Escola Politècnica Superior de Castelldefels

UNIVERSITAT POLITÈCNICA DE CATALUNYA

FINAL DEGREE PROJECT

TITLE: Design and implementation of an opportunistic network based on IEEE 802.15.4

DEGREE: Enginyeria Tècnica de Telecomunicació, especialitat Sistemes de Telecomunicació

AUTHOR: Carlos Serra Monje

DIRECTOR: Lluís Casals, Carles Gómez.

DATE: october 18th 2010.

A mi querida familia y amigos,

Abstract

This project is intended to apply in a real and functional scenario the opportunistic wireless network concept. This new paradigm of wireless network was born from the popularity and development of new wireless technologies and considerable reductions in required infraestructure to build reliable networks with high quality communication. Thanks these improvements, opportunistic networks are being studied and developed. Somo of these are able to estimate the most probable moments of succesful communication establishment and reduce the functional cost whilst the comunication is not possible.

The main objective of this project is to create a reduced network able to periodically collect data from its environment, e.g. temperature or light level, and store it until its transmission is possible. The most important goals are saving as much energy as possible while the data collected must be reliable for further processing.

Finally, the deployment has been based on the IEEE 802.15.4 radio protocol using as hardware base the low energy consumption CC2430 chipset of Texas Instruments. As a relevant detail, a specific hardware has been developed to be in charge of the sensing part. The programming language has been C and the code has been developed under the TI-MAC protocol stack developed by Texas Instruments.

Este proyecto tiene como finalidad la aplicación a un entorno real y funcional del concepto de red oportunística inalámbrica. Este nuevo paradigma de red sin cable nace de la popularidad y gran desarrollo de las tecnologías inalámbricas y de las considerables reducciones de infraestrucutra necesaria para crear redes fiables y con buena calidad de comunicación. Gracias a estos avances, se estudian y desarrollan las redes oportunísticas, capaces de estimar momentos de mayor probabilidad de comunicación para reducir el coste de funcionamiento mientras la comunicación es inviable.

El objetivo principal de este proyecto es crear una pequeña red que sea capaz de adquirir datos de su entorno periódicamente, por ejemplo temperatura y nivel de luz, y almacenarlos hasta que las condiciones hagan que su transmisión sea posible. Las principales exigencias del proyecto son alargar la vida de las baterias lo máximo posible ala vez que obtener datos fiables para un futuro estudio de ellos y garantizar un cierto flujo de datos a las estaciones que se van a encargar de procesarlos.

Para desarrollar dicho sistema nos hemos basado en el protocolo radio IEEE 802.15.4 utilizando como principal hardware el chip de bajo consumo CC2430 de Texas Instruments. Como dato relevante, en combinación se ha desarrollado un hardware específico que se encarga de la parte de sensado. El lenguaje de programación utilizado es C y el codigo se ha desarrolado bajo la pila de protocolos TI-MAC desarrollada por el fabricante.

Acknowledgements

I would like to thank my supervisors Lluis Casals and Carles Gómez for accepting this project and guidind me through its development. I want to express my gratitude for all the help and support. In compensation to all the love and support provided by my family and friends my most deserved acknowledgement is written in our mother tong:

“ En primer lugar quiero agradecer a mi familia, especialmente a mis padres Ángel y Antonia, todo el amor y comprensión que me han ofrecido a lo largo de todo este proyecto y el camino hasta él, ¡os quiero, sin vosotros esto nunca hubiera sido posible!

En segundo lugar quiero agradecer especialmente la ayuda, el ánimo y la amistad de mi gran compañero Javier Rubio a lo largo de la carrera y el desarrollo de este proyecto, gracias por hacer que todo sea más sencillo. Quiero agradecer también a todos los compañeros de clase con los que he recorrido el camino hasta este proyecto y con los que he compartido muchos momentos académicos y extra-académicos que guardaré para siempre: David Ibáñez, Fran Durán, Javier López, Jonathan González, Arnau Raventós, Ricardo Herrero, Javi Gavilán, Roser Farré. Nunca olvidaré las visperas de exámenes, las noches en Castelldefels, las tardes en Beethoven ni las mañanas en la cafetería, de nuevo ¡Gracias!

Especial agradecimiento a Ciclos Valbuena por ayudarme en los primeros años dándome base profesional. Gracias a Pasiona Consulting y a David Teixidó por darme la oportunidad de trabajar en mi primer proyecto tecnológico, gracias por la ayuda y el soporte a Pablo Luengo, David Gutiérrez, Javier de Silóniz, Silvia Lacroizette, Josep Mª Salabert, Martin Schdmit y Ricard Gallego.

Por último, gracias a Verónica Cabanillas por el amor, el cariño, la comprensión, la ayuda y la paciencia que me ha dado durante estos años. ¡Nunca lo olvidaré! ”

Carlos Serra Monje
Barcelona, 18, October 2010

“Anyone who has never made a mistake has never tried anything new”

- Albert Einstein

Contents

List of Figures

List of Tables

Chapter 1

Motivation

The wireless networks are experimenting a great boom since new developments are being published and succesfully tested in a wide range of applications, for example, to provide sensing or control solutions for environmental and industrial stuff. The success of these developments lies on the low cost of the deployments and the flexibility provided by the absence of a hard infraestructure with a complex maintenance.

Radio standards such as Zigbee, BlueTooth or IEEE 802.15.4 have been working hard to bring the support to thisthe Infact, this flexibility is being exploited by researchers to implement these networks in different scenarios where is necessary to monitor what is going on, on a regular basis, and it is required a cheap solution because the benefits of the results could not compensate the deployment of infraestructure.

Besides, returning to the examples and focusing on the projects main aim, for environmental solutions it may be required to monitor the weather conditions in hard accessible areas and it is possible to use nodes consisting of handheld devices carried by people or vehicles and make use of devices mobility to distribute data: information can be stored and passed taking advantage of the opportunity of establishing wireless link when contact between devices is met.

At this point it is possible to talk about opportunistic networking, whith the special requirement of predicting the most probable chances of contact between devices to create a smart protocol able to deal with the variances of these predictions and the random nature of this concept. In the case of this work, the goal is to use this concept to take profit of a mobile device to collect environmental information.

1.1 Objective

Design, build, and test a small wireless opportunistic network of a reduced number of nodes called end-devices or sensor-devices capable of self configuration and reporting to a notebook computer. We have designed a particular scenario taking advantage of a regular bus line. The bus is equipped with a network coordinator that will be receiving information coming from some sensors placed around the areas nearby its route. This sensors will be regularly picking measurements from environment and storing this data. Devices associated to the sensors should be able to establish radiolink and transmit the stored data to the coordinator when the bus is in their range. There are two main premises: saving as much energy as possible and success in as much communication attempts as possible. To achieve this two goals we are going to use the superframe structure defined by the IEEE802.15.4 in the beacon-enabled networks and the possibility of sleeping and waking up of the nodes available in the CC2430 of Texas Instruments.

Figure 1.1 shows the system flow motion: when the coordinator is in any of the devices' range they must be ready to establish the radiolink in order to do the data exchange To achieve this goal it is needed to implement an Opportunistic protocol based on a preliminary estimation of how often the bus will be in range with each of the devices. In addition, the protocol must be tolerant enough to permit delay on the predicted communication chances and moreover, tolerant with the lost of punctual ones. This paradigm of Opportunistic protocol is known as Delay Tolerant Networks, in this scenario the development of this operation mode is based on the sampling theory. End-Devices are required to perform a certain number of scans within an estimated time while coordinator broadcasts beacons, as part of the superframe structure feature, ensuring that at least one beacon will be found by one of the scans. To accomplish the main commitments of the project the number of scans must be little enough to save as much battery as possible and big enough to ensure the meeting between coordinator and End-devices. Finally, the time to exchange the data is the other main handicap in this kind of network. The time in range will be delimited by the speed of the bus and the range of the devices. Once the radiolink is established as result of the coordinator-and-device meeting, the data exchange must be guaranteed. To success in this task, the protocol will lean in the CSMA-CA algorithm, also as part of the superframe structure definition, which defines a slotted structure to transmit data and , as explained in next chapter, an algorithm to get medium access which will be very helpful to commit the data exchange.

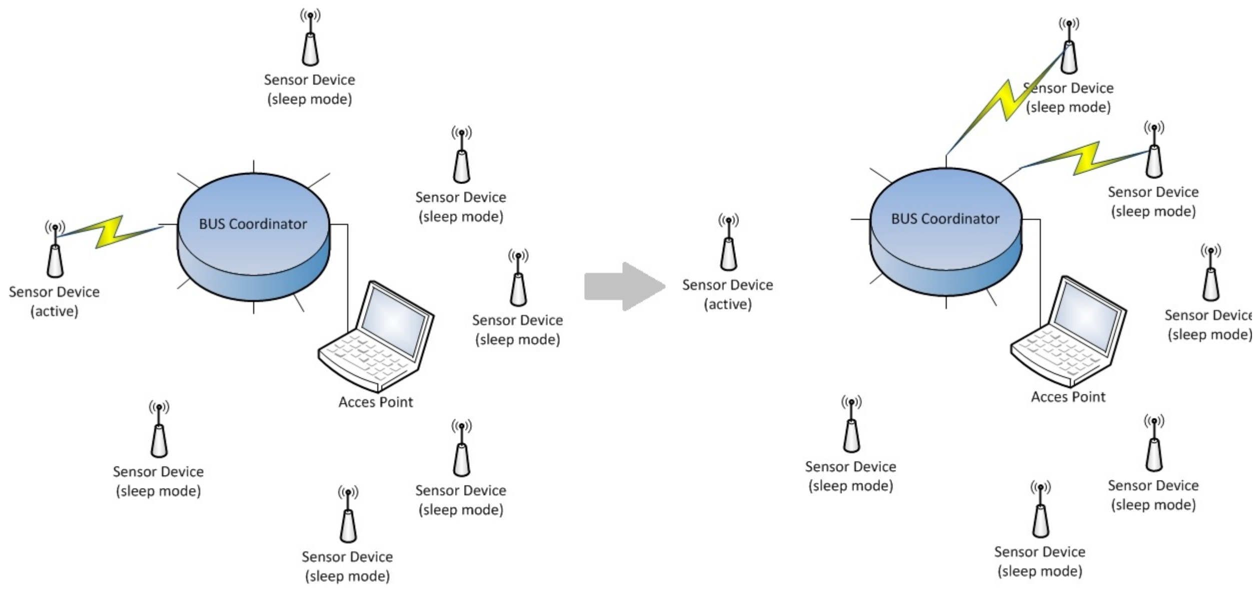


Figure 1.1: Scenario

1.2 Project organisation

The following chapter is about to describe the most important features of IEEE802.15.4 in order to give to the reader the necessary background to the correct understanding of what is explained therefore. Third chapter will introduce the specifications and objectives of the final scenario followed by a detailed study of it, so the following decisions when the software development is explained could be justified. In chapter 4 the hardware deployment is presented, overviewing the whole system and detailing all the built up process. In chapter 5 it is explained the opportunistic flow chart designed with TIMAC protocol stack and all the software generated to conclude this task. In the last chapter there is a section with future work and improvements to this work and some conclusions.

Chapter 2

IEEE 802.15.4 radio technology

In this chapter the radio standard IEEE 802.15.4 is going to be introduced as the base of the project development, highlighting the most important features used in this project.

2.1 Introduction

The low rate WPANs based on IEEE 802.15.4 are intended to serve a set of industrial, residential and medical applications with very low power consumption and cost requirement not considered by other WPANs and with relaxed needs for data rate and QoS. The low data rate enables the LR-WPAN to consume very little power.

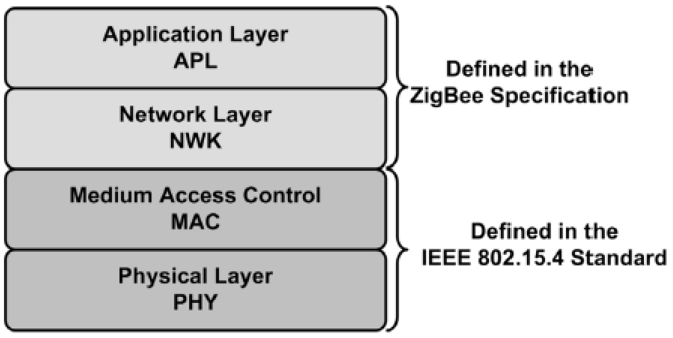


Figure 2.1: ZigBee/IEEE 802.15.4 protocol stack architecture

Using the International Organization for Standardization (ISO) - Open Systems Interconnection (OSI) model, IEEE 802.15.4 standard, defines Physical Layer (PHY) and Medium Access Control (MAC) layer. As example, IEEE 802.15.4 is used as the MAC and PHY layers of Zigbee protocol stack, shown in Fig.2.1. The IEEE 802.15.4 standard stack is not necessarily used in combination with ZigBee, because it is also possible to set up other communication protocols like Internet Protocol

(IP) or fieldbus systems like Wireless Highway Addressable Remote Transducer (HART). Moreover an interesting fact is, that the ZigBee standard is defined and maintained by the ZigBee Alliance, while 802.15.14 is maintained by IEEE.

2.1.1 Device Types

In the IEEE 802.15.4 standard two diferent device types are defined:

- Full-Function Device (FFD) Such a device has the whole set of primitives and functions implemented, which are defined in the standard. A full-function device (FFD) is able to act as coordinator, which provides beacon transmission or to act as simple device. Hence such a device should be able to act as a coordinator, more memory capacity is needed and as a consequence of that, the power consumption and the hardware costs are higher than those of the other type of device.

- Reduced-Function Device (RFD) In contrast to a FFD, the reduced-function device (RFD) only consists of the minimum implementation of the standard. So it is not possible for such a device to act as coordinator. Moreover it is only possible to connect it to one single FFD. Because of the low power consumption and the low hardware requirements, these devices are quite cheaper than FFDs and are used as e.g., small sensors or actuators.

2.1.2 Network Topology

The IEEE 802.15.4 standard offers a few different types of network topologies:

- Star Topology: In such a topology as shown in Fig.2.2, one FFD acts as PAN coordinator and all devices (either a FFD or RFD) in the area of the coordinator can associate with it. If a device wants to communicate with another device in the PAN, it has to send the message to the coordinator, which forwards the message to the destination device.

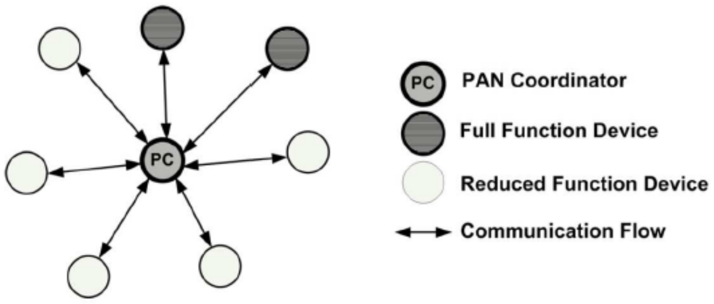


Figure 2.2: Structure of a Star Topology

- Peer-to-Peer Topology In a peer-to-peer network (as shown in Fig.2.3), basically each device can communicate with any other device. If one device wants to communicate with a device, which is not whithin is coverage range, a routing mechanism must be used. This mechanism would forward messages through a number of routing devices to the destination device, whereby routing is performed by FFDs. Such a routing mechanism is not in the scope of the IEEE 802.15.4 standard, but it is implemented by upper layer protocols.

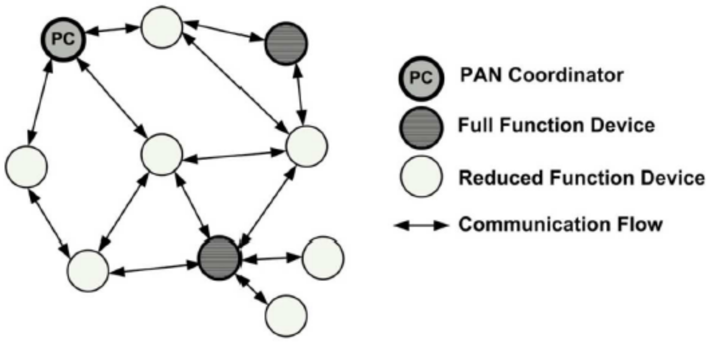


Figure 2.3: Structure of a Peer-to-Peer Topology

2.2 PHY Layer

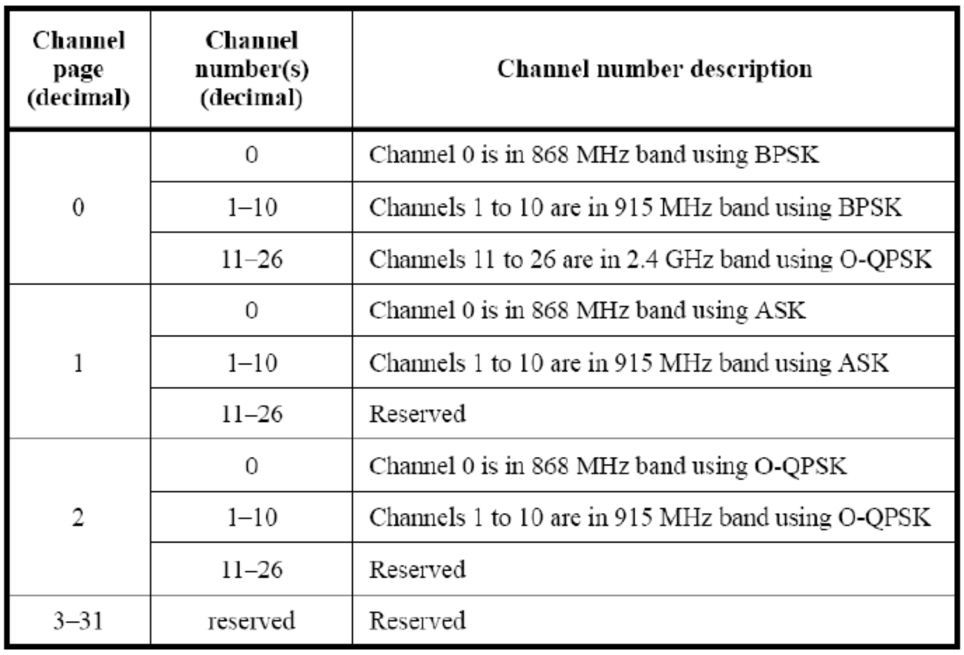
The PHY of IEEE 802.15.4 is used to transmit a basic data frame over the physical medium. Moreover the PHY layer is responsible for modulation and encoding. It also defines frequency band and transmit power.

2.2.1 Frequency Bands

Different frequency bands and modulation methods are defined in this standard which makes it possible to use it in many regions and countries all over the world. The following list gives a short overview about the different frequency bands and the distribution area of them:

- The frequency band at 868,3 MHz is especially defined for Europe.
- The frequency band at 915 MHz is used in the USA.
- The frequency band at 2,46 GHz can be used everywhere around the world.

The most popular frequency band in our region is the 2.4 GHz band. This frequency band is also used by many other wireless technologies and applications. Bluetooth and wireless LAN are the most popular technologies in this frequency band. The use of Bluetooth (IEEE 802.15.1) or Wireless local area network (LAN) (IEEE 802.11) in combination with IEEE 802.15.4 in the 2,4 GHz band causes some problems, like packet errors.



Channel page (decimal)	Channel number(s) (decimal)	Channel number description
0	0	Channel 0 is in 868 MHz band using BPSK
	1–10	Channels 1 to 10 are in 915 MHz band using BPSK
	11–26	Channels 11 to 26 are in 2.4 GHz band using O-QPSK
1	0	Channel 0 is in 868 MHz band using ASK
	1–10	Channels 1 to 10 are in 915 MHz band using ASK
	11–26	Reserved
2	0	Channel 0 is in 868 MHz band using O-QPSK
	1–10	Channels 1 to 10 are in 915 MHz band using O-QPSK
	11–26	Reserved
3–31	reserved	Reserved

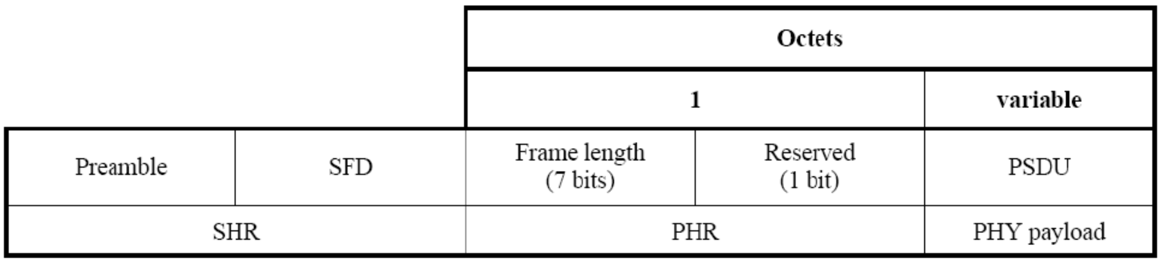
Table 2.1: Table of Channel pages

2.2.2 Transmit Power

The physical layer also specifies the transmit power, whether the maximum physical transmit power depends on the frequency band and the local regulations. The value of the transmit power is not a fixed value, but a variable value, which also can be set by higher layers. The major advantage of an adjustable transmit power is the reduction of power consumption. So if e.g., two communication devices are only a few centimeters away from each other, it is not necessary to transmit a frame with the maximum power. There exists a number of channel pages, which basically indicate the modulation mode. Currently there are only 3 channel pages used, but there is altogether a number of 32 channel pages defined, from which 29 are reserved for future purposes. Basically there exists a number of 27 channels per channel page. Depending on the channel page properties, there are not all combinations of channels with channel pages are valid. A short overview of channels and channel pages is given in Table 2.1.

2.2.3 Physical frame structure

The basic data frame structure, as shown in Fig.2.4, is divided into the synchronisation header (SHR), the PHY header (PHR) and the PHY payload. The SHR contains the preamble field, which is used for symbol synchronization at the receiver and has a variable length depending on the frequency band and the modulation. Moreover the SHR contains the start frame delimiter (SFD) field which has also a variable length and indicates the end of the SHR. The PHR contains only one field which indicates the length of the PHY protocol data unit (PPDU) frame. This field has a length of 8 bits, while one bit is reserved. A valid frame length is between a value of 9 to the maximum defined frame length. The frame length field may contain a special value of 5, which indicates an acknowledgment frame.



		Octets		
		1		variable
Preamble	SFD	Frame length (7 bits)	Reserved (1 bit)	PSDU
SHR		PHR		PHY payload

Figure 2.4: PPDU frame format

2.3 MAC Sublayer

The data link layer (DLL), the second layer of the OSI model is divided into two sublayers, the MAC and the logical link control (LLC). The MAC sublayer is close to the PHY layer and uses services of it. Normally the LLC sublayer sits above the MAC layer and provides services to the NWK layer, but in the IEEE 802.15.4 Standard the LLC sublayer does not exists. Normally the NWK layer of ZigBee sits directly above the MAC layer, but it is also possible to set up an IEEE 802.2 LLC layer above the MAC layer through a service specific convergence sublayer (SSCS).

2.3.1 Introduction

The MAC sublayer of IEEE 802.15.4 provides an interface between the physical layer and the next higher layer and it can be divided into two different parts, as shown in Fig.2.5 The MAC common part sublayer (MCPS) provides the basic data services, like data requests and data indications. The second part of the MAC sublayer is the MAC sublayer management entity (MLME), which is responsible for management services within the MAC sublayer, e.g., association and disassociation.

For a communication in a multi node network a unique address is absolutely necessary. In the IEEE 802.15.4 standard there are two addressing modes specified. Each of them has a unique 64 bit IEEE address which should be assigned to the device during the initialization. This address can be compared to the MAC address in Ethernet (IEEE 802.3). There also exists a 16 bit short

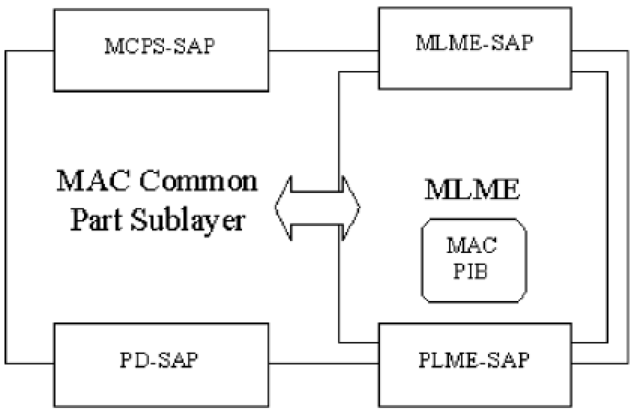


Figure 2.5: MAC sublayer overview

address, which is not assigned in the initialization phase. A short address can be allocated during the association procedure (described in Section 2.3.6). If no short address is defined for a device, its internal address value should be set to 0xffff (broadcast address). Moreover there exists a PAN address, which determines the membership of a node to a PAN (cf. the star-topology in Section 2.1.2). If the source PAN ID of a message is different to the destination PAN ID, the message should be routed through other nodes to the destination PAN, but this routing is out of the scope of the IEEE 802.15.4 standard.

2.3.2 Communication Modes

The MAC sublayer basically provides two options for message transmission. The first and simpler method is the non beacon-enabled mode. In this mode two devices communicate with each other using the medium access protocol called carrier sense multiple access with collision avoidance (CSMA-CA), which is described in Section 2.3.4. Since also time triggered messages should be transmitted, a method called superframe structure in a beacon-enabled mode is introduced, which is described in Section 2.3.3.

2.3.3 Superframe Structure

For critical messages the beacon-enabled mode is used. In this mode a coordinator manages the associated nodes and periodically transmits beacon frames. The time interval between two beacons, called beacon interval, is specied as follows: $aBaseSuperframeDuration * 2^{BO}$ where BO is the Beacon Order. This value is valid between a Range of $0 <= BO < 15$. $BO = 15$ indicates a non-beaconenabled PAN. The aBaseSuperframeDuration is defined as the minimum superframe duration. The time interval between two beacons is divided (shown in Fig.2.6) into two sections, the active period and the inactive period. The length of the active period of the superframe is defined as $aBaseSuperframeDuration * 2^{SO}$ where SO is the Superframe Order. This value is valid in a range of $0 <= SO <= BO < 15$. In the inactive portion it is possible to switch the coordinator and all associated devices in sleep mode to reduce power consumption. The active portion of the superframe is also divided into two sections, the contention access period (CAP) and the contention-free period (CFP). Fig.2.6 shows the portion of the superframe. In the CAP every node, which is associated with the coordinator is allowed to transmit packets using the CSMA-CA algorithm. If a node starts transmitting at the end of the CAP and it cannot finish the transmission before the CAP ends, it has to cancel its transmission. The CFP is used to provide one single node of the PAN exclusive access rights on the channel. Such a time slot is called Guaranteed Time Slot (GTS). In this slot no other device is allowed to transmit a frame. If a node wants to allocate such a GTS it has to request the GTS from the coordinator in a specied time slot. The coordinator checks, if there is a free slot and updates the superframe structure. GTS can be allocated as long as the minimum CAP length is not reached.

If a node receives the beacon it has to check if the requested GTS is allocated in the GTS list. If so, it is allowed to transmit a message within this slot without using CSMA-CA. The whole active period including CAP and CFP is divided into a number of time slots (16 by default).

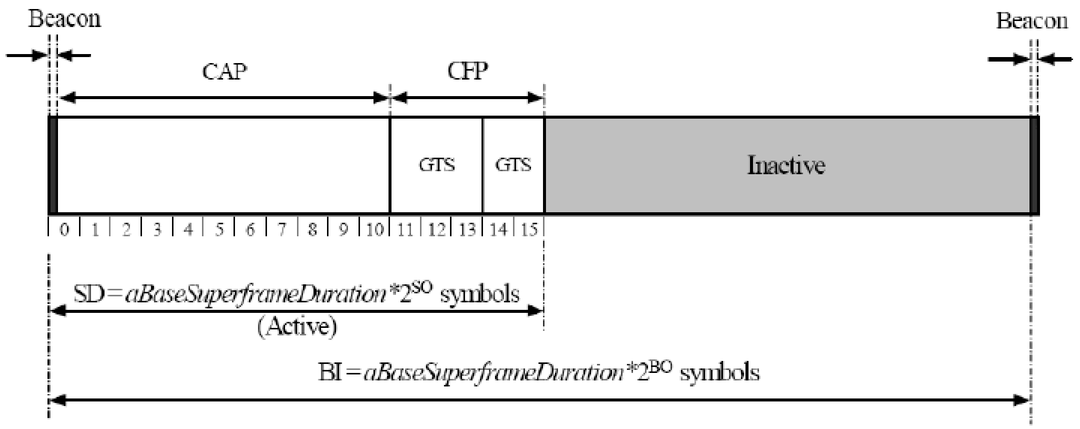

Beacon
Beacon
CAP
CFP
GTS
GTS
Inactive
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
SD=aBaseSuperframeDuration*2^SO symbols
(Active)
BI=aBaseSuperframeDuration*2^BO symbols

Figure 2.6: Superframe timing diagram

2.3.4 CSMA-CA

The CSMA-CA algorithm is basically used to transmit message frames on a shared medium, in which every node is allowed to transmit data. There are existing two versions of the CSMA-CA algorithm, as shown in Fig.2.7. One is used in non beacon-enabled PANs and the other is used in the CAP in a beaconenabled PAN. Before the algorithm is explained, the basic timebase of this algorithm (in general of the whole MAC sublayer) is described.

- A backoff period is the basic time period, which is defined as 20 symbols. The length of a symbol period depends on the frequency band and the modulation method and can be considered as constant. In a beaconenabled PAN the backoff period depends on the internal clock of the coordinator, but in a non beacon-enabled PAN, the backoff period only depends on the internal clock of the MCU.

Basically there are existing three different variables which are necessary to understand the algorithm.

- The beacon exponent (BE) variable determines the maximum time which the algorithm waits before it checks the channel. This delay time is a random number between 0 and $2^{BE}-1$ backoff periods.
- The number of backoffs (NB) variable represents the number of attempts to access the channel. If this number reaches a predefined value the algorithm terminates with a failure.
- The contention window (CW) variable determines the number of backoff periods the channel has to be cleared before transmission of the frame can be started.

CSMA-CA in a non beacon-enabled PAN

In a non beacon-enabled PAN the algorithm acts as follows:

When the algorithm starts, the NB variable is set to zero and the BE variable is set to minBE, then the senderwaits for a random number of backoffs between 0 and $2^{BE} - 1$. After the delay the clear channel assessment (CCA) is performed. This means that the receiver listens to the channel for one backoff period. If the receiver detects no active transmission, the transmitter is enabled and the message can be transmitted. If the receiver detects a transmission on the channel, it increases the NB by one and also increases the BE by one. If the BE reaches the maximum backoff exponent, it does not increase the BE. After this, the algorithm checks, if the number of maximum CSMA-CA backoffs have been reached. If so, the algorithm terminates with a failure. Otherwise the algorithm returns to the begining and waits for a random period, before it performs CCA.

CSMA-CA in a beacon-enabled PAN

In beacon-enabled PANs, the algorithm is quite similar, except a few differences. If the battery life extension is enabled, the BE variable is calculated as follows $BE = min(2; minBE)$ and otherwise the BE variable will be set to the same value as in the non beaconenabled mode ($BE = minBE$). After this, the algorithm checks the boundary of the CAP. If the algorithm can't be terminated before the

CAP ends, the algorithm stops and resumes its execution in the next superframe. After the backoff delay the algorithm performs CCA and if the channel is idle, the CW variable is decremented by one and if the value is not equal to zero, the algorithm performs another CCA. Since the CCA was set to 2 at the beginning of the algorithm, the CCA is performed at least twice before the message can be transmitted. If there is any track on the channel, the CW count will be reset to 2, the BE will be increased by one and the NB will also be increased by one. If the NB reaches the number of maximum CSMA backoffs, the algorithm terminates with a failure. If the NB variable has not reached its maximum value of two, a new CCA will be started after the backoff delay.

2.3.5 Starting a PAN

Before a device can associate to a PAN, it first has to know about reachable PANs and their link quality, which can be determined by the scan procedure. If a node is not associated with a PAN, it has the possibility to check if anything is transmitted on the each channel. So it is possible to detect the signal energy and link quality (Energy detection scan) of each channel. Moreover it is possible to make active or passive scans.

During a passive scan the device turns on its receiver and listens to the channel for a defined period of time. After that it switches to the next specified channel and performs the scan again. If all requested channels were scanned, the MAC sublayer indicates the next higher layer and returns all discovered PANs. The active scan is quite similar to the passive scan. It additionally transmits a beacon request frame and waits a defined period of time for a beacon frame of the coordinator, hence the active scan is used for non beacon-enabled PANs. If a PAN coordinator in a non beacon-enabled PAN receives a beacon request, it has to answer with a single beacon frame.

If on the other hand a PAN coordinator wants to start a PAN, it first has to reset its internal state. After that the device updates its internal variables with the values provided by the next higher layer (e.g., PAN ID, *shortAddress*). After that the device starts operating as PAN coordinator.

2.3.6 Association and Disassociation

- If a device discovered a PAN coordinator through a previous scan, it can start with the association procedure. Before an association request is sent, the internal state of the device must be updated. So for example the address of the coordinator or the PAN ID must be updated. After that an association request command can be sent. If the according coordinator receives this message, it indicates go to the next higher layer. The higher layer decides whether the association is permitted or not. Depending on this decision the MAC sublayer sends a response to the device which requested the association. If the device receives this response, it indicates the next higher layer. If the association was successful, it stores the received short address into its internal data structure.

- If a device wants to disassociate from a PAN the next higher layer of the device sends a disassociation notication frame to the MAC. The MAC sublayer generates a disassociation notification command and transmits it to the coordinator. After receiving an acknowledgment frame from the coordinator, the MAC indicates the next higher layer the disassociation. If the device does not receive an acknowledgment frame, it indicates the disassociation anyway.

If the coordinator wants a device to disassociate it sends a disassociation request to its MAC sublayer. The MAC sends a disassociation notication frame to the device which should be disassociated. This device should send an acknowledgment frame back to the coordinator and should indicate its next higher layer the disassociation. If the coordinator does not receive the acknowledgment frame or the timeout expires, the MAC of the coordinator indicates the next higher layer the disassociation anyway.

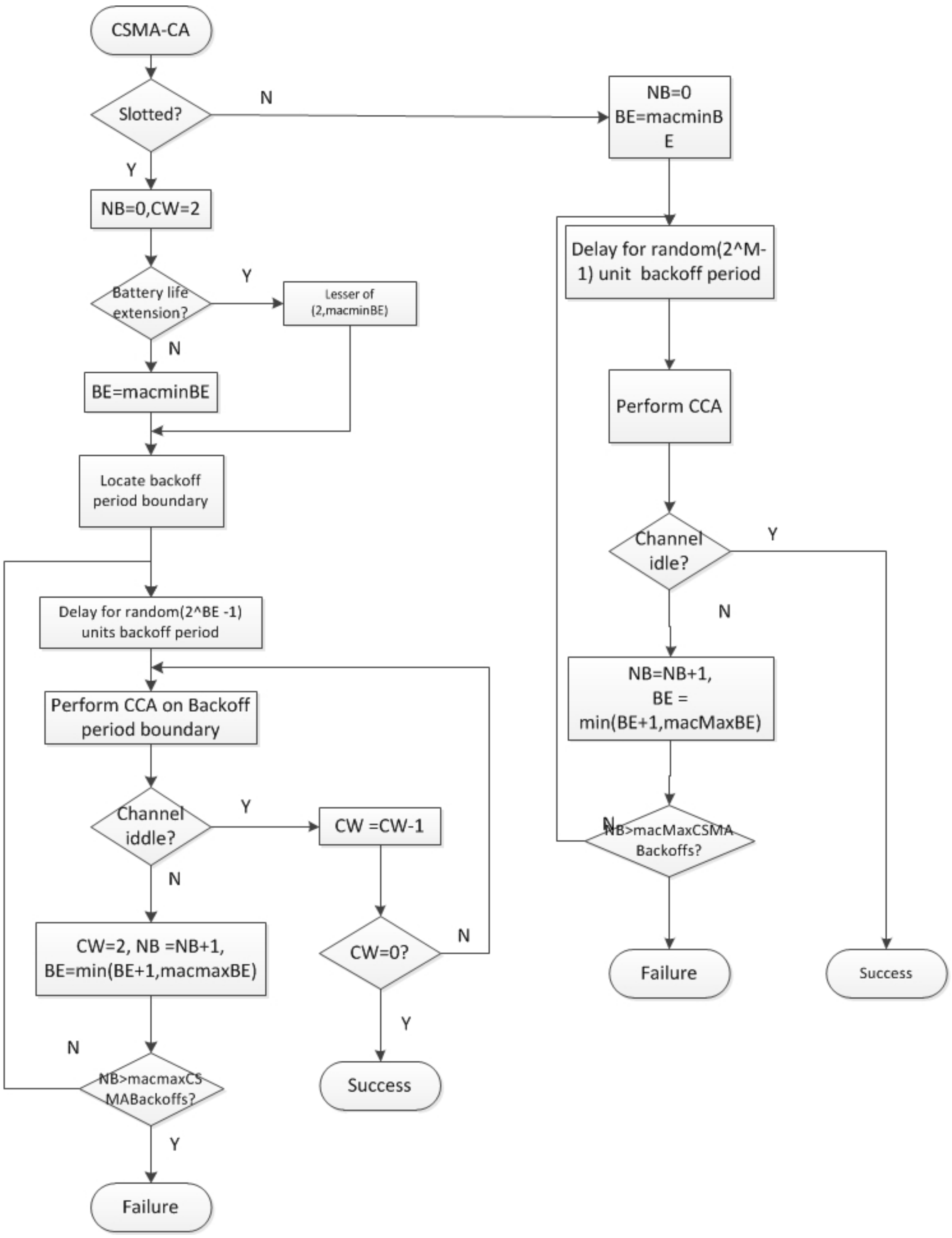


Figure 2.7: CSMA-CA algorithm

Chapter 3

Design approach

This chapter introduces the design decisions and assumptions according to the scenario of the project.

3.1 Specifications

As a minimum development, a small node opportunistic network, composed of a mobile coordinator capable to report data to a laptop and wireless sensor devices capable of measuring and storing temperature and light intensity. The sensor devices must take at least one sample each 10 minutes to obtain worth valuable results and batteries must last as long as possible since sensor-devices will be on their own and no human intervention is required. An optimized communication protocol must be developed in order to succeed in almost every communication attempt despite the randomness of the chances. Thus, the minimum number of medium accesses must be guaranteed to transmit all the info, sending all the stored data as soon as there is a chance of it and not waiting for another future chance to transfer old data that was not able to transmit in a previous connection. All these functionalities must be provided without network infrastructure and devices must be cheap since most of them will be lost when their lifetime end up. All nodes will be based on the CC2430 chipset due to its wide range of power saving features and peripherals. Moreover, the sensing part of the device must be an optimized hardware, to provide power saving and avoid wasting energy resource. Eventually, software architecture will be based on the TI-MAC protocol , built up by Texas Instruments to support their IEEE 802.15.4 hardware.

3.2 Overall design decisions according to specifications

The first settings that must be considered are very related to the future software and hardware development and the IEEE 802.15.4 background given in previous chapter. Eventually, having in mind how the opportunistic script would be, the most important features settled in advance for the OppNet development were the following ones.

Beacon enabled versus non-beacon enabled networking

One of the most important decisions of this project was to use the beacon enabled or not. Beacon enabled networks turns to be more restrictive and push the developer to be careful when data transactions. On the other hand, it was a very helpful tool to commit our purpose of building an OppNet, the fact of sending beacons from the coordinator and leaving devices with the only work of enabling their radio receiver to listen to them when the coordinator is supposed to be around. The flow chart of the protocol was simplified: when a beacon is catched by the device's receiver it has a determined number of superframe periods to exchange the data stored previously. Aversely, if beacons were not used, each device would need to perform active scans, broadcasting beacon requests and wait for a response of the coordinator. This is much inefficient for this scenario since the coordinator will be less energy consumption sensitive than devices.

Direct data transfers in beacon enabled networks

Direct data transfers occurs from the device to the coordinator according to CSMA-CA, trying to send data within a timeslot between each beacon period and, if it does not succeed, waiting for the defined backoff time to retry it. The success of the data exchange comes from the demanded ACK from the coordinator. The scenario we are dealing with, kindly, will not present a crowd of devices attempting to exchange data to the coordinator together. This means that within the 16 timeslots of the superframe structure there is a high probability of getting one. Since, the radiolink is expected to be established for more than one superframe duration the probability increases.

Power management

Power management will be enabled for the end device to commit the objective of saving battery. Coordinator will not take advantage of this feature since it is required to be beaconing to discover the end-devices. Another cause is that it will be connected to a laptop and there will be no battery dependency. The power manager is implemented by the Operating System Abstraction Layer (OSAL) layer of the TIMAC and turns the device into Power Mode 2 (see section 4.1.2) when no task is scheduled. This is possible searching for the next timer expiration and bringing the device to Power Mode 0 when the timer interrupt occurs.

3.3 Overview and assumptions

This part provides a detailed description of the scenario in which our Oppnet will operate. Thus, in next chapter code settings and flow charts will be understandable . A simple vision of the scenario in motion is illustrated in Fig.3.1 . As we can see end-device is regularly measuring and performs a beacon search when it needs to transfer the data, since the coordinator is regularly beaconing too, when the beacon is found the data is transfered and each one continues with its task.

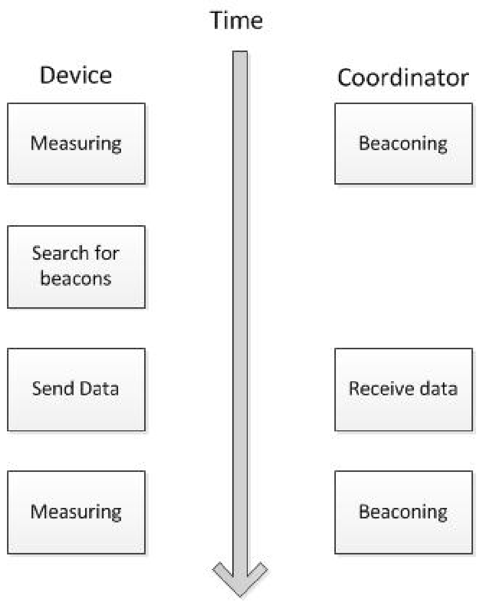


Figure 3.1: Overview of device and coordinator operations

3.3.1 Device range and radiolink specifications

As we introduced in the first chapter, devices will be placed in different locations around a regular bus line route trying to establish radiolink with coordinator located in the bus when both are in range. The scenario is illustrated in Fig. 3.2. Assuming that the bus will travel at about 40km/h (aprox. 11m/s) and the range of the devices will be approximately 15m the maximum time of sight of the coordinator may be between 2-3 seconds (computing the predictions made before the exact value would be 2,7 seconds).

Summary:

- Bus speed: 40 km/h-11m/s
- Device range: 10m
- Time in range: 2,7 s

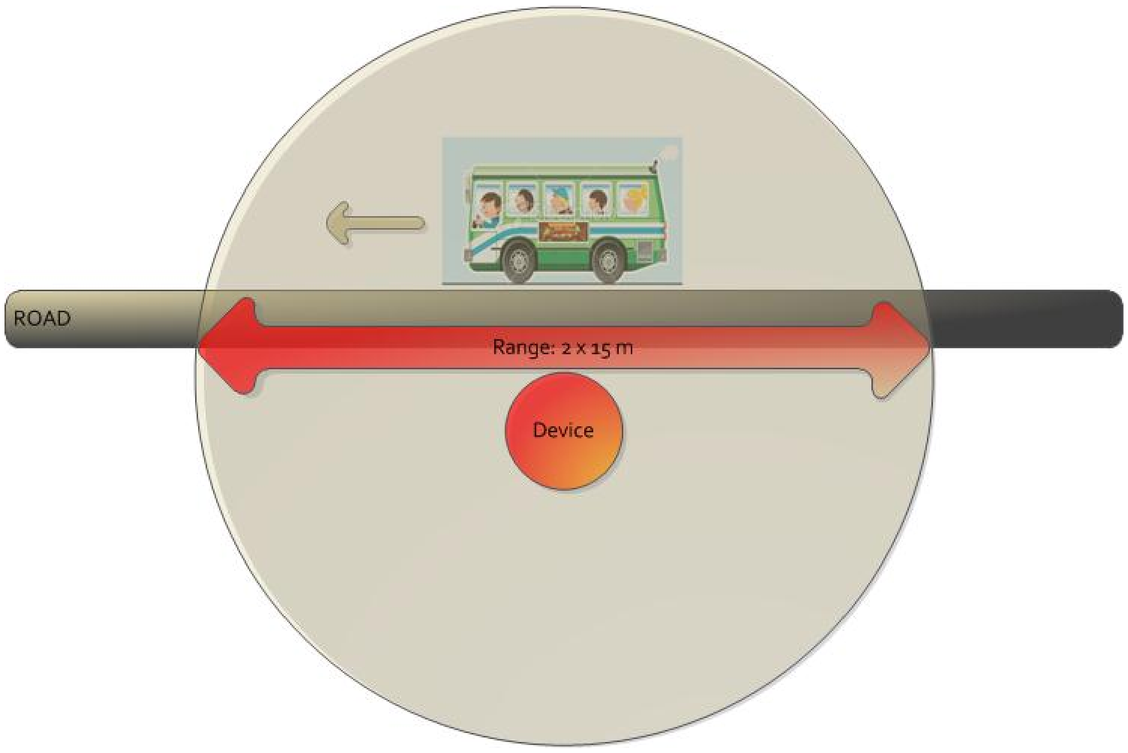


Figure 3.2: Final scenario

3.3.2 Beacon interval

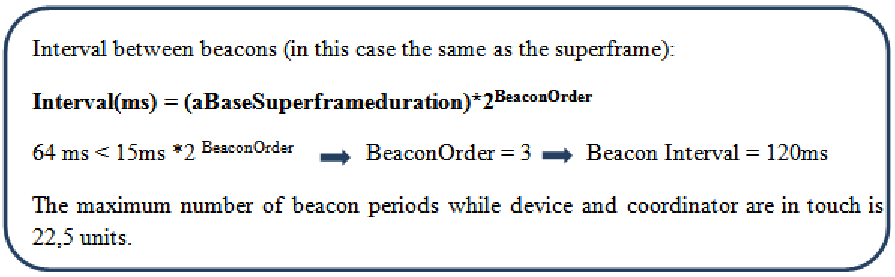
The superframe structure must be adapted to the previous assumption so that data exchanges are finished within each beacon period.

- PHY frame max length: 133 bytes
- Bit rate: 250 kbps
- Frame sending time: 4ms

Each of the 16 time slots must be at least 4,256 ms (rounded to 4 ms) long to ensure a correct data transaction, it represents a superframe period of at least 64ms. The length in time of the superframe is defined by the superframeOrder which must be the same or less than the beacon order. The minimum duration aBaseSuperframeduration is 15 ms and the order must be between 0 and 14.

3.3.3 Sensor sampling frequency

The selection of samplig frequency is based on the required accuracy. For instance, to achieve our goal in the environmental study we have decided to set a period of 10 minutes.


Interval between beacons (in this case the same as the superframe):

Interval(ms) = (aBaseSuperframeduration)*$2^{\text{BeaconOrder}}$

64 ms < 15ms *$2^{\text{BeaconOrder}}$ ➡ BeaconOrder = 3 ➡ Beacon Interval = 120ms

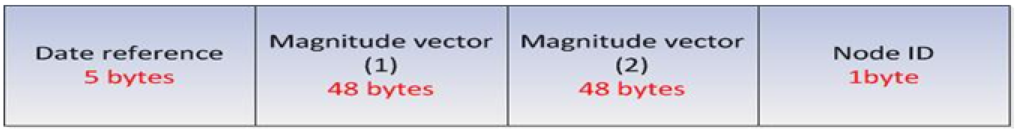
The maximum number of beacon periods while device and coordinator are in touch is 22,5 units.

Figure 3.3: Beacon configuration

$$1440\,\frac{minutes}{day}\cdot 1\,\frac{sample}{10minutes} = 144\,\frac{samples}{day} \tag{3.1}$$

3.3.4 Definition of the data structure

MAC frame definition leaves 102 bytes available to the user's data. The data stored by the sensors will be sent following the structure illustrated on Figure 3.4, which will be contained in the frame payload.



Date reference 5 bytes	Magnitude vector (1) 48 bytes	Magnitude vector (2) 48 bytes	Node ID 1byte

Figure 3.4: Data frame structure

The structure is composed by the following fields:

- Sequence first measurement date: minute, hour, day, month, year. (5 Bytes)
- Sequence for first magnitude: 48 measurements 8 bits codification. (48 Bytes)
- Sequence for second magnitude: 48 measurements 8 bits codification. (48 Bytes)
- Node Id, 1byte integer to help locating a measurement packet. (1 Byte)

3.3.5 Data exchange frequency

The protocol has been developed assuming 3 different possibilities:

- One data exchange per day: 144 measurements x 2 magnitudes = 288bytes, 3 data frames to transfer.

- Two data exchanges per day: 72 measurements x 2 magnitudes = 144bytes, 2 data frames to transfer.
- Three data exchanges per day: 36 measurements x 2 magnitudes = 72bytes 1 data frames to transfer.

As we increment the number of possible data exchanges, memory usage is relieved and data transactions are lighter, also transaction success probability is higher.

Chapter 4

Hardware platform

This chapter presents all the hardaware platform developed to create the wireless network with its sensing part as well. The main features of the CC2430 chipset are introduced and subsequently the custom sensor, built according to the previous design decisions.

4.1 Introducing CC2430 as hardware core

CC2430 is a SOC built up to work with IEEE 802.15.4 radio standard based on a high performance 8051 microcontroller and low energy consumption configuration, providing a wide range of features.

4.1.1 General features

The following features are extracted from the CC2430 datasheet provided by Texas Instruments[1]:

Microcontroller

- High-Performance and Low-Power 8051-Compatible Microcontroller.
- Optimized 8051 core, which typically gives 8x the performance of a standard 8051.
- Dual data pointers.
- In-circuit interactive debugging is supported for the IAR Embedded Workbench through a simple two-wire serial interface.

Memory

- Up to 128 kB Non-volatile Program Memory and 2 x 4 kB Data Memory.

[1]This information can be found in document http://focus.ti.com/lit/ds/symlink/cc2430.pdf

- 32/64/128 kB of non-volatile flash memory in-system programmable through a simple two-wire interface or by the 8051 core.
- Worst-case flash memory endurance: 1000 write/erase cycles.
- Programmable read and write lock of portions of Flash memory for software security.
- 4096 bytes of internal SRAM with data retention in all power modes.
- Additional 4096 bytes of internal SRAM with data retention in power modes 0 and 1.

Peripherals

- DMA Controller
- Power On Reset/Brown-Out Detection
- Eight channel, 8-14 bit ADC (0-3V input)
- Programmable watchdog timer
- Real time clock with 32.768 kHz crystal oscillator
- Four timers: one general 16-bit timer, two general 8-bit timers, one MAC timer
- Two programmable USARTs for master/slave SPI or UART operation
- 21 configurable general-purpose digital I/O-pins

Low power management

- Four flexible power modes for reduced power consumption
- System can wake up on external interrupt or real-time counter event

802.15.4 MAC hardware support

- Automatic preamble generator
- Synchronization word insertion/detection
- CRC-16 computation and checking over the MAC payload
- Clear Channel Assessment
- Energy detection / digital RSSI

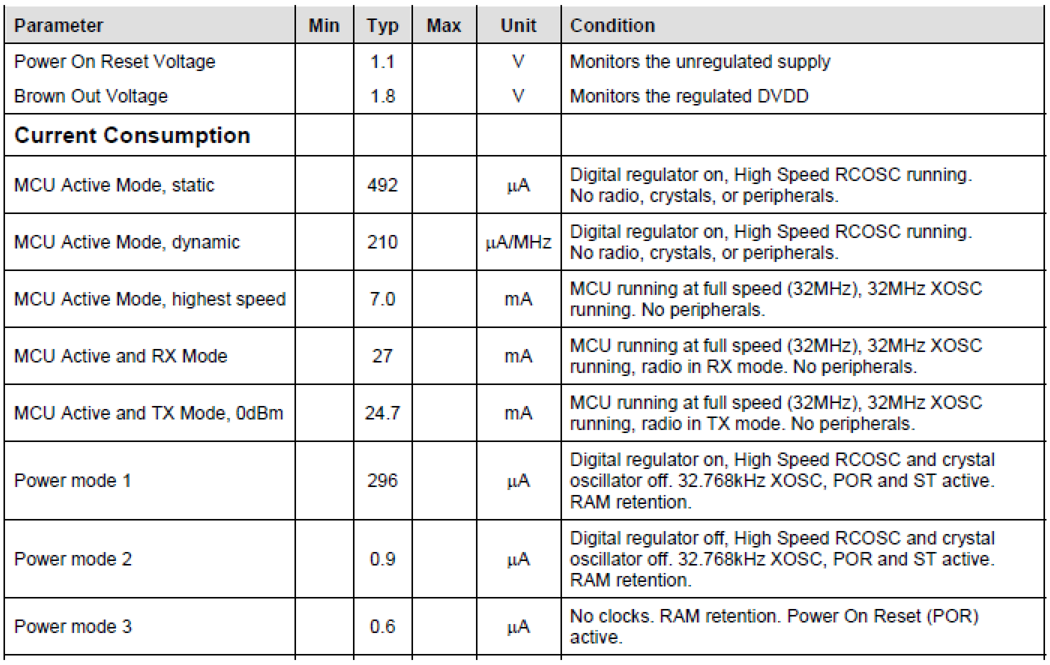
- Link Quality Indication
- CSMA/CA Coprocessor

Integrated 2.4GHz DSSS Digital Radio

- 2.4 GHz IEEE 802.15.4 compliant RF transceiver (based on CC2420 radiocore).
- 250 kbps data rate, 2 MChip/s chip rate
- Typical Rx sensitivity up to -90dbm.
- Tx power up to 0dbm.

4.1.2 Electrical especifications : power modes

Table 4.1 shows the current consumption in the different power modes. When MCU active, Power Mode 0 is running in different hardware configuration which helps to save energy when is needed. The most used power modes will be power mode 0 as the normal usage of the chipset and power mode 2 when the sleep routine is performed since in the rest of low consumption power modes waking up is not possible if an external interrupt is not performed. In this project it is required the use of timer expirations for the waking up.



Parameter	Min	Typ	Max	Unit	Condition
Power On Reset Voltage		1.1		V	Monitors the unregulated supply
Brown Out Voltage		1.8		V	Monitors the regulated DVDD
Current Consumption					
MCU Active Mode, static		492		µA	Digital regulator on, High Speed RCOSC running. No radio, crystals, or peripherals.
MCU Active Mode, dynamic		210		µA/MHz	Digital regulator on, High Speed RCOSC running. No radio, crystals, or peripherals.
MCU Active Mode, highest speed		7.0		mA	MCU running at full speed (32MHz), 32MHz XOSC running. No peripherals.
MCU Active and RX Mode		27		mA	MCU running at full speed (32MHz), 32MHz XOSC running, radio in RX mode. No peripherals.
MCU Active and TX Mode, 0dBm		24.7		mA	MCU running at full speed (32MHz), 32MHz XOSC running, radio in TX mode. No peripherals.
Power mode 1		296		µA	Digital regulator on, High Speed RCOSC and crystal oscillator off. 32.768kHz XOSC, POR and ST active. RAM retention.
Power mode 2		0.9		µA	Digital regulator off, High Speed RCOSC and crystal oscillator off. 32.768kHz XOSC, POR and ST active. RAM retention.
Power mode 3		0.6		µA	No clocks. RAM retention. Power On Reset (POR) active.

Table 4.1: CC2430 electrical specifications

4.2 Coordinator hardware platform

Coordinator node consists of a SmartRF04EB with an associated CC2430EM connected to a laptop via RS232 interface to swap data and USB as power supplier.

4.2.1 Introducing SmartRF04EB and CC2430EM

The SmartRF04EB board shown in Fig. 4.1 provides a gate to interact between the laptop and the CC2430EM. To do so, it is only needed to mount the CC2430 on the board and connect the USB wire to provide power supply. Via USB, it is also possible to debug the code step by step which is a useful tool to develop any application.

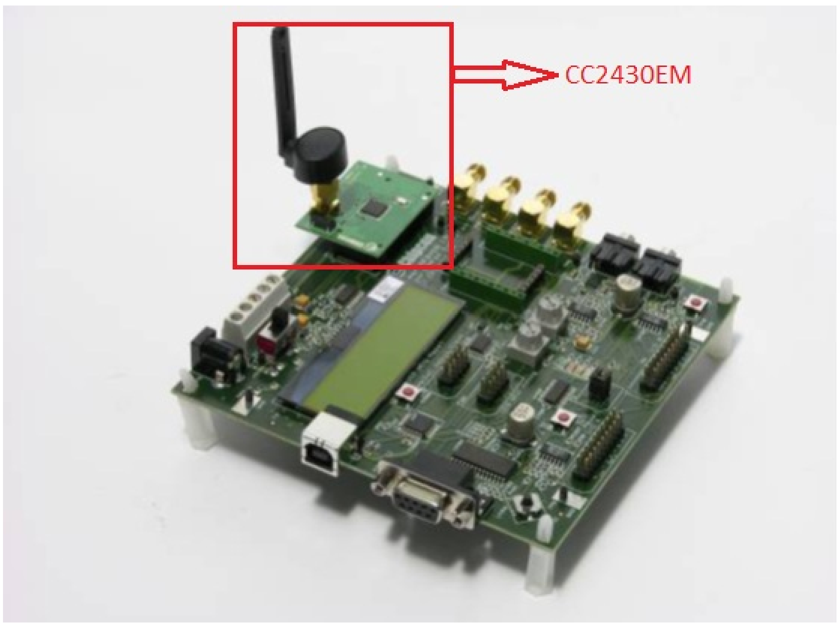


Figure 4.1: SmartRF04EB board

The RS-232 interface allow to swap data between a PC and the SmartRF04EB board with the UART service, it also can be used by custom applications for communication with other devices. The RS-232 interface utilizes a voltage translation device so that the RS-232 port is compatible with bipolar RS-232 levels. Coordinator will exchange the incoming data from the devices with the PC via this interface as shown in Fig. 4.2.

As is mentioned on the release note DN112 from Texas Instrument regarding TI-MAC, the necessary RS-232 wire must use a direct configuration Input-Output.

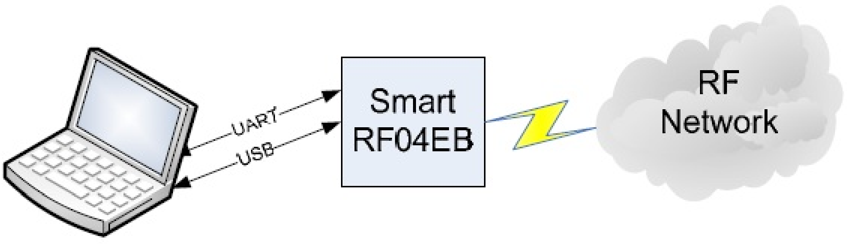


Figure 4.2: RS-232 connection

4.3 End-Device hardware platform

The end device module consists of a CC2430 chip based board and a sensor module that brings temperature and light intensity samples. Fig.4.3 shows the physical implementation of the end-device module.

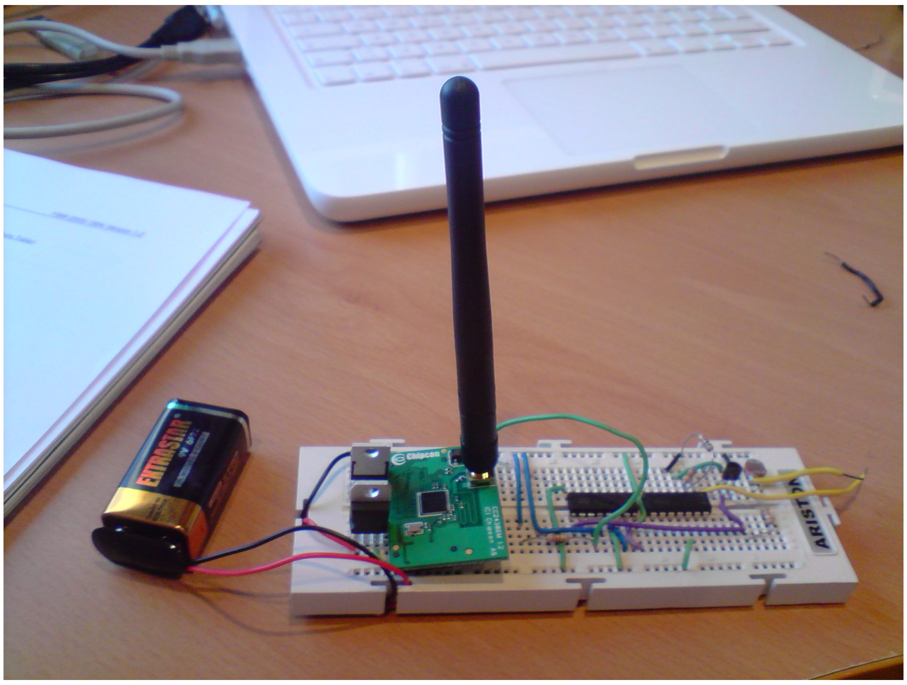


Figure 4.3: End-device module

Regarding the intelligence of the module it is built with a particular variant of the CC2430EM designed by Telematics Engineering department of the EPSC-UPC(onwards CC2430N) which can operate on its own with a 3V battery without being mounted on the SmartRF04EB board. Finally there is the part of the sensor based on an LM35 and a Light Dependent Resistance (LDR) which, mainly, has two outputs corresponding each one to temperature or light intensity either and one input that switches the sensor on/off. The basic hardware feature of the CC2430N chipset employed is the ADC. Two of the pins of I/O ports are configured as input while using the ADC with an 8 bits resolution to convert the output voltage of the sensors on a digital word and other output port is configured to bring a pulse that switches on the sensor part allowing to pick the measure. About the power supply, any battery providing more than 5 V is accepted because of the usage of voltage regulators, which provides the different VDD required within the module.

4.3.1 Overview

This subsection describes the schematic of the full end-device module Fig. 4.4 illustrates the different parts of the system.

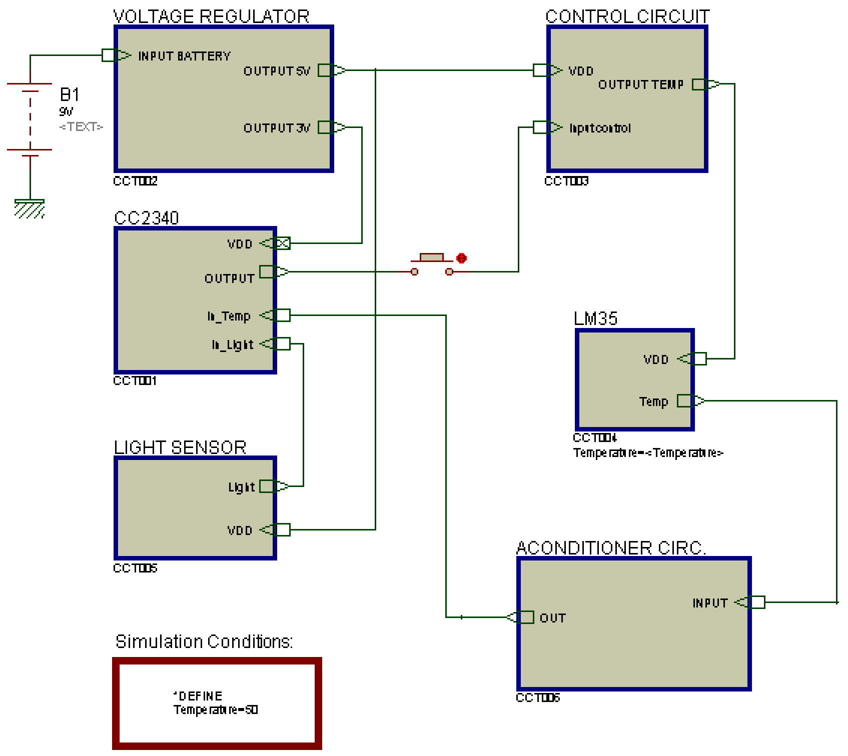


Figure 4.4: End-device schematic

- Power supply (Voltage regulator + batttery)
- Control circuit, acting as switch
- CC2430N chipset
- Light and Temperature(LM35) sensors

- LM35 output aconditioner.

All the devides employed in this module have been selected having in mind the requirement of low cost and low energy consumption to achieve a very cheap and power saving system.

4.3.2 Power supply sub-module

The main purpose of this sub-module, see Fig.4.5, is to provide the energy requirements of the sensing part. The CC2430N is not able to provide the minimum 5V required by the LM35 to its properly operation, since the maximum output voltage available is about 3'3V. This module consists of two voltage regulators connected to a battery that supplies the required energy to the whole module. Instead of using a 3V separate battery for the CC2430N chipset, a 3V regulator converts any input voltage within 5V to 30V to properly feed the chipset VDD intake and another 5V regulator provides the voltage required by sensing circuit.

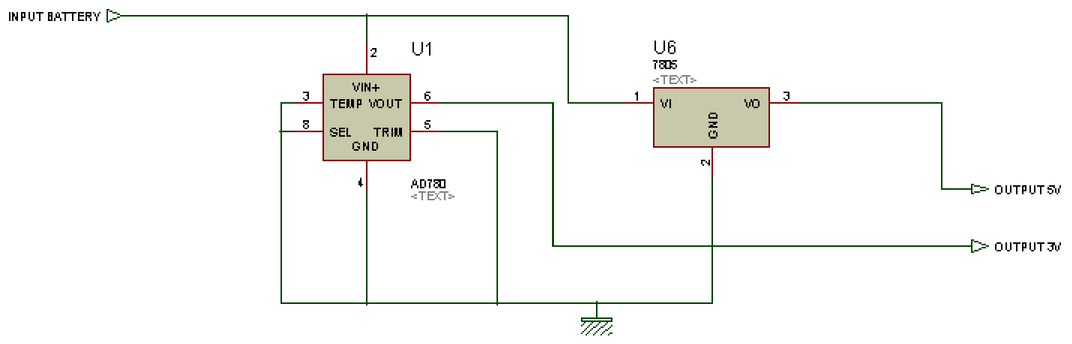


Figure 4.5: Voltage regulator schematic design

4.3.3 Control circuit

The commander of this control circuit is an output port of the CC2430N which is directly settled to high value from the code connected to the input control of the control circuit, see Fig.4.6. The control circuit is based on a JK Flip Flop in Toggle configuration, see state table4.2 . Thus, a pulse coming from the CC2430N sets the output of the control circuit high feeding the sensor sub-module. When another pulse is sent the control circuit puts its output to 0 level cutting off the current source to the sensors.

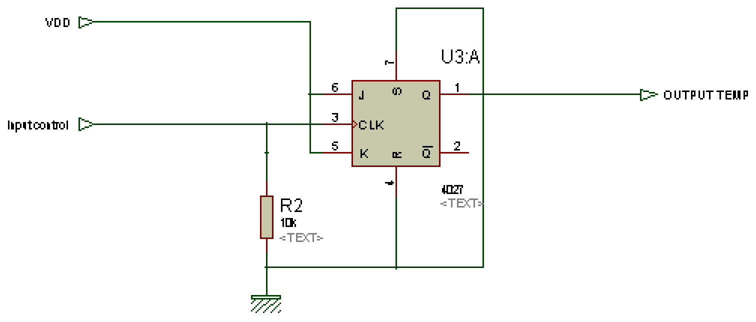


Figure 4.6: Control circuit schematic design

T	Q	Qnext	Comment
0	0	0	Hold state
0	1	1	Hold state
1	0	1	toggle
1	1	0	toggle

Table 4.2: Toggle flip flop states

4.3.4 Temperature and light intensity sensor sub- modules

As it was mentioned before, the sensor is split in two different sensors providing temperature and light intensity sampling. There is still another hardware part intended to save battery when the sensor is not required to be powered on.

Temperature sensor

The temperature is based on an LM35 which is a precision integrated-circuit temperature sensor, whose output voltage is linearly proportional to the Celsius (Centigrade) temperature, no calibration or trimming is needed. It can be used with single power supplies, or with plus and minus supplies. As it draws only 60 µA from its supply, it has very low self-heating, less than 0.1°C in still air. The LM35 is rated to operate over a −55° to +150°C temperature range if the minus supply is provided. In terms of accuracy, the manufacturer guarantees at least 0.5°C at 25ºC. The configuration of the LM35 and the output is shown in Fig. 4.7.

As an example, for a given temperature of 26º the output of the LM35 is:

$$0mV + 10.0mV/^{\circ}C * 26^{\circ}C + -0.5^{\circ}C = 260mV \tag{4.1}$$

In our design we assume extreme conditions of a maximum of 50ºC and a minimum of 0ºC. In order to adapt the LM35 output to the 0-3V dynamic range of the ADC a non-inversor monopolar

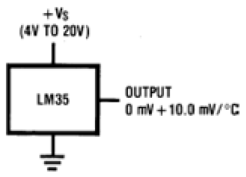


Figure 4.7: LM35 schematic

amplifier is used(LM324N), the ideal required gain is 1.1 since the resolution of the ADC is 11mV, in order to prevent possible errors and bearing in mind that only about 50 values will be expectable the gain proposed is 1.45 according to real resitance values:

Thus, the gain is linear and can be solved with the equation 4.2:

$$V_{out} = G \cdot V_{in}(LM35) \tag{4.2}$$

Configuration of the non-inversor amplifier,or so called aconditioner circuit, see Fig.4.8 :

$$Av = \frac{V_{out}}{V_{in}} = 1 + \frac{R_2}{R_1} = 1,45 \Rightarrow R2 = 1K\Omega,\ R1 = 2,2K\Omega \tag{4.3}$$

The maximum expected value at the ADC input would be:

$$0mV + 10.0mV/{}^{\text{o}}C * 50^{\text{o}}C + -0.5^{\text{o}}C = 500mV$$

$$V_{out} = 1.45 \cdot 500mV = 745mV$$

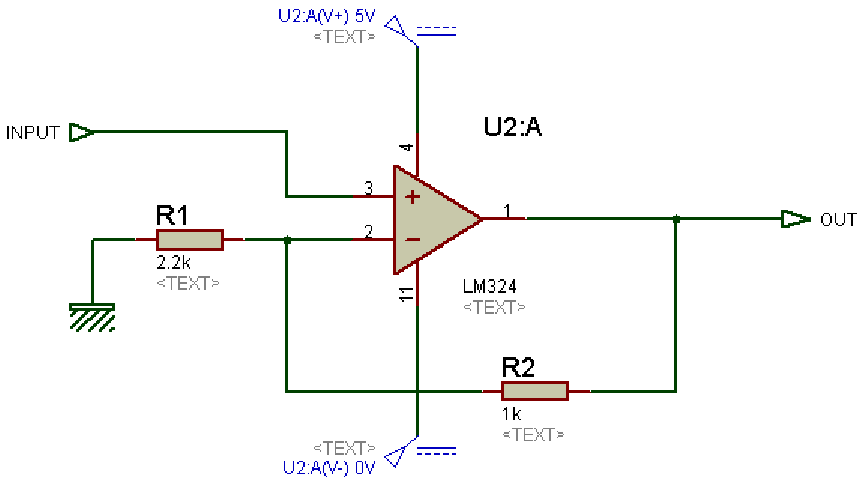


Figure 4.8: Aconditioner circuit schematic design

Light intensity sensor

The light intensity sensor is based on an LDR, which increases its resistance as the intensity of light decreases. A representation of the LDR behaviour can be seen in Fig. 4.9. The source voltage is taken directly from the CC2430N output pin which provides about 3V (it may vary from 2,8V to 3,00V measured).

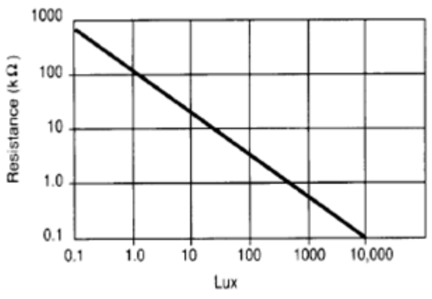


Figure 4.9: LDR Resistance vs Lux

As a reference, about 100 Lux value is found in a clear indoor room and about 10.000 Lux is a bright outdoor light value. This kind of sensor needs a previous calibration in order to correctly measure its ouput value and translate it into a readable value, the 3,3 kΩ resistance is a recommended value. To do so, the output value of the LDR circuit is the result of the equation Eq. 4.4. The circuit schematic design of the LDR sensor is shown in Fig. 4.10.

$$Vo = 3 \cdot \frac{R_{LDR}}{(R_{LDR} + 3.3)} \tag{4.4}$$

Example: For a clear indoor room (100Lux):

$$Vo = 3 \cdot \frac{5K\Omega}{8,3K\Omega} = 1,80V$$

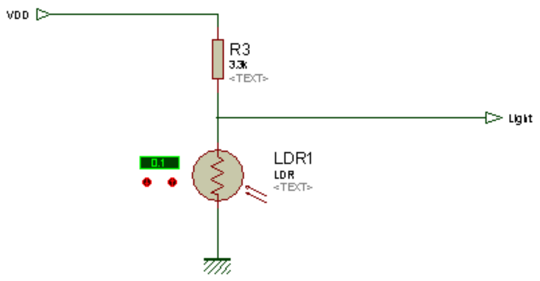


Figure 4.10: LDR schematic design

Chapter 5

Opportunistic protocol: Software development

This chapter is intended to explain the software developed to build an Opportunistic network with TIMAC protocol stack. The code presented in this chpater has been created with three API's of the TIMAC: OSAL , Hardware abstraction layer(HAL) and the MAC API definition. Here we will bring the basic functionalities of each API and the chipset features we can control with each one.

5.1 Description of TIMAC API's

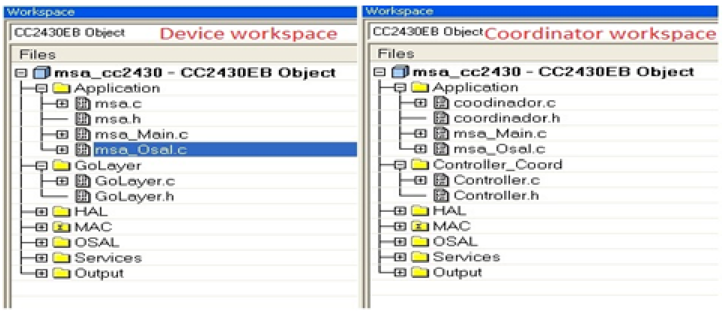
TIMAC is the acronym of Texas Instruments Medium Access Control. This software stack might be used with Texas Instruments hardware that implements IEEE802.15.4. The stack is composed by different integrated layers which allow to control different actions of the device, in this project we have dealt with the following ones:

1. MAC: this layer is totally blocked to the developer, the source code is protected so only the headers are available. It is possible to use them to control the medium access but it is not allowed to modify any of its functions.
2. HAL: the hardware controller layer which allows to control features such as: LED, LCD, ADC, UART, Timers and energy modes, etc.
3. OSAL: this upper layer is able to organize the code in different tasks and events that are triggered by lower layers call-backs and let us to set priority to the different tasks allowing developers to build code that may run simultaneously. Also it controls memory allocation, timers, power management and task synchronization.

5.2 Opportunistic Code project

5.2.1 Workspace architecture

The code project is split in two different workspaces, one for the coordinator and another one for the devices. Each workspace is divided as well in different modules in order to organize the code and make it scalable.



```
Workspace
CC2430EB Object    Device workspace
Files
⊟ msa_cc2430 - CC2430EB Object
  ⊟ Application
    ⊞ msa.c
      msa.h
    ⊞ msa_Main.c
    ⊞ msa_Osal.c
  ⊟ GoLayer
    ⊞ GoLayer.c
      GoLayer.h
  ⊞ HAL
  ⊞ MAC
  ⊞ OSAL
  ⊞ Services
  ⊞ Output
```

```
Workspace
CC2430EB Object    Coordinator workspace
Files
⊟ msa_cc2430 - CC2430EB Object
  ⊟ Application
    ⊞ coodinador.c
      coordinador.h
    ⊞ msa_Main.c
    ⊞ msa_Osal.c
  ⊟ Controller_Coord
    ⊞ Controller.c
      Controller.h
  ⊞ HAL
  ⊞ MAC
  ⊞ OSAL
  ⊞ Services
  ⊞ Output
```

Figure 5.1: Device and coordinator workspaces

msa_Main.c and msa_Osal.c contains the initialization processes required and also initialises the different tasks that the code will run. In coordinador_init(taskId) and msa_init(taskID) it is mandatory to call either MAC_InitCoord() followed by Mac_InitBeaconCoord() in case of coordinator or Mac_InitDevice() followed by Mac_InitBeaconDevice() in case of devices. These functions initializes the MAC to allow to associate with and track a beacon-enabled network. They must be called before any other function in the data or management API is called. After that, the msa or coordinator inits are called, later on they will be explained as part of the code project of each workspace. In the msa and coordinador event processors are declared, when an event is identified and processed, controller and go_layer provides implementation to the functions handled in each process. Indeed, the next part of the chapter is about to describe these functions following each flow chart.

5.2.2 Coordinator code project

5.2.2.1 Overview

The coordinator flow chart,Fig.5.2, starts with an init process, mainly initilising hardware and invoking mandatory routines by the APIs.

The main task of the coordinator is to generate beacons and respond to the association request that will follow to the reception of a beacon from the device. Once this is done, the coordinator only waits for device to send data and acts as gateway with a laptop via the UART.

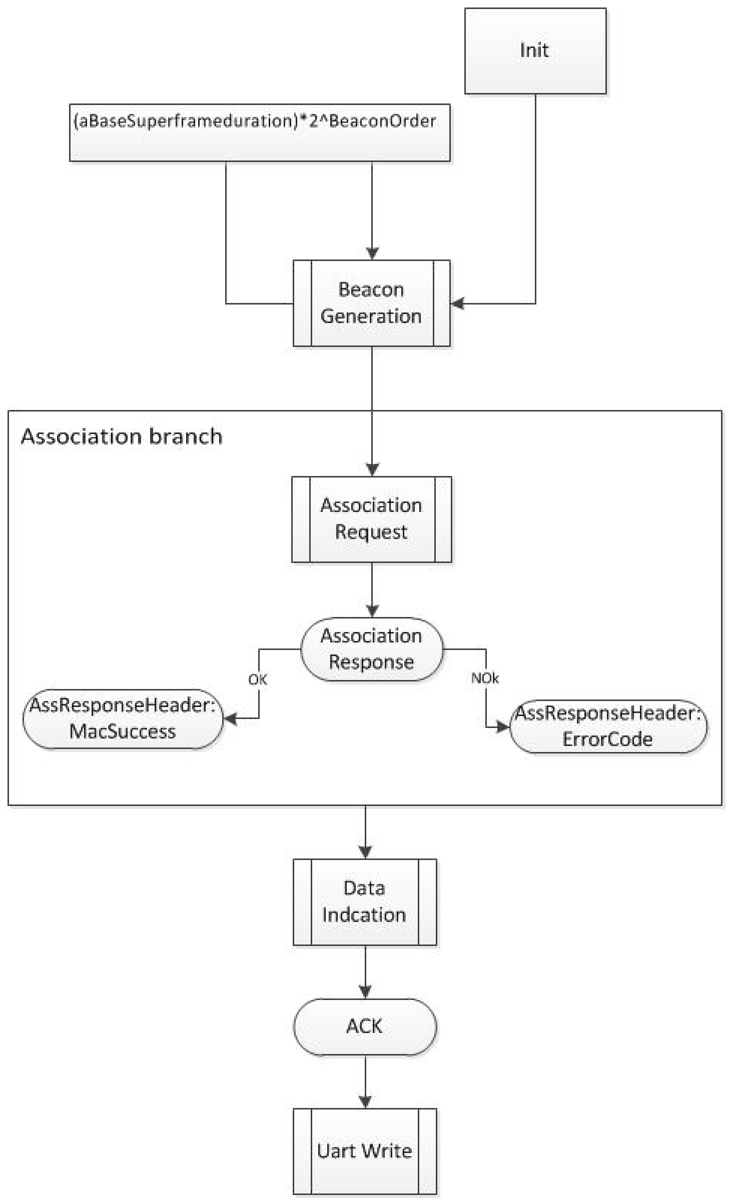


Figure 5.2: Coordinator flow chart

5.2.2.2 Init process and beacon generation

As described before, firstly, all system initialisation functions are invoked as a mandatory requirement. In this case, the UART must be configured to set properly values for further communication via serial port. The following values must be set in advanced in the msa_Main as well as stop bits are set to 1 and parity to none in the main board configuration:

```
HalUARTInit();
halUARTCfg_t uartConfig;
uartConfig.baudRate  = HAL_UART_BR_38400;
uartConfig.flowControl    = TRUE;
uartConfig.tx.maxBufSize   = 64;
HalUARTOpen(0, &uartConfig);
```

The UART service has two ports available, here port 0 is defined and its pins must be configured within the HalUartOpen routine: pin 2 as Rx and pin 3 as TX. Knowing that P0SEL is selecting register for port 0 and P0DIR the direction of the port:

```
if ( config->flowControl )
{
    cfg->flag |= UART_CFG_FLW;
    U0UCR = UCR_FLOW | UCR_STOP;  // Must rely on H/W for RTS
    P0SEL |= (1 << 3) | (1 << 2);   // Set P0.2, P0.3 as USART rx, tx.
    P0DIR = (P0DIR & 0x06) | 0x04;  // Set P0.2 as input, P0.3 as output.
    RX0_FLOW_ON;
}
```

Finally, the UART is configured and is ready to be used when is necessary to communicate the SmartRfBoard to the laptop as it will be shown when a message is received over the radio interface. Moreover when the specific coordinador.c(see Fig.5.1) initialisation is invoked the following PIB attributes and other required flags are settled by calling CONTROLLER_CoordiantorStartup():

```
void CONTROLLER_CoordinatorStartup()
{   /* Setup MAC_EXTENDED_ADDRESS */
    macMlmeStartReq_t   startReq;
    sAddrExtCpy(controller_ExtAddr, controller_ExtAddr1);
    /* Setup MAC_SHORT_ADDRESS */
    MAC_MlmeSetReq(MAC_EXTENDED_ADDRESS,&controller_ExtAddr);
    MAC_MlmeSetReq(MAC_SHORT_ADDRESS, &controller_CoordShortAddr);
    /* Setup MAC_BEACON_PAYLOAD_LENGTH*/
    MAC_MlmeSetReq(MAC_BEACON_PAYLOAD_LENGTH,&controller_BeaconPayloadLen);
    /* Setup MAC_BEACON_PAYLOAD */
    MAC_MlmeSetReq(MAC_BEACON_PAYLOAD, &controller_BeaconPayload);
    /* Enable RX */
    MAC_MlmeSetReq(MAC_RX_ON_WHEN_IDLE, &controller_MACTrue);
    /*Setup AC_ASSOCIATION_PERMIT*/
    MAC_MlmeSetReq(MAC_ASSOCIATION_PERMIT, &controller_MACTrue);
    /* Fill in the information for the start request structure */
```

```
        startReq.startTime = 0;
        startReq.panId = controller_PanId;
        startReq.logicalChannel = CONTROLLER_MAC_CHANNEL;
        startReq.beaconOrder = controller_BeaconOrder;
        startReq.superframeOrder = controller_SuperFrameOrder;
        startReq.panCoordinator = TRUE;
        startReq.batteryLifeExt = FALSE;
        startReq.coordRealignment = FALSE;
        startReq.realignSec.securityLevel = FALSE;
        startReq.beaconSec.securityLevel = FALSE;
        /*Turn off power manager*/
        osal_pwrmgr_device(PWRMGR_ALWAYS_ON);
        /* Call start request to start the device as a coordinator */
        MAC_MlmeStartReq(&startReq);
    }
```

On one hand, the PIB attributes are settled using MAC_MlmeSetReq(uint8 pibAttribute, void *pValue). Firstly, a short and an extended addresses are provided, the beacon payload is set and the radio is turned on even when the coordinator is idle. Also, the association permission is settled. On the other hand, a macMlmeStartReq_t startReq is declared to be filled and execute a MAC_MlmeStartRequest() . This invocation, sets the PanID in which it will become the coordinator and the radio channels in which it will operate, by default, the channel assigned is MAC_CHANNEL_11. Also, the superframe and beacon orders are settled, the value for controller_BeaconOrder and controller_SuperFrameOrder comes from the definitions in the controller.h. Meanwhile, realignment is not allowed since no synchronization request will come from devices and no changes on the superframe definition are expected once devices are associated, though, devices will check it when beacons are retrieved. Security parameters are declared false, as a non employed feature. The coordinator will be none stop beaconing since it is not battery sensitive so the power manager provided by OSAL is turned off by calling osal_pwrmgr_device() with parameter PWRMGR_ALWAYS_ON.

When the MAC_MlmeStartReq() is executed the coordinator starts sending beacons on regular basis defined by the beacon order. Meanwhile it will stand around to an association request.

5.2.2.3 Association branch

The association branch is only performed when a device is not associated to the coordinator. This will occur only once , device and coordinator will be paired since the first association. Devices will check if they are partners each time they retrieve a beacon, when a mismatch is detected it ask for association to prevent being orphaned. When an association request is identified in the event processor the CONTROLLER_AssociateRsp() is performed.

```
Event processor:
    case MAC_MLME_ASSOCIATE_IND:
        CONTROLLER_AssociateRsp((macCbackEvent_t*)pMsg);

Controller layer:

    void CONTROLLER_AssociateRsp(macCbackEvent_t* pMsg)
    {
        /* Assign the short address  for the Device, from pool */
        uint16 assocShortAddress =
        controller_DevShortAddrList[controller_NumOfDevices];
        /* Build the record for this device */
        controller_DeviceRecord[controller_NumOfDevices].devShortAddr =
        controller_DevShortAddrList[controller_NumOfDevices];
        controller_NumOfDevices++;
        /* If devices> MAX_DEVICE_NUM, turn off the association permit */
        if (controller_NumOfDevices == CONTROLLER_MAX_DEVICE_NUM)
        {
            MAC_MlmeSetReq(MAC_ASSOCIATION_PERMIT, &controller_MACFalse);
        }

        /* Fill in association respond message */
        sAddrExtCpy(controller_AssociateRsp.deviceAddress,
        pMsg->associateInd.deviceAddress);
        controller_AssociateRsp.assocShortAddress = assocShortAddress;
        controller_AssociateRsp.status = MAC_SUCCESS;
        controller_AssociateRsp.sec.securityLevel = MAC_SEC_LEVEL_NONE;
        /* Call Associate Response */
        MAC_MlmeAssociateRsp(&controller_AssociateRsp);

    }
```

The first part of the code is intended to check if the number of associated devices has reach the maximum and set the association permission off if so. In case it is still possible to associate, the coordinator responds the device with an association response performed by MAC_MlmeAssociateRsp()

and sending the assigned short address and a MAC_SUCCESS as transaction status. Security level is set to NONE.

5.2.2.4 Data indication

The data indication is the event that finishes the communication between coordinator and the end-device. This event reaches the event processor when the MAC detects that a data message has been sent to the coordinator from a device. The MAC event message is retrieved and the function CONTROLLER_GetInfo defined on the controller layer implements the logic to process it.

```
Event processor:
        case MAC_MCPS_DATA_IND:
                CONTROLLER_GetInfo((macCbackEvent_t*)pMsg);

Controller Layer:

        void CONTROLLER_GetInfo(macCbackEvent_t* pMsg)
        {
                HalUARTWrite(0,pMsg->dataInd.msdu.p,102);
                HalUARTClose(0);
        }
```

As is shown, the whole data packet “msdu.p” is sent by the 0 port of the UART to the serial port of the laptop to achieve the final step of the information sent by the end-device.

5.2.3 Device code project

5.2.3.1 Overview

Device flow chart is a pretty more consistent rutine than coordinator. The init process as well declares the main invocations required by the system and also declares the beginning of the data acquisition from the sensors. Moreover, the coordinator search is started to reach the association and continues to send the data following the opportunistic protocol which as a resume obeys the following rules:

- Pick data on a regular basis.
- Try to get contact with coordinator for a defined number of retries, if this number of retries is overtook a wait period is settled to start the rutine for a second time, and reapeat this process till the contact is reached.

- Send data, if available, and expect a successfully process if not, retry as many times as permitted.

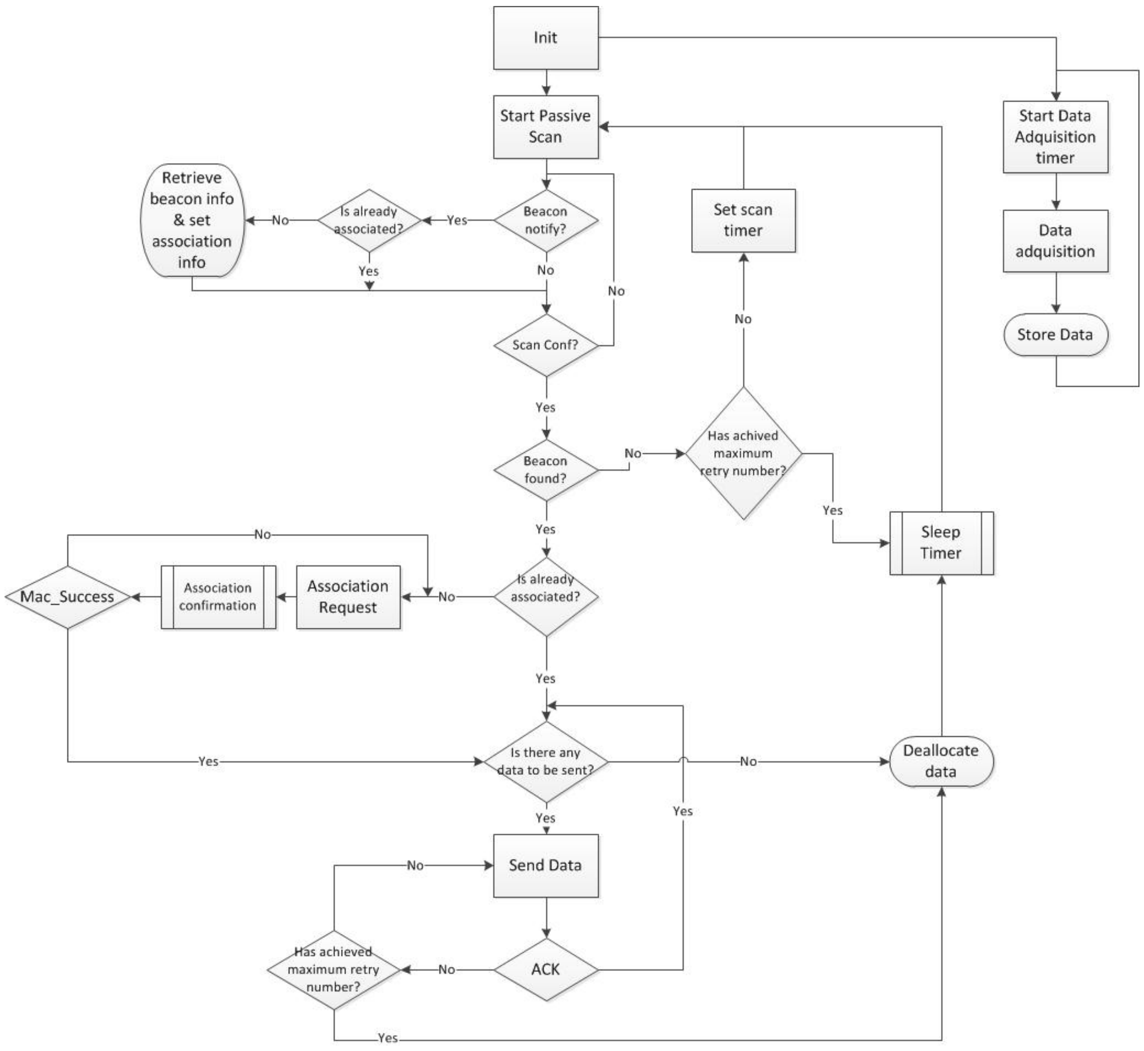


Figure 5.3: End-Device flow chart

5.2.3.2 Init process

The device initialisation process invokes de msa_Main and msa_Osal mandatory functions but it is needed to retrieve at least one beacon to set PIB attributes and flags, which definition will come from the coordinator, such as beacon and superframe order, PanID or Coordinator short address. The data acquisition rutine is initialized too.

At the very beginning of initialization it is needed to set the registers of the I/O ports to allow the procedure of data adquisition to get data form the input ports and to use an output pin to switch the sensor on/off:

```
P0DIR = 0x01;
```

```
P1DIR  |= 0x08;
P1_3 = FALSE;   //P1_3 is set to low level in advance
```

5.2.3.3 Data acquisition from sensor

The first step in this routine is to define the data structure to send the information required.

```
typedef struct DateReference
{
    uint8 minute;
    uint8 hour;     // 0-23
    uint8 day;      // 0-30
    uint8 month;    // 0-11
    uint8 year;    // +(2)000-(2)255
}DateReference_t;// 5bytes

typedef struct
{
    DateReference_t      Date;
    uint8       Data1[MAX_DATA_VECTOR];
    uint8       Data2[MAX_DATA_VECTOR];
    uint8       NodeId;
 }DataStruct_t;
```

DateReference_t, with a 5 bytes length, contains date references from minute to year, it will be useful to get the date of the samples. DataStruct_t has a maximum of 102 bytes available, 48 for each kind of sample, one for the NodeID which is an identifier independent from the Device Id assigned by coordinator at MAC level. Finally, the data reference estructure to get the first sample date. Since the data acquisition will be a regular basis, it is possible to get the first sample date of each packet and compute the following ones by adding the gap between them. In terms of routine, when the MSA_Init is performed an MSA_DATA_ADC event is set. This code performs a rutine that sets itself by a timer each minute, thus, a counter “conTimerDataRenew” is increasing till it achieves a value. Then, instead of setting the same event, the GoLayer_AdcRead() is executed and then is set once again the MSA_DATA_ADC event to start the routine once again. GoLayer_AdcRead is in charge of picking the samples from the ADC, as is detailed in the program extract below. The counter contTimerDataReadRenew controls if the number of samples has overtaken the maximum number of samples in each data packet which are 48 of each kind of sample. When the counter value is equal to 0 the DateReference is load by calling Go_Layer_GetTime() which returns a filled DateRefence estructure with the update. On the other hand, when this counter overtakes 48, it resets its value to 0 and increases the NewDataStructure_Cont and that represents that a new data packet estructre is going to be generated, in other words, the following samples will be placed in another data estructure.

To get the samples first we call to switch the sensor on and call HalAdcRead with the selected channel for each magnitude and the 8 bits resolution, this returns a value that will be stored within the data vector of the data structure. When the process is finished the sensor is switched off. To switch the sensor on/off it is only needed to simulate a pulse, bringing the ouput pin to high level and then return it to low level.

```
if (events &  MSA_DATA_ADC)
{
    if ( contTimerDataReadRenew <10)
    {
        contTimerDataReadRenew++;
        osal_start_timerEx(MSA_TaskId, MSA_DATA_ADC, 60000);
    }
    else
    {
        contTimerDataReadRenew=0;
        GoLayer_AdcRead();
        osal_set_event(MSA_TaskId, MSA_DATA_ADC);
    }
    return events ^ MSA_DATA_ADC;
}
-------------------------------------------------------------------------
void GoLayer_AdcRead()
{
    if(GoLayer_AdcReadCont == 0)
    {
        Go_Layer_GetTime(&GoLayer_Date[NewDataStructure_Cont]);
        GoLayer_Data[NewDataStructure_Cont]
            .Date=GoLayer_Date[NewDataStructure_Cont];
        GoLayer_Data[NewDataStructure_Cont].NodeId= Golayer_NodeId;
    }
    if(GoLayer_AdcReadCont<=MAX_DATA_VECTOR)
    {
        //Represents de maximum number
```

```
             //of samples availbale in a Mac data packet
            SwitchSensorOn();
            GoLayer_Data[NewDataStructure_Cont].Data1[GoLayer_AdcReadCont]=
            HalAdcRead(HAL_ADC_CHANNEL_1,HAL_ADC_RESOLUTION_8);
            GoLayer_Data[NewDataStructure_Cont].Data2[GoLayer_AdcReadCont]=
            HalAdcRead(HAL_ADC_CHANNEL_4,HAL_ADC_RESOLUTION_8);
            GoLayer_AdcReadCont++;
            SwitchSensorOff();
        }
        else if (GoLayer_AdcReadCont>MAX_DATA_VECTOR)
        {
            GoLayer_AdcReadCont=0;
            NewDataStructure_Cont++;
            GoLayer_AdcRead();
        }
    }
```

5.2.3.4 Scanning and retrieving beacons

In order to perform a scan, scanReq structure has to be filled and send it to the MAC by calling MAC_MlmeScanReq(), the entire process is done in the following procedure defined in the GoLayer:

```
    void GoLayer_SearchForBeacons()
    {
        macMlmeScanReq_t scanReq;
        /* Fill in information for scan request structure */
        scanReq.scanChannels = (uint32) 1 <<GoLayer_MAC_CHANNEL;
        scanReq.scanType = MAC_SCAN_PASSIVE;
        scanReq.scanDuration = GoLayer_ScanDuration;
        /* it must be adjusted to the duration of the awake-search*/
        scanReq.channelPage = MAC_CHANNEL_PAGE_0;
        scanReq.result.pPanDescriptor = GoLayer_PanDesc;
        /* Call scan request */
        MAC_MlmeScanReq(&scanReq);
    }
```

Firstly, it is defined within which channels we are scanning inside the defined channel page. In this case, the MAC CHANNEL SELECTED is number 11 and the CHANNEL_PAGE is 0 as well as the coordinator previous settings. The scan type is settled with the byte scanType and the attribute is defined on the MAC.h as MAC_SCAN_PASSIVE, in case of need of performing a passive scan. Byte ScanDuration sets the exponent to compute the duration of the scan as follows: scanDuration(ms) = (aBaseSuperFrameDuration ms) * (2 scanDuration + 1) As shown, it is set to the value of a local variable which can be accessed from the code to set different values. In fact, since the beacon order is 3 (120ms) the scanDuration order is 2 so the scanning time is 75ms which accomplish the sampling minimum frequency of half a beacon period. The pointer to buffer pPanDescriptor points to the value of the variable GoLayer_PanDesc which when beacon is retrieved gets the correct value. Finally, the procedure is executed and the request reaches the MAC to perform the scan in order to search for beacons around. When a beacon is found a MAC_MLME_BEACON_NOTIFY_IND indication arrives and then if the device is not already associated its incoming information is retrieved to fill the association structure.

```
case MAC_MLME_BEACON_NOTIFY_IND:
    if(!DeviceIsAlreadyAssociated())
    {
        GoLayer_SetAssociationRequiredInfo((macCbackEvent_t*)pMsg);
    }
    break;

void GoLayer_SetAssociationRequiredInfo(macCbackEvent_t* pMsg)
{
        //Fill association required info with beacon info
        GoLayer_AssociateReq.logicalChannel = GoLayer_MAC_CHANNEL;
        GoLayer_AssociateReq.coordAddress.addrMode = SADDR_MODE_SHORT;
        GoLayer_AssociateReq.coordAddress.addr.shortAddr=
        pData->beaconNotifyInd.pPanDesc->coordAddress.addr.shortAddr;
        GoLayer_AssociateReq.coordPanId=
        pData->beaconNotifyInd.pPanDesc->coordPanId;
        //Set superframe and beacon order and set local variables
        GoLayer_PanId = pData->beaconNotifyInd.pPanDesc->coordPanId;
        GoLayer_CoordShortAddr=pData->beaconNotifyInd.pPanDesc->
        coordAddress.addr.shortAddr;
        /* Retrieve beacon order and superframe order from the beacon */
        GoLayer_BeaconOrder = MAC_SFS_BEACON_ORDER
            (pData->beaconNotifyInd.pPanDesc->superframeSpec);
        GoLayer_SuperFrameOrder = MAC_SFS_SUPERFRAME_ORDER
            (pData->beaconNotifyInd.pPanDesc->superframeSpec);
}
```

The scan procedure will end up and will notify it to the MAC with a MAC_MLME_SCAN_CNF. The following code within the event processor will be then executed:

```
case MAC_MLME_SCAN_CNF:
    if (pData->scanCnf.hdr.status == MAC_SUCCESS )
    {
        contUnsuccesfulScans= 0;// reset cont of unsuccessful scans
        msa_ScanIsFinished=TRUE;
        if(!DeviceIsAlreadyStarted() && !DeviceIsAlreadyAssociated()
        {
            /* Start the device up as beacon enabled  */
            GoLayer_StartDevice();
            /* Call Associate Req */
            GoLayer_AssociateRequest();
        }
        else
        {
            osal_set_event(MSA_TaskId, MSA_SEND_EVENT);
        }
    }
    else if (pData->scanCnf.hdr.status ==MAC_NO_BEACON)
    {
        if (contUnsuccesfulScans <= msa_MaxScanAttempts)
        {
            //Prepare briefly for another scan
            osal_start_timerEx(MSA_TaskId, MSA_SCAN_EVENT, msa_NextScan);
            //Topup as unsuccesful scan try
            contUnsuccesfulScans= contUnsuccesfulScans +1;
        }
        else
        {
            //More than 4 try, nothing found, prepare for a long
            period sleep and then a scan
            contUnsuccesfulScans=0;
            osal_set_event(MSA_TaskId, MSA_TIMER_RENEW);
        }
    }
    break;
```

The scan confirmation notification header contains a status field which provides either the success or the failure of the scan. When beacons have been found it becomes a MAC_SUCCES and the subsequent action is to start the device and ask for association. In case the device has been initialized before and it is out of the initilisation process when a scan is performed, it is because it is in the ordinary program routine so it will send the data available as expected.

When the scan confirmation status is a MAC_NOBEACON it means no beacon was found, in other words, there is no coordinator around. Besides, if the scanning attempt failed, a new scan is settled using a timer expiration of half a beacon period (reinforcing the effect of scanning for more than a half a beacon period as explained before), while not overcoming the maximum number of scanning attempts. If the maximum number of scanning attempts is reached and no beacon was found the event TIMER_RENEW is set. This event renews the timer every 60 seconds beacause the maximum number of seconds that can be settled by the OSAL timer service is 65 seconds. This "search for coordinator" procedure is intended to discover the coordinator once the device is on its own. It is supposed that device will be initialised with the coordinator around so the previous procedure will not be necessary and, thus, the confirmation status will be success and the device will be started.

5.2.3.5 Device start up

This procedure is executed only if the device is not associated to the current coordinator and sets some of the attributes coming from the beacon notify in the MAC PIB chart. The most important part is to set the PanID and the coordinator short address for further communication attempts. Also, receiver is set to be off when is not waiting for any notification or no scan is going on. This is possible by setting the PIB attribute MAC_RX_ON_WHEN_IDLE to false. In addition, the superframe order and beacon order required from the MAC is settled using the local variable used before on the association required info procedure, if the superframe order is 15 the MAC will read into that beacons are disabled. Finally the power manger is set to true, since the main requirement of the device is to save as much energy as possible.

5.2.3.6 Association request

The association request will execute the following code passing to the MAC the values settled on the association required info procedure.

```
void GoLayer_AssociateRequest()
{
    MAC_MlmeAssociateReq(&GoLayer_AssociateReq);
}
```

An association confirmation is always expected and if it has success a procedure retrieves the assigned short address by the coordinator to be settled in the MAC PIB. After that, the send event is set, to start sending data. In case no success is reached, another confirmation request will be done till the maximum number of attempts is overcome.

```
case MAC_MLME_ASSOCIATE_CNF:
    /* Retrieve the message */
    GoLayer_IsAssociated((macCbackEvent_t*)pMsg);

void GoLayer_IsAssociated(macCbackEvent_t* pMsg)
{
    if ((pData->associateCnf.hdr.status == MAC_SUCCESS))
    {
        GoLayer_IsStarted = TRUE;
        /* Retrieve MAC_SHORT_ADDRESS */
        GoLayer_DevShortAddr = pData->associateCnf.assocShortAddress;
        /* Setup MAC_SHORT_ADDRESS - obtained from Association */
        MAC_MlmeSetReq(MAC_SHORT_ADDRESS, &GoLayer_DevShortAddr);
    }
    else if(GoLayer_AssociationAttempts< GoLayerMaximumAssociationAttempts)
    {
        GoLayer_AssociationAttempts++;
        GoLayer_AssociateRequest();
    }
}
```

5.2.3.7 Sending Data

This rutine is called when a succesful scan has been performed as a result of the expiration of the sleeping time during which the program has been picking samples of temperature and light. Now it is ready to be sent to coordinator, firstly we allocate the buffer at MAC level and get sure there is information available to be sent. Once this process is done, the required structure by the MAC_McpsDataReq is prepared by setting the address mode to short type, destination address and Pan identfier. Optionally we can ask for ACK to ensure the correct arrival of data and continue to send it if there is available.

```
void GoLayer_SendData()
{
    if ((pdata = MAC_McpsDataAlloc(
        102, MAC_SEC_LEVEL_NONE, MAC_KEY_ID_MODE_NONE)) != NULL
        && !GoLayer_NoDataAvailable)) // Look if there's data to send
    {
        pdata->mac.txOptions = MAC_TXOPTION_ACK;
        /* If it's the coordinator and the device is in-direct message */
        /* Copy data */
        osal_memcpy (pdata->msdu.p,
        &GoLayer_Data[NewDataStructure_Cont], 102);
        /* Send out data request */
        MAC_McpsDataReq(pdata);
    }
}
```

When the confirmation arrives, if MAC_SUCCESS is retrieved the procedure InfoHasBeenSent() decreases the data packet to sent, and the Sending data routine is recalled. In case NO_ACK reaches the processor it performs a retry of sending the same data failed before as many times as permitted by the variable msa_NodataConnection. In worst case, i.e. no contact, device goes to sleep and will retry next time the coordinator is found.

```
case MAC_MCPS_DATA_CNF:
    if(pData->dataCnf.hdr.status == MAC_SUCCESS)
    {
        InfoHasBeenSent();
        GoLayer_SendData();
    }
    if(pData->dataCnf.hdr.status == MAC_NO_ACK)
    {
        if( msa_Cont_NodataConnection <= msa_NodataConnection)
        {
            GoLayer_SendData();
            msa_Cont_NodataConnection++;
        }
        else if( msa_Cont_NodataConnection > msa_NodataConnection)
        {
            osal_set_event(MSA_TaskId, MSA_TIMER_RENEW);
        }
    }
```

5.2.3.8 Sleep timer

The sleep timer is performed by renewing a 1 minute timer for a defined number of times, determined by how much time is predicted to wait to the bus to pass nearby. When the counter overcomes the maximum renew times it sets a scan event which performs a scan to continue with the flow chart.

```
(events & MSA_TIMER_RENEW)
{
    if(contTimerRenew < GoLayer_TimesToRenew)
    {
        osal_start_timerEx(MSA_TaskId, MSA_TIMER_RENEW, 60000);
        cont++;
    }
    else
    {
        osal_set_event(MSA_TaskId, MSA_SCAN_EVENT);
    }
    return events ^ MSA_TIMER_RENEW;
}
```

5.2.4 Desktop application

Once the oppnet is properly running, data must be stored and an application must be created to access then for the processing. The final decision was to use SQL Sserver 2008 R2 and Windows Forms (C#) .The database is in charge of storing the data and a public webservice has been developed to insert the new data and get them when they are required, a console application gets the information from the serial port and finally a windows forms application has been created as final development of the project to show the results.

5.2.4.1 Database

The database has been created with SQL Server 2008 R2 and contains different tables which have relevant information not only about the collected data, the location and the owner of each node are also registered. Besides, it will allow the application to show more complete information about what the opportunistic network is providing. The database diagram in Fig.5.4 shows the columns and the relational clauses of each table. The tables Light and Temperature have a unique identifier for each measure and the corresponding value, the date of the sample and the node that picked it. Both tables have a foreign key with the NodeInfo table in which the node related information is stored, thus, any of the devices could be erased from the database by error.

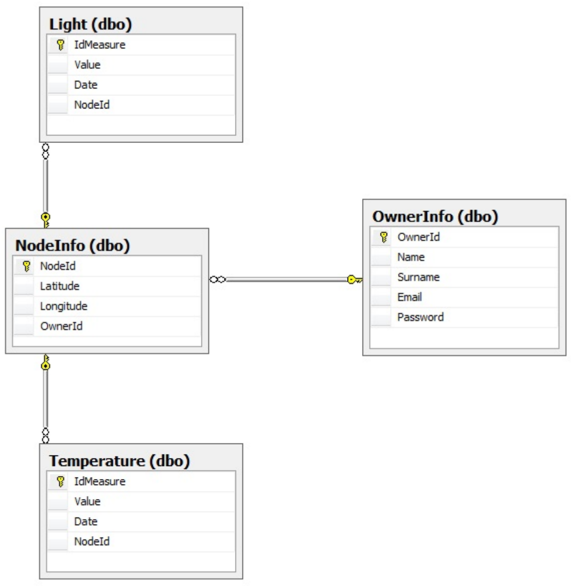


Figure 5.4: Database diagram

Finally, the OwnerInfo table has the relevant information of the owner of the node which is also a part of the NodeInfo as we can see in the foreign key relation. All the following applications includes a project called Oppnet.DTO, there is a Data Transfer Object (DTO) for each table with a property for each field of the corresponding table, except Ids which are defined as identity.

5.2.4.2 Webservice

The webservice provides public functionalities which allow the user building custom applications and not necessarly use the one that has been developed since it may not cover its own necessities. There are methods to insert the samples in the database by Node id (which is contained in the samples DTOs). These methods are only consumed by the application developed for the laptop connected to the coordinator. The other methods are intended to get the information by different parameters and to insert the new nodes which are being added to the network. Eventually, some methods are provided in order to register the owners and then get their nodes information. The webservice is summarized in the following interface-view-mode code extract.

```
public class OppNetWebService : System.Web.Services.WebService {
[WebMethods]
public bool InsertNewUser(DTOUserInfo userInfo)
public DTOUserInfo GetUserByEmailAndPassWord
(string email, string passWord)
public bool InsertLightSamples
(List<DTOLightSample> LightSamples)
public bool InsertTemperatureSamples
(List<DTOTemperatureSample> LightSamples)
public List<DTOLightSample> GetLight_ByDateAndIdUser(DateTime Begin,
DateTime End, int IdUser)
public List<DTOTemperatureSample> GetTemperature_ByDateAndIdUser
(DateTime Begin,DateTime End, int IdUser)
public bool InsertNewNode(DTONodeInfo NodeObj)
public bool DeleteNode(DTONodeInfo NodeObj) }
```

5.2.4.3 Retrieving data from serial port

To retrieve the data from the serial a port a console application is continuously waiting for new incoming data. It is necessary to instantiate a SerialPort class object and initiliasing as the following code extract illustrates, setting the values for COM port used, the baudrate, in this case the absence of parity and handshake, one stop bit and 8 data bits, according to the previous configuration for the smartRfBoard.

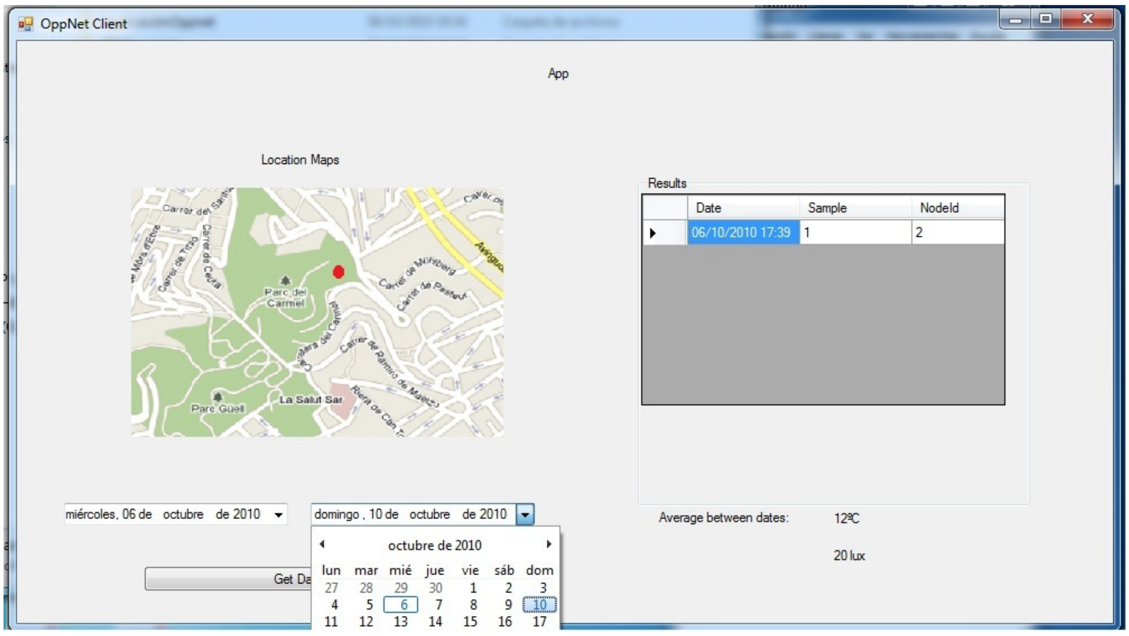
```
SerialPort sp = new SerialPort();
sp.PortName = "COM1";
sp.BaudRate = int.Parse("38400");
sp.Parity = (Parity)Enum.Parse(typeof(Parity), "None");
sp.StopBits = (StopBits)Enum.Parse(typeof(StopBits),"One");
sp.DataBits = int.Parse("8");
sp.Handshake = (Handshake)Enum.Parse(typeof(Handshake), "None");
```

Then an infinite loop performs a serialport read routine which fills a 102 byte array with the incoming data. When the buffer is filled the DTO lists are prepared to be sent via webservice to store the data in the database. As shown in the following routine, each DTO field is filled with the corresponding byte of the array. The first 5 bytes correspond to the date and the following are the samples which its initial position are referred by “five” and “fiftythree” variables. The last position of the buffer is for the NodeId.

```
do{
   List<DTOLightSample> LightList = new List<DTOLightSample>();
   List<DTOTemperatureSample> TempList=
   new List<DTOTemperatureSample>();
   sp.Read(buf,0,102);
   minute = buf[0];hour = buf[1];day = buf[2];
   month = buf[3];year = buf[4] + 2000;
   Date = new DateTime(year,month,day,hour,minute,0);
   for (int i = 0; i < Bufsize; i++)
   {
      //add 10 minutes per sample
      Date = Date + new TimeSpan(0, 10, 0);
      LightList.Add(new DTOLightSample(Date, buf[five], buf[101]));
      TempList.Add(new DTOTemperatureSample
                     (Date, buf[fiftythree], buf[101]));
      i++; five++; fiftythree++;
   }
}while(true);
```

5.2.4.4 Client application

The client application has been developed in order to provide a friendly interface to interact with the OppnNet results, Fig.5.5 shows the final view. A registered user can log in the application and access to the data base, selecting the date of the samples to be shown.



	Date	Sample	NodeId
▸	06/10/2010 17:39	1	2

miércoles, 06 de octubre de 2010

domingo, 10 de octubre de 2010

octubre de 2010

lun	mar	mié	jue	vie	sáb	dom
27	28	29	30	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17

Average between dates: 12ºC

20 lux

Figure 5.5: Client Application

Samples are shown in a grid ordered by date with the corresponding NodeID, another field illustrate the average temperature and light intensity result of the selected dates in the current area.

Google Maps Api has provided the structure to insert the map of the corresponding zone of the current user's OppNet . This is a quite visual gadget which makes the results of the oppnet less abstract because it is possible to locate the measures in a concrete geographic location.

5.2.5 Results

All the code project was tested while it was being developed. The code project explained in previous sections is the final source code after all the required testing to ensure the correct operation. In this section it is going to be tested the final power consumption of the end-device to know whether the goal of a required battery saving system has been achieved or not.

5.2.5.1 Measurements

The CC2430 datasheet provides information about the consumption in each power mode, but first it is recommendable to know the consumption of the end-device hardware platform, see section 4.3, without the chipset to ensure that this part is not going to produce a high current consumption which would finish the source battery.

All the circuit consumption can be measured using a multimeter, measuring the current running through the cable connected to the voltage regulators. This measure will tell the whole system comsumption, see table 5.1 .

End device without chipset consumption	0.0402	mA

Table 5.1: End-device consumption without chipset CC2430

The next measurement required is the end-device with the chipset CC2430 in Power Mode 2, the expected value since the datasheet provides a value of 0,9µA, see table4.1, is 0.0409 mA. The measurement is shown in table **??**.

End device withchipset consumption, PM2	0.0412	mA

Table 5.2: End-device consumption with chipset CC2430 PM2

The last measure is the whole system with receiver activated, as expected the consumption in this case is the greatest of the whole system. The measurement using a -50dBm sensitivity configuration is shown in table5.3.

End-device consumption with chipset CC2430 Rx active	28.25	mA

Table 5.3: End-device consumption with chipset CC2430 Rx active

5.2.5.2 Summary and predictions

In table 5.4 it is shown the summary of the previous measurements. The measurement of TX radio active has been dismissed because of the cunsuming simmilarity with the Rx mode and the difficulty to capture the moment when the device is transmitting, because of that, this moments will be assumed as RX.

Measurement	Value	Unit
End device without chipset consumption	0.0402	mA
End-device consumption with chipset CC2430 PM2	0.0412	mA
End-device consumption with chipset CC2430 Rx active	28.25	mA

Table 5.4: Measurements summary

Although the randomness of the system operation it is possible to estimate the probable duration of each battery device, knowing that the battery has a capacity of 300mAh.

Use in sleep or power mode 2:

$$Time = \frac{Capacity(mAh)}{CurrentConsumption(mA)} = \frac{300mAh}{0.0412mA} = 7281.55h$$

Use in RX active mode:

$$Time = \frac{Capacity(mAh)}{CurrentConsumption(mA)} = \frac{300mAh}{28.25mA} = 10.61h$$

For the case in RX mode it must be divided between the gap time in which device is scanning to find the coordinator, we can estimate an average time of 5 minutes each day assuming only one recollect per day. This is a quite bad case since it is meant to find the coordinator in the predicted moments, so this is a pessimistic estimation for days of use which result is good enough:

$$Days = \frac{Time - in - Rx - Mode}{Gap - Time} = \frac{10.61h * 60min}{5min} = 127days$$

Chapter 6

Conclusion and Future Work

6.1 Conclusion

All along this work we have introduced the emerging opportunistic technique of wireless networks and justified their future application in a concrete scenario. Furthermore, asbtracting the main idea of the development in this project it is easy to see the potential not only for environmental applications, agriculture or industrial solutions may be succesfully deployed with the different possibilities of the wireless opportunistic protocols. In fact, as the work has been going on new ideas have been coming up, much of them are explained in the future work section. Some way, the potential of this kind of network has been shown against in terms of planification since we have got to be focused on the objective of the project and not spreading it into more lines of work that the bounded time has left as future work.

IEEE802.15.4 and the Chipset CC2430 have turned into a good choice as they have amply responded to all the requirements of the project and have been a quite useful tool to develop this academic investigation, alltogether have allowed builidng a long lasting network which can opperate with barely human interaction as the main objectives ruled at the design approach . On the other hand, since the scenario has been only emulated, its correct behaviour in a real context still has to be proven.

In this context, we can say that the objective of building an oppnet and design a possible operation scenario has been succesfully beated but it must be said that there is a lot of work to do which can improve the actual performance and develop new functionalities.

6.2 Future Work

This sample application can be extended in a lot of different scenarios, thus, to create a bigger and a more robust network different facts must be considered. For instance, security stuff is not contemplated in this project but both the radio standard and the chipset are able to deal with encrypted transactions with support to AES-128. This is a considerable improvement which must be implemented when an application deals with sensitive data going around the network.

Another line of work is about the size of the network, in this deployment the network behaviour has been tested with a reduced number of devices, but if a bigger and powerful network is desired then the Guaranteed Time Slots which are part of the Superframe structure definitions can be considerated. This technique allows to ensure greater success chances when different devices want to reach the coordinator together and stablish order on the communications by eliminating part of the randomness of the medium access in the slotted CSMA-CA.

In terms of battery saving, one of the biggest improvements is designing a hardware part able to detect beacons or in any case the signal coming from the coordinator and produce an external interruption on the chip which may allow to use the deep sleep modes of the CC2430 and the radio receiver, which is the most wasting energy part of the current development, will only be turned on when it is known that the coordinator is around.

Finally, another improvement that we have found out whilst the development was in motion is the possibility of combining different star topology networks with the mesh one. This idea brings the oppnet more range since each star network is in charge of collecting the data in its own area but the coordinator could become part of a mesh network able to distribute the data collected by its own network.

Bibliography

[802.4.15] IEEE Std. 802.4.15, “Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)”, *IEEE Std. 802.4.15*, 2009.

[Bis04] S. Biswas, R. Morris, “Opportunistic routing in multi-hop wireless networks”, *ACM SIGCOMM Computer Communication,* pp. 69-74, 2004.

[CC2430] CC2430DK Development Kit, http://www.ti.com/litv/pdf/swru133.

[Che06] Ling-Jyh Chen, Ling-Jyh Chen, Chen-Hung Yu, Tony Sun, Yung-Chih Chen, Hao-hua ChuA, “Hybrid routing approach for opportunistic networks”, *SIGCOMM workshop on Challenged networks*, pp. 213 - 220, 2006, Pisa, Italy.

[Esw05] Eswaran, Anand, Rowe, Anthony y Rajkumar, Raj. “Nano-RK: An Energy-Aware Resource-Centric RTOS for Sensor Networks”, 26th IEEE International Real-Time Systems Symposium (RTSS’05). 2005.

[Far08] Farahani, Shahin. “Zigbee Wireless Networks and Transceivers”, USA : Elsevier Ltd, 2008.

[Gis08] Gislason, Drew. “Zigbee Wireless Networking”, USA : Elsevier Inc, 2008.

[Gro01] M. Grossglauser, D. Tse. “Mobility increases the capacity of ad-hoc wireless networks”. *In IEEE Infocom*, 2001.

[Hof04] Hofmeijer, T.J., y otros. AmbientRT - real time system software support for data centric sensor networks. Proceedings of ISSNIP 2004. Melbourne, Australia : s.n., 2004.

[Kya08] Kyaw, Zin Thein y Sen, Chris. “Using the CC2430 and TIMAC for low-power wireless sensor applications: A power-consumption study”, Analog Applications Journal. s.l. : Texas Instruments, 2008.

[Leb05] J. LeBrun, C.-N. Chuah, D. Ghosal. “Knowledge based opportunistic forwarding in vehicular wireless ad hoc networks”. *In IEEE VTC* Spring, 2005.

[Lin05] A. Lindgren, A. Doria. “Probabilistic routing protocol for intermittently connected networks”, Technical report, draft-lindgren-dtnrg-prophet-01.txt, IETF Internet draft, July 2005.

[Sef07] H. Seferoglu, A. Markopoulou, “Opportunistic network coding for video streaming over wireless”, *IEEE*, pp. 191-200, November 2007.

[Sho07] Sohraby, Kazem, Minoli, Daniel y Znati, Taieb. “Wireless Sensor Networks: Technology, Protocols and Applications”, New York : John Wiley & Sons Ltd., 2007.

[Tex06] Texas Instruments. “Measuring power consumption with CC2430 & Z-Stack”, 2006.

[Tex07] Texas Instruments. MAC Sample Application Software Design. 2007.

[Tex08] Texas Instruments. CC2430: A True System-on-Chip solution for 2.4 GHz IEEE 802.15.4 / ZigBee®. 2008.

[Wan05] Y. Wang, S. Jain, M. Martonosi, K. Fall. “Erasure coding based routing for opportunistic networks. *In ACM SIGCOMM Workshop* on Delay Tolerant Networks, 2005.

[Ye02] Ye, Wei, Heidemann, John y Estrin, Deborah. “An Energy-Efficient MAC Protocol for Wireless Sensor Networks”, s.l. : IEEE, 2002.