Radio Frequency communication for Modular Robots

Master thesis

Mærsk Mc-Kinney Møller Institute
University of Southern Denmark
Odense, Denmark

Author:
Guillem Arimany, *gucas10@student.sdu.dk*
Supervisor:
Kasper Støy

Period:
Spring 2011

Abstract

We explore the suitability of Wireless Radio Frequency (RF) inter-module communication for modular robots. Our hypothesis is that, instead of using Infrared (IR) and wired links, RF could be used for module localization and for local and global communication. We design a *communication board* composed by a TI CC2420 radio chip and a PCB antenna, implement a communication architecture, and we analyze different localization measurements and methodologies to find a suitable module configuration and to solve the current challenges of inter-module communication and neighboring localization. We present both a single and multi-radio architectures and validate their performance through hardware experiments. Results show that wireless radios can provide low-cost, power-efficient and reliable neighbor-to-neighbor global and local communication as well as good approach to neighbor localization for modular robots.

Acknowledgements

We would like to thank our supervisor Kasper Støy, and all the robotics group at The Mærsk Mc-Kinney Møller Institute for providing us inspiration, and supporting our ideas. Also, we would like to thank Henning Bremsøe and Leandro de Haro for all the advices and material they provided us. Thanks to Charlotte Ryberg showing us the PCB process design and implementation, and Palle Hermansen for providing TinyOS guidance. Thanks to Haruhisa Kurokawa for the experienced advices and talks. Finally, we would like to thank everyone from Mærsk Mc-Kinney Møller Institute and University of Southern Denmark for all the help, support and for the terrific moments we spent together.

Contents

List of abbreviations

AOA	Angle Of Arrival
BW	Bandwidth
CCRR	Co-Channel Rejection Ratio
CRC	Cyclic Redundancy Check
DSSS	Direct Sequence Spread Spectrum
FCS	Frame Check Sequence
FHSS	Frequency Hopping Spread spectrum
FSPL	Free-Space Path Loss
HAA	Hardware Abstraction Architecture
IEEE	Institute of Electrical and Electronic Engineers
IR	Infrared
LED	Light-Emitting Diode
LOS	Line Of Sight
LQI	Link Quality Indicator
MAC	Medium Access Control
MPDU	MAC Protocol Data Unit
PAN	Personal Area Network
PCB	Printed Circuit Board
RADAR	Radio Detection And Ranging
RSSI	Received Signal Strength Indicator
RF	Radio Frequency
RX	Receiver
SFD	Start of Frame Delimiter
SMA	SubMiniature version A.
SR	Self-Reconfigurable
TI	Texas Instruments
TOA	Time Of Arrival
TX	Transmitter

1. Introduction

In nature, most constructions are made with modular parts. For instance, an atom is the simplest element which can take part in chemical reaction. The atoms consist of a nucleus with protons (positive charge) and neutrons surrounded by electrons (negative charge). A tidy and defined group of atoms form molecules, which at the same time is the simplest element in a chemical compound. Molecules consist in atoms of one or two elements. For instance, the molecule of water comprises two particles of hydrogen and one of oxygen as shown in Figure 3. A molecule is as smaller that it is impossible to see, but hundred of molecules together form a glass of water.

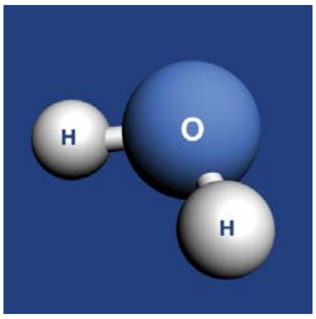


Figure 2. ODIN Modular robot

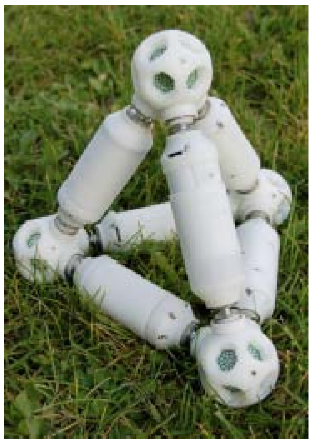


Figure 3. Water molecule

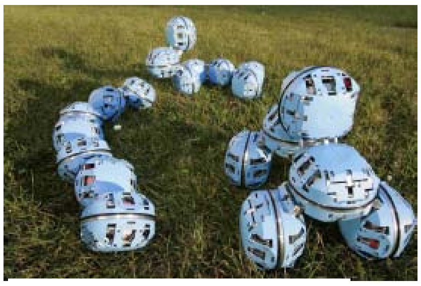


Figure 1. ATRON Modular robot

In modular robots, we use an analogous methodology to construct things similar to the known nature. Therefore, modular robots are built from modules that contain simplest components like actuators, sensors, or motors, and the capacity to communicate. One module alone cannot do anything like the atom, but a group of modules can develop different tasks like molecules do. The atoms of molecules are joined sharing or interchanging some electrons, while the modules in modular robots are joined with an active connector. This feature provides modules the ability to connect and disconnect from other modules producing a self-reconfigurable robot [1]. Figure 2 and

Figure 1 show ODIN and ATRON, two examples of modular robots created at Mærsk Mc-Kinney Møller Institute.

Self-reconfigurable (SR) modular robots are decentralized systems that rely on inter-module communication to afterwards combine into more complex entities. Thus and beyond conventional actuation, self-reconfigurable robots are able to intentionally change their own shape by reorganizing the connectivity of their parts in order to perform new tasks, adapt to new circumstances or recover from damage by coordinating the movements of the modules for reconfiguration and locomotion.

The most important desirable features in most self-reconfigurable robot systems are:

- *Robust*: robustness is born of the redundancy of modules. A hardware or software failure may cause a module to fail, but the remaining modules can compensate for the loss of a module [2].
- *Versatile*: modular robots are built with several modules, and the modules can be combined in many different ways, allowing a wide range of different robots.
- *Adaptable*: the modules of the robot can continually adapt and even change the shape if the task requires it.
- *Cheap compared with their complexity*: the modules can be mass produced and the cost of each one can be reduced [3].
- *Scalability*: the linear relationship between the functionality of the system and the number of modules [4].

These desirable features depend on the communication between the modules and, despite the central role communication plays in SR systems, it remains a major research issue. Provisioning of scalable and robust communication architectures is one of the major challenges that still exist in the practical implementation of these systems.

1.1. Motivation

Nowadays, the communication between modular robots is essential for sensing and coordinating movements in a distributed system. However, the communication between modular robots is still a challenge. The topology of the network dynamically changes, and it is difficult to know the function of each module every time, or if a new module is connecting or disconnecting, etc.

Currently, communication between modular robots is based on infrared (IR), Bluetooth or wired links. However, the use of these technologies has several problems. The main problem of Infrared links is the need of accurate alignment and orientation between both modules to achieve the communication. Further, the communication between two modules when they are connecting or disconnecting is also a challenge. The environment is an important factor as well, since dust and dirt can abrade or obstruct the optics and prevent electrical connections.

Wire links are limited in movements, and it is complicate to design the appropriate connector to do not damage the wire link. However, the main advantage is the wire links communication is very fast, even when using half of full duplex communication lines [5]. Another advantage is the low power consumption they have compared with the other types of communication, as it does not have to convert the signal from electrical current to electromagnetic waves or light.

Bluetooth is effective in small robots and has a high data rate (around 1Mbps), but it suffers from scalability problems on mesh network, and it needs the presence of a central node to maintain the robot topology. That means that robot is not robust, because if the host node fails, the robot fails [6].

Because of these problems, RF seems a good solution. The benefits of using RF solutions are that they do not require precise module alignment and can provide global and local communication. RF is more versatile and it could be installed in any modular robot without regarding to the inter-module docking orientation and mechanism. In addition, the communication of the system is expected to be more reliable. Kuo et al. proposed a scheme using wireless Radio Frequency (RF) links [7] where analyzes module misalignment and local communication issues.

On the other hand, potential drawbacks of RF communication, crosstalk elimination, and the challenge of neighbor detection. Another disadvantage it that RF is slower than wire links, as they have to communicate in both directions through the same medium. Having different modulation on the receiver and transmitter can solve this problem, but it is an expensive solution. According to that, half duplex is the common strategy used.

Kuo demonstrates that the Radio Frequency is the new challenge for Modular Robots, and that motivated us to continue studying the suitability of this kind of communication for modular robots by trying to introduce a new way of using the radios.

1.2. Goal

The goal of this project is to investigate if Wireless Radio Frequency Communication is suitable for communication between modular robots and it could solve some of the problems that other communication technologies have.

The Maersk Mc-Kinney Moller Institute is working hard on the modular robots scope and as we already know, local and global communication, and localization between modules of modular robots is a big challenge. For this purpose, a RF *Communication Board* will be designed, implemented and tested in different situation. Researchers from the Mærsk Mc-Kinney Møller Institute developed a *General Board*, which has an Atmel AT91SAM7S256 microcontroller that provides processing power to variety of applications. Therefore, this board will be used to attach the *Communication*

Board. One important part of the development of the *Communication Board* is the hardware: choose the best components to commit the communication, which kind of antenna is better for each application, which are the characteristics and limitations of the system, etc. Therefore, the purpose of this project is implementing different possible hardware options, analyze them and study if any of them are suitable for modular robots.

On the other hand, the software is another important part needed to communicate the Atmel microcontroller with the *Communication Board*. According to that, TinyOS operating system will adapted to handle all the software applications needed to run the tests. Furthermore, a radio stack and an IEEE 802.15.4 MAC based protocol will be implemented to establish the communication between modules. Finally, in order to work with modules localization and local and global communication, we must design and implement good algorithms and applications to achieve our goals.

1.3. Thesis outline

The thesis will be divided in six general parts. The second chapter describes the basic knowledge about RF Wireless communication, to select the suitable components to obtain communication between modules, and discuss the selection of the components to then design and implement the *Communication Board*.

The third chapter will incur on the measurement of the antenna designs in order to choose the suitable design for the purpose of the system, and the test to check if the hardware initially works as we expected.

The fourth chapter will explain the operating system choice, the software design and implementation, and how the communication between both boards will occur. Thus, the radio stack and the communication protocol designed, implemented and used will be explained.

Following to that, in chapters fifth and sixth, the reader will be introduced to the main objective of the thesis, describing different ways, problems and solutions regarding to local and global communication as well as localization. Thus, once the *Communication Board* is working and communication among the other modules, different tests will be made in order to check the performance and reliability of the system and analyze the suitability of the RF communication in modular robots.

Results show that RF communication links can be implemented between modules of modular robots and can provide both local and global communication. Results show as well that a combination of an omnidirectional plus a directional antenna builds in the same *Communication Board* is enough for communication and neighbor localization.

We conclude on chapter seven that RF communication is suitable for modular robots and, although there is still a research to do, our system demonstrated that local and global communication can be done using RF and provided a solution for neighbor localization.

2.

Design and implementation of the hardware

2.1. Introduction

The purpose of the thesis is design a *Communication Board* that could communicate with Radio Frequency locally and globally. Therefore, in this chapter we describe the basic parameters and techniques about RF Wireless communication we should know to design our system. Accordingly to that, we will analyze different possible components in order to choose the optimal ones for the system. Finally, we will design and implement the board with the chosen components.

2.2. Wireless Communication

Wireless communication is the transfer of information from place to place without cables. The data transmissions without cables between systems are widely used in industrial, commercial and medicine applications. The principal technologies involved in wireless communications are Infrared (IR), Bluetooth and Radiofrequency (RF).

As it was mentioned in the introduction, the common technology used in modular robots is IR. Few modular robots use Bluetooth, but there are no modular robot working with RF. That is the reason why in this point, we will focus in Radio Frequency communication as a wireless communication technology.

RF is a new challenge in modular robots. The main advantages of this technology compared with others are that it permits short and medium range communication, can cross obstacles, do not miss the communication in front of misalignment and does not need a module as a host to communicate. The main components of a RF communication are the transmitter, which modify the original signal to be suitable for the transmission, the mean of transmission where the signal travels between transmitter and receiver, and the receiver, which transforms the information into the original signal to process it later.

Thus, the general idea of the components of a RF communication is shown in Figure 4.

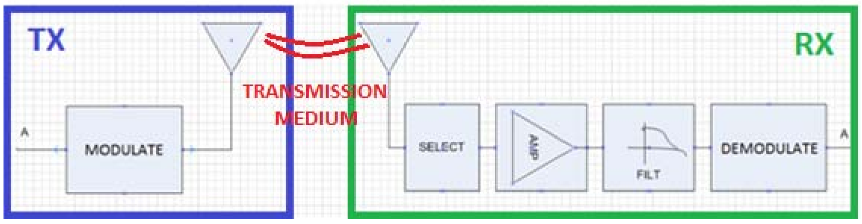


Figure 4. Components for RF Communication

The main components to implement a RF communication system are divided in:

Transmitter (TX):

- *Modulator*: transforms the signal in baseband (modulating signal) into a passband signal (modulate signal) at high frequencies.
- *Antenna*: transmit the signal.

Receiver (RX):

- *Antenna*: receive the signal.
- *Selective module*: the RF receiver cannot operate with all the signals that the antenna receives. When a signal becomes too large, the receiver performance begins to deteriorate; thus, there is a maximum power level that can be read without alter the signal, called saturation. When the signal is lower the sensitivity of the receiver put the limit of the smallest signal it can process. Therefore, the receiver has a dynamic range where it works, between sensitivity and saturation.
- *Amplifier*: the signal received has low power, and it is necessary to amplify to process better the signal.
- *Filter*: to eliminate the spurious or harmonics of the signal. It is necessary to obtain the signal of interest.
- *Demodulator*: convert the received signal into the original signal send it by the TX.

Thus, to implement a RF communication system is essential to implement each component. Nonetheless, there are devices which have the transmitter and receiver circuit combined, also called transceiver, which has implemented the entire characteristic explained before. However, to design a good RF system, it is necessary to take into account several variables such the work frequency or which technique is used to modulate/demodulate the signal. In the following points the relevant characteristics for the objective of the system will be explained.

2.2.1. Kind of RF Wireless Communications

There are different ways to classify the RF Wireless communication. One way is if the devices fulfill a standard protocol or not; another classification is according with the operating frequency that device works. Thus,

- **Fulfill a standard protocol or not**. There are some devices that do not fulfill a standard protocol; however most of them do [8]. There are different protocols working at 2.4 GHz ISM Band and based on the IEEE 802.15.4 standard:
    - *Zigbee*: is used for low data transmission applications. With this protocol is possible to achieve from 10 to 75 meters of distance.
    - *Wireless HART (Highway Addressable Remote Transducer Protocol)*: is an open-standard wireless network technology that provides a robust protocol for the full range of process measurement, control, and asset management applications used on industrial automation.
    - *RF4CE*: provides a RF platform that permits bidirectional, reliable and omnidirectional wireless communication, and the frequency agility to coexist with another 2,4GHz wireless technology.
    - *Synkro Protocol*: is designed to improve the control, monitoring and automation of entertainment electronic devices at home.

- **Operating frequency**. There are some frequencies bands in it is not necessary a license (if it does not overcome the power limits). These frequencies are:
    - < 1GHz (used to be from 300MHz to 900MHz)
    - 2,4GHz, which are normalized in the world.

In this thesis our protocol will be based on the IEEE 802.15.4 standard and we will use a free operating frequency in order to not pay the needed fees. Moreover, there are a large variety of commercial devices that operate in these frequencies.

2.2.2. Techniques used in RF

RF uses two types of technologies, narrowband or broadband which uses all available bandwidth, instead of use a carrier signal. These techniques are useful to do a reliable communication.

Some of these technologies are:

- *DSSS (Direct Sequence Spread Spectrum)*: This technology works transmitting at the same time in different frequencies, incrementing the possibility that the transmitted data arrived to the receiver. At the receiver is necessary to do the reverse process to obtain the original signal, hence the receiver must know the decryption patron to get the original signal. Therefore, only the receivers which the transmitter sends this decryption patron could understand the original signal. This technology achieves the highest reception sensitivity (the smallest RF signal detection).

- *FHSS (Frequency Hopping Spread spectrum)* is a method of transmitting radio signals by rapidly switching a carrier among many frequency channels, using a pseudorandom sequence known for both, transmitter and receiver.

- *RSSI (Receive Signal Strength Indication)* is a measurement of the power strength present at the receiver. It usually measure before the amplifier.

- *LQI (Link Quality Indication)* is a measurement of the quality of the received signal. The higher LQI the better range. This parameter is often linked to RSSI, because a powerful signal will be less affected by noise.

In modular robots, the modules are very close each other, for this reason the RSSI and LQI technique will be very useful to detect if the signal received is from the neighbor module or from another one. For the same reason (small distance between modules), the DSSS technique can be used to detect that other modules receive information that they do not need it. Therefore, some of these characteristics will be checked when we will choose the components for the design of the *Communication Board* in the next section.

2.3. Components selection

In high level, the components which take part in RF communication are the antenna, a transceiver and a microcontroller, as shown in Figure 5. The microcontroller controls and analyzes the data received, the transceiver processes and adapts the transmitted or received signal, and the antenna transmits the information.

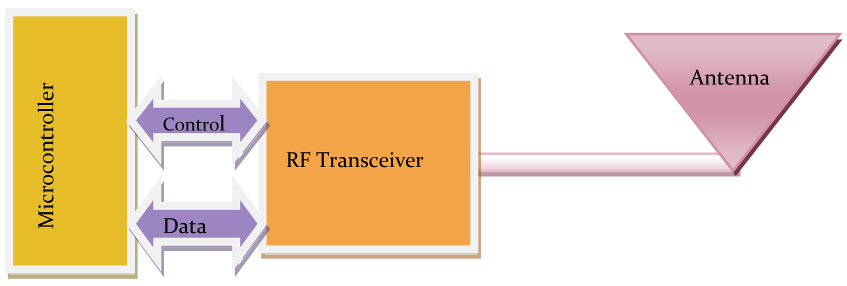


Figure 5. Components in RF Communication

In the following points each component of the RF communication will be analyzed in order to choose the adequate device for this objective.

2.3.1. Microcontroller

The microcontroller is used to control the transceiver. The Mærsk Mc-Kinney Møller Institute proposed us to use the microcontroller developed in the *General Board*[1], which may be integrated in future modular robots. It contains an embedded Atmel AT91SAM7S56microcontroller, which was used to program the *Communication Board*. The principal features of the *General Board* considered to choose the transceiver are:

- It contains a voltage regulator of 3.3V and maximum current of 800mA.
- An isolated save-circuit of 3.3V and 400mA (taken from the 800mA from above).
- SPI port to communicate with different devices.

It can share some microcontroller's pins with other peripherals; however some combinations are not possible.

As a conclusion, the main characteristics of the *General Board* we have to consider to choose the transceiver are that it provides 3.3V, maximum current of 400mA and SPI communication.

2.3.2. Transceiver

There are devices which have the transmitter and receiver circuits are combined, also called transceiver. By using a circuit as transmitter and another one as receiver with different antennas, we could have full duplex communication using different channels to communicate, but it is more complex and expensive.

For this objective is better and easier choosing a transceiver than two different circuits. Furthermore, the RF transceiver has some advantages comparing with the transmitter and receiver circuits like use modules for high speed data transmission.

The principal characteristics of the transceiver should be:

- *Short range*. At the RF wireless communication all of the parameters are related. For instance, the range depends on the operating frequency, the higher frequency the lower range. The range depends on the output power, but also the reception sensitivity. The output power and reception sensitivity also depends on the antenna, the sort of antenna and its features. And at the end, it depends on the environment, if it is indoor or outdoor, simple walls or cement walls, in a factory or in a building.

[1] Datasheet of the *General Board*: 01GeneralBoardV3-Datasheet.v01

Therefore, the range of the operating frequency depends on the frequency, output power, reception sensitivity, antenna's features and work environment.

- *Low power consumption.*
- *High frequency*. The dimensions of the antenna are smaller although is not enough to put inside the device package. Remember that according with the classification of operating frequency, the high frequency that do not need license to operate is 2.4GHz. Furthermore, there are large variety of devices and antennas that work at this frequency [9].

According with the *General Board* specifications, we searched different devices in different distributors such Freescale[2], Texas Instrument[3], Microchip[4], and Cypress[5]. Some of the characteristics of the components searched are in Table 1.

Distributor	Device	External components	Interface	Sensitivity	Baudrate (Kbps)	Output power	Power supply
Freescale	MC13201(Z)	15	SPI	-92dBm	250	-27dBm to 4dBm	2.0 to 3,4 V
Texas Instrument	CC2420 (Z)	10	SPI	-94dBm	250	-25dBm to 0dBm	2,1 to 3,6V
Cypress	CYW6934	10	SPI	-90dBm	62,5	Up to 0dBm	2,7 to 3,6V
Microchip	MRF24J40 (Z)	10	SPI	-95dBm	250	-36 to 0dBm	2,4 to 3,6V

Table 1 . Characteristics of each transceiver

In spite of all the chips are similar, the MRF24J40 has the limitation that is mandatory the use Microchip Technology microcontrollers and for this project we use the microcontroller of the *General Board*. The Cypress devices cannot fulfill a standard protocol in case if it will be necessary, and it is not recommendable for new designs. Therefore, between CC2420 and MC13201, the device selected was CC2420 because has better sensitivity and less external components.

Conclusion

The CC2420 component was selected because it is not necessary to connect it with a specific microcontroller, can fulfill a standard protocol, it has very good sensitivity and baud rate, it is designed for low power wireless applications and it supports 8 discrete

[2] http://www.freescale.com/webapp/sps/site/taxonomy.jsp?code=RF_TRANSCEIVERS
[3]http://focus.ti.com/paramsearch/docs/parametricsearch.tsp?family=analog&familyId=936&uiTemplateId=NODE_STRY_PGE_T
[4]http://www.microchip.com/ParamChartSearch/chart.aspx?branchID=1205&mid=&lang=en&pageId=76
[5] http://www.cypress.com/?id=16&source=products

power levels: 0dBm, −1 dBm, −3 dBm, −5 dBm, −7 dBm, −10 dBm, −15 dBm and −25 dBm at which its power consumption varies from 29 mW to 52 mW [10].

The principal characteristics of the CC2420 device are [11]:

- Is possible use it with or without standard protocol
- The voltage is between 2.1 V and 3.6 V.
- The consumption at RX/TX is between 17.4 mA and 18.8 mA.
- The operating frequency is 2.4 GHz.
- The device use the RSSI , LQI and DSSS techniques
- Standby consume is 0.2 uA
- SPI interface

The RF output of the transceiver is a differential output, hence a differential antenna should be chosen or a match circuit to change the differential output to single output should be designed.

The transceiver selected uses 4 I/O pins (SI, SO, SCLK and CSn) from the microcontroller for the SPI configuration interface. It is possible to interface with the transmitter and receiver FIFO using the FIFO and FIFOP pins, clear the channel assessment with the CCA, and check the timing information in SFD pin.

Hence, looking at the pins of the *General Board*, the connection between the microcontroller and the transceiver of the *Communication Board* was decided as shown in Table 2.

Once the transceiver was selected, the next step is to choose an antenna to connect to it.

General Board		Communication Board	
Pin	**I/O**	**Pin**	**I/O**
20	AD3	**29**	FIFOP
21	PA0	**27**	SFD
23	PA1	**41**	VREG_EN
25	PA2	**21**	RESETn
27	TDW	**30**	FIFO
29	TWCK	**28**	CCA
37	MOSI	**33**	SI
38	NPCS0	**31**	CSn
39	SPCK	**32**	SCLK
40	MISO	**34**	SO
22,24	+3V3	**43**	VREG_IN
15,17,19,30,31,33	GND	**5,9,19,22,23,24**	GND

Table 2. Pin connection between Communication Board and General Board

2.3.3. Antenna

The antenna is an important and complex component in wireless communication. Therefore, in the following points we describe the principal characteristics of the antenna, which describe the performance of the system. However, due to the complexity to design, simulate and implement an antenna, we propose two known omnidirectional antenna designs and two easier manufactured directional antenna designs to connect to the *Communication Board*.

As it will be explain later, the performance of these antennas should be measured in order to test their suitability for our system. Therefore, in the next chapter we will measure their performance and conclude which one should be used in the system.

2.3.3.1. Description

"The Institute of Electrical and Electronics Engineers (IEEE) defines an antenna as a part of a transmitter or receiver system designed to radiate or *receive electromagnetic waves."*

The purpose of the antenna is the radiation of power with specific characteristics according with the application system. The basic function of the antenna is transmitting and receiving with a specific directionality, power, frequency and other characteristics.

There are several classifications of antennas according with different parameters. For instance, according with the radiation pattern, the antennas could be directional or omnidirectional. The antennas also can be divided in single edge antennas (also called unbalanced antennas) and differential antennas, also called balanced antennas. Single edge antennas are fed by a signal which is referenced to ground and the input impedance is usually 50 Ohm. Differential antennas are fed by a reference signal of two different potential, and the impedance is a conjugate value.

The antenna is a part of a bigger system, and it is necessary to define all the parameters of the antenna that allow to see its effects in the system, or specify the desired behavior of the antenna that permit include it in the system. The main parameters are explained below.

Impedance

The impedance is the relationship between the voltage and the current in their terminals. It has real part (Ra) and imaginary part (Xa) [12].

Hence, the impedance of the antenna is: $Z_z = R_a\ (w) + j\ X_x\ (w)$

The real part could be divided in the radiation resistance at the antenna and the ohmic losses resistance at the antenna. Hence, it could be measured the power radiated and the power dissipated by heat.

Adaptation of the antenna

At the transmission and reception, the antenna should be connected to a transmission line, or directly connected to the transmitter and receiver.

If the antenna is well adapted, it means that all the power delivered to the antenna will be radiated, and the signal will not be loss by heat.

To measure the adaptation a Network Analyzer will be used. The S11 parameter shows the transference power in the antenna. This parameter is also called reflection coefficient and is defined as the relationship between the reflected wave and the incident wave.

$$\mathrm{S}_{11} = \frac{\mathrm{Z_A} - \mathrm{Z_0}}{\mathrm{Z_A} + \mathrm{Z_0}}$$

Equation 1. Adaptation of the antenna

Where $\mathrm{Z_A}$ is the input impedance of the antenna, and $\mathrm{Z_0}$ is the characteristic impedance of the transmission line connected to the antenna.

Standing wave ratio (SWR)

It is defined as the relationship between the minimum and maximum of the voltage or current standing wave at the output of the sweeper. In the case of the voltage relation, it is called VSWR (voltage standing wave ratio). This parameter is related with the adaptation of the antenna:

$$VSWR = \frac{1 + |S_{11}|}{1 - |S_{11}|}$$

Equation 2. Voltage Standing Wave Ratio

The return loss, that means the relationship between the incident and reflected wave, as shown in the following formula:

$$S_{11} = 20 \cdot \log \frac{VSWR - 1}{VWSR + 1}$$

Equation 3. Return loss

With the standing wave ratio we can know how much power is transmitted to the charge, or the reflection produced by the lack of adaptation. If VSWR=1, it means that the antenna is adapted (S_{11}=0). If we consider $S_{11} < -10dB$, the VSWR will be higher than 2.

Directivity

The directivity means the capacity of the antenna to concentrate the radiation in a specific direction. Usually power gain is expressed relative to an isotropic radiator or half-wavelength dipole.

Gain

Gain means the power delivered to the antenna in the direction of maximum emission with respect to an omnidirectional antenna and it is related with the directivity. Increasing the gain we can achieve more distance, but the width of the main lobe is reduced, doing the direction of the antenna more critical. Depending on the application where the directional antenna will be used, this balance should be considered in the design of the antenna.

Polarization

The polarization is the direction of the electric field emitted by the antenna. As it depends on the direction, the antenna should have the polarization in the maximum radiation direction. This polarization is always the same in the main lobe.

In radio communication systems, the polarization is important due to the receiver antenna only can notice the power of the same polarization field as itself. Therefore, to communicate two devices both must have the same polarization direction.

Radiation pattern

The radiation pattern is the graphic representation of the antenna radiation in different space directions; it means the graphic representation of the directional properties of the antenna. This representation could be in two planes: azimuth plane, which refers to the horizontal plane, and elevation plane which refers to the vertical. Both could be represented in Polar or Cartesian coordinates.

An example of the radiation pattern of a directional antenna is shown in Figure 6 and an example of the radiation pattern of an omnidirectional antenna is shown in Figure 7. The main lobe is the focal area where the radiation power is maximum. The zones around the maximum with lower amplitude are called secondary lobes. It is useful to determine graphically the directivity of an antenna.

Thus, the radiation pattern identifies the characteristics of the antenna. Depending on the application of the antenna, it will be interesting how directional is the antenna. Hence, analyzing the radiation pattern is easier to choose the antenna.

Bandwidth

The antennas are limited to work in a range of frequencies. This range of frequencies is known as bandwidth of the antenna and it could be defined from the radiation pattern or the reflection coefficient representation.

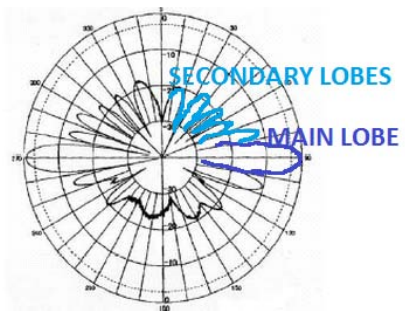
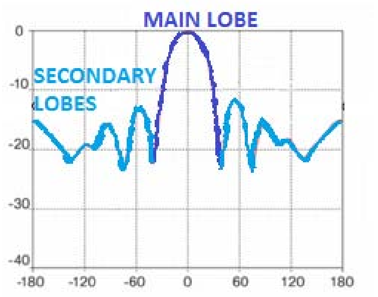


Figure 6. a) Polar coordinates of a directional antenna.



b) Cartesian coordinates of the directional antenna

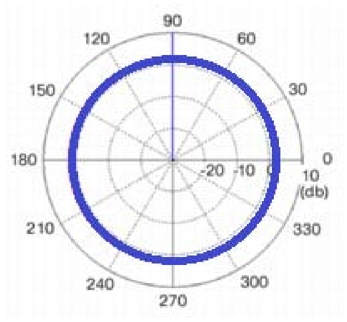
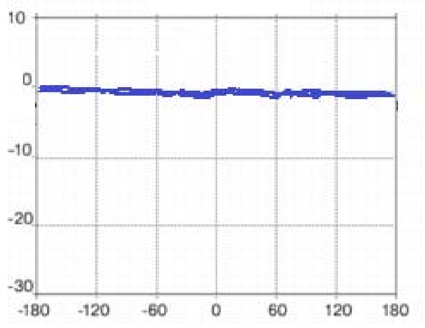


Figure 7. a) Polar coordinates of an omnidirectional antenna



b) Cartesian coordinates of the omnidirectional antenna

2.3.3.2. Antenna design and implementation

Depending on the purpose of the antenna should be necessary consider different parameters: bandwidth, gain, distance range, polarization, etc. Furthermore, there are more factors that influence in the performance of the antenna such the PCB materials, the antenna placement and the ground planes. Hence, design an antenna is not trivial, and there are a lot of characteristics to consider.

This project will be focus in single edge directional and omnidirectional antennas, because there are not the necessaries tools at the laboratory to measure differential antennas.

However, it is not easy to design, simulate, implement and measure an antenna. Therefore, easy known designs will be chosen to analyze if they are suitable for the *Communication Board*, in the following points.

2.3.3.2.1. Omnidirectional antenna

"Omnidirectional antennas radiate maximum power uniformly in all directions, in the horizontal plane[6]."

Omnidirectional antennas could be a good solution for a modular robot system, because a module can communicate with several modules at the same time. If every module has a unique name, only the modules who want to communicate will be active, and the others will be in the sleep mode.

It takes too much knowledge and time to design a good antenna. The lack of both did that some standard antennas references design were chosen to include in the *Communication Board*.

The transceiver chosen have different references antennas designs [13]. From the selection guide the AN043 and DN007 reference designs were chosen, due to the single edge condition, the small dimensions and the efficiency higher than 60%. In the following points both omnidirectional antennas will be described.

AN043 Reference design

The AN043 reference design is shown in Figure 8, and the ideal characteristics of the antenna are described in the reference [14].

Furthermore, another reason to use this choice is that a PCB antenna is easy to implement and is a good solution to reduce cost.

[6] Radio Frequency systems: http://www.rfsworld.com/index.php?p=354&l=1

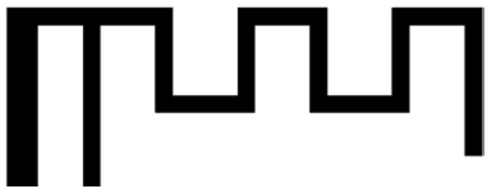


Figure 8. Shape of the AN043 antenna reference design

Some important parameters in PCB antennas are the ground, the dimensions and placement of the antenna, changing its size or position the performance changes. Hence, it is important to consider that some parameters described in the reference design [14] could change with the board (size and thickness), the ground, the proximity of components to the antenna, and so forth.

Hence, the AN043 will be copied from the Texas Instrument's reference design, and tested in the next chapter.

DN007 Reference design

The second reference design chosen from the selection guide [13] is the DN007, which the shape of the antenna is shown in Figure 9. The ideal characteristics of the DN007 reference design are described in the reference [15].

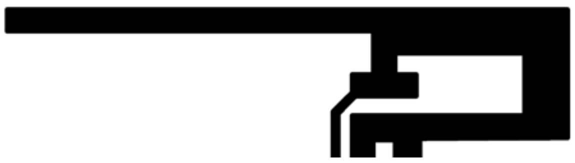


Figure 9. Shape of the DN007 antenna reference design

Moreover, another reason to choose this reference design is that the antenna is used in one of the applications examples of the transceiver chosen.

As the before reference design the performance of the antenna could change with the ground plane, the size of the board and more parameters. For these reason, the DN007 reference design will be copy and tested in the next chapter.

2.3.3.2.2. Directional antenna

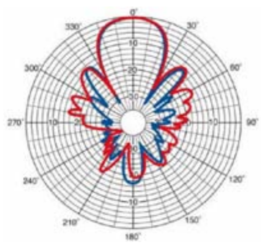
Directional antennas have several advantages such the antenna gain can be utilized to achieve a greater range distance between two devices, but the disadvantage is that the position of the transmitter and receiver must be known. However, in modular robots the distance between modules is small and it is not necessary to have a high gain of the antenna. Furthermore, the possible location of new modules is limited by the mechanic of the module, which defines the points of possible connection with other modules. Therefore, the problem is that the mechanics define more than one possible connection and made difficult to know in which connection is the new module.

The main characteristics of a directional antenna for modular robots are: low gain, medium directivity, small dimensions, low weight and operative frequency at 2.4GHz.

Searching in mouser[7] was impossible to find a commercial directional antenna with the characteristics defined before. For instance, we found a Yagi antenna with the characteristics shown in Figure 10 , or a corner antenna with the characteristics showed in Figure 11. Both antennas use an N female RF connector[8] to interconnect.

Figure 10. Commercial Yagi directional antenna

Gain 15 dBi
Panel Size 7” (17.8 Cm) X 7” (17.8 Cm)
Weight <1 Lb. (.473 Kg)
Prize 130 €



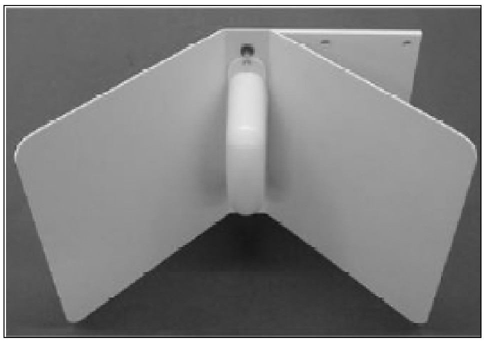
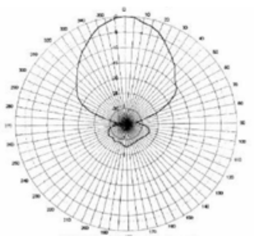


Figure 11. Corner directional antenna

Gain 14 dBi
Panel Size 7 x 7 in (17.8 x 17.8 cm)
Weight 2.0 lb (0.9 kg)
Prize 160 €



[7] Web site: http://dk.mouser.com/Passive-Components/Antennas/_/N-6j78n?P=1z0x01l

[8] Type N connector has consistent performance through 11GHz. Its applications are the termination of medium to miniature size coaxial cable. http://www.amphenolrf.com/products/typen.asp?N=0&sid=4DD9A38025ADE17F&

Furthermore, the radiation pattern of the Yagi antenna is not useful in the architecture of modular robots, due to the modules are close each other and the high secondary lobes of the antenna can easily detect signal when they should not. For instance, in a three modules chain, the module in the middle is communicating with the module in its right, but the module in its left is transmitting higher powerful signal. The secondary lobes of the directional antenna could detect this signal as the information transmitting by the module in its right and it will miss the communication with the right module. However, the radiation pattern defined in the corner antenna could be useful for the architecture of modular robots. The problem is the weight and sizes do not fit with the *Communication Board*.

As a result, we will search for an easy way to manufacture a directional antenna with smaller dimensions and weight, less gain and easily to interconnect with the PCB of the *Communication Board*. In the following points we describe the characteristic of the chosen directional antennas.

Biquad antenna

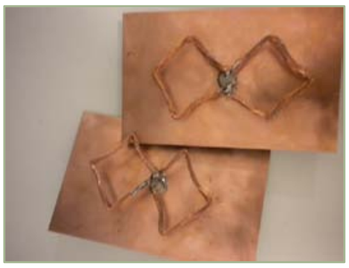


Figure 12. Biquad antenna

The final bachelor project of a student from the Polytechnic university of Valencia consisted in the implementation of a Biquad antenna [16]. We use his work to implement our antenna.

The Biquad antenna is a combination of λ/4 monopoles, which are joined to form two squares as shown in the Figure 12. To build a Biquad antenna is important to take into account the operative frequency and the material used to build it (wires, copper, etc). By theory, the wavelength for the operative frequency is defined as

$$\lambda = \frac{c}{f}$$

Equation 4. Wavelength of the operative frequency

Where λ is the wavelength, c is the speed of light and f is the operating frequency of the antenna. Therefore, using the Equation 4 we calculate the wavelength at 2.4GHz

$$\lambda = \frac{3 \cdot 10^8}{2{,}4 \cdot 10^9} = 0{,}125m$$

Therefore, the length of each monopole is

$$Monopole\ length = \frac{\lambda}{4} = 0{,}03125m$$

The thickness of the copper wire will be as thick as possible due to it has better adaptation.

Joining the four monopoles in two squares, we get an omnidirectional antenna. To get the directionality of the antenna, the two square dipoles will be building on a reflection platform. It is not necessary to modify the dimensions of the platform, because its purpose is creating an image of the radiation pattern, and its radiation will be in only one direction. But, the minimum size of the platform is related with the wavelength. Hence, since 4,5 · (λ/4) it could be considered like an infinite plane without error.

$$Length\,platform = 4{,}5 \cdot \frac{\lambda}{4} = 0{,}140625m$$

After the calculation, we implement the Biquad antenna which its dimensions are around 13 x 8.4cm and its weight is approximately 25gr. The connection is through an SMA connector[9], and the final prize of the antenna is approximately 7 Euros. Although, we improve some characteristic of the directional antenna described in the introduction, we must measure the gain and radiation pattern to estimate it as possible solution.

Circular Open-Waveguide antenna

The second directional antenna design is a circular waveguide, due to it is easy to build it, it has low gain and it works very well in close fields.

The circular waveguide is an excellent transmission line with low losses, which can be used in either frequency choosing the right dimensions. The waveguide is supplied with a SMA connector connected a monopole (λ/4) as shown in Figure 13, which it is introduced inside the circular waveguide. The length and the position of the monopole affect in the adaptation of the operating frequency.

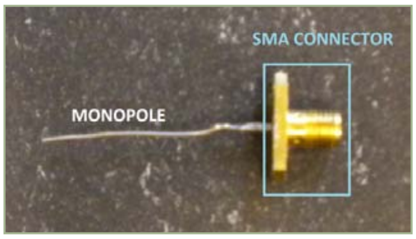


Figure 13. Monopole to supply the waveguide

The diameter of the tube should be enough to propagate the fundamental mode, but it should attenuate the higher modes [17]. To exits signal propagation inside the waveguide, the configuration of the electronic and magnetic fields have to follow some considerations. There are different possibilities of configuration, called modes. The modes are defined according to the direction of the electric and magnetic field of the electromagnetic wave respect to the propagation direction. Hence, there is "transversal electromagnetic mode"(TEM), where the electric and magnetic field is perpendicular to the direction of the propagation wave, the "transversal electric mode"(TE), where the electric field is perpendicular to the propagation direction, and the "transversal magnetic mode" (TM), where the magnetic field is perpendicular to the propagation direction [18].

[9] SMA connectors are semiprecision, subminiature units that provide electrical performance from DC to 18GHz.

The propagation could happen with low losses if the operating wavelengths are lower than the cutoff wavelength. If the wavelength is higher, the losses will be higher. For each propagation mode, there is a different cutoff wavelength. In circular waveguides the limit is determined by the inner diameter of the waveguide. The mode with the lower cutoff frequency which it is propagated in the waveguide is the "electric transversal 1,1" (TE_{11}).

The relationship between frequencies, wavelength and propagation modes are shown in Table 3.

Mode	Cutoff wavelength	Cutoff frequency
TE_{11}	$\lambda_c = 1{,}7065 \cdot D$	$f_c = \frac{c}{\lambda_c} = \frac{3 \cdot 10^8 m/s}{1{,}7065 \cdot D}$
TM_{01}	$\lambda_c = 1{,}31 \cdot D$	$f_c = \frac{c}{\lambda_c} = \frac{3 \cdot 10^8 m/s}{1{,}31 \cdot D}$
TE_{21}	$\lambda_c = 1{,}03 \cdot D$	$f_c = \frac{c}{\lambda_c} = \frac{3 \cdot 10^8 m/s}{1{,}03 \cdot D}$

Table 3. Modes of the waveguide

The best diameter for the waveguide is when the desired operating frequency works between the TE_{11} and TM_{01} modes. The operative frequency should be always below the cutoff frequency in the TE_{21} mode. Hence, only the TE_{11} mode is propagated with low attenuation, and the others modes cannot propagate (or they will propagate with high attenuation).

The advantage allowing only one mode of propagation is that the excitation element (monopole) of the waveguide is easier to calculate. The dimensions of the tube are calculated according with the operative frequency. Using the Equation 4 the wavelength for the operating frequency is $0{,}125m$.

To estimate the diameter of the circular waveguide, the minimum and maximum diameter for the range between 2,28GHz and 2,52GHz (the 5% above and below the operating frequency) will be calculated. However, the constraints are:

- The maximum diameter should not be as bigger that the maximum operative frequency is higher than the cutoff frequency in the TE_{21} mode.
- The minimum diameter should not be as smaller than the minimum operative frequency will be lower than the cutoff frequency in the TE_{11} mode.

According to that, the maximum and minimum diameters of the waveguide for 2,4GHz are:

$$D_{min} = \frac{c}{1{,}71 \cdot f_l} = \frac{3 \cdot 10^8 m/s}{1{,}71 \cdot 2{,}28 \cdot 10^9 Hz} = 0{,}077m = 7{,}7cm$$

$$D_{max} = \frac{c}{1{,}71 \cdot f_h} = \frac{3 \cdot 10^8 m/s}{1{,}03 \cdot 2{,}52 \cdot 10^9 Hz} = 0{,}115m = 11{,}5cm$$

To calculate the wavelength inside the waveguide (λ_g), the following formula is used:

$$\lambda_g = \frac{\lambda_0}{\sqrt{1 - \left(\frac{\lambda_0}{\lambda_{c11}}\right)^2}}$$

Equation 5. Wavelength inside the circular waveguide

where λ_0 is the operating wavelength and λ_{c11} is the cutoff wavelength in the TE_{11} mode.

The signal emitted by the monopole, must to arrive to the bottom of the waveguide with amplitude cero. In this case, the signal is totally reflected with a coefficient reflection of -1, which implies a 180° change phase, and all the generated waves are added in phase to transmit the maximum possible power. This happens when the active element is placed in the first maximum, at $\lambda_g/4$. If the monopole is not placed in the first maximum, there is not adaptation and the reflections waves are not added in phase, generating a wave that not transmits the maximum power. Therefore, the ideal dimensions of the waveguide are shown in Figure 14.

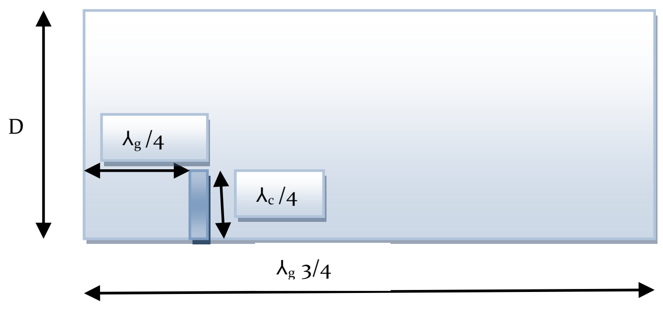


Figure 14. Dimensions of the waveguide

The easiest and cheapest way to implement the circular waveguide is using a can. We found two different size of can which can fit as a directional antenna. The diameter, the wavelength of the TE_{11} mode, the wavelength inside the can λ_g using the Equation 5 and the theoretical waveguide length are shown in Table 4.

Type of can	Diameter	Wavelength λ_{c11}	Wavelength λ_g	Theoretical Waveguide Length	Waveguide Length
Coke	0,065m	0,111m	-	-	0,097m
Corn	0,085m	0,145m	0,245m	0,183m	0,085m

Table 4. Parameters of the can directional antenna

The coke can do not fit between the operative diameter calculated, but its shape is not ideal, the bottom of the can is not flat and the bright painting inside could change the performance of the antenna, for these reasons we will test it in the next chapter if it is suitable for our system.

The length of the corn wavelength does not fit with the theoretical calculated. Reduce the length of the waveguide generate that the monopole is close to the open-aperture, and as a consequence it not possible to generate the λ_g wavelength properly radiating in the free space. The propagation in free space has three effects: attenuation of the signal, the signal does not travel in straight line and multipath occurs. However, the crumples influence in the performance of the antenna, consequently we will measure if this influence positively or negatively for the purpose of the antenna. The implementations of both circular wavelengths are shown in Figure 15 and Figure 16.

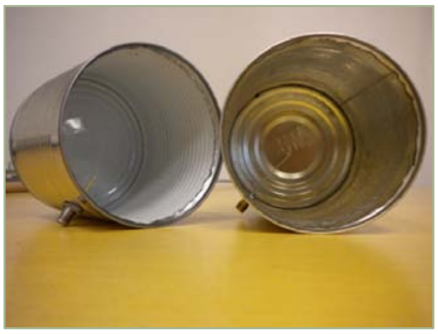


Figure 15. Corn can design

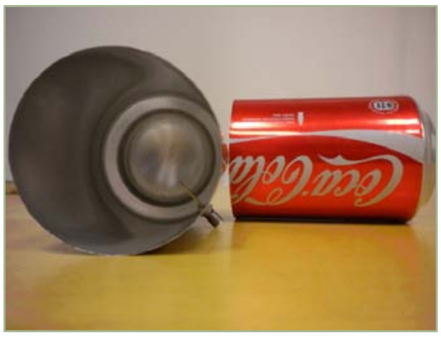


Figure 16. Coke can design

As a conclusion, the dimensions of the coke antenna are 6.5 x 9.7cm and 5gr, and the dimensions of the corn antenna are 8.5 x 8.5 cm and 15gr of weight. The prize of both designs is around 4 Euros. As the previous directional design, we improve the weight and prize of the directional antenna comparing with the commercial designs, but we must measure its performance with the defined size in the next section to estimate if could be a possible solution for the directional *Communication Board* prototype.

2.4. Implementation of the *Communication Board*

Once the components for the *Communication Board* were selected, we designed and implemented the first prototype.

The Eagle 4.1 software was used to design the circuit. Eagle is an application to design schematics and printed circuits. This software is free which have fraught limitation like the size of the boards. Nevertheless, this is not a limitation for us, since we want to design a board with similar size as the *General Board*. The *Communication Board* will be placed on the *General Board*, which has LEDs useful for debugging, but if its size is bigger, it will be difficult to use them.

In the laboratory of the Maersk Mc-Kinney Moller Institute is possible to made one or two layers boards. Hence, we chose two layers: the bottom layer is defined as the ground plane and to place the female connectors to interconnect with the *General Board*, and the top layer to place the transceiver and the antenna. Some VIAS (holes) will be done to interconnect both layers.

The characteristics of the substrate to implement the circuit are important. For instance, the thickness of the substrate influence on the bandwidth, the resonant frequency and the resonant patch length. An increasing on the thickness of the substrate, increase all these parameters. The thickness (h) of the substrate used is 0,8mm and the permeability (μ_r) is 35 µm. The thickness is bigger than the recommended in the DN007 reference design (0.611mm), so we should check the performance of these parameters in the implemented design.

We designed and implemented two boards, one with an omnidirectional antenna and the other with a directional one. The only difference between both circuits is the connection to the antenna. The schematic of the first prototype is included in Appendix E. The implementation of the top and bottom layer of the omnidirectional *Communication Board* is shown in Figure 17 and Figure 18. The implementation of the top and bottom layer of the directional *Communication Board* is shown in Figure 19 and Figure 20. However, we should take into account that the “artisanal” process to manufacture each board, constraint the assumption that all the implemented boards have the same performance.

Furthermore, the performance of the antenna depends on the ground plane, which it is different in each board (size, amount of solder, etc.), thus the performance of the *Communication Board* implemented is similar between each other, but at the same time different. The prize of each board is around 13 Euros.

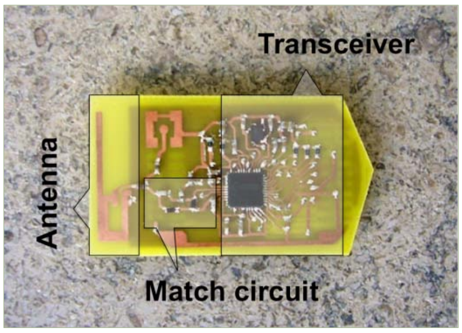


Figure 17. Top layer of the Omnidirectional *Communication Board*.

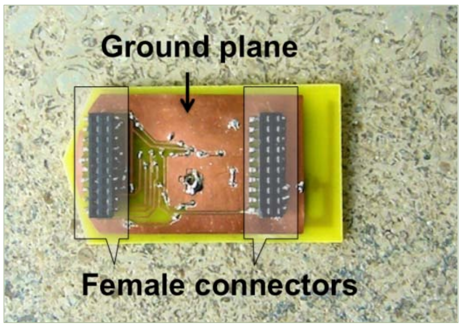


Figure 18. Bottom layer of the Omnidirectional *Communication Board*.

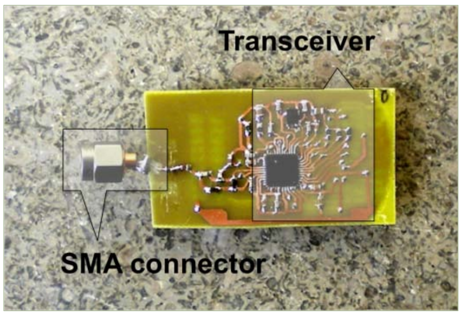


Figure 19. Top layer of the directional *Communication Board*.

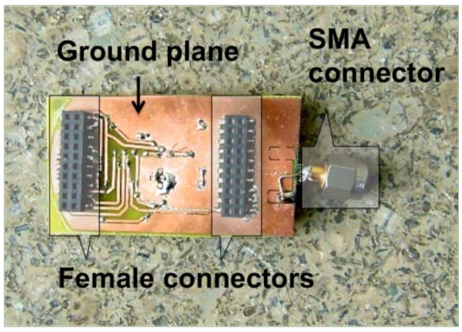


Figure 20. Bottom layer of the directional *Communication Board*.

2.5. Conclusion

We studied the typical parameters used in RF wireless communication in order to design the first "*Communication Board*" prototype to be tested for the objective of this thesis. Based in this study, we chose the CC2420 transceiver among different commercial devices, due to it operates in the 2.4GHz ISM band, includes DSSS, RSSI and LQI techniques, can fulfill a standard protocol if we need, has good sensitivity, high baud rate, need few external components to be implemented, and use SPI interface necessary to connect with the microcontroller of the *General Board* implemented in The Mærsk Mc-Kinney Møller Institute. For the antenna, we propose two known omnidirectional designs and two simple directional designs, in order to measure them in the following chapter to choose the suitable antenna design for our system.

Finally, we implemented the first prototype for the *Communication Board*, which it is divided in two designs: one with an omnidirectional antenna and the other one to be connected with a directional antenna.

3. Hardware measurements

3.1. Introduction

The hardware of the system is divided in the microcontroller of the "*General Board*" and, in the *Communication Board* the transceiver with all the external components and the antenna. The antenna could be measure independently of the rest of the system. However, the hardware of the *Communication Board* needs the microcontroller to be tested.

Therefore, in the following points we will measure the performance of each antenna designed before in order to choose the suitable one for our system, the test to check the entire hardware of the *Communication Board* connected to the microcontroller and finally the discussions about all the measurements.

3.2. Antennas

According with the kind of antenna, directional or omnidirectional, different parameters will be measured. The measurements of an antenna should be ideally performed in an anechoic chamber, but the measurements in the laboratory environment will give us a relative result of the antenna's performance in the real system. Therefore, in the following points will be analyze the measurements of each kind of antenna design in the RF laboratory.

To measure the different parameters of the antenna, the Agilent E5071C Network Analyzer will be used. Before measure anything, the Network analyzer will be calibrate to get a good performance of the instrument. Calibration means measure the attenuation and loss generated by the cables used to connect the antenna to the network analyzer, to compensate them. In this way, when the antenna is measured the cables will not affect the measurement.

After calibrate, the marker will be placed in the interesting point (2,4GHz), and we can measure the features of each antenna.

3.2.1. AN043 Reference Design

Adaptation

The antenna is connected to the Network Analyzer through a RF cable with a SMA connector.

The reflection coefficient, also called S11 parameter, should be 0 or close to infinite in dBm to do not produce reflections and transmit the entire signal.

The adaptation of the antenna is shown in Figure 21. The reflection coefficient at 2.4 GHz is -9 dBm, as a result the antenna is not well adapted.

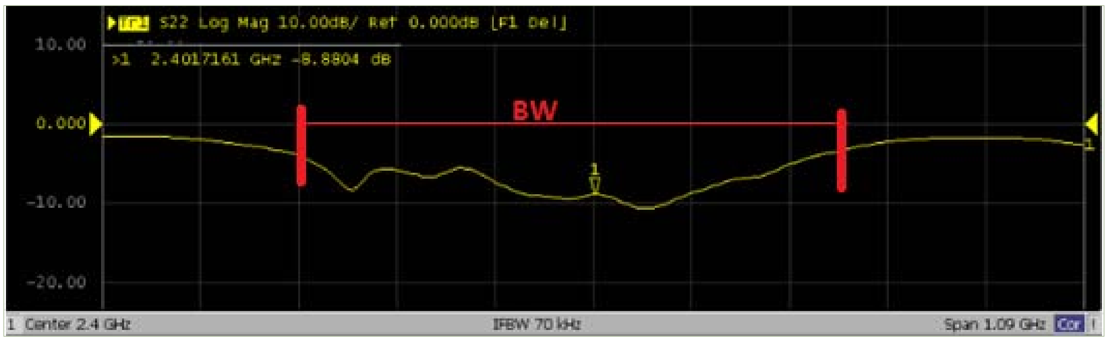


Figure 21. Reflection coefficient of the AN043 reference design

On the other hand, the bandwidth of the antenna is wide (from 2.1 GHz to 2.7 GHz), which could generate interferences and noise in the system if there are any signal between these frequencies in the environment.

The good characteristic of this antenna is that the power received in 4,8GHz is 0 dBm, meaning that the harmonic of the operative frequency does not generate interferences.

Radiation pattern

Depending on the different types of movement the modular robots can perform, it will be interesting measure the power received in different points of the space.

However, we assume that the initial experiments in RF for modular robots are 2D. According to that, we measure the radiation pattern in the azimuth plane (horizontal).

To measure the radiation pattern one antenna is fixed while we move the other antenna from -90° to +90° in 10° steps. The measurements are shown in Table 13 in Appendix A, and the radiation pattern is shown in Figure 22.

As we expected, the radiation pattern of the antenna is not a constant line, and does not radiate the same strength power in all directions of the space as shown in Figure 23.

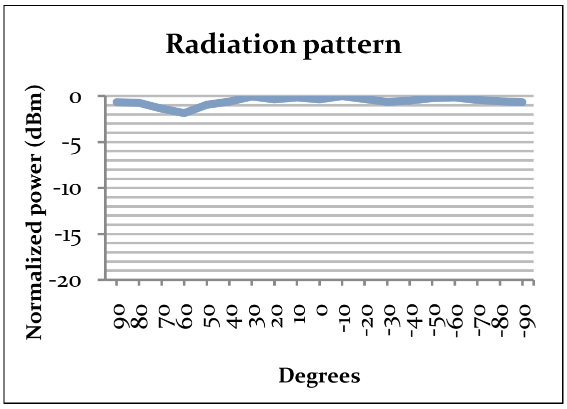


Figure 22. Radiation pattern of the AN043 reference design

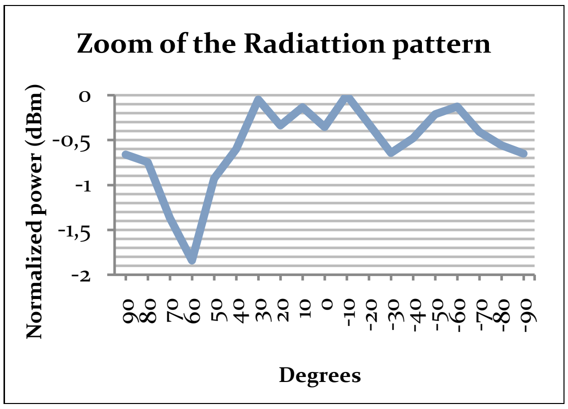


Figure 23. Zoom in the radiation pattern of the AN043 reference design

Gain

As was said in the introduction, the gain of an antenna does not measure the amplification of the signal receiver in the transmitter, it measure how much power concentrate in a direction respect an isotropic antenna.

The gain of an ideal-omnidirectional antenna is 0 dBm, because it propagates the same power in all the space. However, as it shown in the radiation pattern of this omnidirectional antenna is not a constant line, thus it has low gain around 2 dBm as is shown in Figure 23.

Conclusion

The advantage of this reference design is the not interference for harmonics signals. The disadvantages are the not well adaptation and the wide bandwidth that could generate interferences in the system.

The non-ideal radiation pattern of the antenna could be an advantage or disadvantage depending on the purpose of the system, as we will see later.

3.2.2. DN007 Reference Design

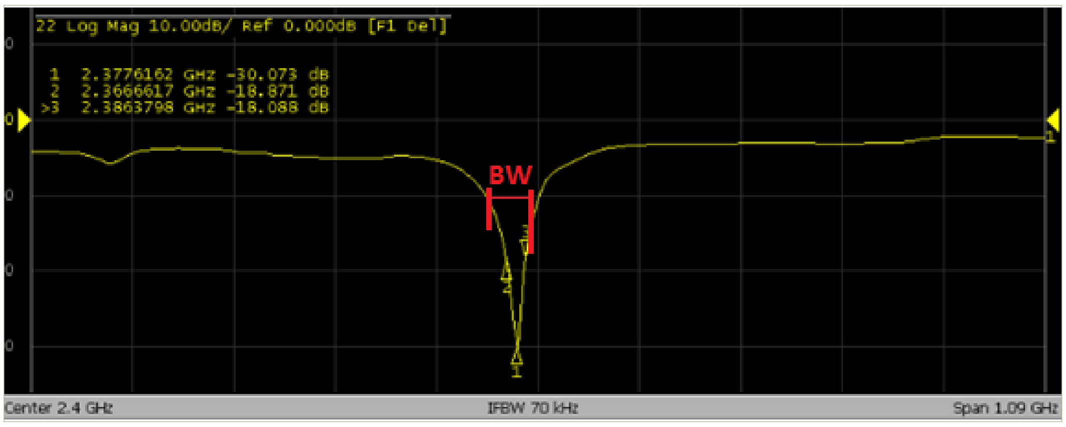
Adaptation

The board was designed with a female SMA connector to interconnect it with the Network Analyzer. The reflection coefficient measured is shown in Figure 24.

The adaptation of the antenna at 2.37 GHz is -30 dBm, which it is very good and it is close to the operative frequency, 2.4 GHz. To achieve the required frequency, we should tune. However in PCB antennas is very difficult remove or add more copper in the board.

The bandwidth is narrow (between 2.36 GHz and 2.38 GHz) reducing the possibility of interferences generated by other frequencies.

Furthermore, the power received in 4,8 GHz is 0 dBm, meaning that the harmonic of the central frequency does not generate interferences.


```
22 Log Mag 10.00dB/ Ref 0.000dB [F1 Del]

  1  2.3776162 GHz -30.073 dB
  2  2.3666617 GHz -18.871 dB
 >3  2.3863798 GHz -18.088 dB
```

Figure 24. Reflection coefficient of the DN007 Reference design

Radiation pattern

Following the same assumption than in the previous antenna, we measure the radiation pattern measured in the azimuth plane, and with the same methodology. In Table 14 of Appendix B is shown the measurements and in Figure 25 is shown graphically the radiation pattern.

As we expected and as the previous omnidirectional antenna measured, the radiation pattern of the antenna is not a constant line, and does not radiate the same strength power in all directions of the space.

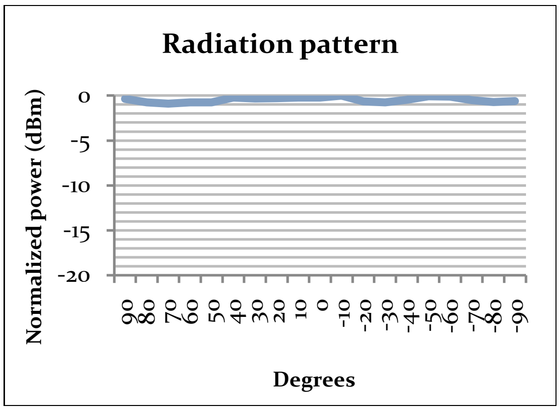


Figure 25. Radiation pattern of the DN007 reference design

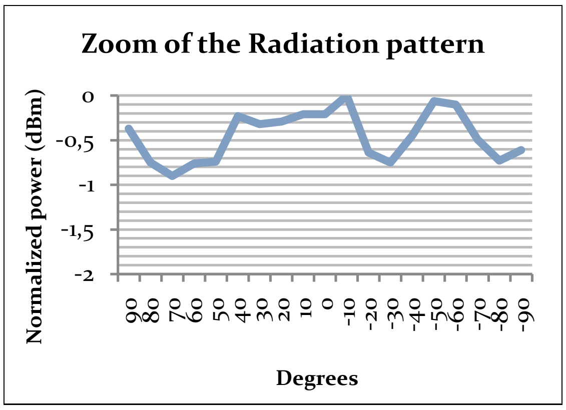


Figure 26. Zoom in the radiation pattern of the DN007 reference design

Gain

As was said in the analysis of the previous antenna, the gain of an ideal-omnidirectional antenna is 0 dBm, but Figure 26 shows that the antenna has low gain, around 1 dBm.

Conclusion

The advantages of this antenna are the well adaptation, the narrow bandwidth and the small gain. As was said before, the non-ideal radiation pattern of the antenna could be an advantage or disadvantage depending on the purpose of the system.

3.2.3. Biquad

Adaptation

The antenna is connected to a female SMA connector to interconnect with the Network Analyzer.

The reflection coefficient parameter of the Biquad was not as we expected. To modify the adaptation we can change the dimensions of the reflection platform or the length of each dipole. In the RF laboratory, we did not have the tools to cut the platform; therefore we modified the length of the dipole to achieve the best adaptation possible.

The final dimensions of the antenna are 2.1 cm for the Dipole length and 13 cm x 8.4 cm for the Platform length. The reflection coefficient finally achieved is shown in Figure 27.

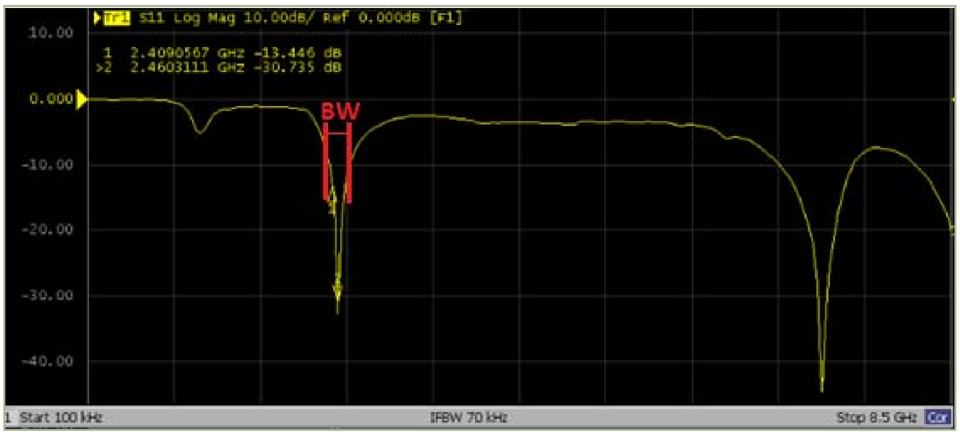


Figure 27. Adaptation Biquad antenna

The reflection coefficient at 2.4 GHz is -14 dBm, although at 2,46 GHz the reflection coefficient is -30 dBm, which is very good. At 7.6 GHz the reflection coefficient is -45 dBm, which it is close to the third harmonic, which could generate interferences in the system.

The bandwidth is narrow (between 2.40 GHz and 2.5 GHz), reducing the possibility of interferences generated by other frequencies.

Radiation pattern

To measure the radiation pattern is necessary two identical antennas (characteristics, shape, dimensions, etc) connected to different ports of the network analyzer, and one in front of each other, as shown in Figure 28.

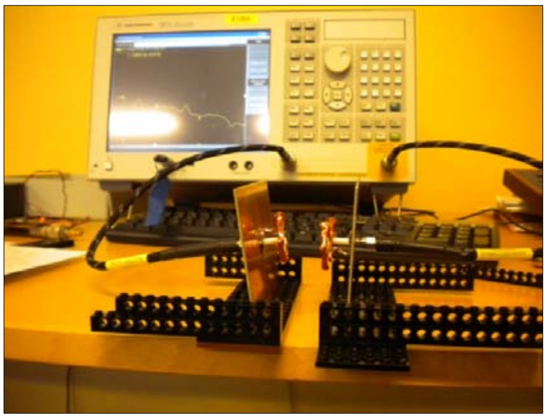


Figure 28. Configuration to measure the radiation pattern of two Biquad antennas.

The initial experiment shows quickly that the antennas at 2.4GHz have directivity, which means difference of the power received when the antennas are pointing each other and when not. In Figure 29 is shown the received power when the antennas are pointing each other, -10 dBm, and in Figure 30 is shown that the power received when the antennas are pointing to the roof is -26 dBm. Hence, there is a difference of 16 dBm.

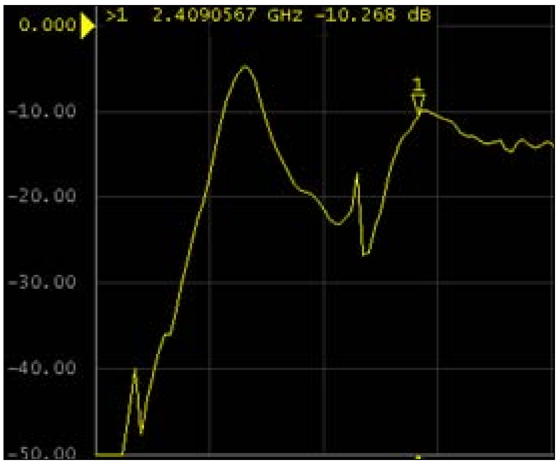


Figure 29. Power received when both Biquad are pointing each other

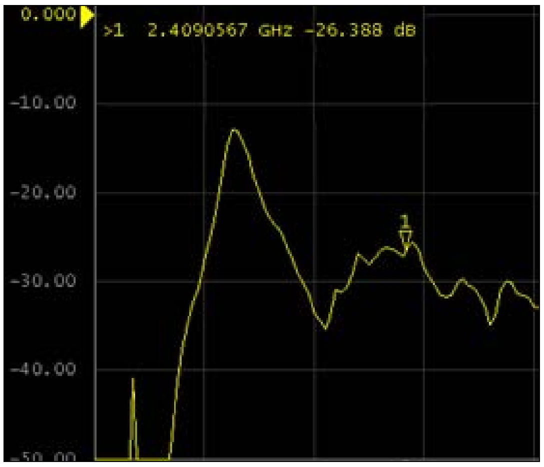


Figure 30. Power received when both Biquad are pointing to the roof

Once, we check that the antenna has directivity, we measure the radiation pattern. To get an invariable radiation pattern, the Fraunhofer region is calculated. The far field effect could affect in the undesired reflections. To avoid these reflections, the antennas should be placed a distance defined by Fraunhofer Region by the following formula:

$$r > \frac{2D^2}{\lambda}$$

Equation 6 . Fraunhofer region

where r is the minimum distance between two antennas, D is the diameter of the antenna and λ is the operation lambda. Therefore, the minimum distance between antennas should be

$$r > \frac{2 \cdot 0{,}154^2}{0{,}125} > 0{,}37m$$

However, the limitation of the cables length of the Network Analyzer, make that the maximum distance between both antennas should be 7cm. We placed the antennas at 7cm between each other knowing that the measurements will not be accurate due to the reflections.

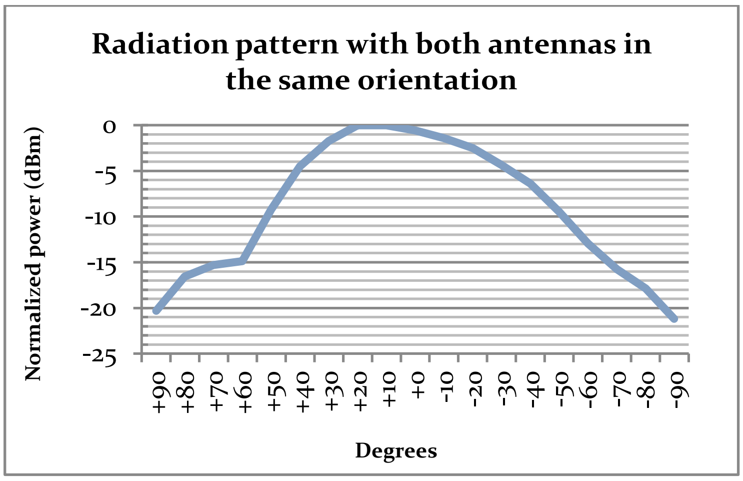


Figure 31. Radiation pattern with the same polarization for both antennas

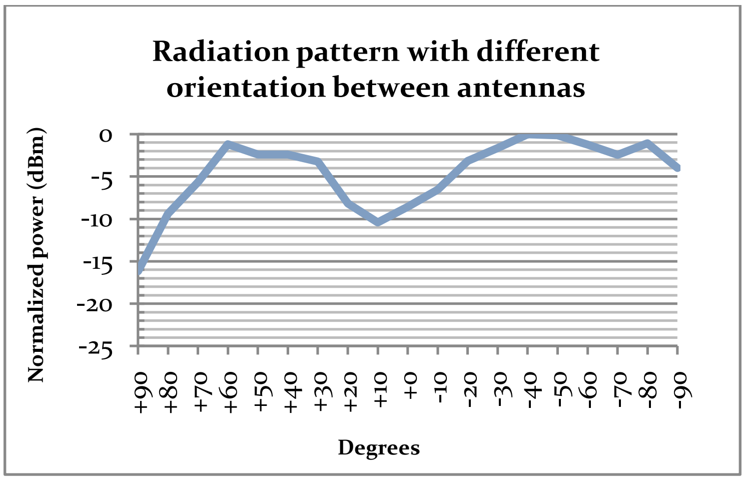


Figure 32. Radiation pattern with different polarization between the antennas

To measure the radiation pattern, one of the antennas is fixed and the other one is turning from -90° to +90° in 10° steps, which each step will be measured 5 times. We measure the radiation pattern when both antennas have the same polarity (both horizontal position) and when the antenna fixed have different polarization (vertical position), while the moving antenna keeps the horizontal position.

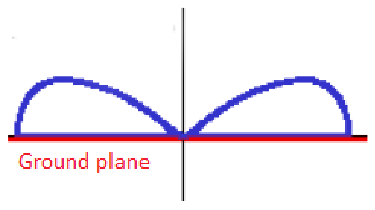


Figure 33. Radiation pattern of a dipole

In Table 15 and Table 16 in Appendix C is shown the measurements of the radiation pattern when both antennas have the same polarization and not. Figure 31 is shown the radiation pattern when both antennas have the same polarization and Figure 32 is shown the representation of the radiation pattern when both antennas have different polarization.

The measurement is what we expected. In Figure 31, the radiation pattern when both antennas have the same polarization is good. There is a difference of 20 dBm when the angle changes 90°. In the Figure 32, the radiation pattern is like the radiation pattern of a dipole with ground plane as shown in Figure 33.

Therefore, when one antenna is fixed and the other one is moving from -90° to +90°, one square dipole describe the power received by the two dipoles with ground plane. However, there are 20 dBm of different at 10° when we change the polarization of the antenna, which means that the antenna is not good in a system where the orientation of the antenna is not fixed.

Directivity

To measure the directivity, the radiation pattern measurements are used with the following formula:

$$D = \frac{4\pi}{\iint_{4\pi} t(\theta, \emptyset) d\Omega} = \frac{4\pi}{\Omega_e}$$

Equation 7. Directivity of an antenna

Where $\Omega_e$is the equivalent solid angle.

$$\Omega_e = \iint_{4\pi} t(\theta, \emptyset)\, d\Omega \approx \Delta\theta_{-3dB}\Delta\emptyset_{-3dB}$$

Equation 8. Equivalent solid angle

where $\Delta\theta_{-3dB}$ is the width of the beam at -3 dBm in θ, and $\Delta\emptyset_{-3dB}$ is the width of the beam at -3 dBm in ϕ.

From the measurements of the radiation pattern in Table 15, the degree where the beam width goes down 3 dBm is taken and shown in the following table:

Degree	$\boldsymbol{\Delta\theta_{-3dB}}$
+30°	4.49
-20°	2.5

The measurements are related with the value of the maximum radiation, in this case is related to +10°.

Using the Equation 7 and Equation 8 we calculated the directivity:

$$\Omega_e = \left(50 \cdot \frac{\pi}{180}\right) = 0.872 rad \qquad D = \frac{4\pi}{\Omega_e} = 14.4$$

Hence, the directivity in logarithms units is:

$$D = 10\log(14.4) = 11.58dB$$

Therefore, the antenna receives 14.4 more power in its maximum direction than an omnidirectional antenna.

Gain

To calculate the gain the Friis formula will be used, which predict the line of sight (LOS) between communications links:

$$\frac{W_{rec}}{W_{ent}} = \frac{G_{tx} \cdot G_{rx}}{\left(\frac{4\pi r}{\lambda}\right)^2} \cdot C_a \cdot C_p \cdot C_m$$

Equation 9. Friis Formula

where $G_{tx} \approx G_{rx}$, W_{rec}/W_{ent} is the maximum level of signal when the antennas are pointing each other, and $C_a \cdot C_p \cdot C_m \approx 1$.

Using the Equation 9 and the maximum radiation values of the Table 15, we get that at 10° is the maximum level of signal, which is -11,109. Thus, $10log\left(\frac{W_{rec}}{W_{ent}}\right) = -11,109$ and $\left(\frac{4\pi r}{\lambda}\right)^2 = \left(\frac{4\pi}{0,125}\right)^2 = 10106,47$ which in logarithms units is $10\log(10106,47) = 40dB$.

Hence, we find the gain from the Equation 9 like:

$$10log\left(\frac{W_{rec}}{W_{ent}}\right) = 2 \cdot G_{ANTENNA}(dB) - 10log\left[\left(\frac{4\pi r}{\lambda}\right)^2\right]$$

The gain of the antenna is calculated as:

$$G_{ANTENNA}(dB) = \frac{-11,109 + 40}{2} = 14,44dB$$

As we expected the gain result is approximately the same that the directivity results.

Conclusion

The manual process to change the orientation of the antenna in 10° steps and the shorter distance between the antennas to calculate the radiation pattern generates some unexpected peaks (reflections) in the measurements. Hence, all the measurements are an approximation of the real radiation pattern.

The advantages of this antenna are that modeling the reflection platform and the dipoles length of the antenna we can modify the adaptation and bandwidth. Hence, we can easily tune the characteristics of the antenna achieving the values necessaries for the system.

The main disadvantage is that the antenna does not receive the same strength power changing its polarization, limiting its position in the system to be useful. Furthermore, the gain of the antenna is higher than we want.

3.2.4. Waveguide

Adaptation

Although, the waveguide antenna building with a coke and corn can do not fit with the theoretical calculation, which the minimum diameter and length of the can should be 7.7cm and 18cm respectively, these antennas will be measured.

The supply element of both circular waveguide is connected to the male SMA connector of the Network Analyzer.

The reflection coefficient measured for the coke can is shown in Figure 34 and for the corn can is shown in Figure 35.

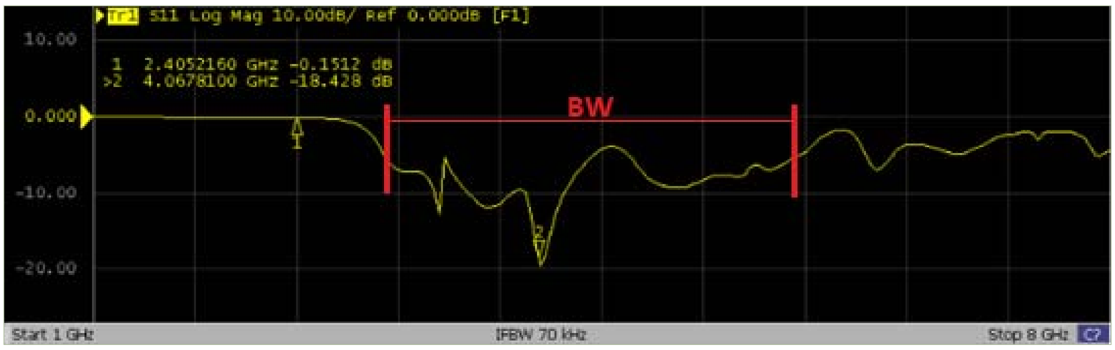


Figure 34. Coefficient Reflection of the coke can antenna

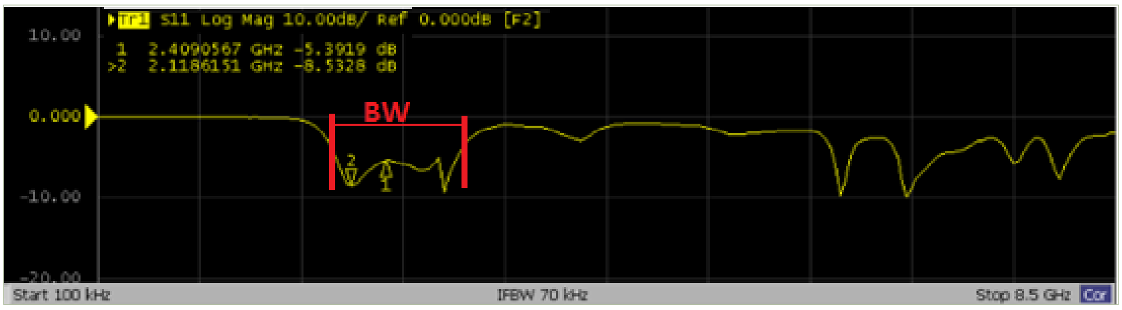


Figure 35. Coefficient Reflection of the corn can antenna

The reflection coefficient at 2,4 GHz in the coke can is almost 0 dBm, which means that the entire signal is reflected and the signal is not transmitted. The reflection coefficient in the corn can is -5 dBm, which it is better than the coke can but not good enough. Furthermore, in the coke can the harmonic is close to -17 dBm generating interferences, while the harmonic in the corn can is close to 0 dBm.

The adaptation depends on the length of the wire inside the can and the connection point. We tried to do the wire longer and shorter, changing the connection point and the length of the can, but the reflection coefficient never was below -15 dBm. The material of the can also influence in the performance of the antenna, like the bright painting and the crumples. Hence, it is very difficult to find the optimal open- waveguide implementation for the adaptation.

On the other hand, the bandwidth of the coke antenna is wide, 2.8 GHz, while the bandwidth of the corn antenna is also wide, 1.2 GHz, but narrower than with the first one.

Radiation pattern

To measure the radiation pattern is necessary two identical antennas (with the same characteristics, shape, dimensions, etc) connected to different ports of the Network Analyzer, and one in front of the other.

One of the antennas is fixed and the other one is routing from -90° to +90° in 10° steps as shown in Figure 36. Each step will be measured 5 times.

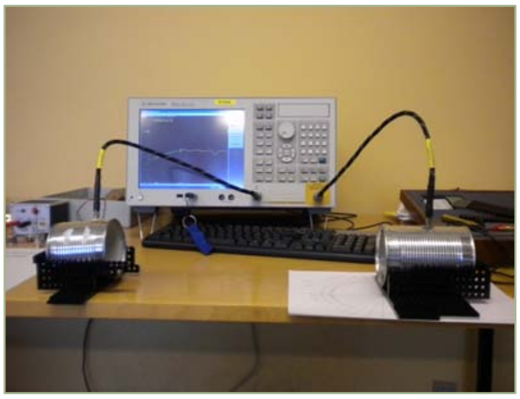
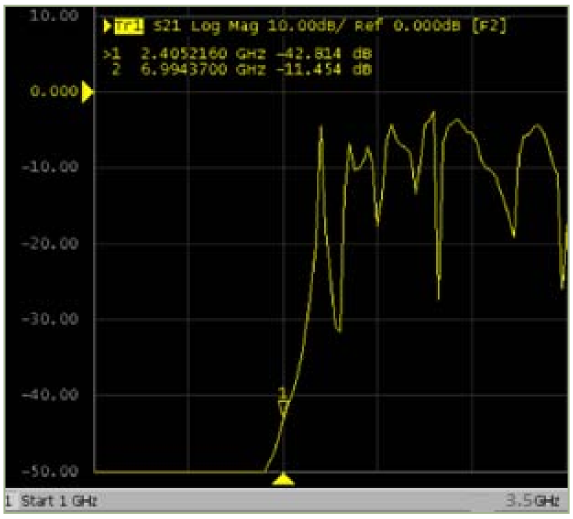


Figure 36. Way to measure the radiation pattern of the waveguide

As in the previous Biquad test, the initial experiment is shown quickly that the antennas have directivity at 2.4GHz. In Figure 37 is shown that the power received when the coke can antennas are pointing each other at 1cm of distance, and in Figure 38 is shown the power received when both coke can antennas are pointing the roof. The performance of the antenna is very good, there is 27 dBm of difference when the antennas are pointing each other and not, but the range values are very low (-42 dBm pointing each other and -69 dBm not pointing). Therefore, we will not use the coke can antennas.

In Figure 39 is shown the power received when the corn can antennas are pointing each other at 3 cm of distance (-10 dBm), and in Figure 40 is shown the power received when both corn can antennas are pointing the roof (-47 dBm), where there is a difference of 37 dBm. Because of the performance of this antenna is good, in the following points we only measure the performance of the corn can antenna.



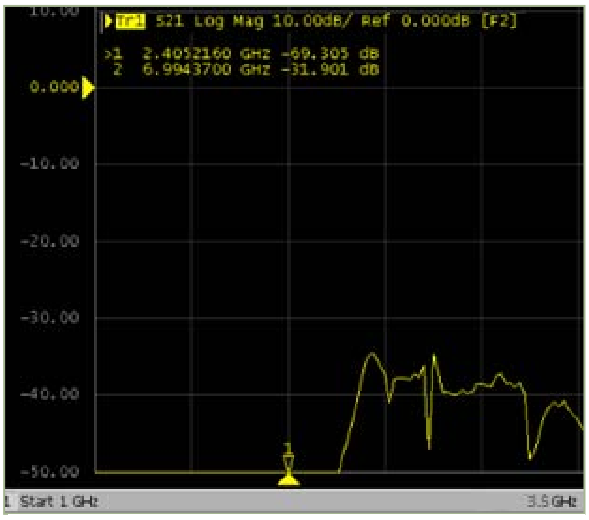
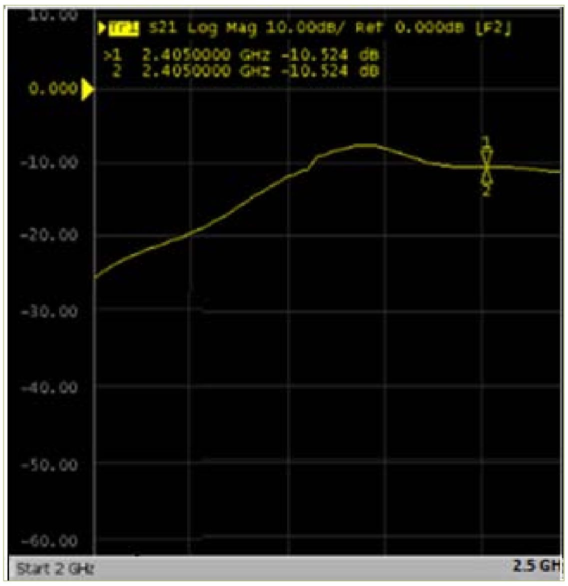


Figure 37. Power received by the coke can when both antennas are pointing each other

Figure 38. Power received by the coke can when both antennas are pointing to the roof



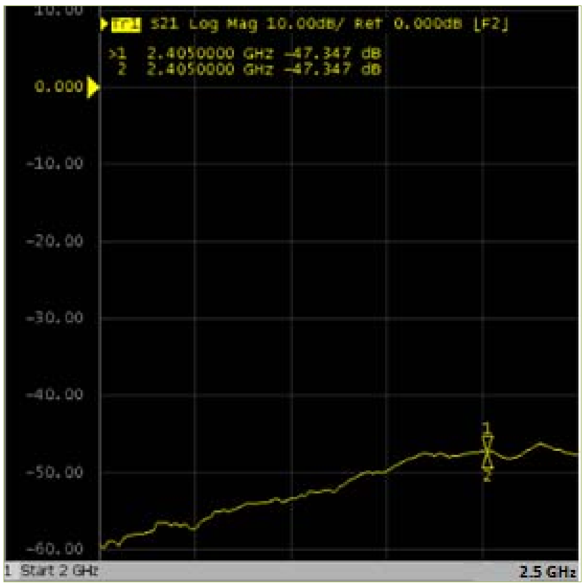


Figure 39. Power received by the corn can when both antennas are pointing each other

Figure 40. Power received by the corn can when both antennas are pointing the roof

In Appendix D, the Table 17 shows the measurements of the radiation pattern, when the distance between the antennas is 13 cm, which it is the maximum distance that the cables of the Network Analyzer provides. The Figure 41 is shown the representation of the radiation pattern.

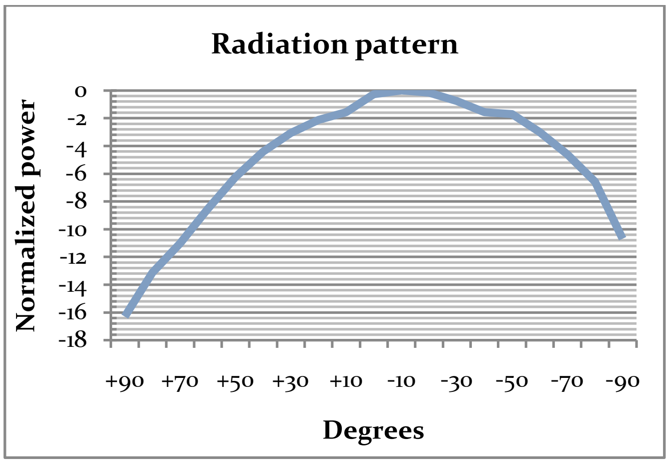


Figure 41. Radiation pattern of the circular waveguide

The difference when both antennas are pointing each other and when the antennas are in perpendicular is approximately 16 dBm. We tested that the difference when both antennas are pointing each other and when they are in perpendicular at 5 cm of distance is 30 dBm. Therefore, the pointing difference increases reducing the distance between antennas.

Directivity

The directivity in a circular open-waveguide aperture is [19]:

$$D = \frac{4\pi}{\lambda^2} \cdot A_p = \frac{4\pi}{\lambda^2} \cdot \pi \cdot a^2$$

Equation 10. Directivity in an open-circular waveguide

where A_p is the aperture area, which in this case is the area of a circle with radius a. The directivity of the corn circular waveguide using the Equation 10 is

$$D = \frac{4\pi}{\lambda^2} \cdot \pi \cdot a^2 = \frac{4 \cdot \pi^2 \cdot (0{,}0425)^2}{(0{,}125)^2} = 4{,}56$$

$$D = 10 \log(4{,}56) = 6{,}58 dB$$

Gain

The gain of a circular waveguide antenna is G=eD [12]. For most aperture antennas the ohmic losses are very small, and

e≈1 and G=D

Therefore, the gain was calculated with the directivity before.

Conclusion

The manual process to change the orientation of the antenna in 10° resolution to calculate the radiation pattern generates some unexpected peak measurements. Hence, all the measurements are an approximation of the real radiation pattern.

The advantage of this antenna is the low gain.

The adaptation and bandwidth change with the length of the dipole and the connection point, but is difficult to tune. The connection point was done with a drill, which we tried unsuccessfully to accurately do the hole in the exactly point. The crumples in the can also influence in the measurements. Therefore, the disadvantage of the circular waveguide is the influence of the material in the performance of the antenna, influencing in the bad adaptation and the wide bandwidth.

3.3. Module general test

We developed an application that runs a general test to check if all the *Communication Board* hardware is working as expected. This program gave us a fast and reliable way to check all the new built modules. The *General Board* has an Atmel AT91SAM7256 microcontroller, which was tested separately, two push buttons and four LEDs used for debugging process and, in this case, the possible errors are shown using a combination of three LEDs.

Therefore, the microcontroller runs a test to check SPI interface, if the signal generation to transmit is well generated, if there is any short circuit on the board and, for instance, the TI CC2420 device in general. The test is divided in four steps: first, some LEDs are switched on to guarantee that the board is working; second, starting with the boot signal, the application configures, initializes and starts the SPI bus of the microcontroller; third, after setting up the SPI bus, the CC2420 Control registers are configured as well as all the interfaces and variables needed to send a short testing message over the radio; and fourth, if all goes well, a short testing broadcast message is sent every second over the radio, to check if the internal state machine of the transceiver is working.

3.4. Discussion

We measured the performance of each antenna to discuss in this section the results and the possible application in the system. In Table 5 is shown the summary of the features of each antenna measured.

Antenna	Adaptation	Bandwidth	Directivity	Gain
AN043	-9 dBm	600 MHz	-	2 dBm
DN007	-30 dBm	20MHz	-	1 dBm
Biquad	-14 dBm	100MHz	11.58 dBm	12.4 dBm
Corn waveguide	-5 dBm	1200MHz	6.58 dBm	6.58 dBm

Table 5. Features of each antenna design

For global communication, the omnidirectional antenna is the best solution, due to the wide range. Between both omnidirectional antennas measured, the DN007 reference design has better features (higher adaptation at 2.4 GHz, narrower bandwidth and lower gain) than the AN043 reference design. A directional antenna cannot provide global communication, due to its directional condition that only can transmit signal in one direction of the space.

In local communication, an omnidirectional antenna could be a good solution. The power transmitted by the antenna will select the wide range of communication: the higher power transmitted the longer wide range. A directional antenna could be used, depending on the architecture of the robot. If the module only has one possible connection and the directional antenna is placed there, it will exit local communication. However if the module have more than one possible connection and the directional antenna is placed in one of these connections, the module could have local communication with the neighbor in front of the directional antenna, but it will be difficult to communicate with others modules in the remains module connections.

In localization the directional antenna is useful to detect if there is a new module to connect in the robot. But, as was said before, we should place a directional antenna in each possible connection of the module to detect each new module. In case of use a directional antenna in our system, we choose the circular open-waveguide due to the power received do not change with the polarization of the antenna and has lower gain. With an omnidirectional antenna is very difficult to localize new modules. Even when the radiation pattern is not ideal, the power receiver in the front and in the back is more or less the same, doing difficult to know where the new neighbor is.

On the other hand, the “Module general test” is an excellent technique to check if the hardware of the *Communication Board* is initially working as expected.

4. Design and implementation of the software

4.1. Introduction

Once the hardware was selected, and the system is designed and implemented, questions about which is the best software solution to work emerge. Much more besides, if we want to define a communication architecture, we have to face questions about the radio stack or the communication protocol.

This chapter will first introduce one of the best operating system solutions for wireless sensors networks and module's wireless intercommunication: TinyOS[10]. Following to that, a discussion about advantages and disadvantages of using this operating system will be done.

Secondly, and after a discussion about if it is suitable to adapt or port TinyOS needed the features to our hardware, the creation of a new platform will be briefly explained.

The last two points will explain the Hardware Abstraction Architecture (HAA) implemented for the radio stack and the IEEE 802.15.4 based MAC communication protocol adapted and used.

Researchers from Mærsk Mc-Kinney Møller Institute build a Linux Image that provides a user friendly programming environment with a preinstalled TinyOS operating system. This software provides an easy way to set up the working environment and because of that, we created a guide to configure and set up all the software needed to start programming the microcontroller. The purpose of the guide is to provide an easy way to start working on other projects and can be seen at the Appendix H.

[10] TinyOS official web page: http://www.tinyos.net

4.2. TinyOS as an operating system for the module

TinyOS operating system is designed to run on small wireless sensors or modules. Networks of these sensors have the potential to revolutionize a wide range of disciplines, fields, and technologies, showing the key differences between them and most other computing systems. First, there is the power usage, meaning that the sensors or modules have to operate for a long periods of time. Second, they gather data from and respond to an unpredictable environment. Finally, the wireless factor gets relevance importance for reasons of cost, deployment simplicity, and robustness. Together, these three issues (longevity, embedment, and wireless communication) cause sensor networks to use different approaches than traditional ones.

Since energy consumption determines module lifetime, they tend to have very limited computational and communication resources. Instead of a full-fledged 32-bit or 64-bit CPU with megabytes or gigabytes of RAM, they have 8-bit, 16-bit or 32-bit microcontrollers with a few kilobytes of RAM. Rather than GHz, these microcontrollers run at 1-10 MHz. Their low-power radios can send hundreds of kilobits per second, rather than 802.11's tens of megabits. As a result, software needs to be very efficient, both with CPU cycles and with memory use.

Figure 42 shows a sample of our module with an integrated PCB antenna. It has a powerful Atmel AT91SAM7S256 32-bits microcontroller with 64 KB of RAM and 512 KB of flash program memory. Despite its powerful specifications, this microcontroller has very low power usage. Its radio chip, a TI CC2420 which follows the 2.4 GHz IEEE 802.15.4 standard, can send up to 250 kbps.

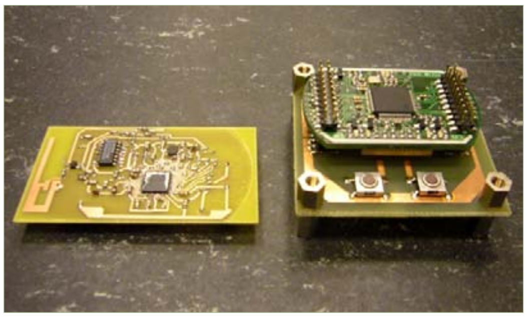


Figure 42. Components of our system

TinyOS makes building sensor network applications easier. It provides a set of important services and abstractions, such as sensing, communication, storage, and timers. It defines a concurrent execution model, so developers can build applications out of reusable services and components without having to worry about unexpected interactions. TinyOS runs on over a dozen generic platforms. Furthermore, TinyOS's structure makes it reasonably easy to port to new platforms which are very important for our hardware design.

TinyOS applications and systems, as well as the OS itself, are written in the nesC language. NesC is a C dialect with features to reduce RAM and code size, enable significant optimizations, and help prevent low-level bugs like race conditions[11]. The place where nesC differs greatly from C is in its linking model. The challenge and complexity isn't in writing software components, but rather in combining a set of components into a working application. However, we do not want to go into details on how nesC differs significantly from other C-like languages. A good documentation and tutorials can be found on Internet[12].

4.2.1. Benefits and inconvenience of using TinyOS

At a high level, TinyOS provides three things that make writing systems and applications easier:

- Component model, which defines how to write small, reusable pieces of code and compose them into larger abstractions.
- Concurrent execution model, which defines how components interleave their computations as well as how interrupt and non-interrupt code interact.
- Application programming interfaces (APIs), services, component libraries and an overall component structure that simplify writing new applications and services.

The component model is grounded in nesC. It allows to write pieces of reusable code which explicitly declare their dependencies. For example, a generic user button component that notifies when a button is pressed is on the top of an interrupt handler. The component model allows the button implementation to be independent of which interrupt is, without requiring complex callbacks.

The concurrent execution model enables TinyOS to support many components needing to act at the same time while requiring little RAM. First, every I/O call in TinyOS is split-phase: rather than block until completion, a request returns immediately and the caller gets a callback when the I/O completes. Since the stack is not waiting for I/O calls to complete, TinyOS only needs one stack, and does not have threads (it has what is called tasks). Any component can post a task, which TinyOS will run at some later time. Because low-power devices must spend most of their time asleep, they have low CPU utilization and so in practice tasks tend to run very soon after they are posted. Furthermore, because tasks cannot preempt each other, task code does not need to worry about data races [20].

Finally, TinyOS itself has a set of APIs for common functionality, such as sending packets, reading sensors, and responding to events as well as a component structure and component libraries. As we will see later, we can use these APIs to build up the software

[11] A race condition occurs when multiple processes access and manipulate the same data concurrently, and the outcome of the execution depends on the particular order in which the access takes place.
[12] http://docs.tinyos.net/index.php/Main_Page

from low-level hardware (i.e. a radio chip) to a hardware-independent abstraction (i.e. sending packets). A sample of a NesC code it is shown in Figure 45.

As an inconvenient, we can say that although is not such a hard effort to learn nesC language if we are familiarized with C programming language, it can be seen as an obstruction to start programming applications. Along with that, most of the C common libraries that are not included in TinyOS cannot be used easily for an application if they are not previously ported to TinyOS.

4.3. Creating the drivers and a new platform for TinyOS

Because TinyOS is ported to many platforms, it is possible to do a comparison between the platform we want to port and the existing ones to see if some components are common. If so, then the code used can be adapted to the new platform. After doing some comparisons, the most suitable platform is the one using at Atmega128 microcontroller (i.e., MicaZ platform). However, due to our Atmel microcontroller only few parts can be adapted and we almost have to create a new platform from the scratch. Therefore, we follow platform structure, but we still write all the code needed to port the microcontroller. This platform also has a working TI CC2420 transceiver driver implemented for the Atmega128 microcontroller. However, due to our Atmel microcontroller and many features already implemented that we do not need, we create a new one.

Adapting or creating a new platform implies the modification or creation of a set of chip-related and hardware-related files to configure the clock, the onboard LEDs, the Serial Peripheral Interface (SPI), and necessary microcontroller and transceiver's registers. Thus, many adjustments must be done such as configuring necessary microcontroller and transceiver's pins or configuring interruptions.

We briefly explicate the most important microcontroller and transceiver hardware configurations between needed to accomplish the communication between the *General Board* and the *Communication Board*. The hardware schematic is attached in Appendix E and the code implemented is attached in Appendix G.

The first step we must do to communicate with the *Communication Board* peripheral is configuring the microcontroller's configuration registers to set up the *General Board*. The board is provided with a voltage regulator that provides 3.3v to power external specific boards. Thus, we must configure that component in order to supply power to the *Communication Board*.

The next step is to configure the peripheral SPI pins. The SPI Atmel microcontroller's circuit is a synchronous serial data link that provides communication with external devices. It consists of two data lines and two control lines. According to that, we need to set up the SPI communication to be able to transmit and receive data to and from the transceiver. In addition, 4 more pins are needed to interface the transceiver.

Because one of the 4 pins of the transceiver should be connected to an interrupt pin of the microcontroller, we have to configure the microcontroller's interruptions in order to permit the transceiver to interrupt the microcontroller if there is any packet ready to be read (received) from the transceiver's reception buffer (RXFIFO).

After configuring the microcontroller, the next step to communicate the *General Board* with the *Communication Board* is to perform the TI CC2420 radio chip configuration. This configuration can be done via the same SPI interface and making use of the different command strobes.

Briefly, we must first enable the transceiver's voltage regulator (VREG). This is necessary for the applications that make use of this power-saving feature. When VREG is disabled, the transceiver switches to a low-power level (e.g. when the radio chip has to be idle for a specific period).

The next transceiver's configuration steps consist of programming all the needed registers. This must be done to reset the device, turn on the crystal oscillator, configure the send and receive buffers, set the transmission power, configure the radio to transmit or receive, and set up all the features needed for our applications. Complete descriptions of the registers are given in the TI CC2420 radio chip's [11].

Once all the registers are programmed accordingly to our necessities, the module is ready to transmit and receive.

4.3.1. Communication between pair of modules

After setting up the module, the next step was to implement the software to test if the transmitter module is transmitting signal, but the first matter to solve was how to know if our transmitter is really transmitting any signal. There is no way to check it with the common instruments (e.g. voltmeter or oscilloscope), and a Spectrum Analyzer was used. A Spectrum Analyzer is a laboratory instrument that displays signal amplitude (strength) as it varies by signal frequency. It is widely used to measure the frequency response, noise and distortion characteristics of all kinds of RF circuitry.

In order to obtain more accurate results, we add a test point into the *Communication Board* by incorporating a female SMA connector to connect the board directly with the Spectrum Analyzer, and check the signal emitted by the transceiver. The Spectrum Analyzer and the *Communication Board* with the test point are shown in Figure 43.

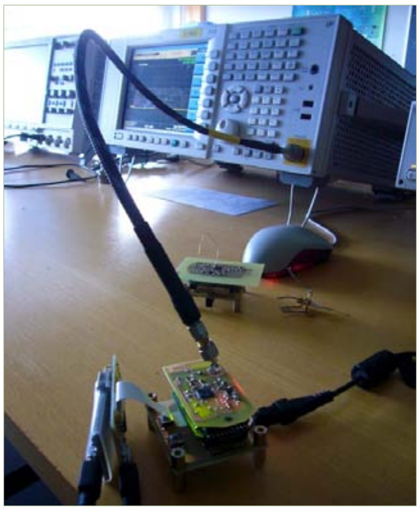


Figure 43. Connection to the Spectrum Analyzer

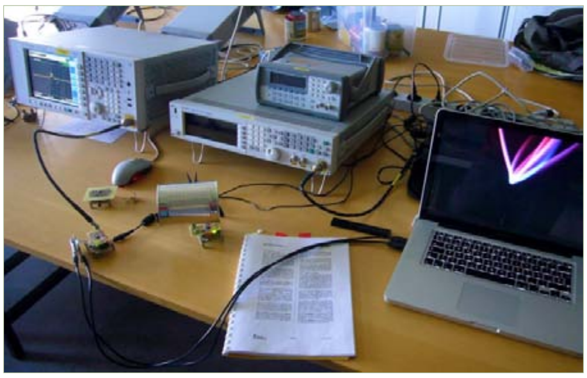


Figure 44. Testing the receiver with the Spectrum Analyzer and Sweeper

To check the communication from a first approach, we used the built-in TI CC2420 transmitter test mode. This mode requires the configuration of a specific device registers in order to transmit random data with an unmodulated carrier, and verify if there is any hardware problem. A sample code showing the test mode function structure and how to modify the CC2420 registers is shown in Figure 45. This function activates one of the test modes available or deactivates all. Once we configured all registers, we were able to see the carrier signal. A plot of single carrier spectrum is shown in Figure 46.

If we want another module as a receiver, we can use the test mode but this time, generating a modulated spectrum to allow it to synchronize. Thus, with the transmitter running on the test mode, a second module configured as a receiver were used to verify the reception of the modulated spectrum. In Figure 44 is shown the deployment resources, where we use a directional antenna for the transmitter. Once we realized that we must adjust some settings in order to success with the communication, we were able to see the modulated signal on the receiver module. A plot of the modulated spectrum received is shown in Figure 47.

```
command void CC2420Spi.setTestMode(bool enable, uint8_t mode)
{
        // Turn off the crystal oscillator and RF
        call CC2420Spi.strobe(CC2420_SRFOFF);
        delay_us(128);

        // if we want to set the test mode, we specify the number in mode
        if (enable) {
                call CC2420Spi.setReg(CC2420_MDMCTRL1, 0x0500 |
                        ((mode << 2) & 0xf));
                call CC2420Spi.setReg(CC2420_DACTST, 0x1800);
        }
        else {
                call CC2420Spi.setReg(CC2420_MDMCTRL1, 0x0500);
                call CC2420Spi.setReg(CC2420_DACTST, 0x0000);
        }

        // we must flush the RXFIFO and TXFIFO
        call CC2420Spi.strobe(CC2420_SFLUSHTX);
        call CC2420Spi.strobe(CC2420_SFLUSHTX);
        call CC2420Spi.strobe(CC2420_SFLUSHRX);
        call CC2420Spi.strobe(CC2420_SFLUSHRX);
}
```

Figure 45. Sample code for the Test Mode

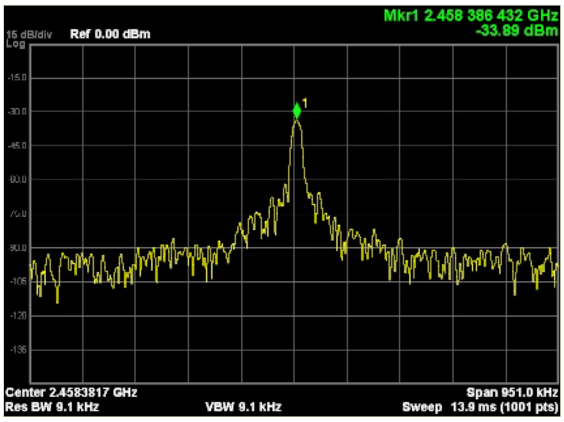


Figure 46. Simple carrier spectrum

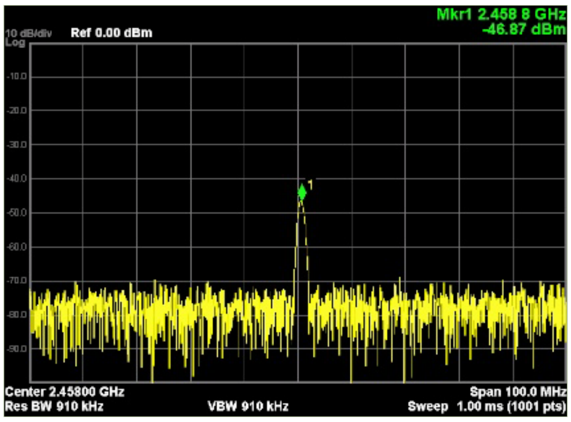


Figure 47. Modulated Spectrum received

Once the devices are communicating, we are ready to implement the radio stack and configure the communication protocol.

4.4. Hardware Abstraction Architecture for CC2420 RF transceiver

4.4.1. Motivation

The TI CC2420 is a 2.4 GHz IEEE 802.15.4 compliant RF transceiver designed for low-power, low-voltage and high-performance wireless applications. Although in this thesis the usage of all the functionalities CC2420 radio chip is out of the scope, the radio chip provides extensive hardware support for packet handling, data buffering, burst transmissions, data encryption, data authentication, clear channel assessment (CCA), link quality indication, and packet timing information. This means that a well defined Hardware Abstraction Architecture (HAA) is required to provide a good portability, scalability, and reusability.

Because the functionalities of the three layers that composes TinyOS CC2420 radio stack are not classified clearly from the viewpoint of hardware dependency, the portability of the CC420 driver to a new platform is not straightforward without spending much time adapting all the functionalities needed. Therefore, a much easy HAA is used.

The functionalities of the new basic HAA for CC2420 transceiver do not include the same ones as the TinyOS radio stack. Indeed, the radio stack built in this thesis is a basic version of the TinyOS driver, adapted to our specific hardware whilst providing at the same time a good abstraction for future work. Therefore, functionalities like CSMA/CA wireless multiple access method, MAC security operations such encryption and authentication, or timing information where not implemented. In addition, the device driver is implemented to support the IEEE 802.15.4 MAC protocol only.

Our HAA consist of three layers and gives an abstraction of CC2420 radio chip hardware to upper layer application software (Figure 48). It hides the hardware implementation details such as processor, memory management, interrupt controller, etc. The three layers are HIL (providing a generic interface to application software), HAL (that manages resources and status) and HPL (built to access directly to CC2420 device to perform a setting to communication with microcontroller). Each one is defined depending on responsibilities and interfaces provided by lower layers.

For further information, the most important parts of the code implemented on each component can be seen in Appendix G.

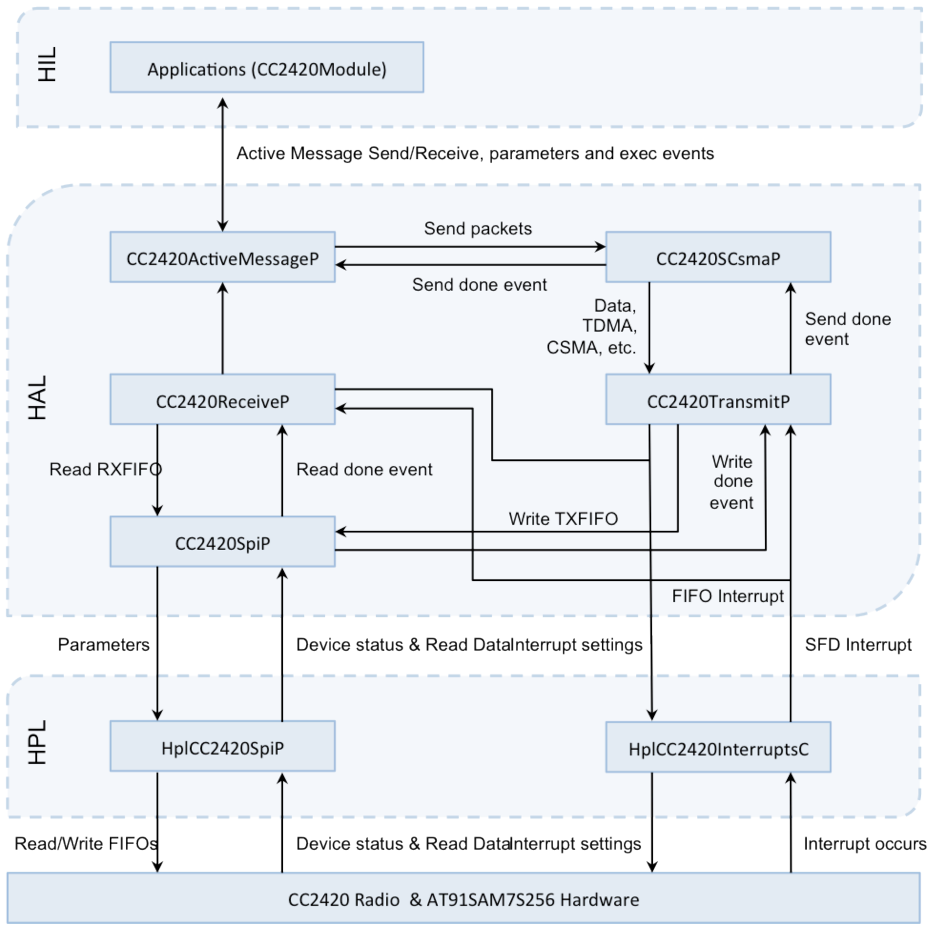


Figure 48. HAA for CC2420 RF transceiver

4.4.2. Hardware Interface Layer (HIL)

The first layer of the HAA is formed by the HIL components that take the platform-specific abstractions provided by the HAL and convert them to hardware-independent interfaces used by portable applications. In other words, it simplifies the development of the software by hiding the hardware differences.

TinyOS uses the Active Message (AM) to transmit and receive the data between modules with a defined structure format called “message_t”. In our system, instead of “message_t” we use a simplified structure named “cc2420_packet”, responsible for filling in details in the packet header and providing information about the packet to the application layer. Furthermore, this structure is compliant with IEEE 802.15.4 the CC2420 device itself uses standard headers in hardware. More details about the IEEE 802.15.4 MAC function and protocol implemented and used by our system will be explained later.

Finally, applications interface CC2420ActiveMessageP component through each component application.

4.4.3. Hardware Adaptation Layer (HAL)

The HAL of the CC2420 is an important part because it manages resources and controls the status of CC2420 device.

The CC2420ActiveMessageP component fills the header packet of “cc2420_packet” and maintains another structure called “rfsettings”, responsible for filling in details regarding specific network and module configuration, like PAN address or module source address. This structure maintains as well a unique data sequence number (DSN) byte for the packet header that is incremented by one per outgoing packet. This is used to detect duplicate packets by the receiver and is defined in the IEEE 802.15.4 specification.

The CC2420SCsmaP component is responsible for defining IEEE 802.15.4 FCF byte information in the outgoing packet, defining the power-up/power-down procedure of the radio and (when implemented) is responsible to detect if a channel is busy or not, providing *backoff* times to avoid a collision. It maintains up to date the other entire packet’s information. The MAC communication protocol will be explained in the next section.

The CC2420TransmitP and CC2420ReceiveP components are responsible for interacting directly with the hardware of the radio through the SPI bus, GPIO lines and interrupts. They are the last step in transmitting and sending the information to the physical layer and for instance they are responsible of filling the transceiver buffers and manage the resources by means of locker variables. CC2420ReceiveP component also maintains updated a “cc2420_packet” structure used to keep the data of the last received packet.

4.4.4. Hardware Presentation Layer (HPL)

The HPL of the CC2420 is directly connected to the radio hardware to perform the setting of the GPIO pin connection to communicate with microcontroller. In addition, the HPL write to and read from CC2420 registers or RAM via SPI bus.

The component responsible to perform the SPI communication and the RAM access is the HplCC2420SpiP. All the functions we need to interact with the registers and the RAM on the CC2420 radio chip is defined here.

The HplCC2420InterruptsC component performs the setting of the interrupt related to the CC2420 radio chip. We use three of the four possible interrupts: SFD, SPI and FIFOP. As we said before, we do not use CCA interrupt because our system will never make use of channel selection methods due to the aim of the thesis.

Finally, another component that is not on shown in the HAA, mainly because it does not have relationship with the radio stack, is the HplAt91SpiP. This component is responsible of configuring the microcontroller pins as well as the SPI interface and the power provided to the *Communication Board*.

4.5. Wireless Medium Access Control (MAC) communication protocol

The TI CC2420 transceiver has hardware support for the IEEE 802.15.4 standard [21]. This standard defines a compatible interconnection for data communication devices using low-data-rate, low-power, and low-complexity short-range radio frequency (RF) transmissions.

The implementation of the physical layer (PHY) and the medium access control (MAC) sublayer of our protocol is based on the IEEE 802.15.4 standard. We do that to make our system compatible among the other modules and make use of the radio chip features that works with this standard [22 s. Chapter 4].

The TI CC2420 hardware support parts of the IEEE 802.15.4 frame format. We will explain briefly how the CC2420 transceiver has to be set up to comply with the standard, by both the description of the MAC and physical layer, as well as the description of the frame structure used.

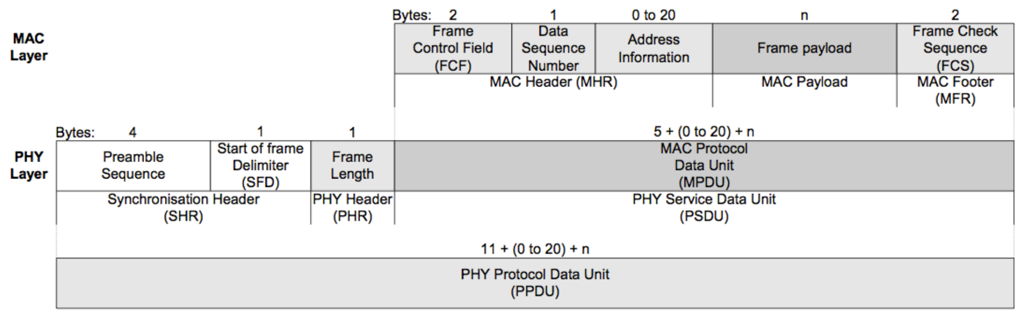
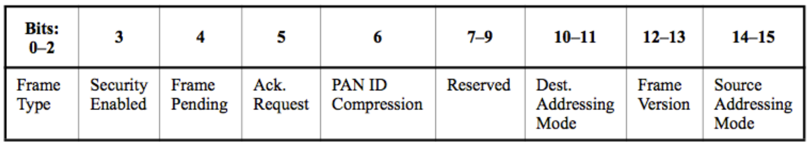


Figure 49. Schematic view of the IEEE 802.15.4 Frame Format [21]



Bits: 0–2	3	4	5	6	7–9	10–11	12–13	14–15
Frame Type	Security Enabled	Frame Pending	Ack. Request	PAN ID Compression	Reserved	Dest. Addressing Mode	Frame Version	Source Addressing Mode

Figure 50. Format of the Frame Control Field (FCF) [21]

Physical Layer (PHY)

The PHY provides an interface between MAC sublayer and the physical radio channel, via the device driver and RF hardware. Each PPDU (PHY Protocol Data Unit) consists of the following components:

- A synchronization header (SHR), which allows the receiver to synchronize and lock onto the bit stream, in order to start receiving. This component is hardware dependent.
- A PHY header (PHR), which contains frame length information.
- A variable length payload, which carries the MAC sublayer frame, which at the same time contains the packet information we want to send.

Medium Access Control (MAC) sublayer

The MAC sublayer handles all access to the physical layer. The MAC forms a MAC Protocol Data Unit and each MPDU consists of the following basic components:

- A MHR, which comprises frame control (FCF), sequence number (DSN) and address information, as we can see at figure X. The FCF provides packet and security-related information.
- A MAC payload, of variable length, which contains information specific to the frame type.
- A MFR, which contains a Frame Check Sequence (FCS) used to verify if the frame is transmitter correctly over the air or is corrupted. On the CC2420 device this check is done by hardware by configuring the appropriate register. Thus, the component content is replaced by the RSSI Signal Strength value.

The frames in the MAC sublayer are described as a sequence of fields in a specific order. It means that when we are building one frame that contains a packet to send, the CC2420 radio chip driver must follow this frame structure.

As explained before, the CC2420SCsmaP component is responsible of filling all the FCF information. For our project purpose, we will not always use the full 802.15.4 MAC packet format. Sometimes, we will not need some components from the MPDU. For example, the DSN number is not useful when you do not need to check if some packets got lost while transmitting. The address information is not always needed when you do not use the address recognition feature included on the CC2420 hardware. Finally, the

FCS component is not needed since the verification of the correct transmission of the packet is done by hardware. This last point is very interesting for our project because the TI CC2420 replace the content of this component by the Received Signal Strength Indicator (RSSI) value plus a bit indicating if the hardware verification is successful or not. The use of the RSSI will be introduced later but briefly, it is the voltage measured by a receiver's received signal strength indicator.

Figure 51 shows a sample code of the FCF filling procedure. In this sample, we can see the configuration of the MPDU's total length, the specification of the frame type (data), and if we will include the source and destination address on the packet.

After filling the FCF information, the next and last step is done by the CC2420TransmitP component that takes care about the insertion of the rest of the components of the MPDU and putting the frame in a specific order before sending it to the transmit buffer.

```
packet->length = len + RF_PACKET_OVERHEAD_SIZE;
packet->fcf &= 0 << IEEE154_FCF_ACK_REQ;
packet->fcf |= ( ( IEEE154_TYPE_DATA << IEEE154_FCF_FRAME_TYPE ) |
             ( 1 << IEEE154_FCF_INTRAPAN ) |
             ( IEEE154_ADDR_SHORT << IEEE154_FCF_DEST_ADDR_MODE ) |
             ( IEEE154_ADDR_SHORT << IEEE154_FCF_SRC_ADDR_MODE ) );
```

Figure 51. Code of the FCF filling procedure

4.6. Conclusions

The main conclusion we drawn of this chapter is that TinyOS makes building sensor network applications easier. It is designed to run on small wireless modules with low storage capacity and providing low-power usage, which makes it suitable to use it in our modules.

We developed a new platform for TinyOS in order to use the Atmel microcontroller with this operating system and the driver for the TI CC2420 transceiver. Thus, a Hardware Abstraction Architecture was created as well as the communication protocol based on the IEEE 802.15.4 standard. All this components provided us all the software implementation necessary for the modules to communicate the *General Board* with the *Communication Board*, and thence make the communication between modules possible.

5. Communication

5.1. Introduction

The implementation of control architecture mostly depends on the communication structure upon which it is built. Communication between modules can be achieved through a global “wireless bus” and/or locally using neighbor-to-neighbor communication.

As we already know, one of the the main objectives of the thesis is to test and analyze the performance of RF communication, and conclude if RF is appropriate for both local and global communication on modular robots.

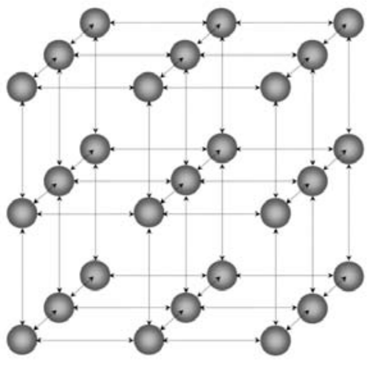


Figure 52. Mesh network example

The best practical way to implement inter-module communication with radios is to deploy a mesh (collection of interconnected modules) networks. This means that mesh networks do not have a central coordinating node like in the star topology ones. Instead, they have no coordinator and each node can connect to multiple neighbors. Besides that, radios do not require precise module alignment to establish a communication and it is vital when we are talking about implementing decentralized algorithms.

Mesh networks like the one shown in the Figure 52 introduce challenges of their own: modules cannot transmit in parallel and they suffer from latency scalability. The seminal paper by Gupta and Kumar [23] demonstrates that the capacity of mesh networks scales by $\Theta(\sqrt{n})$ as the number of modules n grows larger. This also implies that the throughput available to each module is $\Theta(\frac{1}{\sqrt{n}})$, making this kind of networks not suitable for applications with large number of modules. Nevertheless, this cannot be applied when communication is local and where the communication is predominantly neighbor-to-neighbor. Multi Radio Multi Channel mesh network architecture for modular robots that successfully address the scalability problem and demonstrate a channel reuse scheme was presented in [24].

On this chapter, we will first demonstrate the reliability and performance of the system with the aid of some experiments to finally demonstrate the suitability of both local and global communication, making use of the results gathered from the experiments.

5.2. Local and global communication

Modules of a modular robot should communicate both locally and globally. Global communication between modules is important, but local communication without alignment or wired links is the new challenge.

Local communication is based on module-to-module data communication. The main purpose of the local communication is to determine its structure of the robot. That is, how modules are connected each others, allowing the robot to know the task to develop at each moment. Discover the structure or topology of the robot makes use of what is called *localization*. The topology cannot be discovered using global communication because messages do not contain this kind of information. One alternative to know the topology of the robots could be to program this information *off-line*, limiting the shape of the robot and making difficult to maintain the information of its topology when the modules change or when more modules are added to the configuration.

Global communication allows modules in different parts of the system to communicate to each other directly. Also, it is possible to allow global communication through local communication, where the data has to travel through many modules, generating delays and loss of data. However, sometimes is important to transmit information quickly in applications where modules in different parts of the configuration need to work together. But, the problem is that the medium to share information will become saturated if enough modules use it at the same time.

Most common technologies used to communicate modular robots are infrared (IR) communication, Bluetooth and Wire links.

Wire links and IR are good in local communication, but not in global communication. With wire links the robot knows each moment its topology, but it lost the option of add new modules. The global communication is through the local communication, through all the modules. However the advantage is that wire links are fast, they have a very low delay and there is no data lost.

With the IR the robot knows each moment its topology, but it has the challenge of detect a new module due to the maximum distance between two modules to detect each other is very small. The global communication is through the local communication, but it is slow and sometimes the data is lost.

Bluetooth is good in local and global communication. However the main problems are some loss of information from time to time, the saturation of the medium and it is very difficult to localize the position of the neighbors (robot topology). Another disadvantage, it is need a host, and if it fails the robot fails.

Next points will explain all the measurements we made to better understand our system response to these communication techniques and will compare RF communication with others technologies.

5.3. Experiments: system performance measurements

5.3.1. Architecture overview

Our proposed wireless communication system works well in an already set environment. The layout in which modules are placed is a two-dimensional square lattice. Each module has one radio allocated at fixed channel (that is the same for all the modules) with face-to-face antenna configuration and working at 0 dBm strength power (unless otherwise stated).

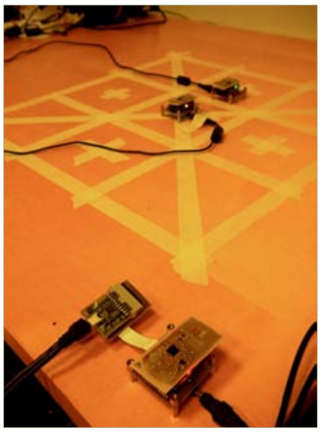


Figure 53. Architecture for the experiments

To test the system performance, we made all the experiments with a pair-wise module: one configured as a transmitter and the other one as a receiver, which it will collect all the experiments data. In order to avoid introducing more undesired communications issues, a third module called sniffer is used. This module will be fixed at different channel than the TX and RX module to receive all the data information sent by the analyzed module. Thus, we can do the measurements in a more isolated environment, while providing non-packet collision scenery between the pair modules being tested and the receiver-sniffer communication by using a different channel.

5.3.2. Experiment setup

In this section we discuss the hardware and the software that is used in the experiments.

5.3.2.1. Hardware selection

The hardware used to test the performance of the system is composed by three modules: a transmitter, a receiver and a sniffer. All this modules are provided of a PCB omnidirectional antenna in the *Communication Board*.

5.3.2.2. Software

The testing software is implemented on the IEEE 802.15.4 stack. Each radio used is setup either as a transmitter or a receiver as well as a sniffer using the same software. This means that we have one application for all the modules instead of having one for each function; we only have to specify the module's function and it will execute its functionalities. The purpose of the software is gather all the information needed for the experiments between transmitter and receiver modules.

The transmitter on startup begins sending beacon packets [22 s. Chap. 4] to a broadcast address, meaning that they are readable for all the transmitters listening at channel 26 (2480 GHz). The beacons are sent every 128ms at the configurable power strength. The receiver, instead, on power up begins listening for any beacon. Once the receiver hears a beacon it gathers all the information needed for the experiments (from RSSI values or timing until channel information, among others) and sends it by a data packet over channel 25 (2475 GHz) to the sniffer module.

The sniffer starts up hearing at channel 25 for data packets to, afterwards, print the entire packet information through a serial port to a monitoring computer.

Finally, the onboard LEDs were used to verify the transmission and reception to identify the role of each module during the tests and for debugging purposes. Also, the memory usage of the software used on this test was very optimized (8.4 Kbytes of ROM and 1.3 Kbytes of RAM).

5.3.3. Experiments and results

Module alignment issues

On neighbor-to-neighbor modules communication, links must tolerate an uncertain alignment because precise alignment presents a difficulty on connectors design. However, radios do not require precise module alignment to communicate locally and globally, as Wire links and Infrared (IR) solutions do. A successful approach to crosstalk and neighbor detection problems in IR or wired communication links was demonstrated in [25].

The *Communication Board* of our system can be provided by a PCB omnidirectional antenna or a directional one. The first antenna does not suffer about misalignment problems due its radiation pattern, which allows the module to communicate with other modules in all directions. It is true that the radiation pattern is not perfectly circular and depending on the transmitting power, the communication can suffer from some limitations which we face in the chapter about neighboring localization.

The radiation pattern of the directional antenna does not allow the radio to communicate in all directions, as it focuses the RF energy to a specific direction. Thus, as the gain of a directional antenna increases, the coverage distance increases, but the effective coverage angle decreases. However, when trying to focus the RF energy in one direction, a little energy can be found on the back side of the antenna due to the secondary lobes.

For our purpose, the coverage of the module's antenna does not need to be high, and therefore, the coverage angle will be bigger. As we will explain later, the directional antenna will be only used during the neighboring discover, meaning that the alignment issues will not affect the inter-modules communication since we use the antenna only to discover the neighbor.

To determine if we have any alignment issues with the directional antenna, we measured its power strength variation when it is in different angle positions to figure out the radiation pattern. A table with all the results and graphics can be seen at chapter 4 (on measurement of antennas section). As we can see in Figure 41, the maximum power is when the antenna is focused to the desired direction and quickly varies when increasing or decreasing the angle of the transmission. Thus, we do not expect many alignment issues because, although the power strength decreases by a factor of 16 between 0 and 90 degrees, the antennas are still able to receive the signal. Thus, the misalignment is not a problem that directly affects our system.

Power consume calculation

To check the approximate power consumption of one of the modules, we calculate separately the "*General Board*" and the *Communication Board* power consumption when they are at maximum load.

Regarding to the microcontroller, we checked its power consumption in the following mode conditions [26]: PLL activated, Voltage regulator on, flash is read, all peripheral clocks activated and the microcontroller running at 50 MHz frequency. According to that, the power consumption is around 30.6mA. In addition, we have to include the LEDs consumption (around 40mA when all of them are switched on) and the 16 μA per MHz of the SPI, which it is 16 μA because the SPI clock runs at 1MHz.

For the TI CC2420 peripheral (included in the *Communication Board*) a power consumption of 15.75mA was found on the empirical experiment for the device [10]. This value is obtained by dividing 52mW per 3.3 V of the microcontroller.

If we put all the values together, our system running at maximum load consumes a total of 284.95 mW (86.35 mA) with places our module in the low-power range.

Throughput

The throughput of a pair of modules communication is limited by the serialization delay, meaning that is not possible to send more data than the bit rate allows (TI CC2420 defines a 250 Kbps bit rate) [11]. However, the bit rate is not the only limiting factor. As we already know, the 802.15.4 physical layer frame format consists of 4-byte preamble, 1-byte Start of Frame Delimiter (SFD), and 1-byte frame length field (FCF). The maximum physical layer payload (MPDU) size is 127 bytes. For the purpose of our experiment, we use the full 802.15.4 MAC packet format but without using the address information field. Thus, our MAC Protocol Data Unit (MPDU) only consists of FCF field, DSN, application payload data, and Frame Check Sequence (FCS) field with CRC and RSSI information. Hence, the maximum information we can send now is 118 bytes and the theoretical throughput can be calculated as:

$$T_t = \frac{118}{4+1+1+118+2} \times 250Kbps \cong 234Kbps$$

Equation 11. Theoretical throughput

To measure the real throughput of a pair of modules, the receiver record the number of bytes received per unit of time. This measurement is done by sending 1000 packets of maximum size (128 bytes) at maximum data rate. However, we cannot calculate the throughput accurately if some packets are lost during the transmission.

To know the lost-packet data rate we can use the acknowledgment packets. These packets act as a packet-reception confirmation and are sent back to the transmitter every time the receiver receives a data packet. If the transmitter does not receive a packet-reception confirmation, it resends the packet again. However, this means duplicate the number of packets sent by both the transmitter and the receiver and the system throughput will result divided by two. Instead of that, we use the Data Sequence Number (DSN) described before. This number acts like a counter that is increased every sent packet, meaning that the receiver will know every time if any packet gets lost.

We obtained a throughput of 221 Kbps which is close to the 234 Kbps theoretical throughput. This difference is due to the microprocessor time delay, the transceiver turnaround time before transmitting and some delays caused by the software.

As an extra result, we tested and demonstrated the system communication robustness by leaving the modules communicating during 30 minutes. The modules were still working after that amount of time and the throughput was constant.

5.4. Experiments with CCRR: testing local and global communication

5.4.1. Architecture overview

For our wireless communication system based on Single Radio Single Channel network architecture, we want to prove that our modules are able to communicate either locally or globally among its neighbors. We place two pairs of modules in a two-dimensional square lattice. Each module will have one radio (one transceiver) and each one will be allocated at the same fixed channel as its neighbors. However, the challenge of how to choose a suitable distance between modules to accomplish our goal is introduced.

To know at which distance resolution we should place the modules on our topology, we apply the Free-Space Path Loss (FSPL) equation, which describes how an electromagnetic wave attenuates in free space [27], to convert the Co-Channel Rejection Ratio (CCRR) value to Euclidean distance to obtain the necessary separation between two radios for interference-free communication:

$$d = 10^{\frac{CCRR+20log_{10}D}{20}}$$

Equation 12. Euclidean distance

where *D* is the maximum Euclidean distance from one module's transmitter antenna until the receiver's one as shown in Figure 54, and *CCRR* is equal to -3 dBm (given by the TI CC2420 hardware specification). CCRR quantifies the necessary signal strength ratio needed between a desired and interfered signal such that the desired signal can be correctly received without any corruption.

Two tests will be performed. On the first one, we demonstrate how our system can provide local as well as global communication and determinate the CCRR of the radios for successful co-channel operation. The second one will demonstrate that two pair of modules can communicate at the same time using the same fixed channel 26 and without causing interferences.

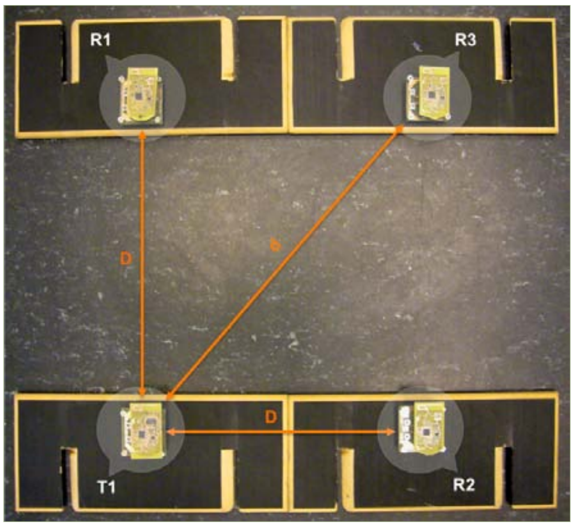


Figure 54. Local and global communication test scheme changing power strength and channels

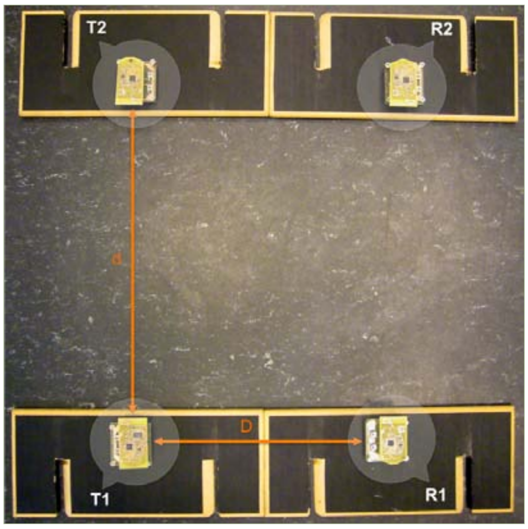


Figure 55. Local and global communication test scheme reusing channels

5.4.2. Experiment setup

In this section we discuss the hardware and the software that is used in the experiments.

5.4.2.1. Hardware selection

For the first test, we use 4 modules: 3 receivers and 1 transmitter. For the second one, 2 modules are configured as transmitters and 2 as receivers (each one provided with the same PCB omnidirectional antenna).

5.4.2.2. Software

The software used is implemented on the IEEE 802.15.4 stack. Each radio used is setup either as a transmitter or a receiver as well as a sniffer using the same software. This means that we have one application for all the modules instead of having one for each function; we only have to specify the module's function and it will execute its functionalities. The purpose of the software is to gather all the information needed for the experiments between transmitter and receiver modules.

In both tests, a module configured as a transmitter will send beacon packets every 128 ms to a broadcast address on a fixed channel 26 and with a power strength of -15 dBm. After certain amount of time, the transmission power will be increased up to -10 dBm, after another certain amount of time, switch the transmission to channel 25 keeping the same transmission power level. In Figure 54 modules R1 and R2 will be configured as a receivers listening to channel 26. Module R3 will be listening half time on channel 25 and half time on channel 26 with a fast switch of 128/2ms.

5.4.2.3. Experiments and results

As we said, in order to accomplish with the goal of the test we must calculate de Euclidean distance d using the Equation 12:

$$d = 10^{\frac{-3+20\cdot log_{10}37}{20}}$$

where D distance is the maximum distance in centimeters where the transmitter can communicate with its immediate neighbors at certain configured power strength (in this test is at -15 dBm). In that point, RSSI neighbor's values are into the limit of the antenna sensibility (≈-80 dBm). Thus, a distance d = 46.60 cm is obtained. With all this information, we place the modules in a two-dimensional square lattice to make two different tests.

Three-receivers/one-transmitter configuration

For the first test we use the configuration showed in Figure 54. To prove local and global communication, we divide this test in 3 steps. In the first step of the experiment, receiver modules R1 and R2 can communicate with the transmitter T1, and R3 do not due to it is out of the range. The second steps of the experiment only increases de transmit power of module T1 from -15 dBm to -10 dBm which will allow all the modules to communicate. This happens because the distance between the modules is the same (distance d does not changed), but the power strength not. On the third step, module T1 changes the transmitting channel to 25 causing all the modules to stop communicating except R3 that is listening in both 25 and 26 channels.

Two-receivers/two-transmitters configuration

For the second test we use the configuration showed in Figure 55 to prove local and global communication in a different way testing a single channel scheme, where the same channel can be reused by another neighbor-neighbor pair. Thus, if we use the figure above as a reference, the two transmitter modules T1 and T2 are able to communicate with their respective receiver's modules R1 and R2 using the same channel 26, without interfering the others communication and avoiding crosstalk issues.

To accomplish that, one pair of modules should be out of the range of the other pair. For instance, the distance between them must be bigger than d.

5.5. Conclusion

Regarding to the question introduced before, if RF is appropriate for local and global communication on modular robots, the answer still has some buts. The experiments showed that we can provide local and global communication limiting the distance between modules, changing the strength power and even using the same channel in the communication. Hence, we tested that we can use RF to communicate modules, but the solution with the chosen hardware is not suitable for modular robots due to the long distance needed between modules for a reliable communication. However, this problem can be solved improving the components chosen in the "communication board".

Comparing RF communication with the other technologies we found that the main difference between RF, Infrared or Bluetooth is that RF can provide reliable local and global communication without regarding to the inter-module docking orientation, mechanism and number of modules connected. As was explained in the introduction, Infrared only can provide neighbor-to-neighbor communication and suffers from module misalignment issues, while Bluetooth can communicate locally and globally but requires a central module that constrains the size of the network. With Radio Frequency we did not have the above problems.

The RF system implemented is a low-power solution, with high-rate data transmission and could achieve long-range communication between modules. The consumption of the whole system is 284.95 mW when the system is operating under full load. The consumption of the transceiver when it is transmitting at maximum power strength is 51.98 mW. The maximum data rate measured of the transceiver is 234 Kbps, making our system fast in communication.

Although we cannot generally compare the features of our system with the other technologies, due to depending on the commercial device chosen, the features of the communication changes significantly, we can compare with a specific example where the IR communication is used, ATRON. The data rate is 115,2Kbps, which is the half of our system. The consumption is lower (at most 3mA per channel), while in our system is 15mA. The range distance in IR is 20cm, while with RF we can achieve 29m as we will demonstrate in the next chapter. Finally, the cost of the IR transceiver is 6.5 Euros, but it is necessary one in each possible connection, while the cost of the Communication board is 13 Euros.

Finally, we conclude that with RF we could get a reliable, faster, cheaper and achieve longer distance in local communication than with IR, and at the same time we can get global communication, although RF consumes more than IR.

6.

Localization estimation

6.1. Introduction

Once we successfully tested that the local and global communication is suitable for modular robots improving some characteristics in the hardware design, we will study in this chapter the second scope of the thesis: the localization.

Briefly, localization means figure out the neighbors of each module in order to know the topology of the robot. In RF communication this is a big challenge due to the reflections and the omnidirectional antenna features. Kuo in his paper [7] proposed a wireless system based on 6 radios per module placed in each side of a square module, which each radio check the signal strength. The signal strength received from a neighbor will be at least 9.5 dBm greater than a signal received as a crosstalk. This solution provides reliable neighbor-to-neighbor communication in a mesh network performed in 3D plane, being suitable for localize modules.

However, this implementation has two main disadvantages: it requires too much hardware, a *Communication Board* for each side of the square structure module, and the modules lost the chance to communicate globally.

These disadvantages at Kuo scheme motivate us to find a solution that use less hardware, and the modules could communicate locally as well as globally. To accomplish that, we will study possible localization measurements and models to use in localization algorithms in order to use them to help in the neighbor detection. According with the technique chosen in the previous study, we will do some experiments to characterize its suitability in the system and find a possible solution for our scope in localization.

6.2. Localization measurements and models

The key to develop reliable localization systems that use pair-wise measurements is to accurately represent the severely degrading effects of the channel in which the measurements are made. The propagation of RF signals in real-world environments, full of obstructions, reflectors, people and objects in motion, make this representation challenging. This chapter discusses some RF localization methods and algorithms that could be applied in order to help in our neighbor detection case and explains the measurement experiments made in order to characterize the localization measurements. We use some methods from the literature that are referenced in order to test if they can provide a solution for our specific problem.

Regarding to that, range and angle measurements used for localization are generally affected by time and static-varying errors, and environment-dependent errors. Time-varying errors (i.e. due to additive noise and interference) can be reduced by averaging multiple measurements over time. Environment-dependent errors are the result of the physical position of objects (i.e. walls, people and furniture) in the particular environment where the radio is operating. Since the environment is unpredictable, these errors are unpredictable and must be modeled as random. However, in a particular environment like our test laboratory, objects are predominantly stationary and environment-dependent errors will be largely constant over time, producing not realistic results. This is a critical problem in modular robots that are constantly moving around and changing its shape to develop new task or to solve different kind of problems, because the system has to be environment independent as much as possible to provide a robust neighbor localization.

In this section, we try different kind of localization technologies to finally come out with a solution. These technologies include a variety of time-of-arrival (TOA), angle-of-arrival (AOA) and technologies using measurements of received signal strength (RSS). We provide some test runs to check the suitability of some of these technologies in our system. We can divide the tests in two different approaches: the first one assumes that every module (or at least some of them) has prior information about the environment (i.e. its coordinates or some offline-obtained information). This could help us to come out with a determinate solution providing useful information in a specific environment. The information can be obtained during the system initialization or can be done regularly.

The second approach assumes that the modules do not have any prior information about the environment and/or positioning. This converts the system in a more realistic solution accordingly to modular robots usage.

Next point begins by discussing the general methodology and goals of the measurement experiments. In order to present a good coverage of this variety of possible methods used to localize, we based our choice on previous work made during the last years that were used in different hardware with diverse goals.

6.2.1. Measurement characterization

Ideally, our tests measurements will proceed as following: deploy a pair of modules with the PCB antennas one in front of the other at 20 cm (unless otherwise stated), completely aligned and in the same environment. However, we must suppose that signal strength measurements will be affected by some factors as multipath, fading and/or shadowing due to the behavior of RF channel [28]. Thus, some similar tests will produce different results and an extra effort will be needed to obtain the more accurate results possible.

6.2.2. Received Signal Strength (RSS)

Received signal strength or RSS is defined as the voltage measured by a receiver's received signal strength indicator (RSSI) circuit. Usually, RSS is equivalently reported as measured power, for example, the squared magnitude of the signal strength. Wireless radios communicate with modules and each receiver can measure RSS values of RF signals during normal data communication, without presenting additional bandwidth or energy requirements. Because RSS measurements are relatively inexpensive and simple to implement by hardware, they are an important and popular topic of localization research. However, RSS measurements are notoriously unpredictable. If the RSS measurements are useful and take part of a robust localization system, their sources of error must be well understood.

In free space, signal power decays proportional to d^{-2} (based on the appropriate signal propagation model), where d is the distance between transmitter and receiver, but calibration is required for mapping RSSI to distance values. In real-world channels, shadowing and multipath signals are the two major sources of environment-dependence in the measured RSS values. Multiple signals with different amplitudes and phases arrive at the receiver, and these signals add constructively or destructively as a function of the frequency, causing frequency-selective fading. Using spread-spectrum methods can reduce the effect of this type of fading, which averages the received power over a wide range of frequencies. Spread-spectrum receivers are a good solution, since spread-spectrum methods also reduce interference in the unlicensed bands in which the CC2420 radio transceiver operate. However, this RSS variability decreases the relation between the distance and the power strength, more accentuated on indoor environments, as we see in later experiments.

RSSI is usually an 8 or 10-bit number and it is hardware independent. It is obtained from the physical layer, and is used for channel related tasks such as issuing of Clear To Send (CTS), to determine whether two modules can communicate or to help on calculating the quality of a wireless link connection (LQI). It is important to note that RSSI does not imply the quality of the signal, only provides helpful information.

The TI CC2420 radio transceiver operates in the 2.4 GHz ISM band and includes a digital direct sequence spread spectrum (DSSS), as we already know. A built-in received signal strength indicator gives an 8-bit digital value, $RSSI_{VAL}$, that is averaged over 8 symbol periods (128 µs) and a status bit indicates when the $RSSI_{VAL}$ is valid (meaning that the receiver was enabled for at least 8 symbol periods). The power P at the RF pins can be obtained directly from $RSSI_{VAL}$ using the following equation:

$$P = RSSI_{VAL} + RSSI_{OFFSET}\ [dBm]$$

Equation 13. Power received by the CC2420 transceiver

where $RSSI_{OFFSET}$ is found empirically from the front-end gain and it is approximately equal to -45 dBm. In the next sections, when we refer to the receiver's RSSI value measured we refer to the power P and not to the $RSSI_{VAL}$ unless otherwise stated.

RSSI variability

In addition to multipath, fading and shadowing of the RF channel, signal strength measurements are also affected by the following factors:

- *Antenna orientation*: Each antenna has its own radiation pattern that is not uniform. In practice, this means that the RSSI value recorded at the receiver for a given pair of communicating nodes and for a given distance between them varies as the pair-wise antenna orientations of the transmitter and the receiver are changed.
- *Transmitter variability*: Different transmitters behave differently even when they are configured exactly in the same way. Depending on the manufacturing process, RSSI circuits and powers will vary from device to device. In practice, this means that when a transmitter is configured to send packets at a power level of p dBm then the transmitter will send these packets at a power level that is very close but not necessarily exactly equal to p dBm. This can alter the received signal strength indication and thus it can give different tests results and it can lead to inaccurate distance and estimation. In fact, modules may be designed to measure and report their own calibration data to their neighbors to solve the problem explained above.
- *Receiver variability*: The sensitivity of the receivers across same radio chips is different (either because of the antenna design or the transceiver). In practice, this means that the RSSI value recorded at different receivers can be different even when all the other parameters that affect the received signal strength are kept constant.

RSSI behavior in a test run

In order to measure the RSSI behavior of our modules, a test run was made. We characterized our PCB omnidirectional antenna attached to one of our modules. Ideally, the radiation pattern of this antenna should be uniform and it should look like a circle (2-D space) or a sphere (3-D space). Of course, this does not hold in practice. However, without knowing our antenna's radiation pattern it would be impossible to attempt inferring distance or location information directly from RSSI measurements.

We measured RSSI values with three modules (a transmitter, a receiver and a module we called sniffer) in the same environment like other tests we made. This means that the antennas will suffer from interferences and multipath (reflections) due to the floor, walls, people, chairs and other furniture, as it will happen if we consider we are in a real situation with modular robots. However, to simulate a more realistic condition, we avoid the computer that gathers the RSSI information on the receiver, which cause more interference than using a sniffer module. The receiver transmits the RSSI information gathered to the sniffer that is operating in another channel.

Initially we tried to run the test at the second lowest power level (-15 dBm) of the radio. We noticed that even with a low power level, transmitter and receiver could communicate for any position of the receiver and with a long-range distance (120 cm for -15 dBm and with 0 dBm of power strength, the communication range reached the unexpected 29 meters). Apparently, even using low power levels the 802.15.4 radio has a large communication range that is able to generate significant reflections [10]. Surprisingly, the RSSI values recorded at the receiver were almost constant despite our expectations due to the reflections and interferences that could occur in such an environment, probably because of our PCB antenna's design.

Therefore, to run the test we used the -15 dBm power lever. The values obtained kept constant and the modules were able to have a reliably communication, as is shown in Figure 56. X-axis represents the number of RSSI samples and y-axis represents the strength of the measured signal. RSSI values of -80 dBm indicate absence of communication between receiver and transmitter, due to environmental facts and due to the nature of the 2.4 GHz frequency.

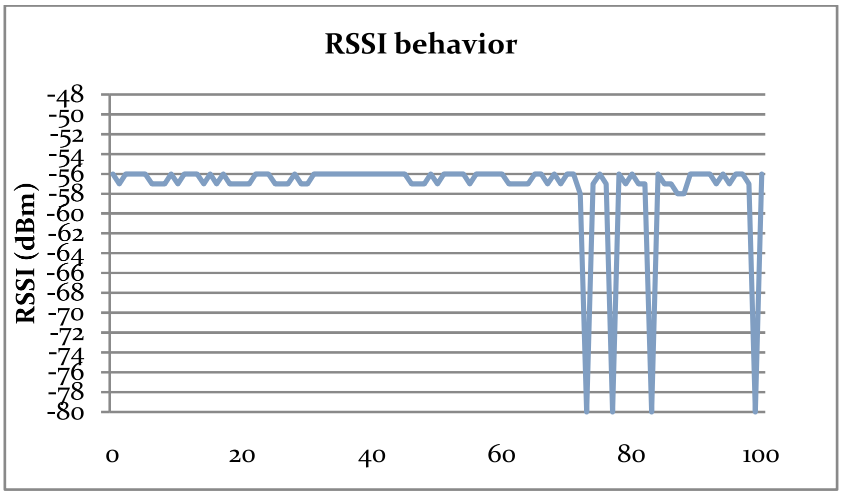


Figure 56. RSSI behavior in a 100-sample test run.

Finally, we placed different obstacles between the transmitter and the receiver in order to map out the effects on the RSSI behavior. The signal strength behavior was as nondeterministic as was expected on the first test due to multipath propagation.

Path Lost Prediction Model: RSSI versus distance

The majority of localization algorithms that do not make use or do not need to know the exact position of a sensor on the environment make use of a signal propagation model that maps RSSI values to estimate the distance [29]. The most widely used signal propagation model is the lognormal shadowing model:

$$RSSI(d) = P_T - P_L(d0) - 10\eta log\frac{10d}{d0} + X_\sigma$$

Equation 14. Lognormal shadowing model

where, P_T is the transmit power, $P_L(d0)$ is the path loss for a reference distance do, η is the path loss exponent and $X\sigma$ is a Gaussian random variable with zero mean and σ^2 variance, that represents the random variation of the RSSI value (multiplicative range error in dBm). Since the location sensor using RSS does not know the exact value of η, and X is a random variable, the calculated distances from this model are not very reliable.

Nevertheless, we tested if there is any relation between distance and RSSI values in our system. Taking measurements with a single pair of nodes, with the receiver and the transmitter on the same plane, we were able to isolate the effects of orientation and calibration. Figure 57 shows the RSSI versus Distance plots. Based on our measurements in the office, RSSI changes linearly with the log of the distance despite some outlier values probably occurred due to reflections.

Figure 57 shows the relation for a fixed orientation setting the transmitting power at 0 dBm (maximum) and mapping out at different distances. Note that sometimes the RSSI data obtained at the receiver cannot be used to infer any distance information. The reason is that significantly different distances can produce the same or almost the same RSSI values. In addition, small variations of the distances correspond to very different RSSI values (up to 11 dBm). In Figure 58, we obtained a more accurate linearity setting the transmitting power at -15 dBm with, obviously, slighter power sets.

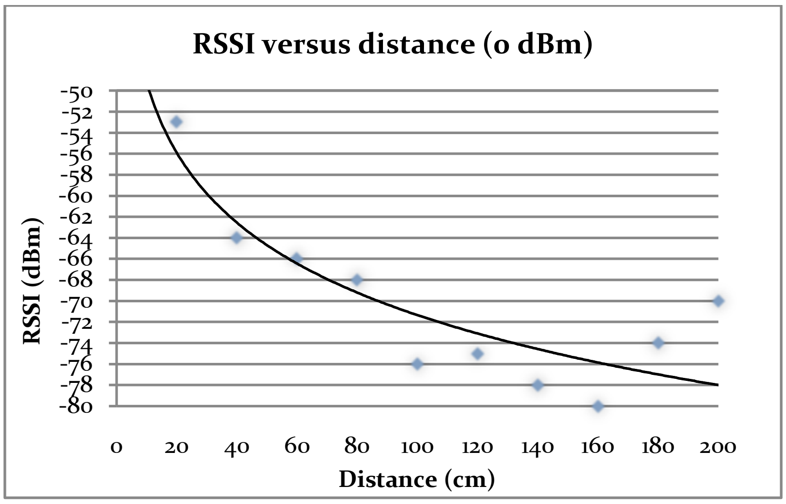


Figure 57. RSSI versus distance (0dBm)

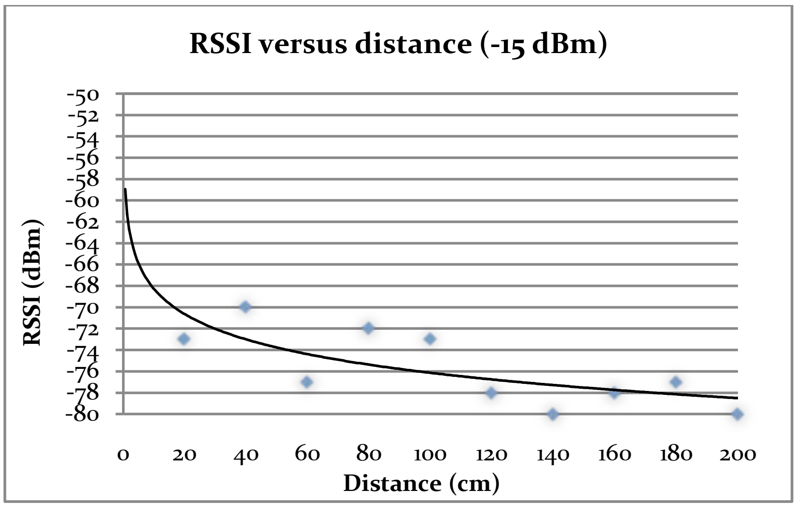


Figure 58. RSSI versus distance (-15dBm)

6.2.3. Time Of Arrival (TOA)

Time Of Arrival (TOA) is the measurement of the transmission and the propagation signal from the transmitter to the receiver. The propagation time is equal to the distance between the transmitter and receiver divided by the propagation velocity. This speed for RF is defined as 106 times faster than the sound speed. Nevertheless, this estimation is in free space, but the propagation velocity is different when the radio signal passes through other materials.

The receiver must find the first arriving peak of the signal, because there is no guarantee that the line-of-sight (LOS) signal will be the strongest of the arrival signal.

The main errors calculating the TOA when the modules are separated by long distance are the multipath and the additive noise. The TOA estimation is the cross-correlation between the multipath received signals and the known transmitted signal. The two main problems in the TOA estimation are when many multipath signals arrive to the receiver, and they made difficult to find the peak from the LOS signal; and when the LOS signal is attenuated compared with the multipath signals causing an error in the TOA estimation. Another issue of TOA is that it requires clock synchronization between the transmitter and the receiver, which requires a high accuracy of the algorithms implemented.

In any case, the error in the TOA estimation is different between indoor and outdoor environments. The error is bigger in indoor than outdoor, due to there are more obstacles that generate multipath.

The distinct advantage of measure TOA between nearby modules is that the LOS signal power increases when the path length decreases [30].

6.2.4. Angle Of Arrival (AOA)

Angle Of Arrival (AOA) estimates the angle of arrival of the signal. This measurement provides localization information complementary to the TOA and RSS described before. It requires multiple antennas (at least two) that contribute to higher cost and larger device size. The major sources of error are additive noise, multipath and the orientation of the sensor.

There are different ways to measure the AOA, but the common one involves measuring the differences in arrival times from the transmitter until the receiver antennas or measuring the difference of signal strength power received in each antenna.

In Figure 59 is shown that the times of arrival from the transmitter until the two antennas of the receiver are different, and is possible to estimate the position of the transmitter. In addition, if the transmitter uses a directional antenna, the power received in each antenna of the receiver is different, and is possible to estimate the position of the transmitter.

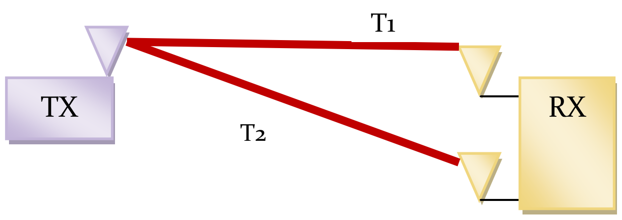


Figure 59. AOA

6.3. Localization algorithms

The design philosophy behind modular robots is that each module is very simple, as we already know. Indeed, a module by itself cannot achieve much, but modules arranged together in a global system can achieve complex tasks such as manipulation and locomotion. Similarly, the control of a single module is usually simple whereas controlling a system of many modules becomes difficult. As we have seen, different control architectures for local and global communication have been implemented for the overall system.

Control architectures can be implemented in either a centralized or decentralized way. In most cases it is easier to develop and analyze a centralized approach. However, the advantage of decentralized control architectures is that computation is shared among the modules since no single unit needs to do all the heavy computation. This is also meant to be more robust and easily lends itself to scale the robot to large numbers of modules. On the other hand, it is easier to implement centralized control to achieve global communication and decentralized to achieve local.

Until this point, models for measurements and the performance of some of them have been presented without discussing any module location estimator. This chapter will introduce a brief review of the existent literature about localization algorithms in order to acquire a good background and learn about relevant techniques and methods that could be useful to solve or not our specific location detection goal: neighboring localization.

6.3.1. Overview of localization algorithm research

The literature in sensor cooperative localization algorithms is extensive and continuously growing. Signal processing, statistics, and computer science communities have published extensively in this area.

To achieve cooperative localization between modules, there is a need to extend existing methods by finding different ways to use the range and angle measurements between pairs of unknown-location modules. The challenge is to allow sensors to be located (sensors that are not in range of any known-location device) and further, to improve the location estimation of all modules. If each module is in range of multiple reference nodes, each sensor's location could be calculated directly and independently.

As mentioned earlier, localization algorithms can be generally divided into centralized algorithms, which collect measurements at a central processor before calculation, and distributed algorithms, which require the modules to share information only with their neighbors, possibly iteratively.

6.3.2. Centralized algorithms

A centralized control system is the one that has a main controller (e.g. the brain) central in location and gathers all the data and measurements at the same location to perform a posteriori calculation.

All the models measurements presented in the previous chapter are suitable to be used in a centralized system. If all the data measured is known to be described well by a particular statistical model (i.e. Gaussian or log-normal) then the system can process these measurements to obtain a solution. This could be applied to the RSS, AOA and TOA cases.

When we discovered that signal strength measurements were heavily affected by multiple factors, a fixed environment was tested. This means that when using RSS or TOA methods, all the values obtained from every module change in a different environment, producing a non-versatile and non-reliable system.

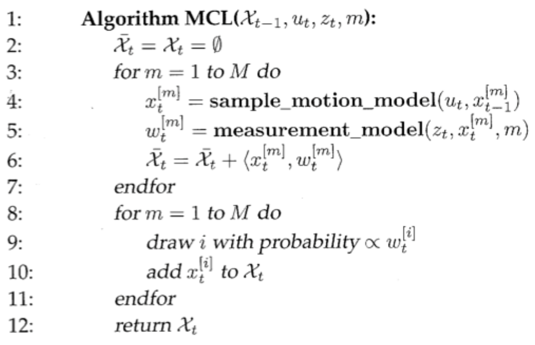
Regarding to the AOA measurement method, this technique involves a previously knowledge of the position of at least two fixed or anchor modules as well as the distance between them, in order to locate the desirable module. In other words, the method triangulates the information to estimate the module localization. Therefore, the position of these modules should be fixed in the robot system. Another option is to provide the anchor modules with a gyroscope or compass to constantly know their position and as a result of this, the rest of the modules positions.

6.3.3. Distributed algorithms

A decentralized control system is the one in which the controller elements are not central in location but are distributed all over the system with each component subsystem controlled by one or more controllers.

There are two big motivations for developing distributed localization algorithms. First, for some systems, no central processor (or none with enough computational power) is available to handle the calculations. Second, when a large network of sensors must forward all measurement data to a single central processor, there is a communication bottleneck at and near the central processor. That is, centralized algorithms in large networks require each sensor's measurements to be passed over many hops to a central processor, while distributed algorithms have sensors send messages only one hop (but possibly make multiple iterations).

One of the main researches done in distributed algorithms for robot location is Monte-Carlo estimation methods. Although the application of these methods is normally placed on the centralized algorithm's group, these particle filtering methods provides us a distribute architecture to work on accurate mobile modules localization. Concretely, and focusing to Monte-Carlo Localization (MCL) method, each module store a set of "particles" (i.e. random candidate representations of its coordinates) scattered in a configuration space and weighted according to their likelihood. At each iteration on the update phase of the algorithm, the likelihood that each hypothetical configuration is correct is updated based on some statistical model as shown in Figure 60 (e.g. a Bayes theorem). When the probability of a hypothetical configuration becomes very low, it is replaced with a new set of random particles. These methods have been used to accurately locate mobile robots [31] [32].



```
Algorithm MCL($\mathcal{X}_{t-1}, u_t, z_t, m$):
    $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
    for $m = 1$ to $M$ do
        $x_t^{[m]}$ = sample_motion_model($u_t, x_{t-1}^{[m]}$)
        $w_t^{[m]}$ = measurement_model($z_t, x_t^{[m]}, m$)
        $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
    endfor
    for $m = 1$ to $M$ do
        draw $i$ with probability $\propto w_t^{[i]}$
        add $x_t^{[i]}$ to $\mathcal{X}_t$
    endfor
    return $\mathcal{X}_t$
```

Figure 60. MCL pseudo code algorithm [30]

Regarding to the thesis goal, this approach could be seen as a good way to keep investigating. However, we found some hardware limitations that make Monte-Carlo estimation methods not useful; during the algorithm update phase, when obtaining the new samples from the weighted set, extra sensor information is needed in order to know if the module to localize is moving and in which direction. Although the goal is to localize the module, we need to know the behavior of it by using different sets of sensors. In neighbor localization, this extra information is reduced to provide the module with an accelerometer or gyroscope. The first extra feature will provide information about the movement of the module we are locating. Although this could be done by RSS model measurements to determinate the distance, we found both the same problems of unknowing in which direction the module was moving. In addition, we already demonstrated RSS measurements are not so precise even in controlled environments.

6.3.4. Discussion

In this section we got into centralized and decentralized localization algorithms. We presented measurements based on statistical models like TOA, AOA, and RSS, and non deterministic models based on localization estimation, like MCL. We used them to help us in our thesis scope and generate localization performance bounds. Such bounds are useful to know all the different approaches we have, among other design considerations, and to compare among localization algorithms.

The methods explained above showed us that our system has to carry with some hardware limitations due to the thesis objective. These limitations can be solved by the usage of extra features like accelerometers and gyroscopes as we explained above but, as a challenge and compared to previous work, we kept working with those limitations to come out with a low-cost and effective solution.

6.4. Neighbor localization

Neighbor localization means to search the neighbors of each module to discover the topology of the robot and provide the necessary movement to develop the task it should implement.

In the previous chapter we analyzed three possible localization methods, concluding that the use of these methods suppose the addition of extra hardware in each module - like a gyroscope- and incurring an extra cost. However, we kept working on the challenge of a low-cost and effective solution. With this limitation, the technology that seems to be useful for our hardware is the RSS. Therefore, in the next sections we test if RSS can be used to the neighbor localization problem. The experiments are divided in three tests: the first one tries to analyze the optimal configuration to localize the neighbors with RSS. With a defined configuration, the second experiment tries to localize the modules changing the output power of the transmitter and the distance of the receiver from the transmitter. The unsuccessful results caused the implementation of the last experiment, where we try to map out the neighbors using two antennas in the same board. Finally, we analyze and discuss the results of each experiment.

6.4.1. Experiments using RSSI I: Localize the module

6.4.1.1. Architecture overview

The radiation pattern of the omnidirectional antenna designed and implemented on the *Communication Board* is not a perfectly circular. However, we can use this characteristic of the antenna along with the signal strength variation to check if it is possible to figure out the position of a neighbor in order to localize it.

This experiment can be divided in two parts. The first part will test if exist an optimal configuration of the modules (orientation) to know where they are placed according with the RSSI value obtained. Once one configuration is defined, the second part of the experiment will test if it is possible to localize the neighboring modules by changing the distance between them and the output power level of the transmitter.

To do that, we will measure the power strength variation in each 4-possible positions (the 4 cardinal points respect the transmit module at the center) and with all 4-possible orientations of the receiver respect to the transmitter. We will deploy the modules on a square lattice of 16 cm squares. Each module will have one radio and will be allocated at the same fixed channel as its neighbors. The module acting as a transmitter is placed in the center of the grid, and the receiver is placed around the transmitter, in the four possible positions. A *sniffer* is also used to collect the measurements. All the possible positions of the receiver are shown in Figure 62.

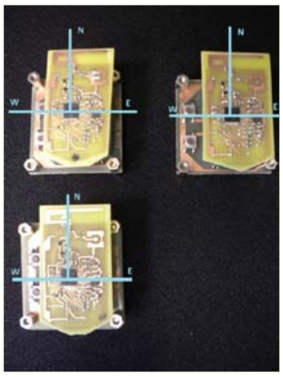


Figure 61. Configuration of the modules: antenna looking to the north

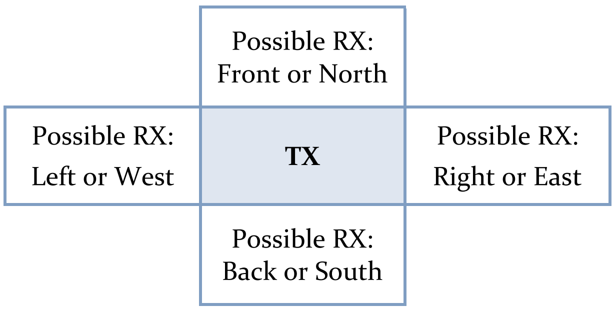

Possible RX: Front or North
Possible RX: Left or West
TX
Possible RX: Right or East
Possible RX: Back or South

Figure 62. Configuration of possible modules connection

All the measurements were done deploying the modules on a big table on our office. As we already know, if the environment changes, the RSSI value too. This means that when the modules are placed on the floor of the office, the RSSI values received were reduced due to the reflections and this must be aware of that when we are measuring.

6.4.1.2. Experiment setup

In this section we discuss the hardware and the software that is used in the experiments.

6.4.1.2.1. Hardware selection

The hardware used for this experiment is composed by three modules: a transmitter, a receiver and a sniffer. All this modules are composed of one PCB omnidirectional antenna, DN007 reference design.

6.4.1.2.2. Software

The software used is implemented on the IEEE 802.15.4 stack. Each radio used is setup either as a transmitter or a receiver as well as a sniffer using the same software. This means that we have one application for all the modules instead of having one for each function; we only have to specify the module's function and it will execute its functionalities. The purpose of the software is to gather all the information needed for the experiments between transmitter and receiver modules.

The transmitter on startup begins sending beacon packets to a broadcast address, meaning that they are readable for all the transmitters listening at channel 26 (2480 GHz). The beacons are sent every 128 ms at configurable power strength. The receiver, instead, on power up begins listening for any beacon. Once the receiver hears a beacon it gathers all the information needed for the experiments (from RSSI values or timing until channel information, among others) and sends it by a data packet over channel 25 (2475 GHz) to the sniffer module.

The sniffer starts up hearing at channel 25 for data packets to, afterwards, print the packet information through a serial port to a monitoring computer.

Finally, the onboard LEDs were used to verify the transmission and reception, to identify the role of each module during the tests and for debugging purposes. Also, the memory usage of the software used on this test was very optimized (8.4 Kbytes of ROM and 1.3 Kbytes of RAM).

6.4.1.3. Experiments and results

RSSI versus distance

For the first test, the variation of the power received respect with the distance between transmitter and receiver is measured. The transmit module sends different output power levels while the receiver is moving around it in the four possible positions and with different distances from the transmitter. We do that by leaving the receive module in his measurement position until its RSSI gets stable and more or less constant. The results are shown in Table 6, Table 7 and Table 8 and they are represented in Figure 63 and Figure 64. Each table shows, for both 0 and -15 dBm power strength signals sent by the transmitter, the signal strength obtained by the receiver in each position (with the same orientation), around the transmit module (noted with an X) and a different distances.

At 0 dBm of transmit power, we can see that, at 5 cm, 15 cm, and 30 cm there are 11 dBm, 9 dBm and 9 dBm respectively of difference when the receiver is placed on the right or on the left side of the transmitter. Even, when the power level is decreased (-15 dBm), the difference between both sides is evident. These are good results in order to use RSSI to localize the module. However, we noticed that at higher distances, this difference becomes imperceptible due to the radiation pattern of the antenna that is not completely circular. This causes that, the saturation levels (when the antenna is at the limit of reception) are closer on the left and right sides that the north and south. In addition, there is not too much difference when the receiver module is placed on the right side of the transmitter and when it is placed in front of it, also caused by the not perfect radiation pattern of the antenna.

Regarding to the other cardinal points, the differences are 3 dBm, 5 dBm, and 4 dBm at 5 cm, 15 cm, and 30 cm respectively, when the receiver module is placed on the south or on the north side of the transmitter and it is using a 0 dBm power signal.

TX power	RX received power (dBm)		
0 dBm		-28	
	-39	**X**	-28
		-31	
-15 dBm		-55	
	-66	**X**	-60
		-59	

Table 6. Power strength when the receiver is placed at 5 cm from the transmitter.

TX power	RX received power (dBm)		
0 dBm		-42	
	-49	**X**	-40
		-37	
-15 dBm		-68	
	-73	**X**	-67
		-64	

Table 7. Power strength when the receiver is placed at 15 cm from the transmitter.

TX power	RX received power (dBm)		
0 dBm		-49	
	-60	**X**	-69
		-45	
-15 dBm		-77	
	-81	**X**	-80
		-68	

Table 8. Power strength when the receiver is placed at 30 cm from the transmitter.

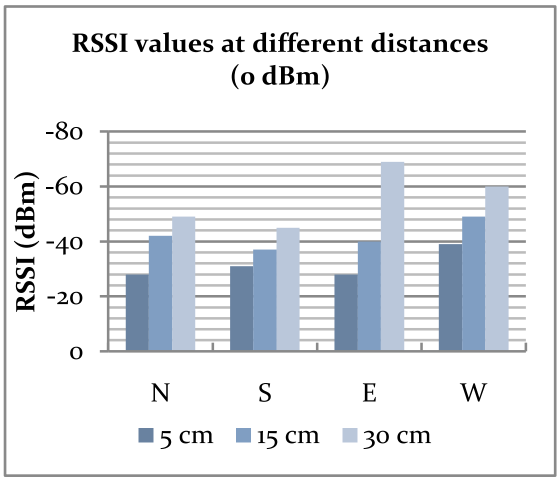


Figure 63. RSSI values at different distances (0dBm)

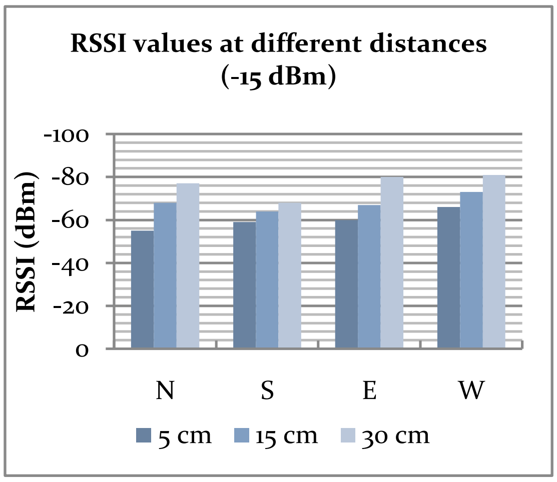


Figure 64. RSSI values at different distances (-15dBm)

From the results, we can see that the lower power level we use, the less difference between RSSI values we get. The omnidirectional antenna is not perfect, but is possible that changing the configuration of the modules (orientation) or the polarity of the antenna, we get different levels of signal to figure out where is the neighbor.

RSSI versus angle variability

The second experiment quantifies the variability among different antenna orientation of a pair-wise module communication. Using the transmitter module at the same position with the same antenna orientation, we changed the receiver's antenna orientation in order to map out the radio signal strength variation. The TX module was transmitting at -15 dBm in four different orientations (0, 90, 180 and 270 degrees). The receiver was fixed at 5 cm from the transmitter and also tested at the same four different orientations. Figure 65 shows the RSSI values obtained at the receiver for all four-orientation combination.

For each transmitter's orientation, the RSSI average value and its standard deviation were computed, obtaining: -7,21 dBm for 0º, -4,77 dBm for 90º, -4,36 dBm for 180º and 4,14 dBm for 270º. Averaging all the averaged standard deviations for all different orientations, we obtain that the overall standard deviation of the RSSI value obtained is 5,12 dBm. These results give us valuable information to use in the next experiments and in order to choose the best configuration of the modules (orientation).

As is shown in Table 9 different antenna configurations were tested and the best candidate is the one who has the greater standard deviation. In addition, we must be aware of the scalability of the modular system. It means that, in order to make the system scalable, the modules should be configured as the central one. Therefore, the configuration with the greater standard deviation, the one with all the modules with the antenna looking to the north, is the best choice.

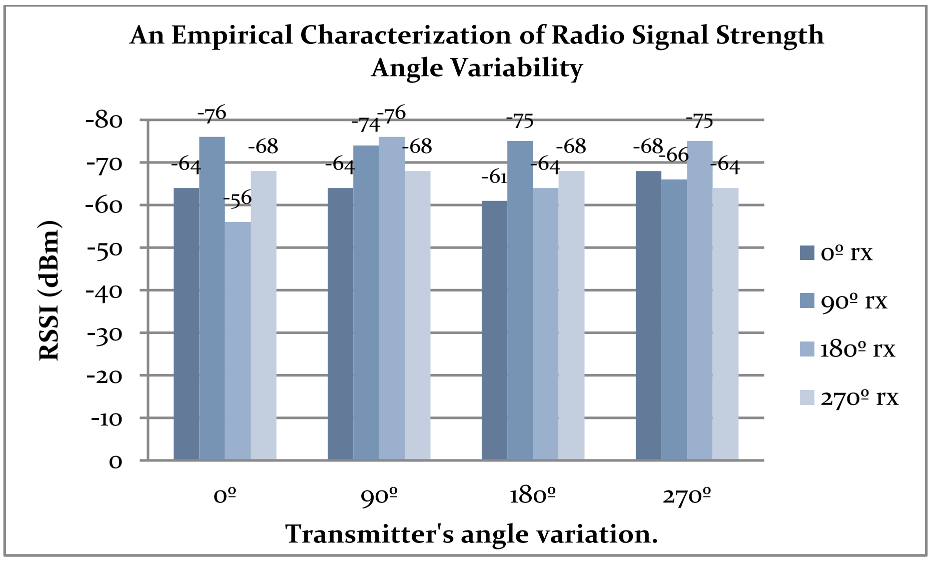


Figure 65. Characterization of Radio Signal Strength Angle Variability

Once the configuration is chosen, we measure the maximum distance that the receiver can receive the signal when the transmitter is transmitting at different output power levels and orientations. We do that to get another approach to the localization problem using the different power sets of the TI CC2420 device. This will give us more chance to localize the module in combination with the RSSI results obtained above, which is not very conclusive.

When the transmitter module uses -25 dBm of power, the maximum distance where the receiver module receives signal is at 4 cm, placed at the north side of the transmitter. At the rest of the sides, the receiver is not receiving signal. However, when the module transmits at -15 dBm, the maximum distance that the other module receives signal is at 120 cm at north, 77 at south, 34 at east and 39 at west.

	Mean	Standard deviation
0º tx	-66 dBm	-7,21 dBm
90º tx	-70,5 dBm	-4,77 dBm
180º tx	-67 dBm	-4,36 dBm
270º tx	-68,25 dBm	-4,14 dBm

Table 9. RSSI mean and standard deviation regarding to the transmitter orientation.

Due to the big distance hops between the two lowest power levels of the TI CC2420 radio chip, we cannot use these results for any of our experiments. However, the transceiver has smaller steps of output power in higher levels: 0 dBm, -1 dBm and -3 dBm. Therefore, in the next experiment we modify the module's antenna to use the higher levers of power strength and improve the distance results of this experiment.

6.4.2. Experiments using RSSI II: Localize the module

This experiment proposes to erase the antenna to increase the output power range in order to analyze if, together with the results gathered from the last experiment and the measurement of the module´s distance, the neighbor localization is suitable.

6.4.2.1. Architecture overview

As a result that we cannot use the signal strength variation with distance because of the big difference between the lowest power levels in the transceiver, another solution is implemented and tested in this experiment. We deteriorate the performance of the antenna to reduce the power strength transmitted by the transceiver. There are two different solutions to accomplish that:

a) *Erase the printed antenna*: the antenna is design to work at 2,4GHz with the best adaptation. If we modify or erase the printed antenna, the adaptation will be worse and therefore, the power will be lose by heat in the circuit, and less power will be transmitted.
b) *Design an attenuator on the Communication Board*: with only three resistances is possible to attenuate the power of the signal. We can use an attenuator of -10 dBm in each module, so when the module transmit 0dBm, the other module will receive -20 dBm (-10dBm in the TX module -10 dBm in the RX module).

The best solution is to implement an attenuator into the *Communication Board*, but this solution supposes a new hardware design and implementation of the board. Therefore, we chose to deteriorate the pad to connect the directional antenna of the *Communication Board*.

Hence, as concluded in the previous experiment, the configuration of the modules the one that has all the modules with the antenna looking to the north as shown in Figure 61.

6.4.2.2. Experiment setup

In this section we discuss the hardware and the software that is used in the experiment.

6.4.2.2.1. Hardware selection

The hardware used for this experiment is composed by three modules: a transmitter, a receiver and a sniffer. Both the transmitter and receiver are composed of one PCB omnidirectional antenna with the modified design. The sniffer uses the normal PCB omnidirectional design.

6.4.2.2.2. Software

The software used is in this experiment is the same used in the last one because the purpose is to test a new module doing the same experiments to, finally compare the results gathered.

6.4.2.3. Experiments and results

Erasing the antenna, the transmission power is reduced. Thus, with -25 dBm of transmission power and the modules without antenna, the receiver must be placed at 0 cm from the transmitter to receive the signal. When the power transmitted is increased, is possible to start using the RSSI values in combination with distances between modules. Therefore, we placed the transmitter in the middle of the grid defined in Figure 62 and with the receiver we calculate the maximum distance from the transmitter where the module still receives signal.

Table 10 shows that there is not relationship between the power and the maximum distance from the receiver. When the power transmitted is -15 dBm, the maximum distance on the right, left and front sides is similar (8 cm), while when the power transmitted is -10 dBm, the maximum distance is different on the front (13 cm) and right (34 cm) sides as well as the left and back (23 cm) sides. The reason of this behavior is the lack of antenna.

In any case, we tried to define the size of the module in order to map out the neighbors of the module, transmitting different power levels.

TX power	Saturation distance (cm)		
-10 dBm		13	
	24	X	34
		23	
-15 dBm		8	
	9	X	8
		3	

Table 10. Maximum distance of the receiver from the transmitter, where the module still receive signal

We defined the distance between the transmitter and the four possible neighbors according with the results of Table 10. The distance between the modules connected to the right or left of the transmitting module is 26 cm, and the modules connected in the front or back of the transmitting modules is 7 cm. In this case, checking when the power transmitted is -15 dBm, only the module placed in the front of the transmitting module can receive the signal. Figure 66 shows the distance defined from the transmitter to the receivers and the red line shows the range calculated in Table 10 when the TX module transmits at -15 dBm.

When the power transmitted is -10 dBm, only the modules placed at the right and back can receive the signal. Figure 67 shows the modules which receive signal (with the aim of the red line) according to Table 10 (the red RX module is already detected). We can see that we have two possible options, where we should check the power received in both sides, and see if it is possible to figure out where is placed using the RSSI threshold values from the first RSSI experiment. The power received when the module is transmitting -10 dBm is shown in Table 12.

The power difference between the module placed in the right and in the back is approximate 8 dBm. Therefore, we can use a RSSI threshold for example, in order to know where the receiver is (in our case, a -75 dBm threshold is appropriate).

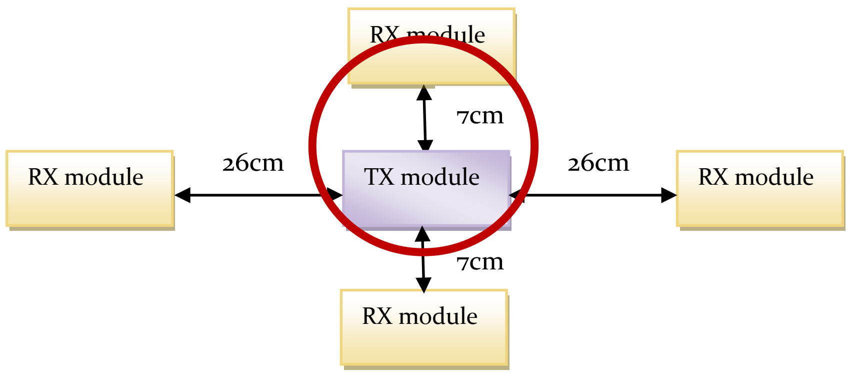


Figure 66. Strength signal received in the defined configuration

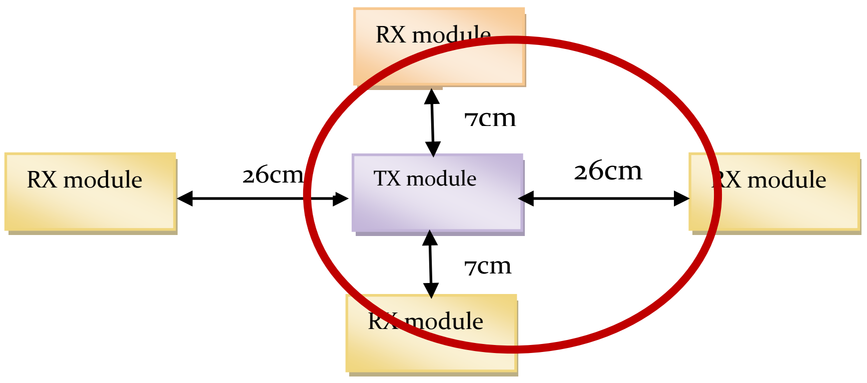


Figure 67. Strength signal received in the defined configuration when

The next step is to increase more the power. When the power transmitted is -7 dBm, all the neighbors received the signal. However, we have only two possible positions where the module can be: left or back, and left or right. The power received when the module is transmitting at -7 dBm is shown in Table 11.

The power difference between the modules of the two possible options left (left and back sides) is only 3 dBm. It is a very small difference, which it is not enough to differentiate the sides to know where the module to be detected is.

TX power	RX received power (dBm)		
-7 dBm		-66	
	-62	X	-65
		-65	

Table 11. Strength power received when the TX module transmit -7dBm

TX power	RX received power (dBm)		
-10 dBm		-74	
	-71	X	-72
		-79	

Table 12. Strength power received when the TX module transmit -10dBm

As a conclusion, with the designed hardware is not possible to localize the modules limiting the distance between neighbors and changing the output power transmission. We saw that in the last two steps, the difference between RSSI values was very small. However, these measurements are not reliable. The performance of the *Communication Board* without antenna is unpredictable meaning that, we should try again with a known antenna and with a transceiver with more output power levels.

6.4.3. Experiments using RSSI III: Localize the module

This experiment proposes the implementation of a directional and omnidirectional antenna in the same board. In the following points, we explain the architecture of the experiments and the results.

6.4.3.1. Architecture overview

Our modules are provided with one PCB omnidirectional antenna. As demonstrated in early experiments, using the signal power strength and distances between the modules we cannot determinate how to localize the neighbors. This motivates us to look for other solutions.

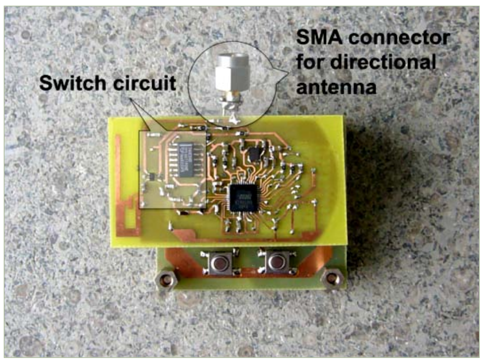


Figure 68. Redesign *Communication Board*

One approach is to use four directional antennas, one on each side of the *Communication Board*. However, it supposes the utilization of too many antennas, increase the size and the hardware complexity of the module, and an extra antenna to provide global communication.

Another possible configuration is use an omnidirectional and directional antenna in the same board. The directional antenna could be used to localize the neighbor and the omnidirectional antenna for the local and global communication. Assuming that the modules can rotate the directional antenna (e.g., using a gear), each module could start as a transmitter using the directional antenna and check the power received in each possible neighbor position, while the neighbors are working as a receiver with the omnidirectional antenna switched on.

In theory, a well designed directional antenna detects more signal in the direction where is looking at. Only some lobes at sides and back can appear, as it was seen at the antenna measurements on the chapter 3. Using the module configuration choice from the last experiment and with an ideal directional antenna, each module can know the position of their neighbors depending on the reception or not of the signal. However, an ideal antenna does not exist and it is necessary to test the new configuration. In addition, we used an easy and cheap antenna design that will be enough to simulate the conditions explained above.

In this experiment, we implemented one module with the new antenna design. The new design permits to switch between both antennas by using a switch circuit controlled by software. The new *Communication Board* design is shown in Figure 68 and the schematic can be seen in Appendix F. The other modules placed around, were configured as a receivers with the normal PCB ominirectional antenna design.

6.4.3.2. Experiment setup

In this section we discuss the hardware and the software that is used in the experiment.

6.4.3.2.1. Hardware selection

The hardware used for this experiment is composed by six modules: a transmitter, four receivers and a *sniffer*. The transmitter is composed of one new *Communication Board* design with one omnidirectional and one directional antennas. The receivers and the *sniffer* use the normal PCB omnidirectional design.

6.4.3.2.2. Software

The software used is implemented on the IEEE 802.15.4 stack. Each radio used is setup either as a transmitter or a receiver as well as a sniffer using the same software. This means that we have one application for all the modules instead of having one for each function; we only have to specify the module's function and it will execute its functionalities. The purpose of the software is to gather all the information needed for the experiments between transmitter and receiver modules and control the antennas on the *Communication Board* by using the switch circuit.

The transmitter on startup begins sending beacon packets to a broadcast address using the omnidirectional antenna, meaning that they are readable for all the transmitters listening at channel 26 (2480 GHz). The beacons are sent every 128 ms at configurable power strength. The receiver, instead, on power up begins listening for any beacon. Once the receiver hears a beacon, it responds to it and the application starts. The transmitter switch to the directional antenna and sends 1000 data packets to the receiver (with information about the actual orientation among others), which keeps averaging the data until the transmitter simulates a rotation of the directional antenna. This rotation is indicated by the onboard LEDs and we have to change the orientation of the module manually.

The receiver gathers all the information needed for the experiments (from RSSI values or timing until channel information, among others) and sends it by a data packet over channel 25 (2475 GHz) to the sniffer module. The sniffer starts up hearing at channel 25 for data packets to, afterwards, print the packet information through a serial port to a monitoring computer. When the transmitter passed by all the 4 cardinal points (possible neighbor situations), the RSSI averages values are compared, and the possible neighbor position is obtained. We use the LEDs in order to know when the application finished.

The memory usage of the software used on this test was 9.8 Kbytes of ROM and 1.8 Kbytes of RAM.

6.4.3.3. Experiments and results

To prove the neighbor localization, we first measure the signal strength received (RSSI values) on the receiver module placed on each position around the transmitter (front, back, left and right) on a square lattice configuration and with a distance resolution of 10 cm until reach the 50 cm. These measurements will give us a good approach to verify the directivity of the antenna and know from which distance the values become not useful. As shown in Figure 69, the experiment is done placing 4 receivers around the transmitter but our system is able to detect one new neighbor at the same time. This means that only one neighbor connection can be done during each neighboring detection process.

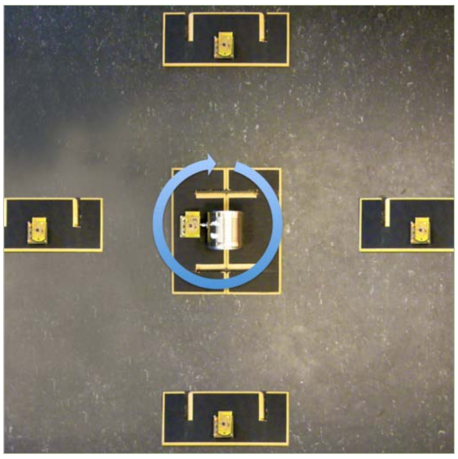


Figure 69. Deployment of the modules. The center module turns itself to detect neighbors.

On the other hand, the central module (the transmitter) sends the packets using the maximum power strength signal (0 dBm) in order to focus all the power on the module to localize. To collect the RSSI values from the receiver, the sniffer is configured at different channel than the one used on the transmitter-receiver communication to avoid interferences, in order to gather the data measurements from the receiver. In Figure 70 it is shown the RSSI values obtained from each receiver situated at North, South, West and East, at different distances and when the transmitter's directional antenna is locking to the North. From the values, we can see that the more distance between the transmitter and the receiver we have, the more useful are the results. With 10 cm of distance, the difference of power strength between all the module positions is very small. This is produced by the secondary lobes from the directional antenna.

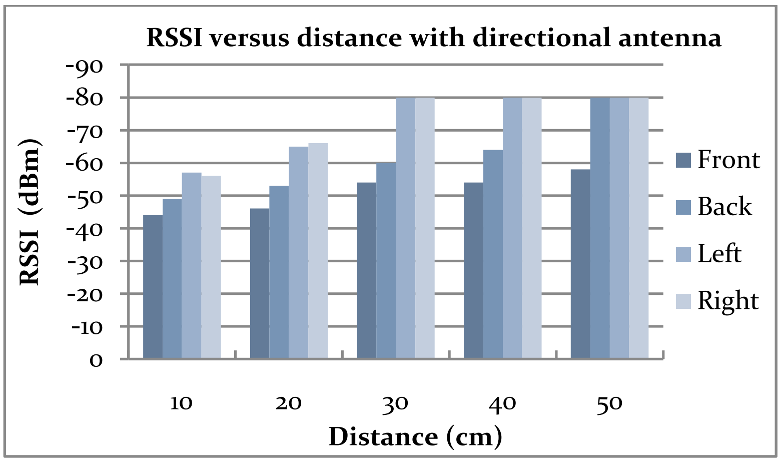


Figure 70. RSSI values obtained at different distances with the central node looking at the North. A value of -80 dBm means that there is no communication between the transmitter and the receiver.

Looking at the front and back sides, we can see that the difference of power strength is smaller than the left and right sides. This occurs again because of the directional antenna, which has bigger secondary lobes at the rear side. As we can see, transmitting at maximum power the difference between the front and the back sides at 40 cm is only 9 dBm, but at 50 cm, the directivity of the antenna becomes ideal (the other modules are not receiving).

Finally, as we already measured on the first RSSI experiment, the angle variation of the module produces a deviation of -7.21 dBm, meaning that it will affect on the RSSI values obtained from the receiver when the transmitter rotates the antenna to start the localization.

However, the gain of the open-waveguide antenna is very low (4 dBm), and consequently the main lobe of the directional antenna is almost the same than the omnidirectional one, which its gain is around 1 dBm in the XY plane. As a consequence, when the directional antenna is pointing to a module the strength power received varies few dBms than when we are transmitting with the omnidirectional antenna. According to that, even with RSSI values with a difference of -7.21 dBm, the central module will detect the highest signal coming from the front and therefore, localize the neighbor.

After testing the directivity of the directional antenna in a real-world test, we can conclude that the design must be improved. The distances where the directivity is useful are very high and become useless for modular robots. However, it is demonstrated that the directivity can be used for neighbor localization but it must be designing a better directional antenna.

6.5. Conclusion

We described three different types of measurements useful for localization algorithms: Received Signal Strength (RSSI), Time Of Arrival (TOA) and Angle Of Arrival (AOA). TOA is less sensitive to increases in distances among sensors, making this technology more suitable for low-density networks. AOA (as well as TOA) can achieve higher accuracy than RSS, but incurring in higher device size and cost (AOA needs multiple antennas). They need to know the position of two modules to triangulate and calculate the position of the third one, restraining the position of two modules in the robot or adding a compass in each module to know its relative position. RSS is attractive for low-cost deployments of denser networks with low accuracy requirements. However, these measurements have environment-dependant problems (path loss, multipath, additive noise, shadowing, etc.), and depends of the system architecture (module orientation and number of antennas).

The architecture defined in a modular robot makes impossible to use some of these measurements and makes difficult the implementation of localization algorithms, basically due to the continuously changing environment that modular robots are involved. Thus, the measurement that best fits on modular robots is RSS. For this reason, we implemented some measurements on our system in order to find a solution for neighbor localization problem. The measurements evaluated the relationship between distance and RSS, and determined the relationship between RSS and module´s orientation to, finally, define the best configuration of the *Communication Board*.

The results show that RSS is not suitable for neighboring localization with the aim of only one omnidirectional antenna in the *Communication Board*. In addition, the qualitative analysis of the relationship between RSS and distance demonstrated that RSS is extremely environment dependant and it cannot give us enough information to solve the localization problem. However, we successfully demonstrated that with the aim of a directional antenna plus the omnidirectional one on the same *Communication Board* is possible to solve the neighbor localization problem. The module is able to detect the position of a new neighbor that wants to join to the module in one of its 4 sides –north, south, left and right– using the directional antenna like the RADAR. The omnidirectional antenna is used by the module that has to be detected.

Therefore, with this configuration the modules can localize its neighbors, communicate to each other locally or globally depending on the task to develop, and without increasing too much the cost of the module by the use of only two antennas. However, as a future work, we propose to improve the design of the directional antenna due to its dimensions and its radiation pattern.

As an unexpected lesson, we learned the value of a packet sniffer in developing and debugging code and during the run of the tests.

7. Conclusions and future work

This thesis has explored the suitability of implementing Radio Frequency for modular robots. The main objective of the project was to demonstrate if RF is a scalable and robust communication system as an alternative to the most used hardwired and Infrared (IR) connection methods. In addition, we explored and experimented if RF can be used to solve local and global communication, and localization problems.

According to that, we designed and implemented what is called a *Communication Board* that uses the powerful Atmel AT91SAM7 microcontroller and comprise a TI CC2420 radio chip making use of a PCB antenna. Some of the reasons of why we mainly chosen this hardware where explained but as we were going deep into the thesis and running the experiments, many questions regarding the hardware selection came across.

One of the biggest problems we found out is that the 2,4GHz band frequency in which the CC2420 device works is too much noisy and full of interferences. Several devices work at that frequency (i.e. all WI-FI devices) and generate interferences. However, there is not a large variety of commercial devices which work at higher frequencies. Because of that, one solution may be to design an extra circuit to change the operating frequency to a free one with less interference. One reasonable candidate could be the free frequency 5,4 GHz where not too many kind of systems use it (e.g. RADARs). In addition, the usage of a higher frequency will incur on a smaller antenna dimensions.

Another problem was the antenna design. The directional antenna with the can design, was not good enough. However, if the circuit in charge of changing the operating frequency of the system is implemented, the dimensions of the antenna will be reduced making it suitable for modular robots in terms of size and performance. In addition to that and as a alternative solution, we were not able to find on the market a suitable antenna design to work with, but some research designed a small (4 x 4.4 cm) adjustable directivity antenna for biomedical radar applications [33].

Regarding to the PCB omnidirectional antenna, we explored that the radiation pattern depends on the orientation of the module. At some point, this characteristic was an advantage and made us work hard on exploiting its benefits applying it to solve the localization problem. However, it was not good enough to solve the problem mainly due to the busy 2.4 GHz band and that every antenna has a very different radiation pattern (probably because of the manufacturing process). Therefore, if we do not want to depend of the module´s configuration, we should use another kind of antenna like a monopole's model.

Keeping on the hardware selection discussion and regarding to the TI CC2420 device, we noticed that, although it was not an inconvenient at first point, we had problems with the only 8 steps of output power levels of the device. Specifically and when we tried to use the lowest levels, we found a big step between -25 dBm and -15 dBm, and between -15 dBm and -10 dBm, which is reasonably big to be a solution for RSSI distance and measurements. Consequently, in a future design another transceiver which uses more steps of output power levels should be chosen. The TI CC2500 transceiver seems to be a good solution if we do not need to use *ZigBee* and we do not care about implementing DSSS by software.

Once the hardware was chosen, we worked on adapting and implementing all the software need to communicate a pair of modules. The choice of the lightweight TinyOS operating system was good enough because it is designed for ultra low-power wireless sensors and provides a set of important services and abstractions to work on implementing network applications. In spite of that, many problems were found trying to use all the TinyOS advantages and a new redesigned HAA was build making use of some predefined TinyOS structures. Thus, a device driver was implemented to support the IEEE 802.15.4 MAC protocol and make the system compliant to the standard (indeed needed by the hardware). This provided us a completely control in implementing only the needed features of the radio stack and designing a communication protocol optimized for our system but compliant with the standards at the same time.

Finally, we explored the local and global communication, the neighbor exploration and the localization topics, discussing and arguing about potential solutions to solve the problems and if they are suitable or not by means of experiments.

The main conclusion we drawn from the experiment results is that our system is relatively simple, low-cost (the estimated prize of the *Communication Board* is around 13 €), low-power (consuming around 284.95 mW at maximum load) and robust.

We successfully demonstrated that the RF communication solves the problems of misalignment in IR communication, and we obtained good results implementing and demonstrating the suitability of RF in local and global communication in modular robot's context. However, for localization we had more problems to find out a solution. Nevertheless, the results we obtained from the experiments demonstrate that using a directional and an omnidirectional antenna in the same board is possible to localize the neighbors. The transmitter use the directional antenna to focus in a specific area to search the strength power received from a neighbor, which works with the omnidirectional antenna. The system works similar as RADAR, meaning that the transmitter moves around itself in steps of 90 degrees searching for all its possible connected neighbors.

The tests we made to the system explained above gave us sufficient results to demonstrate that is a suitable solution for neighboring detection. However it needs a more deep analysis of performance in different situations. One experiment could the incorporation of more transmit and receive modules to our system to simulate a more realistic environment, what would introduce an increased crosstalk and multipath problems. Another one could be to check more accurately the influence generated by the directional antenna to the omnidirectional one. That means testing exactly the behavior of the PCB omnidirectional antenna while having a directional one at the same board.

8.

References

1. **Kasper Stoy, David Brandt and David J. Christensen.** *Self-Reconfigurable Robots: An introduction (Intelligent Robotics and Autonomous Agents series).* s.l. : The MIT Press, 2010. 0262013711; Pages 27-49 and 86-89.

2. **Mark Yim, Ying Zhang and David Duff.** Modular Robots. When a task or terrain varies, reconfigurable robots can change their shape to get the job done. *IEEE Spectrum.* 2002; Pag. 31.

3. **Mark Yim, Wei-Min Shen, Behnam Salemi, Daniela Rus, Mark Moll, Hod Lipson, Eric Klavins and Gregory S. Chirikjian.** Modular Self-Reconfigurable Robot Systems. Challenges and Opportunities for the Future. *IEEE Robotics & Automation Magazine.* 2007. Page 45.

4. *Self-Reconfigurable robots. Shape-Changing Cellular Robots Can Exceed Conventional Robot Flexibility.* **Kurokawa, Satoshi Murata and Haruhisa.** s.l. : IEEE RoboticsAutomation Magazine , 2007. Pages 71-75.

5. **Nielsen, Jacob.** *User-Configurable Modular Robotics. Design and use.* Odense : The Maersk Mc-Kinney Moller Institute. University of Southern Denmark, December 2008. Chapter 2.

6. **Lal, Robert Fitch and Ritesh.** *Experiments with a ZigBee Wireless Communication System for Self-Reconfiguring Modular Robots.* University of Sydney, NSW Australia : s.n., 2009. s. 1947.

7. **Fitch, Victor Kuo and Robert Charles.** *A Multi-Radio Architecture for Neighbor-to-Neighbor Communication in Modular Robots.* University of Sydney, NSW, Australia : 2011 IEEE International Conference on Robotics and Automation, 2011.

8. **Mayné, Jordi.** *Estado actual de las Comunicaciones por Radio Frequencia.* s.l. : Silica, 2009. Pages 3-5; 8-10; 21-41.

9. **Miron, Douglas B.** *Small Antenna Design (Communications Engineering).* s.l. : Newnes (March 7, 2006), 2006. 0750678615. Chapter 1.

10. **Dimitrios Lymberopoulos, Quentin Lindsey and Andreas Savvides.** *An Empirical Characterization of Radio Signal Strengt Variability in 3-D IEEE 802.15.4 Networks Using Monopole Antennas.* Yale University, New Haven, CT 06520, USA : Embedded Networks and Applications Lab, ENALAB.

11. **Instrument, ChipCon Products from Texas.** CC2420.2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver. [Online] 2006. http://www-mtl.mit.edu/Courses/6.111/labkit/datasheets/CC2420.pdf.

12. **A.Thiele, Warren L. Stuzman and Gary.** *Antenna Theory and Design.* s.l. : John Wiley & Sons , 1981. 047104458X. Pages 17-40 and 397-421.

13. Antenna Selection Quick Guide. DN035. [Online] 2010. http://focus.tij.co.jp/jp/lit/an/swra351/swra351.pdf.

14. **Andersen, Audun.** Texas Instrument. AN043. [Online] 2008. http://focus.ti.com/lit/an/swra117d/swra117d.pdf.

15. Design note DN007. 2,4 GHz Inverted F Antenna . [Online] 2008. http://focus.ti.com/lit/an/swru120b/swru120b.pdf.

16. **Mezquida, Carlos.** *Disenio y optimización de una antena impresa para Wireless LAN.* Valencia : Universidad Politécnica de Valencia, 2004. Pages 40-42, 49-50 and 71-72.

17. **Wade, Paul.** *Practical Microwave Antennas Part 1.* s.l. : ARRL, 1994. Årg. UHF/Microwave Projects Manual, Volume 2. ISBN 0-87259-631-1. Pages 1-5 and 14.

18. Modos de propagación en guías de onda circulares. [Online] http://www.lu3hba.com.ar/ARTICULOS%2010/Modos%20de%20propagaci%C3%B3n%20en%20gu%C3%ADas%20de%20ondas%20circulares.pdf.

19. **Balanis, Constantine A.** *Antenna Theory: Analysis and Design, 3rd Edition.* s.l. : Wiley-Interscience; 3 edition (April 4, 2005), 2005. 047166782X. Pages 683-693.

20. **Gay, Philip Levis and DAvid.** *TinyOS programming guide.* 2009.

21. **Society, IEEE Computer.** *Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs).* New York, NY 10016-5997, USA : s.n., revision 2003.

22. **Willig, Holger Karl and Andreas.** *Protocols and Architectures for Wireless Sensor Networks.* s.l. : Wiley-Interscience, 1 edition (October 30, 2007). 0470519231.

23. **Li, C. Blake, D. De Couto, H.I. Lee and R. Morris.** *Capacity of ad hoc wireless networks.* s.l. : In Proc. of MobiCom, 2001. Pages 61-69.

24. **Fitch, V.Kuo and R.** *Towards a parallel wireless radio communication architecture for modular robots.* s.l. : in Proc. of ARAA Australasian Conference on Robotics and Automation (ACRA), 2009.

25. **David Johan Christensen, David Brandt, Ulrik Pagh Schultz and Kasper Stoy.** *Neighbor Detection and Crosstalk Elimination in Self-Reconfigurable Robots.* University of Southern Denmark, Odense, Denmark : Proc. of RoboComm, 2007. Pages 1-8.

26. **Atmel.** AT91SAM7S256 datasheet. *ATMEL.* [Online] 2007. http://www.atmel.com/dyn/resources/prod_documents/doc6175.pdf. Pages 560-562.

27. **Rappaport, T.** *Wireless Communications: Principles and Practice (2nd Edition).* NJ, USA : Prentice Hall PTR, Upper Saddler River, 2001. 0130422320. Pages 138-140.

28. **Levesque, Kaveh Pahlavan and Allen H.** *Wireless Information Networks (Wiley Series in Telecommunications and Signal Processing).* s.l. : Wiley, 2 edition (September 26, 2005). 0471725420. Pages 17-18.

29. **Rapport, Scott Y. Seidel and Theodore S.** *914MHz path loss prediction model for indoor wireless communication in multifloored buildings.* s.l. : IEEE Transactions on Antennas and Propagation, 40 (2), February 1992. Pages 207-217.

30. **K. Pahlavan, P. Krishnamurthy, and J. Beneat.** *Wideband radio propagation modeling for indoor geolocation applications.* s.l. : EEE Comm. Magazine, April 1998.

31. **Thrun, S.** *Probabilistic robotics.* Communications of the ACM, vol. 45, no. 3, pp. 52–57, March 2002.

32. **Sebastian Thrun, Wolfram Burgard and Dieter Fox.** *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents series).* s.l. : The MIT Press (August 19, 2005). 9780262201629. Pages 250-267.

33. **U. Schwarz, M. Helbig, J. Sachs, f. Seifert, R. Stephan, F. Thiel and M.A. Hein.** *Physically small and adjustable double-ridge horn antenna for biomedical UWB radar applications.* Hannover (Germany) : IEEE ICUWB 2008, 2008.

34. **Ricardo Franco Mendoza, Kasper Stoy, David Johan Christensen and Andreas Lyder.** *A self-Reconfigurable Communication Network for Modular Robots.* Odense, Denmark : The Maersk Mc-Kinney Moller Institute. University of Southern Denmark, 2007.

9.

Appendices

9.1. Appendix A

In this appendix is shown the measurements of the AN043 reference design. One antenna is fixed while the other one is routing from -90° to +90° in 10° steps, with respect to the fixed antenna. The measurement oscillates, so we took five measurements in each step, and then the average will be calculated.

Degrees	Mes1	Mes2	Mes3	Mes4	Mes5	Average
+90	-15,78	-15,83	-15,86	-15,84	-15,86	-15,834
+80	-16,21	-15,81	-15,87	-15,89	-15,82	-15,92
+70	-16,52	-16,41	-16,57	-16,63	-16,58	-16,542
+60	-17,21	-16,94	-17,12	-16,93	-16,87	-17,014
+50	-15,97	-15,99	-16,02	-16,23	-16,31	-16,104
+40	-15,79	-15,71	-15,75	-15,8	-15,83	-15,776
+30	-15,61	-15,59	-15,65	-15,72	-15,67	-15,648
+20	-15,51	-15,53	-15,49	-15,55	-15,48	-15,512
+10	-15,29	-15,31	-15,41	-15,32	-15,21	-15,308
+0	-15,58	-15,6	-15,42	-15,48	-15,57	-15,53
-10	-15,19	-15,16	-15,27	-15,09	-15,16	-15,174
-20	-15,51	-15,49	-15,55	-15,47	-15,45	-15,494
-30	-15,89	-15,87	-15,79	-15,77	-15,76	-15,816
-40	-15,72	-15,69	-15,65	-15,59	-15,61	-15,652
-50	-15,42	-15,38	-15,35	-15,41	-15,37	-15,386
-60	-15,26	-15,3	-15,33	-15,27	-15,35	-15,302
-70	-15,57	-15,55	-15,61	-15,57	-15,62	-15,584
-80	-15,73	-15,77	-15,68	-15,73	-15,75	-15,732
-90	-15,83	-15,79	-15,86	-15,83	-15,81	-15,824

Table 13 . Measurements of the radiation pattern of the AN043

9.2. Appendix B

In this appendix is shown the measurements of the DN007 reference design. One antenna is fixed while the other one is routing from -90° to +90° in 10° steps, with respect to the fixed antenna. The measurement oscillates, so we took five measurements in each step, and then the average will be calculated.

Degrees	Mes1	Mes2	Mes3	Mes4	Mes5	Average
+90	-15,67	-15,49	-15,52	-15,48	-15,56	-15,54
+80	-16,21	-15,81	-15,87	-15,89	-15,82	-15,92
+70	-16,03	-16,11	-16,01	-16,13	-16,09	-16,07
+60	-16,02	-15,94	-15,89	-15,93	-15,87	-15,93
+50	-15,95	-15,86	-15,87	-15,96	-15,93	-15,91
+40	-15,84	-15,27	-15,38	-15,27	-15,26	-15,4
+30	-15,56	-15,57	-15,46	-15,42	-15,47	-15,49
+20	-15,49	-15,46	-15,49	-15,47	-15,43	-15,46
+10	-15,3	-15,42	-15,41	-15,44	-15,36	-15,38
+0	-15,38	-15,4	-15,37	-15,39	-15,36	-15,38
-10	-15,19	-15,16	-15,27	-15,09	-15,16	-15,17
-20	-15,82	-15,84	-15,86	-15,77	-15,79	-15,81
-30	-16,13	-15,97	-15,87	-15,84	-15,82	-15,92
-40	-15,7	-15,668	-15,73	-15,42	-15,59	-15,62
-50	-15,37	-15,11	-15,21	-15,23	-15,27	-15,23
-60	-15,24	-15,27	-15,26	-15,27	-15,31	-15,27
-70	-15,53	-15,67	-15,73	-15,71	-15,67	-15,66
-80	-15,87	-15,93	-15,88	-15,92	-15,91	-15,9
-90	-15,81	-15,75	-15,71	-15,87	-15,8	-15,78

Table 14. Measurements of the radiation pattern of the DN007

9.3. Appendix C

In this appendix is shown the measurements of the Biquad. Table 15 is shown the measurements when both antennas have the same polarization. One Biquad is fixed while the other one is routing from -90° to +90° in 10° steps, with respect to the fixed antenna. The measurement oscillates, so we took five measurements in each step, and then the average will be calculated.

Degrees	Mes1	Mes2	Mes3	Mes4	Mes5	Average
+90	-29,178	-31,928	-31,789	-32,088	-32,037	-31,404
+80	-28,25	-28,164	-28,167	-28,312	-28,271	-28,2328
+70	-27,013	-26,215	-26,272	-26,241	-26,221	-26,3924
+60	-25,778	-26,223	-26,059	-25,94	-25,918	-25,9836
+50	-20,345	-20,362	-20,378	-20,351	-20,301	-20,3474
+40	-15,669	-15,693	-15,479	-15,576	-15,594	-15,6022
+30	-12,907	-12,854	-12,812	-12,78	-12,784	-12,8274
+20	-11,139	-11,126	-11,129	-11,13	-11,114	-11,1276
+10	-11,043	-11,157	-11,139	-11,092	-11,114	-11,109
+0	-11,752	-11,731	-11,729	-11,713	-11,449	-11,6748
-10	-12,519	-12,524	-12,52	-12,518	-12,498	-12,5158
-20	-13,631	-13,614	-13,62	-13,579	-13,642	-13,6172
-30	-15,427	-15,507	-15,492	-15,526	-15,475	-15,4854
-40	-17,492	-17,532	-17,512	-17,444	-17,527	-17,5014
-50	-21,09	-20,86	-20,482	-20,148	-20,347	-20,5854
-60	-23,987	-24,117	-23,993	-24,174	-23,957	-24,0456
-70	-27,013	-27,083	-26,784	-26,715	-26,667	-26,8524
-80	-29,042	-28,707	-28,927	-29,183	-29,002	-28,9722
-90	-33,736	-31,771	-32,253	-31,423	-32,332	-32,303

Table 15. Biquad measurements with the same polarization

The Table 16 is shown the measurements when both antennas have different polarization. The fixed Biquad is in the vertical position while the horizontal Biquad route from -90° to +90° in 10° steps, with respect to the fixed antenna. The measurement oscillates, so we take five measurements in each step, and then the average will be calculated.

Degrees	Mes1	Mes2	Mes3	Mes4	Mes5	Average
+90	-35,777	-35,78	-35,936	-34,877	-34,662	-35,4064
+80	-27,915	-27,896	-28,844	-28,976	-29,444	-28,615
+70	-23,891	-24,958	-25,295	-25,195	-25,114	-24,8906
+60	-20,623	-20,392	-20,4	-20,327	-20,436	-20,4356
+50	-21,591	-21,642	-21,68	-21,582	-21,664	-21,6318
+40	-21,683	-21,668	-21,63	-21,612	-21,637	-21,646
+30	-22,467	-22,587	-22,481	-22,32	-22,541	-22,4792
+20	-26,187	-27,771	-27,635	-27,793	-27,778	-27,4328
+10	-30,837	-30,587	-29,779	-28,158	-28,936	-29,6594
0	-27,814	-28,135	-27,987	-27,016	-27,93	-27,7764
-10	-26,012	-25,946	-25,947	-25,505	-25,572	-25,7964
-20	-22,301	-22,115	-22,96	-22,479	-22,137	-22,3984
-30	-21,013	-20,936	-21,112	-20,793	-20,527	-20,8762
-40	-19,336	-19,214	-19,077	-19,361	-19,18	-19,2336
-50	-19,387	-19,436	-19,224	-19,34	-19,41	-19,3594
-60	-20,47	-20,387	-20,447	-20,486	-20,47	-20,452
-70	-21,668	-21,735	-21,55	-21,632	-21,739	-21,6648
-80	-21,389	-20,847	-20,735	-21,394	-20,908	-21,0546
-90	-23,217	-23,248	-23,214	-23,046	-23,308	-23,2066

Table 16. Biquad measurements with different polarization

9.4. Appendix D

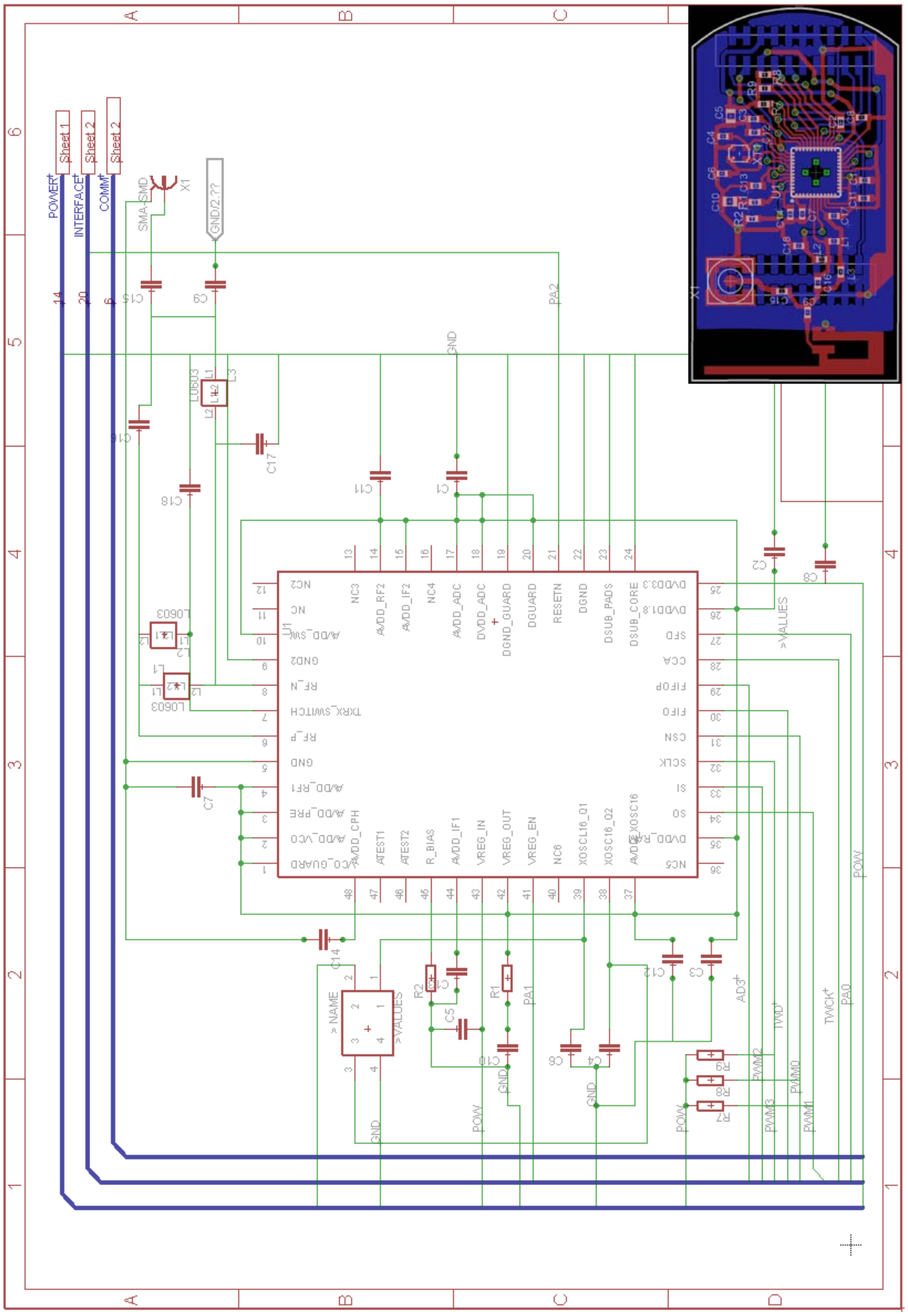
In this appendix is shown the measurements of the circular open-waveguide. Table 17 is shown the measurements when one antenna is routing from -90° to +90° in 10° steps respect to the fixed circular waveguide. The measurement oscillates, so we take five measurements in each step, and then the average will be calculated.

Degrees	Mes1	Mes2	Mes3	Mes4	Mes5	Average
+90	-44,462	-44,87	-44,704	-44,638	-44,615	-44,6578
+80	-41,57	-41,328	-41,41	-41,51	-41,623	-41,4882
+70	-39,668	-39,082	-39,406	-39,326	-39,37	-39,3704
+60	-37,117	-36,759	-37,156	-36,89	-36,62	-36,9084
+50	-34,714	-34,622	-34,592	-34,57	-34,419	-34,5834
+40	-32,658	-32,919	-32,791	-32,731	-32,69	-32,7578
+30	-31,403	-31,51	-31,426	-31,53	-31,308	-31,4354
+20	-30,319	-30,647	-30,306	-30,678	-30,515	-30,493
+10	-29,783	-29,819	-30,003	-29,955	-30,042	-29,9204
0	-28,419	-28,669	-28,62	-28,702	-28,753	-28,6326
-10	-28,218	-28,423	-28,419	-28,374	-28,39	-28,3648
-20	-28,096	-28,47	-28,79	-28,641	-28,652	-28,5298
-30	-28,849	-29,217	-29,23	-29,127	-29,227	-29,13
-40	-29,875	-30,045	-29,915	-29,879	-29,942	-29,9312
-50	-29,919	-30,076	-30,119	-30,126	-30,113	-30,0706
-60	-31,27	-31,339	-31,409	-31,392	-31,37	-31,356
-70	-33,012	-33,042	-32,938	-32,917	-32,945	-32,9708
-80	-34,715	-34,792	-34,689	-35,326	-35,219	-34,9482
-90	-39,121	-38,987	-39,142	-39,02	-38,956	-39,0452

Table 17. Measurements of the radiation pattern of a waveguide

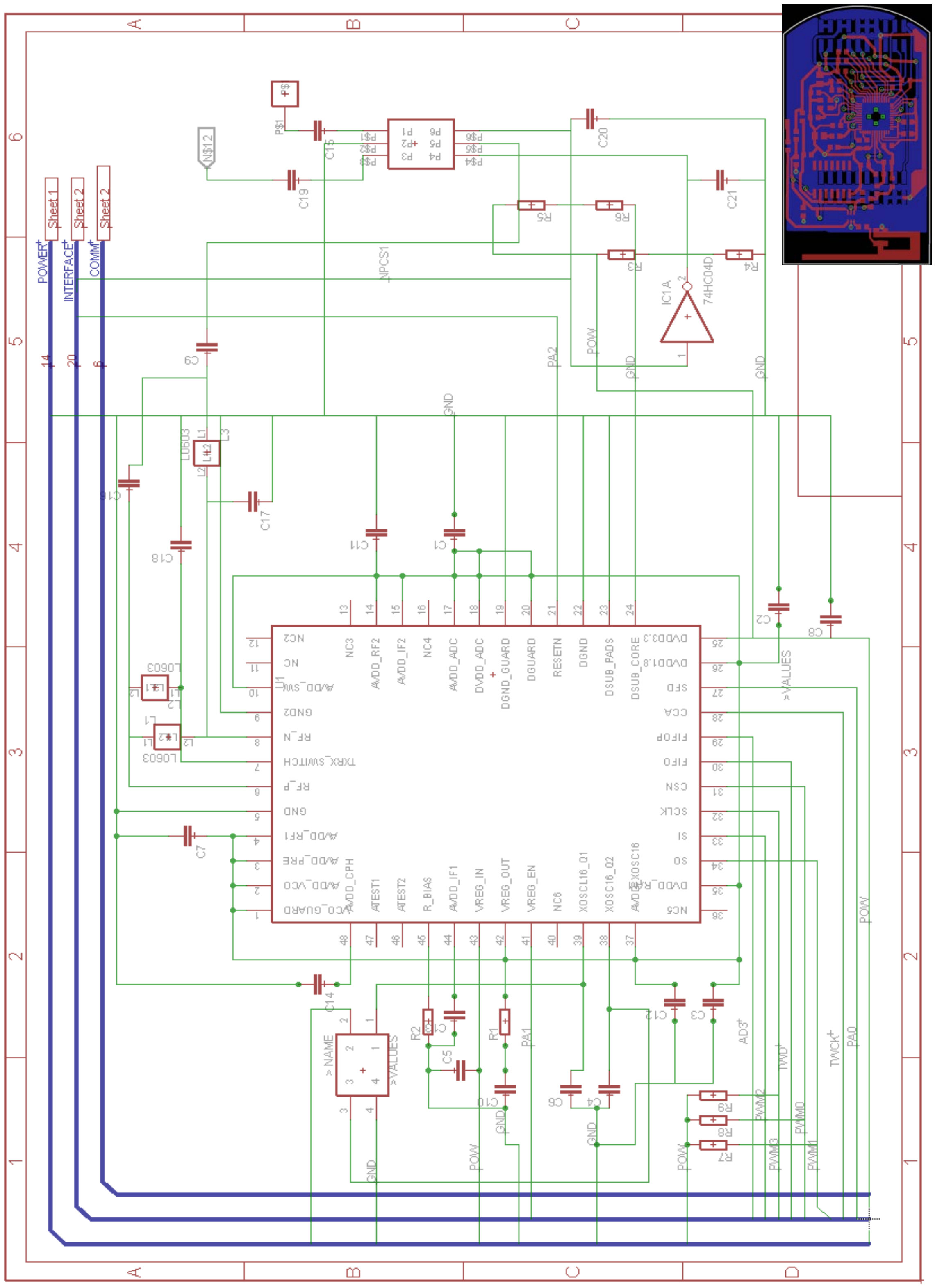
9.5. Appendix E

Schematic of the first prototype of the *Communication Board*.



9.6. Appendix F

Schematic of the second prototype, with two antennas in the same board. The components added are the S2PT and the inverter 74HC04D.



9.7. Appendix G

Code of the three *CC2420ModuleC* applications, Transmit and Receive functions (*CC2420TransmitP* and *CC2420ReceiveP*) and Transceiver's setup (*HplCC2420SpiP*).

```
#include "dbgu.h"
#include "Timer.h"
#include "RF.h"
#include "lib_AT91SAM7S256.h"
#include "Ieee154.h"

module CC2420ModuleC {

  uses interface HplAt91Spi as HplSpi;
    uses interface SplitControl as CC2420Control;
    uses interface HplCC2420Spi as CC;
    uses interface CC2420ActiveMessage as CC2420AM1;
    uses interface Timer<TMilli> as Timer0;
    uses interface Leds;
    uses interface Boot;

}

implementation {

  uint8_t counter = 0;
  uint32_t rtct = 0;
  cc2420_packet_tx packet;

  typedef nx_struct radio_count_msg {
      nx_uint8_t counter;
      nx_uint16_t rssi;
      nx_uint16_t id;
  } radio_count_msg_t;

  /**
  * Signaled when the system is booted. It is responsible to start
  * the SPI interface and the TI CC2420 control registers.
  */
  event void Boot.booted() {

      call Leds.led0On();
      call Leds.led1On();
      call Leds.led2On();

      // Start the device
      call CC2420Control.start();
  }

  /**
  * Signaled in response to an accepted send request.
  *
  * @param  'cc2420_packet_tx* packet_tx' the packet which was submitted as a send
  request
  * @param  time a value indicating the transmitt time.
  * @param  result SUCCESS if it was sent successfully, FAIL if it was not.
  */
  event void CC2420AM1.sendDone(cc2420_packet_tx* packet_tx, uint32_t time, error_t
  result) {

    // If we are the transmitter, we check and print the transmit time.
    if (MOTE_TYPE == 0) {

      DBGU_PutChar('>');
      // To measure the throughput of the system
      rtct = (sizeof(radio_count_msg_t)*8)/((uint16_t)time);

      // rtct = Kbps
      DBGU_PutChar(rtct>>8);
      DBGU_PutChar(rtct);

    }

  }

  /**
  * Signaled when a packet is received.
  *
  * @param  'cc2420_packet_rx* packet_rx' the receied packet.
  * @param  rx_length  the length of the data region pointed to by payload.
  * @param  result  SUCCESS if it the reception was successfully, FAIL if it was
  not.
  */
```

```
event void CC2420AM1.receive(cc2420_packet_rx* packet_rx, uint8_t rx_length, error_t result) {

  int8_t rssi = 0;

  DBGU_PutChar('<');

  // If we are the transmitter or the sniffer, just print the RSSI value received over the air
  if ((MOTE_TYPE == 0) | (MOTE_TYPE == 2)) {

    DBGU_PutChar('*');
    //DBGU_PutChar('C');DBGU_PutChar(':');
    DBGU_PutChar((packet_rx->payload)[0]);
    DBGU_PutChar('I');DBGU_PutChar('D');DBGU_PutChar(':');
    DBGU_PutChar((packet_rx->payload)[4]);
    DBGU_PutChar('R');DBGU_PutChar('S');DBGU_PutChar('S');DBGU_PutChar(':');
    DBGU_PutChar((packet_rx->payload)[2]);

    rssi = (packet_rx->payload)[2];

    // Convert Ca2 negative number to positive and add 45 as offset (data sheet explanation)
    if ((rssi & 0x80) != 0) {

      rssi = ~(rssi) + 1;
      rssi = rssi & 0x7F;
      rssi = rssi + 45;
      DBGU_PutChar(rssi);
      DBGU_PutChar('-');
    }
    else {
      DBGU_PutChar('+');DBGU_PutChar('+');
    }

    DBGU_PutChar('*');
  }

  // If we are the receiver, when the beacon is received, just change the
  // channel and send the RSSI value over the air (for the sniffer).
  else if (MOTE_TYPE == 1) {

    radio_count_msg_t* rcm = (radio_count_msg_t*)(packet.payload);
    counter++;
    rcm->counter = counter;
    rcm->rssi = packet_rx->rssi;
    rcm->id = MY_ADDR;

    DBGU_PutChar('#');
    DBGU_PutChar(packet_rx->rssi);
    DBGU_PutChar('#');

    call CC.set_channel(25);
    delay_us(128);

    if (call CC2420AM1.send(IEEE154_BROADCAST_ADDR, &packet, sizeof(radio_count_msg_t)) == SUCCESS) {

      call Leds.led0Toggle();
      call CC.set_channel(26);
      delay_us(128);
      }
  }

  call Leds.led1Toggle();
}

/**
* Signaled when the SPI interface is initialized successfully. At the same time, it initializes
* the CC2420 control registers and checks if the initialization was successfull.
*
* @param  error  SUCCESS if it the initialization was successfully, FAIL if it was not.
*/
event void CC2420Control.startDone(error_t err) {

  if (err == SUCCESS) {

        // Init structures and interfaces
        if (call CC2420AM1.init() == SUCCESS) {

          if (MOTE_TYPE == 0) {

          // PA_LEVELS:    31, 27, 23, 19, 15, 11, 7, 3, 0
          // corresponds to 0, -1, -3, -5, -7, -10, -15, -25 dBm in the
          call CC.set_txpower(3);

          call Timer0.startPeriodic( 6000 );

          }
        else if (MOTE_TYPE == 2) {

          call Leds.led3On();
            call CC.set_channel(25);
        }
        }
        else {


          DBGU_PutChar('E');DBGU_PutChar('R');DBGU_PutChar('R');DBGU_PutChar('O');D
          BGU_PutChar('R');DBGU_PutChar('/');
          call Leds.led2Off();
        }
    }
    else {

      call CC2420Control.start();
    }
  }

  event void CC2420Control.stopDone(error_t err) {
    // do nothing
  }

  /**
  * Signaled when the timer expires (signaled periodically if the timer is called
  with
  * the startPeriodic caller.
  */
  event void Timer0.fired()
  {

    radio_count_msg_t* rcm = (radio_count_msg_t*)(packet.payload);

    counter++;
    rcm->counter = counter;
    rcm->rssi = 0;
    rcm->id = MY_ADDR;

    if (call CC2420AM1.send(IEEE154_BROADCAST_ADDR, &packet,
    sizeof(radio_count_msg_t)) == SUCCESS) {

        call Leds.led0Toggle();

      }
  }
} // end implementation
```

```
#include "dbgu.h"
#include "Timer.h"
#include "RF.h"
#include "lib_AT91SAM7S256.h"
#include "Ieee154.h"

module CC2420ModuleC {

    uses interface HplAt91Spi as HplSpi;
    uses interface SplitControl as CC2420Control;
    uses interface HplCC2420Spi as CC;
    uses interface CC2420ActiveMessage as CC2420AM1;
    uses interface Timer<TMilli> as Timer0;
    uses interface Timer<TMilli> as Timer1;
    uses interface Leds;
    uses interface Boot;

}

implementation {

  uint8_t counter = 0;
  uint32_t rtct = 0;
  cc2420_packet_tx packet;
  int16_t rssi = 0;
  uint8_t radar_check = 1;

  typedef nx_struct radio_count_msg {
      nx_uint8_t counter;
      nx_uint16_t rssi;
      nx_uint16_t id;
      nx_uint8_t detect;
  } radio_count_msg_t;

  /**
  * Signaled when the system is booted. It is responsible to start the the SPI interface
  * and the TI CC2420 control registers.
  */
  event void Boot.booted() {

      call Leds.led0On();
      call Leds.led1On();
      call Leds.led2On();

      // Start the device
      call CC2420Control.start();
  }

  /**
  * Signaled in response to an accepted send request.
  *
  * @param  'cc2420_packet_tx* packet_tx' the packet which was submitted as a send request
  * @param  time a value indicating the transmitt time.
  * @param  result SUCCESS if it was sent successfully, FAIL if it was not.
  */
  event void CC2420AM1.sendDone(cc2420_packet_tx* packet_tx, uint32_t time, error_t result) {

    // If we are the transmitter, we check and print the transmit time.
    if (MOTE_TYPE == 0) {

      DBGU_PutChar('>');
      rtct = (sizeof(radio_count_msg_t)*8)/((uint16_t)time);

      // rtct in Kbps
      DBGU_PutChar(rtct>>8);
      DBGU_PutChar(rtct);

    }
  }

  /**
  * Signaled when a packet is received.
  *
  * @param  'cc2420_packet_rx* packet_rx'  the receied packet.
  * @param  rx_length  the length of the data region pointed to by payload.
* @param  result  SUCCESS if it the reception was successfully, FAIL if it was not.
*/
event void CC2420AM1.receive(cc2420_packet_rx* packet_rx, uint8_t rx_length, error_t result) {

  //DBGU_PutChar('<');
  rssi = (packet_rx->payload)[2];

  // If we are the transmitter or the sniffer, just print the RSSI value received over the air
  if ((MOTE_TYPE == 0) | (MOTE_TYPE == 2)) {

    // If the connection between both modules is stablished, we start the neighbor detection process.
    if (packet_rx->payload[5] != 0) {

      call Timer0.stop();
      counter = 0;
      DIRECTIONAL_ON();

        call Leds.led0Off();
        call Leds.led1On();
        call Leds.led2On();
        call Leds.led3On();

        call Timer1.startPeriodic( 10000 );
        return;
    }

    DBGU_PutChar('*');
    DBGU_PutChar('I');DBGU_PutChar('D');DBGU_PutChar(':');
    DBGU_PutChar((packet_rx->payload)[4]);
    DBGU_PutChar('R');DBGU_PutChar('S');DBGU_PutChar('S');DBGU_PutChar(':');
    DBGU_PutChar((packet_rx->payload)[2]);

    // Convert Ca2 negative number to positive and add 45 as offset (data sheet explanation)
    if ((rssi & 0x80) != 0) {

      rssi = ~(rssi) + 1;
      rssi = rssi & 0x7F;
      rssi = rssi + 45;
      DBGU_PutChar(rssi);
      DBGU_PutChar('-');
    }
    else {
      DBGU_PutChar('+');DBGU_PutChar('+');
    }

    DBGU_PutChar('*');

  }
  // If we are the receiver, when the beacon is received, just change the channel and send the RSSI value over the air (for the sniffer).
  else if (MOTE_TYPE == 1) {

    radio_count_msg_t* rcm = (radio_count_msg_t*)(packet.payload);
    counter++;
    rcm->counter = counter;
    rcm->rssi = packet_rx->rssi;
    rcm->id = MY_ADDR;
    rcm->detect = radar_check;
    radar_check = 0;

    DBGU_PutChar('#');
    DBGU_PutChar(packet_rx->rssi);
    DBGU_PutChar('#');

    call CC.set_channel(25);
    delay_us(128);

    if (call CC2420AM1.send(IEEE154_BROADCAST_ADDR, &packet, sizeof(radio_count_msg_t)) == SUCCESS) {

      call Leds.led0Toggle();
      }

      call CC.set_channel(26);
      delay_us(128);

    }

    call Leds.led1Toggle();
  }

  /**
  * Signaled when the SPI interface is initialized successfully. At the same time,
  it initializes
  * the CC2420 control registers and checks if the initialization was successfull.
  *
  * @param  error  SUCCESS if it the initialization was successfully, FAIL if it
  was not.
  */
  event void CC2420Control.startDone(error_t err) {

    if (err == SUCCESS) {

        // Init structures and interfaces
        if (call CC2420AM1.init() == SUCCESS) {

          if (MOTE_TYPE == 0) {

          // PA_LEVELS:    31, 27, 23, 19, 15, 11, 7, 3, 0
          // correspond to 0, -1, -3, -5, -7, -10, -15, -25 dBm in the
          call CC.set_txpower(31);

          OMNIDIRECTIONAL_ON();

          call Timer0.startPeriodic( 10000 );

          }
        else if (MOTE_TYPE == 2) {

          call Leds.led3On();
            call CC.set_channel(25);
        }
        }
        else {


          DBGU_PutChar('E');DBGU_PutChar('R');DBGU_PutChar('R');DBGU_PutChar('O');D
          BGU_PutChar('R');DBGU_PutChar('/');
          call Leds.led2Off();
        }
    }
    else {

      call CC2420Control.start();
    }
  }

  event void CC2420Control.stopDone(error_t err) {
    // do nothing
  }

  /**
  * Signaled when the timer expires (signaled periodically if the timer is called
  with
  * the startPeriodic caller.
  */
  event void Timer0.fired()
  {

    radio_count_msg_t* rcm = (radio_count_msg_t*)(packet.payload);

    counter++;
    rcm->counter = counter;
    rcm->rssi = 0;
    rcm->id = MY_ADDR;
    rcm->detect = 0;

    if (call CC2420AM1.send(IEEE154_BROADCAST_ADDR, &packet,
    sizeof(radio_count_msg_t)) == SUCCESS) {

        call Leds.led0Toggle();
      }
  }

  /**
  * Signaled when the timer expires (signaled periodically if the timer is called
  with
  * the startPeriodic caller.
  */
  event void Timer1.fired()
  {

    radio_count_msg_t* rcm = (radio_count_msg_t*)(packet.payload);
    counter++;

    if (counter == 0x01) {
      call Leds.led0Off();
    }

    if (counter == 0x40) {
      call Leds.led0Off();
      call Leds.led1Toggle();
    }

    if (counter == 0x80) {
      call Leds.led1Off();
      call Leds.led2Toggle();
    }

    if (counter == 0xC0) {
      call Leds.led2Off();
      call Leds.led3Toggle();
    }

    if (counter == 0xFF) {
      call Leds.led3Off();
      call Leds.led0Toggle();
      call Leds.led1Toggle();
      call Leds.led2Toggle();
      call Leds.led3Toggle();
      call Timer1.stop();
      return;
    }

    rcm->counter = counter;
    rcm->rssi = rssi;
    rcm->id = MY_ADDR;
    rcm->detect = 1;

    if (call CC2420AM1.send(IEEE154_BROADCAST_ADDR, &packet,
    sizeof(radio_count_msg_t)) == FAIL) {
      DBGU_PutChar('E');DBGU_PutChar('R');DBGU_PutChar('R');DBGU_PutChar('!');
    }
  }
} // End implementation
```

```
/*** Test application for AT91SAM7S256 motes with CC2420 transceiver ***/
/*

This application tests de peripherial mounted on a general board that have an Atmel AT91SAM7256 microcontroller and communicate
the results of the test using the onboard leds.

The peripherial is a RF board that contains a CC2420 transceiver, an anthena and uses the SPI bus to communicate with the
microcontroller.

The purpose of this application is far away of testing the microcontroller but the erros on it can not be solver. It means that
it is supposed that the microcontroller board works.

The steps of the tester are the ones below:

1- Once the microcontroller finished the booting process, leds 0, 1 and 2 are switched off to warranty that the board is working.
   Led 3 will remain always switched on.
2- With the boot signal, the application configure, initialize and starts the SPI bus of the processor.
3- After setting up the SPI bus, the CC2420 Control registers are configured as well as all the interfaces and variables needed
   to send a short testing message over the radio.
4- If all goes well, a short testing broadcast message is sended every second over the radio, blinking leds 0 and 1 at the same time at every sent.
   If the sent did not work, both leds are switched on.

Legend of the on-board leds (looking at the board from the front):

    [0][1] [2][3]

Led 0: ON -> Peripherial not connected or not well connected, what it means that the crystal oscillator is not working.
             (returns an unknown status 0xFF).
Led 1: ON -> Crystal oscillator not working or not initialized properly (it never stabilizes, always returns status 0x00).
Led 2: ON -> Wrong initialization of the SPI bus. The program will retry indefinetely so the peripherial may work with this
       led switched on if the others are switched off.
Led 3: OFF -> Critical inicialization of the general board or microcontroller. Should be always switched on.

Leds 0 & 1 remains ON -> Message not sended.
Leds 0 & 1 remains OFF -> Crystal status incorrect because PLL is out of Lock.
Leds 0 & 1 toggling -> Message is sended.

*/

#include "dbgu.h"
#include "Timer.h"
#include "spi.h"
#include "lib_AT91SAM7S256.h"
#include "Ieee154.h"

module CC2420ModuleC {

    uses interface HplAt91Spi as HplSpi;
    uses interface SplitControl as CC2420Control;
    uses interface CC2420ActiveMessage as CC2420AM1;
    uses interface Timer<TMilli> as Timer0;
    uses interface Leds;
    uses interface Boot;
}

implementation {

  uint8_t dummy = 0;
  cc2420_packet_tx packet;

  typedef nx_struct radio_count_msg {
    nx_uint8_t dummy;
  } radio_count_msg_t;

  event void Boot.booted() {


      DBGU_PutChar('B');DBGU_PutChar('o');DBGU_PutChar('o');DBGU_PutChar('t');DBGU_
      PutChar('/');

      delay_ms(500);

      call Leds.led0On();
      call Leds.led1On();
      call Leds.led2On();

      // Starts SPI bus
      call CC2420Control.start();

  }

  event void CC2420AM1.sendDone(cc2420_packet_tx* packet_tx, error_t result) {}

  event void CC2420AM1.receive(cc2420_packet_rx* packet_rx, uint8_t rx_length,
  error_t result) {}

  event void CC2420Control.startDone(error_t err) {

    if (err == SUCCESS) {

        // Init structures and interfaces
        if (call CC2420AM1.init() == SUCCESS) {

        call Timer0.startPeriodic( 40000 );
        }
        else {


          DBGU_PutChar('E');DBGU_PutChar('R');DBGU_PutChar('R');DBGU_PutChar('O');D
          BGU_PutChar('R');DBGU_PutChar('/');
        }
    }
    else {

      call Leds.led2Off();
      call CC2420Control.start();
    }
  }

  event void CC2420Control.stopDone(error_t err) {
    // do nothing
  }

  event void Timer0.fired()
  {

    radio_count_msg_t* rcm = (radio_count_msg_t*)(packet.payload);
    rcm->dummy = 4;

    if (call CC2420AM1.send(IEEE154_BROADCAST_ADDR, &packet,
    sizeof(radio_count_msg_t)) == SUCCESS) {

        call Leds.led0Toggle();
        call Leds.led1Toggle();
      }
      else {

        call Leds.led0Off();
        call Leds.led1Off();
      }
  }

} // End of implementation

```

```
#include "AT91SAM7S256.h"
#include "dbgu.h"

module CC2420TransmitP @safe()
{

  provides interface StdControl as SubControl;
  provides interface CC2420Transmit;

  uses interface HplCC2420Spi as CC2420Spi;
  uses interface HplCC2420Interrupts as CC2420Interrupts;

}

implementation {

  uint32_t dummy, rtc, rtc2, rtct = 0;
  uint8_t locked, lock_on, lock_off;

  command error_t SubControl.start() {

  return SUCCESS;
  }

  command error_t SubControl.stop() {

  return SUCCESS;
  }

    /**
    * Send a packet over the air.
    * If send returns SUCCESS, then the component will signal the sendDone.
    * If send returns an error, it will not signal the event.
    *
    * @param 'cc2420_packet_tx* packet_tx'    the packet to send including all the
    data needed.
    * @param cca   indicates wether to use the clear channel assessment or not.
    * @return       SUCCESS if the request to send succeeded and a
    *               sendDone will be signaled later, FAIL if the communication
    layer is not
    *               in a state that can send or if there is any packet error.
    */
  command error_t CC2420Transmit.send (cc2420_packet_tx* packet_tx, bool cca)
  {

    uint8_t j;

    // Wait for any previous transmission to finish.
    while (call CC2420Spi.statusRead() & CC2420_STATUS_TX_ACTIVE);

    // Wait until the transceiver is idle.
    while (FIFOP_IS_1 | SFD_IS_1);

    rtc = AT91F_RTTReadValue(AT91C_BASE_RTTC);

      // Write packet to TX FIFO.
      call CC2420Spi.strobe(CC2420_SFLUSHTX);
      call CC2420Spi.strobe(CC2420_SFLUSHTX);

      call CC2420Spi.writeFIFO((uint8_t*)&packet_tx->length, 1);
      call CC2420Spi.writeFIFO((uint8_t*)&packet_tx->fcf, 2);
      call CC2420Spi.writeFIFO((uint8_t*)&packet_tx->dsn, 1);
      call CC2420Spi.writeFIFO((uint8_t*)&packet_tx->destpan, 2);
      call CC2420Spi.writeFIFO((uint8_t*)&packet_tx->dest, 2);
      call CC2420Spi.writeFIFO((uint8_t*)&packet_tx->src, 2);
      call CC2420Spi.writeFIFO((uint8_t*)&packet_tx->payload,
        packet_tx->length - RF_PACKET_OVERHEAD_SIZE);

      // The TX FIFO can only hold one packet. Make sure to not overrun
      // FIFO by waiting for transmission to start here and synchronizing
      // with the CC2420_TX_ACTIVE check in cc2420_send.
      // Note, that we may have to wait up to 192 us (12 symbols) before
      // transmission starts.

    call CC2420Spi.strobe(CC2420_STXCAL);
    delay_us(128);

    if (cca) {
      call CC2420Spi.strobe(CC2420_SRXON);
      while (call CC2420Spi.statusRead() & CC2420_STATUS_RSSI_VALID);

      call CC2420Spi.strobe(CC2420_STXONCCA);
    }
    else {
      call CC2420Spi.strobe(CC2420_STXON);
    }

    delay_us(128);

      for (j = LOOP; j > 0; j--) {

        if (SFD_IS_1) { //SFD is 1

            if( !(call CC2420Spi.statusRead() & CC2420_STATUS_TX_ACTIVE) ) {
            // SFD went high yet we are not transmitting.
            // We started receiving a packet right now

            return FAIL; //ERROR
            }

            // We wait until transmission has ended so that we get an
            // accurate measurement of the transmission time.
            while(call CC2420Spi.statusRead() & CC2420_STATUS_TX_ACTIVE);

            rtc2 = AT91F_RTTReadValue(AT91C_BASE_RTTC);
            rtct = rtc2-rtc;

            if (call CC2420Spi.statusRead() & CC2420_STATUS_TX_UNDERFLOW) {

            call CC2420Spi.strobe(CC2420_SFLUSHTX);
            call CC2420Spi.strobe(CC2420_SFLUSHTX);

              return FAIL; // ERROR underflow

            }

            if (MOTE_TYPE == 1) {
              // We need to explicitly turn off the radio,
            // since STXON[CCA] -> TX_ACTIVE -> RX_ACTIVE
              //call CC2420Spi.off();
            }

        signal CC2420Transmit.sendDone(packet_tx, rtct, SUCCESS);

            return SUCCESS; //OK
        }
    }

    call CC2420Interrupts.enableInterrupt(TRUE);

    // If we are using WITH_SEND_CCA, we get here if the packet wasn't
    // transmitted because of other channel activity.
      return FAIL; //ERROR Transmission never started
  }

  event void CC2420Interrupts.interrupt() {}

} // End of implementation
```

```
#include "AT91SAM7S256.h"
#include "dbgu.h"

module CC2420ReceiveP @safe()
{
  provides interface CC2420Receive;

  uses interface HplCC2420Spi as CC2420Spi;
  uses interface HplCC2420Interrupts as CC2420Interrupts;
}

implementation {

  uint32_t dummy;
  cc2420_packet_rx packet_rx;
  uint8_t rx_length = 0;

  command void CC2420Receive.beginReceive() {}

  task void readFIFO() {

    uint8_t crc;

    // Nothing to read if FIFOP is not 1
    if (FIFOP_IS_1) {

      if (FIFOP_IS_1 & !(FIFO_IS_1)) {

        //Always read at least 1 byte before flushing RXFIFO (cc2420 Data sheet pg 62)
          call CC2420Spi.readFIFO((uint8_t*) &dummy, 1);
          call CC2420Spi.strobe(CC2420_SFLUSHRX);
          call CC2420Spi.strobe(CC2420_SFLUSHRX);

          signal CC2420Receive.receive (&packet_rx, rx_length, 1);
          call CC2420Interrupts.enableInterrupt(TRUE);

          return; // FIFO Overflow.
      }

      call CC2420Spi.readFIFO(&packet_rx.length, 1);

      // We only want the data payload length part
      packet_rx.length = packet_rx.length - RF_PACKET_OVERHEAD_SIZE;
      rx_length = packet_rx.length;

      if (packet_rx.length == 0) {

          call CC2420Spi.readFIFO((uint8_t*) &dummy, 1);
          call CC2420Spi.strobe(CC2420_SFLUSHRX);
          call CC2420Spi.strobe(CC2420_SFLUSHRX);

          signal CC2420Receive.receive (&packet_rx, rx_length, 2);
          call CC2420Interrupts.enableInterrupt(TRUE);

        return; // Packet too short
      }

      if (packet_rx.length > MAX_PAYLOAD_SIZE) {

          call CC2420Spi.readFIFO((uint8_t*) &dummy, 1);
          call CC2420Spi.strobe(CC2420_SFLUSHRX);
          call CC2420Spi.strobe(CC2420_SFLUSHRX);

          signal CC2420Receive.receive (&packet_rx, rx_length, 3);
          call CC2420Interrupts.enableInterrupt(TRUE);

        return; // Packet too long
      }

      call CC2420Spi.readFIFO((uint8_t*)&packet_rx.fcf, 2);
      call CC2420Spi.readFIFO((uint8_t*)&packet_rx.dsn, 1);

      // Shall we send an acknowledgment packet?
      if (ACK_ENABLED) {

      // (+6) Problems with little/big endian processors
      DBGU_PutChar((packet_rx.fcf >> (IEEE154_FCF_ACK_REQ + 6)));
```

```

        if ( (((packet_rx.fcf >> (IEEE154_FCF_ACK_REQ+6)) & 0x01) == 1) &
            (((packet_rx.fcf >> (IEEE154_FCF_FRAME_TYPE+6)) & 7) ==
            IEEE154_TYPE_DATA) ) {

          call CC2420Spi.strobe(CC2420_SACK);
          signal CC2420Receive.receive (&packet_rx, rx_length, 0);
            call CC2420Interrupts.enableInterrupt(TRUE);

            DBGU_PutChar('A');DBGU_PutChar('C');DBGU_PutChar('K');DBGU_PutChar('1')
            ;
        }
      }

      // Is this an acknowledgment packet?
      if (((packet_rx.fcf >> (IEEE154_FCF_ACK_REQ+11)) & 0x01) == 1)) {

        call CC2420Spi.readFIFO((uint8_t*)&crc, 1);

        signal CC2420Receive.receive (&packet_rx, rx_length, 9);
          call CC2420Interrupts.enableInterrupt(TRUE);

          DBGU_PutChar('A');DBGU_PutChar('C');DBGU_PutChar('K');DBGU_PutChar('2');
          return;
      }

      // Skip the dest PAN and dest Address (it's controlled by hardware)
      call CC2420Spi.readFIFO((uint8_t*) &dummy, 4);
      // Read source address
      call CC2420Spi.readFIFO((uint8_t*)&packet_rx.src, 2);
      call CC2420Spi.readFIFO((uint8_t*)&packet_rx.payload, packet_rx.length);
      // Read the footer to get de RSSI value
      call CC2420Spi.readFIFO((uint8_t*)&packet_rx.rssi, 1);
      call CC2420Spi.readFIFO((uint8_t*)&crc, 1);

      // Check if the CRC is valid (mask with the most significant bit of the last
      byte is hight, otherwise low).
      if (crc & RF_CRC_OK_BM) {

        signal CC2420Receive.receive (&packet_rx, rx_length, 0);
        call CC2420Interrupts.enableInterrupt(TRUE);

        return; // OK

      } else {

          call CC2420Spi.readFIFO((uint8_t*) &dummy, 1);
          call CC2420Spi.strobe(CC2420_SFLUSHRX);
          call CC2420Spi.strobe(CC2420_SFLUSHRX);

          signal CC2420Receive.receive (&packet_rx, rx_length, 4);
          call CC2420Interrupts.enableInterrupt(TRUE);

          return; // CRC invalid
      }
    }

    signal CC2420Receive.receive (&packet_rx, rx_length, 5);
    call CC2420Interrupts.enableInterrupt(TRUE);

    return; // Nothing to read. FIFOP is 0.
  }

  event void CC2420Interrupts.interrupt() {

    DBGU_PutChar('!');DBGU_PutChar('!');

    call CC2420Interrupts.enableInterrupt(FALSE);
      post readFIFO();

  }
} // End implementation
```

```
#include "CC2420.h"
#include "dbgu.h"
#include "RF.h"

module HplCC2420SpiP
{
  provides interface HplCC2420Spi as CC2420Spi;
  provides interface SplitControl;

  uses interface HplAt91Spi as HplSpi;
  uses interface Leds;
}

implementation
{

  uint32_t dummy;
  uint16_t pan_id;
  cc2420_status_t status;


  command error_t SplitControl.start() {

  return SUCCESS;
  }

  command error_t SplitControl.stop() {

  return SUCCESS;
  }

  command uint8_t CC2420Spi.enableVREG(bool enable) {

  if (enable) {
    AT91F_PIO_CfgOutput(AT91C_BASE_PIOA, AT91C_PIO_PA1);
    AT91F_PIO_SetOutput(AT91C_BASE_PIOA, AT91C_PIO_PA1);

    // Voltage regulator in normal mode.
    AT91F_VREG_Disable_LowPowerMode(AT91C_BASE_VREG);
  }

  else {
    // WE MUST RESET CC2420 BEFORE DISABLING VOLTAGE REGULATOR
    call CC2420Spi.setReg(CC2420_MAIN, 0x0000); // Active reset CC2420
    delay_us(128);
    call CC2420Spi.setReg(CC2420_MAIN, 0xf800); // Deactive reset CC2420
    delay_us(128);

    AT91F_PIO_Disable(AT91C_BASE_PIOA, AT91C_PIO_PA1);
    AT91F_PIO_ClearOutput(AT91C_BASE_PIOA, AT91C_PIO_PA1);

    // Voltage regulator in standby mode (low-power mode).
    AT91F_VREG_Enable_LowPowerMode(AT91C_BASE_VREG);
  }

  command error_t CC2420Spi.init()
  {

    call CC2420Spi.enableVREG(TRUE); // Activate VREG (Voltage regulator normal
    mode)
    delay_us(250);
    call CC2420Spi.setReg(CC2420_MAIN, 0x0000); // Active reset CC2420
    delay_us(128);
    call CC2420Spi.setReg(CC2420_MAIN, 0xf800); // Deactive reset CC2420
    delay_us(128);

    call CC2420Spi.strobe(CC2420_SXOSCON); //turn on CC2420 Osc
    delay_ms(128);

    status = call CC2420Spi.statusRead();

    if((status == 0xff) | (status == 0x00)) {
      return FAIL;
    }

    // Wait for the crystal oscillator to become stable
    while ((call CC2420Spi.statusRead() & CC2420_STATUS_XOSC16M_STABLE) == 0);

  // Std Preamble, CRC, no auto ack, no hw addr decoding
  call CC2420Spi.setReg(CC2420_MDMCTRL0, 0x02E2);

  // Change default values as recomended in the data sheet,
    // correlation threshold = 20, RX bandpass filter = 1.3uA.
    call CC2420Spi.setReg(CC2420_MDMCTRL1, 0x0500);

    // Setting FIFOP to 127 bytes active low (between 0x0000 and 0x007F)
  call CC2420Spi.setReg(CC2420_IOCFG0, 0x007F);

  // Turn off Security
  call CC2420Spi.setReg(CC2420_SECCTRL0, 0x01C4);

  // All default except: reference bias current to RX, bandpass filter is set to 3uA
  call CC2420Spi.setReg(CC2420_RXCTRL1, 0x1A56);
  //call CC2420Spi.setReg(CC2420_RXCTRL1, 0x2A56);

  //Set the wait time after STXON:is 8 symbol periods(128us)
  //call CC2420Spi.setReg(CC2420_TXCTRL,0x80FF);
  //0xEC80 = -20, 0xEA80 = -22,
  //call CC2420Spi.setReg(CC2420_RSSI,0xE080);

  pan_id = PAN_ADDR;
  call CC2420Spi.writeRAM((uint8_t*)&pan_id, 2, CC2420_RAM_PANID);
  delay_us(128);

  // Possible channels: 11 to 26
  call CC2420Spi.set_channel(26);
  delay_us(128);

  // PA_LEVELS: 31, 27, 23, 19, 15, 11, 7, 3
  call CC2420Spi.set_txpower(31);

  if (TEST_MODE) {

    call CC2420Spi.setTestMode(TRUE, TEST_MODE);
    delay_us(128);
  }
  else {
    call CC2420Spi.setTestMode(FALSE, TEST_MODE);
  }

  if (MOTE_TYPE == 0) {
    call CC2420Spi.strobe(CC2420_STXON);
  }
  else {
    call CC2420Spi.strobe(CC2420_SRXON);
    while ((call CC2420Spi.statusRead() & CC2420_STATUS_RSSI_VALID) == 0);
  }

  delay_us(128);

  if(call CC2420Spi.statusRead() & CC2420_STATUS_LOCK) {
    DBGU_PutChar('O');DBGU_PutChar('s');DBGU_PutChar('c');DBGU_PutChar('-');
    return SUCCESS;
  }

  return SUCCESS;
}

command void CC2420Spi.setTestMode(bool enable, uint8_t mode)
{
  // Turn off the crystal oscillator and RF
  call CC2420Spi.strobe(CC2420_SRFOFF);
  delay_us(128);

  // if we want to set the test mode, we specify the number in mode
  if (enable) {
    call CC2420Spi.setReg(CC2420_MDMCTRL1, 0x0500 | ((mode << 2) & 0xf));
    call CC2420Spi.setReg(CC2420_DACTST, 0x1800);
  }
  else {
    call CC2420Spi.setReg(CC2420_MDMCTRL1, 0x0500);
    call CC2420Spi.setReg(CC2420_DACTST, 0x0000);
  }

  // we must flush the RXFIFO and TXFIFO
```

```
    call CC2420Spi.strobe(CC2420_SFLUSHTX);
    call CC2420Spi.strobe(CC2420_SFLUSHTX);
    call CC2420Spi.strobe(CC2420_SFLUSHRX);
    call CC2420Spi.strobe(CC2420_SFLUSHRX);
  }

  command void CC2420Spi.set_txpower(uint8_t power)
  {
    uint16_t reg;

    // All PA settings can be used, the ones given in the datasheet
    // correspond to 0, -1, -3, -5, -7, -10, -15, -25 dBm in the
    // reference design.

    if (power > PA_LEVEL) {
      power = PA_LEVEL;
    }

    reg = call CC2420Spi.getReg(CC2420_TXCTRL);
    reg = (reg & 0xA0E0) | (power & 0x1f);
    call CC2420Spi.setReg(CC2420_TXCTRL, reg);
  }

  command uint8_t CC2420Spi.statusRead()
  {
    return call CC2420Spi.strobe(CC2420_SNOP);
  }

  command uint8_t CC2420Spi.stateMachineRead()
  {
    return call CC2420Spi.getReg(CC2420_FMSTATE);
  }

  command void CC2420Spi.set_channel(uint8_t c)
  {
    uint16_t f;

    // 16 possible channels within the 2.4 GHz, numbered 11 to 26
    // (11 = 2405 MHz to 26 = 2480 MHz)

    // Derive frequency programming from the given channel number
    f = (uint16_t) (c - 11); // Subtract the base channel
    f = f + (f << 2);            // Multiply with 5, which is the channel
    spacing
    f = f + 357 + 0x4000;        // 357 is 2405-2048, 0x4000 is LOCK_THR = 1

    while ((call CC2420Spi.statusRead() & CC2420_STATUS_XOSC16M_STABLE) == 0);

    // Wait for any transmission to end
    while (call CC2420Spi.statusRead() & CC2420_STATUS_TX_ACTIVE);

    call CC2420Spi.setReg(CC2420_FSCTRL, f);

  }

  command void CC2420Spi.on(void)
  {

    // PUT VREG low mode !

    //rfsettings.receive_on = 1;
    //call HplIrq0.enable();
    call CC2420Spi.strobe(CC2420_SRXON);

    call CC2420Spi.readFIFO((uint8_t*) &dummy, 1);
    call CC2420Spi.strobe(CC2420_SFLUSHRX);
    call CC2420Spi.strobe(CC2420_SFLUSHRX);
  }

  command void CC2420Spi.off(void)
  {

    // PUT VREG normal mode !

    //rfsettings.receive_on = 0;

    // Wait for transmission to end before turning radio off.
    while (call CC2420Spi.statusRead() & CC2420_STATUS_TX_ACTIVE);

    call CC2420Spi.strobe(CC2420_SRFOFF);
    //call HplIrq0.disable();
  }

  command uint8_t CC2420Spi.strobe(uint8_t Tx)
  {
    uint32_t data = 0;
    data |= Tx;

    //MASK UNUSED BITS
    data &= 0x000000FF;
    //WAIT UNTIL TRANSMIT REGISTER IS EMPTY
    while (!(AT91C_BASE_SPI ->SPI_SR & AT91C_SPI_TXEMPTY));
    // TRANSMIT DATA
    AT91C_BASE_SPI ->SPI_TDR = data;
    //READ THE RECEIVED DATA
    //WAIT UNTIL RECEIVE REGISTER IS FULL
    while (!(AT91C_BASE_SPI -> SPI_SR & AT91C_SPI_RDRF));
    //READ RDR AND MASK UNUSED BITS
    data = (AT91C_BASE_SPI -> SPI_RDR & 0x000000FF);

    return data;
  }

  command void CC2420Spi.setReg(uint8_t add, uint16_t value)
  {
    uint8_t Tx;

    disableIRQ();

    call CC2420Spi.strobe(add);

    //MASK UNUSED BITS
    value &= 0x0000FFFF;
    //WAIT UNTIL TRANSMIT REGISTER IS EMPTY
    while (!(AT91C_BASE_SPI ->SPI_SR & AT91C_SPI_TXEMPTY));
    // TRANSMIT DATA
    AT91C_BASE_SPI ->SPI_TDR = (value >> 8);
    //READ THE RECEIVED DATA
    //WAIT UNTIL RECEIVE REGISTER IS FULL
    while (!(AT91C_BASE_SPI -> SPI_SR & AT91C_SPI_RDRF));
    //READ RDR AND MASK UNUSED BITS
    Tx = (AT91C_BASE_SPI -> SPI_RDR & 0x0000FFFF);

    //WAIT UNTIL TRANSMIT REGISTER IS EMPTY
    while (!(AT91C_BASE_SPI ->SPI_SR & AT91C_SPI_TXEMPTY));
    // TRANSMIT DATA
    AT91C_BASE_SPI ->SPI_TDR = value;
    //READ THE RECEIVED DATA
    //WAIT UNTIL RECEIVE REGISTER IS FULL
    while (!(AT91C_BASE_SPI -> SPI_SR & AT91C_SPI_RDRF));
    //READ RDR AND MASK UNUSED BITS
    Tx = (AT91C_BASE_SPI -> SPI_RDR & 0x0000FFFF);

    END_SPI_TRANSFER();
    enableIRQ();
  }

  command uint16_t CC2420Spi.getReg(uint8_t add)
  {
    uint16_t Tx;

    call CC2420Spi.strobe(add | 0x40);

    //WAIT UNTIL TRANSMIT REGISTER IS EMPTY
    while (!(AT91C_BASE_SPI ->SPI_SR & AT91C_SPI_TXEMPTY));
    // TRANSMIT DATA
    AT91C_BASE_SPI ->SPI_TDR = 0x00;
    //READ THE RECEIVED DATA
    //WAIT UNTIL RECEIVE REGISTER IS FULL
    while (!(AT91C_BASE_SPI -> SPI_SR & AT91C_SPI_RDRF));
    //READ RDR AND MASK UNUSED BITS
    Tx = (AT91C_BASE_SPI -> SPI_RDR & 0x0000FFFF);
    Tx = (Tx << 8);

    //WAIT UNTIL TRANSMIT REGISTER IS EMPTY
    while (!(AT91C_BASE_SPI ->SPI_SR & AT91C_SPI_TXEMPTY));
    // TRANSMIT DATA
    AT91C_BASE_SPI ->SPI_TDR = 0x00;
    //READ THE RECEIVED DATA
    //WAIT UNTIL RECEIVE REGISTER IS FULL
    while (!(AT91C_BASE_SPI -> SPI_SR & AT91C_SPI_RDRF));
    //READ RDR AND MASK UNUSED BITS
    Tx += (AT91C_BASE_SPI -> SPI_RDR & 0x0000FFFF);

    END_SPI_TRANSFER();
    enableIRQ();

    return Tx;
  }

  command void CC2420Spi.writeFIFO(uint8_t *value, uint8_t length)
  {
    uint8_t i, Tx;
    disableIRQ();

    call CC2420Spi.strobe(CC2420_TXFIFO);

    for (i = 0; i < length; i++) {

      //ENABLE 8 BIT TRANSFER
      AT91C_BASE_SPI->SPI_CSR[0] |= AT91C_SPI_BITS_8;
      //WAIT UNTIL TRANSMIT REGISTER IS EMPTY
      while (!(AT91C_BASE_SPI ->SPI_SR & AT91C_SPI_TXEMPTY));
      AT91C_BASE_SPI ->SPI_TDR = value[i];
      //READ THE RECEIVED DATA
      //WAIT UNTIL RECEIVE REGISTER IS FULL
      while (!(AT91C_BASE_SPI -> SPI_SR & AT91C_SPI_RDRF));
      //READ RDR AND MASK UNUSED BITS
      Tx = (AT91C_BASE_SPI -> SPI_RDR & 0x000000FF);
    }

    delay_us(15);
    END_SPI_TRANSFER();
    enableIRQ();
  }

  command void CC2420Spi.readFIFO(uint8_t *value, uint8_t length)
  {
    uint8_t i;

    disableIRQ();

    call CC2420Spi.strobe(CC2420_RXFIFO | 0x40);

    for (i = 0; i < length; i++) {

      //ENABLE 8 BIT TRANSFER
      AT91C_BASE_SPI->SPI_CSR[0] |= AT91C_SPI_BITS_8;
      //WAIT UNTIL TRANSMIT REGISTER IS EMPTY
      while (!(AT91C_BASE_SPI ->SPI_SR & AT91C_SPI_TXEMPTY));
      // TRANSMIT DATA
      AT91C_BASE_SPI ->SPI_TDR = 0x00;
      //READ THE RECEIVED DATA
      //WAIT UNTIL RECEIVE REGISTER IS FULL
      while (!(AT91C_BASE_SPI -> SPI_SR & AT91C_SPI_RDRF));
      //READ RDR AND MASK UNUSED BITS
      value[i] = (AT91C_BASE_SPI -> SPI_RDR & 0x000000FF);
    }

    delay_us(15);
    END_SPI_TRANSFER();
    enableIRQ();
  }

  command void CC2420Spi.readRAM(uint8_t *value, uint8_t length, uint16_t add)
  {
    uint8_t i, Tx;
    disableIRQ();

    //WAIT UNTIL TRANSMIT REGISTER IS EMPTY
    while (!(AT91C_BASE_SPI ->SPI_SR & AT91C_SPI_TXEMPTY));
    // TRANSMIT DATA
    AT91C_BASE_SPI ->SPI_TDR = (0x80 | (add & 0x7F));
    //READ THE RECEIVED DATA
    //WAIT UNTIL RECEIVE REGISTER IS FULL
    while (!(AT91C_BASE_SPI -> SPI_SR & AT91C_SPI_RDRF));
    //READ RDR AND MASK UNUSED BITS
    Tx = (AT91C_BASE_SPI -> SPI_RDR & 0x000000FF);

    //WAIT UNTIL TRANSMIT REGISTER IS EMPTY
    while (!(AT91C_BASE_SPI ->SPI_SR & AT91C_SPI_TXEMPTY));
    // TRANSMIT DATA
    AT91C_BASE_SPI ->SPI_TDR = (((add >> 1) & 0xC0) | 0x20);
    //READ THE RECEIVED DATA
    //WAIT UNTIL RECEIVE REGISTER IS FULL
    while (!(AT91C_BASE_SPI -> SPI_SR & AT91C_SPI_RDRF));
    //READ RDR AND MASK UNUSED BITS
    Tx = (AT91C_BASE_SPI -> SPI_RDR & 0x000000FF);

    for (i = 0; i < length; i++) {

      //WAIT UNTIL TRANSMIT REGISTER IS EMPTY
      while (!(AT91C_BASE_SPI ->SPI_SR & AT91C_SPI_TXEMPTY));
      // TRANSMIT DATA
      AT91C_BASE_SPI ->SPI_TDR = 0x00;
      //READ THE RECEIVED DATA
      //WAIT UNTIL RECEIVE REGISTER IS FULL
      while (!(AT91C_BASE_SPI -> SPI_SR & AT91C_SPI_RDRF));
      //READ RDR AND MASK UNUSED BITS
      value[i] = (AT91C_BASE_SPI -> SPI_RDR & 0x000000FF);
    }

    delay_us(15);
    END_SPI_TRANSFER();
    enableIRQ();
  }

  command void CC2420Spi.writeRAM(uint8_t *value, uint8_t length, uint16_t add)
  {
    uint8_t i, Tx;
    disableIRQ();

    //WAIT UNTIL TRANSMIT REGISTER IS EMPTY
    while (!(AT91C_BASE_SPI ->SPI_SR & AT91C_SPI_TXEMPTY));
    // TRANSMIT DATA
    AT91C_BASE_SPI ->SPI_TDR = (0x80 | (add & 0x7F));
    //READ THE RECEIVED DATA
    //WAIT UNTIL RECEIVE REGISTER IS FULL
    while (!(AT91C_BASE_SPI -> SPI_SR & AT91C_SPI_RDRF));
    //READ RDR AND MASK UNUSED BITS
    Tx = (AT91C_BASE_SPI -> SPI_RDR & 0x000000FF);

    //WAIT UNTIL TRANSMIT REGISTER IS EMPTY
    while (!(AT91C_BASE_SPI ->SPI_SR & AT91C_SPI_TXEMPTY));
    // TRANSMIT DATA
    AT91C_BASE_SPI ->SPI_TDR = ((add >> 1) & 0xC0);
    //READ THE RECEIVED DATA
    //WAIT UNTIL RECEIVE REGISTER IS FULL
    while (!(AT91C_BASE_SPI -> SPI_SR & AT91C_SPI_RDRF));
    //READ RDR AND MASK UNUSED BITS
    Tx = (AT91C_BASE_SPI -> SPI_RDR & 0x000000FF);

    for (i = 0; i < length; i++) {

      //WAIT UNTIL TRANSMIT REGISTER IS EMPTY
      while (!(AT91C_BASE_SPI ->SPI_SR & AT91C_SPI_TXEMPTY));
      // TRANSMIT DATA
      AT91C_BASE_SPI ->SPI_TDR = value[i];
      //READ THE RECEIVED DATA
      //WAIT UNTIL RECEIVE REGISTER IS FULL
      while (!(AT91C_BASE_SPI -> SPI_SR & AT91C_SPI_RDRF));
      //READ RDR AND MASK UNUSED BITS
      Tx = (AT91C_BASE_SPI -> SPI_RDR & 0x000000FF);
    }

    delay_us(15);
    END_SPI_TRANSFER();
    enableIRQ();
  }
} // End implementation
```

9.8. Appendix H

Setting up the TinyOS environment to work with Locomorph (v0.1)

G. Arimany, {*gucas10@student.sdu.dk*}

1. Introduction

The role of any operating system (OS) is to promote development of reliable application software by providing a convenient and safe abstraction of hardware resources. In a conventional PCs and servers, the OS allocates application-processing threads to processors, maps virtual addresses to locations in memory and manipulates disks, networks, and peripherals on the application's behalf. This OS/application division in conventional computing is less common in the embedded world, where applications are tightly bound to particular hardware and weave in high-level logic with manipulation of the physical equipment. This is due to very limited hardware resources, highly specialized applications and long, rigorous, development cycles often dominated by mechanical and other factors.

TinyOS was designed specifically for Wireless Sensors Networks (WSNs). It introduces a structured event-driven execution model and a component-based software design that supports a high degree of concurrency in a small footprint, enhances robustness, and minimizes power consumption while facilitating implementation of sophisticated protocols and algorithms.

In order to embed TinyOS to our board provided with an **Atmel AT91SAM7S256** microcontroller and set-up the user friendly programming environment, it is good to start reading the *Getting started with the Locomorph Programming Kit* guide [http://modular.mmmi.sdu.dk/files/getting-startedv0.4.pdf] to mainly be familiarized to all the hardware, put them together and be ready to start loading the embedded TinyOS as well as executing a basic application for the first time. Note that the software set-up included in that guide will be already done on the programming environment provided with this document.

After reading this guide the user should be ready to use the especially designed programming environment to load a basic application programmed on TinyOS embedded system.

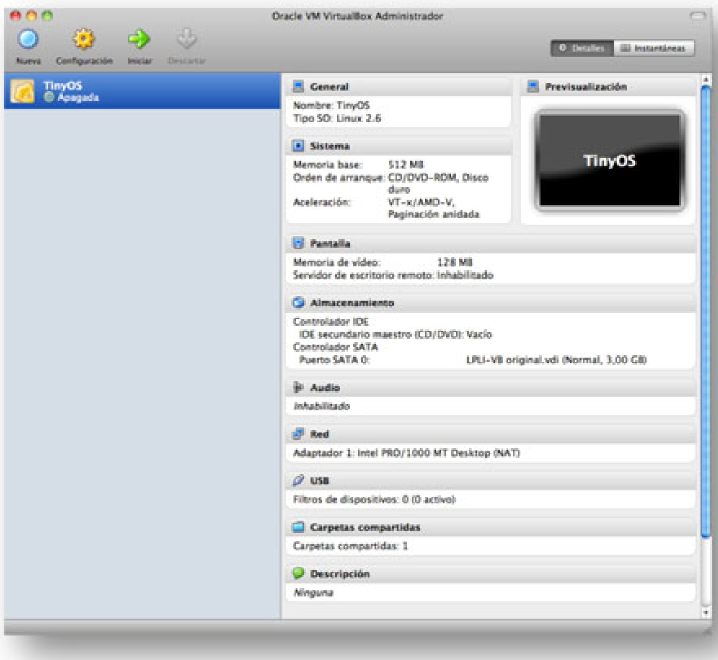
2. Configuring and setting up the Linux image

In order to install and use the Linux image it is necessary first to download and install the free of charge VirtualBox virtual machine and our Linux image. The first one can be downloaded from the official web page of VirtualBox [http://www.virtualbox.org] by choosing the right platform download link from the download menu. On the other hand, the Linux image is available at the Modular Robotics Lab's website.

2.1. Virtual machine set-up

Once installed, the user must set-up the Linux image to use it. To do it, an install of a new virtual image is required with the recommended settings shown in Figure 1.

Note: the user must be aware of selecting the previously downloaded Linux image as a hard drive when setting up a new virtual image.



General
Nombre: TinyOS
Tipo SO: Linux 2.6

Sistema
Memoria base: 512 MB
Orden de arranque: CD/DVD-ROM, Disco duro
Aceleración: VT-x/AMD-V, Paginación anidada

Pantalla
Memoria de vídeo: 128 MB
Servidor de escritorio remoto: Inhabilitado

Almacenamiento
Controlador IDE
IDE secundario maestro (CD/DVD): Vacío
Controlador SATA
Puerto SATA 0: LPLI-VB original.vdi (Normal, 3,00 GB)

Audio
Inhabilitado

Red
Adaptador 1: Intel PRO/1000 MT Desktop (NAT)

USB
Filtros de dispositivos: 0 (0 activo)

Carpetas compartidas
Carpetas compartidas: 1

Descripción
Ninguna

Figure 1: **Settings for VirtualBox with the TinyOS virtual machine already installed.**

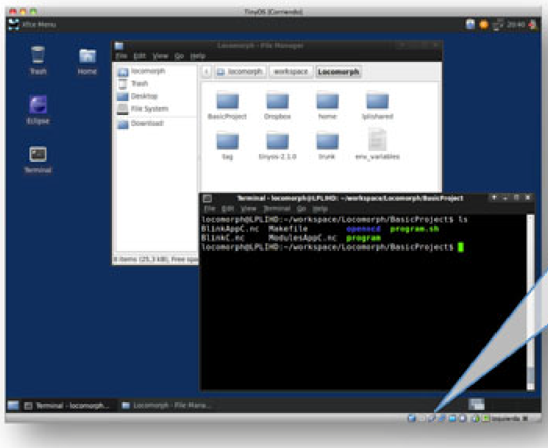
If all the settings are well configured the user must be able now to run the virtual machine and see the desktop of the Linux image after the booting process (Figure 2).

In the other hand, it is recommended to update the image source files from the Modular lab's SVN server version to get the latest versions of them (explained at the end of this document).

Notice that this is a XUbuntu Linux image running the kernel 2.6.32-21-generic with all the required software already installed. It means that in this image, the user can find pre-installed the TinyOS 2.1.0 operating system, the example codes, the compiler, the OpenOCD program loader and the necessary USB drivers, as well as an Eclipse editor with a build-in nesC plug-in. Some of this software will be explained with more detail in the next sections of this document and how to install the software not provided for extra features.

Figure 3 shows a JTAGKey device properly installed and connected to the virtual machine.

Note: if either the guest system or the virtual machine does not detect the JTAGKey device, the user must install the proper USB drivers. They can be found at the Amontec's web site: http://www.amontec.com/jtagkey-tiny.shtml.



```
locomorph@LPLIHD:~/workspace/Locomorph/BasicProject$ ls
BlinkAppC.nc  Makefile        openocd  program.sh
BlinkC.nc     ModulesAppC.nc  program
locomorph@LPLIHD:~/workspace/Locomorph/BasicProject$
```

Figure 2: **Desktop image of the virtual machine running our Linux image.**

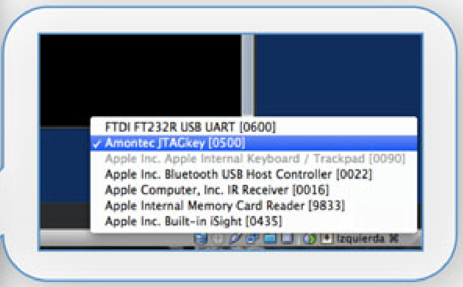

FTDI FT232R USB UART [0600]
✓ Amontec JTAGkey [0500]
Apple Inc. Apple Internal Keyboard / Trackpad [0090]
Apple Inc. Bluetooth USB Host Controller [0022]
Apple Computer, Inc. IR Receiver [0016]
Apple Internal Memory Card Reader [9833]
Apple Inc. Built-in iSight [0435]
Izquierda ⌘

Figure 3: **Once JTAG device is plugged-in, the user must select the proper USB driver to connect it to the virtual machine.**

3. Getting used to TinyOS files and structure

Under the Locomorph directory in our Linux image the user can find a set of different folders related to the development of TinyOS applications. The user also can find the root directory of TinyOS called "*tinyos-2.1.0/*" as well as a home folder to store the different applications of users and a *"tag/"* folder for the code ported to our platform. We structured the directories in this way to have more flexibility and versatility in order, among other advantages, to use a SVN server.

In this section, the user will be able to understand slightly more what is and how is TinyOS structured as well as the set of directories we have created to maintain a SVN server.

3.1. nesC as a C dialect

For someone who has experience with C or C++ languages, writing a simple nesC program is fairly straightforward: all the programmer needs to do is implement a module and wire it to the others. The code inside a module is similar to C coding but configurations are not. To understand the sense of the different files that the basic application uses, the user must know their meaning.

A nesC program is a collection of components and every component is in its own source file. There are two kinds of components: modules and configurations and they have similar *signature* sections but mainly differ in their implementation ones. Module implementation sections consist of nesC code that looks like C (declares variables and functions, calls functions and compiles to assembly code). Configurations are the major difference between nesC and C as they consist in a set of wirings that connect components together into larger abstractions.

That is, looking at the basic project, the user can find two components: "*BlinkC.nc*" and "*BlinkAppC.nc*". The first one has all the code needed to run the application and the second one is the top-level configuration of the *Blink* application and connects the *BlinkC* module to the other components needed (the programmer can explicitly select which components implementation of a determinate function to use.

Nevertheless, the aim of this document is to introduce the user to a basic application and is far away from explaining how to write nesC code but the user can find a good introduction and good manuals over Internet[1].

3.2. The TinyOS directory tree

In TinyOS 2.x, each set of files such as hardware-related and chip-related files as well as component-related files is located in a specific directory below *tinyos-2.x* folder. In Table 1 are shown the main directories with a brief explanation of what the user can find inside.

tinyos-2.1.0/			
	apps/		A directory to store the applications (not used for our apps).
	doc/		The doc build by *make <platform> doc.*
	support/		The system files for make, c, java or phyton.
	tos/		
		chips/	Each specific chip has its own directory.
		lib/	For generic components.
		platforms/	Each platform has its own directory.
		sensorboards/	Each platform has its own directory.
		system/	System components (i.e. scheduler).
		types/	Header files

Table 1: **List of the main directories of a TinyOS installation.**

As an example, a file dealing with the ADC of an Atmega128 is located in "*tinyos-2.1.0/tos/chips/atm128/adc/*" and the files related to an Atmega128 microcontroller for a specific platform supported by TinyOS is available at "*tinyos-2.1.0/tos/platforms/<platform>*".

3.3. Our folder structure

Our system has, apart of the main TinyOS directory, another set of directories maintained separately. Those can be seen in Table 2.

In the one named "*tag/*", the user can find all the directories and files related to our specific platform. In other words, all the files needed to port the board with an AT91SAM7S256 microcontroller like the code for a specific chip or all related to the adaptation process of our platform to TinyOS.

Then the user can find a directory called "*home/*" where all the applications are stored separately in user-named subfolders and another one called "*BasicProject/*" whose content will be explained later. The user should use his named folder to store the programs instead of the folder "*apps/*" under the "*tinyos-2.1.0/*" directory.

Finally, there is another directory called "*trunk/*" used only for the SVN version server to store the newest versions of the system the ones are not official (i.e. beta versions).

[1] Elementary documentation a how to program in TinyOS manual could be found at TinyOS official web page http://www.tinyos.net.

tag/				
	thor-0.0.3/			The name of the directory indicates the actual version of the Operating System.
		support/		The system files for make, c and OpenOCD files.
		tos/		
			chips/	Specific directory for the chips of our platform (i.e. AT91SAM7S256 microcontroller).
			platforms/	Specific directory for our platform.

Table 2: **List of the secondary set of directories to store all our system modifications and applications.**

4. Compiling, loading and debugging a basic application

To start loading a basic application to our mote, the user only needs to execute an already programmed shell script that automatizes all the process of compilation and programming of the mote, as it is explained below together with a way of software debugging using a console. The "*openocd/*" folder should be present at every folder where an application is stored so as to perform the compilation.

4.1. Compiling and programming the microcontroller

Whilst we are still working on the port for TinyOS, we provide a basic program written in nesC. This program uses the *Leds* and *TimerMilli* interfaces of TinyOS to create a timer that fires an event every certain time that toggles the led 0 of the mote and writes a specific character to the debugging console.

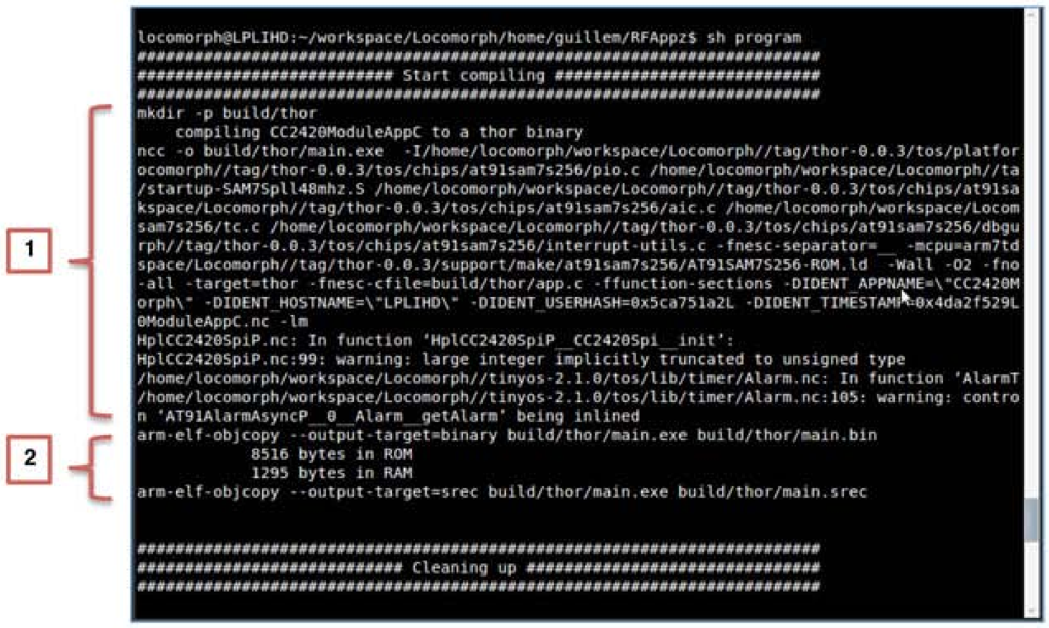
To compile this program, the user should move to the folder called "*BasicProject/*" and execute the shell script called *program.sh*. Then the shell script will execute the commands to compile the program and the command line will show all the steps. After compilation, the shell script will start programming the microcontroller automatically by loading the binaries using OpenOCD. If all the process goes well, the mote's led 0 should be blinking.

The whole process is explained below and could be seen in Figures 4 and 6.

1. Firstly, the shell script will start compiling the source files. That is, the nesC files (with extension *.nc) will be converted in one C file named "*app.c*" that afterwards will be compiled to an exec file called "*main.exe*" if no problems during the compilation where found (at this point the user should see the possible compilation errors).

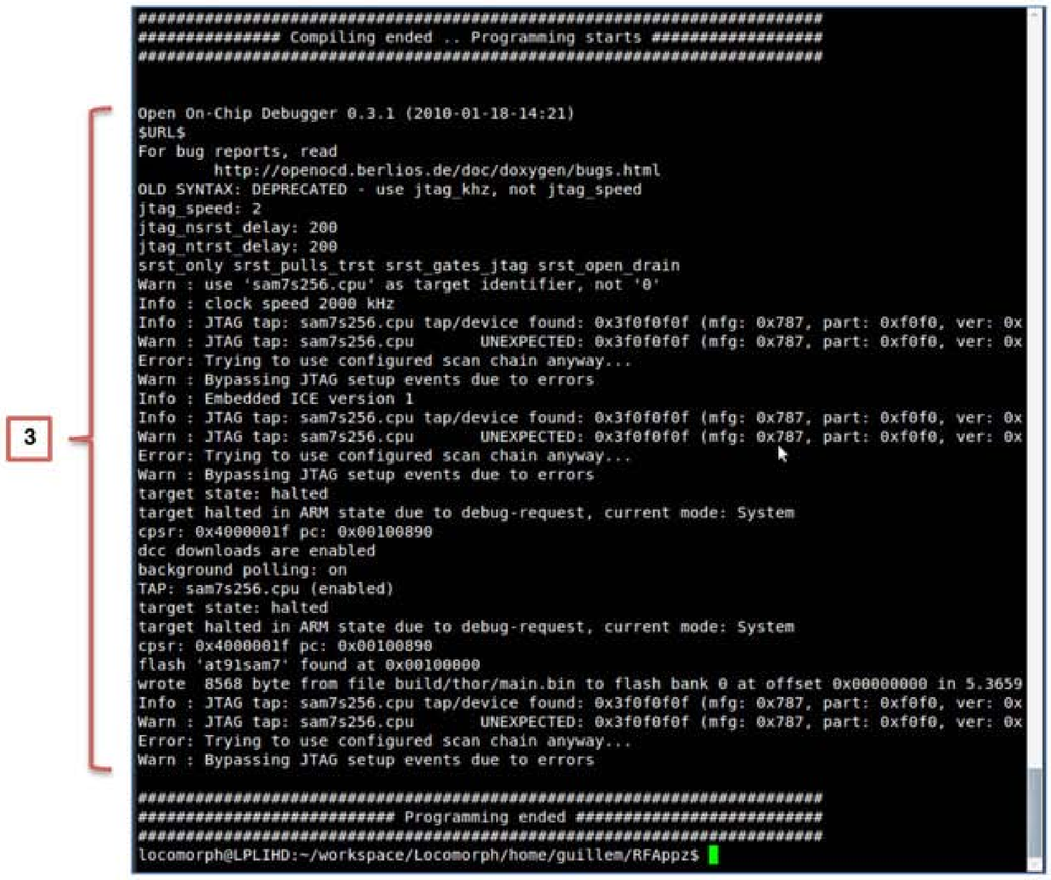
2. To finish the compilation, the binary file is translated in many formats by using *arm-elf-objcopy* with platform-specific options that will be ready to program the mote. The amount of RAM and ROM memory necessary to use into the mote is shown after building the binary.

3. Finally the mote is programmed by using the programming board options specified in the files "*amontec_jtag_key_cfg.script*", "*write_flash.script*" and "*unprotect_flash.script*" located on the "*openocd/*" folder at the same directory as the application we are compiling (it is normal to get several errors with OpenOCD related to its implementation that we can ignore).



```
locomorph@LPLIHD:~/workspace/Locomorph/home/guillem/RFAppz$ sh program
########################################################################
########################### Start compiling ############################
########################################################################
mkdir -p build/thor
    compiling CC2420ModuleAppC to a thor binary
ncc -o build/thor/main.exe  -I/home/locomorph/workspace/Locomorph//tag/thor-0.0.3/tos/platfor
ocomorph//tag/thor-0.0.3/tos/chips/at91sam7s256/pio.c /home/locomorph/workspace/Locomorph//ta
/startup-SAM7Spll48mhz.S /home/locomorph/workspace/Locomorph//tag/thor-0.0.3/tos/chips/at91sa
kspace/Locomorph//tag/thor-0.0.3/tos/chips/at91sam7s256/aic.c /home/locomorph/workspace/Locom
sam7s256/tc.c /home/locomorph/workspace/Locomorph//tag/thor-0.0.3/tos/chips/at91sam7s256/dbgu
rph//tag/thor-0.0.3/tos/chips/at91sam7s256/interrupt-utils.c -fnesc-separator=__ -mcpu=arm7td
space/Locomorph//tag/thor-0.0.3/support/make/at91sam7s256/AT91SAM7S256-ROM.ld  -Wall -O2 -fno
-all -target=thor -fnesc-cfile=build/thor/app.c -ffunction-sections -DIDENT_APPNAME=\"CC2420M
orph\" -DIDENT_HOSTNAME=\"LPLIHD\" -DIDENT_USERHASH=0x5ca751a2L -DIDENT_TIMESTAMP=0x4da2f529L
0ModuleAppC.nc -lm
HplCC2420SpiP.nc: In function 'HplCC2420SpiP__CC2420Spi__init':
HplCC2420SpiP.nc:99: warning: large integer implicitly truncated to unsigned type
/home/locomorph/workspace/Locomorph//tinyos-2.1.0/tos/lib/timer/Alarm.nc: In function 'AlarmT
/home/locomorph/workspace/Locomorph//tinyos-2.1.0/tos/lib/timer/Alarm.nc:105: warning: contro
n 'AT91AlarmAsyncP__0__Alarm__getAlarm' being inlined
arm-elf-objcopy --output-target=binary build/thor/main.exe build/thor/main.bin
            8516 bytes in ROM
            1295 bytes in RAM
arm-elf-objcopy --output-target=srec build/thor/main.exe build/thor/main.srec


########################################################################
############################ Cleaning up ###############################
########################################################################
```



```
########################################################################
############### Compiling ended .. Programming starts ##################
########################################################################


Open On-Chip Debugger 0.3.1 (2010-01-18-14:21)
$URL$
For bug reports, read
        http://openocd.berlios.de/doc/doxygen/bugs.html
OLD SYNTAX: DEPRECATED - use jtag_khz, not jtag_speed
jtag_speed: 2
jtag_nsrst_delay: 200
jtag_ntrst_delay: 200
srst_only srst_pulls_trst srst_gates_jtag srst_open_drain
Warn : use 'sam7s256.cpu' as target identifier, not '0'
Info : clock speed 2000 kHz
Info : JTAG tap: sam7s256.cpu tap/device found: 0x3f0f0f0f (mfg: 0x787, part: 0xf0f0, ver: 0x
Warn : JTAG tap: sam7s256.cpu       UNEXPECTED: 0x3f0f0f0f (mfg: 0x787, part: 0xf0f0, ver: 0x
Error: Trying to use configured scan chain anyway...
Warn : Bypassing JTAG setup events due to errors
Info : Embedded ICE version 1
Info : JTAG tap: sam7s256.cpu tap/device found: 0x3f0f0f0f (mfg: 0x787, part: 0xf0f0, ver: 0x
Warn : JTAG tap: sam7s256.cpu       UNEXPECTED: 0x3f0f0f0f (mfg: 0x787, part: 0xf0f0, ver: 0x
Error: Trying to use configured scan chain anyway...
Warn : Bypassing JTAG setup events due to errors
target state: halted
target halted in ARM state due to debug-request, current mode: System
cpsr: 0x4000001f pc: 0x00100890
dcc downloads are enabled
background polling: on
TAP: sam7s256.cpu (enabled)
target state: halted
target halted in ARM state due to debug-request, current mode: System
cpsr: 0x4000001f pc: 0x00100890
flash 'at91sam7' found at 0x00100000
wrote  8568 byte from file build/thor/main.bin to flash bank 0 at offset 0x00000000 in 5.3659
Info : JTAG tap: sam7s256.cpu tap/device found: 0x3f0f0f0f (mfg: 0x787, part: 0xf0f0, ver: 0x
Warn : JTAG tap: sam7s256.cpu       UNEXPECTED: 0x3f0f0f0f (mfg: 0x787, part: 0xf0f0, ver: 0x
Error: Trying to use configured scan chain anyway...
Warn : Bypassing JTAG setup events due to errors

########################################################################
########################### Programming ended ##########################
########################################################################
locomorph@LPLIHD:~/workspace/Locomorph/home/guillem/RFAppz$ 
```

Figures 4 & 6: Output dialog when executing the *program.sh* shell script.

4.2. Software debugging using the console

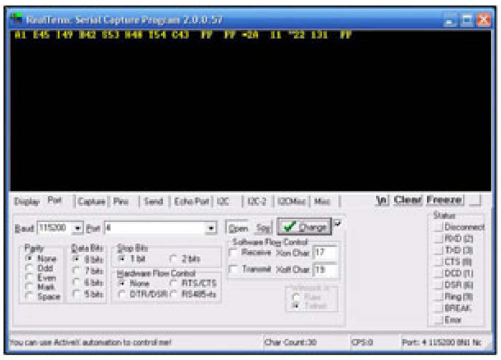
Because the purpose of the console is merely a debugging functionality, we use that console to simulate a *printf* function that at this moment is not ported to our microcontroller.

We provide a simply one byte serial debugger to help the programmer to develop his applications. It means that the debugger can hold only one byte in every call due to some hardware limitations.

The *Realterm* [http://realterm.sourceforge.net] terminal program that is specially designed for capturing, controlling and debugging binary and other data streams is used because it is far better for debugging communications than *Hyperterminal* on Windows. It is only available on Windows but the user can use Wine[2] to execute it on Linux or download another recommended programs like *CuteCom* or *CoolTerm* for MacOS, for example.

The configuration is the same for all the platforms, it is straightforward and the user has to change only a few parameters after the program is installed properly on the computer. Those are: **115200 bauds**, **8 bits of data**, **no parity**, **1 stop bit** and **no flow control**. Without further modifications, when running the microcontroller board with pre-loaded basic program, the console should display something similar like what it is shown in **¡ Error! No se encuentra el origen de la referencia.** and 6.

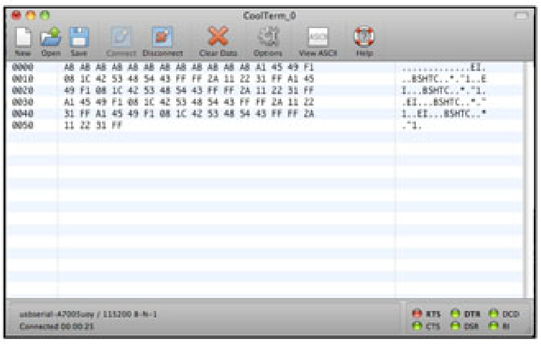
Note that the user should check that the Console is connected to the right serial device (port number). If the port is not shown on the serial port list, the user may have to install either the JTAG drivers or the FTDI VCP drivers that can be downloaded from [http://www.ftdichip.com/Drivers/VCP.htm].


RealTerm: Serial Capture Program 2.0.0.57

```
A1 E45 I49 B42 S53 H48 T54 C43  FF  FF *2A  11 "22 131  FF
```

Baud 115200 Port 4
Parity: None; Data Bits: 8 bits; Stop Bits: 1 bit; Hardware Flow Control: None
Software Flow Control: Xon Char 17, Xoff Char 19

You can use ActiveX automation to control me! Char Count:30 CPS:0 Port: 4 115200 8N1 Nc


CoolTerm_0

```
0000    A8 A8 A8 A8 A8 A8 A8 A8 A8 A8 A8 A8 A1 45 49 F1    .............EI.
0010    08 1C 42 53 48 54 43 FF FF 2A 11 22 31 FF A1 45    ..BSHTC..*."1..E
0020    49 F1 08 1C 42 53 48 54 43 FF FF 2A 11 22 31 FF    I...BSHTC..*."1.
0030    A1 45 49 F1 08 1C 42 53 48 54 43 FF FF 2A 11 22    .EI...BSHTC..*."
0040    31 FF A1 45 49 F1 08 1C 42 53 48 54 43 FF FF 2A    1..EI...BSHTC..*
0050    11 22 31 FF                                        ."1.
```

usbserial-A700fuzy / 115200 8-N-1
Connected 00:00:21

Figures 5 & 6: *RealTerm* and *CoolTerm* consoles receiving serial data (Hex + ASCII) from the JTAG interface connected to the microcontroller board.

The user can find the implementation code under the directory of the microcontroller chip "*tag/thor-0.0.3/tos/chips/at91sam7s256/*" and in files named *"dbgu.c"* and *"dbgu.h"*.

[2] Wine is a program that makes possible to run Windows applications on other OS. The user can find more information in http://www.winehq.org.

5. Additional information

5.1. Hardware documentation

The user will probably want to interface external electronics to the board using the connector pins included. In order to know more about the microcontroller board and the hardware we are interfacing, it is highly recommended to read the microcontroller datasheet [http://www.atmel.com/dyn/resources/prod_documents/doc6175.pdf] and the general board datasheet and its schematics diagram [http://modular.mmmi.sdu.dk/files/getting-startedv0.4.pdf].

5.2. Using an SVN version server to have an up-to-date system

A SVN version server is not installed but the configuration is ready to use. The user must install a SVN using either the command *apt-get install* or *Synaptic Package Manager* to use this extra functionality and have a working account to access to the Institute version server.

In order to update the files from the system, the user should type the command *"svn update"* in a terminal opened in *"/home/locomorph/workspace/Locomorph"* folder to update all the required files.

IT IS RECOMMENDED TO UPDATE ALL THE FILES BEFORE WORKING WITH THE SYSTEM.