchronization. However, it is sensitive to fading and interference.
- It was used in the first generation (1G) of cell-phone systems

## 3.6 Time Division Multiple Access (TDMA)

Characteristics [86]:

- The time is divided into units which are further on divided into slots. Each device has a time slot allocated, only in which it can transmit. During its time slot, each node can use the whole bandwidth
- The advantages are a higher data rate and the possibility to listen during the periods of inactivity, in order to find out which is the better *Base Station* (if there are multiple) and to prepare the switching. However, temporal guards have to be used because the devices cannot decrease the transmission power to zero instantaneously and each time slot might require a new synchronization.

# Chapter 4

# Security

The continuous expansion of the *IoT* applications' range and of the hardware and software solutions for these applications comes with a big challenge: the security. In a *WSN*, the information which is sent from one node to another can be intercepted, jammed, modified or stolen much more easily than in a wired network. These are only some of the security related problems which can occur in a *WSN* and for which protection solutions have to be implemented in order to fulfill the requirements of trust, security and privacy [73]. Accomplishing these requirements is much more difficult for the embedded devices because they are very computationally-limited and also they have to be very aware regarding the power consumption in order to not affect the lifetime of the used batteries.

## 4.1 Security Requirements

Like any other network, a general *WSN* has multiple security requirements explained in [103]:

- *Data Confidentiality* - a message from the network should be understood only by the desired receiver
- *Data Integrity* - a message from the network should not be altered on the way from transmitter to receiver
- *Availability* - the services provided by a node should be available always
- *Data Freshness* - the messages transmitted in the network are new and not old resent messages (unless it is intended to)
- *Self-organization* - the dynamic nature of a *WSN* requires that the node self-organize continuously, not only for multi-hop routing but also to carryout key management

- *Secure Localization* - the position of each node should be able to be determined by different techniques than through location-type messages which can be manipulated by attackers
- *Time synchronization* - the security mechanisms, as well as other applications of *WSN* requires to be time-synchronized
- *Authentication* - a received message should be able to be verified that it came from the expected transmitter

## 4.2 Attacks

In [103], the attacks which can affect *WSNs* are divided into three main categories:

- *Attacks on secrecy and authentication* - examples: eavesdropping (understanding of the messages' content by the attacker), packet reply attacks, spoofing packets, camouflage
- *Attacks on network availability* - ofen referred as *Denial-of-Service (DoS)* attacks. These can affect every layer of a *WSN* (Fig. 4.1):
  - *Physical layer attacks*
    - jamming (interference with the radio frequency used by the *WSN*)
    - tampering (extraction of the cryptographic keys from the attacked node, tamper with its circuitry and modification of the program code)
  - *Link layer attacks*
    - continuously transmission of messages in order to generate collisions which lead to high delays (due to backoff times) and resource exhaustion
  - *Network layer attacks*
    - spoofed routing information
    - selective forwarding
    - sinkhole (by altering the routing information, a compromised node can be made to look more attractive to its neighbours)
    - sybil attack (in the same network, a node presents more than one identity)
    - wormhole (the attacker replays network messages over a low latency link between two parts of a network)
    - blackhole attack (a captured node falsely advertises good paths during the path-finding process)

    - grayhole attack (the malicious node intermittently drops data packets)
    - HELLO flood (an attacker sends with high transmission power an introducing packet, fooling a large number of nodes that they are within its neighbourhood)
    - acknowledgment spoofing
  - *Transport layer attacks*
    - flooding (an attacker sends repeatedly new connection requests)
    - de-synchronization (an attacker repeatedly spoofs messages to an end host, causing the host to request the retransmission of missed frames and try to recover from these errors instead on focusing on the real current messages)

- *Attacks against service integrity*

| Layer | Attacks | Defense |
|---|---|---|
| Physical | Jamming | Spread-spectrum, priority messages, lower duty cycle, region mapping, mode change |
| Link | Collision<br>Exhaustion<br>Unfairness | Error-correcting code<br>Rate limitation<br>Small frames |
| Network | Spoofed routing information & Selective forwarding<br>Sinkhole<br>Sybil<br>Wormhole<br>HELLO Flood<br><br>Acknowledgment flooding | Egress filtering, authentication, monitoring<br><br>Redundancy probing<br>Authentication, monitoring, redundancy<br>Authentication, probing<br>Authentication, packet leashes by using geographic and temporal info<br>Authentication, verify the bi-directional link authentication |
| Transport | Flooding<br>De-synchronization | Client puzzles<br>Authentication |

Fig. 4.1: Different attacks on *WSN's* layers and their countermeasures [103]

## 4.3 Cryptographic Methods

As for the regular networks, two types of cryptographic methods exist in *WSNs* [103]:

- *Asymmetric*: uses pair of keys: public keys and private keys. These methods are very computationally intensive and not recommended for low-capabilities embedded devices. *RSA* and *ECC* are the most popular methods from this category
- *Symmetric*: uses a single shared key between the two devices which communicate. It is less computationally intensive as the asymmetric methods but has to securely distribute the shared key between the communicating devices. *RC4*, *RC5*, *IDEA*, *SHA*-1 and *MD5* are the most popular methods from this category

## 4.4 Key Management Protocols

Key management protocols have to take care of generation, exchange, storage, usage, replacement and destruction of the cryptographic keys. Due to the limited capabilities of embedded devices, most of the key management protocols are using the shared keys. According to [103], these protocols can be divided into:

- *Centralized key scheme* - only one device controls the key-related activities, named *Key Distribution Center (KDC)*
- *Distributed key scheme* - different devices control the key-related activities, having the advantage of not having a single point of failure
- *Probabilistic key scheme* - the keys for each node are randomly chosen from a large key pool
- *Deterministic key scheme* - the keys for each node are deterministically chosen from a large key pool

## 4.5 Security for the *CC1350* microcontroller

In [68] and [45] it is specified that the *CC1350* microcontroller contains a dedicated *128 bit AES (Advanced Encryption Standard) hardware accelerator*. This module supports a *128 bit* key in hardware for encryption and decryption of the packets transmitted between two devices. It uses a symmetric algorithm, which means that the encryption and the decryption keys are identical. Based on this accelerator, the *CC1350*'s *SDK* includes the implementations of the *ECB (Electronic Codebook)* and *CCM (Counter with CBC-MAC)* (encryption and authentication) methods and supports also the *CTR (Counter)* (encryption), *CBC-MAC (Cipher Block Chaining - Message Authentication Code)* (authentication) and *CBC (Cipher Block Chaining)* methods.
Another important security feature of the *CC1350* microcontroller is the implementation of the *ECC (Elliptic Curve Cryptography)* core in the *ROM* in order to leave more *Flash* memory for the application [68]. These core functions can be used for *ECDH (Elliptic Curve Diffie-Hellman)*, *ECDSA (Elliptic Curve Digital Signature Algorithm)* and *ECJPAKE (Elliptic Curve Password Authenticated Key Exchange by Juggling)* techniques which are implemented within the *CC1350*'s *SDK* [55]. *ECDH* is a key agreement scheme which generates a shared secret and derived symmetric key between two devices over an insecure channel without providing authentication. *ECDSA* is a message authentication scheme which ensures message authentication and integrity between two devices. *ECJPAKE* is a key agreement scheme which establishes a secure channel over an insecure network without needing a public key infrastructure (requires

only to share a password offline).
Another valuable features of the *CC1350* microcontroller for ensuring security are the *True Random Number Generator* (built on *24* ring oscillators and a complex nonlinear combinatorial circuit based on *XOR* gates) and the unique *die ID* associated to every chip [48] [68].
The *TI 15.4 Stack* uses *AES 128 bit CCM* based encryption and authentication with shared keys, as part of the *IEEE 802.15.4e* security standard for the *MAC Layer*. In [123] and [30] it is specified that the *Network Layer Security* and the *Application Support Sublayer (APS)* of *Zigbee* uses also *AES* encryption and authentication with a key length of *128 bits*, as well as the standard *MAC Layer* security for the IEEE802.15.4e. *LoraWAN* uses also the *AES 128 bit* encryption algorithm but only for generating a stream of keys which are afterwards *XOR*-ed with the data, solution which is considered to be a vulnerability because it can be easily decrypted by an attacker [114]. So, the *CC1350*'s *AES hardware accelerator* can be successfully used for these communication protocols in order to improve the efficiency of the whole system.

# Chapter 5

# IoT Standards and Protocols

This chapter contains an overview of the most popular wireless communication standards, protocols and technologies used in *IoT* and not only. These solutions are compared taking into account: frequency used, topology, range, data rate, layers covered and overhead of each message. In multiple literature sources exist comparisons between *IoT* communication protocols and technologies but only for smaller subsets of them ([28], [101], [19], [33], [81]). Thus, by having an analysis of these solutions and by comparing them with the requirements for the considered application (Appendix A), a solid decision can be made: reuse an already existent complete communication solution, combine parts of several existent solutions or implement a solution from scratch.
This chapter contains also an overview of the most popular *Application Layer* protocols. In general, these can be added on top of any communication protocol for creating an interface between the end device and the network. These can provide different services which can bring valuable advantages for the implementation of the whole system.

## 5.1 TCP/IP (Transmission Control Protocol/Internet Protocol) model

Characteristics [27] [11] [76] [31]:

- It is also known as the *Internet Protocol Suite*
- It was developed as the networking solution for the wired global Internet
- Provides end-to-end data communication by defining how the data should be split in packets, addressed, transmitted, routed and received
- It defines 4 abstraction layers (Fig. 5.1):

- *The Network Access & Physical Layer* (the *Link Layer*) - focused on the communication within a single network segment, without any routers involved, considering the hardware involved (e.g. wires, radio), and the physical addressing protocols
- The *Internet Layer* - focused on the communication between interconnected networks, considering the routers which connects networks together. The *IP* is the most used protocol for this layer
- The *Transport Layer* - focused on providing end-to-end communication services to applications (e.g. same order delivery, reliability, flow control, multiplexing). The *TCP* and the *UDP* are the most used protocols for this layer
- The *Application Layer* - focused on providing messaging for the application level (e.g. *HTTP/S* is a popular application layer protocol adopted across the Internet)

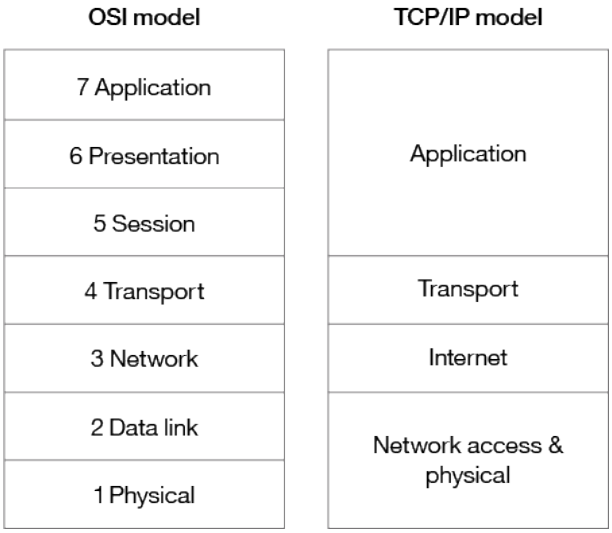


Fig. 5.1: The correspondence between the *OSI model* (left) and the *TCP/IP model* (right) [31]

### 5.1.1 IP (Internet Protocol)

Characteristics [27] [11] [76]:

- It is the main protocol for the *Internet Layer* of the *TCP/IP model*
- Its focus is to deliver data (packets) between devices on different interconenected networks
- It is universally-addressed in order to be able to send data between two points which might not be in the same network
- It does not use acknowledgement messages for confirming the correct receiving of a packet

- It is a connectionless protocol, meaning that it does not set a connection between the sending and the receiving point before delivering a packet

### 5.1.2 TCP (Transmission Control Protocol) and UDP (User Datagram Protocol)

Characteristics [27] [11] [76]:

- Are the main protocols for the *Transport layer*
- Define how communication channels are created across a network and how a message is split into several smaller packets on the sending side and assembled back on the receiving side
- *UDP* sends the packets without applying any error-correction techniques, thus decreasing the latency. In contrast, *TCP* ensures reliability by tracking the packets such that no data is lost or corrupted in transport

### 5.1.3 IPv4

Characteristics [120]:

- It is the 4th version of the *IP* protocol used from the beginning of the Internet
- Uses *32 bits* for the Internet addresses. Due to the rapid increase in the number of Internet-enabled devices (e.g. smartphones, tablets, laptops, development boards), the main problem encountered by *IPv4* is address exhaustion

### 5.1.4 IPv6

Characteristics [120]:

- It is the 6th revision of the *IP* protocol
- The main feature is that it enlarges the address space by using *128 bits* addresses
- The disadvantage is that it is not backward compatible with *IPv4*. Thus, it can be said that *IPv6* has defined a parallel and independent network
- The transition to *IPv6* will be a slow process, due to the large number of *IPv4* devices (including expensive devices like servers and routers) which have to be changed

### 5.1.5 6LoWPAN (IPv6 over Low-Power WPANs)

Characteristics [33] [28]:

- Represents the specifications for mapping the services required by the *IPv6* over *LoWPANs* (Low-Power Wireless Personal Area Networks). Thus, an *IPv6* network can be maintained for the limited processing capable devices which should be connected to Internet
- It works as an adaptation layer between *IPv6* packets and *IEEE 802.15.4* networks
- Includes:
  - header compression (for reducing the *IPv6* header from *40 Bytes* to *2 Bytes*)
  - fragmentation (for splitting the *IPv6 1280 Bytes* packets to *IEEE 802.15.4 127 Bytes* packets)
  - auto-configuration of *IPv6* addresses

### 5.1.6 uIP (Micro IP)

Characteristics [113]:

- It is an open-source implementation of the *TCP/IP* network protocol, dedicated for *8* and *16 bits* microcontrollers
- It minimizes the required code and data memory footprints
- It is designed for computer systems without an operating system, thus no tasks are used
- It can handle multiple concurrent *TCP* and *UDP* connections (without using tasks)

### 5.1.7 nanoIP

Characteristics [14] [106]:

- It is an open-source implementation of a networking protocol stack, dedicated for control, automation and sensor networks
- It does not perform any routing, so the embedded devices connected are visible to the global Internet through a Gateway
- *MAC* (Medium Access Control) addresses are used instead of *IP* addresses
- A range of compact compatible tools were developed: *nanoUDP*, *nanoTCP*, *nanoHTTP*, *nanoSLP*

### 5.1.8 TCP/IP in IoT

Due to the fact that the *IoT* implies a large number of differences than the standard wired Internet network (e.g. *IoT* devices have limited computational capabilities, have to operate over long time on batteries, sometimes they are in sleep-mode), the *TCP/IP* is considered to bring some difficulties when it has to be applied to the *IoT* world. These issues, their solutions and an adapted architecture are described in [105].

- Problems at the *Network* Layer
  - Small MTU (Maximum Transmission Unit) - most of the *IoT* physical layers restrict the frame size to a lower value than the one required by *IPv6*. *6LoWPAN* introduces mechanisms for header compression and link-layer fragmentation (see Section 5.1.5) in order to overcome this problem
  - Multi-link subnet - *IoT* networks may adopt a mesh topology for covering larger areas without placing multiple base stations. Though, the initial *IP* addressing architecture is not considering that two devices can communicate by using other devices in between and not routers. Solutions for mesh network routing at the link layer or at the network layer were developed: *IEEE 802.15.5* (for mesh networks based on *IEEE 802.15.4* links) and *RPL* (IPv6 Routing Protocol for Low-Power and Lossy Networks), respectively.
  - Multicast efficiency - multicast represents a challenge for *IoT* mesh networks because the receivers may have different data transmission rates, some of the receiver may be switched to sleep-mode for energy saving and they might miss the sent packet and a large number of nodes which are in sleep-mode might be woken up when sending a multicast packet over multiple hops along multiple paths. One solution is the *IPv6 Neighbor Discovery* optimization for *6LoWPAN*
- Problems at the *Transport* Layer - a large number of *IoT* protocols chose to implement transport functionalities into the *Application* layer and use *UDP* as the *Transport* layer protocol instead of *TCP*, due to its disadvantages (a long-lived connection cannot be maintained between devices which are switching to sleep-mode frequently; for short messages - the regular case in *IoT* - it is unnecessary to maintain a long-lived connection; the handshaking protocol may introduce unwanted large delays for time-critical applications)
- Problems at the *Application* Layer
  - Security - maintaing the channel-based security in *IoT* is difficult (due to the overhead implied, the memory consumption on both ends of the connection - which are limited capabilities devices - and the difficulty of ensuring the safety for the data

when it gets out of the channel). So, object-based security can be used instead (by securing the data, rather than the channel, using signatures and encryptions)

– Caching - by moving the caching functionality from the *Application* layer to the *Network* layer (due to the intermittent connectivity caused by entering in the sleep-mode of the nodes and to the unstable network environment), efficiency and flexibility is achieved for the *IoT* case

The architecture proposed in [105] (Fig. 5.2) is based on the *"everything over REST"* paradigm, instead of the classical *"everything over IP"*. Thus, the *REST*-related components (e.g. URI-based communication, caching, object security) are placed in the core of the *Network* layer.

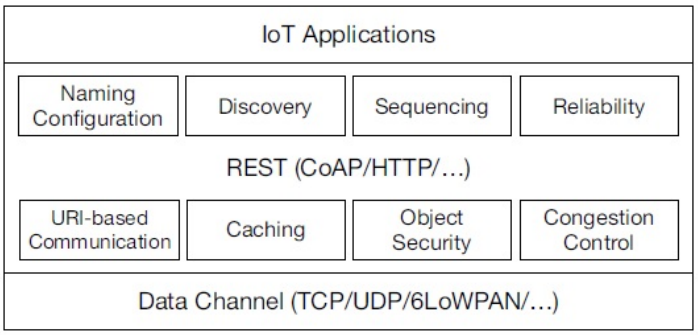


Fig. 5.2: The IoT stack from the application's perspective proposed in [105]

## 5.2 Bluetooth

Characteristics [10] [79] [12] [81] [101]:

- **Standard used**: IEEE 802.15.1
- **Frequency**: 2.4 GHz
- **Topology**: Piconet and Scatternet
- **Range**: Up to 100 meters
- **Data Rate**: Up to 3 Mbps
- **Other characteristics**:
  - Used for *WPANs* in order to replace the cables for computer peripherals (e.g. keyboard, mouse, printer)

- Uses two connectivity topologies: *Piconet* (consisting of *1* Master, up to *7* Active Slaves and up to *255* Inactive Slaves; All the members are synchronized with the Master's clock and hopping sequence) and *Scatternet* (*2* or more interconnected *Piconets* through common devices - a member of one *Piconet* (Master or Slave) can participate as Slave in another *Piconet*)
- Splits the bandwidth in *79 channels* with *1 MHz bandwidth* each which are used for the *Frequency Hopping* technique
- The Bluetooth layer stack structure does not follow the *OSI* or *TCP/IP* models. Though, its layers cover all the layers of the *OSI* model (Fig. 5.3).

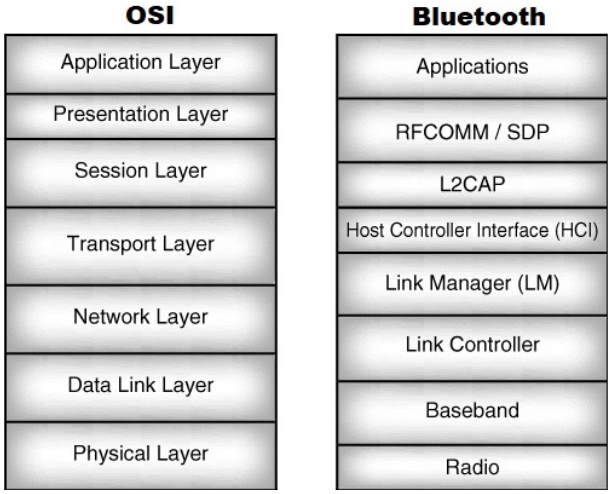


Fig. 5.3: The correspondence between the *OSI model* (left) and the *Bluetooth stack* (right) [12]

### 5.2.1 Bluetooth Low Energy (BLE)

Characteristics [9] [10] [89] [79] [101] [34]:

- **Standard used**: IEEE 802.15.1
- **Frequency**: 2.4 GHz
- **Topology**: Point-to-Point, Broadcast, Mesh
- **Range**: Up to 400 meters
- **Data Rate**: Up to 2 Mbps
- **Other characteristics**:
  - Introduced as the *version 4.0* of Bluetooth

  - Used for *WPANs* for applications which require lower power consumption, higher range and lower data rate (e.g. healthcare, fitness, beacons, sensors)
  - It is not backward compatible with the standard *Bluetooth* because the modulation technique used is different
  - It can handle unlimited number of connected nodes
  - Its power consumption varies between *100* and *2* times smaller than Bluetooth, depending on the use case
  - *BLE Beacons* are devices which can send *BLE signals* at a certain time interval, used for localization, navigation and tracking. Various protocols exist, one of the most popular being *Eddystone* developed by *Google*. According to this protocol, the payload can be of *4* types, according to what it contains: *UID* (Unique ID), *URL*, *TLM* (Telemetry data - for example temperature or battery voltage) and *EID* (Ephemeral Identifier - an identifier which changes at a certain time interval).

## 5.3 IEEE 802.15.4

Characteristics [28] [33]:

- Designed for *LR-WPANs* (Low-Rate WPANs)
- Defines the *PHY* (Physical) layer and the *MAC* (Medium Access Control) layer
- Uses the *868 MHz* (used in Europe; *20 Kbps* standard data rate; *BPSK* modulation), *915 MHz* (used in North America; *40 Kbps* standard data rate; *BPSK* modulation) and *2.4 GHz* (used worldwide; *250 Kbps* standard data rate; *O-QPSK* modulation) frequency bands
- Uses *CSMA/CA* (Carrier-Sense Multiple Access with Collision Avoidance) protocol for reducing the number of possible collisions
- Two types of devices are involved in a network: one *FFD* (Full-Function Device) (usually it is the coordinator of the network) and multiple *RFD* (Reduced-Function Device) (which can talk only with *FFDs*)
- Standard network topologies are: Star, Tree and Mesh

### 5.3.1 Zigbee

Characteristics [81] [112] [101] [6] [123]:

- **Standard used**: IEEE 802.15.4

- **Frequency**: 868 MHz, 915 MHz, 2.4 GHz (depending on the region)
- **Topology**: Star, Tree, Mesh
- **Range**: Up to 100 meters (for 2.4 GHz)
- **Data Rate**: Up to 250 Kbps
- **Other characteristics**:
  - Used for *LR-WPANs* (e.g. home security, lightning control, wireless sensor networks)
  - *IEEE 802.15.4* defines the *PHY* layer and the *MAC* layer, whereas Zigbee defines the upper layers (*Network* layer and *Application* layer)
  - Two types of devices are involved in a network: one *FFD* and multiple *RFD*
  - It can handle more than *65000* connected devices
  - It is less expensive than *Bluetooth* and *Wi-Fi*
  - It uses *DSSS* spread spectrum modulation
  - It includes complex security techniques for ensuring key establishment, secure networks, key transport and frame security. These services are implemented for the *Network Layer* (*128 bit AES* encryption) and for the *Application Support Sublayer (APS)*, a sublayer of the *Application Layer*
  - With the version *PRO 2015*, *Zigbee* supports for the first time the *Sub-1 GHz* frequency band
  - With the version *PRO 2017*, *Zigbee* becomes a mesh network capable of operating in *2* frequency bands simultaneously (*Sub-1 GHz* and *2.4 GHz*). This is an advantage for creating large *IoT* networks across large areas and buildings
  - *The Zigbee Alliance's Dotdot* Application layer is an universal and standard language for smart devices to communicate over any network (e.g. Wi-Fi, Bluetooth, Thread, Zigbee)

### 5.3.2 Thread

Characteristics [94] [93] [109] [5]:

- **Standard used**: IEEE 802.15.4, 6LoWPAN
- **Frequency**: 2.4 GHz

- **Topology**: Mesh
- **Range**: Up to 30 meters
- **Data Rate**: Up to 250 Kbps
- **Other characteristics**:
  - It is an *6LoWPAN* networking protocol designed for low-power *IEEE 802.15.4* mesh networks
  - It is indented to be used in and around the houses and the buildings for connecting devices between them and directly to the Internet and to the Cloud
  - It is described as using ”banking-class” encryption for ensuring a high level of security
  - Can be applied directly and with no effort to most of the wireless devices based on the *IEEE 802.15.4*
  - Due to the compatibility with *IPv6*, *Thread* can work with protocols based on *IP* like *Wi-Fi*, *Ethernet* or *4G LTE*, in contrast with *Zigbee* and *Z-Wave*
  - It is designed to not have a single point of failure
  - It can handle up to *300* connected devices
  - Being part of the *Dotdot* Application Layer (see Section 5.3.1),*Thread's* advantage of being able to connect over IPv6 networks can be used by other protocols, like *Zigbee*

### 5.3.3 WirelessHART

Characteristics [82]:

- **Standard used**: IEEE 802.15.4, IEC 62591
- **Frequency**: 2.4 GHz
- **Topology**: Mesh
- **Range**: Up to 30 meters indoor and 100 meters outdoor
- **Data Rate**: Up to 250 Kbps
- **Other characteristics**:

  - It is a wireless sensor networking technology based on the *HART* (Highway Addressable Remote Transducer Protocol). HART is a hybrid analog - digital industrial automation protocol which can communicate over *4 - 20 mA* analog instrumentation current loops, sharing the wires used by the analog host system
  - It supports the *HART* technology by using *Adapters* which interfaces the classical *HART* devices with the *WirelessHART* network
  - It is based on the *IEEE 802.15.4 PHY* layer, but new *Data Link*, *Network*, *Transport* and *Application* layers are specified
  - It is a *TDMA* based network, requiring that all the devices are time synchronized
  - Uses *Frequency Hopping* technique for reducing the interferences over the *2.4 GHz* frequency band
  - Provides end-to-end and hop-to-hop security measures by using payload encryption and message authentication

### 5.3.4 TI 15.4 Stack

Characteristics [59] [36]:

- **Standard used**: IEEE 802.15.4e/g (e - review for industrial applications (it adds mechanisms as time slotted access, multichannel communication and channel hopping), g - review for smart utility networks)
- **Frequency**: 433 MHz, 868 MHZ, 915 MHz (depending on the region)
- **Topology**: Star
- **Range**: Up to 20 kilometers
- **Data Rate**: Up to 200 Kbps
- **Other characteristics**:
  - It is a wireless protocol stack based on the *IEEE 802.15.4* PHY and MAC layers, developed by *Texas Instruments*
  - It includes *Frequency Hopping* technique for reducing the interference
  - It allows changes in the *PHY* layer (*Long Range Modes* for improved ranges but lower data rates (e.g. *625 bps*, *5 Kbps*))
  - The medium access is done using the *CSMA/CA* technique

  - It has *Synchronous* (Beacon Enabled Mode) and *Asynchronous* (Non Beacon Mode) operation modes for different traffic profiles
  - Includes a security layer based on *AES* with pre-shared key
  - Supports up to *150* connected nodes

## 5.4 IEEE 802.11

Characteristics [20] [37]:

- **Frequency**: initially it was using only the 2.4 GHz frequency band, but later reviews of the standard added the 900 MHz, 3.6 GHz, 5 GHz and 60 GHz frequency bands
- **Topology**: Point-to-Point, Star
- **Range**: Indoor ranges between 3 and 70 meters, outdoor ranges between 100 and 5000 meters, depending on the protocol's version
- **Data Rate**: Between 1 Mbps and 20 Gbps, depending on the protocol's version
- **Other characteristics**:
  - It is a standard which defines the *PHY* and the *MAC* layers for wireless connectivity for fixed, portable and moving stations within a local area
  - It is the base of the modern WLANs (Wireless Local Area Networks)
  - It uses different modulation techniques (e.g. *OFDM*, *FHSS*, *DSSS*), depending on the protocol's version

### 5.4.1 Wi-Fi

Characteristics [23]:

- It is a trademark of *Wi-Fi Alliance* organization, which can be used by those devices which respect the *IEEE 802.11* standard and are completely interoperable with other devices based on the same standard
- In order to obtain the *Wi-Fi* certification, a device must pass a set of "gold standards" tests, based on some of the *IEEE 802.11* specification but not all of them. Although, there are some requirements which are added to these tests which are not specified by *IEEE 802.11* (Fig. 5.4)

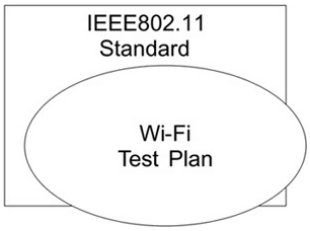


Fig. 5.4: The correspondence between the IEEE 802.11 standard and WiFi [23]

## 5.5 IEEE 802.15.3a

### 5.5.1 UWB (Ultra-Wideband)

Characteristics [95] [81] [77]:

- **Standard used**: IEEE 802.15.3a
- **Frequency**: between 3.1 GHz and 10.6 GHz
- **Topology**: Piconet
- **Range**: Up to 10 meters
- **Data Rate**: Up to 480 Mbps
- **Other characteristics**:
  - It is a short-range wireless communication technology with a very high data rate which can sustain audio and video delivery in home networking
  - It is based on the idea of sending short signal pulses over a broad spectrum, using multiple channels at once. Thus, no new and unoccupied frequencies are used, but the old ones are reused
  - It is still a technology under development but it has a great potential

## 5.6 IEEE 802.16

- It is a standard for development and deployment of broadband WMANs (Wireless Metropolitan Area Networks)

### 5.6.1 WiMAX (Worldwide Interoperability for Microwave Access)

Characteristics [119] [26]:

- **Standard used**: IEEE 802.16
- **Frequency**: 2.5 GHz, 3.5 GHz, 5.8 GHz
- **Topology**: Point-to-Multipoint, Mesh
- **Range**: Over 30 kilometers
- **Data Rate**: Up to 1 Gbps, depending on the protocol's version
- **Other characteristics**:
  - It was designed as an alternative to cable and DSL connections
  - The main advantages are the high range and the high data rate. Thus, it can be used as a portable mobile broadband connectivity across cities, to provide Internet connectivity or for smart grids and metering
  - Its *MAC* is connection-oriented (it is based on a scheduling algorithm run by the Base Station), in contrast to *Wi-Fi*'s *MAC* which is based on the *CSMA/CA* (Carrier Sense Multiple Access with Collision Avoidance) protocol (a station can transmit anytime, after it checked if the carrier is free)

## 5.7 Sub-1 GHz

Characteristics [69]:

- Represents the ISM (Industrial, Scientific and Medical) frequency bands below *1 GHz* - typically *433 MHz*, *868 MHz* and *915 MHz*
- It has multiple advantages over the other frequency bands (e.g. *2.4 GHz*):
  - Longer range: can reach *100 kilometers* (*20 kilometers* only using a coin cell battery)
  - Lower power: several years of operation using a single coin cell battery can be achieved, depending on the application and the used hardware
  - Less interference: these frequencies are yet not very used, so not many devices are transmitting on these bands. Also, lower frequencies can pass through walls and weave between obstacles better

- Due to its low popularity at the moment, there are not any extremely popular standards (like *Bluetooth* is for *2.4 GHz* band, for example), only proprietary solutions
- It can be used for *IoT* applications, in industry or in households, where data from sensors placed in large areas has to be collected in a simple manner (with one hop, without using intermediary points)

### 5.7.1 SigFox

Characteristics [98] [101] [3]:

- **Standard used**: -
- **Frequency**: Sub-1 GHz (868 MHz, 902 MHz, depending on the region)
- **Topology**: Star
- **Range**: Up to 50 kilometers in rural areas, and up to 10 kilometers in urban areas
- **Data Rate**: 100 bps
- **Other characteristics**:
  - It provides end-to-end *LPWAN* (Low-Power Wide Area Network) connectivity solution together with some partner network operators
  - The *SigFox Network Operators* provides proprietary base stations (based on standard hardware chips on which the *SigFox* software stack is installed)
  - The end devices (i.e. nodes) uses *UNB* (Ultra Narrow Band) technology to connect to these base stations, thus the bandwidth is utilized efficiently, the noise has a very low level, the receiver's sensitivity is higher and the power consumption is very low. Though, the data rate is very low, only *100 bps*
  - It initially supported only uplink communication, but later downlink was added. Due to the regional regulations on use of license-free spectrum, the number and size over the uplink are limited to *140 12-Bytes* messages per day, and *4 8-Bytes* downlink messages per day
  - Thus, the target of *SigFox* are the low throughput applications
  - Each *SigFox* base station can handle up to a million connected objects

### 5.7.2 LoRa (Long Range)

Characteristics [98] [7]:

- **Standard used**: -
- **Frequency**: Sub-1 GHz (433 MHz, 868 MHz, 915 MHz, depending on the region)
- **Topology**: Star-of-Stars
- **Range**: Up to 25 kilometers
- **Data Rate**: Up to 50 Kbps
- **Other characteristics**:
  - It is a proprietary wireless communication system, which enables very long range transmissions, consuming very low power.
  - It is composed of two parts: the *PHY* layer and the *MAC* protocol (LoRaWAN)
  - The modulation technique used by the *PHY* layer is a patented technology (*Chirp Spread Spectrum* [104]) from *Semtech*
  - *LoRaWAN* is an open standard
  - The messages transmitted can be received by multiple base stations which are in the coverage are, thus creating a *star-of-stars* topology. Also, the number of successfully received messages is increased
  - The localization of the transmitting device is done when multiple base stations receive the same message
  - *LoRaWAN* defines 3 different classes of end-devices, each of them with different downlink and power characteristics:
    - *class A* - longest lifetime but highest latency
    - *class B* - schedule donwlink receptions from the base station at certain intervals
    - *class C* - continuously listen and receive downlink transmissions with the lowest latency
  - *LoRaWAN* implements a security level for authenticating the devices and ensure the data privacy

## 5.8 Cellular network

Characteristics [18] [100] [35] [97] [102]:

- **Frequency**: 900 MHz, 1.8 GHz, 1.9 GHz, 2.1 GHz
- **Topology**: One-to-One (between the mobile transceiver and the base station), Broadcast
- **Range**: Up to 200 kilometers (for HSPA)
- **Data Rate**: Around 1 Gbps (for 5G)
- **Other characteristics**:
  - It is a communication network which covers large areas by splitting them into cells, each of them served by at least one fixed base stations. Only the last segment, the one between the base station and the end-devices is wireless
  - The main advantage of using the cellular network in *IoT* is given by the already exiting infrastructure
  - Though, using this infrastructure can be costly, depending on the amount of data sent / received
  - Due to the division into cells, a bigger volume of data can be transmitted at the same time because the same frequency can be reused in different cells and lower transmitting power has to be used because the distance to a base station will not be bigger than the cell size
  - Different technologies were developed along the years (e.g. *GSM*, *CDMA*, *GPRS*, *EDGE*, *UMTS*, *HSPA*, *LTE*), each of them with different characteristics (e.g. range, data rate, modulation techniques, digital / analog)
  - Versions for *Iot* were also developed, like *LTE-M* (Long Term Evolution (4G) category M1) (for *IoT* devices to connect directly to a *4G* network, without a gateway) and *NB-IoT* (Narrowband-IoT) (it can operates within *GSM* or *LTE* spectrum only by doing a software upgrade, so no new frequencies have to be allocated, or it can use independently licensed bands; it consumes less power than *LTE-M*, has a lower data rate but it can pass through obstacles better) or *EC-GSM-IoT* (Extended Coverage-GSM-IoT)

## 5.9 Other communication protocols / technologies

### 5.9.1 Z-Wave

Characteristics [115] [28] [33] [101]:

- **Standard used**: -
- **Frequency**: 868 MHz, 915 MHz, 2.4 GHz (depending on the region)
- **Topology**: Mesh
- **Range**: Up to 30 meters
- **Data Rate**: Up to 40 Kbps
- **Other characteristics**:
  - It is a low-power wireless communication protocol used for applications based on small messages (maximum *64 Bytes* payload) over small distances (e.g. household appliance control, wearable sensors, fire detection)
  - It defines three layers: *Radio* (similar to the *PHY* layer), *Network* (*MAC*, *Transport* and *Routing* sublayers) and *Application*
  - It can handle up to *232* connected devices
  - The number of hops of a message is limited to maximum *4*

### 5.9.2 INSTEON

Characteristics [33]:

- **Standard used**: -
- **Frequency**: 915 MHz for RF, 131.65 KHz for power line
- **Topology**: Dual-Mesh
- **Range**: RF range up to 45 meters
- **Data Rate**: Up to 38.4 Kbps
- **Other characteristics**:
  - It is a dual-mesh communication technology based on RF and power line, designed for home automation applications

  - Each device can communicate with the others through RF, power line, or both
  - All devices are peers, meaning that each device can send, receive or repeat a message
  - Due to the mesh topology, the sending of a message between 2 devices which are not within the same range can be achieved by a multihopping technique. Each device retransmits every message received if they are not the recipient, but maximum *4* hops is allowed for each message
  - The devices not connected to the power lines transmit messages asynchronously, whereas the ones connected to the power lines transmit messages synchronously, based on time slots defined by the power line's zero crossings
  - Each device has its own unique ID which is required for adding the device to a network. Alternatively, a device can be added to a network by pressing a specific button, requiring the physical posession of the device. Moreover, the security is enhanced by using encrypted messages (only for the application which are set to encrypt the messages, like the door locks)
  - A very large number of devices can be added to the network (on the order of thousands)

### 5.9.3 Wavenis

Characteristics [33]:

- **Standard used**: -
- **Frequency**: 433 MHz, 868 MHZ, 915 MHz (depending on the region), 2.4 GHz
- **Topology**: Star, Tree
- **Range**: Up to 200 meters indoor and 1 kilometer outdoor
- **Data Rate**: Up to 100 Kbps
- **Other characteristics**:
  - It is a wireless protocol stack designed for ultra-low power consumption and ultra-long range for controlling and monitoring applications in various environments, including home automation
  - It defines the *PHY* layer, the *Data Link* layer and the *Network* layer. On top of the *Wavenis* stack, any application stack can be placed

  - The network layer specifies a *4 level* virtual hierarchical tree, where the root node is usually the gateway. A new device which joins a *Wavenis* network broadcasts a request message for finding a parent, specifying a sufficient *QoS* (Quality of Service) value. This value is computed by the potential parents based on the values of *RSSI* (Received Signal Strength Indicator) and battery level and on the number of devices already attached to the device

### 5.9.4 EnOcean

Characteristics [77]:

- **Standard used**: ISO/IEC 14543-3-10
- **Frequency**: 315 MHz, 902 MHz, 928 MHz, 868 MHz
- **Topology**: Point-to-Point
- **Range**: Up to 30 meters indoor and 100 meters outdoor
- **Data Rate**: Up to 125 Kbps
- **Other characteristics**:
  - It is a wireless technology which uses energy harvesting for generating the required energy from the environment (e.g. ambient light, temperature), thus increasing the lifetime of the battery-less devices
  - It is more energy efficient than other wireless technologies (e.g. *Bluetooth*, *Zigbee*)
  - It is used for home and industrial automation
  - It defines the *PHY*, *Data Link* and *Networking* layers

### 5.9.5 NFC (Near Field Communication)

Characteristics [19] [101]:

- **Standard used**: ISO/IEC 14443 A-2, ISO/IEC 18092
- **Frequency**: 13.56 MHz
- **Topology**: One-to-One
- **Range**: Up to 10 centimeters
- **Data Rate**: Up to 424 Kbps

- **Other characteristics**:
  - It is a short range wireless communication technology, designed for mobile phones services like payment, loyalty applications and access keys but also for data exchange
  - It is a half-duplex protocol which ensures a very high level of security (also due to the short communication distance)
  - The communication takes place only between 2 devices: the *initiator* (starts and guides the data exchange) and the *target* (responds to the data exchange request)
  - There are 2 modes of operation: *active* (both devices use their own energy) and *passive* (the target uses the energy from the request signal sent by the initiator)
  - There are 3 types of NFC devices: *NFC mobile*, *NFC tag* and *NFC reader*

### 5.9.6 RFID (Radio Frequency Identification)

Characteristics [77] [101]:

- **Standard used**: Different, regarding various topics (e.g. animals identification, items identification)
- **Frequency**: Different, between 125 KHz and 5.8 GHz
- **Topology**: One-to-One, One-to-Many, Many-to-One
- **Range**: Up to 200 meters, depending on the frequency
- **Data Rate**: Up to 4 Mbps, depending on the frequency
- **Other characteristics**:
  - It is a short range bi-directional wireless communication technology, designed for identifying and tracking tags attached to objects, people, animals
  - There are 2 types of entities: *readers* and *tags*
  - The tag stores the identification information. It can be *active* (has a battery) or *passive* (does not have a battery; uses the energy from the interrogation signal)
  - The main disadvantage is the security - any reader can read any tag (if the tag is passive, then more complex processing can be done and a higher level of security can be achieved)

## 5.10 Application Layer Protocols

The Application Layer is composed not only on the main application but also on the management functionalities, like the ones for the network [2]. This layer represents the interface between the end device and the network. It also provides various services and defines a set of protocols for message passing at the application level. By using a standard *Application layer* protocol, the inter-operation between different devices can be ensured.

### 5.10.1 CoAP (Constrained Application Protocol)

Characteristics [28] [91]:

- It is an *Application* layer protocol, designed for the constrained devices and the constrained networks used in IoT
- It defines a web transfer protocol based on *REST* (Representational State Transfer) on top of *HTTP* functionalities (it is considered to be the lightweight version of HTTP with support for REST)
- It can be used:
  - between devices on the same constrained network
  - between devices and general nodes on the Internet
  - between devices on different constrained networks joined by the Internet
- It can run on the most devices which support *UDP* or *UDP analogue*
- It has 2 sub-layers:
  - The *messaging* sub-layer: detects duplications and provides reliable communications over *UDP*
  - The *request/response* sub-layer: handles the *REST* communications
- It is a secure protocol because it is build on top of *DTLS* (Datagram Transport Layer Security)
- It does not require a broker server like *MQTT*

### 5.10.2 MQTT (Message Queue Telemetry Transport)

Characteristics [28] [91] [87] [38]:

- It is a lightweight *Application* layer protocol, designed for an efficient asynchronous distribution of information between remote devices where a small code size is required (due to the limited capabilities of the devices) or small messages are required (due to the fact that the network bandwidth is limited)
- It is used when the data is not delivered based on address but based on content and interest
- It is based on the *Client-Server* and *Publish-Subscribe* communication models
- It is built on top of *TCP/IP* protocol
- *MQTT* defines 3 types of elements:
  - *subscriber* (a device which is interested in receiving information about a specific topic)
  - *publisher* (a device which generates information on different topics)
  - *broker* (receives the information generated by the publishers and redirects it only to those subscribers which are interested in that topic; ensures the security of the network by verifying the authorization of the publishers and subscribers)
- The *clients* are the *publishers* and the *subscribers* and the *server* is the *broker*. A *client* can be at the same time *publisher* and *subscriber*
- The *broker* can perform filtering of the received messages based on subject, content and type. It is also responsible for the authentication and authorization
- One advantage of *MQTT* is that the *publishers* and the *subscribers* are decoupled by time and space
- Three *QoS* levels regarding the messages sent from the publisher to the broker and from the broker to the client are defined:
  - *At most once* (*QoS 0*) - no guarantee of delivery (used when the loss of some messages is acceptable)
  - *At least once* (*QoS 1*) - guarantees that a message is received at least one time by the receiver (used when each message is needed and duplicates can be handled)
  - *Exactly once* (*QoS 2*) - guarantees that a message is received exactly once by the receiver (used when each message is needed and duplicates cannot be handled)

- The *broker* can use a persistent session for each client who wants this, by storing all the subscription of the client, all the *QoS 1* or *2* messages that the client did not confirm yet, all the new *QoS 1* or *2* messages that the client missed while it was offline, all the *QoS 2* messages received from the client that were not acknowledged completely yet

- Besides the packets which containt the acquired values from the sensor, *MQTT* includes acknowledgement, *Ping* (for signalizing that a Node is still alive, even if it didn't sent any messages recently) and *Last Will messages* (the messages sent by the Broker to all the clients who subscribed to these messages when a connection with a Node is unexpectedly interrupted)

### MQTT-SN

- The *MQTT-SN* version is designed for sensor networks. It was developed to be as close as possible to *MQTT* but to overcome the disadvantages brought by the usage of wireless communication (e.g. low bandwith, short messages)

- It is not based on the *TCP/IP* model but it can be used on top of any communication protocol (e.g. *Zigbee*, *Z-Wave*, *BLE*).

- *MQTT-SN* contains in addition *Gateways* which are connecting the *clients* and the *broker*. The connections between the *clients* and the *gateways* are done using *MQTT-SN* and the connections between the *gateways* and the *broker* are done using *MQTT*. So, it can be said that the *gateways* are converting *MQTT* messages in *MQTT-SN* and vice-versa. The *gateway* can be *transparent* (for each *MQTT-SN client* there is a separate *MQTT* connection with the *broker*) or *aggregating* (all the *MQTT-SN* connections converge to only one *MQTT* connection with the *broker*) (Fig. 5.5)

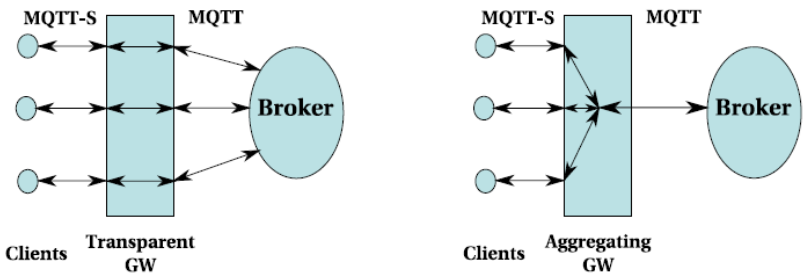


Fig. 5.5: The transparent and the aggregating gateway in MQTT-SN [40]

### 5.10.3 XMPP (Extensible Messaging and Presence Protocol)

Characteristics [28] [8] [29]:

- It is an *Application* layer protocol for *MOM* (message-oriented middleware) based on XML (Extensible Markup Language)
- It is designed for real-time applications (e.g. Instant Messaging)
- It uses the *Client-Server* architecture based on *TCP*. When a client connects to the a server, 2 *XML* streams are opened: one from the client to the server and one from the server to the client
- The most basic unit of communications is the *XML stanza*. An *XML stanza* contains 3 components (which can have different attributes):
  - *presence* (used to advertise the network availability to the other entities connected)
  - *message* (used to send data between *XMPP* entities)
  - *iq* (Info/Query) (used to get some information from the server or to apply some settings to the server))
- It is secure
- It allows other applications to be added on top of the core protocol
- *XMPP-IoT* is the extension of *XMPP* for *IoT*, designed for machine-to-people and machine-to machine communication, not only people-to-people as in the case of *XMPP*. Some improvements of this version are: *scalability* (the *IoT* involves a large and variable number of connected devices) and *autonomy* (in *IoT* some devices are not operated by humans, so they need to be autonomous (e.g. delegation of computationally expensive tasks from limited devices to more capable servers)).

### 5.10.4 AMQP (Advanced Message and Queuing Protocol)

Characteristics [28]:

- It is an *Application* layer protocol for message-oriented middleware, designed for *IoT*
- It supports *Point-to-Point* and *Publish-Subscribe* communication models
- It is designed to support a wide variety of messaging applications in an efficient way
- It provides message-delivery guarantees

- It is based on *TCP*
- The communications are handled by 2 components:
  - *Exchanges* (used to route the messages to the appropriate queues)
  - *Message queues* (messages are stored in queues and then sent to receivers)
- The routing between exchanges and message queues is based on pre-defined rules and conditions
- *AMQP* defines a layer of messaging on top of its own transport layer

### 5.10.5 DDS (Data Distribution Service)

Characteristics [28] [92]:

- It is an *Application* layer protocol, designed for real-time and scalable communication
- It is based on the *Publish-Subscribe* communication model
- Does not require a *broker* (like *MQTT* and *AMQP*)
- It defines 2 layers:
  - *Data-Centric Publish-Subscribe* (DCPS) (delivers the information to the subscribers)
  - *Data-Local Reconstruction Layer* (DLRL) (interfaces the *DCPS* layer, facilitates the sharing of distributed data among distributed objects, it is optional)
- Five entities are involved in the *DCPS* layer:
  - *Publisher* - generates data
  - *Data Writer* - used by the application to interact with the *Publisher* about the values and changes of the data related with a specific *Topic*
  - *Subscriber* - receives the published data and sends it to the application
  - *Data Reader* - called by the *Subscriber* to access the received data related with a specific *Topic*
  - *Topic*

## 5.11 Discussions

### 5.11.1 Comparison regarding the defined layers

A comparison between the previously presented communication protocols regarding the defined layers of the communication stack and the standards used it is shown in Fig. 5.6. The communication stack it is considered to have *4* layers ([2]):

- The *Physical* (*PHY*) Layer - is responsible for the frequency selection, the generation of the carrier frequency and the detection, the modulation and the encryption of the signal
- The *Medium Access Control* (*MAC*) Layer - is responsible for managing the transactions between neighbouring devices (Point-to-Point) (including framing/de-framing, flow control, transmission retries, acknowledgement management, collision resolution)
- The *Network* Layer - is responsible for the network structure, routing and security
- The *Application* Layer - includes the main application but in the case of the comparison made, it represents the framework necessary for developing the main application.

Due to the fact that the *CC1350* microcontroller will be used in this project, the communication protocols which can handle the *Sub-1 GHz* and / or the *2.4 GHz* frequency bands are of interest. These are *Zigbee*, *TI 15.4 Stack*, *Thread*, *WirelessHART*, *Bluetooth*, *BLE*, *LoRa WAN*, *SigFox*, *EnOcean*, *Wavenis* and *Z-Wave*. All of them except *Wavenis* and *EnOcean* cover all the layers. In addition, even if a protocol already covers the *Application Layer*, the specific *Application Layer protocols* (*CoAP*, *MQTT*, *XMPP*, *AMQP*, *DDS*) can still be used. They will bring some useful extra features at the expense of a higher complexity which will affect the processing time and the memory used.
Depending on the project, the *TCP/IP* protocol can also be used, by implementing it on top of standard *MAC* and *Physical Layers* (e.g IEEE 802.15.4) and adding an *Application Layer* framework. However, some changes have to made in this protocol for being applied to IoT (see Section 5.1.8). *6LoWPAN* can also be used for the cases when a connection to the Internet is desired for every node, like in the case of the *TCP/IP* protocol. However, the uncovered layers have to be filled. A solution for using it is the *Thread* protocol. For the considered project, the Sensor Nodes does not have to be capable of providing the acquired data directly to the Internet, so the main features of the *TCP/IP* and *6LoWPAN* protocols would not be used, even though other features would be useful (e.g. how the data should be split in packets, addressed, and routed)

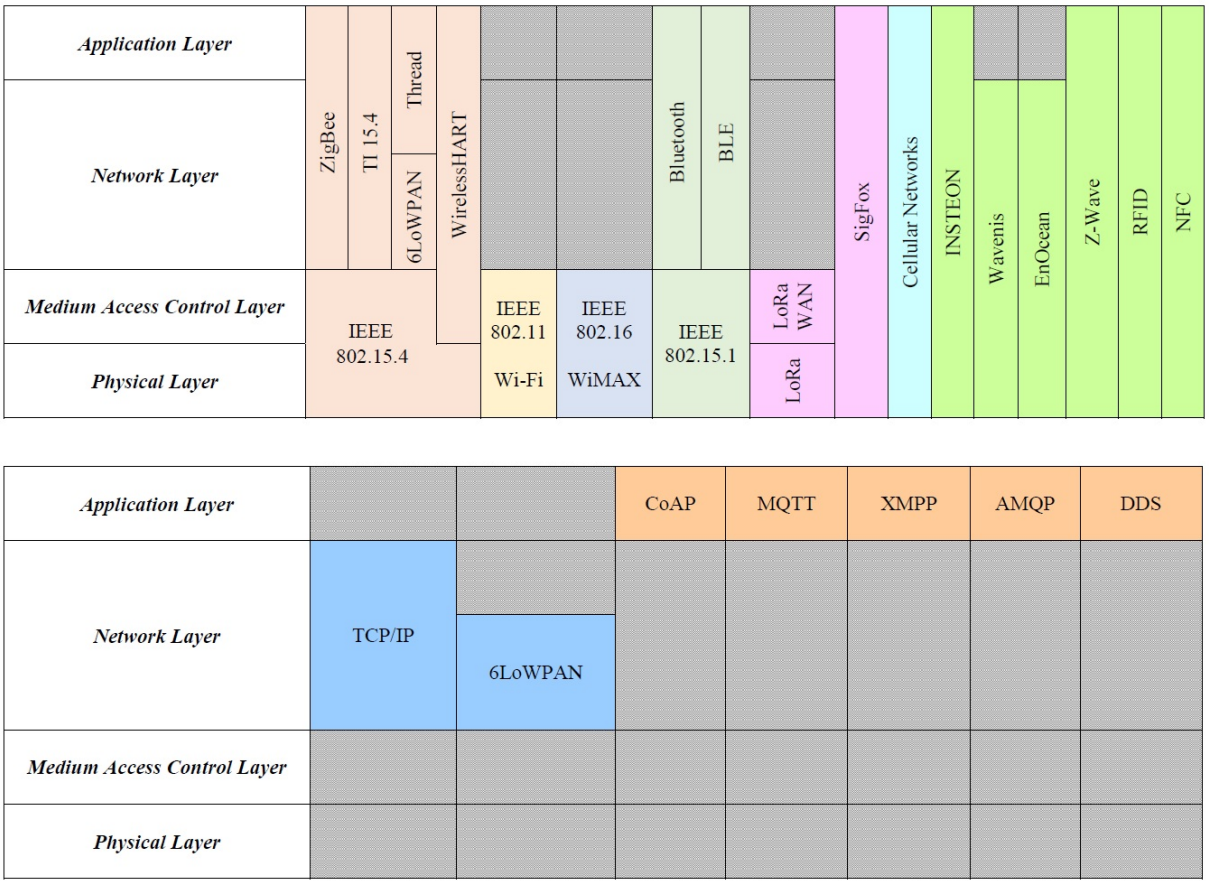


Fig. 5.6: Comparison between the previously presented communications protocols used in *IoT*

### 5.11.2 Comparison regarding the overhead of each message

Each packet sent does not contain only the desired payload. An overhead has to be added to each message which contains various information, according to each communication protocol (e.g. length of the payload, destination address, source address). Due to the regulations which are restricting the transmitted time (see Chapter 2), small overheads are desired in order to not waste the transmission time on information which does not have a direct influence in the fulfillment of the project's requirements. At the same time, the needed information have to be included in the overhead, such that its functionality and features to work. Moreover, the payload size is also playing an important role because in the case when it is big enough, it can compensate a big overhead (assuming that there is enough useful and available information which can be placed in the same packet).

Table 5.1 shows the overhead and the payload sizes for some of the communication protocols previously discussed based on the information from [49], [47], [87], [78] and [4]. For *MQTT-SN*, the overhead size is very small (only *2 Bytes*) compared to the maximum payload size (*65535 Bytes*), meaning that the overhead is insignificant. This representing another advantage

Table 5.1: The overhead size and the maximum payload size for each packet for some of the discussed communication protocols (including *EasyLink*, the abstraction layer on top of the *CC1350*'s *RF Driver* which is the starting point for implementing proprietary *Sub-1 GHz* communication protocols)

| **Communication Protocol** | **Overhead Size** | **Maximum Payload Size** |
|---|---|---|
| *EasyLink* | 2 - 9 Bytes | 128 Bytes |
| *TI 15.4 Stack* | 28, 32 or 40 Bytes, depending on the security scheme used: MIC-32, -64 or -128, respectively | 500 Bytes |
| *MQTT-SN* | 2 or 4 Bytes | 65535 Bytes |
| *Zigbee* | 39 Bytes if no security | 100 Bytes |
| | 57 Bytes if network security | 82 Bytes |
| | 66 Bytes if network and *APS* security are used | 73 Bytes |
| *LoRaWAN* | 13 - 28 Bytes only for the *MAC Layer* | 59 - 230 Bytes, depending on the used data rate (250 bps for 59 Bytes and 50 Kbps for 230 Bytes) |

of the usage of this *Application Layer* protocol on top of other communication protocols. If the *EasyLink* abstraction layer is used directly, a good ratio between the overhead and the payload sizes is obtained. For the *TI 15.4 Stack*, the overhead represents the 13th part from the total packet size in the worst case (when *MIC-128* security scheme is used) which is by far better than *Zigbee*. For this protocol, in the best case the overhead represents one third of the total packet size when no security is used. When both network and *APS (Application Support Sublayer)* security are used, the overhead size is almost equal with the payload size. This represents a major drawback of *Zigbee* when there are restrictions for the quantity of data which can be transmitted. Finally, *LoRaWAN* has a good balance at higher data rates but for the lower ones, the overhead reaches the same ratio of one third from the total packet size.

# Chapter 6

# Solution and Discussions

Based on the overview and comparisons of various existent communication protocols, *Application layer* protocols, spectrum access methods and security techniques, this chapter contains the discussions regarding the most suited choices for the considered use case of a *WSN* for the construction industry, taking into account the requirements and specifications initially defined (Appendix A). By putting all these components together, a final architecture of the *communication stack* is obtained and described. This *communication stack* represents the core contribution of this thesis. Alongside this, an architecture of the system based on a *WSN* which is intended to be used in the construction industry for tracking and monitoring tools, vehicles and workers is presented.

## 6.1 Choice of the Communication Protocol

As previously specified, the *CC1350* microcontroller developed by *Texas Instruments* is used in this project. This is a dual-band wireless microcontroller, which uses the *Sub-1 GHz* (*868 MHz* for Europe) and the *2.4 GHz* frequency bands. Further on, a discussion about the possible communication protocols which can be used for these frequencies, related to the requirements and the specifications of the project described in the previous chapter, will be done.

- **2.4 GHz**: The presented communication protocols which support this frequency band are: *Bluetooth*, *BLE*, *Zigbee*, *Thread*, *WirelessHART*, *Z-Wave*, *Wi-Fi* and *Wavenis*. This frequency band will be used for the connection between a handheld device (smartphone or tablet) and a Sensor Node / the Gateway in order to set it up or for visualizing the last acquired / received values. Thus, based on the specifications described in the previous chapter, the following requirements should be fulfilled by the chosen communication protocol:
  - *Point-to-Point* topology and a number of participants in the connection always equal

with *2* (the handheld device will communicate at one time either to a single Sensor Node or to the Gateway)

- Small range (the operator which has the handheld device should be able to set up or to visualize the data of a Sensor Node or of the Gateway from a maximum distance of *10 meters*)
- Small volume of data exchanged (sending commands for setting up or receiving the last acquired / received values or information about the current status, requires messages with the maximum payload on the order of tens of Bytes) which implies a small data rate (around *1 Kbps*)
- As low as possible power consumption (in order to keep the lifetime of the Sensor Nodes' batteries as long as possible)

Thus, between the previously mentioned *2.4 GHz* communication protocols, the simplest solution which covers all the communication layers (thus, there is no need to implement features lower than the *Application* layer) and fits the best to these requirements (without having too many features not used like support for Internet connection) is the *Bluetooth Low Energy (BLE)*. In addition, it has the advantage of a big range (up to *400 meters*) and the possibility to use *Beacons* (which can be used for indoor localization). The *CC1350* microcontroller has a *2.4 GHz* RF transceiver compatible with the *BLE 4.2 specifications*. Also, the *SDK* provided by *Texas Instruments* for this microcontroller includes a *BLE stack*, which makes the development of the 2.4 GHz applications easier, safer and faster.

- **Sub-1 GHz (*868 MHz* for Europe)**: The presented communication protocols which support this frequency band are: *Zigbee*, *TI 15.4 Stack*, *Z-Wave*, *SigFox*, *LoRa*, *Wavenis* and *EnOcean*. Due to the previously presented advantages of this frequency band, it will be used for transmitting the messages between the Sensor Nodes and the Gateway. Thus, based on the specifications described in the previous chapter, the following requirements should be fulfilled by the chosen communication protocol:
  - Ideally, *Mesh* topology should be used (for covering a larger area by using a single base station (gateway)), but a *Point-to-Point* topology is intended to be used in the beginning, for a gradual development of the *Wireless Sensor Network*. After tests will be performed with the *Point-to-Point* topology, it will be decided if the *Mesh* topology is necessary for the purpose of this project and if it should be used
  - Around *200* connected nodes should be handled by the network
  - As big as possible range (according to the specifications, the range of the CC1350 microcontroller is up to *4 kilometers* )

- Small data rate (around *1 Kbps*) (due to the fact that the sensor's reporting rate will be around *1 minute* and the messages' payloads will not be large (order of tens of Bytes); Also, a smaller data rate will lead to a higher range)
- As low as possible power consumption (in order to keep the lifetime of the Sensor Nodes' batteries as long as possible)

*SigFox* has a very restrictive limitation for the number of sent and received messages, *LoRa* is very sensitive to interferences and due to its closed *PHY* layer, specific hardware has to be used for both Sensor Nodes and Gateway which will loose the advantage of CC1350's dual-band, *Wavenis* and *EnOcean* have low ranges because they are intended to be used mainly for applications related with home automations or which does not require a large spatial spread of the Sensor Nodes (though, *EnOcean* has the advantage of a very low power consumption). Thus, between the previously mentioned *Sub-1 GHz* communication protocols, the solutions which covers all the communication layers (thus, there is no need to implement features lower than the *Application* layer) and fits the best to these requirements are the *Zigbee* and the *TI 15.4 Stack*. For *Zigbee*, the *PRO 2015* version should be used, which is not at this time supported in the *CC1350's SDK*. For using it, it should be ported from scratch, due to the fact that not even the standard *Zigbee* is included in the *CC1350's SDK*. Moreover, the *Duty Cycle* restriction has to be considered for *Zigbee* because it does not use the *Frequency Hopping* technique due to the fact that it uses *DSSS* as spectrum access method. Thus, the large overhead size compared to the low payload size for each of the message, makes *Zigbee* not a good option for being used according to the Sub-1 GHz regulations for *Non-specific SRDs*. Although, the Star topology and the inclusion in the *CC1350's SDK*, makes the *TI 15.4 Stack* the best choice for the first development step of the the *WSN* of the considered project. Further on, if a *Mesh* topology will be needed, either implementing this feature on top of the *TI 15.4 Stack* or using *Zigbee PRO 2015* or *Zigbee PRO 2017* (by porting it or using it from the *SDK* directly if they will be added) together with a very strict approach of sending messages will be considered.

## 6.2 Choice of the Application Layer Protocol

*CoAP*, *MQTT*, *XMPP*, *AMQP* and *DDS* are based on the *TCP* or *UDP* transport protocols which are not yet available for the *CC1350* microcontroller, so they would need to be implemented or ported. *XMPP* is based on *XML Stanzas* which implies the usage of *XML tags* which are bringing a very large overhead to each message. *MQTT*, *MQTT-SN*,

*AMQP* and *DDS* are based on the *Publish-Subscribe* communication model which would bring extra complexity without any benefit for the case when multiple Sensor Nodes just have to send all the acquired data to a single Gateway. *CoAP*'s similitude to *HTTP* is not useful because the Sensor Nodes does not have to provide the acquired data directly to the Internet, only the Gateway can do this in order to keep the simplicity and the extended lifetime on the Sensor Nodes. So, *CoAP*, *MQTT*, *XMPP*, *AMQP* and *DDS* would bring more value to be used for the connection between the Gateway and the Cloud, not between the Sensor Nodes and the Gateway. However, between the considered *Application Layer* protocols, *MQTT-SN* is the most suited to be used for this project due to the fact that:

- it is not built on top of the *TCP* or *UDP* transport protocols; Any wireless protocol can be used (e.g. *Zigbee*, *BLE*, *Z-Wave*, proprietary *Sub-1 GHz* protocol) for transport
- it is asynchronous (no synchronization between the clients and the broker has to be done)
- the overhead for each message is very small (*2* or *4 Bytes*, depending on how big the payload is desired to be (up to *256 Bytes* for a *2 Bytes* overhead, or up to *65535 Bytes* for a *4 Bytes* overhead)
- it supports different levels of *QoS* which can be interchanged according to how important is the sent packet, guaranteeing when is the case that the packet arrived successfully at the receiver
- the usage of a standard protocol would allow third-party *Sensor Nodes* to join the network without adapting their message format. However, the *topic ID* used by the new nodes would still have to be adapted in order to not have any collisions
- the protocols for creating and maintaining the network (advertisements sent by the gateway, searching-messages sent by the clients, connection messages, acknowledgement messages, *Ping* messages (for signalizing that a device is still alive), will messages) and sending/receiving messages (how the packets are composed and decomposed) are defined
- the *broker* can filter the messages with respect to the subject, the content or the type and it performs authentication and authorization
- a *client* can have a mixed role by being both *subscriber* and *publisher* at the same time

On the other hand, using *MQTT-SN* for this project implies some disadvantages:

- for the situation when *Sensor Nodes* just have to send the acquired values from the sensors to a *Gateway*, using the *Publish-Subscribe* mechanism would not bring any

advantage (there will be only *publishers*). However, if *Monitoring* devices (hand-held devices) are used, then these can be represented by the *subscribers*

- no caching of the messages is provided on the *Publisher* side, only on the *Broker* side. So, the caching problem still has to be solved separately for the case when the Sensor Node is out of the range of the Gateway
- some implementations for the *C programming language* are open-source available online (e.g. the Eclipse Paho project **??**), but they have to be checked and adapted for fitting in the project

Another problem which arises when using *MQTT-SN* is the situation when the *Frequency Hopping* technique is used. This feature involves the usage of four message types (*PAS*, *PA*, *PCS* and *PC* which are described in Chapter 3) which are not covered by the message formats described in the *MQTT-SN Specifications* [110]. Thus, these messages will have to be added, so the standard version of *MQTT-SN* could not be used, a customized one being needed. This counters one of the biggest advantages of *MQTT-SN* - the generality of adding to the network third-party Sensor Nodes which communicate over the *MQTT-SN* protocol. Moreover, if a Sensor Node which uses the standard *MQTT-SN* protocol is placed in the coverage area of to the network which uses the customized version of *MQTT-SN*, it will be allowed to join the network, because the searching and the connection messages will be the same but it will not be able to exchange the *Frequency Hopping* related messages and it will work on a single channel, perturbing it. So, in the case when *Frequency Hopping* is used, changes in the connection messages should also be done in order to not allow devices using the standard *MQTT-SN* to communicate.
So, even though some modifications should be done to the standard version in order to fulfill the requirements of the considered project, the features already available within *MQTT-SN* are valuable enough in order to worth the effort involved by these changes and to overcome its disadvantages. Thus, *MQTT-SN* represents an useful element for this project's communication protocol.

## 6.3 Choice of the Spectrum Access Method

As stated in Section 2.4.1 from Chapter 2, the *SRDs* have to respect some regulations imposed by *CEPT* and *ETSI* regarding the *Spectrum Access Methods*. For the frequency bands which are of interest for this project (*863 - 870 MHz* and *868 - 868.6 MHz*) two choices are given: *Duty Cycle* or *LBT* combined with *AFA*. The advantages and disadvantages related to the considered project of these two *Spectrum Access Methods* are described below:

- **Duty Cycle**
  - The *868 - 868.6 MHz* frequency band will be used because it offers a higher *duty cycle* (*1%* instead of *0.1%* for the *863 - 870 MHz* frequency band) and multiple channels are not needed (because only one channel will be used constantly)
  - Due to the fact that only one channel will be used by all the nodes, the probability of having a collision is higher. *LBT* can be used also in this case, which will decrease the number of collisions but it will increase the delays of the sent messages
  - The higher duty cycle permits a higher amount of data to be sent every hour, though this is still a tight constraint. A comparison of the maximum data which can be transmitted using different data rates for both *0.1%* and *1% duty cycles* is shown in Table 6.1. Using the values from this table, a comparison of the maximum number of different types of messages (containing either *GPS*, *IMU* or *ACK* data) which can be sent using different data rates and different communication protocols for both *0.1%* and *1% duty cycles* is shown in Table 6.2. For the lowest data rate, *625 bps*, the one which ensures the longest range, sending a message containing the *GPS* information every minute would be impossible with *Zigbee*. With *TI 15.4 Stack* it is possible with the *1% duty cycle* but no other messages could be sent by the same node. For *50 Kbps*, the standard data rate for *IEEE 802.15.4g*, all the protocols are capable of sending a message containing the *GPS* information with a period smaller than *one minute* (i.e. *Zigbee*, the protocol which has the highest overhead, can send almost *5 GPS* messages every minute) and also sending other messages without crossing the limitations. However, from the Gateway point of view, which has to send most of the time only *ACK* messages, none of the data rates can ensure a sufficient number of messages to send in response to multiple nodes sending messages at the same time (i.e assuming that only *GPS* messages are sent by the Sensor Nodes and that they are sending the maximum possible number of messages in one hour, the Gateway can handle maximum *5 nodes* for *EasyLink*, and only *one node* completely for *TI 15.4 Stack* and *Zigbee*). This represents a severe limitation which can be overpassed simply by not using the *ACK* messages
  - When a Sensor Node goes out of the Gateway's range, it should store the acquired sensor values and then send all of them once it is again in the Gatway's range. However, being out of the range for a long time might lead to a large block of data which has to be sent. Transmitting all the values from this block which might consume the whole quantity of data which can be send in one hour, not giving any chance of the following acquired values within one hour to be transmitted. A solution of this problem is to send only some of the previous unsent values uniformly over time

Table 6.1: The maximum data which can be transmitted with the *0.1%* and *1% duty cycles* and different *data rates*

| | **Maximum data which can be transmitted** | | | |
|---|---|---|---|---|
| | **0.1% Duty Cycle** | | **1% Duty Cycle** | |
| | **per hour** | **per minute** | **per hour** | **per minute** |
| **625 bps** | 281.2 Bytes | 4.68 Bytes | 2812 Bytes | 46.8 Bytes |
| **5 Kbps** | 2304 Bytes | 38.4 Bytes | 23040 Bytes | 384 Bytes |
| **50 Kbps** | 23040 Bytes | 384 Bytes | 230400 Bytes | 3840 Bytes |
| **500 Kbps** | 230400 Bytes | 3840 Bytes | 2304000 Bytes | 38400 Bytes |

when the node is back in the Gateway's range (the exact number of messages should be calculated according to the periods of the new messages might be sent), and the rest of them during the night, when most likely there will be no new values to be sent

- This *Spectrum Access Method* is easy to implement:
  - a list of all the messages' sizes from the last hour has to be created and continuously updated
  - a timer for counting the off time required between *2* consecutive transmissions has to be used
  - a maximum message size has to be considered according to the data rate used and the value of the maximum continuous transmission time
- Another advantage is represented by the messages used which are all respecting the standard *MQTT-SN* format. Thus, third-party Sensor Nodes can be added to the Network without any firmware changes

- **LBT + AFA**
  - The *863 - 870 MHz* frequency band will be used because it offers *34 channels* for the *Frequency Hopping* technique. Thus, a lower probability of collisions is obtained which implies a higher robustness and lower delays due to retransmissions or waiting times
  - The main problem of the Duty Cycle Spectrum Access Method - the limitation of the transmission time - is not present anymore, which implies that the number of the messages sent by a single node is unbounded. Even that a node will send a very high number of messages, due to the frequency hopping technique, the other channels will be free and can be used without any problems by the other nodes
  - The main disadvantage is that this solution is more difficult to implement. However, starting from the *version 2.0.1* of the *TI 15.4 Stack*, *Frequency Hopping* is included.

Table 6.2: The maximum number of different types of messages which can be transmitted *per hour* using different *data rates* and different *communication protocols* with the *0.1%* and *1% duty cycles*. Three types of messages are considered: *GPS* (uses *20 Bytes* and contains: Latitude and Longitude (Degree (*1 Byte*), Minute (*2 Bytes*) and Azimuth (*1 Byte*) for each), Altitude (*4 Bytes*), Number of available satellites (*1 Byte*), UTC time (hour, minutes, seconds - *1 Byte each*), Timestamp (*4 Bytes*)), *IMU* (uses *16 Bytes* and contains: values from *Accelerometer* (one value (*2 Bytes*) for each of the *3 Axes*), values from *Gyroscope* (one value (*2 Bytes*) for each of the *3 Axes*), Timestamp (*4 Bytes*)) and *ACK* (uses *2 Bytes*). The overheads considered are *2 Bytes*, *28 Bytes* and *57 Bytes* for *EasyLink*, *TI 15.4 Stack* and *Zigbee*, respectively (see Chapter 5 for more information)

| | | **Raw (No protocol)** | | **EasyLink** | | **TI 15.4 Stack** | | **Zigbee** | |
|---|---|---|---|---|---|---|---|---|---|
| | | **0.1%** | **1%** | **0.1%** | **1%** | **0.1%** | **1%** | **0.1%** | **1%** |
| **625 bps** | **GPS** | 14 | 140 | 12 | 127 | 5 | 58 | 3 | 35 |
| | **IMU** | 17 | 175 | 15 | 156 | 6 | 63 | 3 | 38 |
| | **ACK** | 140 | 1406 | 70 | 703 | 9 | 93 | 4 | 47 |
| **5 Kbps** | **GPS** | 115 | 1152 | 104 | 1047 | 48 | 480 | 29 | 299 |
| | **IMU** | 144 | 1440 | 128 | 1280 | 52 | 523 | 31 | 315 |
| | **ACK** | 1152 | 11520 | 576 | 5760 | 76 | 768 | 39 | 390 |
| **50 Kbps** | **GPS** | 1152 | 11520 | 1047 | 10470 | 480 | 4800 | 299 | 2992 |
| | **IMU** | 1440 | 14400 | 1280 | 12800 | 523 | 5236 | 315 | 3156 |
| | **ACK** | 11520 | 115200 | 5760 | 57600 | 768 | 7680 | 390 | 3905 |
| **500 Kbps** | **GPS** | 11520 | 115200 | 10470 | 104700 | 4800 | 48000 | 2992 | 29922 |
| | **IMU** | 14400 | 14400 | 12800 | 128000 | 5236 | 52363 | 3156 | 31561 |
| | **ACK** | 115200 | 1152000 | 57600 | 576000 | 7680 | 76800 | 3905 | 39050 |

The *LBT* implementation is already included in the *EasyLink API*. Only the *adaptive* feature of *Frequency Hopping* (i.e. the identification of the fixed sources of interference, their exclusion from the list of available channels in order to never ”hop” on them [39]) has to be implemented. Afterward, these three blocks have to be combined in order to work together. An example of combining these blocks is given in the *Pseudocode* algorithm 1

– Due to the additional messages which have to be exchanged for the *Frequency Hopping* features, the *MQTT-SN* application layer has to be adapted because its available messages types do not include these ones. Thus, third-party Sensor Nodes which also use *MQTT-SN* cannot be used directly in the network but their firmware has to be updated first. So, the generality feature brought by *MQTT-SN* is lost

**Algorithm 1** LBT + AFA core - called every *Dwell period*

```
repeat
    new_channel ← next value generated by DH1CF
until new_channel NOT IN list_excluded_channels

loop
    LBT_result ← LBT(new_channel)

    if LBT_result == SUCCESS then
        Reset(Count_Consec_Fail_Attempts[new_channel])
        return new_channel
    else
        Count_Consec_Fail_Attempts[new_channel]++
        if Count_Consec_Fail_Attempt[new_channel] == MAX_FAIL_ATTEMPT then
            Add new_channel to list_excluded_channels
            return −1
        end if
        Backoff_time ← Random value
        Wait(Backoff_time)
    end if
end loop
```

So, due to the very tight limitations of the *Duty Cycle* method, especially for the Gateway, no matter what data rate or communication protocol is used, the *LBT* combined with *AFA* method is a better choice. Even though it is more difficult to implement it and to keep it working continuously (the synchronization is a key aspect), it does not impose any restrictions on the number or size of transmitted messages which improves the robustness of the whole communication system.

## 6.4 Choice of the Security Methods

As explained in Chapter 4 , due to the availability of the dedicated hardware accelerator and to the acceptance of different standard communication protocols, the *AES CCM* technique based on a *shared key* of *128 bits* represents the best encryption/decryption and authentication solution to be used for any communication protocol choice. However, the chosen protocol's implementation of this technique should be adapted in order to use the available accelerator.
Based on the advantages of having the *ECC* core functions in the *ROM* memory and the implementations of different schemes in the *CC1350*'s *SDK* (see Chapter 4), the key agreement can be done using the *ECDH* technique and the authentication can be performed based on the *ECDSA* scheme. This would ensure a higher level of security in comparison with the *AES CCM* based solution.
So, a deeper analysis focused on the *CC1350*'s specifications has to be done, in which the performances obtained, the computational effort needed and the utilized resources have to be compared for each of the considered communication protocols (e.g. *TI 15.4 Stack*, *Zigbee*). Thus, it can be found out for each communication protocol if it is possible and if it is better to add extra features, like *ECHD* and *ECDSA*, or to use their offered standard security solutions.

## 6.5 The Architecture of the Communication Stack

In Fig. 6.1 it is shown the complete *Communication Protocol Stack* for the considered project.
The *Sub-1 GHz* communication is based on the *TI 15.4 Stack* due to its advantages explained previously in this chapter (i.e. star topology, frequency hopping, unlimited messages sent if *LBT* and *AFA* restrictions are respected) and its implementation in the *SDK* of the *CC1350* microcontroller. This stack covers all the four layers: *Physical*, *MAC*, *Network* and *Application*. The *Physical* and *MAC* layers are based on the *IEEE 802.15.4 e/g* standards. The only difference to these standards is in *MAC* layer, where the *Adaptability* feature of the *Frequency Hopping* technique has to be added in order to respect the *Sub-1 GHz Spectrum Access* regulations. The security (encryption and authentication) is based only on the *AEM 128 bit CCM* method, as specified in the *IEEE 802.15.4 e/g* standards. The *Network* layer, based on a star topology, is specific for the *TI 15.4 Stack*. The *Framework for the Application Layer* provided by the same stack is combined with the *MQTT-SN* framework. Thus, the features of these two frameworks are used in the main *User Application* together with the drivers of the *CC1350* microcontroller and of the used peripheral devices, for fulfilling the requirements of the considered project.
The *2.4 GHz* communication is based on the *BLE core Stack 4.2 version 2.3.2* which is available within the same *SDK* of the *CC1350* microcontroller. This stack, like the *TI 15.4 Stack*, covers all the layers between the *Physical* and the *Framework for the Application Layer*.

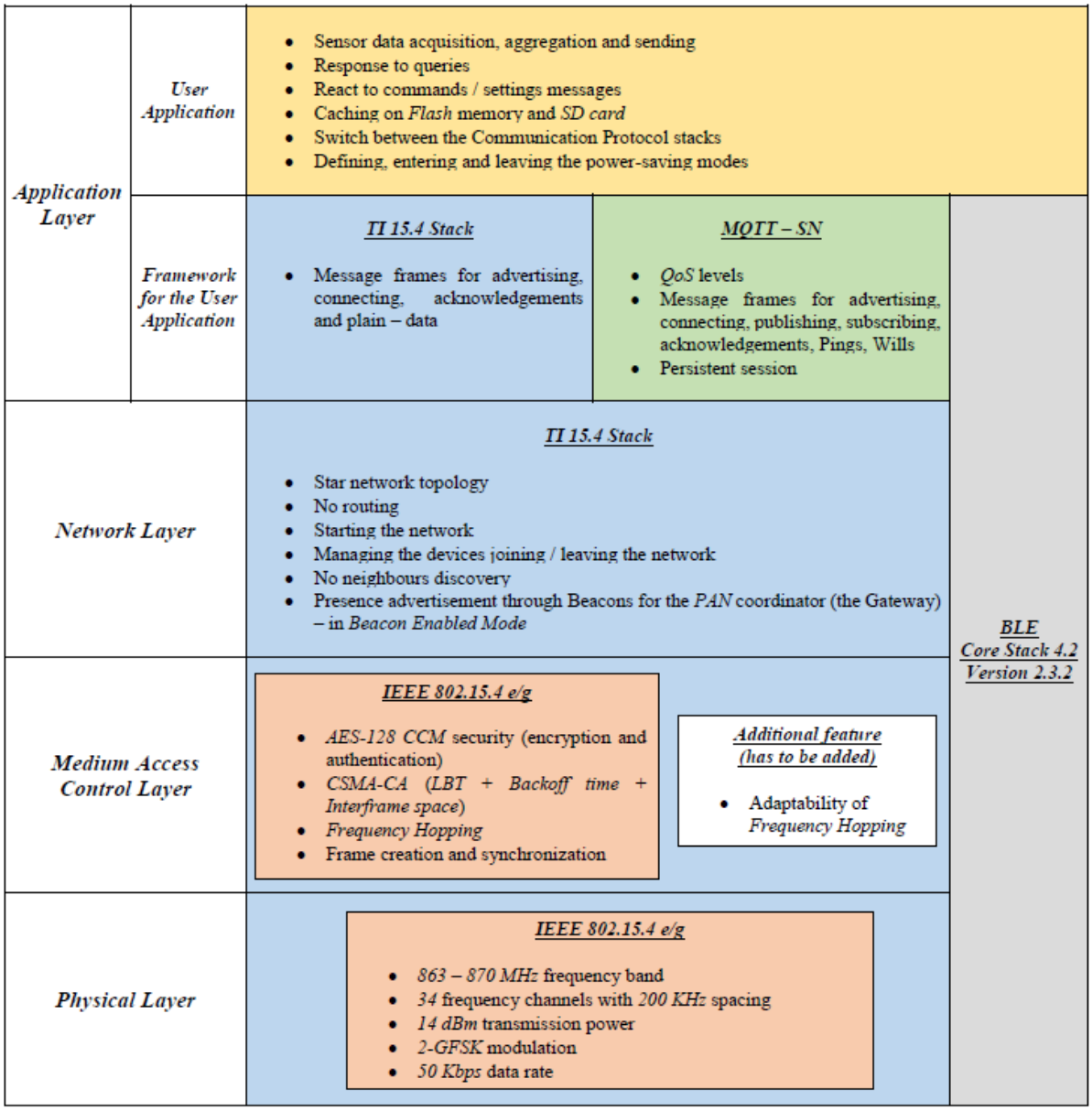


Fig. 6.1: The complete Communication Stack used for the considered project

However, due to the limited size of the *Flash* memory, not both *TI 15.4* and *BLE* stacks can be loaded, so they cannot run concurrently (see Chapter 2). When a change from *Sub-1 GHz* to *BLE* or vice-versa is needed, first the required stack has to be loaded in the *Flash* memory.

## 6.6 The Architecture of the System

The architecture of the whole system is shown in Fig. 6.2.

- The *Sensor Nodes* are communicating over the *Sub-1 GHz* frequency band with the *Gateway*, using the previously described protocol stack, and over the *2.4 GHz* frequency band with the *Basic Monitoring BLE Nodes*, using the *BLE* Stack. They can be either MQTT-SN Publishers or both MQTT-SN *Publishers* and *Subscribers* at the same time, whether they need the values from other nodes in order to fulfill their acquisition or not

- There are two types of *Basic Monitoring Nodes*: the ones which are connected directly to the *Sensor Nodes* over the *2.4 GHz* frequency band and the ones which are connected directly to the *Gateway* over the *Sub-1 GHz* frequency band, which can get more information (e.g. more values from the past, processed values, decisions) due to the connection with the *Broker*

- The *Sub-1 GHz Sink* combined with the *Raspberry Pi* board for having *Internet* connection over an *LTE* module represents the *Gateway*. It acts as a *MQTT-SN Aggregating Gateway*. It has on one side only *MQTT-SN* connections (from the Sensor, Monitoring and Display Nodes) and on the other side has a single *MQTT* connection with a *Broker* placed in the *Cloud*.

- The *Broker* is connected to remote *Advanced Monitoring Devices* (e.g. handheld devices, PCs) acting as *Subscribers*

- At the same time, the *Broker* is connected to a Data Processing Module, also placed in the *Cloud*, which is both a *Subscriber* and a *Publisher*. Its role is to process the acquired sensor values and to take decisions which are then sent to the Display Nodes

- The *Display Nodes* are devices used for showing the most recent commands and the current status to the construction site workers

- Optionally, the *Basic* and the *Advanced Monitoring Nodes and Devices* can be used for setting up various parameters of the networks (e.g. sampling period, threshold values). In this case, the ones which are using the *MQTT* protocol will be both *Subscribers* and *Publishers* and the ones which are using *BLE* will also send *Data Unit* frames, not only receiving

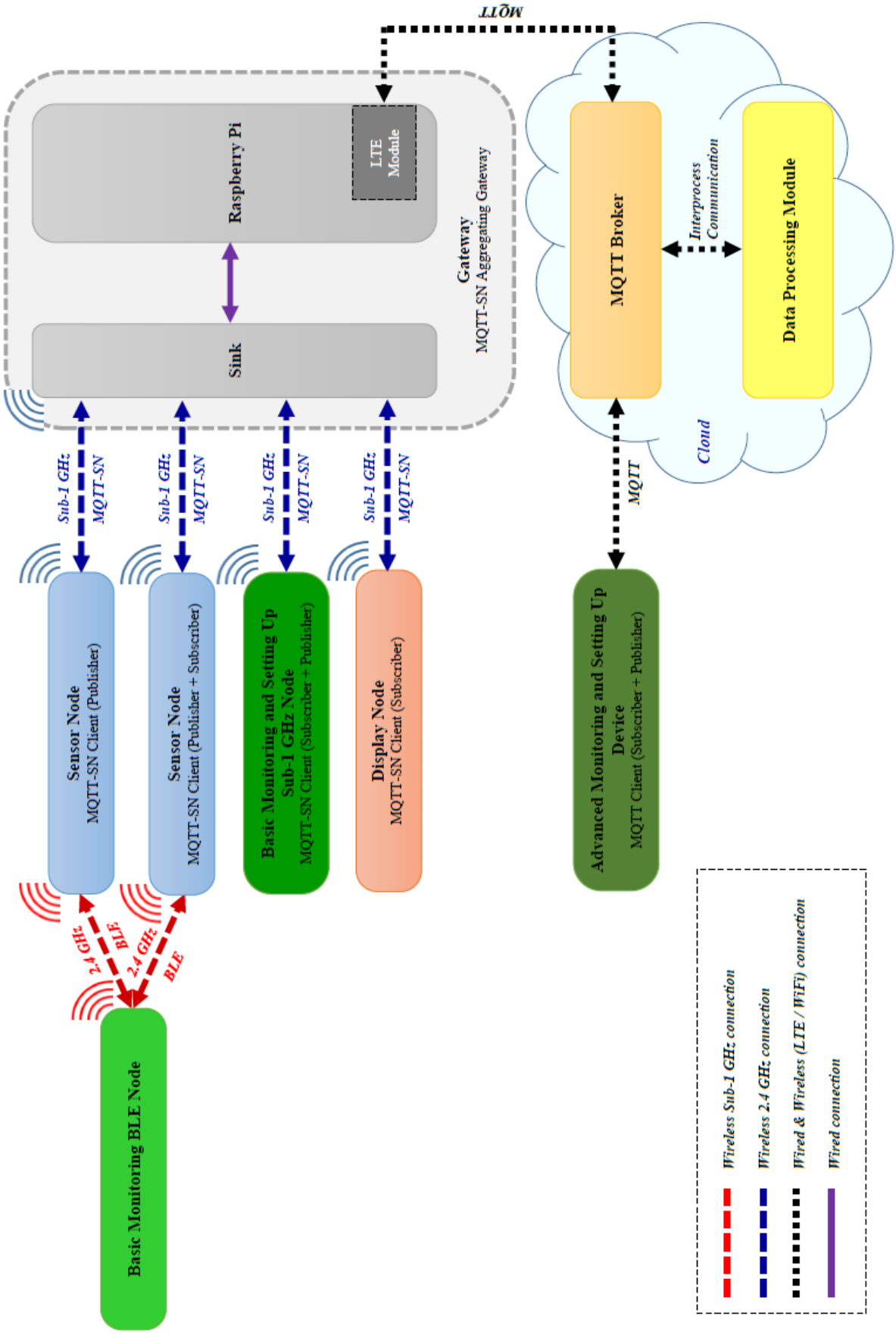


Fig. 6.2: The architecture of the whole system used in the considered project

# Chapter 7

# Implementation

## 7.1 Use Case

The scenario of a highway construction site is considered (Fig. 7.1). An excavator is situated at a small distance (i.e. several kilomters) from the area where the current section of the highway is constructed. It extracts soil from a specific area and then loads it into trucks. The excavator has to extract soil several times in order to fill a truck. The trucks are then carrying the soil to the current section under construction of the highway and unloading it. This soil is used to bring the base of the highway at the same level. A compactor is then used to eliminate the air between the soil grains, thus, resulting a solid base layer.
*Sensor Nodes* are desired to be placed on trucks in order to acquire information which will be used to improve the entire process by determining the optimal number of trucks, their routes, the number of loading and transporting cycles done per day, the number and the length of the breaks. For this, information about the location, the waiting time for being loaded by the excavator and how many buckets of soil are needed in order to be filled have to be acquired.
Due to the incremental construction of the highway, the site moves along it. So, there is not any fixed point where some modular offices could be placed (like in the case of buildings' construction). The ideal position of the *Gateway* would have been on top of one of these modular office because they are usually placed in the center of the construction site and they have a continuous power supply. However, in this case, it has to be placed on the excavator because this is now the central point. The *Gateway* can be powered up in this case from the excavator's battery or it can be also designed to use a battery and to have low power consumption.

## 7.2 The proprietary CC1350-based Sensor Node

For this Use Case, a *Sensor Node* based on the *CC1350* microcontroller was designed within *DFKI (Deutsche Forschungszentrum für Künstliche Intelligenz GmbH)* (Fig. 7.2). The main


Fig. 7.1: An excavator loading a Truck on a highway construction site from Albacete, Spain, operated by the *Ferrovial* company

included components are:

- the *CC1350* microcontroller developed by *Texas Instruments*
- the temperature and the power supply voltage sensors withing the microcontroller
- an external *8 MBytes Flash* memory
- a pressure and temperature sensor (*BMP280*)
- a humidity and temperature sensor (*HDC1010YPAR*)
- a *9-axis IMU (Inertial Measurement Unit)* (*3-axis* gyroscope, *3-axis* accelerometer, *3-axis* magnetometer) (*MPU9250*)
- a *mikroBUS* connector for the *Click* boards developed by *mikroElektronika*. The *GNSS 4 Click* module was used for obtaining localization information
- a voltage divider connected to one of the analog input pins of the microcontroller for measuring the battery voltage (not the voltage measured by the voltage sensor within the microcontroller, which is the output of the voltage regulator)
- a combined *PCB* antenna for both *Sub-1 GHz* (*868 MHz*) and *2.4 GHz* frequency bands
- a rechargeable battery

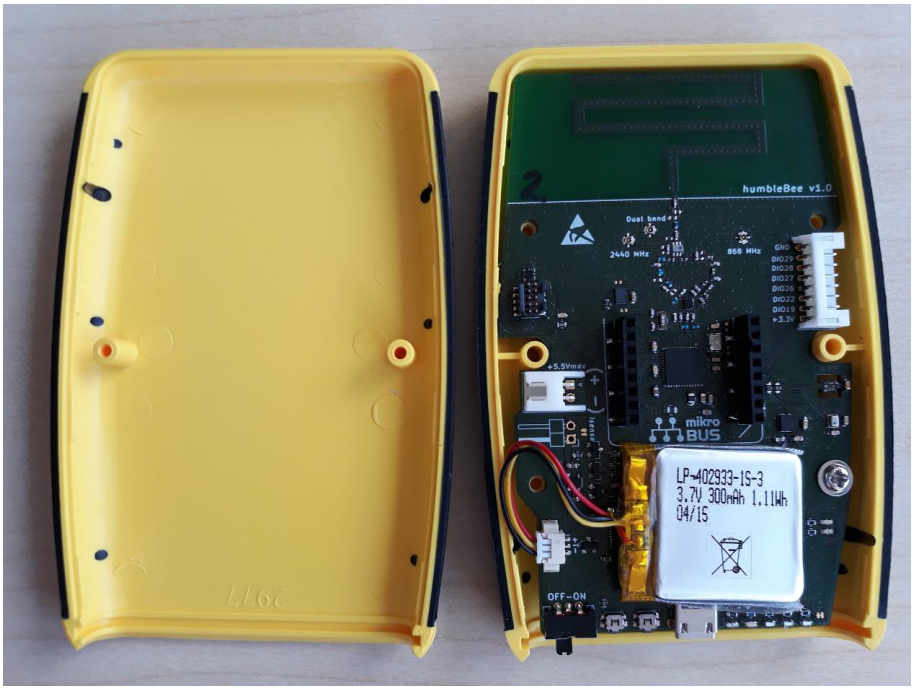


Fig. 7.2: The Sensor Node designed within *DFKI (Deutsche Forschungszentrum für Künstliche Intelligenz GmbH)* for the considered Use Case, based on the *CC1350* microcontroller

### 7.2.1 Drivers

For the development of the microcontroller's firmware for the *Sensor Nodes*, the following main open-source drivers for the *CC1350* microcontroller were used:

- *GPIO*, *I2C*, *ADC*, *NVS*, *UART*, *Watchdog* and *RF* drivers from *CC1350 SDK* [55]
- *TI-RTOS* related drivers from *CC1350 SDK* [62]
- *BMP280*, *HDC1000*, *MPU9250* sensors' drivers from *CC1350 SDK* [67]
- *GNSS 4 Click* board driver provided by *mikroElektronika* [85]

The following changes and additions had to be done:

- The board's configuration file had to be updated because some pins of the microcontroller have different connections than the ones of the *LaunchPad* (e.g. the *LEDs* and the *Push Button* are connected to different pins; on this board there is only one *Push Button*, not two like on the *LauchPad*)
- A driver for reading the battery voltage based on the available voltage divider had to be developed. It uses the *ADC* and *GPIO* drivers
- A function for the magnetometer's driver of the *MPU9250 IMU* sensor had to be added in order to convert the raw read values into *micro Tesla* units

- After the microcontroller is powered up or reset, a sequence for resetting the *GNSS* module (by keeping the *Reset* pin of the module on the *Low* level for *300 ms*) was added

## 7.3 The Gateway

For the *Gateway's Sink*, a *CC1350 LaunchPad* was used with an external antenna attached in order to improve the reception. A *Raspberry Pi* board is connected to the sink, having an *LTE* module connected in order to be able to upload the received sensors' values to the *Cloud*.

### 7.3.1 Drivers

For the development of the microcontroller's firmware for the *Gateway's Sink*, the following main open-source drivers for the *CC1350* microcontroller were used:

- *GPIO*, *NVS*, *UART*, *Watchdog* and *RF* drivers from *CC1350 SDK* [55]
- *TI-RTOS* related drivers from *CC1350 SDK SDK* [62]

## 7.4 Developed Firmware

Each *Sensor Node* reads periodically the connected sensors and sends them to the *Sink* over the *868 MHz* frequency band. For each successfully received packet, the *Sink* sends an *Acknowledgment* message to the sender. If the sender does not receive the *Acknowledgment* message in a predefined time, it repeats the sending process after a random *backoff* time for a predefined maximum number of times. Also, each *Sensor Node* sends periodically *"I Am Alive"* messages to the *Sink*, to let it know that it is still functional, even though maybe it did not send yet any new packets containing sensors' values due to various reasons. For each successfully received packet, the *Sink* sends the contained values over *UART* using the *JSON* format to the connected *Raspberry Pi* board. Also, the *Sink* sends in addition, periodic *"I Am Alive"* messages in the *JSON* format to the *Raspberry Pi* board. This board stores in a local database the received messages and sends them using an *LTE* module to a *Cloud* database [1]. No caching of the sensors' values was done by the *Sensor Node* nor the *Sink* due to the small *Flash* memory available related with the sampling frequency used. Due to the development only for testing purpose, the regulations regarding the *Spectrum Access Method* were not considered.

More implementation details of the firmware for the **Sensor Nodes**:

[1] The functionalities of the *Raspberry Pi* board were developed by other members of this project's team

- The sensors' data acquisition and the sending / receiving functionalities are separated in different tasks and functions. The synchronization between them is done using *events*
- A task is created for each sensor (*GNSS* module, *MPU9250* (*IMU*), *BMP280* (temperature and pressure), battery voltage). Each task is executed periodically. Each execution implies the reading of the sensor, the writing of the acquired values in global variables in a safe way (using *semaphores*) and the waiting in the sleep (blocked) state until the next period
- Another periodic task is responsible for reading the values placed in the global variables in a safe way (using *semaphores*), reading the current time, increasing the packet counter and calling the function for placing all these values in a packet and sending them over the *868 MHz* frequency band. Two versions of this function were developed and tested:
  - The first one places all the values in the same packet
  - The second one creates *4* packets, each with a different ID number:
    - *Packet 1*: timestamp, packet number, battery voltage, temperature, pressure, UTC time, the number of available satellites
    - *Packet 2*: timestamp, packet number, acceleration (3 values), angular velocity (3 values)
    - *Packet 3*: timestamp, packet number, amplitude of the magnetic field (3 values)
    - *Packet 4*: timestamp, packet number, latitude (degrees, minutes, azimuth), longitude (degrees, minutes, azimuth), altitude
- The values are placed in the packets using a specific data frame, bit by bit, using bitwise operations. The frame has a *header* and a *payload*. The *header* contains the destination address, the sender address, the packet type (in order to distinguish between data, *Acknowledgement* and *"I Am Alive"* packets), the packet subtype (in order to distinguish between the four possible data packets previously described) and the packet's length. The *payload* is empty for the *Acknowledgement* and *"I Am Alive"* packets and contains the sensors' values for the data packets
- Each Sensor Nodes filters the received messages and ignores the ones which are not for them (e.g. the *Acknowledgement* packets for the other nodes)

More implementation details of the firmware for the **Sink**:

- The data processing and the sending / receiving functionalities are separated in different tasks and functions. The synchronization between them is done using *events*

- According to the type and subtype of each successfully received packet, the payload's bits are concatenated according to the splitting procedure done by the Sensor Nodes. Then, the content together with the *Sink*'s receiving timestamp are sent over *UART* using the *JSON* format

- An *Acknowledgement* message is sent back to the *Sensor Node* if the packet was received successfully (i.e. a packet is not received successfully when for example there is a configuration error or when the *RX* buffer is full)

### 7.4.1 The Communication Protocol Stack

Regarding the *communication protocol stack*, a proprietary light one was developed, as an initial step in the incremental development, based on the *EasyLink* abstraction layer of the *RF Driver*. The stack is based on the one used in the *rfWsnConcentrator* and *rfWsnNodeBleAdv* examples provided in [64]. Compared to the stack designed and presented in Chapter 6, this solution is characterized by:

- The same *Physical layer*: *868 MHz* frequency band only with one channel, *14 dBm* transmission power, *2-GFSK* modulation and *50 Kbps* data rate

- The *MAC Layer* is reduced only to the usage of standard *EasyLink* frames and the usage of the *Backoff* time (when the transmission is not successful). No *Spectrum Access Method* is used. The channel is not verified if it is busy before sending. No security is used.

- Regarding the *Network layer*, Star topology is used without any management of the devices in the network

- The *Framework for the User Application* was created, by setting the frames of the different packet types and by filling these packets bit by bit with the values provided

## 7.5 Performed Tests

A field test was performed on a highway construction site from Albacete, Spain, operated by the *Ferrovial* company. The purpose was to test the functioning of the hardware of the *Sensor Nodes* and *Gateway* in a real environment and to acquire the sensors' data for further analysis and developments.
Only one *Sensor Node* was used and it was placed on a truck. The *Gateway* was placed *20 meters* away from the location of the excavator. A monitoring *Sink* (i.e. a *Sink* with the same address as the one used in the *Gateway* but without the *Acknowledgment* packet sending mechanism) was connected to a Laptop and placed in the same position as the *Gateway*. The *Sensor*

*Node* was acquiring and sending continuously the *IMU* data with a frequency of *100 Hz* and the other sensors' values with a frequency of *1 Hz*. The precision of the IMU data was desired to be higher, so each acceleration and angular velocity value sent had *64 Bytes* (instead of *16 Bytes* in the regular case). Due to the higher sending frequency and the large packet size, no retransmission were done if a packet was not successfully received. While the truck was in the coverage area of the *Gateway* and of the monitoring *Sink*, the sent packets were received and printed on the *UART* terminal from the connected Laptop by the monitoring *Sink*. Due to the choice of the *Physical layer* characteristics and the design of antennae used for the *Sensor Node* and for the *Sinks*, the maximum range achieved was *2 kilometers*.

Tests were performed also in the laboratory. Two *Sensor Nodes* were sending the acquired values to a monitoring *Sink*. If an *Acknowledgement* packet was not received by a node after sending a data packet, the same packet was retransmitted after a random backoff time.

### 7.5.1 Evaluation of Results

The results obtained for the laboratory rests are shown in Table 7.1. The following characteristics were considered:

- *1* or *2* nodes sending at the same time
- *1000 ms* (*1 Hz*) and *100 ms* (*10 Hz*) packet sending periods (the same values were used also as the sampling periods of each sensor used)
- *1 long* packet containing all the sensor's values (*67 Bytes* packet size) or *4 shorter* packets (*31 Bytes* each):
  - *GPS* data (latitude, longitude, altitude)
  - Accelerometer and Gyroscope data
  - Magnetometer data
  - Other values (battery voltage, temperature, pressure, number of available satellites, *UTC* time)
- *625 bps*, *2.5 Kbps* and *50 Kbps* data rates
- maximum *3* attempts to transmit a packet
- random backoff time bounded between *20* and *100 milliseconds*

The average numbers per minute of successfully received packets for each attempt, lost packets and duplicate packets received were counted and displayed in this table. Based on these values, the following observations can be made:

- For multiple nodes, as well as for a smaller sending periods, multiple packets are lost due to the higher probability of collisions
- When shorter packets are used, more packets are successfully received but at the same time, more packets are lost. Overall, less complete sensors' values batches are successfully received if *4* shorter packets are used instead on a single big one. However, the disadvantage of using a single big packet is given by the need of having all the sensors' values available at the same time (which is not always recommended because different sampling periods are used by different sensors)
- On average, if a packet was not successfully received in the first attempt, then it was lost (i.e. the second and the third attempts did not solve the problem). A cause for this might be the too small backoff period. However, a larger backoff period might cross with the following sampling period
- A higher data rate leads to more sent packets (successfully received or not). Thus, it can be said that the lower data rates, even though they guarantee a higher range and a better bending over the obstacles' corners, represents a "bottle neck" for the packet transmission process. However, for the case of the *50 Kbps* data rate, it can be noticed the high number of lost packets. This is due to the fact that the *Sink* was not ready to receive the new packets as fast as the new ones were arriving

So, from these results it can be concluded that finding the right balance between the sizes and number of sent packets, sending periods, sampling periods, data rates, backoff time values and the number of maximum attempts to send a packet can improve the performances of the *WSN* communications a lot.

## 7.6 Further Developments

- For the *Communication Protocol Stack*:
  - The incremental addition of the missing features of the designed *communication protocol stack* described in Chapter 6, in parallel for the *Sensor Nodes* and for the *Sink* (including the development of the *BLE* communication)
  - Implementation of the *MQTT-SN Publish - Subscribe* mechanism (i.e. implementation of the *Broker*)
  - A deep quantitative analysis of *Sub-1 GHz Physical layer* different configurations with respect to the range covered, the number of collided packets and the number of lost packets when different numbers of *Sensor Nodes* are connected to the network at the same time

Table 7.1: The laboratory test results obtained

| No. Sensor Nodes | Sending Period | No. packets and their type | Data rate | Average successfully received packets per minute per attempt | | | Average lost packets per minute | Average duplicate received packets per minute |
|---|---|---|---|---|---|---|---|---|
| | | | | 1st | 2nd | 3rd | | |
| 1 | 1000 ms | 1 long | 625 bps | 28 | 0 | 0 | 0 | 0 |
| | | | 2.5 Kbps | 46 | 0 | 0 | 0 | 0 |
| | | | 50 Kbps | 59 | 0 | 0 | 0 | 0 |
| | | 4 short | 625 bps | 80 | 0 | 0 | 0 | 0 |
| | | | 2.5 Kbps | 153 | 0 | 0 | 0 | 0 |
| | | | 50 Kbps | 174 | 0 | 0 | 57 | 0 |
| | 100 ms | 1 long | 625 bps | 45 | 0 | 0 | 0 | 0 |
| | | | 2.5 Kbps | 148 | 0 | 0 | 0 | 0 |
| | | | 50 Kbps | 498 | 0 | 0 | 0 | 0 |
| | | 4 short | 625 bps | 104 | 0 | 0 | 0 | 0 |
| | | | 2.5 Kbps | 348 | 0 | 0 | 0 | 0 |
| | | | 50 Kbps | 996 | 0 | 0 | 622 | 0 |
| 2 | 1000 ms | 1 long | 625 bps | 3 | 1 | 0 | 6 | 6 |
| | | | 2.5 Kbps | 45 | 0 | 0 | 0 | 0 |
| | | | 50 Kbps | 58 | 0 | 0 | 0 | 0 |
| | | 4 short | 625 bps | 12 | 0 | 0 | 8 | 13 |
| | | | 2.5 Kbps | 135 | 1 | 0 | 2 | 1 |
| | | | 50 Kbps | 174 | 0 | 0 | 57 | 0 |
| | 100 ms | 1 long | 625 bps | 5 | 0 | 0 | 6 | 9 |
| | | | 2.5 Kbps | 13 | 0 | 0 | 13 | 12 |
| | | | 50 Kbps | 497 | 1 | 0 | 0 | 0 |
| | | 4 short | 625 bps | 42 | 0 | 0 | 42 | 2 |
| | | | 2.5 Kbps | 58 | 1 | 3 | 11 | 14 |
| | | | 50 Kbps | 571 | 6 | 1 | 681 | 11 |

  - A quantitative analysis of the influence of the packets' length in the number of collisions

- For the whole system:
  - Development of the firmware for all the entities present in the system's architecture from Chapter 6
  - Switch to the new version of the hardware for the *Sensor Nodes* (which includes more sensors and an *SD card* slot for more storage memory)
  - Addition of the *caching of the last acquired values* module
  - Addition for the *Sensor Nodes* of the *caching and resending when back in range* mechanism for the situation when the *Gateway* is not in range

- In general:
  - Continuous thorough testing of all the features with various number of *Sensor Nodes* in different environments (e.g. indoor, outdoor with buildings nearby, outdoor in open space, construction site)

# Chapter 8

# Conclusions

The scope of this thesis is to design a *Communication Protocol Stack* used for a *Wireless Sensor Network* on construction sites. In order to take care of the employees, increase the profit and optimize the workflow, it is needed to track the equipments, materials and workers and to monitor the statuses of the machines and tools (i.e. to predict when the reparations are needed), the health of the workers and also the environment (e.g. temperature, pressure). By combining the *IoT* and *WSNs*, an elegant and powerful solution can be developed, which is characterized by predictability, scalability, mobility, realiability, robustness and privacy.
A *WSN* based on the ultra-low power dual-band wireless *CC1350* microcontroller produced by *Texas Instruments* and on the real-time operating system *TI-RTOS*, due to their advantages, is considered. In this thesis, the capabilities of this microcontroller are described thoroughly and a comparison with the newer versions recently announced is done, concluding that it is better to use the basic version due to its stability and large community experience. An analysis and a comparison of different communication protocols and standards used in *IoT* is performed. The most suitable ones for both *Sub-1 GHz* and *2.4 GHz* frequency bands with respect to the initial described requirements and specifications are chosen: *TI 15.4 Stack* (based on the *IEEE 802.15.4 e/g* standard) and *BLE*, respectively. A comparison between different *Application layer* protocols is also done and a decision regarding the usage of one of these on top of the communication protocol is taken: the usage of the version dedicated to *Sensor Networks* of *MQTT* (*MQTT-SN*) would bring the most advantages. Based on the restrictions imposed by the *European Commission* through *CEPT* and *ETSI*, a discussion regarding the possible spectrum access methods, their advantages and disadvantages is done and the most suitable option is chosen: *Listen Before Talk (LBT)* combined with *Adaptive Frequency Agility (AFA)* techniques. Moreover, a brief description of the security vulnerabilities and solutions based on the specific features of the *CC1350* microcontroller is done - the *AES CCM 128 bit shared key* technique represents the best option. Thus, by combining these modules, the most suited *Communication Protocol Stack* for the considered application is obtained.

From the theoretical point of view, the solution fulfills the described requirements. For the practical side, the use case of a highway construction site where the location, the loading level and other characteristics are needed to be monitored is considered. A proprietary already developed board for the *Sensor Nodes* based on the *CC1350* microcontroller is used, alongside the *CC1350 LaunchPad* for the *Gateway's Sink*. A basic version of the communication protocol between multiple *Sensor Nodes* and a *Gateway Node* was implemented (i.e. the same *Physical* layer but reduced *MAC*, *Network* and *Application* layers). Modules for sensors' data acquisition were also implemented and combined with the *communication stack*. A laboratory test analysis regarding the numbers of successfully received, lost and duplicated packets for different number of *Sensor Nodes*, sending periods and packet lengths is performed and the results are discussed. A field test was also performed and the results are described.
In future work, all the features of the designed *Communication Protocol Stack* have to be implemented, tested and added to the core version already developed. Thorough analyses of the network's performances obtained with different values for the tunable parameters (e.g. packet size, sending period, sampling rate, data rate, backoff time, maximum attempts to send a packet) have to be done. The firmware around this *Communication Protocol Stack* for all the entities involved in the designed system have to be implemented and tested. In the end, thorough testings in different environments (e.g. indoor, outdoor with buildings nearby, outdoor in open space, different types of construction sites) have to be performed and the best values for the tunable parameters should be determined.

# Bibliography

[1] Ian F. Akyildiz and Mehmet Can Vuran. *Wireless Sensor Networks*. Chichester, West Sussex, United Kingdom: John Wiley & Sons Ltd, 2010.

[2] Ian Fuat Akyildiz and Mehmet Can Vuran. *Wireless sensor networks*. Ian F. Akyildiz series in communications and networking. Chichester: John Wiley, 2010.

[3] Khaldoun Al Agha, Guy author Pujolle and Tara author Yahiha. *Mobile and Wireless Networks*. 1st. 2016. ISBN: 978-1-119-00756-2. URL: https://books.google.de/books?id=_BzfDAAAQBAJ&pg=PA241&redir_esc=y#v=onepage&q&f=false (visited on 09/08/2018).

[4] LoRa Alliance. *LoRaWAN 1.1 Specification*. 2017. URL: https://lora-alliance.org/sites/default/files/2018-04/lorawantm_specification_-v1.1.pdf (visited on 19/09/2018).

[5] Zigbee Alliance. *The Zigbee Alliance and Thread Group Address IoT Industry Fragmentation with the Availability of the Dotdot Specification over Thread's IP Network*. 2017. URL: https://www.zigbee.org/zigbee-alliance-and-thread-group-address-iot-industry-fragmentation-with-dotdot-over-thread/ (visited on 08/08/2018).

[6] Zigbee Alliance. *The Zigbee Alliance Introduces First Multi-Band IoT Mesh Network Technology for Massive IoT Deployments*. 2017. URL: https://www.zigbee.org/zigbee-introduces-multi-band-iot-mesh-network/ (visited on 10/08/2018).

[7] Aloÿs Augustin, Jiazi Yi and Clausen, Thomas, Townsley, William Mark. 'A Study of LoRa: Long Range & Low Power Networks for the Internet of Things'. In: *Sensors* 16.9 (2016). URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5038744/ (visited on 09/08/2018).

[8] blikoon. *A friendly introduction to XMPP*. 2016. URL: http://www.blikoontech.com/xmpp/xmpp-a-soft-friendly-introduction (visited on 13/08/2018).

[9] Bluetooth. *Bluetooth 5 Quadruples Range, Doubles Speed, Increases Data Broadcasting Capacity by 800%*. 2016. URL: https://www.bluetooth.com/news/pressreleases/2016/06/16/-bluetooth-5-quadruples-rangedoubles-speedincreases-data-broadcasting-capacity-by-800 (visited on 06/08/2018).

[10] Bluetooth. *Topology Options*. 2016. URL: https://www.bluetooth.com/bluetooth-technology/topology-options (visited on 06/08/2018).

[11] R. Branden. 'Requirements for Internet Hosts - Communication Layers'. In: (1989). URL: https://tools.ietf.org/html/rfc1122 (visited on 07/08/2018).

[12] Jennifer Bray and Charles F. Sturman. *Bluetooth: Unifying the Telecommunications and Computing Industries*. 2002. URL: http://www.informit.com/articles/article.aspx?p=27591&seqNum=1 (visited on 06/08/2018).

[13] Building Radar. *EU construction sector in 2018*. 2017. URL: https://buildingradar.com/construction-blog/eu-construction-sector-2018/ (visited on 08/10/2018).

[14] Centre for Wireless Communications, University of Oulu, ed. *NanoIP: The Zen of embedded networking*. 2005. URL: http://www.cwc.oulu.fi/nanoip/index.html (visited on 07/08/2018).

[15] Weng-Fong Cheung, Tzu-Hsuan Lin and Yu-Cheng Lin. 'A Real-Time Construction Safety Monitoring System for Hazardous Gas Integrating Wireless Sensor Network and Building Information Modeling Technologies'. In: *Sensors* 18 2 (2018).

[16] Electronic Communications Committee. *Compatibility of Planned SRD Applications with Currently Existing Radiocommunication Applications in the Frequency Band 868 - 870 MHz: ECC Report 37*. 2004. URL: https://www.ecodocdb.dk/download/760ebce0-dcf6/ECCREP037.PDF (visited on 17/09/2018).

[17] Electronic Communications Committee. *ERC Recommendation 70-03: Relating to the use of Short Range Devices (SRD)*. 2018. URL: https://www.ecodocdb.dk/download/25c41779-cd6e/Rec7003.pdf (visited on 14/09/2018).

[18] RS Components. *11 Internet of Things (IoT) Protocols You Need to Know About*. 2015. URL: https://www.rs-online.com/designspark/eleven-internet-of-things-iot-protocols-you-need-to-know-about.

[19] Vedat Coskun and Busra Ozdenizci. 'A Survey on Near Field Communication (NFC) Technology'. In: *Wireless Personal Communications* 71.3 (2013), pp. 2259–2294.

[20] B. P. Crow et al. 'IEEE 802.11 Wireless Local Area Networks'. In: *IEEE Communications Magazine* 35.9 (1997), pp. 116–126.

[21] Waltenegus Dargie and Christian Poellabauer. *Fundamentals of Wireless Sensor Networks. Theory and Practice*. Chichester, West Sussex, United Kingdom: John Wiley & Sons Ltd, 2010.

[22] Konstantinos Domdouzis, Chimay Anumba and Antony Thorpe. 'Wireless sensor networking in the construction industry implementation scenarios'. In: 2 (Jan. 2005), pp. 789–796.

[23] eTutorials. *Relationship Between Wi-Fi and IEEE 802.11*. URL: http://etutorials.org/Networking/802.11+security.+wi-fi+protected+access+and+802.11i/Part+II+The+Design+of+Wi-Fi+Security/Chapter+7.+WPA+RSN+and+IEEE+802.11i/Relationship+Between+Wi-Fi+and+IEEE+802.11/ (visited on 08/08/2018).

[24] European Construction Industry Federation. *Annual Report 2018*. 2018. URL: http://www.fiec.eu/en/library-619/annual-report-english.aspx (visited on 08/10/2018).

[25] Eurostat. *Accidents at work statistics*. 2018. URL: https://ec.europa.eu/eurostat/statistics-explained/index.php/Accidents_at_work_statistics (visited on 08/10/2018).

[26] WiMAX Forum. *WiMAX Mobile 4G*. URL: http://wimaxforum.org/Page/Initiatives/WiMAX-Advanced (visited on 09/08/2018).

[27] Lou Frenzel. *What's The Difference Between The OSI Seven-Layer Network Model And TCP/IP?* 2013. URL: https://www.electronicdesign.com/what-s-difference-between/what-s-difference-between-osi-seven-layer-network-model-and-tcpip (visited on 07/08/2018).

[28] Ala Al-Fuqaha et al. 'Internet of Things: A Survey on Enabling Technologies, Protocols and Applications'. In: *IEEE Communications Surveys & Tutorials* 17.4 (2015).

[29] Afshar Ganjali. *XMPP: Swiss Army Knife for Internet of Things (IoT)*. 2016. URL: https://blog.securitycompass.com/xmpp-swiss-army-knife-for-internet-of-things-iot-9eff783c44ba (visited on 13/08/2018).

[30] David Gascón. *Security in 802.15.4 and ZigBee networks*. 2009. URL: http://www.libelium.com/security-802-15-4-zigbee/ (visited on 26/09/2018).

[31] Anna Gerber. *Connecting all the things in the Internet of Things*. 2017. URL: https://www.ibm.com/developerworks/library/iot-lp101-connectivity-network-protocols/index.html (visited on 16/08/2018).

[32] IMST GmbH. *Channel Access Rules for SRDs*. 2012. URL: https://www.bundesnetzagentur.de/SharedDocs/Downloads/DE/Sachgebiete/Telekommunikation/Unternehmen_Institutionen/Koexistenzstudie_EN.pdf?__blob=publicationFile&v=2 (visited on 14/09/2018).

[33] Carles Gomez and Josep Paradells. 'Wireless Home Automation Networks: A Survey of Architectures and Technologies'. In: *IEEE Communications Magazine* 48.6 (2010), pp. 92–101.

[34] Google. *Google Beacon Platform: Platform Overview*. 2018. URL: https://developers.google.com/beacons/overview (visited on 10/08/2018).

[35] GSMA. *Extended Coverage – GSM – Internet of Things (EC-GSM-IoT )*. URL: https://www.gsma.com/iot/extended-coverage-gsm-internet-of-things-ec-gsm-iot/ (visited on 10/08/2018).

[36] Domenico de Guglielmo, Simone Brienza and Giuseppe Anastasi. 'IEEE 802.15.4e: A survey'. In: *Computer Communications* 88 (2016), pp. 1–24.

[37] Mike Harwood. *Network+ Exam Cram: Wireless Networking*. 2009. URL: http://www.pearsonitcertification.com/articles/article.aspx?p=1329709.

[38] HiveMQ. *All you need to know about MQTT*. 2015. URL: http://forkbomb-blog.de/2015/all-you-need-to-know-about-mqtt (visited on 20/09/2018).

[39] Charles Hodgdon. *Adaptive Frequency Hopping for Reduced Interference between Bluetooth and Wireless LAN*. 2003. URL: https://www.design-reuse.com/articles/5715/adaptive-frequency-hopping-for-reduced-interference-between-bluetooth-and-wireless-lan.html (visited on 19/09/2018).

[40] Urs Hunkeler, Hong Linh Truong and Andy Stanford-Clark. 'MQTT-S — A publish/subscribe protocol for Wireless Sensor Networks'. In: *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE '08)*. IEEE, 20082008, pp. 791–798.

[41] Magdy Ibrahim and Osama Moselhi. 'Wireless Sensor Networks Configurations for Applications in Construction'. In: *Procedia Engineering* 85 (2014), pp. 260–273. URL: http://www.sciencedirect.com/science/article/pii/S1877705814019171.

[42] Magdy Ibrahim and Osama Moselhi. 'Wireless Sensor Networks Configurations for Applications in Construction'. In: *Procedia Engineering* 85 (2014), pp. 260–273. ISSN: 18777058.

[43] Texas Instruments. *Bluetooth Low Energy Software Developer's Guide*. URL: http://dev.ti.com/tirex/content/simplelink_cc13x0_sdk_1_30_00_06/docs/blestack/html/cc1350/index.html (visited on 13/09/2018).

[44] Texas Instruments. *CC1350 SimpleLink Ultra-Low Power Dual Band Wireless Microcontroller*. URL: http://www.ti.com/product/cc1350 (visited on 17/08/2018).

[45] Texas Instruments. *CC1350 SimpleLink™ Ultra-Low-Power Dual-Band Wireless MCU Datasheet*. URL: http://www.ti.com/lit/ds/symlink/cc1350.pdf (visited on 07/05/2018).

[46] Texas Instruments. *CC13x0 Proprietary RF User's Guide*. URL: http://dev.ti.com/tirex/content/simplelink_cc13x0_sdk_1_00_00_13/docs/proprietary-rf/html/cc13x0/index.html (visited on 08/05/2018).

[47] Texas Instruments. *CC13x0 TI 15.4-Stack User's Guide*. URL: http://dev.ti.com/tirex/content/simplelink_cc13x0_sdk_2_20_00_38/docs/ti154stack/html/ti154stack-guide/index-cc13x0.html (visited on 11/05/2018).

[48] Texas Instruments. *CC13x0, CC26x0 SimpleLink™ Wireless MCU Technical Reference Manual*. 2015. URL: http://www.ti.com/lit/ug/swcu117h/swcu117h.pdf (visited on 07/05/2018).

[49] Texas Instruments. *EasyLink layer*. URL: http://dev.ti.com/tirex/content/simplelink_cc13x0_sdk_1_00_00_13/docs/proprietary-rf/html/easylink/index.html (visited on 08/05/2018).

[50] Texas Instruments. *Getting Started With the CC13xx and CC26xx Sensor Controller*. 2017. URL: http://www.ti.com/lit/an/swra578/swra578.pdf (visited on 07/05/2018).

[51] Texas Instruments. *GPS Tracking Using Sub-1GHz Devices*. Application Report. Texas Instruments, 2017. URL: http://www.ti.com/lit/an/swra584/swra584.pdf.

[52] Texas Instruments. *Network Algorithms for LPWAN*. Technical Brief. Texas Instruments, 2017. URL: http://www.ti.com/lit/an/sprt734/sprt734.pdf.

[53] Texas Instruments. *POSIX Project Zero*. URL: http://dev.ti.com/tirex/content/simplelink_academy_cc13x0sdk_2_10_02_10/modules/rtos/posix_project_zero/posix_project_zero.html (visited on 09/05/2018).

[54] Texas Instruments. *SimpleLink™ Dual-Band CC1350 Wireless MCU LaunchPad Development Kit*. URL: http://www.ti.com/tool/launchxl-cc1350 (visited on 17/04/2018).

[55] Texas Instruments. *SimpleLink MCU SDK Driver API Reference: Drivers*. URL: http://dev.ti.com/tirex/content/simplelink_cc13x0_sdk_2_20_00_38/docs/tidrivers/doxygen/html/index.html (visited on 26/09/2018).

[56] Texas Instruments. *SimpleLink MCU SDK User's Guide*. URL: http://dev.ti.com/tirex/content/simplelink_cc13x0_sdk_1_40_00_10/docs/simplelink_mcu_sdk/Users_Guide.html#introduction-to-simplelink-mcu-sdk (visited on 11/05/2018).

[57] Texas Instruments. *SimpleLink™ Multi-band CC1352P Wireless MCU With Integrated Power Amplifier*. URL: http://www.ti.com/product/CC1352P/description (visited on 04/09/2018).

[58] Texas Instruments. *SimpleLink Multi-Band CC1352R Wireless MCU*. URL: http://www.ti.com/product/CC1352R/description (visited on 04/09/2018).

[59] Texas Instruments. *SimpleLink™ Sub-1 GHz CC13x0 Software Development Kit*. URL: http://www.ti.com/tool/SIMPLELINK-CC13X0-SDK (visited on 18/04/2018).

[60] Texas Instruments. *SimpleLink™ Sub-1 GHz wireless MCUs*. URL: http://www.ti.com/wireless-connectivity/simplelink-solutions/sub-1-ghz/overview.html (visited on 07/05/2018).

[61] Texas Instruments. *SimpleLinkTM MCU Platform*. URL: http://www.ti.com/wireless-connectivity/simplelink-solutions/overview/overview.html (visited on 08/05/2018).

[62] Texas Instruments. *SYS/BIOS API Documentation*. URL: http://dev.ti.com/tirex/content/simplelink_cc13x0_sdk_2_20_00_38/docs/tirtos/sysbios/docs/cdoc/index.html (visited on 01/10/2018).

[63] Texas Instruments. *The SimpleLink™ Software Development Kit*. URL: http://www.ti.com/wireless-connectivity/simplelink-solutions/overview/software.html (visited on 02/05/2018).

[64] Texas Instruments. *TI Resource Explorer*. URL: http://dev.ti.com/tirex/#/ (visited on 07/05/2018).

[65] Texas Instruments. *TI-RTOS 2.20 User's Guide*. 2016. URL: http://www.ti.com/lit/ug/spruhd4m/spruhd4m.pdf (visited on 08/05/2018).

[66] Texas Instruments. *TI-RTOS Basics*. URL: http://dev.ti.com/tirex/content/simplelink_academy_cc2640r2sdk_1_12_00_00/modules/tirtos_basics/tirtos_basics.html (visited on 09/05/2018).

[67] Texas Instruments. *TI-RTOS Drivers: sensors Directory Reference*. URL: http://software-dl.ti.com/dsps/dsps_public_sw/sdo_sb/targetcontent/tirtos/2_21_01_08/exports/tirtos_full_2_21_01_08/products/tidrivers_cc13xx_cc26xx_2_21_01_01/docs/doxygen/html/dir_df123555319038bcf9f919c48d80be.html (visited on 01/10/2018).

[68] Texas Instruments. *Understanding security features for SimpleLink™ Sub-1 GHz CC13x0 MCUs*. 2017. URL: http://www.ti.com/lit/ml/swpb017a/swpb017a.pdf (visited on 25/09/2018).

[69] Texas Instruments. *Why Sub-1 GHz?* 2016. URL: http://www.ti.com/lit/ml/swrw006/swrw006.pdf (visited on 09/08/2018).

[70] Texas Instruments. *Wireless Door and Window Sensors With the Sub-1GHz SimpleLink™ Wireless MCU*. Application Report. Texas Instruments, 2018. URL: http://www.ti.com/lit/an/swra575a/swra575a.pdf.

[71] Texas Instruments. *Wireless Motion Detector With Sub-1 GHz SimpleLink™ Wireless MCU*. Application Report. Texas Instruments, 2018. URL: http://www.ti.com/lit/an/swra562b/swra562b.pdf.

[72] Texas Instruments. *Wireless Smoke Alarms With SimpleLink™ Sub-1 GHz Wireless MCU*. Application Report. Texas Instruments, 2018. URL: http://www.ti.com/lit/an/swra567b/swra567b.pdf.

[73] International Electrotechnical Commission. *Internet of Things: Wireless Sensor Networks: White Paper*. 2014. URL: http://www.iec.ch/whitepaper/pdf/iecWP-internetofthings-LR-en.pdf (visited on 08/05/2018).

[74] Kendall Jones. *Construction Equipment Theft*. 2017. URL: https://www.constructconnect.com/blog/operating-insights/high-cost-construction-equipment-theft/ (visited on 08/10/2018).

[75] Philo Juang et al. 'Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with ZebraNet'. In: *Tenth International Conference on Architectural Support for Programming Languages and Operating Systems* (2002).

[76] Charles M. Kozierok. *The TCP/IP Guide*. 2003. URL: http://www.tcpipguide.com/index.htm (visited on 07/08/2018).

[77] M. Kuzlu, M. Pipattanasomporn and S. Rahman. 'Review of Communication Technologies for Smart Homes/Building Applications'. In: *IEEE Innovative Smart Grid Technologies - Asia (ISGT ASIA)* (2015).

[78] Silicon Labs. *Zigbee & Thread Community and Support*. URL: https://www.silabs.com/community/wireless/zigbee-and-thread (visited on 19/09/2018).

[79] Laird. *BLE Overview*. 2013. URL: http://www.summitdata.com/blog/ble-overview/.

[80] Jørgen Langfelt, Nick Smith and Svein Vetti. *Ultra-Low Power Designs With the CC13x2 and CC26x2 Sensor Controller*. 2018. URL: http://www.ti.com/lit/an/swra598/swra598.pdf (visited on 07/05/2018).

[81] Jin-Shyan Lee, Yu-Wei Su and Chung-Chou Shen. 'A Comparative Study of Wireless Protocols: Bluetooth, UWB, ZigBee, and Wi-Fi'. In: *IECON 2007 - 33rd Annual Conference of the IEEE Industrial Electronics Society* (2007), pp. 46–51.

[82] Tomas Lennvall, Stefan Svensson and Frederik Hekland. 'A Comparison of WirelessHART and ZigBee for Industrial Applications'. In: *IEEE International Workshop on Factory Communication Systems* (2008).

[83] Matthew Loy, Raju Karingattil and Louis Williams. *ISM-Band and Short Range Device Regulatory Compliance Overview*. 2005. URL: http://www.ti.com/lit/an/swra048/swra048.pdf (visited on 14/09/2018).

[84] MikroElektronika. *Bluetooth Low Energy - Part 1: Introduction To BLE*. 2016. URL: https://www.mikroe.com/blog/bluetooth-low-energy-part-1-introduction-ble (visited on 13/09/2018).

[85] MikroElektronika. *GNSS 4 click*. URL: https://www.mikroe.com/gnss-4-click (visited on 01/10/2018).

[86] Andreas F. Molisch. *Wireless Communications*. Chichester, West Sussex, United Kingdom: John Wiley & Sons Ltd, 2011.

[87] MQTT. *MQTT*. URL: http://mqtt.org/ (visited on 05/09/2018).

[88] Sarfraz Nawaz et al. 'Monitoring A Large Construction Site Using Wireless Sensor Networks'. In: *Proceedings of the 6th ACM Workshop on Real World Wireless Sensor Networks*. New York, NY, USA: ACM, 2015, pp. 27–30. URL: http://doi.acm.org/10.1145/2820990.2820997.

[89] David Nield. *Bluetooth 5: everything you need to know*. 2016. URL: https://www.techradar.com/news/networking/bluetooth-5-everything-you-need-to-know-1323060 (visited on 06/08/2018).

[90] Eclipse Paho. *The Eclipse Paho project*. URL: https://www.eclipse.org/paho/ (visited on 11/10/2018).

[91] Ayan Pahwa. *Communication Protocols for the Internet of Things: A Few Choices*. 2017. URL: https://opensourceforu.com/2017/10/communication-protocols-internet-things-choices/ (visited on 13/08/2018).

[92] Gerardo Pardo-Castellote, Bert Farabaugh and Rick Warren. *An Introduction to DDS and Data-Centric Communications*. 2015. (Visited on 13/08/2018).

[93] Kevin Parrish. *ZigBee, Z-Wave, Thread and WeMo: What's the Difference?* 2017. URL: https://www.tomsguide.com/us/smart-home-wireless-network-primer,news-21085.html (visited on 08/08/2018).

[94] Alisa Pfeil. *Introducing Thread: A New Wireless Networking Protocol For The Home*. 2014. URL: https://www.threadgroup.org/news-events/press-releases/ID/20/Introducing-Thread-A-New-Wireless-Networking-Protocol-for-the-Home (visited on 08/08/2018).

[95] D. Porcino and W. Hirt. 'Ultra-wideband radio technology: potential and challenges ahead'. In: *IEEE Communications Magazine* 41.7 (2003), pp. 66–74.

[96] Qualcomm. *Qualcomm Wi-Fi SON and distributed networking*. URL: https://www.qualcomm.com/solutions/networking/features/wi-fi-son (visited on 08/05/2018).

[97] Brian Ray. *What is LTE-M?* 2017. URL: https://www.link-labs.com/blog/what-is-lte-m (visited on 10/08/2018).

[98] Usman Raza, Parag Kulkarni and Mahesh Sooriyabandara. 'Low Power Wide Area Networks: An Overview'. In: *IEEE Communications Surveys & Tutorials* 19.2 (2017), pp. 855–873.

[99] K. Ganesh Reddy and P. Santhi Thilagam. 'MAC layer security issues in wireless mesh networks'. In: 1715 (Mar. 2016). URL: https://aip.scitation.org/doi/abs/10.1063/1.4942710.

[100] Farzad Samie, Lars Bauer and Jörg Henkel. 'IoT technologies for embedded computing: A survey'. In: *2016 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)* (2016).

[101] Shadi Al-Sarawi et al. 'Internet of Things (IoT) Communication Protocols: Review'. In: *2017 8th International Conference on Information Technology (ICIT)* (2017), pp. 685–690.

[102] Glenn Schatz. *An Overview of Narrowband IoT (NB-IoT)*. 2018. URL: https://www.link-labs.com/blog/overview-of-narrowband-iot (visited on 10/08/2018).

[103] Jaydip Sen. *Security in Wireless Sensor Networks*. 2013. URL: http://arxiv.org/pdf/1301.5065v1.

[104] Francois Sforza. 'Fractional-N synthesized chirp generator'. US8406275B2. URL: https://patents.google.com/patent/US8406275 (visited on 09/08/2018).

[105] Wentao Shang et al. *Challenges in IoT Networking via TCP/IP Architecture*. 10th Feb. 2016.

[106] Zach Shelby et al. 'NanoIP: The Zen of Embedded Networking'. In: *IEEE International Conference on Communications* 2 (2003).

[107] Xuesong Shen, Wu Chen and Ming Lu. 'Wireless Sensor Networks for Resources Tracking at Building Construction Sites'. In: *Tsinghua Science & Technology* 13 (2008), pp. 78–83. URL: http://www.sciencedirect.com/science/article/pii/S1007021408701305.

[108] Xuesong Shen, Wu Cheng and Ming Lu. 'Wireless sensor networks for resources tracking at building construction sites'. In: *Tsinghua Science and Technology* 13.S1 (2008), pp. 78–83.

[109] SiliconLabs. *UG103.11: Thread Fundamentals*. URL: https://www.silabs.com/documents/public/user-guides/ug103-11-appdevfundamentals-thread.pdf (visited on 08/08/2018).

[110] Andy Standford-Clark and Hong Linh Truong. *MQTT For Sensor Networks (MQTT-SN) - Protocol Specification Version 1.2*. 2013. URL: http://mqtt.org/new/wp-content/uploads/2009/06/MQTT-SN_spec_v1.2.pdf (visited on 20/09/2018).

[111] Accuware Support. *Bluetooth Beacon Tracker – Accuracy*. URL: https://www.accuware.com/support/bluetooth-beacon-tracker-accuracy/ (visited on 25/04/2018).

[112] Gonçalo Nuno Sol Teixeira and Laura Margarita Rodríguez Peralta. 'Comparing ZigBee, Bluetooth, UWB, and Wi-Fi'. In: *Encyclopedia of Networked and Virtual Organizations* (2008), pp. 288–297.

[113] Texas Instruments, ed. *Using the Stellaris Ethernet Controller with Micro IP (uIP): Application Note*. 2010. URL: http://www.ti.com/lit/an/spma026b/spma026b.pdf (visited on 07/08/2018).

[114] Clemens Valens. *LoRaWAN security vulnerabilities exposed*. 2016. URL: https://www.elektormagazine.com/news/lorawan (visited on 25/09/2018).

[115] Vesternet. *Understanding Z-Wave Networks, Nodes & Devices*. 2012. URL: https://www.vesternet.com/resources/technology-indepth/understanding-z-wave-networks/ (visited on 06/08/2018).

[116] Svein Vetti, Jonas Olsson and Jeanna Copley. *Sub-1 GHz long-range communication and smartphone connection for IoT applications*. URL: http://www.ti.com/lit/wp/swry026/swry026.pdf (visited on 07/05/2018).

[117] Kumaran Vijayasankar and Prasad Movva. *TI 15.4-Stack Frequency Hopping Mode FCC Compliance: Application Report*. 2016. URL: http://www.ti.com/lit/an/swra529a/swra529a.pdf (visited on 17/09/2019).

[118] Kumaran Vijayasankar and Roberto Sandre. *Frequency hopping for long-range IoT networks*. 2016. URL: http://www.ti.com/lit/wp/swry025/swry025.pdf (visited on 17/09/2018).

[119] Vizocom. *What is WiMAX and How Does it Differ from WiFi?* 2016. URL: http://www.vizocom.com/blog/wimax-differ-wifi/ (visited on 09/08/2018).

[120] Peng Wu et al. 'Transition from IPv4 to IPv6: A State-of-the-Art Survey'. In: *IEEE Communications Surveys & Tutorials* 15.3 (2013), pp. 1407–1424.

[121] Jennifer Yick, Biswanath Mukherjee and Dipak Ghosal. 'Wireless Sensor Network Survey'. In: *Comput. Netw.* 52.12 (Aug. 2008), pp. 2292–2330.

[122] Pei Zheng et al. *Wireless Networking Complete*. Burlington, MA, USA: Elsevier, 2010.

[123] Tobias Zillner. *Zigbee Exploited: The good, the bad and the ugly*. 2015. URL: https://www.blackhat.com/docs/us-15/materials/us-15-Zillner-ZigBee-Exploited-The-Good-The-Bad-And-The-Ugly-wp.pdf (visited on 25/09/2018).

# Appendix A

# Requirements and Specifications

Starting from the general properties of a network (i.e. predictability, scalability, mobility, robustness and privacy), a particular list of requirements and specifications for the considered case of a *CC1350* microcontroller - based *WSN* used in the construction industry for various purposes (e.g. tracking and monitoring tools, vehicles, workers) was created (Table A.1).

Table A.1: Description of the Requirements and Specifications

| **Requirements** | **Specifications** |
|---|---|
| Fulfill the reporting time constraints | Due to the fact that the *WSN* will be used for monitoring and not for real-time control of a process, the deadline of a new measured value (or the aggregate value of multiple measured values) to be available in the Gateway Node should be between 1 and 15 $minutes$ [42]. This represents a soft constraint: if the constraint is not met, the usefulness decreases but this not leads to a system failure |
| Add / Remove easily Sensor Nodes to / from the network | |
| Each sensor should store the last acquired values | The values should be stored in a circular manner, using the whole available memory (the *TI CC1350 microcontroller* has a *Flash memory* of 128 $KB$ but an external *Flash memory* and/or an *SD card* can be connected and used) |

| | |
|---|---|
| Possibility to connect with a device (e.g. smartphone, tablet) to each Sensor Node or Gateway Node individually for setting it up (e.g. add / remove it to / from the network) and for visualizing the last acquired values stored in the memory | The connection will be done using an application developed for the *Android* mobile operating system |
| Ensure a sufficient spatial spread of the network | According to the specifications, the range of the *TI CC1350 microcontroller* is up to $20\,km$. This means that the total area covered by the network for a star topology (which is the maximum area covered by the Gateway Node) is up to $1250\,km^2$ (without considering any wireless communication issues). For a mesh network topology, the total area covered by the network could be bigger because the transmitted information can be routed through multiple Sensor Nodes to the Gateway Node and not sent directly |
| Ensure the security of the network | Usage of message data encryption, protected addition / removal of Sensor Nodes |
| Ensure the correctness of data | Usage of the the Cyclic Redundancy Check (CRC) |
| Correct the transmission errors | Retransmission of missing packets |
| Correct the loss of information | Retransmission of missing packets |
| Possibility to add new Sensor Nodes to the network | According to the specifications of the *TI 15.4-Stack* within the *SimpleLink Sub-1 GHz CC1350 software development kit (SDK)*, the maximum number of connections a node can have is 150. This limitation is due to the size of the RAM and Flash memory when using the MAC layer security. If the security of the MAC layer is disabled, then the maximum number of connections a node can have can be increased up to 65536. Another solution to increase the number of nodes of the network is to combine multiple smaller networks by connecting their Gateway Nodes together in a tree structure |

| | |
|---|---|
| The data rate should be sufficient to fulfill the reporting time constrains | Due to the fact that the minimum reporting time for a Sensor Node is $1\,minute$ and also the messages transmitted in the network are small (a message contains only one measured value), the data rate does not have to be a high one (i.e. $1\,Kbps$ is sufficient) |
| Ensure mobility of Sensor Nodes | The Wireless Sensor Network should continue to function without any limitations if the Sensor Nodes are changing their position (e.g. the sensors placed on tools), as long as they remain in the range of the Gateway Node |
| Low energy consumption | The sensor nodes should not acquire and send data with a high frequency in order to consume as less energy as possible |
| All the nodes of the network should be able to both send and receive messages | |
| The Gateway Node should have an updated list of the active Sensor Nodes | |
| Network Nodes Tracking [1] | The accuracy of tracking using Bluetooth beacons is around $3\,meters$ but this depends on the solution used [111] |

[1] Resource tracking using Bluetooth beacons is an optional feature

# Appendix B

# Abbreviations

ADC = Analog-to-Digital Converter

AES = Advanced Encryption Standard

AFA = Adaptive Frequency Agility

AMQP = Advanced Message and Queuing Protocol

APS = Application Support Sublayer

CAN = Controller Area Network

CBC = Cipher Block Chaining

CBC-MAC = Cipher Block Chaining - Message Authentication Code

CCM = Counter with CBC-MAC

CDMA = Code-Division Multiple Access

CEPT = The European Conference of Postal and Telecommunications Administrations

CoAP = Constrained Application Protocol

CSMA = Carrier-Sense Multiple Access

CSMA/CA = Carrier-Sense Multiple Access with Collision Avoidance

CSMA/CD = Carrier Sense Multiple Access with Collision Detection

CSMA/CR = Carrier Sense Multiple Access with Collision Resolution

CTR = Counter

DCPS = Data-Centric Publish-Subscribe

DDS = Data Distribution Service

DFKI = Deutsche Forschungszentrum für Künstliche Intelligenz GmbH

DH1CF = direct hash channel function

DLRL = Data-Local Reconstruction Layer

DoS = Denial-of-Service

DSL = Digital Subscriber Line

DSSS = Direct Sequence Spread Spectrum

DTLS = Datagram Transport Layer Security

EC-GSM-IoT = Extended Coverage-GSM-IoT

ECB = Electronic Codebook

ECC = Elliptic Curve Cryptography

ECDH = Elliptic Curve Diffie-Hellman

ECDSA = Elliptic Curve Digital Signature Algorithm

ECDSA = Elliptic Curve Digital Signature Algorithm

ECJPAKE = Elliptic Curve Password Authenticated Key Exchange by Juggling

EDGE = Enhanced Data Rates for GSM Evolution

ETSI = The European Telecommunications Standards Institute

FAN = Field Area Network

FDMA = Frequency Division Multiple Access

FFD = Full-Function Device

FHSS = Frequency-Hopping Spread Spectrum

GFSK = Gaussian Frequency-Shift Keying

GMK = Group Master Key

GNSS = Global Navigation Satellite System

GPIO = General Purpose Input Output

GPRS = General Packet Radio Service

GSM = Global System for Mobile communications

HART = Highway Addressable Remote Transducer Protocol

HSPA = High Speed Packet Access

HTTP = Hypertext Transfer Protocol

I2C = Inter-Integrated Circuit Protocol

IMU = Inertial Measurement Unit

IoT = Internet of Things

IP = Internet Protocol

ISM = Industrial, Scientific and Medical radio bands

KDC = Key Distribution Center

LBT = Listen Before Talk

LoWPAN = Low-Power Wireless Personal Area Network

LPWAN = Low-Power Wide Area Network

LR-WPAN = Low-Rate Wireless Personal Area Network

LTE = Long-Term Evolution

LTE-M = Long Term Evolution category M1

MAC = Medium Access Control

MAC = Medium Access Control

MIC = Message Integrity Code

MOM = Message-Oriented Middleware

MQTT = Message Queue Telemetry Transport

MQTT-SN = MQTT for Sensor Networks

MTU = Maximum Transmission Unit

NB-IoT = Narrowband-IoT

NFC = Near Field Communication

NVS = Non-Volatile Storage

OFDM = Orthogonal Frequency-Division Multiplexing

PA = PAN Advertisement

PAS = PAN Advertisement Solicit

PC = PAN Configuration

PCS = PAN Configuration Solicit

PHY = Physical (layer)

QoS = Quality of Service

REST = Representational State Transfer

RFD = Reduced-Function Device

RFID = Radio Frequency Identification

RPL = IPv6 Routing Protocol for Low-Power and Lossy Networks

RSA = Rivest–Shamir–Adleman (Algorithm)

RSSI = Received Signal Strength Indicator

SHA = Secure Hash Algorithm

TCP = Transmission Control Protocol

TDMA = Time Division Multiple Access

TDMA = Time-Division Multiple Access

UART = Universal Asynchronous Receiver-Transmitter

UDP = User Datagram Protocol

uIP = micro IP = micro Internet Protocol

UMTS = Universal Mobile Telecommunications System

UNB = Ultra Narrow Band

UWB = Ultra-Wideband

Wi-SUN = Wireless Smart Utility Network

WiMAX = Worldwide Interoperability for Microwave Access

WLAN = Wireless Local Area Network

WMAN = Wireless Metropolitan Area Network

WPAN = Wireless Personal Area Network

XML = Extensible Markup Language

XMPP = Extensible Messaging and Presence Protocol

XMPP-IoT = XMPP for IoT