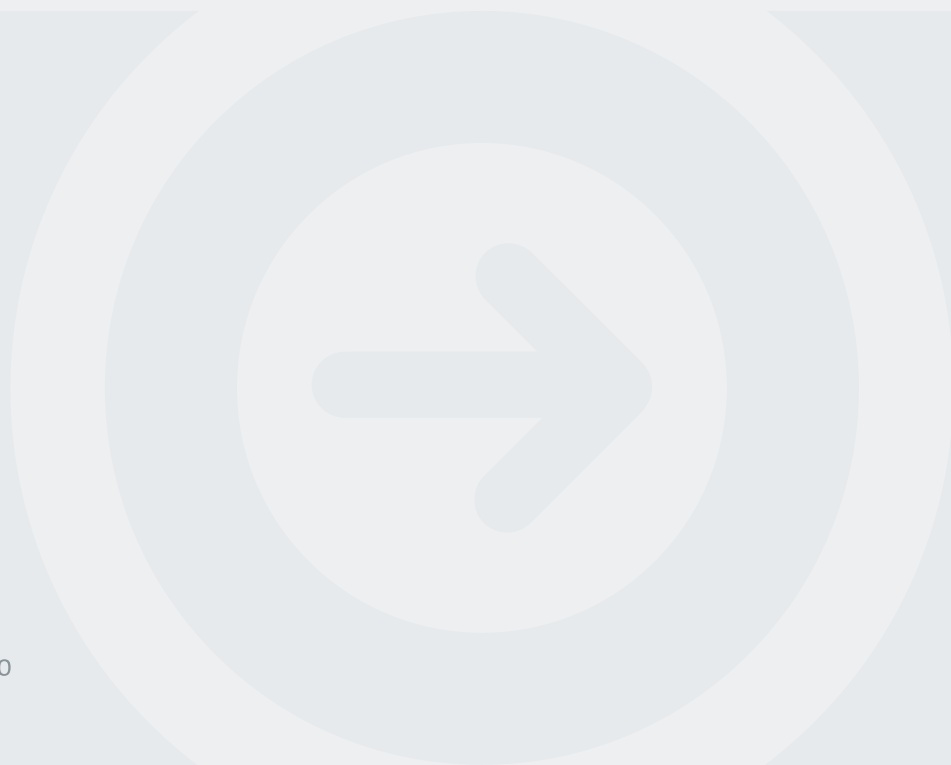


## EFFICIENT IMPLEMENTATION OF A DOMINANCE PROTOCOL FOR WIRELESS MEDIUM ACCESS

Ricardo Filipe Teixeira Gomes

Novembro de 2008



# EFFICIENT IMPLEMENTATION OF A DOMINANCE PROTOCOL FOR WIRELESS MEDIUM ACCESS

Ricardo Filipe Teixeira Gomes



Master degree in Electrical and Computer Engineering

Automation and Systems

Electrical Engineering Department

Polytechnic Institute of Porto – School of Engineering

2008



A thesis submitted in partial fulfilment of the specified requirements for the degree of  
Master in Electrical and Computer Engineering

Candidate: Ricardo Filipe Teixeira Gomes, [rftg.development@gmail.com](mailto:rftg.development@gmail.com)

Supervisor: Björn Andersson, [bandersson@dei.isep.ipp.pt](mailto:bandersson@dei.isep.ipp.pt)

Co-Supervisor: Nuno Pereira, [nap@isep.ipp.pt](mailto:nap@isep.ipp.pt)



Master degree in Electrical and Computer Engineering

Automation and Systems

Electrical Engineering Department

Polytechnic Institute of Porto – School of Engineering

9 de December de 2008



*“The wireless telegraph is not difficult to understand. The ordinary telegraph is like a very long cat. You pull the tail in New York, and it meows in Los Angeles. The wireless is the same, only without the cat.”* Albert Einstein, 1879-1955



To my beloved future wife, Vanessa: Since that rainy February afternoon, back in 2001...  
Then and always, thank you for not letting me give up.





## *Acknowledgements*

I would like to thank all my colleagues at the CISTER/IPP- Hurray! R&D group for all their precious support, especially Nuno Pereira, Björn Andersson and Eduardo Tovar. To me, without your precious teachings, Real-Time Systems would be a mirage and wireless dominance protocols would never exist. Having the pride of working on such a reputed R&D group was never in my thoughts, although you have made it real. The value of the knowledge you have transferred to me is priceless, thank you for trusting it on me.

Jacques Bastian, from RF Monolithics, for all the technical support related with the radio devices integration and the extraordinary commitment in speeding up the components delivery process.

To my friends, family and the ones who have already past, I must thank with all my heart. You have always said I could make it this far, and never doubted it! I can only promise that I will keep on giving the best of me.



## *Abstract*

Embedded computing systems went through extraordinary evolutions during the past two decades, representing nowadays one of the most promising technologies for improving a wide range of application areas such as energy/resource management, safety, health or entertainment. New sensors and actuators are leading to an unprecedented level of interaction between computing systems and their surrounding physical environment. These embedded computers tend to be networked, often wirelessly, and they are becoming denser, of larger scale and more pervasively deployed.

Since the wireless channel is a "natural resource" which must be shared between this large number of embedded computers, the medium access control (MAC) protocol significantly influences the performance of the entire system. In particular, satisfying real-time requirements — something that is needed for a computer to tightly interact with its physical environment — plays an important role. One solution was recently proposed by Pereira, Andersson and Tovar. It was a prioritized and collision-free MAC protocol belonging to a family of protocols called dominance/binary countdown protocols. This solution was implemented in commercial-off-the-shelf (COTS) wireless sensor networks (WSN) platforms and the implementation was demonstrated to be working. Unfortunately, those platforms had (for the MAC protocol) unfavourable characteristics which lead to limited efficiency and excessive overhead of the MAC protocol.

This work presents a new hardware platform, in the form of a network adapter for common WSN platforms, that allows an efficient implementation of dominance protocols for wireless medium access, allowing the medium access to be performed in less than 5 ms for  $2^{16}$  priority levels, which represents an overhead reduction of more than ten times as compared to the protocol implementation in COTS WSN platforms. Additionally, the overall energy consumption was reduced by approximately 45 % when compared to the theoretical best-case performance of the protocol implementation in COTS WSN platforms.

This work also allowed, for the first time ever, an aggregate computation scheme for WSN to work exploiting the new efficient implementation of a binary/dominance countdown protocol.

***Keywords***

Wireless Sensor Networks, Medium Access Control, Wireless Dominance Protocol, WiDom, Real-Time Systems, Dense Large-scale Networks.

## *Resumo*

No decurso das duas últimas décadas, os sistemas de computação embebidos sofreram uma extraordinária evolução, representando hoje em dia uma das mais promissoras tecnologias para possibilitar melhorias em áreas de aplicação tão diversas como a gestão de energia/recursos, segurança, saúde ou entretenimento. Novos sensores e actuadores conduzem a interacção entre os sistemas de computação e o ambiente físico onde se encontram inseridos até um nível sem precedentes. Estes computadores embebidos tendem a encontrar-se interligados (frequentemente recorrendo a ligações sem fios), sendo instalados de forma intensiva, e cada vez em maior escala.

Sendo o meio de transmissão sem fios um “recurso natural” que deve ser partilhado entre este número elevado de computadores embebidos, o protocolo de controlo de acesso ao meio (do Inglês *Medium Access Control* - MAC) influencia significativamente o desempenho de todo o sistema. Em particular, a satisfação de requisitos temporais — algo que é necessário para uma estreita interacção entre um computador e o meio físico onde se insere — desempenha um papel muito importante. Uma solução foi recentemente proposta por Pereira, Andersson e Tovar, consistindo num protocolo MAC que implementa escalonamento baseado em prioridades e livre de colisões. Esta solução foi implementada em plataformas comerciais de redes de sensores sem fios (do Inglês *Wireless Sensor Networks* - WSN), demonstrando-se funcional. Infelizmente, para a implementação do protocolo MAC, tais plataformas possuem características desfavoráveis, que conduzem a uma eficiência limitada do mesmo.

Este trabalho apresenta uma nova plataforma de *hardware*, sob a forma de um adaptador de rede, para plataformas de WSN disponíveis comercialmente, que permite que a realização do acesso ao meio decorra em menos de 5 ms (para  $2^{16}$  níveis de prioridades). Tal representa uma redução superior a dez vezes no custo da execução do protocolo, quando comparada com a anterior implementação em plataformas comerciais de WSN. Adicionalmente, o consumo global de energia foi reduzido em aproximadamente 45 %, quando comparado com o melhor desempenho teórico possível da implementação em plataformas comerciais de WSN.

Este trabalho permitiu ainda que, pela primeira vez, um esquema de computação agregada para WSN operasse tirando partido da implementação eficiente de um protocolo MAC apresentada neste trabalho.

***Palavras-chave***

Wireless Sensor Networks, Medium Access Control, Wireless Dominance Protocol, WiDom, Real-Time Systems, Dense Large-scale Networks.

# Index

<b>ACKNOWLEDGEMENTS</b> .....	<b>I</b>
<b>ABSTRACT</b> .....	<b>III</b>
<b>RESUMO</b> .....	<b>V</b>
<b>INDEX</b> .....	<b>VII</b>
<b>FIGURES INDEX</b> .....	<b>XI</b>
<b>TABLES INDEX</b> .....	<b>XV</b>
<b>ACRONYMS</b> .....	<b>XVII</b>
<b>1. INTRODUCTION</b> .....	<b>1</b>
<b>2. BACKGROUND</b> .....	<b>5</b>
2.1. REAL-TIME SYSTEMS TERMINOLOGY AND GENERAL DEFINITIONS .....	7
2.1.1. <i>Real-time systems classification</i> .....	8
2.1.2. <i>Events and its occurrence nature</i> .....	8
2.2. WIRELESS SENSOR NETWORKS .....	9
2.2.1. <i>Commonly used WSN platforms</i> .....	10
2.2.1.1. The MICAz .....	11
2.2.1.2. The CMU-FF .....	13
2.2.2. <i>The importance of the MAC protocol in WSN</i> .....	16
2.2.2.1. Basic techniques in WSN MAC protocol design .....	17
2.2.2.2. WSN MAC protocol topology concerns .....	19
2.2.3. <i>Real-time communications in WSN</i> .....	20
2.3. WiDOM .....	22
2.3.1. <i>Dominance/Binary Countdown Protocols</i> .....	23
2.3.2. <i>Dominance protocols for wireless medium access</i> .....	25
2.3.3. <i>WiDom in detail</i> .....	25
<b>3. A PLATFORM FOR WIRELESS DOMINANCE MAC PROTOCOL IMPLEMENTATION ...</b>	<b>33</b>
3.1. OVERVIEW OF THE ARCHITECTURE .....	37
3.1.1. <i>WiFLEX platform architecture</i> .....	38
3.2. HARDWARE .....	41
3.2.1. <i>General PCB design considerations</i> .....	43
3.2.2. <i>Radio devices</i> .....	47
3.2.2.1. Receiving devices digital output .....	48
3.2.3. <i>Microcontroller</i> .....	49
3.2.4. <i>HF switch</i> .....	50
3.2.5. <i>Miscellaneous hardware</i> .....	51



3.2.6.	<i>WiFLEX_main board design</i> .....	52
3.2.6.1.	Dimensioning transmission circuitry .....	52
3.2.6.2.	Dimensioning receiver circuitry .....	53
3.2.6.3.	MCU, interfaces and miscellaneous circuitry .....	55
3.2.6.4.	Components placement, details and outline .....	56
3.2.7.	<i>WiFLEX_rxsync board design</i> .....	58
3.2.7.1.	Dimensioning receiver circuitry .....	59
3.2.7.2.	Components placement, details and outline .....	60
3.2.8.	<i>WiFLEX_txsync board design</i> .....	61
3.2.8.1.	Dimensioning transmitter circuitry .....	61
3.2.8.2.	MCU, interfaces and miscellaneous circuitry .....	61
3.2.8.3.	Components placement, details and outline .....	62
3.3.	COMMUNICATION DETAILS .....	63
3.3.1.	<i>Bit-stuffing</i> .....	63
3.3.2.	<i>Preamble</i> .....	67
3.3.3.	<i>Capture effect</i> .....	67
3.4.	SOFTWARE .....	68
3.4.1.	<i>Low level functions common to WiFLEX_main and WiFLEX_txsync boards</i> .....	71
3.4.2.	<i>Hardware mapping</i> .....	72
3.4.3.	<i>Drivers</i> .....	73
3.4.3.1.	WiFLEX_main board drivers .....	73
3.4.3.2.	WiFLEX_txsync board drivers .....	75
3.4.3.3.	Board initialization .....	75
3.4.4.	<i>Application</i> .....	76
3.4.4.1.	Communication with the host WSN platform .....	76
3.4.4.2.	Protocol implementation .....	79
<b>4.</b>	<b>EXPERIMENTAL EVALUATION</b> .....	<b>83</b>
4.1.	RADIO CHARACTERISTICS .....	84
4.1.1.	<i>Self detection and carrier detection delay</i> .....	84
4.1.2.	<i>Determining pulse width and switching time</i> .....	87
4.1.3.	<i>Reliability versus distance</i> .....	88
4.1.4.	<i>Energy consumption</i> .....	89
4.2.	AGGREGATE DATA COMPUTATION .....	92
4.2.1.	<i>Background on WiSe-CAN</i> .....	93
4.2.2.	<i>Computing MIN</i> .....	94
4.2.3.	<i>Wireless interpolation</i> .....	96
<b>5.</b>	<b>CONCLUSIONS AND FUTURE WORK</b> .....	<b>99</b>
	<b>REFERENCES</b> .....	<b>101</b>
	<b>APPENDIX A. MICAZ'S SCHEMATICS AND DETAILED INFORMATION</b> .....	<b>107</b>
	<b>APPENDIX B. CMU-FIREFLY'S SCHEMATIC AND DETAILED INFORMATION</b> .....	<b>113</b>
	<b>APPENDIX C. RECEIVING RADIO DEVICES' DIGITAL OUTPUT</b> .....	<b>119</b>

<b>APPENDIX D. RADIO DEVICES' EXTERNAL CIRCUITRY DIMENSIONING .....</b>	<b>121</b>
<b>APPENDIX E. WIFLEX_MAIN AND WIFLEX_TXSYNC BOARDS HARDWARE MAPPING ....</b>	<b>125</b>
<b>APPENDIX F. DETERMINING PULSE WIDTH AND SWITCHING TIME EXPERIMENTAL RESULTS .....</b>	<b>127</b>
<b>APPENDIX G. RELIABILITY VERSUS DISTANCE EXPERIMENTAL RESULTS.....</b>	<b>137</b>



## Figures Index

Figure 1	Crossbow's MICAz WSN platform .....	12
Figure 2	CMU-FF v2.1 with mini-SD card slot (left) and top view of the CMU-FF v2.1 with full wave size RP-SMA 2.4 GHz antenna and alternative ten pin GPIO connector (right) .....	14
Figure 3	CMU-FF's Sensor Expansion Board (prototype version) [24] .....	15
Figure 4	FireFly-power-control expansion board [23] .....	15
Figure 5	CMUcam3 [17] .....	15
Figure 6	AM/FM signal receiving board [22] .....	16
Figure 7	Simple TDMA MAC .....	18
Figure 8	Hidden node problem .....	19
Figure 9	Exposed node problem .....	20
Figure 10	IEEE 802.15.4 superframe structure .....	22
Figure 11	Arbitration in dominance/binary countdown protocols .....	24
Figure 12	WiDom protocol state machine in timed-automata like notation [49] .....	28
Figure 13	Application (a), MAC protocol (b) and Radio (c) activity example in two nodes, $N_1$ and $N_2$ [49] .....	31
Figure 14	Radio architectures: a) transceiver based; b) separated transmitter and receiver with non-shared antenna; c) separated transmitter and receiver with shared antenna through an HF switch .....	35
Figure 15	Non-concurrent MAC protocol execution architecture .....	36
Figure 16	Concurrent MAC protocol execution co-processor based architecture .....	37
Figure 17	System overview .....	38
Figure 18	Overview of the WiFLEX_main board architecture, including WiFLEX_rxsync board. ....	40
Figure 19	WiFLEX_main board hardware architecture, including WiFLEX_rxsync board .....	42
Figure 20	WiFLEX_txsync board hardware architecture .....	43
Figure 21	Current spikes in a wrong decoupling capacitor placement (left) and correctly decoupling capacitor placement (right) [6] .....	45
Figure 22	Additional filtering using an inductor [6] .....	45
Figure 23	Transitions between different trace widths: a) smooth transition (best); b) multi-step transition (acceptable); c) single-step transition (avoid) .....	46
Figure 24	Corner shapes: a), b) and c) simple yet well performing corner shapes; d) best corner shape; e) and f) bad performance corner shape (avoid) .....	47
Figure 25	Antennas: (a) $\frac{1}{4}$ wave full-size antenna; (b) $\frac{1}{4}$ wave compact antenna [39] .....	51
Figure 26	OOK transmitter with minimum components configuration [56] .....	52
Figure 27	Robust TX5002 circuitry configuration .....	53

Figure 28	OOK receiver with minimum components configuration [54] .....	54
Figure 29	Robust RX5002 circuitry configuration .....	54
Figure 30	MCU connections, reset, external oscillator and LED circuitry.....	55
Figure 31	ISP, UART and interfacing connectors (for CMU-FF, MICAz and WiFLEX_rxsync)... .....	55
Figure 32	HF switch configuration.....	56
Figure 33	Main board sections.....	57
Figure 34	Components distribution over top (a) and bottom (b) layers of the WiFLEX_main board; (c) PCB dimensions.....	57
Figure 35	WiFLEX_main board fitted in the CMU-FF (left) and in the MICAz (right).....	58
Figure 36	Robust RX5001 circuitry configuration .....	59
Figure 37	WiFLEX_rxsync interface connectors .....	59
Figure 38	Components distribution over top (a) and bottom (b) layers of the WiFLEX_rxsync board; (c) PCB dimensions.....	60
Figure 39	WiFLEX_rxsync board fitting in the WiFLEX_main board.....	60
Figure 40	Robust TX5001 circuitry configuration .....	61
Figure 41	MCU connections, reset, external oscillator, GPIO and LED circuitry .....	62
Figure 42	Power, ISP and UART connectors .....	62
Figure 43	Components distribution over top (a) and bottom (b) layers of the WiFLEX_txsync board; (c) PCB dimensions.....	63
Figure 44	Receiver signal processing [52].....	64
Figure 45	Minimum and maximum pulse width in a non return to zero (NRZ) bit encoding [52] .. .....	64
Figure 46	Manchester bit encoding [52].....	65
Figure 47	Manchester encoding (top) and reverse Manchester encoding (bottom).....	66
Figure 48	Bit stuffing approach .....	66
Figure 49	AVR programmer selection menu.....	68
Figure 50	AVRISP mkII programmer configuration menu.....	69
Figure 51	Advance and Board settings tabs.....	70
Figure 52	AVR Studio v4 IDE .....	71
Figure 53	Communication between host WSN platform and WiFLEX_main board .....	76
Figure 54	Algorithm of the communication protocol with the host WSN platform for data receiving.....	77
Figure 55	Algorithm of the communication protocol with the host WSN platform for data transmission.....	78
Figure 56	Protocol implementation .....	79
Figure 57	Tournament phase .....	80
Figure 58	Dominant bit.....	81
Figure 59	RX5000 Series ASH Receiver Block Diagram [54] .....	85

Figure 60	Response time of the RXDATA pin to a transmitted carrier wave .....	86
Figure 61	Response time of the RXDATA pin after turning off the transmission of a carrier wave .....	86
Figure 62	Experimental set-up for minimum pulse width and silence interval determination.....	88
Figure 63	Failed tournaments with distance .....	89
Figure 64	Trace of current consumption, during a tournament phase, for the WiDom implementation using the MICAz's CC2420 radio transceiver .....	91
Figure 65	Trace of current consumption, during a tournament phase, for the WiDom implementation using the MICAz's with the WiFLEX_main and WiFLEX_rxsync boards...	91
Figure 66	Iterations of the interpolation scheme [4].....	94
Figure 67	Time to compute MIN as a function of the number of nodes ( $m$ ).....	96
Figure 68	Interpolation experiment .....	97
Figure 69	MICAz's 51 pins expansion connector schematic [20].....	107
Figure 70	MICAz's ATmega128 configuration schematic [20].....	108
Figure 71	MICAz's CC2420 radio transceiver configuration schematic [20].....	109
Figure 72	MICAz's battery connections, ON/OFF button and ADC1 schematics [20] .....	110
Figure 73	MICAz's flash memory, serial ID, LEDs and USART, configuration schematics [20] ..	110
Figure 74	MICAz's dimensions [20].....	111
Figure 75	MICAz's main components (top layer only).....	111
Figure 76	CMU-FF's ATmega128L configuration and 10 pin expansion connector, for ADC purposes (Header 2), schematic [57].....	113
Figure 77	CMU-FF's CC2420 radio transceiver configuration schematic [57].....	114
Figure 78	CMU-FF's expansion connectors for UART (Header 1), ISP programming (Header 3), mini-SD memory card or GPIO (Header 4), push-button and LEDs schematics [57].....	115
Figure 79	CMU_FF's battery connections, voltage regulator, ON/OFF button and ADC_BUS_0 schematics [57].....	115
Figure 80	CMU-FF's mini-SD card slot schematic [57] .....	116
Figure 81	CMU-FF's dimensions [57] .....	116
Figure 82	CMU-FF's v2.1, without mini-SD card slot, main components overview.....	117
Figure 83	Instance of the RXDATA pin of the RX5001 (green) and RX5002 (blue) radio devices .....	119
Figure 84	Response of the RXDATA pin to carrier wave transmission.....	119



## Tables Index

Table 1	Mote-series evolution [19][66].....	11
Table 2	Expansion boards for MICAz and its respective features [21].....	13
Table 3	Raw GPIO control macros .....	72
Table 4	WiFLEX_main drivers.....	74
Table 5	WiFLEX_txsync drives.....	75
Table 6	Interpolation experiment results over varying $d$ .....	97
Table 7	Fundamental components for TX5002 OOK configuration.....	121
Table 8	Fundamental components for RX5002 OOK configuration .....	122
Table 9	Fundamental components for RX5001 OOK configuration .....	123
Table 10	Fundamental components for TX5001 OOK configuration.....	124
Table 11	WiFLEX_main board hardware mapping .....	125
Table 12	WiFLEX_txsync board hardware mapping.....	126
Table 13	Winning node alone .....	127
Table 14	Losing node alone .....	127
Table 15	Winning node at 0 m distance .....	128
Table 16	Losing node at 0 m distance .....	128
Table 17	Winning node at 1 m distance .....	129
Table 18	Losing node at 1 m distance .....	129
Table 19	Winning node at 5 m distance .....	130
Table 20	Losing node at 5 m distance .....	130
Table 21	Winning node at 10 m distance .....	131
Table 22	Losing node at 10 m distance .....	131
Table 23	Winning node at 15 m distance .....	132
Table 24	Losing node at 15 m distance .....	132
Table 25	Winning node at 20 m distance .....	133
Table 26	Losing node at 20 m distance .....	133
Table 27	Combined probability of winning node success from 0 m to 20 m.....	134
Table 28	Combined probability of losing node success from 0 m to 20 m.....	134
Table 29	Combined probability of both nodes success probability from 0 m to 20 m.....	135
Table 30	Failure to lose the tournament counting.....	137
Table 31	Failure to win the tournament counting .....	138





## *Acronyms*

ADC	- Analogue to Digital Converter
ASH	- Amplifier-Sequenced Hybrid
BI	- Beacon Interval
CAP	- Contention Access Period
CAN	- Controller Area Network
CCA	- Clear Channel Assessment
CFP	- Contention Free Period
COTS	- Commercial off-the-shelf
CPU	- Central Processing Unit
CSMA	- Carrier Sensing Multiple Access
CSMA/CA	- Carrier Sensing Multiple Access / Collision Avoidance
CSMA/CD	- Carrier Sensing Multiple Access / Collision Detection
EDF	- Earliest Deadline First
EMC	- Electromagnetic Compatibility
ESR	- Equivalent Series Resistance
GPIO	- General Purpose Input/Output
GPS	- Global Positioning System
GTS	- Guaranteed Time Slot

HF	- High-Frequency
IC	- Integrated Circuit
IDE	- Integrated Development Environment
ISP	- In-System Programming
LED	- Light Emitting Diode
LF	- Low-Frequency
LPF	- Low-Pass Filter
LR-PAN	- Low-Rate Personal Area Network
MAC	- Medium Access Control
MBD	- Multiple Broadcast Domain
MCU	- Microcontroller Unit
MSOP	- Mini Small Outline Package
OOK	- On-Off Keyed
PAN	- Personal Area Network
PCB	- Printed Circuit Board
PPDU	- PHY Protocol Data Unit
RF	- Radio Frequency
RISC	- Reduced Instruction Set Computer
RP-SMA	- Reverse Polarity – Standard Male Adapter
RSSI	- Received Signal Strength Indicator
RTOS	- Real-Time Operating System

- RTS - Real-Time System
- SD - Superframe Duration
- SNR - Signal-Noise Ratio
- SPDT - Single Pole Double Through
- TDMA - Time Division Multiple Access
- WSN - Wireless Sensor Network



# 1. INTRODUCTION

Nowadays the usefulness of computers and computing systems is undeniable. In the past two decades, mankind witnessed revolutionary technology developments that solved, or greatly alleviated, an immense number of daily problems through the use of computer based solutions integrated with communications.

Besides the developments of computers in terms of processing speed, memory, overall size and weight, energy consumption and cost reduction, we are witnessing unprecedented and intensive growth of the interaction between computing systems and the physical environment. New sensors and actuators are changing the way computers perceive and interact with our world.

As an example, a mid-class vehicle is orchestrated by a network of embedded computing systems. From the front to rear, messages are exchanged between the vehicle onboard devices, providing status as diverse as the level of fuel in the tank to the position of the cylinders, or the on-vehicle temperature to next fuel injection timing. Such embedded computing systems have gone through extraordinary evolutions and their applications proliferate, improving each day our safety, comfort, health, quality of life and entertainment.

Traditionally, networks of embedded computing systems, or simply networks of embedded computers employed wires or optic fibers as their communications channel. But given the unfeasibility of using wired communication channels in many new applications, for example in agricultural monitoring requiring the deployment of tens (or hundreds) of computing nodes over a vast area, wireless based solutions won the battle of the sensor networks paradigm, giving birth to the WSN. Due to the dissemination of this technology, costs are falling rapidly and applications are growing at the same speed. Expectations around massive WSN (in scale and/or density) are becoming real, although several questions are yet unanswered. At the present moment, researchers giving the small jump that will lead mankind's giant leap to the next level of embedded computing, computers interacting with physical environment and communication.

Problems are naturally arising. The obstacles inherent to the density and scalability of such WSN impose several limitations. Physical dynamics require that the WSN responds quickly. Timeliness aware applications impose real-time requirements not only to the applications that run in the WSN platforms, but also at the communications level. In fact, the way communication is performed in WSN is critical for its usefulness. While the routing layer is being exhaustively explored by the research community, the MAC layer as yet several unexplored paths, in particular the ones concerning real-time applications [42]. The MAC layer has a crucial effect on how the communication channel is shared and efficiently used, and it is undeniable that scalability and density problems, naturally inherent to massively populated networks, may find solutions in the improvement of such important discipline of WSN. In fact, the category of the dominance/binary countdown MAC protocols, deeply explored for the wired systems by the Controller Area Network (CAN) protocol, is well known for providing real-time communications. In what concerns the wireless medium, due to its intrinsic physical characteristics, such a category of MAC protocols has not yet seen such extraordinary developments and applicability. Behind that lies the difficulty in obtaining the logic AND behaviour of the channel required for the MAC protocol implementation, along with several hardware limitations. The work done so far by the WiDom research team of the CISTER/IPP-Hurray! R&D group showed how gracefully such problems could be solved, and exploited the usefulness of such protocols not only for wireless MAC, but also for the efficient computation of aggregate quantities in dense networks, ultimately allowing such computations to be independent of the number of

nodes in the network and spending only one message transmission time to calculate certain quantities.

Nevertheless, the inefficiency of the physical layer of the MAC layer still does not allow extracting the full potential of such novel improvements. The hardware shortcomings associated to existing COTS WSN platforms do not comply with this new WSN MAC paradigm.

Therefore, in this Thesis, I present a platform to solve the problem of the absence of a platform to implement wireless dominance protocols in an efficient way. The overhead of the new platform is one order of magnitude less when compared to the original WiDom implementation in COTS WSN platforms, allowing the prioritized and collision-free medium access to be performed in less than 5 ms.

The remaining of this Thesis is organized as follows:

- Section 2 provides the necessary background on the work developed. The first part of this section, Sub-section 2.1, states basic definitions on RTS in order to introduce some of the concepts referred along the subsequent text. Sub-section 2.2 summarizes aspects related with commonly used WSN platforms, and places the reader in the context and scope of the developed work — real-time aware MAC in WSN. Sub-section 2.3 covers the necessary background on dominance/binary countdown protocols, and explains the details of the adaptation of such protocols to the wireless medium by presenting the WiDom protocol.
- The main work behind this Thesis is presented on Section 3. This section describes the system architecture, the hardware and the software development. Sub-section 3.1 is specially focused on the options that led to the developed hardware. The hardware design is detailed in Sub-section 3.2, tackling aspects related with fundamental design considerations, discussion about component selection and the details behind the architecture of the implementation. Sub-section 3.3 describes aspects related with radio communications that must be considered prior to the software development present in Sub-section 3.4. This last sub-section describes the developed software, including the necessary drivers for the application layer implemented on the developed platforms
- Section 4 describes the experiments conducted to test the developed platforms. Such experiments were mainly divided in two sub-sections: Sub-section 4.1 is dedicated to



the experiments that allowed the determination of the developed platforms performance; whereas, Sub-section 4.2 presents a real-world test on a demanding application.

- In Section 5, the main conclusions drawn from the results obtained in Section 4 are summarized. Finally, the major contributions of this Thesis and the related future work are discussed.

## 2. BACKGROUND

Merging wireless communications and Real-Time Systems (RTS) is undoubtedly a complex task. Moreover, in the panorama of WSN, where agents' inherently scarce reliability compromise system's predictability [63], critical applications tend to be set aside whenever wired based solutions are possible. However such goal is not impossible and applications proliferate where the requirements inherent to RTS are met. Additionally, the scale and density of future envisioned sensor networks applications impose requirements, namely in terms of (wired) links establishment, that will result in unfeasible challenges unless wireless based solutions are pursuit.

RTS is a very meticulous discipline, where concepts and definitions must be extremely well stated to clearly transmit their correct meaning. However some of these concepts may be briefly outlined to provide the necessary knowledge to understand its paradigm. In the particular scope of this thesis, it is important to understand the nature of the events that may occur in a RTS, along with the requirements inherent to them. Sub-section 2.1 presents this information.

WSN is as vast as complex. It gathers numerous areas of knowledge that cross all areas of Engineering, Physics and Mathematics. As an example, to achieve the best battery duration it is important not only to select the components consuming the lowest energy, but also to

envision the less demanding solution for the target application, the more efficient program, the most efficient antenna, the appropriate materials and design, *etc.*, and simultaneously cope with hard restriction in terms of cost. In this sense Sub-section 2.2 introduces some of the most prominent WSN platforms currently used, as well as it underlines the most important aspects inherent to real-time communications in WSN, and its related state of the art along with the current limitations. Special emphasis is given to MAC issues, which is justified by its critical role in the achievement of an efficient, yet real-time performing solution.

Sub-section 2.3 focuses on the perspective of solving the current state of the art limitations through the implementation of a MAC protocol based on the special family of dominance/binary countdown based schemes. Here is introduced the WiDom protocol, the reasoning behind its conception, and the limitations imposed by the currently available WSN platforms. This former issue unveils the requirements inherent to the platform developed in the scope of this thesis.

## 2.1. REAL-TIME SYSTEMS TERMINOLOGY AND GENERAL DEFINITIONS

In the wide context of engineering, the word “system” is usually applied to an abstraction layer, over a given group of procedures, in which the designer only wants (or needs) to know the combined output responses of such group to stimulus applied to its inputs. Therefore will be considered that a system is a mapping of a set of inputs into a set of outputs [37].

Not only in the engineering field, but in every real system (biological, natural, etc.), due to the nature of every physical phenomenon, responses do not present themselves instantaneously to a given stimulus. Every reaction takes time to occur, more correctly a finite time. This time between the presentation of a set of inputs to a system (stimulus) and the realization of the required behaviour (response), including the availability of all associated outputs, is called the response time of the system [37]. For example, imagine an electric heater (system) to which is applied electric energy (stimulus); the time elapsed until warm is sensed at its heating elements (response), is the system’s response time.

When the response time of a system is not subject of concern, generally, the focus of its analysis goes to the correctness of its outputs, according to a characteristic behaviour, in response to the applied inputs. In such systems, correct responsiveness is time independent. Providing a fast or a slow response does not benefit, nor harm, the correctness of the results (a pocket calculator taking five seconds to calculate a sum is as accurate as one doing the same task in one nanosecond).

In contrast with the former type of systems, in RTS, logical correctness is based on both correctness of the outputs and their timeliness [37]. Although a system does not have to process data in microseconds to be considered real-time it must simply have response times that are constrained. Rather than being fast (which is a relative term anyway), the most important property of a real-time system should be predictability; that is, its functional and timing behaviour should be as deterministic<sup>1</sup> as necessary to satisfy system specifications. For example, correctly solving all questions of an Algebra exam in twenty

---

<sup>1</sup> In this context, “deterministic timing” is related with the determinism of upper bounds rather than the occurrence of individual events. If it is possible to determine an upper bound on the time from when the computer is requested to perform a computation until the computation has finished for all possible events, the timeliness of individual events will be bounded hence possible to deal with in a predictable fashion.

minutes gives you the same grade as if it was solved in fifty minutes when having an hour to do such task.

Fast computing is helpful in meeting stringent timing specifications, but fast computing alone does not guarantee predictability [65].

#### 2.1.1. REAL-TIME SYSTEMS CLASSIFICATION

It might seem that all practical systems are, in some way, RTS. In fact [36], [38], and [65], clearly state it. In this sense, some definitions arose to classify RTS in a way that could group some of their main aspects, and provide a clear distinction from general non-real-time systems.

Generally, a system in which performance is degraded but not destroyed by failure to meet response-time constraints is classified as a **soft RTS** (experiencing typing delay in a computer running a word processor does not introduce spelling errors but is uncomfortable to the writer). All practical systems minimally represent such systems. Those in which a few missed deadlines will not lead to total failure, but missing more than a few may lead to complete and catastrophic system failure, are named **firm RTS** (failing to take an antibiotic pill at the correct time once may be harmless, but failing more than once may compromise the complete treatment). Ultimately, a **hard RTS** is one in which failure to meet a single deadline may lead to complete and catastrophic system failure [37] (in a car crash collision, failing to fire an airbag at time may lead to severe injury or death of the driver).

#### 2.1.2. EVENTS AND ITS OCCURRENCE NATURE

In any system, the input stimulus is conditioned by some kind of event. The event nature and its inherent characteristics determine several aspects related with the respective RTS. For example, if a system is time constrained, *i.e.* a RTS, its responsiveness must comply with the event occurrence in order to achieve a correct result. In this sense the event occurrence nature might be classified as follows:

- Periodic events: events in which the occurrence is periodic, and such period is well defined [12][18].
- Aperiodic events: events in which the occurrence period is absent [18].

- Sporadic events: events in which the precise occurrence time is unknown, but a minimum bound in its occurrence is known [12].

While periodic events are efficiently dealt with using simple polling based solutions, when it comes to dealing with sporadic events in RTS efficiency and real-time behaviour is not trivial to be achieved. On one hand if the deadline associated to the sporadic event is short, the polling period must also be short in order to satisfy this time constraint. Although, if the minimum bound between two consecutive occurrences of the same event is simultaneously long, short polling periods are extremely inefficient. For example, imagine the scenario of a fire alarm: it is desired that a fire alarm reports the occurrence of the hazardous event in a few seconds, surely less than 10 s after the detection occurs; however a fire may only occur days, months or years after the detection system activation, hence polling a fire sensor with a period that copes with the desired system response time is extremely inefficient. Moreover, if the number of sensors is large, the overall efficiency decreases or the response time is compromised.

## **2.2. WIRELESS SENSOR NETWORKS**

Typically, sensor networks are considered distributed computing platforms severely constrained by limited central processing unit (CPU), memory, energy and bandwidth resources. The unreliability inherent to individual nodes in the network is common and the network topology changes dynamically. In comparison with other networks, sensor networks also differ due to its tight interaction with the surrounding physical environment through sensors and actuators [63].

Moreover, WSN are characterized by having a large number of nodes that must work collaboratively to accomplish a given objective. It is expected that the incoming future brings the necessary evolution of processing, memory and transmission of information in wireless networks that enable the construction of WSN composed by thousands of tiny individually inexpensive nodes [2]. In this sense, it is predicted that density and scalability of the WSN will bring some of the most important challenges in terms of transmitting the information from each node, and the way that the transmission medium is shared among every individual nodes. Clustering and data aggregation may provide solutions to handle large quantities of information in dense or large-scale WSN, however one may predict that, alone, such solutions will not be sufficient due to the effort of clustering such amounts of

information and its impact over system's response times. In this context the MAC protocol takes a prominent position in the global performance of the WSN, in the sense that it defines the way the nodes in the network access, in a shared way, the communication channel, hence the way such information is conveyed and shared.

Along the following sub-sections, after an overview of some of the commonly used WSN platforms (with special attention on the FireFly, from Carnegie Mellon University, and MICAz, from Crossbow Inc. and Berkeley University of California), will be discussed some of the most important issues related with the MAC in WSN, its impact and actual state of the art, in the scope of real-time communication. The remainder of this sub-section (2.2) presents the MAC protocol behind the main motivation of this thesis's related work.

### 2.2.1. COMMONLY USED WSN PLATFORMS

Typical WSN platforms are designed to be as compact and inexpensive as possible, and are usually composed by a microcontroller unit (MCU), radio transmitter/receiver, sensors, programming and extension connectors, deployed over a printed circuit board (PCB). Eventually, external memory, interfaces (like light emitting diodes (LED) and buttons) and antennas, or antenna connectors may also be present.

Nowadays several WSN platforms are available, which differ in sensing capabilities, CPU speed and architecture, radio device and data-rate, available memory for data and program, transmission range, among many others. For example, the work present in [66], although far from being complete and having concluded in March 2007, reports a state of the art in WSN platforms where more than forty different devices are referred.

One of the most known WSN platforms within the research and development community are the Mote-series from the University of California Berkeley [66], in particular the MICA family of WSN platforms, which by 2003, already had more than one hundred research groups using it [30]. Although the success inherent to these platforms can be related to commercial effort of Crossbow Inc., which is the official vendor of the referred devices, the fact that UC Berkeley started developing this devices in 1998, with the WeC mote, and simultaneously started the development of an OS, the TinyOS (that comfortably supported the necessary hardware abstraction for application developers), greatly motivated the dissemination of the Mote-series. Table 1 summarizes the evolution of the Mote-series from the development of the WeC mote to the MICAz.

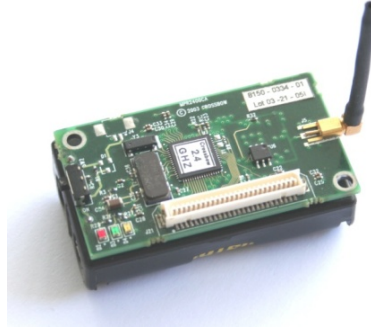
Table 1 **Mote-series evolution [19][66]**

Model	WeC	Dot	MICA	MICA2Dot	MICA2	MICAz
Release year	1998	2000	2001	2002	2002	2004
Microcontroller						
Model	AT90LS85 35	ATmega163	ATmega128	ATmega128	ATmega128	ATmega128
Program memory (kB)	8	16	128	128	128	128
RAM (kB)	0.5	1	4	4	4	4
Radio transceiver						
Model	RFM's TR1000	RFM's TR1000	RFM's TR1000	Chipcon's CC1000	Chipcon's CC1000	Chipcon's CC2420
Data rate (kb/s)	10	10	40	38.4	38.4	250
Modulation type	OOK	OOK	ASK	FSK	FSK	O-QPSK
Receive mode power consumption (mW)	9	9	12	29	29	38
Transmit mode power consumption at 0 dBm (mW)	36	36	36	42	42	35
I/O Expansion interface	None	None	51 pin	19 pin	51 pin	51 pin

#### 2.2.1.1. THE MICAz

The MICAz, depicted in Figure 1, is one of the latest generation of motes from Crossbow Inc.. Its conceptual base comes back from the MICA WSN platform developed at UC Berkeley in 2001. It uses the Chipcon's CC2420, IEEE 802.15.4 compliant, ZigBee ready radio transceiver, along with an Atmega128L MCU. The same MICA2, 51 pin I/O connector, and 512 kB serial flash memory is used, and all MICA2 application software, sensor and actuators boards are compatible with the MICAz [19]. Most of the architectural improvements, relative to prior versions like MICA and MICA2, were inspired in the work presented in [30], which led to considerable performance improvements.












**Figure 1 Crossbow's MICAz WSN platform**

The strong similarities with its predecessors, along with the tight cooperation of Crossbow Inc. with UC Berkeley, made TinyOS to be the native OS for the MICAz. The integration of this device with MoteWorks™ software enables the development of custom sensor applications. MoteWorks™ is based on the open-source TinyOS and provides ad-hoc mesh networking, over-the-air-programming capabilities, cross development tools, server middleware for enterprise network integration and client user interface for analysis and a configuration [19].

Throughout Appendix A are depicted the hardware schematics of the MICAz WSN platform, including detailed overview of its 51 pins expansion connector, as well as its physical dimensions and main components outline.

The MICAz WSN platform comes without any onboard sensor or actuators. Such capabilities are achieved through a group of seven available sensor and actuators expansion boards (which fit in its 51 pins expansion connector) allowing the sensing of acceleration (2 axis), light, pressure, magnetic field, sound, temperature and humidity; a global position system (GPS) enabled board is also available as well as extension boards featuring twelve and sixteen bits analogue to digital converter (ADC) inputs; relay actuators, general purpose input/output (GPIO) and audio output (through a buzzer) can also be found in the referred expansion boards [21]. Table 2 summarizes the available expansion boards and its main features.

Table 2 Expansion boards for MICAz and its respective features [21]

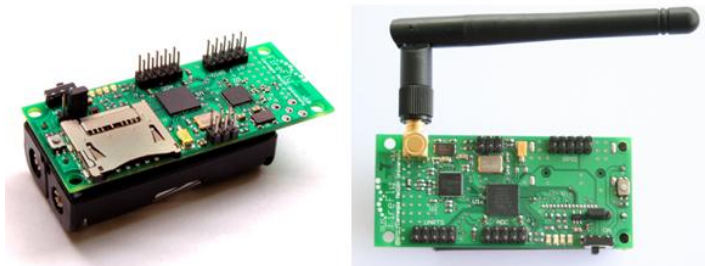
Board	MDA300	MDA320	MDA100	MTS300	MTS310	MTS400	MTS420
Features							
Accelerometer (2 axis)					YES	YES	YES
Magnetic field					YES		
Barometric pressure and temperature						YES	YES
Relative humidity and temperature	YES					YES	YES
Thermistor			YES	YES	YES		
Photo-sensitive light						YES	YES
Photoresistor			YES	YES	YES		
Ambient light						YES	YES
Microphone				YES	YES		
External analogue sensor inputs	YES (12 b)	YES (16 b)	YES (10 b)				
GPS							YES
GPIO	YES	YES	YES				
Actuator relays	YES						
Buzzer				YES	YES		

### 2.2.1.2. THE CMU-FF

The FireFly is a low-cost and low-power WSN hardware platform designed at the Carnegie Mellon University (reason why it is designated as CMU-FF) with an overall architecture similar to the MICAz WSN platform. In order to better support real-time applications, the system is built around maintaining global time synchronization [57]. Although capable of running TinyOS, the CMU-FF has its own dedicated OS – the Nano-RK OS. Nano-RK is a fully preemptive reservation-based real-time operating system (RTOS) from Carnegie Mellon University (CMU) with multi-hop networking support for use in WSN. Currently it runs on the CMU-FF and MICAz WSN platforms. It includes a light-weight embedded resource kernel (RK) with rich functionality and timing support using less than 2 kB of

RAM and 18 kB of ROM. Nano-RK supports fixed-priority preemptive multitasking for ensuring that task deadlines are met, along with support for CPU, network, as well as sensor and actuator reservations. Tasks can specify their resource demands and the operating system provides timely, guaranteed and controlled access to CPU cycles and network packets. Together these resources form virtual energy reservations that allow the OS to enforce system and task level energy budgets [44].

Figure 2 presents two identical versions of CMU-FF v2.1 (one with mini-SD card slot and the other without mini-SD card slot but with alternative GPIO expansion connector). This device uses an Atmel ATmega1281 8-bit MCU with 8 kB of RAM and 128 kB of ROM along with Chipcon's CC2420 IEEE 802.15.4 standard-compliant radio transceiver for communication. Although having an optional RP-SMA connector for an external antenna, the CMU-FF has a built-in 2.4 GHz PCB antenna. This last characteristic, along with the fact of only having components on the top layer of the PCB, allowed an effective cost reduction in what concerned components and manufacturing costs. Unlike the MICAz WSN platform, several low-cost and easy prototyping connectors are featured in the top surface of the PCB for communications, programming, coupling of add-on sensor and actuators boards, and GPIO.



**Figure 2 CMU-FF v2.1 with mini-SD card slot (left) and top view of the CMU-FF v2.1 with full wave size RP-SMA 2.4 GHz antenna and alternative ten pin GPIO connector (right)**

In Appendix B are presented several hardware schematics of the CMU-FF WSN platform. Details on the MCU and radio transceiver configuration are depicted, as well as the platform's interfacing connectors, physical dimensions and main components outline.

Although the first version of the CMU-FF had featured on-board sensing capabilities (motion, light, temperature and acceleration) the current version (v2.x) does not feature any on-board sensor. However simple human-machine interaction can be performed through a simple push-button. Currently have been developed five different expansion

boards, from which two of them are so far dedicated to R&D purposes, ranging a diverse set of features:

- **Sensor Expansion Card:** this add-on board, depicted in Figure 3, provides light, temperature, audio, passive infrared motion, three axis acceleration and battery voltage sensing [24].

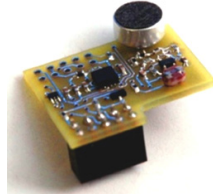


Figure 3 CMU-FF's Sensor Expansion Board (prototype version) [24]

- **FireFly-power-control:** this expansion board for the CMU-FF was developed with goal of controlling AC loads and monitor its energy consumption. The unit is able to measure energy consumptions from 10 W to 1000 W [23].



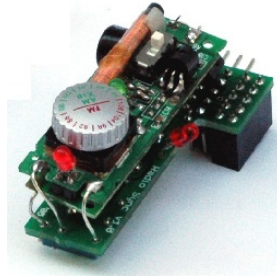
Figure 4 FireFly-power-control expansion board [23]

- **CMUcam3:** the CMUcam3 is an ARM7TDMI based fully programmable embedded computer vision sensor which provide simple vision capabilities to small embedded systems in the form of an intelligent sensor [16]. Figure 5 depicts the CMUcam3 kit.



Figure 5 CMUcam3 [17]

- **Hardware Clock Synchronization:** Figure 6 depicts the hardware board, dedicated for R&D purposes, to exploit the reception of global synchronization AM/FM signal [25].



**Figure 6** AM/FM signal receiving board [22]

### 2.2.2. THE IMPORTANCE OF THE MAC PROTOCOL IN WSN

Taking in account the fact that the data transmission is typically the most expensive operation performed in WSN in terms of energy expenditure [30], the MAC protocol arises as one of the most important sub-layers inherent to such operation. In [74] several key characteristics of WSN that justify the effort of investigating MAC protocols are identified. These can be summarized by:

- **Collaborative nature of WSN:** the computer nodes that compose the WSN usually have to work in collaborative way to accomplish a reduced number of applications, although, at a given moment, one single node may have more relevant information to the system than all the nodes together, and in this sense, conversely to other usual communication systems, fairness at node level becomes less relevant than overall application performance.
- **Sporadic information processing and delivery:** The tight interconnection of WSN towards its surrounding environment, often leads its applications to be designed to respond to stimuli from such environment. These stimuli commonly occur sporadically, which requires that nodes stay idle most of the time, *i.e.*, working at small duty-cycles. Although, when important events occur, the network activity is set up to high levels. Furthermore, if the network density is such that neighbour nodes can sense the same event, or its propagation, this activity is often spatially-correlated [32].
- **In-network processing:** It is common that nodes organize themselves into a convergecast tree with a base station at the tree root. Leaf nodes broadcast their data, having all other nodes waiting for a broadcast of its respective children. A node having children, aggregates its incoming data and makes a single broadcast towards the root (or

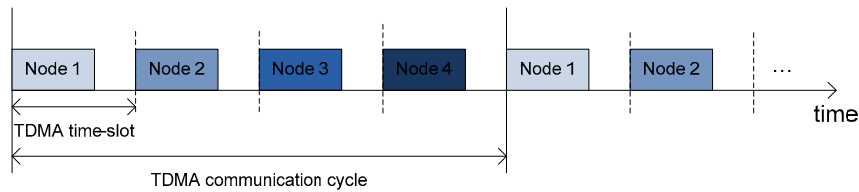
to a parent having similar behaviour). In this way or another, instead of blindly forwarding all data, nodes can perform some processing over the received data, and hence avoid spurious or irrelevant transmissions. Such techniques do not make any assumptions about the MAC protocol used, hence taking no advantage of it. Conversely, [4] and [75] showed that is possible to greatly reduce the number of transmitted messages necessary for performing distributed computations or gathering certain types of data, exploiting the proprieties of the used MAC protocol.

- **Lack of mobility:** In the majority of the WSN applications nodes are assumed to be static, hence the MAC protocol may be designed to exploit the relatively static neighbourhood of each node. Nevertheless, several factors may compromise the effectiveness of that strategy. As an example, it has been shown that radio irregularities cause connectivity between nodes to change even when its location remains static [76]. Even though researchers referred that this problem mainly affects the routing layer [76], the MAC protocol must still deal with this situations in an efficient way.
- **Energy efficiency, scalability and robustness:** Standard MAC protocol objectives, like fairness or latency, are usually considered secondary in detriment of energy efficiency, scalability and robustness of the network for sake of its lifetime. The typical ad-hoc manner in the deployment of WSN applications, the unpredictable operating environments, the scalability and adaptability to changes in the network's size, density and topology are also top relevant factors in the design of MAC protocols for WSN.

Although these characteristics are not always present in typical ad-hoc networks, especially simultaneously, the basic techniques in WSN MAC protocol design are based on the ones used for conventional ad-hoc networks, namely Time Division Multiple Access and Carrier Sensing Multiple Access. The following sub-section outlines the key aspects of the referred techniques, and their drawbacks in timeliness aware wireless MAC.

#### 2.2.2.1. BASIC TECHNIQUES IN WSN MAC PROTOCOL DESIGN

Besides coping with CPU, memory and energy constraints, data communications in WSN must be timely aware to effectively allow computing applications to interact with real-world physical events [63]. In this sense several strategies for sharing the communication channel in WSN have been exploited. The most relevant had as a reference the following techniques:



**Figure 7 Simple TDMA MAC**

- Time Division Multiple Access (TDMA): messages are assigned to time slots in a way that only one node transmits at each time. Commonly, the communication protocols operate based on TDMA cycles, where a node is assigned one or more time slots. Usually, both slots and cycles have a fixed length and number. Since a TDMA cycle is a fixed and known time duration, upper bounds on messages queuing can be determined. Figure 7 depicts a simple TDMA MAC scheme where four nodes are assigned a TDMA time slot, within a TDMA communication cycle; when the TDMA communication cycle is static, time slots are repeatedly assigned at each cycle. This technique requires that all nodes in the network share a common time reference in order to correctly access its assigned time slot.
- Carrier Sensing Multiple Access (CSMA): a considerable number of different versions of CSMA have been developed. In wired networks, like Ethernet, Carrier Sensing Multiple Access/Collision Detection (CSMA/CD) versions are the most outstanding. A CSMA/CD MAC relies on principle of sensing the medium for activity, before attempting to transmit. In case the medium is found busy the transmitter waits until it becomes idle, otherwise the transmission is done immediately. The fact that two or more transmitters sense the communication channel to be idle may lead to simultaneous transmissions and hence collisions occur. In such case, all colliding transmitters terminate transmission, wait a random time, and then repeat the medium access procedure again. Due to the inability to detect collisions during transmission, in wireless networks is often used the collision avoidance version of CSMA (CSMA/CA).

Conversely to CSMA solutions, TDMA-based techniques may seem extremely appealing for implementing MAC protocols in WSN, since it does not cause overhearing nor collisions (two of the main problems that must be avoided to achieve maximum transmission efficiency) and additionally nodes may go into sleep, and/or turn their radio devices off, between assigned communication slots. Although, the fact TDMA is only efficient when the network traffic is periodic, while many WSN applications are known to

cause bursty and/or sporadic traffic, added to the effort of setting-up the TDMA schedule between nodes may be a complex task, requiring the generation/exchanging of several messages (thus energy wasteful), may compromise its applicability in several real-world applications. Pre-runtime configuration may alleviate some of these problems, however, it goes against the self-organization and mobility premises of the WSN paradigm [66].

#### 2.2.2.2. WSN MAC PROTOCOL TOPOLOGY CONCERNS

As previously referred, WSN are characterized by its dynamic topology, even when nodes are static. This fact raises some problems that must be considered when designing a MAC protocol for WSN. Typically, in these networks two common problems occur: the *hidden node problem*; and the *exposed node problem*.

The hidden node problem occurs whenever, at least, two nodes are out of each other's range, while both are within the range of a third node. Figure 8 shows the simplest hidden node problem. In this case  $N_1$  and  $N_3$  do not share their transmission coverage, hence they cannot detect each other's transmissions, however  $N_2$  can be reached by both and *vice-versa*. In this case is very likely to occur collisions at  $N_2$ . The exposed node problem is typical in multiple broadcast domain (MBD) wireless networks [30]. Although not so important, in the perspective that it does not corrupt data transmissions, the exposed node problem reduces the number of parallel transmissions, hence reducing the overall throughput of the WSN and its responsiveness. Consider the scenario depicted in Figure 9. Notice that both  $N_2$  and  $N_3$  may refrain from transmitting to, respectively,  $N_1$  and  $N_4$  due to each other, *i.e.*, if  $N_2$  attempts to transmit to  $N_1$ , then  $N_3$  will not be able to simultaneously transmit to  $N_4$  because it detects  $N_2$  transmission. Conversely that same will happen for  $N_2$ . On the other hand  $N_1$  and  $N_4$  may, respectively, transmit to  $N_2$  and  $N_3$ . Whenever this occurs is said that  $N_2$  is exposed to  $N_3$ , and that  $N_3$  is exposed to  $N_2$ .

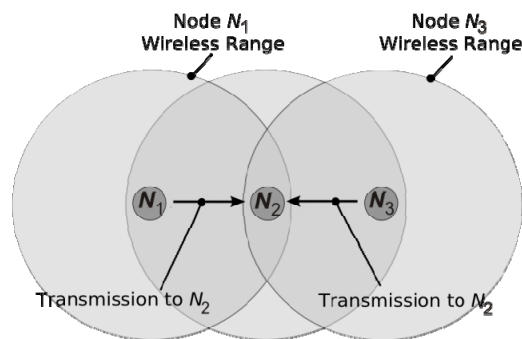


Figure 8 Hidden node problem



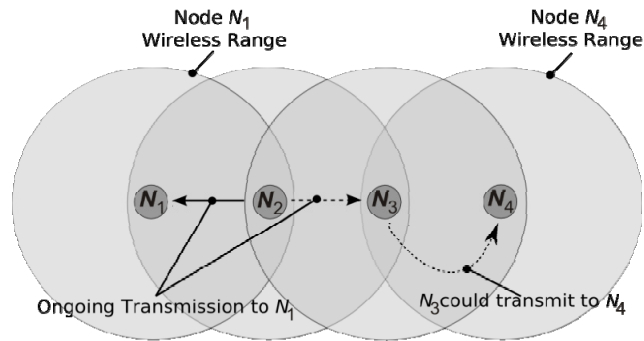


Figure 9 Exposed node problem

### 2.2.3. REAL-TIME COMMUNICATIONS IN WSN

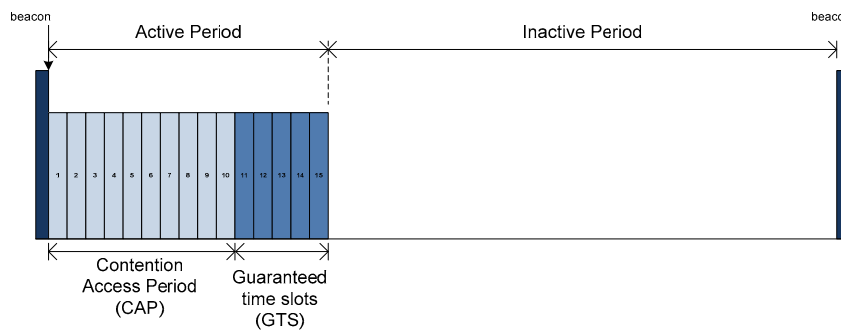
Supporting messages with deadline requirements in ad-hoc WSN is not trivial due to the limitations inherent to this kind of wireless networks. Adaptations of real-time protocols designed for typical ad-hoc wireless networks are not suitable due to the scarce CPU, memory and energy resources, typical in WSN platforms; the challenges that arise from dense and large scale WSN also compromise such adaptations. In such a specific context, the most relevant MAC protocols that fulfil the necessary requirements of real-time communications are based in the TDMA concept. This trend is tightly related with the fact that TDMA allows the possibility of scheduling precise transmission and receiving timings with the security of collision-free medium access, which are premises to very energy efficient communication channel sharing.

As referred in Sub-section 2.2.2.1, TDMA based protocols impose that all nodes share a common time reference. Such reference may be achieved through a global synchronization signal, exploiting out-of-band signalling transmissions that cover all broadcast domains. Although, the nodes synchronization is typically assisted by in-band signalling, and the multiple broadcast domain (MBD) problem is dealt with the formation of hierarchical organizations of the nodes into tree, clusters or cell based schemes, having a master node providing central coordination. This approach is inflexible to changes in the network topology and the number of participant nodes. Furthermore, TDMA based schemes have the drawback of requiring that sporadic message streams are dealt with resorting in polling, which is inefficient, especially when the deadline is short compared to the minimum inter-arrival time of the messages (refer to Sub-section 2.1.2).

Along the next lines some relevant MAC protocols used in WSN are shortly outlined, having in mind timeliness requirements:

- TRAMA: the traffic-adaptive medium access protocol (TRAMA) [59] is a MAC protocol based on TDMA that constructs schedules in a distributed manner and on an on-demand basis. Both scheduled slots and CSMA-based contention slots, for node admission and network management, are supported; and time slots assignment to nodes with no traffic to send is avoided. Nodes are also allowed to determine when they become idle and do not listen to the channel using traffic information. Unfortunately, this protocol may consume significant CPU and memory resources, since there is the need to maintain and perform computations upon the two-hop neighbourhood list of a node, which can be very large in dense WSN.
- RT-Link: in order to avoid in-band signalling (which reduce network performance), the authors in [58] developed a hardware platform to support a TDMA protocol based in out-of-band signalling for synchronization purposes. Further along, in [41], was explored the maximization of parallel transmissions over a TDMA network using RT-Link, providing optimal end-to-end throughput by identifying the maximal set of concurrent transmitters across the network, while maintaining a bounded delay. The result is achieved relying in a regular structure of node's deployment, which in practice is a non-typical scenario.
- Implicit EDF: the Earliest Deadline First (EDF) scheduling algorithm was originally developed for real-time tasks scheduling in computer processors. The implicit EDF approach implemented in [13] is based on the assumption that all nodes know the traffic on the other nodes that compete for medium, and all these nodes execute the same EDF scheduling algorithm. A node is only granted the permission to transmit if the message elected by the EDF scheduling algorithm is in its outgoing queue of messages. Since it is based on the assumption that a node knows the arrival time of the messages on other nodes, hence nodes must be accordingly deployed in static cells, and channel assignment needs to be carefully handled to avoid interference between neighbour cells. Nevertheless, this protocol also implies the use of polling to deal with sporadic message streams.
- IEEE 802.15.4: The IEEE 802.15.4 standard, described in [31], covers the physical and MAC layers of a Low-Rate Wireless Personal Area Network (LR-WPAN), and must not be confused with the Zigbee industry consortium, described in [77], which has the

goal of ensuring interoperability between different devices. ZigBee uses the services provided by IEEE 802.15.4 and defines the higher network layer and application interfaces. Several operating modes are defined in the IEEE 802.15.4 MAC layer, from which the most important, in what concerns dealing with messages having deadline requirements, is the beacon-enabled mode. When running this operating mode, nodes organize themselves in a Personal Area Network (PAN) and a coordinator node (named PAN coordinator) manages channel access and data transmission in a structure called *superframe*. As Figure 10 shows, the superframe is divided in two different periods, named the *active period* and the *inactive period*; while during the inactive period no transmission occur, and hence nodes can turn off their radio devices to save energy, during the active period sixteen time slots exists dedicated to medium contention and data transmission. The first time slot (slot 0) is dedicated to the beacon frame, and the remaining fifteen (slot 1 to 15) are used for the Contention Access Period (CAP) and for a maximum of seven Guaranteed Time Slots (GTS). Throughout the CAP, nodes access the medium using CSMA/CA, while the GTS is used for reservation-based TDMA. Nodes perform reservation requests of GTS slots during the CAP, to which the PAN coordinator may respond positively allocating the requested GTS slots to the node. Although, the beacon-enabled mode of 802.15.4 uses TDMA, and consequently it suffers from the drawback of TDMA schemes, *i.e.*, poor ability to deal with sporadic messages with short deadline and long minimum inter-arrival time.



**Figure 10 IEEE 802.15.4 superframe structure**

### 2.3. WiDOM

The WiDom protocol is a wireless medium access control protocol. Its primary goal is to coordinate wireless medium channel access in order to allow the scheduling of sporadic message streams in an efficient way, opposing the conventional TDMA based protocols or

polling based solutions (which are inefficient when the deadline is short and the minimum time between two consecutive requests is long). Additionally, it supports a large number of priority levels and is fully distributed. The characteristics of this MAC protocol also allow a prioritized and collision-free wireless medium channel access. Although WiDom is an adaptation of the CAN bus protocol to wireless systems, the fact that it is based on wired-AND behaviour of the bus (where the dominant signal overwrites the recessive one) together with the requirement that the nodes must be able to monitor the medium while transmitting, makes such adaptation non-trivial [49]. The work presented in [49] also describes a response-time analysis that allows the determination of an upper bound on the response times.

WiDom belongs to a class of protocols called dominance/binary countdown protocol. The main idea of dominance/binary countdown protocols is explained in Sub-section 2.3.1 whereas its use in wireless networks is discussed in Sub-section 2.3.2 and the details are given in Sub-section 2.3.3.

### 2.3.1. **DOMINANCE/BINARY COUNTDOWN PROTOCOLS**

In dominance protocols, nodes, or eventually messages, are assigned unique priorities at pre-runtime or during the system runtime. A node that performs a request for transmission waits for a pre-determined time interval until the communication channels gets idle. After that, a conflict resolution phase takes place, i.e. the arbitration phase. Then, each node that requested permission for transmission contends for the medium by sending its unique priority, bit-by-bit, starting with the most significant one, while simultaneously monitoring the communication channel. The communication channel must behave in a way that nodes only perceive a recessive bit if no node contends with a dominant bit. Conversely, if at least one node transmits a dominant bit all nodes should detect that dominant bit. It is imperative that, during the arbitration, a node that detects a dominant bit refrains from transmitting any subsequent bits. In the last case, the node will only monitor the communication channel in order to access the priority of the winning node. The node that reaches the end of the arbitration without perceiving any dominant bit will be the winning node, hence having the permission to transmit the queued message.

Figure 11 depicts the arbitration performed in dominance/binary countdown protocols. In this example three nodes (node 1, node 2 and node 3) with different priorities (respectively

01011111, 01110011 and 01010111) contend for the communication channel using a priority number based in  $2^8$  levels ( $2^n$  priority bits). In accordance with [43], let a bit that is “0” represent a dominant bit and, hence, a recessive bit be “1”. This will imply that lower priority numbers represent higher priorities. At the start of the arbitration phase all nodes are potential winners. All nodes contend for the channel starting with its most significant bit, going bit-by-bit until the least significant one is used. In the first contention moment, i.e. when the most significant bit is used, all nodes have the same priority bit, more precisely a dominant one. Since the resultant medium status is equal to each node priority bit, none assumes a defeated position, maintaining its potentially winner status. The same occurs for the next priority bit, although in this case all nodes contend with a recessive one. By the time the third priority bit is used, node 1 and node 3 contend with a dominant bit, and node 2 contends with a recessive one, meaning that node 2 will experience a channel status different from the bit it used during contention, which will result in losing the potential winner condition and consequently it withdraws from contention. Further along, until the end of the contention phase, this node will only monitor the medium status to determine the winner’s priority (represented by the dotted trace). The conflict resolution phase continues accordingly with the explained behaviour, resulting in the loss of potential winner condition of node 1 by the time of the fifth priority bit. After this bit, node 1 will act similarly to node 2, and node 3 will continue until it had contended with the last priority bit. By this time, node 3 will recognize it has won the conflict resolution phase, and starts to broadcasts its previously queued message while, conversely, node 1 and node 2 will know that node 3 has won, going in to reception mode.

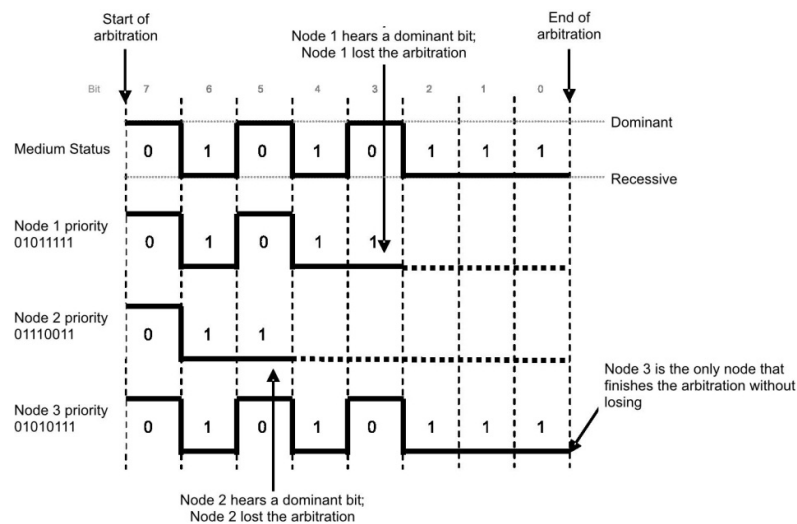


Figure 11 Arbitration in dominance/binary countdown protocols

The latter adaptation of the dominance protocol presented in [43] in the widespread CAN bus [11], led to a MAC protocol which is collision-free and prioritized, and therefore it is possible to schedule the communication channel, *i.e.* the bus, such that if the message characteristics, like periods, transmission times, jitter, *etc.*, are known, then it is possible to compute upper bounds on message delays [68].

### 2.3.2. DOMINANCE PROTOCOLS FOR WIRELESS MEDIUM ACCESS

Wired dominance MAC protocols, like the CAN bus, require that a node has the ability to perceive an incoming bit from the communication channel and simultaneously transmit a bit to the same channel [11][41]. Such behaviour is impossible to achieve on a wireless channel, and its reasoning can be justified resorting to the demonstration presented in [14]:

The free-space loss on a line-of-sight path, is due to spherical dispersion of the radio wave, and such loss is given by:

$$L = \left(\frac{4\pi l}{\lambda}\right)^2 = \left(\frac{4\pi f l}{c}\right)^2 \quad (1)$$

in which  $\lambda$  is the wavelength,  $f$  the signal frequency,  $c$  the speed of light and  $l$  is the point-to-point distance in line-of-sight. For a fixed wavelength, the loss increases with the  $l^2$ . Moreover, if  $l$  is expressed in kilometres (km) and  $f$  in GHz, Equation (1) becomes:

$$L_{dB} = 92.4 + 20\log_{10}f_{GHz} + 20\log_{10}l_{km} \quad (2)$$

showing that  $L_{dB}$  increases as the logarithm of  $l$ , rather than in direct proportion to path length. Hence, for example, doubling the path length increases the loss by 6 dB.

### 2.3.3. WIDOM IN DETAIL

To understand the concept behind the WiDom protocol, a few primitives related with the system model must be explained. Let us consider  $n$  message streams  $\tau_1, \tau_2, \tau_3, \dots, \tau_n$ , and  $m$  computer nodes  $N_1, N_2, N_3, \dots, N_m$ , assuming only one message stream is assigned to each node, although several message streams can be assigned to one node. The following definitions are fundamental [49]:

- **Workload:** a message stream  $\tau_i$  makes an infinite sequence of requests to transmit with an exact time of transmission request unknown but yet being known a lower bound on the time between two consecutive transmission requests from the same message stream (sporadic model). This lower bound is denoted as  $T_i$ . Every message from  $\tau_i$  requires  $C_i$  contiguous time units to transmit. The maximum time elapsed from the time instant of a

request from  $\tau_i$  to the conclusion of the transmission of that message is called the *response time* of  $\tau_i$ , and is denoted as  $R_i$ .

- **Success and failure:** if there is an overlap between a pair of transmitted data bits, then a collision has occurred and both transmissions have failed. Every time a message from  $\tau_i$ , is requested to be transmitted it needs to finish the transmission at most  $D_i$  (relative deadline of  $\tau_i$ ) time units after it was requested. The goal of the protocol is to schedule all messages in all message streams such that all transmissions are accomplished before their relative deadlines, and in absence of any collision of data bits. In this case the protocol has succeeded.
- **Priorities:** priorities are assigned univocally to message streams; these priorities are non-negative integers. The number of priority bits used to represent the priority numbers are denoted *npriobits*.
- **Propagation:** the time-of-flight of radio waves transmitted between two arbitrary nodes  $N_i$  and  $N_j$  is unknown, yet it is a non-negative value and there exists an upper bound  $\alpha$  on such time-of-flights. A single broadcast domain is assumed. When a node transmits a message and there is no collision, then all nodes receive exactly one copy of the message; that is, no hidden terminals exist.
- **Nodes:** nodes are equipped with real-time clocks, although these are not synchronized, i.e., their values may be different among themselves. Therefore is considered that for every unit of real-time, the clock increases by an amount in the range  $[1-\varepsilon, 1+\varepsilon]$ , for  $0 < \varepsilon < 1$ . The granularity of the clock is denote as *CLK*.

The protocol can deal with both messages having one intended node as receiver or all nodes (respectively unicast or broadcast). It is assumed than when a node receives a given message no acknowledgement is sent; this assumption can be removed for unicast messages by adding the acknowledgement time to the message transmission time. Other nodes transmission can only be sensed by a specific node if himself is not transmitting. It is not assumed any particular modulation technique or coding scheme for the data bits, although, when they are being transmitted, there is no interval of continuous idle time exceeding  $F$  time units. The time for detecting that a carrier wave was transmitted is denoted as *TFCS*, and the time for switching between transmission and reception modes as *SWX*.

The authors of [49] opted for describing the protocol state machine in timed-automata like notation. Such notation implies the assumptions that states are represented as vertices and transitions as edges. An edge is described by its guard (a condition which must be true in order for the protocol to make the transition) and an update (an action that occurs when the transition takes place). In the graphic representations based in timed-automata notation, “/” separates the guards and the updates; the guards are before “/” and the update appear next. The symbol “=” denotes the test for equality, and “:=” denotes assignment to a variable. When a timeout transition is enabled, it occurs immediately. The corresponding update of that transition and a continuing path of enabled transitions occur at most  $L$  time units later, meaning that  $L$  represents the delay due to execution on a finite-speed processor.

Figure 12 depicts the WiDom protocol in the referred notation. The protocol is composed by three main phases:

- **Synchronization phase:** since each node’s real-time clock is not synchronized, the protocol must ensure a common time reference between all nodes that participate in the sub-sequent protocol phases; this phase is responsible for establishing such reference.
- **Tournament phase:** when messages contend for the channel a conflict resolution phase, similar to the dominance/binary countdown arbitration phase outlined in Sub-section 2.3.1, takes place. In the WiDom protocol, this conflict resolution phase is named *tournament*. During this phase nodes transmit the priority of the message contending for the medium bit-by-bit. Although, due to the inability/impossibility to perform carrier sensing of carriers sent by other nodes when a node is transmitting (refer to Sub-section 2.3.2), when the transmitted bit is dominant there is no need to sense the medium, whereas, when a bit to transmit is recessive, nothing has to be effectively sent, instead only the medium state has to be monitored. The protocol distinguishes a bit of the tournament from a data bit, in the sense that a bit in the tournament has a fixed duration of time, which is considerably longer than a data bit.
- **Receive/Transmit phase:** Conversely to the tournament phase, when a node is authorized to transmit, data bits are much smaller and the full data rate of the transceiver is allowed to be used.



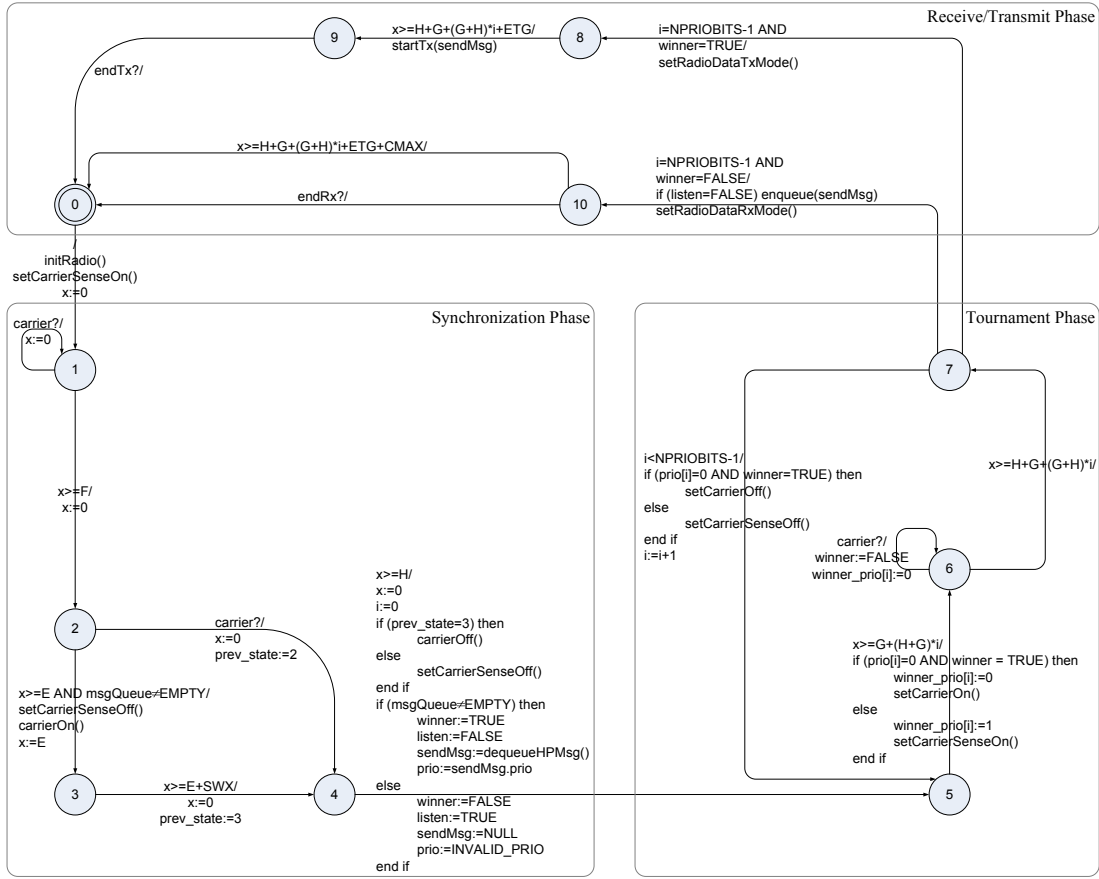


Figure 12 WiDom protocol state machine in timed-automata like notation [49]

Considering that clock imperfections, time-of-flight of carrier waves and delays in the transitions are negligible, for the sake of simplicity, they are omitted in the protocol representation. States are numbered from 0 to 10, and state 0 is the initial state. Each node as associated the following variables: a clock  $x$ ; an integer  $i$  within  $[0, npriobits - 1]$ ; an integer  $prio$  occupying  $npriobits$  bits; an integer  $winner\_prio$  occupying  $npriobits$  bits; and a boolean variable  $winner$ . Let  $winner\_prio[i]$  denote the bit  $i$  in the variable  $winner\_prio$ , and analogously for  $prio[i]$ .

Along the protocol representation, seven functions are called to represent determined actions:

- **initRadio()** – initialize the radio device and set it into a known starting state
- **setRadioDataRxMode()** – prepare the radio device to receive a data packet
- **setRadioDataTxMode()** – set the radio device for packet transmission
- **carrierOn()** – start the transmission of a carrier wave until the function `carrierOff()` is called

- **carrierOff()** – stop the transmission of a carrier wave
- **setCarrierSenseOn()** – set the radio device into receive mode and start detecting carrier wave pulses
- **setCarrierSenseOff()** – stop detecting carrier wave pulses
- **dequeueHPMsg()** – get the highest priority message from the local queue of message requests.

The symbol “carrier?” is used with the following meaning: sense for a carrier and if there is a carrier then “carrier?” is equal to true. Several different timeouts are used –  $F$ ,  $G$ ,  $H$ ,  $ETH$ ,  $E$  and  $SWX$ . Those timeouts are constants, and their values and meaning will be defined and reasoned out further along the text.

Consider Figure 12. States 1-4 belong to the Synchronization Phase, where a common time reference to all nodes that request to transmit is established. During State 1, nodes wait for a long period of silence ( $F$ ) such that no node disrupts an ongoing tournament. Then, nodes with a pending message perform transition 2 to 3 after  $E$  time units. This design is such that the duration of  $E$  encompasses possible clock differences between the nodes and guarantees that all nodes have time to listen for  $F$  time units of silence. Nodes that make the transition from 2 to 3 start transmitting a carrier pulse that signals the start of a tournament and establishes a common time reference. Other nodes in the network may perform one of the following sequences of transitions: (i) if in State 2 with pending messages and it did not perceive a carrier wave for  $E$  time units, it makes the transition to State 3; (ii) if in State 2 (either because it does not have any pending messages or is waiting to perform transition from State 2 to 3) and detects that a carrier wave pulse is being sent by other node(s) then it makes transition from State 2 to 4. While nodes performing as described in (ii) immediately reset their timers at State 4, nodes acting as described in (i) wait  $SWX$  time units before doing so, since only at that time the carrier wave pulse is actually transmitted. After that they stay in State 3 transmitting the synchronization signalling. After  $H$  time units in State 4, the transition to State 5 occurs, and at this point the Synchronization Phase ends with nodes resetting their timers.

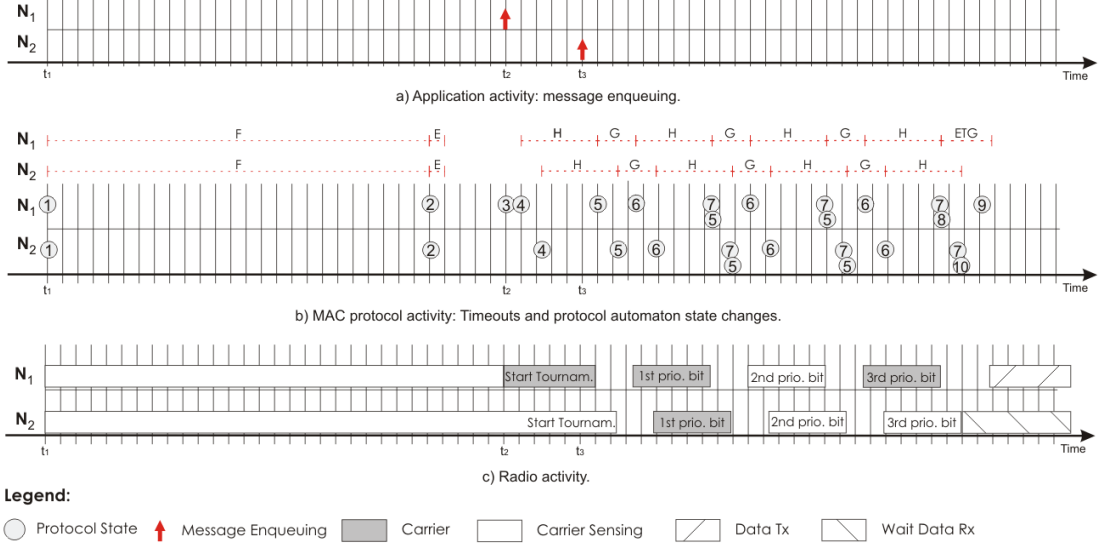
The Tournament Phase occurs along State 5 to 7. During this phase, a node which has lost the contention of a bit will refrain from transmitting any further contention bits, remaining the rest of the tournament listening to the ongoing tournament to access the winner’s priority. Nodes that do not lose the contention proceed contending bit-by-bit until the end

of the tournament. If a node contends with a dominant bit (“0”) then it starts transmitting a carrier wave pulse in the transition from State 5 to 6. Whereas, a node contending with a recessive bit (“1”), at the transition from State 5 to 6, starts performing carrier sensing. While at State 6, if a node contended with a recessive bit but perceived a carrier wave, then it has lost the contention.

The Receive/Transmit Phase takes place after the tournament ends. The unique winning node makes the transition to State 8, where it waits for a given amount of time so that the radios of all the other nodes can go into receive mode. After that, it transits to State 9, where it transmits the data content of the message. At the end of the data transmission the transition to State 0 is performed.

The nodes which have lost the tournament must take different path along the state machine evolution. During the tournament, in case of contention loss, the nodes continue in the tournament, and if they have a recessive bit the same transitions are made (this is due to the fact that during a recessive bit no carrier wave transmission is made). Although, if a node has a dominant bit and it has previously lost the tournament when it contended with a recessive one (the Boolean variable **winner** is FALSE), then the protocol performs differently from the case when it had won, no carrier wave is transmitted. Such nodes, at the end of the tournament, make the transition to State 10, waiting to receive the message or timeout. A node only receiving, *i.e.* with no queued messages, acts like a node losing the tournament from the start. This is because the variable **winner** is assigned FALSE before the tournament phase (at transition from State 4 to 5).

In order to understand the timeout parameters  $F$ ,  $G$ ,  $H$ ,  $ETG$  and  $E$ , consider the activity of  $N_l$  in Figure 13b.  $N_l$  enters State 1 at time  $t_l$ . From this moment on, node  $N_l$  starts performing carrier sensing until  $F$  time units elapse;  $F$  is then the initial idle time period duration. The time taken by  $N_l$  to transmit or detect a carrier is represent by  $H$ , therefore this represents the duration of a carrier wave pulse. A “guarding” time interval appears separating carrier wave pulses in order to make the protocol robust against clock inaccuracies and slight drifts, and takes into account that signals need a non-zero time to propagate from one node to another; this time is denoted as  $G$ .  $ETG$  is the gap that a winner must introduce at the end of the tournament, and  $E$  is a timeout used to improve the reliability introduced by imperfections imposed by the hardware during the synchronization (such as clock inaccuracies and transmit/receive switching times).



**Figure 13 Application (a), MAC protocol (b) and Radio (c) activity example in two nodes,  $N_1$  and  $N_2$  [49]**

Considering again Figure 12, one can understand that if the path of the transitions of the winning node are followed, and observed the last timeout (the transition from State 8 to 9), the transmission time of a message taking the overhead of the protocol into account can be computed. The time to transmit a message and perform the tournament when nodes are already synchronized is denoted as  $C_i'$ , and is given by:

$$C_i' = C_i + 2H + G + (G + H) \times (npriobits - 1) + ETG + E + \max\{TFCS, SWX\} + 2L \quad (3)$$

where  $C_i$  denotes the time required to transmit a message from message stream  $\tau_i$ . The time to transmit a message and perform the tournament when nodes are not yet synchronized is denoted  $C_i''$ , and takes into account the initial idle time:

$$C_i'' = C_i' + F \quad (4)$$

Assignment of the referred protocol parameters is explained in [49], as well as other details behind embracing the rationale of the design and its correctness.

The reliability achieved in the experiments described in [49] justified the study of schedulability analysis techniques for sporadic messages in wireless networks. In this sense, the authors also extended their work into the response-time analysis for WiDom, showing that it is possible to engineer industrial applications (from the timeliness perspective) to be evaluated in a similar fashion to what engineers do currently for CAN-based systems in industrial environments.



### 3. A PLATFORM FOR WIRELESS DOMINANCE MAC PROTOCOL IMPLEMENTATION

While wireless dominance was successfully achieved by WiDom [48], the implementations available are based on COTS WSN platforms, with a radio transceiver that does not have favourable characteristics for the implementation of a wireless dominance protocol, and thus, these implementations exhibited a considerable overhead. Specifically, the radio transceiver used in [48] and [49] was the Chipcon's CC2420 [15], a radio transceiver found in many WSN platforms. This transceiver, does not offer the most desirable characteristics for the implementation of wireless dominance protocols. While the specific reasons may vary, this is unfortunately also true for a number of other similar radios currently used in WSN platforms.

First, it is necessary to transmit a carrier wave for a small duration of time. While some radio transceivers allow to do this (like the CC2420), other radio transceivers only have a

byte interface with the microprocessor, which limits the granularity of the duration for the transmission of carriers and introduces unnecessary overhead.

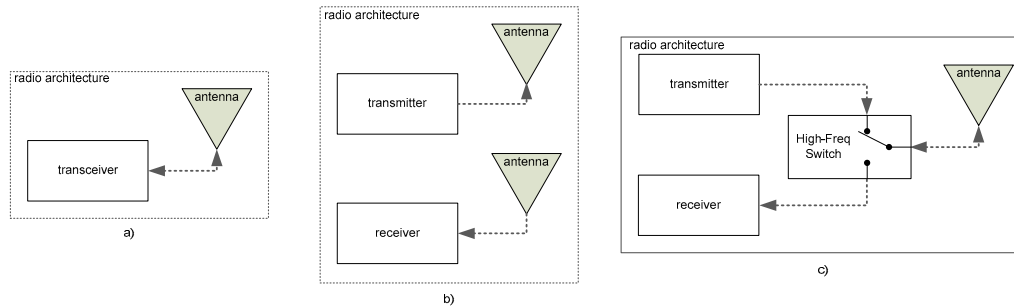
Second, it is necessary that the radio is able to detect whether other nodes transmit a carrier wave. The ability to perform detection of small pulses of carrier waves is instrumental for the development of an efficient wireless dominance protocol. For example, the CC2420, can detect pulses of carrier waves, using its Clear Channel Assessment (CCA) functionality. The CCA functionality of the CC2420 radio computes the average Received Signal Strength Indicator (RSSI) over the last 128  $\mu\text{s}$ . To make a decision, this average is compared to a configurable threshold and then the CC2420 sets the CCA digital output pin accordingly. In practice, this means that TFCS will never be smaller than 128  $\mu\text{s}$ . As an example, in [48], carrier pulses needed to be  $\text{TFCS} = 486 \mu\text{s}$  long in order to be reliably detected (using the transceiver's default threshold).

Finally, it is also necessary that the time to switch between transmission and reception modes is small. The CC2420, as an example, can take up to 192  $\mu\text{s}$  to switch between these two modes, and then it needs another 128  $\mu\text{s}$  ( $\text{SWX} = 320 \mu\text{s}$ ) until the first CCA operation can be made.

This hardware shortcomings imposed deeply penalizing restrictions in protocol parameters that directly influence the protocol's overhead, namely, in the switching time between receive and transmit modes ( $\text{SWX}$ ), unmodulated carrier wave pulse width ( $H$ ) and the necessary carrier sensing time ( $\text{TFCS}$ ), and consequently the guarding time interval to separate pulses of carrier waves ( $G$ ). In this way, in order to achieve an efficient implementation of the WiDom protocol, a new platform should fulfil the following requirements:

- short switching time between receive and transmit modes;
- smaller unmodulated carrier wave pulse width;
- carrier sensing time smaller than the unmodulated carrier wave pulse width.

It is therefore important to create a platform, with these issues in mind, envisaging an efficient implementation of the referred protocol.



**Figure 14 Radio architectures: a) transceiver based; b) separated transmitter and receiver with non-shared antenna; c) separated transmitter and receiver with shared antenna through an HF switch**

Figure 14 represent the three basic radio architectures to discuss in order to reduce the switching between reception and transmission modes. In (Figure 14a) a single radio device with transmission and reception capabilities, i.e. a transceiver; (Figure 14b) a transmitter and a receiver, working at the same frequency, with a dedicated antenna for each one; and (Figure 14c) a transmitter and a receiver, working at the same frequency, with a commonly shared antenna.

Disregarding the positive fact that the architecture represented in Figure 14a uses fewer components, (thus being potentially less expensive and using a smaller PCB area) such solution does not encompass the goal of reducing or eliminating the switching time between TX and RX modes, due to the fact that typical transceivers using on-off keyed (OOK) modulation present a considerable switching time between these two operating modes<sup>2</sup>. In this sense, considering that both transmitter and receiver are simultaneously ready to operate, the second solution (Figure 14b) brings the possibility of, theoretically, instantaneously switch between TX and RX modes. Nevertheless, since that in the same node cannot effectively transmit and receive at the same time (the receiver would only perceive the transmitter fitted nearby itself), the third option (Figure 14c) results as the most appealing one. In fact the use of an high-frequency (HF) switch to control the antenna path (from transmitter-to-antenna or from antenna-to-receiver), that has a switching time

---

<sup>2</sup> As an example, the Chipcon's CC2420 take up to 192  $\mu$ s to switch between TX and RX modes, and then it needs another 128  $\mu$ s until the first CCA operation can be made (which results in a 320  $\mu$ s switching time); and the RFM's TRxxxx series take at least 300  $\mu$ s to switch from TX to RX mode, spending 12  $\mu$ s to go from RX to TX mode.

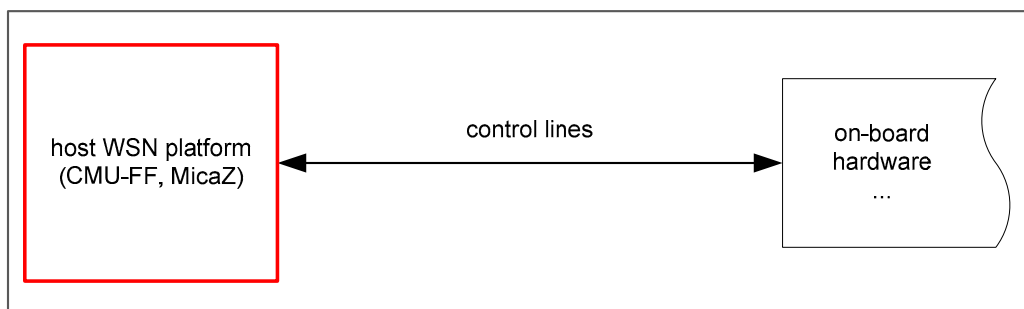


less than the duration of the CPU tick resolution is enough to consider the switching time negligible.

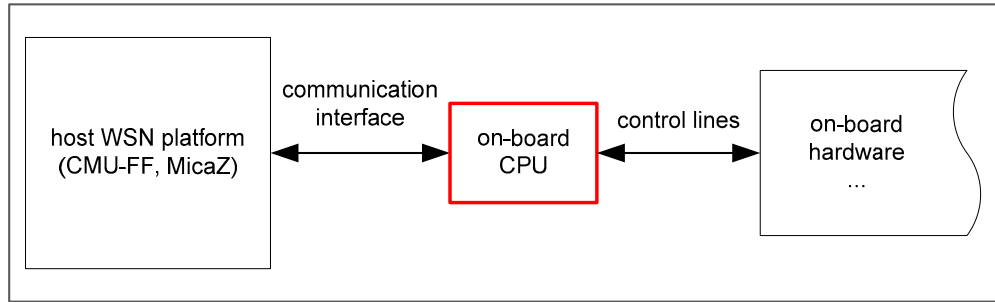
Another concern regarding the system architecture was the decision of (i) providing the platform with a CPU to run the protocol, being the interface with the host WSN platform made in a high-level fashion through a simple synchronous communication bus; or (ii) connecting all the necessary control lines to the host WSN platform, in which that protocol would run directly.

Figure 15 illustrates the concept behind (ii). According to the author of [30], this architecture is the most efficient in terms of resources utilization. Moreover, the author states that the inclusion of a dedicated protocol processor does not reduce the amount of computation that must be performed; it only provides a concurrency mechanism that allows application and protocol processing to occur simultaneously. Therefore, such partitioning of resources leads to non-optimal system resources utilization.

However, several reasons arise in this particular implementation to avoid such scheme. In the first place, the timing requirements for an efficient implementation of the WiDom protocol are not compatible with the applications runtime, and these resource requirements compromise the sharing of the CPU with most realistic applications. Secondly, the I/O pins needed to control the necessary hardware, and the additional fact that the platform must be compatible with the CMU-FF and the MICAz WSN platforms brings several hardware design difficulties and limits the exploitation and use of existing COTS add-on sensor boards. Finally, the adoption of such architecture would force the protocol implementation to be platform dependent (due to the different resources such as timers, radio, *etc...*, available at each host WSN platform).



**Figure 15 Non-concurrent MAC protocol execution architecture**



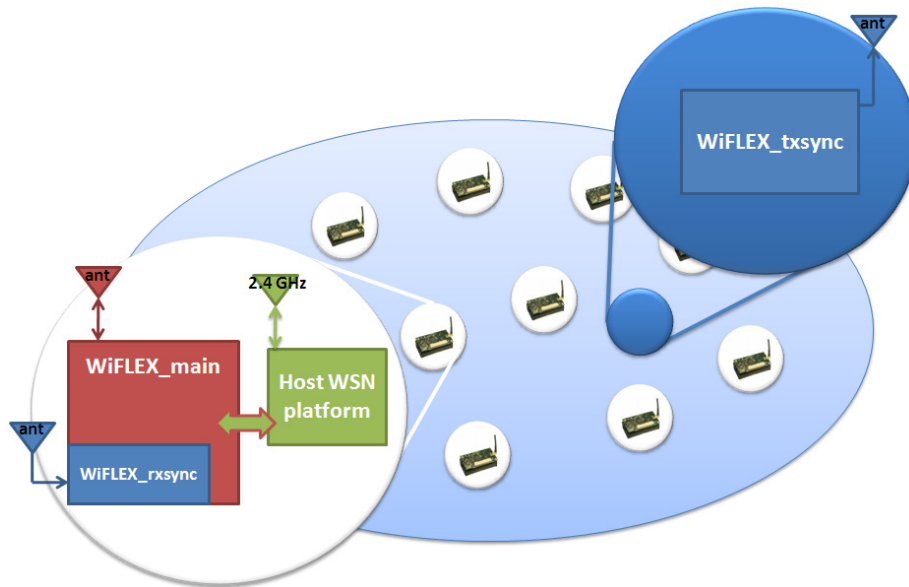
**Figure 16 Concurrent MAC protocol execution co-processor based architecture**

Figure 16 represents the adopted architecture for the platform design. Such architecture concerns both radio devices and the protocol processing needs to satisfy the necessary requirements. In this case, the protocol execution is confined to a dedicated CPU that controls the on-board hardware and provides a communication interface to the host WSN platform.

Along the next sub-sections, will be described the referred architecture (Sub-section 3.1), presenting in detail its specifications. Sub-section 3.2 will discuss all the details concerning the components involved, and the hardware design of the platform itself. In Sub-section 3.3 will be approached the underlying problems related with wireless communication and its reliability in this context. Finally, Sub-section 3.4 covers all the aspects inherent to the developed software for the platform, going from the communication interface with the host WSN platform to the WiDom protocol implementation.

### **3.1. OVERVIEW OF THE ARCHITECTURE**

Since the goal of the developed hardware platform is to allow the efficient implementation of the WiDom protocol versions, hence being expected the sufficient hardware flexibility for such purpose, it was named WiFLEX – FLEXible physical layer for WiDom implementations. Figure 17 presents an overview of the system architecture. Computing nodes present in the WSN are represented by a number ranging from 1 to  $n$  (1, 2, 3, 4, ...,  $x$ , ...  $n$ ). Each of them is composed by a host WSN platform (CMU-FF or MICAz) to which is coupled the main board – WiFLEX\_main board (or main board). The host WSN platform is responsible for all the WSN related activities (energy management, sensor reading, computation, actuators control, data communication, *etc...*) except the MAC. This last task is confined to the WiFLEX\_main board, which works as a peripheral of the host WSN platform, providing MAC capabilities to it in a transparent way.



**Figure 17 System overview**

The WiFLEX\_main board uses a specific band (different from the data transmission band) to perform the MAC. If the host WSN platform is granted medium access, then it uses its on-board transceiver to transmit data at a high data rate, as well as dedicated communication protocol. Nevertheless, it is also possible for the WiFLEX\_main board to transmit data, however at a lower data rate.

One of the WiDom protocol versions uses the reception of an out-of-band signal for synchronization purposes, allowing the reduction of the protocol's overhead by approximately 50 %. This can be achieved through the use of two additional components in the system, the WiFLEX\_rsync board (or daughter board) and the WiFLEX\_txsync board (or out-of-band synchronization signal transmission board), both of them relying on frequency band different from the one used to perform the MAC and the data transmission.

The next sub-section will introduce the referred platform and its main components, outlining its physical characteristics.

### 3.1.1.1. WIFLEX PLATFORM ARCHITECTURE

In order to allow the MAC execution to be independent of the platform, and/or its OS, and hence to allow an effective abstraction from the MAC layer, the WiFLEX\_main board was fitted with a low-power microcontroller unit (MCU) that runs the MAC protocol and offers the necessary mechanism so that the host WSN platform can, in a high-level fashion,

communicate with it. In this way, the developed platform pairs its operation similarly to common widespread WSN embedded radio modules like, for example, CC2420 [15], allowing the protocol to run in a concurrent way along with the host WSN platform's applications. This concurrent MAC protocol execution co-processor based architecture also copes with the well known constraints of scarce CPU and memory resources that are typical characteristics of common WSN platforms [63].

The well known fact that wireless channels typically have significantly higher noise levels than wired ones, and that the detection of pulses of short duration is difficult [69], made wireless communication systems often use long codes [60] to increase the probability of a correctly received message. This fact, allied with the resources limitations and added to the goal of low-power consumption in WSN, led the worldwide developers of radio devices for such applications to respond with mixed signal integrated devices that exploit high-end modulation techniques (such as spread spectrum) in a transparent way, taking care of all the lower level details, like modulation technique, carrier sensing, and so on, and only offers a communication and control interface in a high-level fashion.

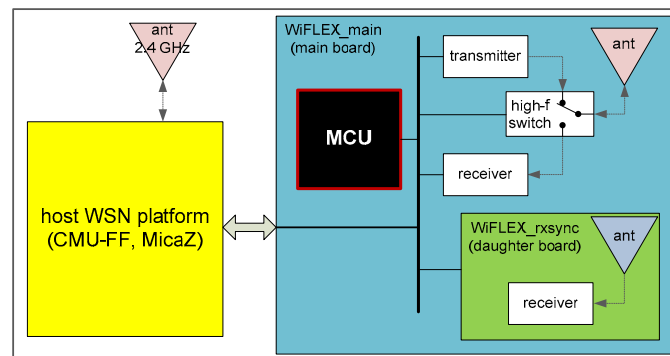
For the majority of the applications the benefits of such devices and implementations are undeniable. However, in this particular application it is against the basic requirements for the protocol implementation. Long codes operate on message-level and this is too coarse; there is the need to demodulate and decode an individual bit so that a decision can be made whether the next priority bit should be transmitted; and spread spectrum modulation cannot be used on priority bits because it requires nodes that attempt to detect the priority bits be accurately synchronized with the sending nodes (there are many senders and they can all send a priority bit at approximately the same time, so a node trying to receive a priority bit cannot be synchronized with all of the senders).

Contrarily to the current state of the art in WSN radio devices, in order to achieve the desired medium activity during the contention resolution phase of the WiDom protocol, the use of OOK modulation is imperative. In addition, it is also crucial to, at the same time, have the smallest switching time possible between TX and RX modes, as well as the size of each bit should be as small as its reliable detection allows. According to the desired priority value, there is the need to transmit bits in such way that the transmission of the unmodulated carrier wave might be turned on or off several times during a contention for the medium, and carrier sensing has to be preformed every time a recessive bit occur. In

addition, the working principle of the protocol implies that a node must refrain from contention as soon as it realizes it has lost the tournament, hence, to implement WiDom, it is necessary to access the state (dominant or recessive) of each received bit immediately after it has been perceived. These constraints force that the radio devices (transmitter/receiver or transceiver) must have a low-level control that directly allows turning on and off the unmodulated carrier wave, and also that each received bit must be conveyed to the device output before receiving the next. This last characteristic also required control of timing granularity of the minimum bit size.

Currently available transceivers present a considerable switching time between receive (RX) and transmit (TX) modes when OOK modulation is used. Therefore, the problem of reducing the switching time was tackled by resorting to independent radio devices: one receiver and one transmitter. In this way, the architecture of the WiFLEX\_main board opposes common bi-directional radio modules design, that rely on a transceiver to perform both receive and transmit operations. Theoretically this (having two separate radio devices) would allow that the switching time from RX to TX, and *vice-versa*, to be instantaneous or negligible, although, as it will be further explained in Sub-section 4.1.1, such characteristic will not be so close to theory. To allow the use of only one antenna, both devices share a common one, using a HF switch.

Figure 18 presents a high-level outline of the WiFLEX\_main board architecture. Notice that the WiFLEX\_rxsync board is included. This board can be fitted in the WiFLEX\_main board, as a daughter board, to allow the reception of an out-of-band signal. The WiFLEX\_rxsync board has the necessary hardware to receive such signal.



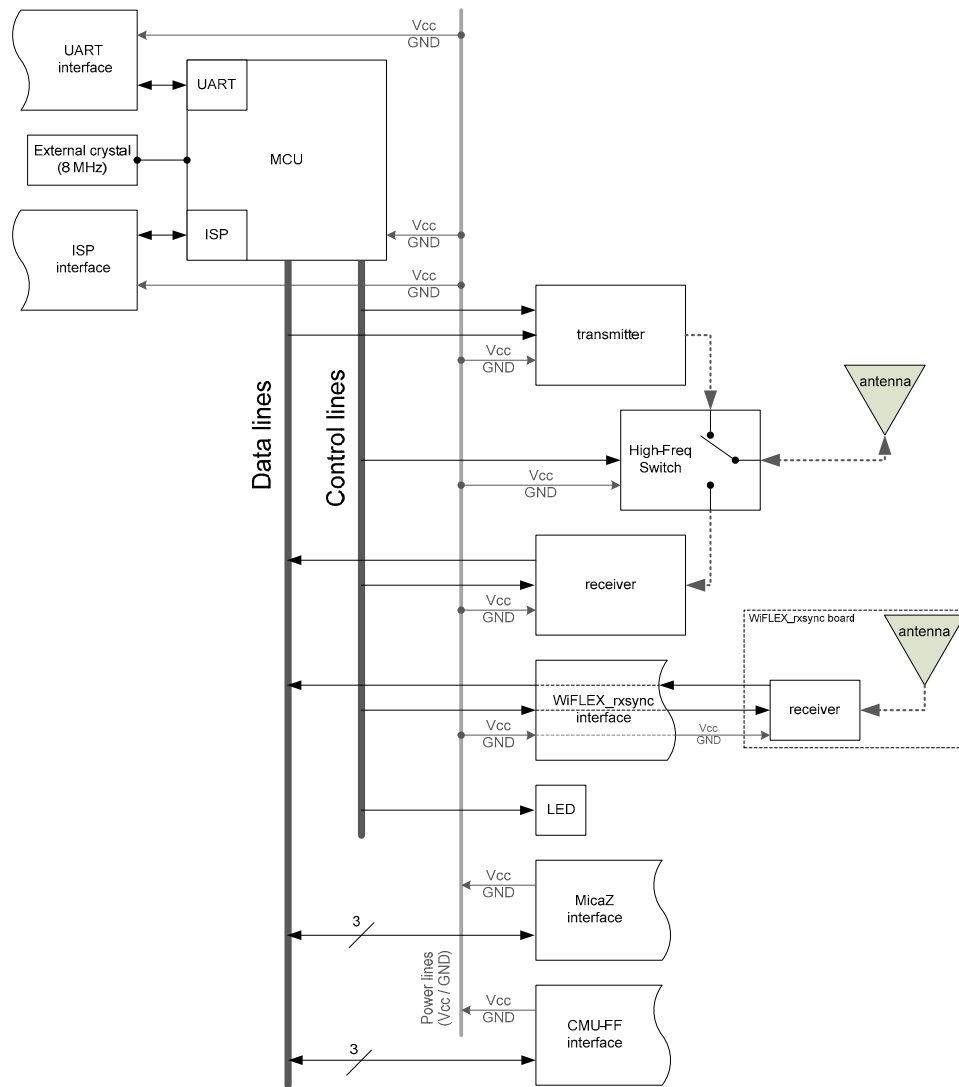
**Figure 18 Overview of the WiFLEX\_main board architecture, including WiFLEX\_rxsync board**

The board responsible for transmitting the out-of-band signal was named WiFLEX\_txsync board. Since its primary goal was to transmit a global synchronization signal, it was mainly composed by a MCU and a single radio transmitter working in the same band as the WiFLEX\_rxsync board.

The details of the referred hardware will be presented in the next sub-section, where the main effort will be the detailed description of the hardware architecture, followed by the main reasoning behind the chosen components, its dimensioning, and the way their physical connection takes places in each board.

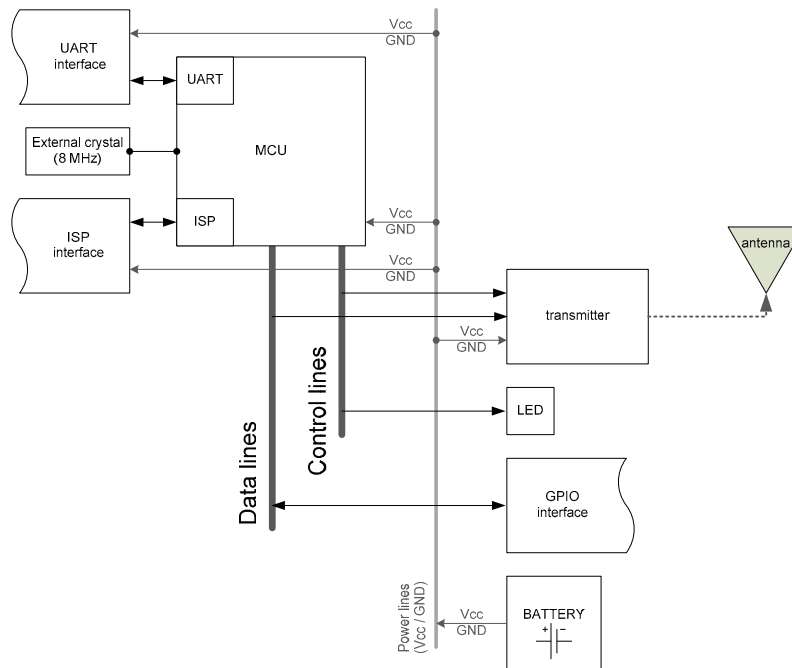
### **3.2. HARDWARE**

In Figure 19 is represented the detailed hardware architecture of the WiFLEX\_main board. Starting from the board's core, there is an MCU, which is responsible for controlling all the on-board hardware (through a group of GPIO pins that form a **data bus** and a **control bus**), and additionally provides communication capabilities for debugging (through an embedded UART) and connection to the host WSN platform. The same MCU is also provided with an embedded In-System Programming (ISP) interface, and an external clock generation circuit by means of a crystal. Figure 19 also depicts the main radio devices, *i.e.*, the independent transmitter and receiver, which are connected to the main antenna by the HF switch (responsible for controlling the antenna path). A LED was also introduced in the outline in order to allow fast and simple visual determination of the state of the program running in the MCU. The WiFLEX\_rxsync interface is simply provided by one data line and one control line connected to the respective buses (along with the necessary power lines). Although the board can only be fitted in one host WSN platform at a time, in order to implement the communication with the host WSN platform, namely the CMU-FF and the MICAz, were reserved six dedicated data lines. Three of them were reserved for the CMU-FF and the other three for the MICAz.



**Figure 19 WiFLEX\_main board hardware architecture, including WiFLEX\_rxsync board**

As it can be observed in Figure 20, the WiFLEX\_txsync board has a much simpler architecture than the previous one. For sake of simplicity of design (hardware and software) and cost, the main core of the board is the same as the one of the WiFLEX\_main board. An MCU was set to deal with hardware control, which has embedded an UART and ISP interface. The same external clock circuitry is used, as well as the LED. The main differences arise in the control of only one transmitter, the presence of three GPIO (that may be necessary for future developments such as synchronization between several WiFLEX\_txsync boards) and the need for a dedicated power source (batteries) due to the fact that this board is not coupled to another host WSN platform.



**Figure 20 WiFLEX\_txsync board hardware architecture**

Along the next sections will be discussed several key issues related with the hardware design. More important than the choice of a specific component, the main characteristics for selecting components capable of bringing advantages in the implementation of a physical layer for wireless dominance protocols will be outlined. Nevertheless, the fundamental rules for hardware design will be introduced along with the discussion of the hardware design, especially the ones related with the components placement and routing in double layer PCB.

### 3.2.1. GENERAL PCB DESIGN CONSIDERATIONS

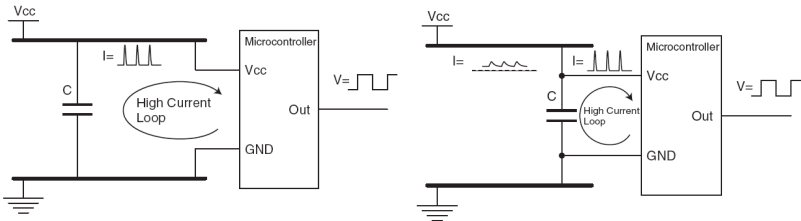
Designing circuits where components operate at high frequencies is a challenging task. Joining high-speed digital integrated circuits (such as an MCU) along with HF analogue and mixed-signal components (like radio devices and HF switches), and getting a good performance of the overall system can only be accomplished if a wide group of disciplines (in the area of Electronics Engineering and Physics), are correctly understood and their concepts put in practice. The foremost aim of this sub-section is to highlight some of the major guidelines, taken into account during the design of each PCB that was developed. These guidelines influenced the components selection and oriented its accommodation.



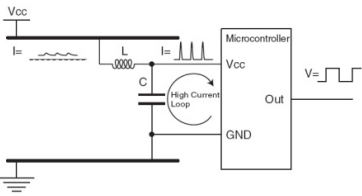
Such guidelines can be found in several literature related with the development of Electromagnetic Compatibility (EMC) compliant devices. Without going through exhaustive details, and omitting any mathematical or physic derivation, the following list summarizes the most important aspects referred [6], [8], [28], [47], [50], [52] and [61]:

- Path to ground: one of the most important tools to achieve good EMC performance is to provide a controlled path to ground for all signals, ensuring that such path is kept away from signals and circuits that may be disturbed. In this way, the transmitted noise will find a short path to ground, as well as the received noise, before reaching sensitive components in the circuit.
- Board sections: establishment of physical separation between digital and analogue (and mixed signal) components decrease the mutual noise interference, and confine its higher values to specific zones. Hence, creating dedicated zones in the PCB allows decomposing the overall system design into a group of smaller and less complicate problems, along with the implementation of simple dedicated filters for each section.
- RF immunity and sealing: PCB tracks acts likes antennas in the board picking up environment noise as well as delivering it out of the board. Once inside the system, the noise can be coupled into more sensitive signal lines. The introduction of low-pass filters (LPF) in the most critical lines prevents HF signals to corrupt its values. Inductors and ferrite beads are well suited to be introduced in series, due to its high impedance levels at HF while having low impedance at low-frequencies (LF). Conversely, decoupling capacitors are good to shorten HF signals to ground.
- Decoupling capacitors: the capacitors used for decoupling, rather than having a high capacitance, should have a low equivalent series resistance (ESR) to effectively respond to transient energy draws from the power line. Putting more than one similar capacitor in parallel, to achieve the desired capacitance, is usually a good technique to reduce the ESR. For example, although a MCU may only consume a few mA according to its absolute ratings, one must know that those values are medium values; the current spikes drained by digital circuits can be several hundred of mA on the clock edges (specially if the GPIO are supplying other components), reason why a proper decoupling should be implemented to avoid introducing such noise in the power line. The positioning of the decoupling capacitor is crucial to obtain the desired results; Figure 21 depicts the effects of supplying a MCU with a wrong positioned decoupling capacitor. Once placed far from the target component, the high current loop greatly affects the power supply lines;

hence these components must always be placed as close as possible to the envisioned component, confining the high current loop to a short path. As depicted in Figure 22, additional filtering can be obtained resorting in an inductor, although this must present a very low equivalent resistance. Although both figures refer examples where a MCU is the target component, all analogue or mixed signal components that either require such disrupting current demands or gets influenced by its side effects, should have such filtering.



**Figure 21 Current spikes in a wrong decoupling capacitor placement (left) and correctly decoupling capacitor placement (right) [6]**

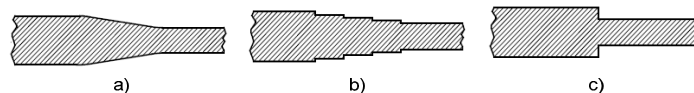


**Figure 22 Additional filtering using an inductor [6]**

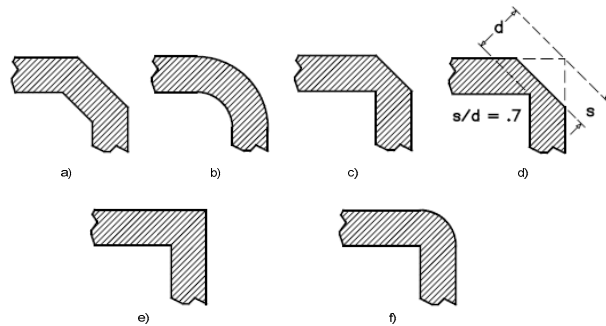
- Routing and ground plane: current always flows in loops, from the higher voltage point to the lowest one, and through the lowest impedance path. A current flow, either from power supply or signal, emits noise, which increases directly with the loop length, current and frequency. In this sense, the PCB routing must always provide the lowest impedance through the shortest path, and every noisy path must have a low impedance path to ground so the noise can be captured. This can be achieved by surrounding such noisy paths by low impedance ground paths, and this is better addressed resorting in well designed ground planes. Nevertheless, one must remember that at HF the better path for the current to flow may not be the copper trace on the PCB but the PCB material itself. Designing a good ground plane, *i.e.* with a low impedance, in a double layer board, and at same time cover all noisy paths and zones of the board, may be achieved if a dedicated ground plane is set in the opposite layer of the components and

routing, along with well distributed group of low impedance vias connecting it to the top layer remaining cooper, that is also connected to the system ground. Vias spacing must be dimensioned accordingly to the expected noise frequency and deployed generously to reduce its inherent parasitic inductance and capacitance; usually their spacing must be less than  $1/20^{\text{th}}$  the wavelength ( $\lambda$ ) of the radio frequency (RF) signals to obtain good performance. When the PCB is divided in sections to avoid interference between, for example, digital and analogue signals, the traces must be confined to the respective section, i.e., traces must not be routed into adjoining sections.

- PCB traces: the shape and dimensions of PCB traces for signals with frequencies above 100 MHz have a crucial impact on the overall trace impedance, frequently resulting in impedance mismatching and consequent signal reflections, attenuation and undesired propagation in the path from the source to the destination component (for example, from the radio transmitter output to the antenna). Traces shorter than  $1/20^{\text{th}}$  the wavelength ( $\lambda$ ) of the radio frequency (RF) signals usually do not need impedance matching, although special attention to traces shape must be taken into account. Due to the vias' parasitic impedance and capacitance, they must never be used to route any RF or high-speed digital signals. In Figure 23 are depicted three typical configurations of transitions between different trace widths or component pad connections; while the trace in Figure 23a (smooth transition) gets the best impedance matching between the wider and the narrowest section of the trace, the one in Figure 23c (single-step transition) gets the poorest performance, and hence must be avoided; the trace represented in Figure 23b (multi-step transition) is a fair option when the one in Figure 23a cannot be used. The PCB trace cornering shape must also be taken into account to avoid the referred effects. The best corner shape to be adopted is depicted in Figure 24d, where a ratio of  $s/d = 0.7$  brings the best coupling between the first and the second section of the trace; although, for sake of simplicity, the shapes depicted in Figure 24a, b and c are good alternatives; the corner shapes in Figure 24e and f are not suitable for a good impedance matching.



**Figure 23 Transitions between different trace widths: a) smooth transition (best); b) multi-step transition (acceptable); c) single-step transition (avoid)**



**Figure 24 Corner shapes: a), b) and c) simple yet well performing corner shapes; d) best corner shape; e) and f) bad performance corner shape (avoid)**

- Components placement: due to effects of the radiated energy from components, especially inductors, capacitors, resistors and integrated circuits (IC), special care must be taken into account considering relative positioning towards neighbour components. In the first place, unless otherwise is specified, the external components of an IC must be placed as close as possible to its respective pin in order to avoid radiation from connecting PCB traces and impedance mismatching, especially in what concerns decoupling and filtering components, which is also valid for the case of connectors. Secondly, due to every component radiated magnetic field, parallel positioning must be avoided when a minimum distance of, at least, one times their minimum height cannot be left between them; a 90 or 180 degrees positioning is always preferred, especially when placing inductors.
- Components size and materials: two of the main factors that influence the received and transmitted noise are the components size and manufacturing materials. Bigger components are more susceptible to work as antennas, and certain materials, at HF, work as a path for signals, even though they are good insulators at LF. In what concerns size, the choice of using smaller components is preferential. Components suitable to work at HF usually employ materials like ceramics and derived compositions.

### 3.2.2. RADIO DEVICES

The hardware architecture described in the beginning of Sub-section 3.1 identifies the need for four different radio devices that together form two transmitter-receiver pairs working at different frequencies.

RF Monolithics has components capable of satisfying the previously outlined constraints in a wide range of frequencies. LF components offer the advantages of simplicity inherent to

the PCB design, components positioning, impedance matching, etc. On the other hand, HF components offer the advantages of using small antennas, large bandwidth and higher bit rates. Since this was a project dedicated exclusively to R&D purposes, LF components were selected which led to the selection of the pairs TX5001-RX5001 and TX5002-RX5002, respectively working in the 315 MHz and 418 MHz bands.

The choice of which pair should be used in WiFLEX\_main board, and in the WiFLEX\_txsync and WiFLEX\_rxsync boards, was based on the basic principles derived from the *Friis transmission equation* as follows. Equation (5) represents the referred equation for an ideal radio propagation scenario [32][33]:

$$P_r^o = P_t G_t G_r \left( \frac{\lambda}{4\pi d} \right)^2 \quad (5)$$

Were:

$P_t$  = transmitting antenna power

$P_r^o$  = receiving antenna power

$G_t$  = transmitting antenna gain

$G_r$  = receiving antenna gain

$d$  = distance between transmitter and receiver antennas

$\lambda$  = wavelenght

Equation (5) is valid considering a non-obstructed free space and multipath-free propagation, proper antenna alignment and polarization and perfect antenna matching. Assuming that all the terms in the right side of Equation (5) are fixed except  $\lambda$ , it can be derived that  $P_r^o$  is proportional to  $\lambda^2$ , hence larger carrier wave wavelengths result in higher values of receiving power at the receiver's antenna. Knowing that frequency is inversely proportional to the wavelength, LF are preferred when bigger ranges are to be achieved for a fixed transmission power. With this knowledge in mind, and knowing that it is instrumental that the out-of-band signalling covers all the broadcast domains of the nodes in the network, the TX5001 and RX5001, both operating in the 315 MHz band, were the ones fitted, respectively, in the WiFLEX\_txsync board and WiFLEX\_rxsync board. The TX5002 and RX5002, operating at 418 MHz, were thus assigned to the WiFLEX\_main board.

### 3.2.2.1. RECEIVING DEVICES DIGITAL OUTPUT

As previously referred, the raw digital output of the receiving radio devices (RX5001 and RX5002) is extremely important for the purpose of implementing the WiDom protocol.

Although, a closer analysis of its functionality is crucial for its correct interpretation *via* the MCU. Ideally, in order to allow the use of an interrupt driven application, it was desired that digital output pins of the referred radio devices would respond univocally to the detection of a carrier wave, but due to the variations of the environment noise, as well as the signal-noise ratio (SNR), such implementation is not possible. The TXDATA pin of the RX5001 and RX5002 radio devices, even in the absence of any transmitted carrier in its operating band (respectively 315 MHz and 418 MHz) respond as depicted in Figure 83 of Appendix C. In this sense, it is necessary to perform additional signal processing, at the MCU, to distinguish the transmitted signals from noise. This may be achieved through a synchronization mechanism as explained in Sub-section 3.3.

Nevertheless, during stable operation, the radio devices digital output work as expected in the presence of a transmitted carrier, i.e., as depicted in Figure 84 of Appendix C, the detection of a carrier wave is signalled by a high-level of the RXDATA pin, and the non-detection by a low-level.

### 3.2.3. MICROCONTROLLER

The key factors that led to the selection of the Atmel's AVR® MCU ATmega168V were related with the following characteristics [9][10]:

- Low-power consumption: active mode: 250  $\mu$ A at 1 MHz, 1.8 V; power-down mode: 0.1  $\mu$ A at 1.8 V
- Low-voltage operation: 1.8 – 5.5 V
- PCB footprint size: 9x9 mm in 32 pad Micro Lead Frame (MLF) package
- In-System Programming (ISP) interface
- Number of I/O pins: 23 programmable I/O lines
- Memory size: 16 kB FLASH + 512 B SRAM + 1 kB SRAM
- Timers: two 8-bit timer/counter + one 16-bit timer/counter
- Real-time counter with separate oscillator
- Internal calibrated oscillator at 8 MHz
- Up to 1 MIP throughput per 1 MHz
- External and internal interrupt sources
- UART

It was also taken in account that, if a future hardware version requires less CPU capacity than the current one, in order to lower bill of materials costs, a downgrade is possible to ATMEGA88V or ATMEGA48V with full pin compatibility.

In what concerned the MCU clock speed, considering a minimum interval between carrier wave pulses of 30  $\mu$ s, an MCU running at 8 MHz provides a minimum of 240 instructions. This throughput is sufficient for running the necessary protocol instructions between two consecutive priority bits.

#### 3.2.4. HF SWITCH

The key factors in the selection of the HF switch, responsible for commuting the antenna path between the receiving and transmitting devices, were related with the following parameters:

- Low-voltage and low power operation
- High off isolation at the working frequency
- Low insertion loss at the working frequency
- High isolation between paths (*i.e.* low crosstalk)
- Fast switching time
- Path selection through digital input
- Single-Pole Double-through (SPDT) configuration

The Analog Devices ADG918 is a digitally controlled SPDT HF switch capable of satisfying the necessary requirements since, accordingly with [3], its main characteristics are:

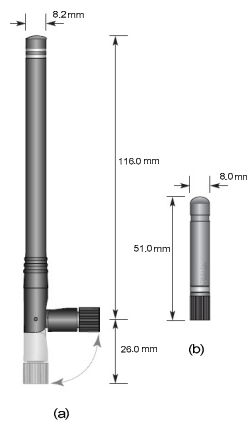
- Voltage operation: 1.65 V to 2.75 V
- Power consumption: 1  $\mu$ A at 2.75 V
- Off isolation: 49 dB at 500 MHz
- Insertion loss: 0.5 dB at 500 MHz
- Isolation between paths: 44dB at 500 MHz
- Switching time: less than 30 ns

This component is provided in a packaging compliant with the desired overall size of the WiFLEX\_main board, since it presents itself in a eight leads Mini Small Outline Package (MSOP) of 3x3 mm.

### 3.2.5. MISCELLANEOUS HARDWARE

The selection of the remaining used components followed the next reasoning:

- Quartz crystal: was selected to provide an external clock source for the MCU, and intended to provide a background solution in case the internal oscillator of the MCU showed a significant clock drift or a lack of precision that compromised the WiDom protocol implementation. The chosen quartz crystal, ABMM2@8 MHz from Abracon Corporation, presents a maximum frequency tolerance of  $\pm 20$  ppm (at 25 °C), low power consumption (at most 500  $\mu$ W), requiring a load capacitance of 18 pF and presenting small size package [1].
- RP-SMA (reverse polarity – standard male adaptor) antenna connector: the selected antenna connector allows to adapted different types of antennas [39]. In this way is possible to use antennas with different gains and evaluate its impact on the system performance.
- Antennas: due to its reduced size, in a first approach the antennas selected to radiate the 418 MHz band carrier waves were  $\frac{1}{4}$  wave length compact (Figure 25(b)). Further on, for the sake of a longer range,  $\frac{1}{4}$  wave length full-size type were also tested (Figure 25(a)). Although the global transmission board is provided with a  $\frac{1}{4}$  wave full-size antenna, the daughter boards, dedicated to receive the global synchronization signal, are  $\frac{1}{4}$  wave compact ones.
- Connectors: in order to reduce the parasitic impedance associated to the connector's fittings, the connectors used in the WiFLEX\_main board to the WiFLEX\_rxsync are gold-plated. The same option was made to the ones providing connection to the host WSN platform.



**Figure 25 Antennas: (a)  $\frac{1}{4}$  wave full-size antenna; (b)  $\frac{1}{4}$  wave compact antenna [39]**



### 3.2.6. WIFLEX\_MAIN BOARD DESIGN

Due to several constrains, namely the ones related with the number of controlled hardware, interfaces provided, architecture and also overall size, the WIFLEX\_main board design was the most demanding in comparison with the other two boards (WIFLEX\_rxsync and WIFLEX\_txsync). The following sub-sections provide the necessary information to understand the rationale behind such design and outlines the most important hardware characteristics of the WIFLEX\_main board.

#### 3.2.6.1. DIMENSIONING TRANSMISSION CIRCUITRY

The external components necessary for all the radio devices used (RF Monolithics TX5001, RX5001, TX5002 and RX5002) were dimensioned resorting to the application note “ASH DESIGNER’S ASSISTANT – v2.5.5”, provided by RF Monolithics [46]. This application note requires, among others, the definition of the desired modulation type and the data encoding parameters, and computes the required external components.

Taking into account the requirements previously referred, after selecting the radio device (TX5002), the application input parameters were configured envisaging OOK modulation, custom pulse width comprised within a minimum of 30  $\mu$ s and a maximum 40  $\mu$ s, maximum transmit power, and the voltage applied to “TXMOD” pin (pin 8) of 3 V with a standard circuitry. Such configuration resulted in the circuit configuration depicted in Figure 26. The component characteristics are presented in Table 7 of Appendix D, along with a brief description of its function.

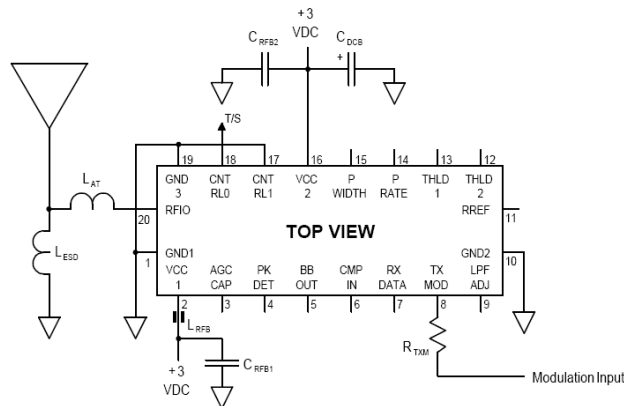
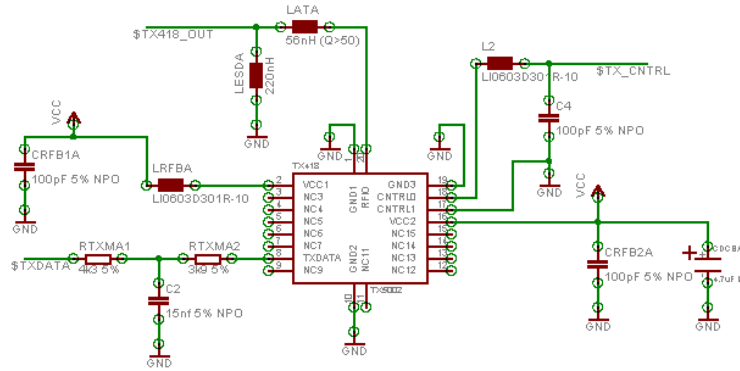


Figure 26 OOK transmitter with minimum components configuration [56]



**Figure 27 Robust TX5002 circuitry configuration**

Additionally, in order to reduce the transmitter sidebands outside of the band or sub-band of operation, the circuitry at the input of pin 8 (TXMOD) was changed. A simple RC LPF was introduced resorting on the decomposition of  $R_{TXM}$  in two resistors,  $R_{TXM1}$  and  $R_{TXM2}$ , of (respectively) 4.3 k $\Omega$  and 3.9 k $\Omega$ . In order to assure that the tolerance inherent to the value of the resistor will not seriously affect the final desired value ( $R_{TXM} = R_{TXM1} + R_{TXM2}$ )  $R_{TXM1}$  and  $R_{TXM2}$  tolerance was reduced to 0.5 %. The capacitive component, of the RC LPF, was obtained through a 15 nF NPO capacitor.

Furthermore, to suppress any digital interference between the microcontroller pin that will control the operating mode (sleep mode or TX in OOK mode) and the “Modulation & Bias Control” of the radio device [56], a bypass capacitor ( $C_4$ ) and an RF decoupling coil was placed at pin 17 of the TX5002 (CNTL1).

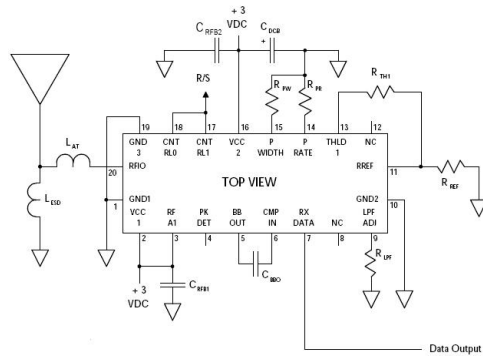
These modifications to the basic configuration circuitry result in the schematic present in Figure 27. Notice that every component reference concerning the TX5002 configuration is followed by the letter “A”.

### 3.2.6.2. DIMENSIONING RECEIVER CIRCUITRY

Again, with the help of [46], were dimensioned the external components necessary for RX5002 operation. This time, after selecting the radio device, RX5002, the application input parameters were configured envisaging OOK modulation, custom pulse width comprised within a minimum of 30  $\mu$ s and a maximum 40  $\mu$ s and the first data slicer<sup>3</sup>

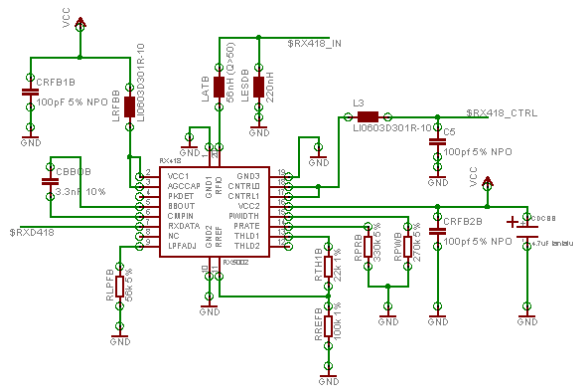
<sup>3</sup> The *data slicers* are capacitor-coupled comparators responsible for converting the analogue signal from BBOUT to a digital stream. The threshold value of the first data slicer (DS1) is set by the resistor  $R_{TH1}$ . The threshold value of the

threshold of 20 mV. The second data slicer was disabled and, due to the fact that the automatic gain control is not necessary when operating in OOK [54], it was also disabled. The pulse width/rate was set to standard, and the RSSI and RXDATA circuitry were also disabled. The configuration results in the basic configuration depicted in Figure 28. The components used in the referred configuration are presented in Table 8 of Appendix D, along with a brief description of its function.



**Figure 28 OOK receiver with minimum components configuration [54]**

Despite the resultant configuration, additional measures were taken in order to increase the circuit robustness to digital noise. Similarly to what happened in the TX5002 configuration, the control lines were bypassed with a 100 pF capacitor ( $C_5$ ) and decoupled through the use of an RF coil. The positive supply voltage pin for the receiver baseband circuitry (pin 2 – VCC1) also had the same protection. These modifications to the basic configuration circuitry result in the schematic present in Figure 29.



**Figure 29 Robust RX5002 circuitry configuration**

second data slicer (DS2) is set analogously by the resistor  $R_{TH2}$ . Omitting  $R_{TH1}$  and  $R_{TH2}$  disables, respectively, DS1 and DS2 operation [52].

### 3.2.6.3. MCU, INTERFACES AND MISCELLANEOUS CIRCUITRY

In order to control all the onboard hardware, as well as to run the WiDom protocol, and provide a communication interface with the host WSN platform, a MCU was fitted and provided with the necessary connections. Figure 30 depicts the MCU output pins connections, along with its reset, external oscillator and LED circuitry. The green lines represent wires or connections. Those appearing discontinued have a label referring to the wire's name, which, in a similar way, appear at the respective connected component's or connector's pad.

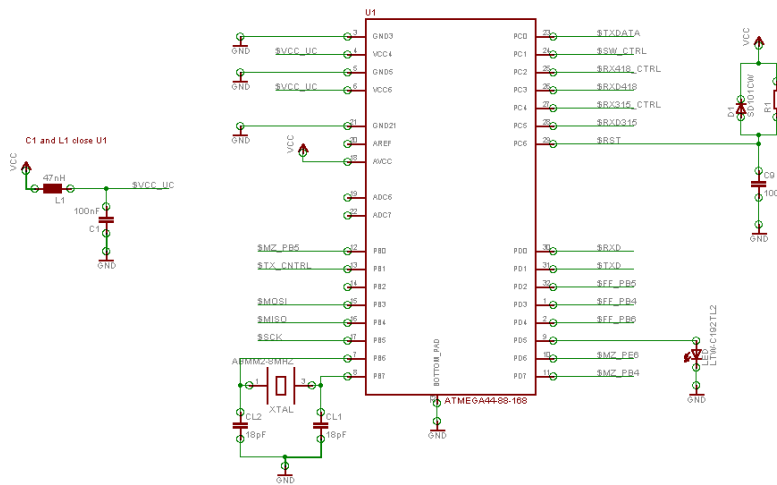


Figure 30 MCU connections, reset, external oscillator and LED circuitry

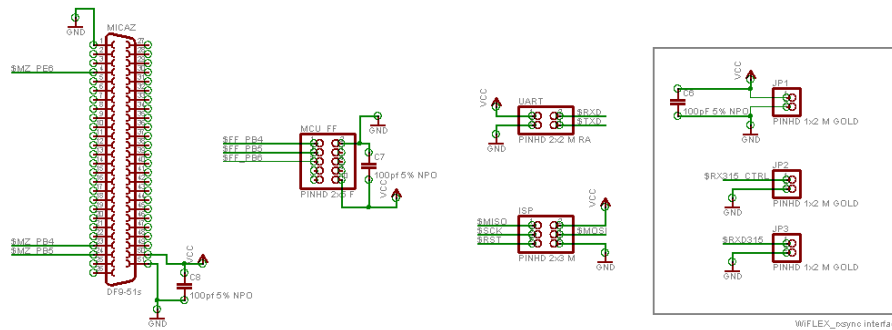
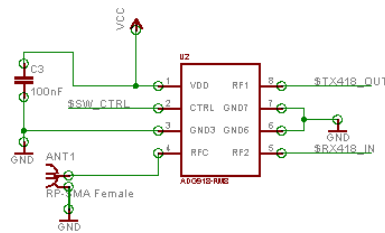


Figure 31 ISP, UART and interfacing connectors (for CMU-FF, MICAz and WiFLEX\_rxsync)

Figure 31 depicts the schematic of the connectors present in the WiFLEX\_main board. JP1, JP2 and JP3 are meant to be connected the WiFLEX\_rxsync board, providing power and simultaneously control and data lines (respectively RX315\_CTRL and RXD315). The UART connector allows connecting the MCU pins dedicated for UART communications as well as power lines (VCC and GND) for interfacing a TTL-RS232 transceiver. The six

pins connector is a standard ISP connector, for programming purposes, to be used with ATMEL MCU programming devices. The target WSN platforms, MICAz and CMU-FF, have different type of connectors, hence two different types of connectors were fitted in the WiFLEX\_main board. A standard MICAz's 51 pin connector (named as MICAz in the referred schematic) provides the necessary connections for the WiFLEX\_main platform to communicate with the MICAz, as well as for accessing its power source. In a similar way, the 10 pin GPIO connector of the CMU-FF (Header 4) is used for interfacing and power accessing; the label CMU\_FF refers to the former WiFLEX\_main connector.

The configuration of the HF switch, used to control the antenna path, is the one depicted in Figure 32. In order to provide a SPDT configuration from the point of view of the antenna, the RF1 and RF2, pins 8 and 5, where respectively connected to TX5002 output and RX5002 input, while the antenna connector was assigned to the RFC (pin 4). The HF switch control line, SW\_CTRL, connected to the CTRL (pin 2) is connected to the MCU's first pin of port C (PC1 - pin 24).



**Figure 32 HF switch configuration**

#### 3.2.6.4. COMPONENTS PLACEMENT, DETAILS AND OUTLINE

The main board can be divided in three main sections: the digital, analogue and mixed-signal. The digital section refers to all the components in which only digital signals are conveyed. In opposition, the analogue section refers to the ones where only analogue signals are present. Since both sections must interact, some components are naturally mixed-signal. These sections do not include any components that are related with energy distribution and filtering, those are not fitted in any particular section. The MCU unit and its reset circuitry, interfaces with the host WSN, digital interface with the WiFLEX\_rxsync board, UART, programming interface, external clock generation circuitry and signalling LED are included in the digital section. Conversely the antenna and its connector are solely analogue. The radio devices and its fundamental external components, along with the HF switch, figure in the mixed-signal section. Figure 33 depicts the referred sections.

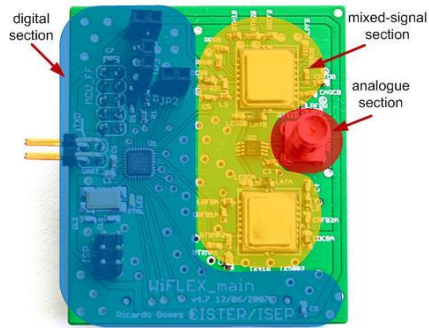


Figure 33 Main board sections

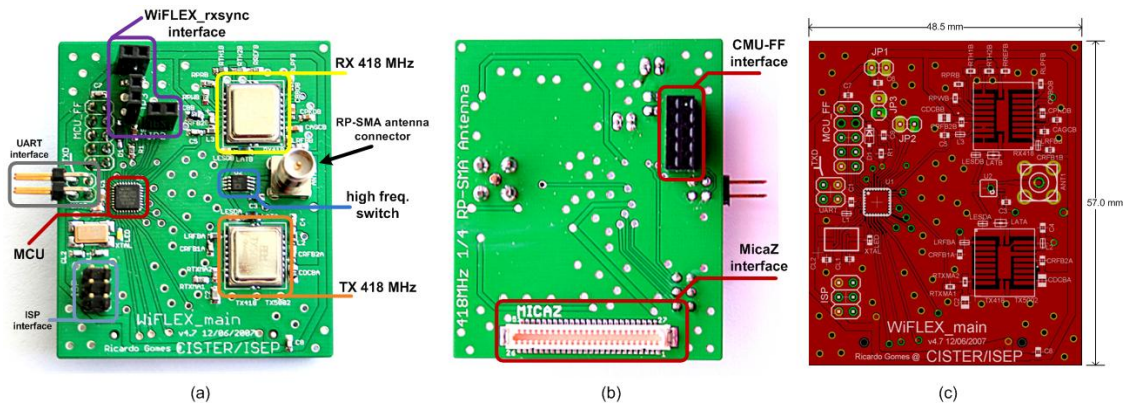
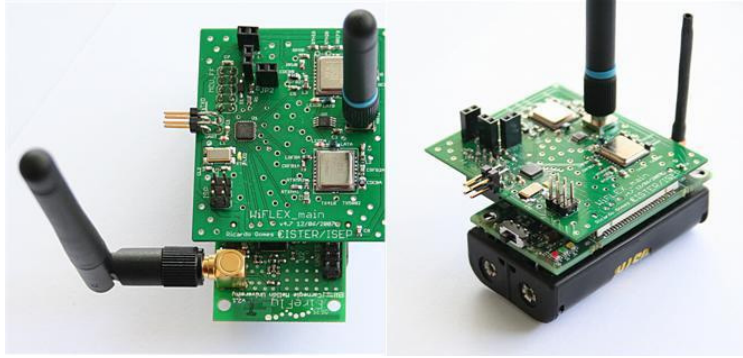


Figure 34 Components distribution over top (a) and bottom (b) layers of the WiFLEX\_main board; (c) PCB dimensions

In accordance with the guidelines described over Sub-section 3.2.1, the primary effort of the components distribution in the PCB should be dedicated to create a physical separation between the digital section and the analogue and mixed-signal sections. In this way, the digital noise stays confined to a limited PCB region, making difficult for its propagation to the analogue section due to decoupling (through filtering), and lower impedance to the ground path. Furthermore, the external components dedicated to each integrated component were placed as close as possible to its interconnection pin, always ensuring that all capacitors and inductors, that referred to the same pin, were positioned forming a  $180^\circ$  or  $90^\circ$  angle.



**Figure 35** WiFLEX\_main board fitted in the CMU-FF (left) and in the MICAz (right)

The accommodation of the most important components and interfaces over the PCB is presented in Figure 34(a) and (b). In these figures can be observed the high density of vias deployed between the digital and mixed-signal and analogue sections in order to reduce the interference between these sections. The integrity of the bottom GND plane can also be observed. Overall dimensions of the PCB are presented in Figure 34(c). The final PCB thickness is 0.8 mm.

The WiFLEX\_rxsync interface was designed in a way that such board could not be fitted in the WiFLEX\_main board in a wrong position. The UART interface was placed in 90 degrees towards the PCB in order to reduce the interference of the TTL-RS232 transceiver board and allow the simultaneous fitting of the ISP programmer. The interfaces for CMU-FF and MICAz WSN platforms, visible in Figure 34(b), allow the WiFLEX\_main board to fit in the target WSN platforms as depicted in Figure 35.

### 3.2.7. WIFLEX\_RXSYNC BOARD DESIGN

The WiFLEX\_rxsync board is an extension of the main board that allows the reception of a global synchronization signal in a different band from the one used for the medium contention. In this sense, the board only has a receiving device, along with its external components, an antenna connector and the necessary connectors for the interface with the WiFLEX\_main board. Although the hardware present in this board could be fitted in the WiFLEX\_main board, the opinion of designing it as an add-on board was motivated by the fact that the use of the out-of-band synchronization is not needed for the implementation of all the WiDom protocol versions; in this sense unnecessary energy consumption, electromagnetic interference, as well as size and overall cost can be reduced for the implementations that do not rely on the out-of-band synchronization.

### 3.2.7.1. DIMENSIONING RECEIVER CIRCUITRY

The external components required by the radio receiver were dimensioned resorting to [46], however, a different pulse width configuration was selected from the one used for the WiFLEX\_main board. Since the pulse width of the synchronization signal does not influence the protocol overhead, and reliability may be extended at the cost of larger pulse duration [51], the maximum pulse width was configured to be 60  $\mu$ s. The first data slicer threshold was set to 10 mV, and the second data slicer was enabled and configured for 60 mV threshold. The results of the computation performed by [46] are summarized throughout Table 9 of Appendix D.

Similar measures to the ones taken to achieve higher robustness in the RX5002 were considered. In this sense the resultant circuitry was the one depicted in Figure 36.

Since the WiFLEX\_rxsync board must only interface with the WiFLEX\_main board, it has three simple connectors that provide connection for power (VCC and GND), control and data lines; its schematic is depicted in Figure 37.

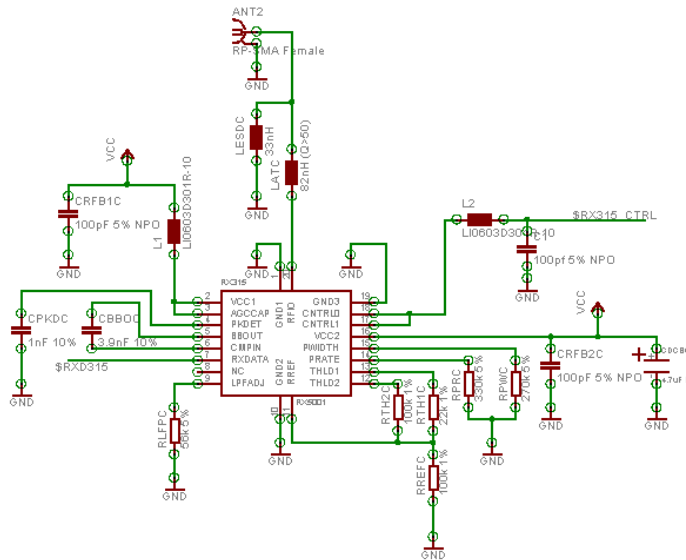


Figure 36 Robust RX5001 circuitry configuration

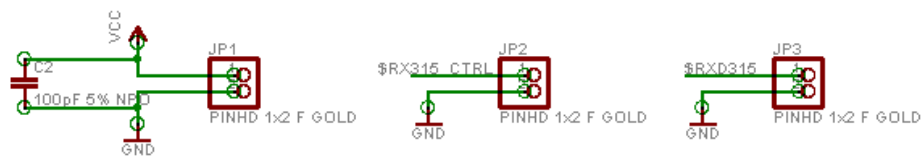


Figure 37 WiFLEX\_rxsync interface connectors



### 3.2.7.2. COMPONENTS PLACEMENT, DETAILS AND OUTLINE

Due to the simplicity of the involved hardware, despite the guidelines outlined in Sub-section 3.2.1, no special techniques were used in the PCB design. Figure 38(a) and (b) depicts both top and bottom planes of the WiFLEX\_rxsync. On the top layer were deployed the antenna connector, and the RX5001 radio with its required external components. The bottom layer in turn was only fitted with the connectors necessary for the interface with the WiFLEX\_main board; its positioning avoids wrong connections made by users of the board. As depicted in Figure 38(c), the PCB dimensions are 35x36 mm, and a final thickness of 0.8 mm was demanded at production time.

The WiFLEX\_rxsync board is fitted in the WiFLEX\_main board as depicted in Figure 39. The antenna connectors opposing positions were meant to reduce the mutual attenuation caused by both 315 MHz and 418 MHz antennas. The stack design ensures that the WiFLEX\_main board can be fitted in any of both target host WSN platforms.

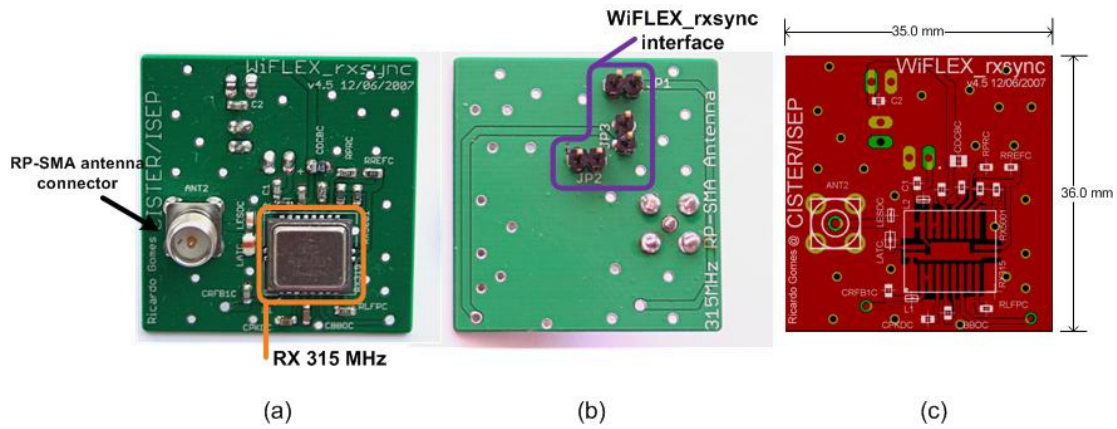


Figure 38 Components distribution over top (a) and bottom (b) layers of the WiFLEX\_rxsync board; (c) PCB dimensions

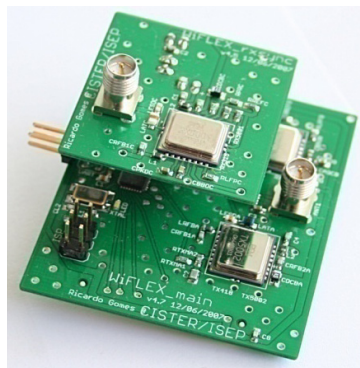


Figure 39 WiFLEX\_rxsync board fitting in the WiFLEX\_main board



and in order to download programs to the MCU an ISP programming interface was fitted. For extended debug and communications capabilities an interface to the MCU's UART was also introduced. This led to the circuitry depicted in Figure 41 and Figure 42; the former presents the MCU connections, reset, external oscillator, LED and GPIO interface; while the latter depicts the referred connectors.

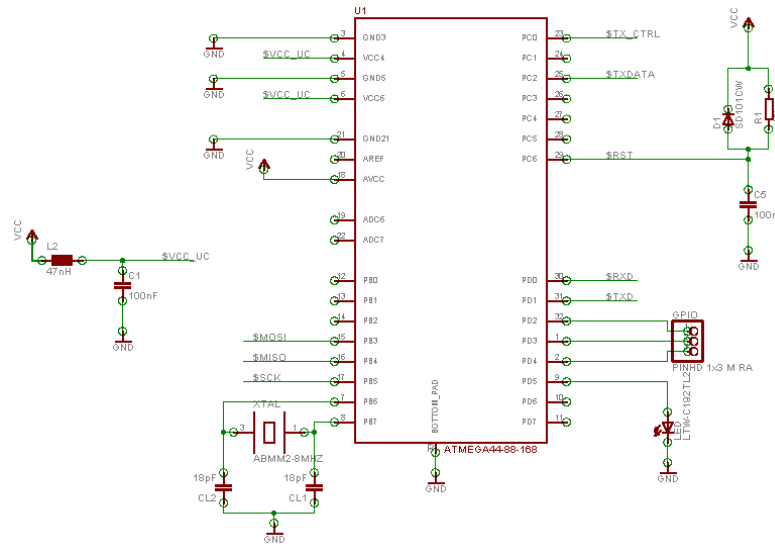


Figure 41 MCU connections, reset, external oscillator, GPIO and LED circuitry

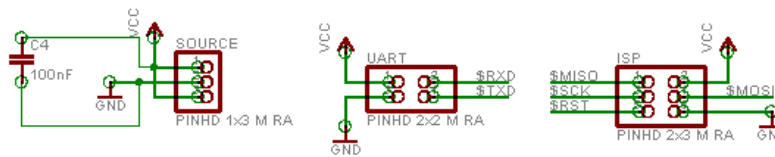


Figure 42 Power, ISP and UART connectors

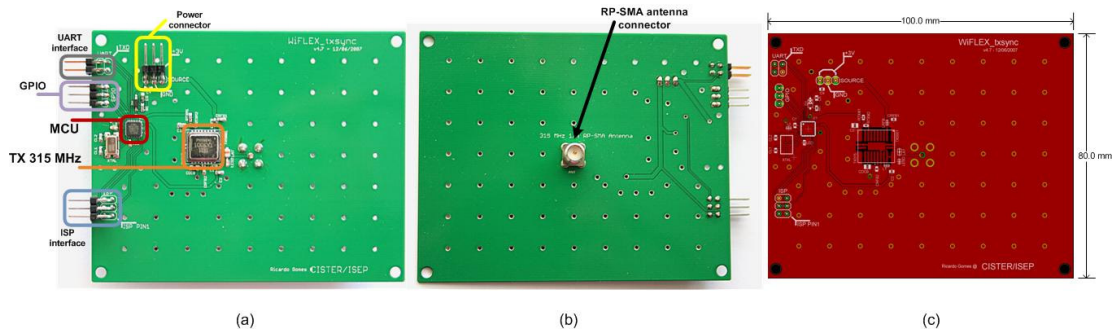
### 3.2.8.3. COMPONENTS PLACEMENT, DETAILS AND OUTLINE

Since size was not a critical constraint, the PCB dimensions were extended to provide a bigger ground plane area. In this way, it was expected that the transmission range was extended, in comparison with the main board.

In what concerns the components distribution over the board, the same guidelines referred for the WiFLEX\_main board are applicable, meaning that a sectioning of the board in accordance with the type of components involved (digital, mixed-signal and analogue) was implemented. Nevertheless, the summarized guidelines presented in Sub-section 3.2.1 were also taken in account. Figure 43(a) and (b) depicts the top and bottom layer of the

referred PCB. The antenna connector was placed on the PCB's bottom side, since in this way it was possible to achieve a better performing ground plane through a more uniform copper distribution. The less populated PCB area, along with absence in size restriction, allowed components and routing to better distributed. The vias deployment was also possible to be distributed over the PCB in a grid fashion.

Figure 43(c) depicts the physical dimensions of the PCB. Its final thickness was fixed to be 0.8 mm.



**Figure 43 Components distribution over top (a) and bottom (b) layers of the WiFLEX\_txsync board; (c) PCB dimensions**

### 3.3. COMMUNICATION DETAILS

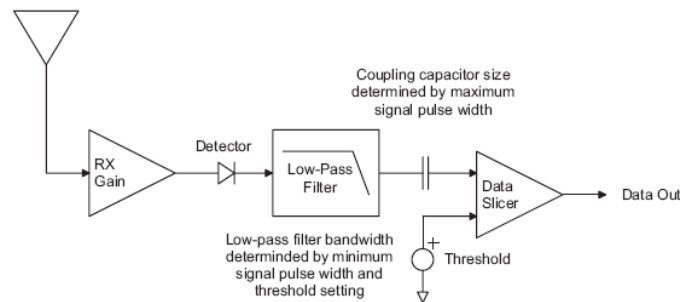
In order to understand the software implementation, some details around communication aspects concerning the used hardware must be explained. The reasons why using bit-stuffing and preamble was imperative are explained throughout the next sub-sections, as well as the capture effect phenomenon is briefly explained along with the presentation of a simple solution to reduce its impact.

#### 3.3.1. BIT-STUFFING

The receiver devices present in the WiFLEX\_main and WiFLEX\_rxsync boards use amplifier-sequenced hybrid (ASH) technology. As depicted in Figure 44, such receivers are AC-coupled between its receiver base-band output and the subsequent comparator input, hence the data bit stream received should be properly encoded for good DC-balance, otherwise, when the medium is idle, or in the presence of a long carrier wave, the referred receivers become unreliable due to poor DC-balance [52].

Bit encoding is greatly exploited in data transmissions in order to, among other factors, maximize data transmission reliability and maintain channel activity for synchronization purposes. Generally this is done by maintaining a proper DC-balance. The performance of a radio system depends on how well the data encoding scheme conditions the signal for AC-coupling. Such scheme should achieve DC-balance, *i.e.*, the encoded signal has a “1” value for 50 % of the time and a “0” value for the remaining 50 %, yielding therefore no DC component [62]. The bit encoding scheme also limits for how many bit periods the encoded signal remains at a “1” or a “0”, *i.e.* the maximum pulse (or gap) width that can occur in a transmitted signal [52]. Figure 45 depicts the former bit encoding characteristic.

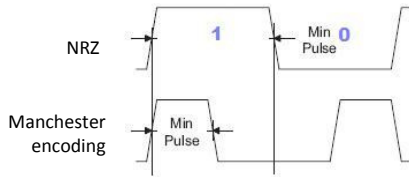
In general, it is assumed that only one sender will be transmitting a given data during a certain time, which means that network nodes do not need to change between RX and TX modes as frequently as for each bit duration. In common applications, several encoding techniques could be used to assure proper DC-balance at the receiver. Although, due to the non-negligible switching time between receive and transmit modes, a suitable bit encoding for our application is not trivial.



**Figure 44 Receiver signal processing [52]**



**Figure 45 Minimum and maximum pulse width in a non return to zero (NRZ) bit encoding [52]**



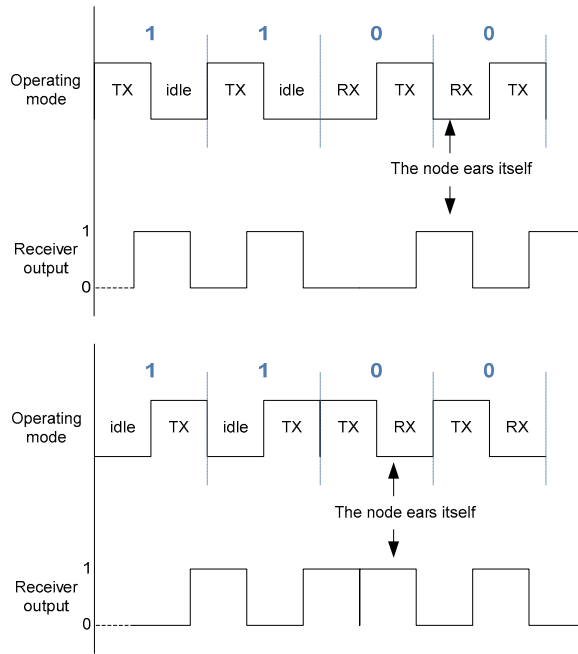
**Figure 46 Manchester bit encoding [52]**

Manchester bit encoding is typically a good bit encoding technique to assure a correct DC-balance and does not require many resources (memory and/or CPU) to be implemented. Figure 46 compares Manchester bit encoding with a non-encoded bit transmission scheme. The efficiency of Manchester bit encoding, in providing good DC-balance, relies on the fact that a “1” bit is composed by a “1” + “0” signal, and a “0” bit is composed by a “0” + “1” signal [52], [62]. In this way, despite the number of consecutive “1” or “0” bits that have to be transmitted, a corrected DC-balance is always assured.

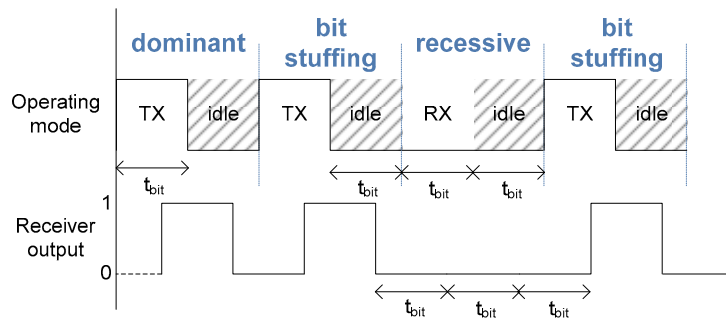
In a first approach such scheme might seem to work for WiDom implementation, but this approach fails due to the necessary switching time between TX and RX modes. Attend on Figure 47. Let us assume that the propagation delay is negligible in the range of a typical WSN. Due to non-idealities, the transmitter’s oscillator takes a given amount of time before the effective transmission of a carrier wave takes place; additionally, at the receiver, another given time elapses before its digital output reflects the data sent by the transmitter (more details in Sub-section 4.1.1). The effect of the sum of these non-idealities imposes a non-negligible switching time, hence the Manchester bit encoding is not possible to be used.

According to [52], the consecutive occurrence of “1” or “0” pulses should not be higher than four in order to achieve a satisfactory DC-balance.

Alternatively to the use of a bit encoding technique, a bit stuffing technique can be exploited. One of principles behind the use of bit stuffing techniques is to introduce redundant information to maintain channel activity [14]. In this sense, and considering the constraints inherent to a satisfactory DC-balance achievement, a stuff bit must be introduced after each information bit. Figure 48 demonstrates this reasoning. Considering that the idle times can be used for switching between TX to RX mode, and *vice-versa*, the self-hearing problem does not occur, hence the switching time is correctly accommodated.



**Figure 47 Manchester encoding (top) and reverse Manchester encoding (bottom)**



**Figure 48 Bit stuffing approach**

By coding a dominant bit through a “1” + “0” signal sequence, and a recessive bit by a sequence of two “0”, the introduction of a stuff bit composed by a “1” + “0”, signals after each dominant or recessive bit only introduces a sequence of at most three consecutive equal signals, more precisely three consecutive “0”. Although not perfect, the achieved DC-balance is satisfactory for a correct AC-coupling, hence allowing correct data recovery at the receiver.

Due to the fact the switching from reception to transmission mode can occur in a negligible time, the idle periods that follow the reception period in the recessive bits can be used to accommodate the guard time due to non perfect synchronization.

### 3.3.2. PREAMBLE

As summarized in Sub-section 3.2.2.1, the digital output of the receiving devices does not offer a conditioned and treated signal, meaning that such data contains several errors due to the, for example, variations of the SNR, which commonly results in the presence of false carrier wave detections. Additionally, since the communication is not always maintained, the synchronization clock for sampling the output data of the receiver is constantly lost. For that reason a sampling synchronization mechanism must be implemented.

The work presented in [45] summarizes an appropriate procedure to correctly synchronize the RFM's RX5001 and RX5002 radio devices, based on the transmission of a *special start symbol* composed by sequence of three consecutive zeros, followed by eight consecutive ones and another zero (00011111110); in other terms a transmitter should first maintain no carrier transmission during the time of three symbols, followed by a carrier transmission during the time of eight symbols and then refrain from transmitting during another symbol duration. After the last symbol the receiver would be properly stabilized and synchronized for accurately receive incoming data. Although, this mechanism is intended for point-to-point, or point-to-multipoint transmissions, and the overhead introduced is considerably high (twelve times symbol duration).

In order to overcome this requirement a different preamble was used. Due to the medium activity introduced by the use of the previously explained bit-stuffing technique, a preamble composed by two consecutive dominant bits was sufficient to allow the correct distinguish between noise and an incoming transmission start, limiting the introduced overhead to four times the symbol duration.

### 3.3.3. CAPTURE EFFECT

The *capture effect* is a phenomenon associated to variations of relative amplitude of two (or more) signals, causing one to dominate over the other (or others) at a given receiver, suddenly displacing the first at the demodulated output [14]. In this implementation this effect revealed to be problematic as a node would refrain to detect a carrier transmission from a node far way due to the previous detection of a carrier transmission of a node nearby, hence compromising the receiving node from refraining from the tournament and/or correctly perceiving the winners priority. This shortcoming was even more critical knowing that the radio devices suffer from problems related with self-detection and carrier

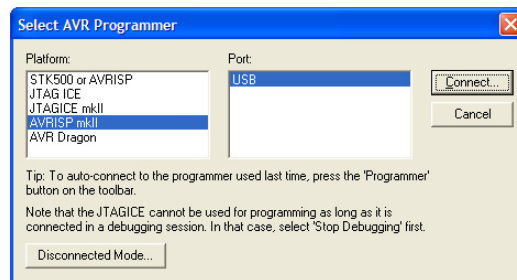


detection delay. To overcome such problem, as explained in Sub-section 4.1.1, a silence time was introduced after every carrier wave transmission.

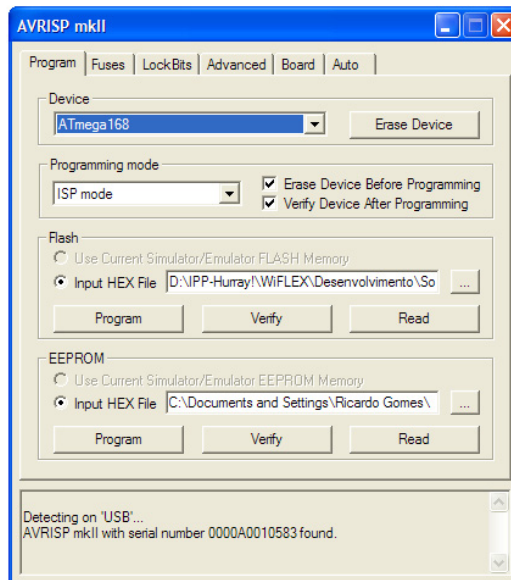
### 3.4. SOFTWARE

In order to develop the necessary code for the target MCU, the Atmel ATmega168V, was used the Atmel's AVR Studio® v4.13 integrated development environment (IDE). This IDE allows the writing, either in C language or Assembly, and debugging the developed applications for AVR® devices. In order to program the referred target MUC is necessary an ISP enabled programmer. The one used during this project was the Atmel's AVRISP mkII In-System Programmer, which in combination with the AVR Studio® enables the programming of the AVR 8-bit RISC (reduced instruction set computer) architecture MCU.

Setting up together the referred IDE and ISP programming device requires the installation of the IDE in a Windows® 9x/NT/2000/XP environment, along with the installation of the required USB drivers for the ISP programming unit. At start up, the IDE application automatically detects the programming unit, which should be selected in the sub-menu *Tools>Program AVR>Connect*. Figure 49 depicts a possible example of the referred sub-menu.



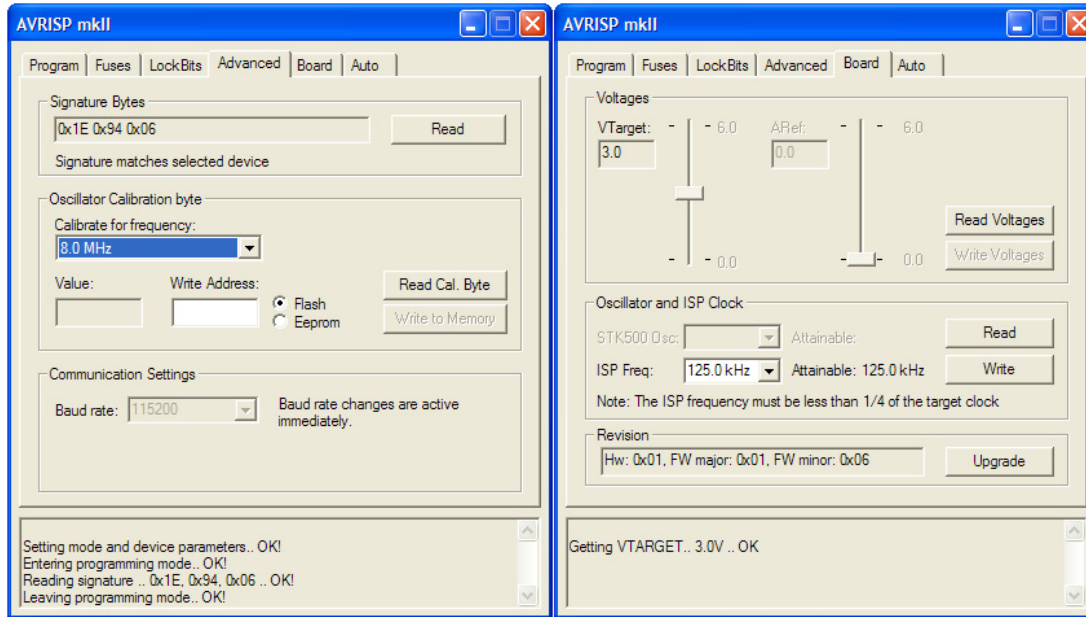
**Figure 49 AVR programmer selection menu**



**Figure 50 AVRISP mkII programmer configuration menu**

Next to this setup, the user will be prompted the *AVRISP mkII* configuration menu, as depicted in Figure 50. The target device must then be selected in the *Device* scroll down box, as well the *Programming* mode must be selected. The compiled Flash and/or EEPROM source files are selected through, respectively, *Flash* and *EEPROM* sections. The *Fuses* and *LockBits* tabs are used to configure several MCU parameters required at the program download (e.g. clock speed and source, watchdog timer, brown-out detection level, application and boot loader protection, etc...). The *Auto* tab is dedicated to the definition of a group automatic programming sequence of procedures in order to easy the programming of several identical target MCU.

The *Advanced* tab (Figure 51(a)) is dedicated to confirmation of the matching between the selected target device, and the one to which to the ISP programmer is connected to. Additional calibration of the MCU's internal RC oscillator can also be preformed through this tab in the *Oscillator Calibration Byte* section. Figure 51(b) depicts the *Board* tab. This tab is used for verifying the status of the target device voltage (*Voltages* section), defining the ISP clock frequency (*Oscillator and ISP Clock* section; notice that the ISP frequency must always be less than  $\frac{1}{4}$  of the target clock, since this may lead to target device programming failure and/or permanent damage) and also for updating the ISP programmer firmware (*Revision* section).



(a)

(b)

**Figure 51 Advance and Board settings tabs**

Figure 52 depicts an overview of the AVR Studio V4 IDE. In the top of the window are located a customizable set of quick access bars. In centre of the main window is visible the sub-window where the code writing takes place, while at the left side is located the project tree (*AVR GCC*) where the several files “.c” and “.h” files can be selected for presentation in the code writing sub-window. At the right side is visible the *I/O View* sub-window; this is used for debugging and simulation purposes, allowing the evaluation of the MCU’s peripherals state at pre-runtime (when simulating) and at runtime (when debugging). The lower located sub-window provides project build related messages.

Due to the similarity of both *WiFLEX\_main* and *WiFLEX\_txsync* board’s core, the software project was created and managed as one. In this sense, prior to code compilation, was defined the target board (*WiFLEX\_main* or *WiFLEX\_txsync*) and then downloaded the respective code.

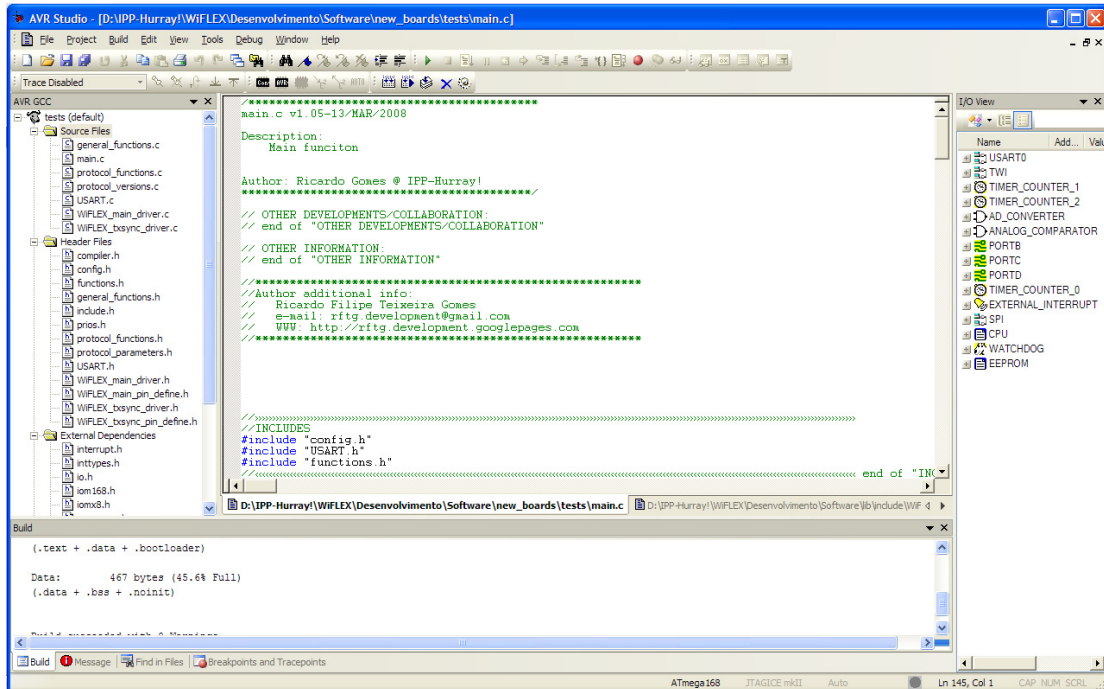


Figure 52 AVR Studio v4 IDE

The software developed for the MCU fitted in the `WiFLEX_main` and `WiFLEX_txsync` boards can be divided in three different layers:

- hardware mapping: mapping of the MCU GPIO pins accordingly to its respective hardware connection with the on-board hardware
- drivers: group of functions capable of abstracting the application programmer from the hardware operation details; MCU operation definitions
- application: protocol implementation and protocol parameters definitions; communication with the host WSN platform.

Along the following sub-sections will be described the general aspects of the up referred program layers.

### 3.4.1. LOW LEVEL FUNCTIONS COMMON TO `WiFLEX_MAIN` AND `WiFLEX_TXSYNC` BOARDS

As previously referred, using the same MCU in both `WiFLEX_main` and `WiFLEX_txsync` boards brings several advantages in what concerns the code development. This property allowed exploiting the development of a group of functions that can be used independently from the target board.

Table 3 **Raw GPIO control macros**

Prototype	Description
<code>gpio_direction( _port, _pin, _pin_direction )</code>	Allows to define a specific GPIO pin direction; usage: port, pin and direction (PD_INPUT or PD_OUTPUT for direction); ex: <code>GPIO_DIRECTION( B, GPIO1, PD_INPUT)</code>
<code>gpio_set( _port, _pin )</code>	Sets a GPIO pin to high level; usage: port and pin; ex: <code>gpio_set( B, GPIO1 )</code>
<code>gpio_clr( _port, _pin )</code>	Sets a GPIO pin to low level; usage: port and pin; ex: <code>gpio_clr( B, GPIO1 )</code>
<code>gpio_get( _port, _pin )</code>	Returns a GPIO pin value; usage: port and pin; ex: <code>gpio_get( B, GPIO1 )</code>
<code>gpio_toggle( _port, _pin )</code>	Toggles a GPIO pin actual state; usage: port and pin; ex: <code>gpio_toggle( B, GPIO1)</code>

Those that were intended for raw GPIO control were implemented in a single file, named “general\_functions.h”, as macros. Table 3 presents the prototype and a summary description of such functions.

### 3.4.2. **HARDWARE MAPPING**

Taking in account the schematic of the MCU interconnection with the on-board hardware of the WiFLEX\_main board (recall Figure 30 in Sub-section 3.2.6.3), the GPIO pins mapping was implemented accordingly to Table 11 of Appendix E.

Due to the use of the same MCU, the WiFLEX\_txsync board figured a similar MCU’s GPIO mapping. Table 12 of Appendix E presents the resultant mapping from the schematic depicted in Figure 41, outlined in Sub-section 3.2.8.2.

Both of these hardware mapping tables were implemented in a “.h” file included in the software project, respectively “WiFLEX\_main\_pin\_define.h” and “WiFLEX\_txsync\_pin\_define.h”, has a constant definition, meaning that, for example “MOSI” represents the definition of the constant decimal number “2”. The following code extract outlines an example:

```
(...)  
#define MOSI           3 // PB.3  
#define MISO           4 // PB.4  
#define SCK            5 // PB.5  
(...)
```

This sort of abstraction was crucial to develop the remaining software, in the sense that the hardware was clearly referred to its function acronym, instead of a meaningless decimal number.

### 3.4.3. DRIVERS

In order to operate some of the on-board hardware (like the radio devices) specific procedures must be carried out to guarantee its correct behaviour, which is independent of the protocol version that is implemented. Additionally, routines like setting the system to start transmission involve the set up of different components and running different procedures. These requirements led to the development of a low-level layer (drivers) that abstracted the application programmer from such detailed and specific operations. These drivers were independently developed for each board.

The `WiFLEX_main` and `WiFLEX_txsync` boards drivers were implemented in a “.c” file included in the software project, respectively named as “`WiFLEX_main_driver.c`” and “`WiFLEX_txsync_driver.c`”, having their function’s prototype definitions present in a “.h” with the same name.

Along the following sub-sections will be described the most important functions that compose the referred drivers.

#### 3.4.3.1. **WiFLEX\_MAIN BOARD DRIVERS**

The several hardware present in the `WiFLEX_main` board is controlled resorting on the drivers described in Table 4.

Table 4 **WiFLEX\_main drivers**

Prototype	Description
<code>inline void antenna_to_tx(void)</code>	Drives the antenna path to the transmitting device
<code>inline void antenna_to_rx(void)</code>	Drives the antenna path to the receiving device
<code>inline void TX418_active(void)</code> <sup>4</sup>	Sets the 418 MHz transmitter (TX5002) into <u>active</u> mode; the antenna path is set to the transmitting device. <b>Pseudo code:</b> TX418_DATA = 0; antenna_to_tx(); pause 2 ms; TX418_CNTRL = 1; pause 25 ms END
<code>inline void TX418_sleep(void)</code>	Sets the 418 MHz transmitter (TX5002) into <u>sleep</u> mode; the antenna path is left unchanged. <b>Pseudo code:</b> TX418_DATA = 0; TX418_CNTRL = 0; pause 25 ms END
<code>inline void RX418_active(void)</code> <sup>4</sup>	Sets the 418 MHz receiver (RX5002) into <u>active</u> mode; the antenna path is set to the receiving device during wake-up from sleep time, and then is set to the transmitting device. <b>Pseudo code:</b> antenna_to_tx(); pause 2 ms; RX418_CNTRL = 1; pause 25 ms; antenna_to_tx() END
<code>inline void RX418_sleep(void)</code>	Sets the 418 MHz receiver (RX5002) into <u>sleep</u> mode; the antenna path is left unchanged. <b>Pseudo code:</b> RX418_CNTRL = 0; pause 25 ms END
<code>inline void RX315_active(void)</code> <sup>4</sup>	Sets the 315 MHz receiver (RX5001) into <u>active</u> mode <b>Pseudo code:</b> RX315_CNTRL = 1; pause 25 ms END
<code>inline void RX315_sleep(void)</code>	Sets the 315 MHz receiver (RX5001) into <u>sleep</u> mode. <b>Pseudo code:</b> RX315_CNTRL = 0; pause 25 ms END
<code>inline void main_LED_on(void)</code>	Turns the LED on
<code>inline void main_LED_off(void)</code>	Turns the LED off
<code>inline void main_LED_toggle(void)</code>	Toggles the state of the LED
<code>inline void carrier418_on(void)</code>	Turns <u>on</u> the transmission of the 418 MHz carrier wave
<code>inline void carrier418_off(void)</code>	Turns <u>off</u> the transmission of the 418 MHz carrier wave
<code>inline bool carrier418_sense(void)</code>	Returns the present value of the 418 MHz receiver (RX5002) output; “1” if carrier is being received, “0” if no carrier is being received

---

<sup>4</sup> The radios initialization procedure follows the recommendations present in [53], [54] and [56], which resumes that the radio devices take at most 15 ms to become operational and stabilized. To ensure that incorrect stabilization do not occur this value was extended to 25 ms

### 3.4.3.2. WIFLEX\_TXSYNC BOARD DRIVERS

The WIFLEX\_txsync board drivers were developed in a similar way then the ones developed for the WIFLEX\_main board, although they are reduce to the ones presented in Table 5.

Table 5 WIFLEX\_txsync drives

Prototype	Description
inline void TX315_active(void) <sup>5</sup>	Sets the 315 MHz transmitter (TX5001) into <u>active</u> mode; the antenna path is set to the transmitting device. <b>Pseudo code:</b> TX315_DATA = 0; antenna_to_tx(); pause 2 ms; TX315_CNTRL = 1; pause 25 ms END
inline void TX315_sleep(void)	Sets the 315 MHz transmitter (TX5001) into <u>sleep</u> mode; the antenna path is left unchanged. <b>Pseudo code:</b> TX315_DATA = 0; TX315_CNTRL = 0; pause 25 ms END
inline void sync_LED_on(void)	Turns the LED on
inline void sync_LED_off(void)	Turns the LED off
inline void sync_LED_toggle(void)	Toggles the state of the LED
inline void carrier315_on(void)	Turns <u>on</u> the transmission of the 315 MHz carrier wave
inline void carrier315_off(void)	Turns <u>off</u> the transmission of the 315 MHz carrier wave

### 3.4.3.3. BOARD INITIALIZATION

The WIFLEX\_main and WIFLEX\_txsync boards initialization must accomplish a series of steps to correctly initialize the MCU and all on-board hardware. To start with, it is necessary to initialize the MCU GPIO ports accordingly to each pin direction – input pin or output pin. After this step the UART can be initialized, as well as the radio devices.

Each radio device (respectively TX5002, RX5002 and RX5001, and TX5001) is activated, resorting in the respective functions, separated by intervals of 50 ms between themselves. This procedure can be carried out through the call of the function *void init\_WiFLEX\_main\_board(void)*, for the WIFLEX\_main board, and the function *void init\_WiFLEX\_txsync\_board(void)*, for the WIFLEX\_txsync board.

---

<sup>5</sup> The radio initialization procedure follows the recommendations present in [55], which resumes that the radio device take at most 15 ms to become operational and stabilized. To ensure that incorrect stabilization do not occur this value was extended to 25 ms

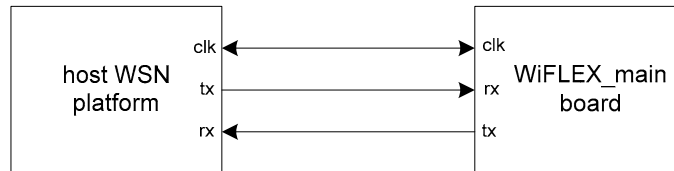


### 3.4.4. APPLICATION

The application layer is divided into two different sections. One concerns the communication mechanism developed to communicate with the host WSN platform, enabling the reception of a queued priority value; the other concerns the implementation of the WiDom protocol, in which also is included a protocol definition and configuration file, defined prior to compilation.

#### 3.4.4.1. COMMUNICATION WITH THE HOST WSN PLATFORM

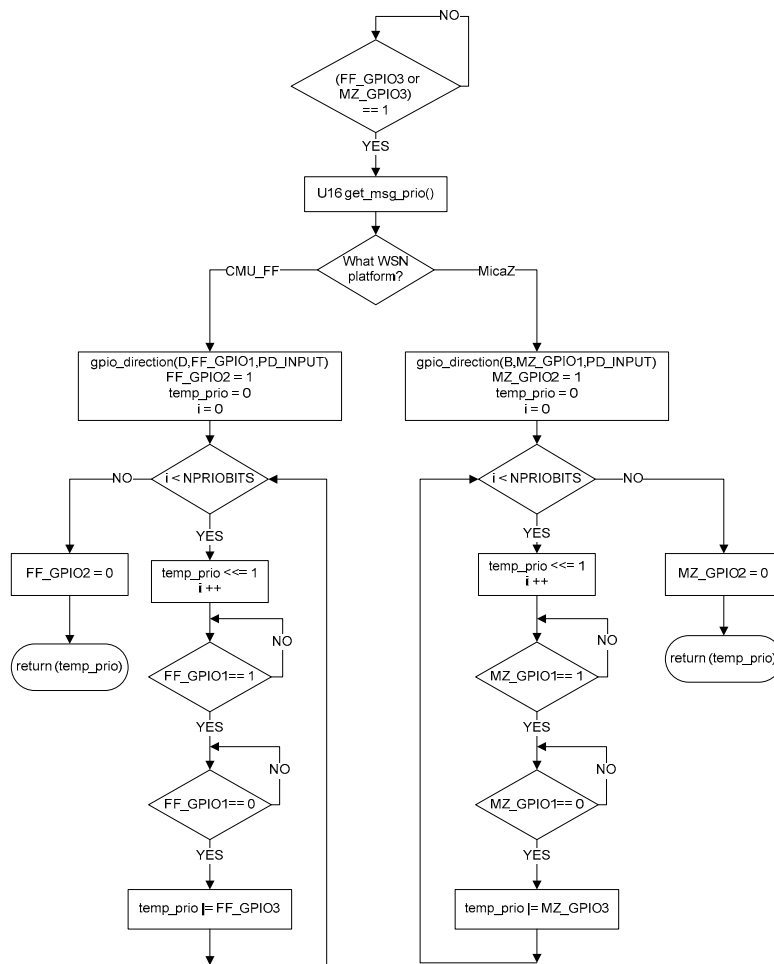
The concurrent co-processor based architecture implemented, requires that the WiFLEX\_main board is able to communicate with its host WSN platform, namely the MICAz and the CMU-FF. This was achieved through the use of three dedicated data lines that are connected to the host WSN platform by the interface connector. The communication protocol that was implemented resorts to a custom master-master synchronous unicast transmission protocol, where a common clock line is shared between both masters and two data lines are used to convey data (receive and transmit lines). The master that transmits data uses its *clk* line to impose a time reference for the receiving master to know when the data bits should be sampled. Those data bits are shifted to a unidirectional output data line, the *tx* line, that is connected to the input data line, the *rx* line, of the receiving master. When the masters shift their role, the *clk* line direction is inverted. Figure 53 depicts the connection scheme previously explained.



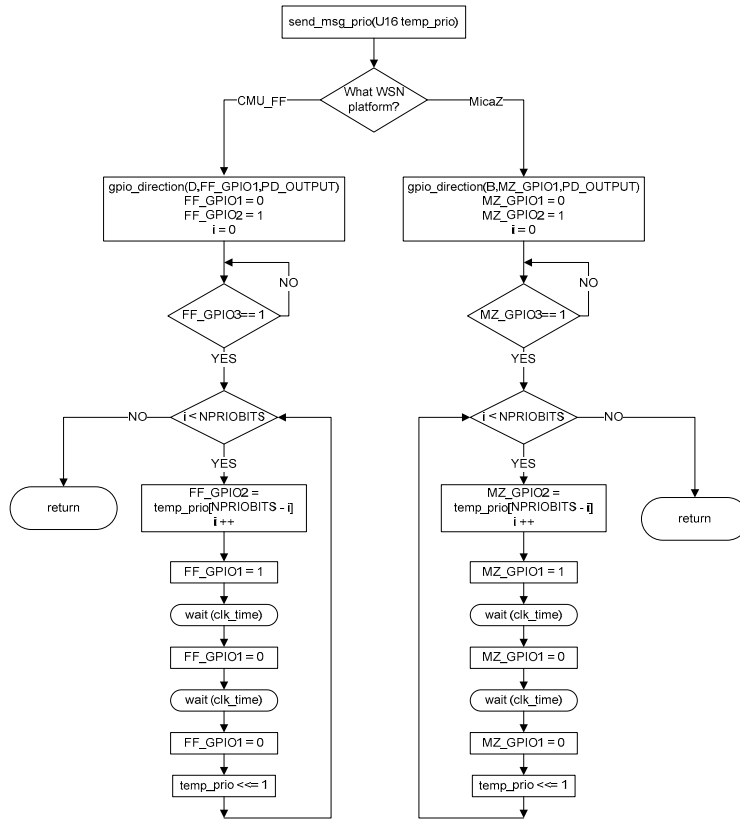
**Figure 53 Communication between host WSN platform and WiFLEX\_main board**

Since the master that first communicates is always known (it is always the host WSN platform the first to transmit the priority that will be used in the tournament phase) even when the node does not actively participate in the tournament, no concurrency resolution scheme is needed. Let us assume that, after the board's boot-up, its first state is waiting for receiving a message priority value to be used in the tournament phase. This state corresponds to the start point of the algorithm depicted in Figure 54. Taking in account the point of view of the WiFLEX\_main board, consider that the *clk* line corresponds to FF\_GPIO1 and MZ\_GPIO1; the *rx* line corresponds to the FF\_GPIO3 and MZ\_GPIO3;

and the *tx* line corresponds to the FF\_GPIO2 and MZ\_GPIO2. At that referred starting point, the MCU is waiting to detect the signalling that the host WSN platform will start the transmission of a message priority value, i.e., the *rx* line in the WiFLEX\_main board to become equal to “1”. When this event occurs it is called the function *U16 get\_msg\_prio()*, which is responsible for receiving the message priority value that will be used in the tournament. In the first place the *clk* line is set as input line, to which follows the signalling that the reception is ready by driving the *tx* line of the WiFLEX\_main board high. Bits are received, starting from the most significant one, with the *rx* line being sampled at the falling edge of the clock. The procedure is repeated, until NPRIOBITS are received. After that, the function returns the message priority value and the program, running in the WiFLEX\_main board MCU, continues its flow.



**Figure 54 Algorithm of the communication protocol with the host WSN platform for data receiving**



**Figure 55 Algorithm of the communication protocol with the host WSN platform for data transmission**

After the tournament phase ends, every node must access if it has won or lost the contention for the medium. In fact all nodes know the priority of the winning node. Because of these two reasons it is necessary that the WiFLEX\_main board conveys such information for the host WSN host platform, and that is performed in a similar way to the data reception, although this time the host WSN platform acts like the receiver.

In the WiFLEX\_main board, after the tournament phase end, the function *send\_msg\_prio(U16 temp\_prio)* is called. In the first place it will now set the *clk* line as a output data line, after what it will signal the host WSN platform that it is ready to perform the transmission through driving its *tx* line high. Conversely to what happened in the previous scenario, the WiFLEX\_main board will now wait for the host WSN platform to signal that it is ready to start receiving the priority value of the winning message. After that moment, bits will start being transmitted from the most significant to the least significant ones (let  $temp\_prio[n]$  denote the  $n^{th}$  bit of the variable *temp\_prio*), at the rate imposed by the falling edge of the *clk* line. The state of the node after the tournament, *i.e.* winner or

loser, is not conveyed in the transmission, it is instead determined by the host WSN platform after it as finished receiving the priority value. Figure 55 depicts the described algorithm.

#### 3.4.4.2. PROTOCOL IMPLEMENTATION

The implemented version of the WiDom protocol was meant for a single broadcast domain and used an external out-of-band synchronization source. Nevertheless, at cost of a higher overhead, the original version of the WiDom protocol, *i.e.* without external out-of-band synchronization (as outlined in Sub-section 2.3.3), can be implemented with minor changes by adapting the implemented synchronization phase.

For sake of simplicity, the following code examples omit several functions and call-outs regarding the energy management of the WiFLEX\_main platform, such as sleep periods and its procedures and conditions.

The protocol implementation relies on the cyclic execution of the function `inline void WiDOM_SBD_MN_ofb`. The following code extract, depicted in Figure 56, outlines the implementation for the MICAz has host platform.

```
(1) inline void WiDOM_SBD_MN_ofb (void)
(2) {
(3)     U8 i=0;
(4)     U16 received_prio = 0;
...
(5)     do
(6)     {
...
//waits for WSN platform to signal priority transmission
(7)         while (gpio_get (D, MZ_GPIO3) == 0);
(8)         //receive priority
(9)         received_prio = get_msg_prio();
(10)        winner = 1;
...
//waits for out-of-band synchronization signal to start the tournament
(11)        while (get_out_of_band_sync() != 0);
(12)        TCNT1 = 0;           //resets TCNT1
(13)        TCCR1B = 1;        //Turns TCNT1 on without prescaling
//jumps to tournament phase
(14)        winner_prio = tournament_phase(received_prio, &winner);
//reports the winner's priority to the WSN platform
(15)        send_msg_prio(winner_prio);
(16)    }while(1);
(17) }
```

**Figure 56 Protocol implementation**

In the first place is performed the reception, from the host WSN platform, of the priority that will be used in the tournament phase (lines (7) to (10)). After that it is waited for the detection of the external synchronization signal, transmitted by the WiFLEX\_txsync board, which is given by the function `get_out_of_band_sync()` (line (11)); this function returns a

value different from “0” when the detection of the synchronization signal occurs). At this point nodes reset their timers and are synchronized (lines (12) and (13)), hence the tournament phase may be performed (line (14)). Until the next tournament phase takes place nodes do not get their timers synchronized, *i.e.* the timer is not reset in the sequence of the reception of an out-of-band synchronization signal. In this way, all computations based on the elapsed time since the synchronization moment suffer only from the drift inherent to the hardware clock, and never from the executed instructions like timer reset. This implementation requires a timer capable of counting at least one tournament phase without being reset. When using an MCU running at 8 MHz, a 16 bit timer is sufficient since the clock granularity is 0.125  $\mu$ s, which makes possible to count  $0.125 \mu\text{s} \times 2^{16} = 8192 \mu\text{s}$ .

As a result of the tournament phase, the winner’s priority is accessed and the procedure for transmitting the winner’s priority to the host WSN platform is carried out by the function `send_msg_prio(winner_prio)` (line (19)).

The tournament phase was implemented as explained in the WiDom protocol details in Sub-section 2.3.3. Each priority bit is used in the contention, and when a contention is lost the node participates in tournament by only hearing the medium. The code extract depicted in Figure 57 presents the most important parts of the tournament phase.

```
(1) inline U16 tournament_phase(U16 my_prio, bool *winner)
(2) {
(3)     U8 i = 0;
(4)     U16 winner_prio = 0;
(5)     ...
(6)     //start of preamble
(7)     dominant_test();
(8)     dominant_test();
(9)     //end of preamble
(10)    ...
(11)    //tournament phase
(12)    for (i = 0; i < NPRIOBITS ; i++)
(13)    {
(14)        winner_prio <<= 1;
(15)        if ( ((my_prio & PRIO_COMPARE) == PRIO_COMPARE) || (*winner == 0))
(16)        {
(17)            temp = recessive();
(18)            if (*winner == 1 && temp == 1) *winner = 0;
(19)            else if (temp == 0) winner_prio |= 1;
(20)        }
(21)        else
(22)        {
(23)            dominant();
(24)        }
(25)        if (*winner == 1) stuffing();
(26)        else if (*winner == 0) waiting();
(27)        my_prio <<= 1;
(28)    }
(29)    //end of tournament phase
(30)    return (winner_prio);
(31) }
```

**Figure 57** Tournament phase

As it can be observed, if the winner condition is lost, i.e. the boolean variable *winner* is equal to “0”, the node only hears the on-going tournament (observe the condition present in line (11)); even the bit-stuffing, performed by the function `stuffing()` (line (20)), is substituted (by the function `waiting()` at line(21)) and the node transmit no carrier wave pulse. At the end of the tournament phase, the winner’s priority value is returned (line (24)).

During the arbitration, every priority bit can be dominant or recessive. While recessive bits demand carrier sensing to be performed, the dominant ones imply the transmission of a carrier wave pulse during a predetermined period of time. Let the code depicted in Figure 58 represent the essential procedures inherent to the transmission of a dominant bit:

```
(1) inline void dominant (void)
(2) {
(3)     act_time = TCNT1;
(4)     antenna_to_tx();           //antenna path set to TX
(5)     carrier418_on();
(6)     while(TCNT1 < (U16) (PULSE_W + act_time) );
(7)     carrier418_off();
(8)     while(TCNT1 < (U16) (PULSE_W + SILENCE_T + act_time));
(9) }
```

**Figure 58 Dominant bit**

At the main entrance point of the procedure the current value of MCU timer is read to the variable `act_time` (line (3)). Next, the antenna path is set to enable the antenna connecting to the transmitter output (line (4)) and the carrier transmission is set on (line (5)). The carrier transmission is performed during the time defined by `PULSE_W`. In order to avoid clock drifts, the condition is calculated based on the sum of the value defined by `PULSE_W` and the clock value stored in `act_time` (line (6)). After this time the carrier transmission is set to off. In order to make every transmitted bit last for the correct time, before returning to the function call-out point, it is waited the necessary time for the MCU clock to reach the sum of `PULSE_W+SILENCE_T`. The same happens in case of recessive bits, although the time spent in carrier transmission is used for carrier sensing, and the function returns a boolean variable that is true in case a carrier was sensed.

The function `stuffing()` (responsible for introducing the bit-suffing) follows the implementation of the one used for the dominant bits. Conversely, the function `waiting()` does nothing, i.e. neither transmits or senses the medium, but introducing a pause for the duration of a stuffing bit.



## 4. EXPERIMENTAL EVALUATION

In order to assess important parameters of the developed platforms, such as energy consumption, reliability, transmission range, *etc....*, several experiments were conducted. This chapter is intended to summarize the most important experiments that were performed, in order to achieve the sufficient knowledge on the WiFLEX platform that allows the implementation of applications, as well as highlight problems that might be helpful for future developments based on the developed architecture.

The opportunities for efficient aggregated computations enabled by the WiDom protocol implementation in this new hardware platform are also highlighted across this section.

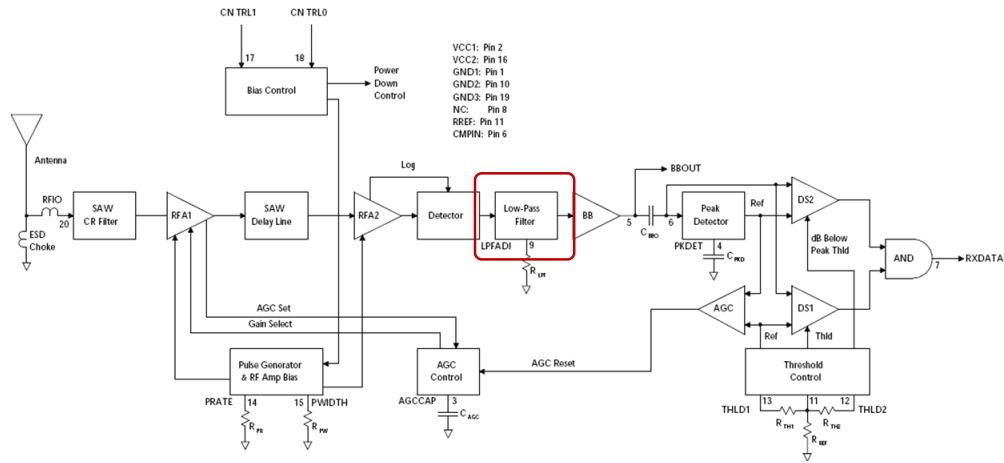


## 4.1. RADIO CHARACTERISTICS

Prior to any implementation or real-world scenario testing, a wide group of experiments were performed to assess some of the most important characteristics of the developed hardware. Phenomenon like self detection and carrier detection delay were exerted over Sub-section 4.1.1, opening the path for experiments that allow the determination of the carrier pulse widths characteristics that were found in Sub-section 4.1.2 and Sub-section 4.1.3. Although energy consumption was not a major issue for this work, such performance parameter is discussed in Sub-section 4.1.4, were a comparison between the best performing implementation of the WiDom protocol in the COTS MICAz WSN platforms is made with the implementation of the same protocol in the developed WiFLEX platform.

### 4.1.1. SELF DETECTION AND CARRIER DETECTION DELAY

Although the WiFLEX\_main board uses a separated transmitter and receiver, the switching between transmission and reception modes does not occur instantaneously. This is not due to the hardware inability to perform such context switching, but due to the fact that the receiving device (RFM's RX5002) presents a non-negligible delay between the moment it receives the unmodulated carrier wave and the moment its digital output signals such event. Accordingly to [54], the LPF group responsible for providing a 3 dB attenuation for the out of bandwidth signals, highlighted in Figure 59, introduces a delay that is a function of the filter's 3 dB bandwidth; such delay is therefore correlated with the value of the resistor  $R_{LPF}$ , and its maximum value is given by the approximated formula  $t = 1.21 * R_{LPF}$ , where  $R_{LPF}$  is in  $k\Omega$ , and  $t$  is in  $\mu s$  ([52] refers that, among other factors,  $t$  may vary with voltage and temperature). This results that, in the case of the adopted configuration (refer to Table 8), the maximum value expected for the delay introduce by the LPF group should be approximately  $t = 1.21 * 56 = 67.8 \mu s$ .



**Figure 59 RX5000 Series ASH Receiver Block Diagram [54]**

This limitation would force that, prior from going to receive mode, or more precisely, before reading the RXDATA pin of the RX5002 receiver device, a given amount of time should be waited if a transmission had previously occurred; otherwise, the value read from the receiving device would still be the result of the previous transmission.

Prior to determining a safe switching time from reception to transmission modes, an experiment was conducted to determine the carrier detection delay and to evaluate the accuracy of the referred approach, using a single node. To do so, one WiFLEX\_main was programmed to repeatedly transmit an unmodulated carrier wave for a period of 100  $\mu$ s followed by a silent period of 100  $\mu$ s, while the RXDATA pin of the RX5002 was monitored with a digital oscilloscope (detailed information about the used digital oscilloscope may be found in [67]). Additionally, an unused GPIO pin of the MCU was programmed to remain at high-level during the period of time between the rising edge of the signal applied to the TXMOD pin and the moment where the MCU's pin, connected to the RXDATA pin, perceives it went high. The triggering signal was obtained through the monitoring of the referred GPIO pin. In this way, it was possible to determine the response time of the RXDATA pin to a transmitted carrier wave.

Figure 60 shows an instance of the measurements taken during the experiment. These measurements were based on the GPIO pin high state duration, using the pulse trigger functionality of the digital oscilloscope. Successive approximations of the select pulse trigger width allowed to identify the maximum value of the detection delay. The analysis of the obtained results allows concluding that the signalling of a carrier detection, in a

single node, only occurs 40  $\mu\text{s}$  after the TXMOD pin of the transmitting device is set at high-level.

There was also the interest of knowing the response time of the RXDATA pin after turning off the transmission of a carrier wave. This procedure was similar to the one used in the previous experiment, however, the GPIO pin high-level was triggered by the falling edge of the TXMOD pin, and the low-level when the MCU perceived the RXDATA went low-level. Figure 61 depicts the instance of the referred measurements that represented the highest duration of the GPIO high-level state. This in term allowed concluding that the RXDATA pin can present a delay in going to low-level of, at most, 47  $\mu\text{s}$  after the TXMOD pin of the TX5002 is set to low-level.

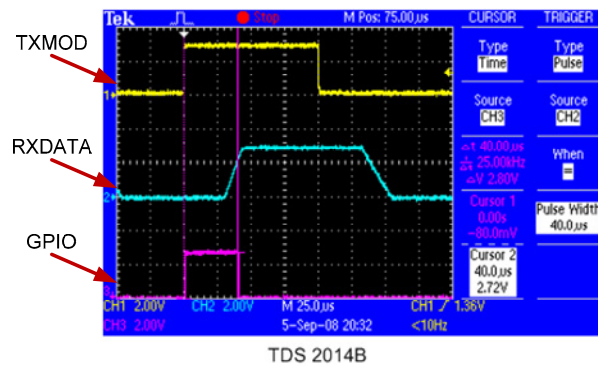


Figure 60 Response time of the RXDATA pin to a transmitted carrier wave

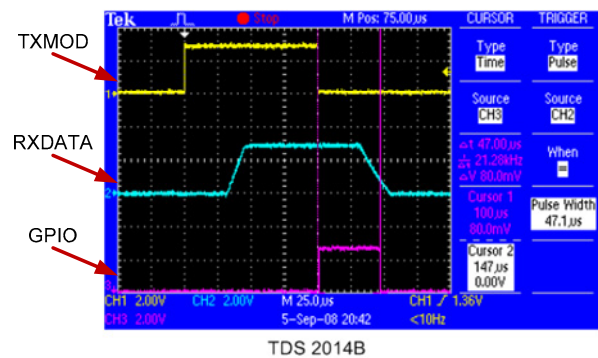


Figure 61 Response time of the RXDATA pin after turning off the transmission of a carrier wave

Other periods of transmission and silence were evaluated to determine if there was any influence in the previously described effects (response times of RXDATA pin to carrier detection). Even though, the same results were obtained for both cases of delay in detecting the start and the stop of a carrier transmission.

#### 4.1.2. DETERMINING PULSE WIDTH AND SWITCHING TIME

Although the radio devices had been hardware configured to work using specific minimum pulse width (minimum value of  $H$ ), one was interested in testing if operating the hardware in a different way, through the software that runs on the MCU, would improve the performance obtained. Nevertheless, such test would also allow accessing the performance characteristics when operating accordingly to hardware configuration.

As outlined in Sub-section 3.3 and Sub-section 4.1.1, despite the implemented architecture, due to several reasons, it is known that the switching time between reception and transmission modes cannot be either instantaneous or negligible. In this sense this parameter also had to be evaluated in order to determine the correct WiDom protocol parameters  $SWX$  and  $G$ .

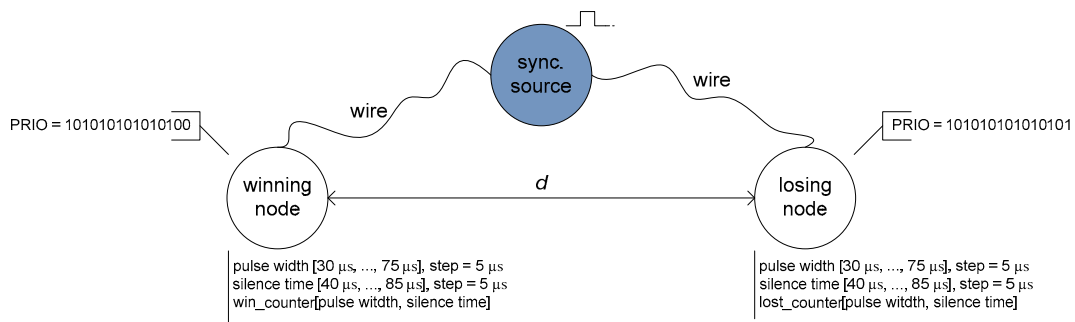
In order to determine such parameters, two nodes were set a distance  $d$  apart, and increased  $d$  in steps of 5 m, starting with the nodes as close as possible to each other, and up to 20 m. An intermediate step at a distance of 1 m was also tested. For each step, 1000 tournaments were performed using a static priority value in each node, and different combinations of minimum pulse width and intervals of silence. These combinations were formed by varying the minimum pulse width in the range from 30  $\mu\text{s}$  to 75  $\mu\text{s}$ , in steps of 5  $\mu\text{s}$ , and the value of the interval of silence from 40  $\mu\text{s}$  to 85  $\mu\text{s}$ , also in steps of 5  $\mu\text{s}$ . Static priorities were assigned in order that one of the nodes would always be the winner, and also in a way that the worst cases of consecutive periods of carrier transmission and no carrier transmission could be examined (refer to Sub-section 3.3.1). Additionally, each node was tested individually, hence always being the winner, in order to access the pulse width and silence times duration that resulted in its self hearing.

Figure 62 depicts an overview of the experimental setup. This test was performed in an open-field, with non-obstructed line-of-sight and 1 m above ground level. For each step, the combinations were tested automatically, *i.e.*, both nodes ran a program that performed the 1000 tournaments, and at the end stored the obtained results and incremented the value of the minimum pulse width and silence duration automatically.

After all values have been tested, the results were downloaded to a PC, through a RS232 link, and copied to an MS Excel<sup>TM</sup> cell sheet for future analysis. In order to ensure that no synchronization drift occurs, a GPIO pin in both nodes was used to receive a start of

tournament signal, through a wired connection, from a third node. This last did not participated on the tournament. During the test, since the host WSN platform was only necessary to provide power to the WiFLEX\_main board, its MCU was left running a busy loop.

The interpretation of the results presented in the Appendix F (Table 13 to Table 28), allowed to admit the values present in Table 29 highlighted from light-green to green colour as valid for switching time - silence time pairs with a reliability above 99.9 %. Combinations that result in a reliability value below 99.0 % are highlighted in red colour, and the ones within the interval ] 99.0 %, 99.9 % [ are highlighted in intermediate colours from red to light-green. The combinations that result in the minimum sum of time (total duration of 90  $\mu$ s) and simultaneously a reliability of 100 % are the ones using a carrier pulse width of 35  $\mu$ s and switching time of 55  $\mu$ s, and a carrier pulse width of 40  $\mu$ s and switching time of 50  $\mu$ s.



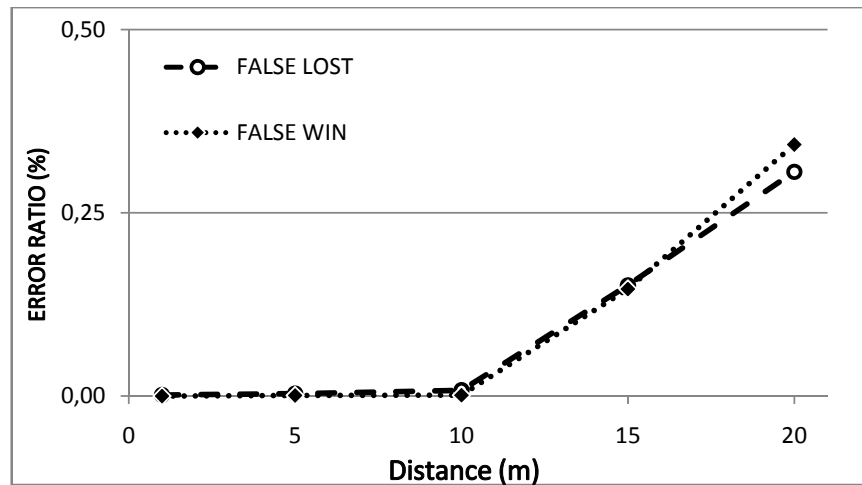
**Figure 62 Experimental set-up for minimum pulse width and silence interval determination**

#### 4.1.3. RELIABILITY VERSUS DISTANCE

In order to exhaustively test the reliability of the tournament phase at different distances, a similar experiment to the one described in the previous section was performed. Although this time 10 nodes were used. In addition, to test for cases where nodes can detect carrier pulses from a node close by, but not from a node far away, they were divided into two groups and one group was placed at each end. In each group, nodes were set with a minimum of 30 cm apart, and the distance between both groups varied from 1 m to 20 m, in steps of 5 m. To access if the priority value influenced the reliability of the tournament phase, and also not limiting the wining potential to a specific node, a random priorities table was produced, and the winner of each tournament was computed offline. This table was then downloaded to the nodes and the priorities in the table were used in cycles until

150000 tournaments were performed for each step. Since nodes know the priority of the winner in each tournament from the table results computed offline, the number of tournaments failed (both events of failing to win or lose a tournament were individually counted) could be collected from each individual node and downloaded to a PC for future evaluation. As a result of the previous experiment, the carrier pulse width and switching time was configured to be, respectively, 40  $\mu$ s and 50  $\mu$ s.

The results of the experiment are shown in Appendix G, from which can be resumed the graphic presented in Figure 63. The error ratio is presented as a function of the distance between the two groups of nodes, and both events of failing to win the tournament and failing in losing the tournament (*i.e.* a node that should have lost the tournament, but instead it won) are depicted. The results demonstrate that, for networks where the distance between all nodes is below 5 m, the communication can be considered error-free, and also the number of tournaments failed at a distance of 20 m is very small.



**Figure 63 Failed tournaments with distance**

#### 4.1.4. ENERGY CONSUMPTION

One of the parameters regarding the efficiency of the WiFLEX platform was its energy consumption. This evaluation was made considering the comparison between the implementation of the WiDom protocol in the WiFLEX platform and the previous implementation, presented in [48] and [49] that used the CC2420 radio transceiver present

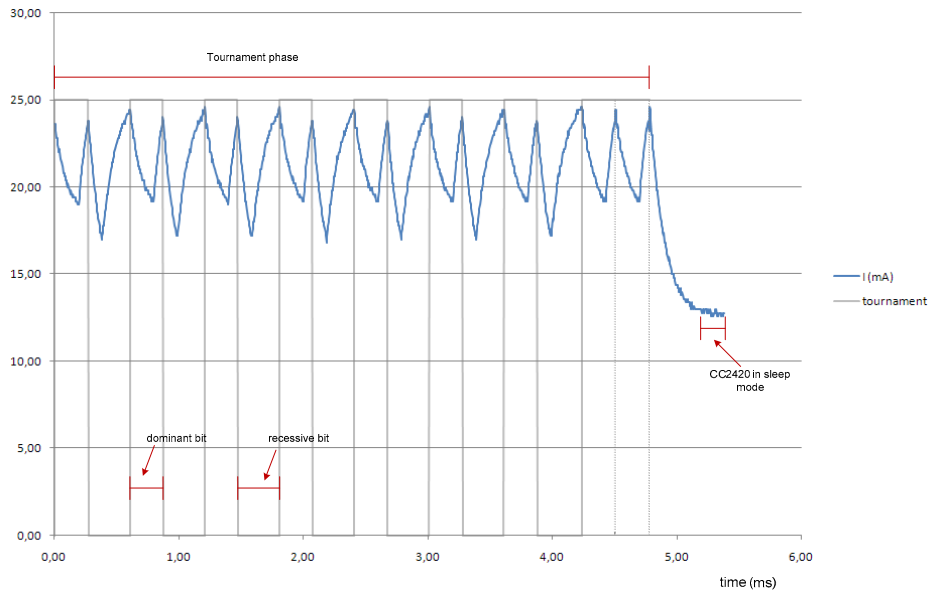
in the MICAz WSN platform. The comparison was made through the analysis of the energy consumed, during a complete tournament, by both platforms.

To accommodate the added weight of the bit-stuffing introduced in the WiFLEX platform, to shape the dominant and recessive bits, the dominant and recessive bits transmitted during the measurements realized in the WiFLEX platform were not removed, although they were not used in the implementation using the CC2420. The experiment compares the theoretical best case performance of the CC2420 operating (as required) in OOK modulation, which accordingly to [15] gives a minimum of 256  $\mu\text{s}$  for the dominant bits (to allow a receiving node to perform carrier sensing twice) and 320  $\mu\text{s}$  for the recessive ones (this is the time to switch between transmission/reception mode – 192  $\mu\text{s}$ , added to the time until the first CCA operation). Note that these parameters do not result in a valid implementation of the WiDom protocol, and experiments in [49] have showed that these are very short in order to have a reliable implementation.

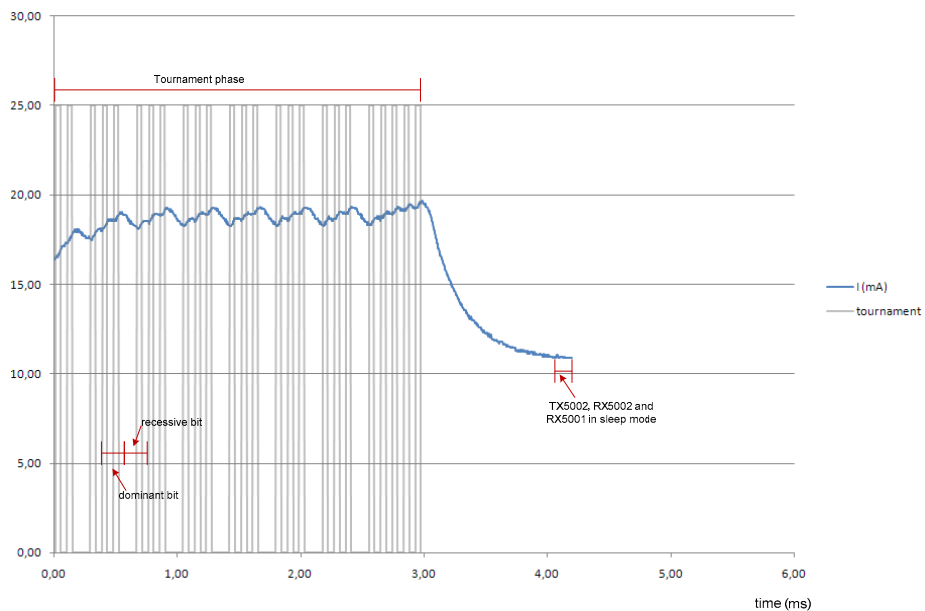
In the experiment using the WiFLEX platform the bit parameters used are the same as the ones in the experiment described in Sub-section 4.1.3 (carrier wave pulse width of 40  $\mu\text{s}$  and switching time of 50  $\mu\text{s}$ ).

The measurements were realized using the digital storage oscilloscope TEKTRONIX TDS2014B [67]. Results were acquired and exported to MS Excel<sup>TM</sup> for future interpretation. During the experiment that tested the WiFLEX platform energy consumption, the CC2420 radio transceiver onboard the MICAz WSN platform was turned off, and the microcontroller performed a busy loop. The WiFLEX\_main board was fitted with the WiFLEX\_rxsync board and all the hardware was set into active mode. During the experiment that measured the energy consumption of the previous implementation, the WiFLEX\_main board (along with the WiFLEX\_rxsync) was disconnected from the MICAz WSN platform and the CC2420 transceiver was used. In both experiments, a  $2^{16}$  level priority value was used. In order to simplify the data acquisition was used a priority value of 01010101010100 (binary).

Figure 64 depicts the trace of the current consumed using the MICAz WSN platform at a supply voltage of approximately 3 V, whereas Figure 65 presents the trace of the current consumed using the WiFLEX\_main board fitted with the WiFLEX\_rxsync board during a tournament.



**Figure 64 Trace of current consumption, during a tournament phase, for the WiDom implementation using the MICAz's CC2420 radio transceiver**



**Figure 65 Trace of current consumption, during a tournament phase, for the WiDom implementation using the MICAz's with the WiFLEX\_main and WiFLEX\_rxsync boards**

It can be observed that the implementation of the WiDom protocol using on the MICAz WSN platform, relying on the Chipcon's CC2420 radio transceiver, not also takes more time to perform a tournament (approximately 4.78 ms against the 2.97 ms when using the WiFLEX platform) but also consumes more current to perform the referred tournament.



Since during the experiment the value of the batteries voltage was also collected, and being known the sampling period used by the digital oscilloscope, it was possible to calculate the energy consumption and integrate such value to indirectly calculate the energy consumed during a tournament by each platform. The performed calculations revealed that the WiDom implementation in the MICAz platform consumed approximately 0.293 mJ and the WiFLEX platform based implementation consumed a total energy of 0.160 mJ to perform a tournament phase.

It was also possible to determine that the MICAz WSN platform having the CC2420 radio transceiver in sleep mode consumes more energy than the MICAz WSN platform fitted with the WiFKEX\_main+WiFLEX\_rxsync boards, when the radio devices of the former boards (TX5002, RX5002 and RX5001) are set into sleep mode.

## **4.2. AGGREGATE DATA COMPUTATION**

It is inevitable that large-scale, sensor-rich networked systems will generate a massive amount of sensor data. Techniques have been proposed for computing aggregated quantities, such as minimum (MIN) and maximum (MAX) values, the number of nodes (COUNT) and the MEDIAN among a set of sensor nodes, usually involving the organization of nodes in a converge-cast tree having leaf nodes broadcasting its data. These approaches offer a good performance, although their time complexity is dependent of the number of sensor nodes.

Andersson *et. al* presented a solution for such problem in [4], showing that it is possible to calculate, among others, such aggregated quantities with a time complexity independent of the number of network nodes. The referred work was implemented in a wired system. However, since it is based on a dominance/binary countdown MAC protocol, its concept may be used wirelessly.

In this line, the mechanism presented in [4] was implemented in NANO-RK OS, By Nuno Pereira, resorting to the WiFLEX platform running the WiDom protocol (providing the necessary dominance/binary countdown MAC protocol).

In this way, it was possible to test the developed platforms in a demanding application, but also to validated for the first time the computation of aggregate quantities in wireless

medium using a dominance/binary countdown MAC protocol, with an approach where the time-complexity of computations is independent of the number of nodes in the network.

The next sub-section (Sub-section 4.2.1) presents a resumed background on the work presented in [4] in order to understand its basic work principle. In this sense, the demands inherent to such aggregate quantities computation can be better understood, as well as the encouraging presented results. Sub-section 4.2.2 establishes a clear comparison between the necessary time to compute MIN in a WSN using IEEE 802.15.4 at its top performance versus the time required using the implementation described along this thesis when the interpolation scheme presented in [4] is enabled. Along Sub-section 4.2.3 are described the experimental procedure, as well as the results provided from the performed experiment.

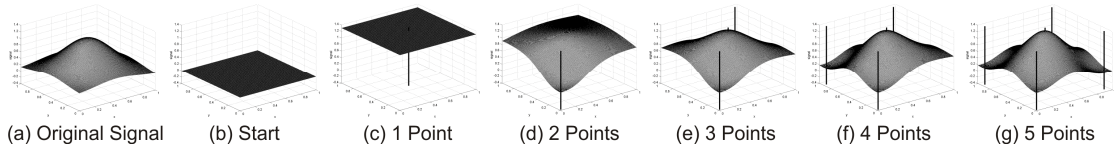
#### 4.2.1. BACKGROUND ON WISE-CAN

The basis for the design of the interpolation scheme is similar to the one proposed, implemented and tested in [4]. In this scheme, all nodes use the same type of function to interpolate the sensor data. Nodes start with the interpolation function being a flat surface and compute the error between their sensor reading and the interpolation function. Exploiting the dominance MAC protocol, the nodes then use their error as part of the priority used to contend for the medium, such that the data point with the MAX of all errors is found. Thanks to the dominance MAC protocol propriety, all nodes in the network access the priority of the winner, *i.e.* the referred MAX value. The data point found is then used by all nodes to recompute the interpolation function. Nodes iterate through this procedure for a predefined number of times ( $k$ ). At the end of these iterations, the subset of  $k$  nodes that contribute to the interpolation is found.  $k$  is a parameter that defines the accuracy of the interpolation. The reasoning for finding only a subset of nodes is explained by the fact that sensor readings often exhibit spatial locality [27]; that is, nodes that are close in space give similar sensor readings. For this reason, the interpolation offers a low error even if only a small number of carefully selected sensors reading is used. Note that not just any subset of nodes is selected; it is the subset of nodes that has, at each iteration, the biggest error to the interpolation function, and thus can contribute the most to approximate the physical phenomenon.

The previous description, impose that nodes must broadcast their location after contending for the medium. This requires that the priority used for the contention is divided in two

parts: one used for the value computed as a function of the physical quantity of interest, and the other part for a unique identifier of the node. In this way, is guaranteed that only one node can win the contention for the medium and no collisions occur during the transmission of the location data.

Figure 66 illustrates the operation of the interpolation scheme for  $k = 5$ , for the original signal in Figure 66(a). At the beginning (b), no points are selected. Therefore, the interpolation is a plane surface. Then (c), each node calculates the error between its sensor reading and the starting plane surface. This error is used in the contention of the MAC protocol, causing the node with the largest error to win, and thus, it is selected; from (d) to (g) nodes proceed similarly, calculating their error to the currently interpolated surface and adding the node with the largest error to the interpolation, until  $k$  points have been selected.



**Figure 66 Iterations of the interpolation scheme [4]**

#### 4.2.2. COMPUTING MIN

We will now present the implementation of a simple application to compute MIN among a set of neighbour nodes in real-world platforms, using standard IEEE 802.15.4 and compare this implementation with one using a dominance MAC protocol. We will analyze the time to compute MIN using a naïve algorithm, when all nodes send their sensor readings.

The analysis assumes that message transmission times dominate, and thus only considers the time needed to convey all messages necessary to compute MIN. For the sake of simplicity, it is also assumed that the number of nodes  $m$  is known, when using the naïve algorithm.

Referring to [31], in the beacon-enabled mode, beacon frames are periodically sent by a coordinator (the PAN coordinator) to synchronize nodes that are associated with it. The Beacon Interval ( $BI$ ) defines the time between two consecutive beacon frames. The beacon interval can be divided into an active period, and optionally an inactive period. The active period, called Superframe, is divided into sixteen equally-sized timeslots, and its duration is defined by a parameter called Superframe Duration ( $SD$ ). For this analysis, to impose the

minimum possible protocol overhead, the smallest  $BI$  and  $SD$  is desired. Given this constraint, was considered  $BI = SD = 15360 \mu\text{s}$ , which is the minimum  $SD$  defined by the standard. This means that nodes are at a 100 % duty cycle and the duration of each timeslot is  $T_s = SD/16 = 15360/16 = 960 \mu\text{s}$ . In IEEE 802.15.4, a Superframe is further divided into a CAP, where a slotted version of CSMA-CA is employed and a contention-free period (CFP), composed of a maximum of seven timeslots, here named GTS. Furthermore, within the same Superframe, each node can only have one GTS upstream and one GTS downstream.

For the given comparison, only the CFP is assume that, at each  $BI$ , the PAN coordinator assigns one timeslot to each node, such that all nodes are queried as soon as possible. Thus, the minimum time to compute MIN  $T_{computeMIN}$  is the time to send  $m$  messages, and can be computed as follows:

$$T_{computeMin} = \left\lceil \frac{m}{7} \right\rceil \times BI + (m \times \text{mod } 7 - 1) \times T_s + T_{MinReply}(S) + T_{MinCAPLen} \quad (6)$$

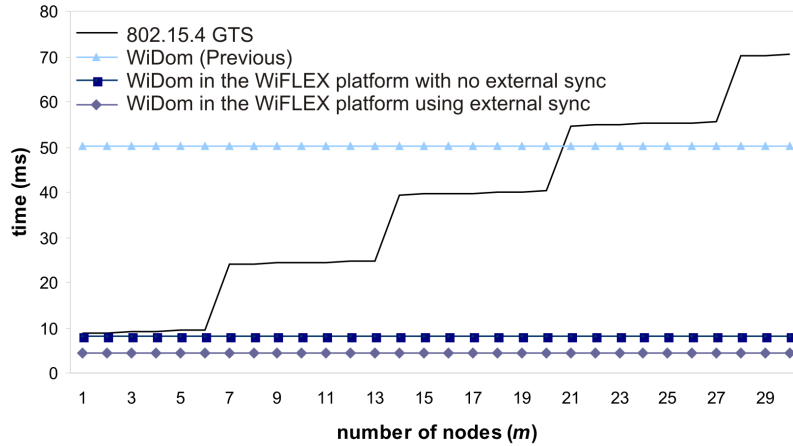
where the size of the CAP in the last Superframe necessary to contact all nodes is  $T_{MinCAPLen} = 7040 \mu\text{s}$ , the minimum defined in the standard. Note however that this is a lower bound on the time to send  $m$  messages.

$T_{MinReply}(S)$  is the minimum time to transmit the PHY protocol data unit (PPDU) with the minimum size allowed by the standard, with a payload of size  $S$  bits:

$$T_{MinPPDU} = (S_{MinHeaders} + S + S_{Footer}) \times \tau_{bit} \quad (7)$$

where  $S_{MinHeaders}$  is sum of the sizes of the synchronization header (SHR), PHY header (PHR) and MAC header (MHR; from [31]:  $S_{SHR} = 40$ ;  $S_{PHR} = 8$ ;  $S_{MHR} = 56$  bits, considering the minimum size of the addressing fields). The size of the MAC footer is  $S_{Footer} = 16$  bits. The time to transmit one bit is  $\tau_{bit} = 4 \mu\text{s}$  (for a data rate of 250 kb/s). We consider that our reply is 16 bits long, thus  $S = 16$ .

Exploiting a dominance MAC protocol to compute MIN relying in the basis of the implementation described in the previous sub-section, results that the time to compute MIN is equal to the time to perform one arbitration of the dominance MAC protocol. Previous implementations of such dominance MAC protocol for wireless systems [48], have show that this protocol can be implemented, and the time to perform the arbitration is  $T_{ComputeMIN} = 50244 \mu\text{s}$ .



**Figure 67 Time to compute MIN as a function of the number of nodes ( $m$ )**

Applying this analysis to study the time to compute MIN as a function of the number of nodes results in Figure 67. One can observe that computing MIN using the implementation based on the dominance MAC protocol using the hardware platform described in this thesis is always more advantageous than the implementation using IEEE 802.15.4. For systems with  $m > 21$ , even the previous implementation of WiDom is superior to the setup using IEEE 802.15.4.

#### 4.2.3. WIRELESS INTERPOLATION

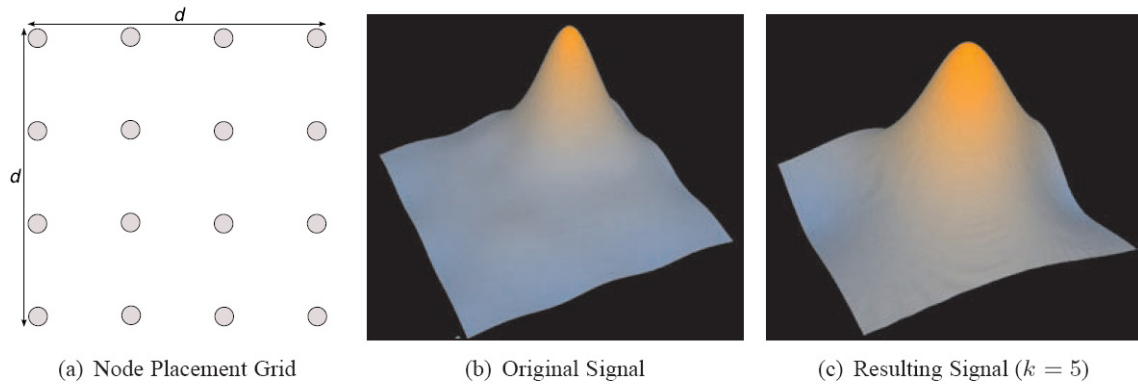
The interpolation scheme was implemented in the operating system NANO-RK and used in MICAz WSN platforms. The algorithm for the interpolation implemented was as presented in [4]. The improved algorithm avoids floating point calculations, which is very expensive in terms of processing for these WSN platforms, was used.

Using this implementation, an experiment was setup to study the reliability of the interpolation. Sixteen nodes<sup>6</sup> were deployed at equal distance from each other, to form a grid of  $d \times d$ , as depicted in Figure 68(a). A signal as depicted in Figure 68(b) was constructed, and fixed the data points in each node according to this signal. In this way, the resulting interpolation (depicted in Figure 68(c)) was known in advance. The interpolation was then performed 10000 times for each step of  $d$  that was tested. The experiments were

---

<sup>6</sup> Experiments with larger number of nodes could not be performed due to the limited number of WiFLEX platforms available.

conducted in an office environment (being the maximum value  $d$  constrained by the available space) one meter above ground level.



**Figure 68 Interpolation experiment**

Performing the interpolation requires that not only the node with the highest priority wins, but also that all other nodes perceive the correct priority of the winner. Table 6 presents the results obtained, indicating the percentage of errors in 10000 cycles of the interpolation for each value of  $d$ . It is possible to observe that even in such a demanding application, the WiFLEX platform offers reliable support for dominance-based aggregate computations over wireless medium.

**Table 6 Interpolation experiment results over varying  $d$**

d (m)	Error Ratio (%)
1.5	0.106
3	0.148
6	0.155



## 5. CONCLUSIONS AND FUTURE WORK

This thesis presented a new hardware platform for the implementation of dominance protocols for wireless medium access. In the absence of a satisfactory COTS solution, a dedicated hardware architecture was developed, discussed and evaluated. Notably, although introducing several additional hardware components, the overall system power consumption remained below the power consumption using the widespread CC2420 radio transceiver, spending less 45 % of the required energy to perform a tournament phase of the WiDom protocol, even though such comparison was not fair towards the newly developed platform.

Although an acceptable energy consumption improvement was achieved, the protocol's overhead reduction was one of the most relevant accomplishments. The implementation of the WiDom MAC protocol relying in the WiFLEX platform presented an overhead reduction of more than ten times when compared with the implementation based on COTS WSN platforms presented in [48].

It is desirable to perform a wider range of tests with the WiFLEX platform, namely the ones regarding density and scalability, *i.e.*, setting up a network with a larger number of



nodes (*e.g.* 100) and also extend the inter-node distance and coverage area, testing the MAC protocol reliability. Further developments should also take in account the implementation of the MBD version of the WiDom protocol. Nevertheless, future work involving hardware design for the implementation of wireless dominance protocols based on OOK modulation, and relying on the results presented in this thesis, must always consider the most prominent protocol parameters. In a summarized way, the necessary switching time between RX and TX modes, and the necessary pulse width for proper carrier detection, must be as low as possible. Regarding the used radio devices, if the shortcomings referred in Sub-section 3.2.2.1 and Sub-section 4.1.1 could be eliminated, along with the bit-stuffing, the protocol overhead could be dramatically reduced, representing, comparatively with the obtain results, half overhead and energy consumption.

Studying the performance of the developed platform, as well as the protocol itself, in non-static WSN would also be an interesting and important exercise to support future developments.

Nevertheless, the developments around wireless Network-on-Chip could also greatly benefit from an architecture similar to the one presented in this thesis, or to the reasoning behind its conception. As an example, in the work presented on [70], the presented system could benefit from a wireless dominance/dominance countdown protocol to eliminate the wires used to perform the MAC, and rely on the external wireless synchronization for extremely precise and constant synchronization requirements.

Despite the quantifiable results achieved with the developed hardware platform, the foremost relevant contribution of this thesis was proving that, at the moment of the presentation of [48] and today, the implementation of dominance/binary countdown protocols for wireless MAC is possible, feasible and reliable, and that the limitations inherent to their use may be alleviated at the cost of dedicated hardware platforms and radio devices. Despite other issues, that are out of the scope of this thesis (*e.g.* security), it is possible to bring hard real-time applications relying in a dominance/binary countdown MAC protocol and dedicated efficient hardware platforms to WSN, along with all its benefits and wide range of applications.

## References

- [1] ABRACON CORPORATION-*Datasheet: Ceramic Surface Mount Low Profile Quartz Crystals – ABMM2*. Esperanza, Rancho Santa Margarita, California. Abracon Corporation, 1 May 2007.
- [2] AKYILDIZ, I. F.; SU, W.; SANKARASUBRANABIAM, Y.; CAYIRCI, E.-*Wireless sensor networks: A survey*. Computer Networks, vol. 38, issue 4, pp. 393-422. March 2002.
- [3] ANALOG DEVICES-*Datasheet ADG918/ADG919: Wideband 4 GHz, 43 dB Isolation at 1 GHz, CMOS 1.65 V to 2.75 V, 2:1 Mux/SPDT Switches*. Norwood, USA. Analog Devices, Inc. 2004.
- [4] ANDERSSON, Björn; PEREIRA, Nuno; ELMENREICH, Wilfried; TOVAR, Eduardo; PACHECO, Filipe; CRUZ, Nuno-*A Scalable and Efficient Approach for Obtaining Measurements in CAN-Based Control Systems*. IEEE Transactions on Industrial Informatics (TII08), pp. 80-91. IEEE, May 2008.
- [5] ASH, Darrell L.-*SAW-based Hybrid Transceivers in Slam Packaging With Frequency Range from 200 to 1000 MHz*. Dallas, Texas. RF Monolithics, Inc..
- [6] ATMEL-*Application Note AVR040: EMC Design Considerations*. San Jose, USA. ATMEL, June 2006.
- [7] ATMEL-*Application Note AVR042: AVR Hardware Design Considerations, PRELIMINARY*. San Jose, USA. ATMEL, June 2006.
- [8] ATMEL-*Application Note: EMC Improvement Guidelines*. San Jose, USA. ATMEL, August 2003.
- [9] ATMEL-*Datasheet: ATmega48/88/168V*. San Jose, USA. ATMEL, September 2007.
- [10] ATMEL-*Datasheet: ATmega48/88/168V, SUMMARY*. San Jose, USA. ATMEL, August 2007.
- [11] BOSCH-*CAN Specification, ver2.0*. Stuttgart, Germany. Robert Bosch GmbH, 1991.
- [12] BURNS, Alan; WELLINGS, Andy-*Real-Time Systems and Programming Languages*. 3<sup>rd</sup> Ed.. Addison Wesley, March 2001. ISBN: 0201729881.
- [13] CACCAMO, M.; ZHANG, Y.-*An Implicit Prioritized Access Protocol for Wireless Sensor Networks*. In proceedings of the 23rd IEEE Real-Time Systems Symposium (RTSS'02), pp. 39-48. Austin, Texas. IEEE, 2002.
- [14] CARLSON, A. Bruce-*Communication Systems – An Introduction to Signals and Noise in Electrical Communication*. 3<sup>rd</sup> Ed. McGraw-Hill Companies, Inc, 1986. ISBN 0-07-100560-9.
- [15] CHIPCON AS-*Datasheet: CC2420 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver*. Oslo, Norway. Chipcon AS, 2006.

- [16] *CMUcam3*. Online at: <http://www.nanork.org/nano-RK/wiki/cmucam3>.
- [17] *CMUcam3: Open Source Programmable Embedded Color Vision Platform*. Online at: <http://www.cmucam.org/>.
- [18] COTTET, Francis; DELACROIX, Joëlle; KAIZER, Claude; MAMMERI, Zoubir-  
*Scheduling in Real-Time Systems*. West Sussex, England. John Wiley & Sons, LTD, 2002. ISBN 0-470-84766-2.
- [19] *CROSSBOW-Datasheet: MICAz*. San Jose, USA. Crossbow Technology, Inc., 2004.
- [20] *CROSSBOW-MPR-MIB Users Manual*. San Jose, USA. Crossbow Technology, Inc., June 2007.
- [21] *CROSSBOW-Product Reference Guide*. San Jose, USA. Crossbow Technology, Inc., 2007.
- [22] *FireFly Hardware Clock Synchronization*. Online at: <http://www.nanork.org/nano-RK/wiki/firefly-hardware-clock-sync>.
- [23] *FireFly Power Monitoring and Control Board*. Online at: <http://www.nanork.org/nano-RK/wiki/firefly-power-control>.
- [24] *FireFly Sensor Board*. Online at: <http://www.nanork.org/nano-RK/wiki/firefly-sensor-basic>.
- [25] *FireFly Sensor Networks: Real-Time Wireless Sensor Network Platform*. Online at: <http://www.ece.cmu.edu/firefly/>.
- [26] FULLMER, C. L.; GARCIA-LUNA-ACEVES, J. J.-*Floor acquisition multiple access for packet-radio networks*. In proceedings of the Proc. ACM SIGCOMM 95, pp. 262-273. Cambridge, Massachusetts, USA. ACM, 1995. ISBN: 0-89791-711-1.
- [27] GUESTRIN, C.; BODIK, P.; THIBAU, R.; PASKIN, M.; MADDEN, S.-  
*Distributed regression: an efficient framework for modelling sensor network data*. In Proceedings of the 3rd International Conference on Information Processing in Sensor Networks (IPSN04). New York, USA. ACM, 2004. ISBN: 1-58113-846-6.
- [28] HARTLEY, Rick-*RF/Microwave PC Board Design and Layout*. L-3 Avionics Systems.
- [29] HASS, Z. J.; DENG, J.; TABRIZI, S.-*Collision-free Medium Access Control Scheme for Ad Hoc Networks*. In proceedings of the Proc. of IEEE Military Communications Conference (MILCOM '99), pp. 276-280. Atlantic City, NJ, USA, 1999. ISBN: 0-7803-5538-5.
- [30] HILL, Jason Lester-*System Architecture for Wireless Sensor Networks*. Dissertation of PhD in Computer Science. University of California, Berkeley. Spring 2003.
- [31] *IEEE-IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements - Part 14.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (LR-WPANs)*: LAN/MAN Standards Committee of the IEEE Computer Society. October 2003.

- [32] JAMIESON, K.; BALAKRISHANAN, H.; TAY, Y. C.-*Sift: a MAC Protocol for Event-Driven Wireless Sensor Networks*. In proceedings of the 3rd European Workshop on Wireless Sensor Networks (EWSN). Zurich, Switzerland, 2006.
- [33] JANASWAMY, Ranakrishna-*Radiowave Propagation and Smart Antennas for Wireless Communications*. Springer, 2001. ISBN: 0792372417.
- [34] KARN, P.-*MACA - A New Channel Access Method for Packet Radio*. Proceedings of the ARRL/CRRL Amateur Radio 9th Computer Networking Conference, pp. 134-140. Ontario, Canada, 1990.
- [35] KASU, Srinivasa Reddy; BELLANA, Santosh Kumar; KUMAR, Chiranjeev-*A Binary Countdown Medium Access Protocol Scheme for Wireless Sensor Networks*. 10<sup>th</sup> International Conference on Information Technology (ICIT 2007), pp. 122-126. IEEE, 2007. ISBN: 0-7695-3068-0.
- [36] KRISHNA, C. M.; SHIN, Kang G.-*Real-Time Systems*. McGraw-Hill Companies, Inc, 1997. ISBN 0-07-114243-6.
- [37] LAPLANTE, Philip A.-*Real-Time Systems Design and Analysis*. 3rd Ed. IEEE, 2004. ISBN 0-471-22855-9.
- [38] LANN, Gérard Le- *Critical Issues for the Development of Distributed Real-Time Computing Systems*. France. IEEE, October 1990. ISBN 0-8186-2088-9.
- [39] LINX-*LINX Technologies – Product Overview Guide*. Merlin, USA. Linx Technologies, Inc., 2007.
- [40] MAHALIK, Nitaigour P. (Ed.)-*Sensor Networks and Configuration – Fundamentals, Standards, Platforms and Applications*. Springer-Verlag, 2007. ISBN 3-540-37364-0.
- [41] MANGHARAM, R.; RAJKUMAR, R.-*MAX: A Maximal Transmission Concurrency MAC for Wireless Networks with Regular Structure*. In proceedings of the IEEE 3rd International Conference on Broadband Communications, Networks and Systems (IEEE BROADNETS'06). San Jose, USA. IEEE, 2006.
- [42] MARRON, P.; MINDER, D.-*Embedded WiSeNts Research Roadmap (Deliverable 3.3)*. Berlin, Germany. Logos Verlag, 2006. ISBN: 3-8325-1424-4.
- [43] MOK, Aloysius K.; WARD, Stephen A.-*Distributed Broadcast Channel Access*. In Proceedings of Computer Networks 3, 1979, pp. 327-335. 1979.
- [44] *Nano-RK: A Wireless Sensor Networking Real-Time Operating System*. Online at: <http://www.nanork.org/nano-RK/>.
- [45] NELSON, Bob-*Simple Clock & Data Recovery*. Dallas, Texas. RF Monolithics, Inc., 2 February 2003.
- [46] NELSON, Dan R.-*ASH Design Assistant software v2.5.5*. RF Monolithics, Inc., RFM Europe. Online at: <http://www.rfm.com/corp/ashdesign.htm>.
- [47] PAUL, Clayton R.-*Introduction to Electromagnetic Compatibility*. Wiley-Interscience, 2006. ISBN: 0471758140.

- [48] PEREIRA, Nuno; ANDERSSON, Björn; TOVAR, Eduardo-*Implementation of a Dominance Protocol for Wireless Medium Access*. In Proceedings of the 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA06) IEEE, pp. 162-172. IEEE, 2006. ISBN: 1533-2306.
- [49] PEREIRA, Nuno; ANDERSSON, Björn; TOVAR, Eduardo-*WiDom: A Dominance Protocol for Wireless Medium Access*. In Proceedings of IEEE Transactions on Industrial Informatics (TII07), pp. 120-130. IEEE, May 2007. ISBN: 1551-3203.
- [50] PROKOP, Jon S.-*Assembling SRD Products Onto Customer's PWBs*. Dallas, Texas. RF Monolithics, Inc.
- [51] *RFLINX OOK add-on board a.k.a. Wings*. Online at: <http://www.nanork.org/nano-RK/wiki/wings>.
- [52] RF MONOLITHICS, INC.-*ASH Transceiver Designer's Guide*. Dallas, Texas. RF Monolithics, Inc., 19 May 2004.
- [53] RF MONOLITHICS, INC.-*Datasheet: RX5001 – 315.00 MHz Hybrid Receiver*. RF Monolithics, Inc., RFM Europe.
- [54] RF MONOLITHICS, INC.-*Datasheet: RX5002 – 418.00 MHz Hybrid Receiver*. RF Monolithics, Inc., RFM Europe.
- [55] RF MONOLITHICS, INC.-*Datasheet: TX5001 – 315.00 MHz Hybrid Transmitter*. RF Monolithics, Inc., RFM Europe.
- [56] RF MONOLITHICS, INC.-*Datasheet: TX5002 – 418.00 MHz Hybrid Transmitter*. RF Monolithics, Inc., RFM Europe.
- [57] ROWE, Anthony-*FireFly 2.2 Preliminary Datasheet v0.90*. Pittsburgh, Pennsylvania. Real-Time and Multimedia Lab, Carnegie Mellon University, 20 August 2007.
- [58] ROWE, A.; MANGHARAM, R.; RAJKUMAR, R.-*RT-Link: A Time-Synchronized Link Protocol for Energy-Constrained Multi-hop Wireless Networks*. In proceedings of the 3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks (SECON '06), pp. 402-411. IEEE, 2006.
- [59] RAJENDRAN, V.; OBRACZKA, K.; GARCIA-LUNA-ACEVES, J. J.-*Energy-Efficient, Collision-Free Medium Access Control for Wireless Sensor Networks*. In proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SensSys'03), pp. 181-192. Los Angeles, California, USA. 2003.
- [60] SHANNON, C.E.-*A mathematical theory of communication*. Bell System Technical Journal, 27. 379-423 and 623-656, 1948.
- [61] SMITH, Kent-*Antennas for Low Power Applications*. Dallas, Texas. RF Monolithics, Inc.
- [62] STALLINGS, William-*DATA & Comp. Communications*. 6<sup>th</sup> Ed.. New Jersey, USA. Prentice Hall International, Inc., 2000. ISBN: 0-13-086388-2.

- [63] STANKOVIC, John A.; ABDELZAHER, Tarek F.; LU, Chenyang; SHA, Lui; HOU, Jennifer C.-*Real-Time Communication and Coordination in Embedded Sensor Networks*. In Proceedings of the IEEE, vol.91, No.7. IEEE, July 2003.
- [64] STANKOVIC, John A.; LEE, Insup; MOK, Aloysius; RAJKUMAR, Raj-*Opportunities and Obligations for Physical Computing Systems*. IEEE Computer Society, November 2005.
- [65] STANKOVIC, John A.-*Misconceptions About Real-Time Computing: A Serious Problem for Next-Generation Systems*. IEEE Computer, October 1988.
- [66] STOK, Peter van der-*WASP: Deliverable D1.2 State of the Art*. Information Society Technologies, March 2007.
- [67] TEKTRONIX-*TDS1000B and TDS2000B Series Digital Storage Oscilloscope User Manual*. Beaverton, USA. Tektronix.
- [68] TINDELL, K.; HANSSON, H.; WELLINGS, A.-*Analysing real-time communications: controller area network (CAN)*. In 15th Real-Time Systems Symposium (RTSS'94), pp. 259-263. IEEE, December 2004. ISBN: 0-8186-6600-5.
- [69] TOBAGI, F.A.; KLEINROCK, L.-*Packet Switching in Radio Channels: Part II - The Hidden Terminal Problem in Carrier Sense Multiple-Access and the Busy-Tone Solution*. IEEE Transactions on Communication, vol.23 issue 12 pp. 1417-1433, IEEE, December 1975.
- [70] WANG, Yi; ZHAO, Dan-*The Design and Synthesis of A Synchronous and Distributed MAC Protocol for Wireless Network-on-Chip*. Proceedings of Computer-Aided Design, 2007. ICCAD 2007. IEEE/ACM International Conference, pp. 612 - 617. IEEE, 2007. ISBN: 978-1-4244-1382-9.
- [71] YOU, T.; Yeh, C.H.; HASSANEIN, H. S.-*A New Class of Collision - Prevention MAC Protocols for Ad Hoc Wireless Networks*. Proceedings of the IEEE International Conference on Communications, pp. 1135 - 1140 vol.2. IEEE, 2003. ISBN: 0-7803-7802-4.
- [72] YOU, T.; Yeh, C.H.; HASSANEIN, H. S.-*BROADEN: An efficient collision-free MAC protocol for ad hoc wireless networks*. Proceedings of the IEEE International Conference on Local Computer Networks, pp. 698 – 707. IEEE, 2003. ISBN: 0-7695-2037-5.
- [73] YOU, T.; Yeh, C.H.; HASSANEIN, H. S.-*CSMA/IC: A New Class of Collision-free MAC Protocols for Ad Hoc Wireless Networks*. Proceedings of the 8th IEEE International Symposium on Computers and Communication, pp. 843-848. IEEE, 2003. ISBN: 0-7695-1961.
- [74] ZHAO, F.; GUIBAS, L. J.-*Wireless sensor networks: an information processing approach*. Amsterdam. Elsevier, 2004. ISBN: 1558609148.
- [75] ZHENG, R.; SHA, L.-*MAC Layer Support for Group Communication in Wireless Sensor Networks*. Department of Computer Science, University of Houston UH-CS-05-14, July 21 2005. online at: <http://www.cs.uh.edu/Preprints/preprint/uh-cs-05-14.pdf>.

- [76] ZHOU, G.; HE, T.; KRISHNAMURTHY, S.; STANKOVIC, J. A.-*Models and solutions for radio irregularity in wireless sensor networks*. ACM Transactions on Sensor Networks (TOSN), vol. 2, issue 2, pp. 221-262. 2006.
- [77] *ZigBee Alliance Website*. Online at: <http://www.zigbee.org/en/index.asp>.

# Appendix A. MICAz's schematics and detailed information

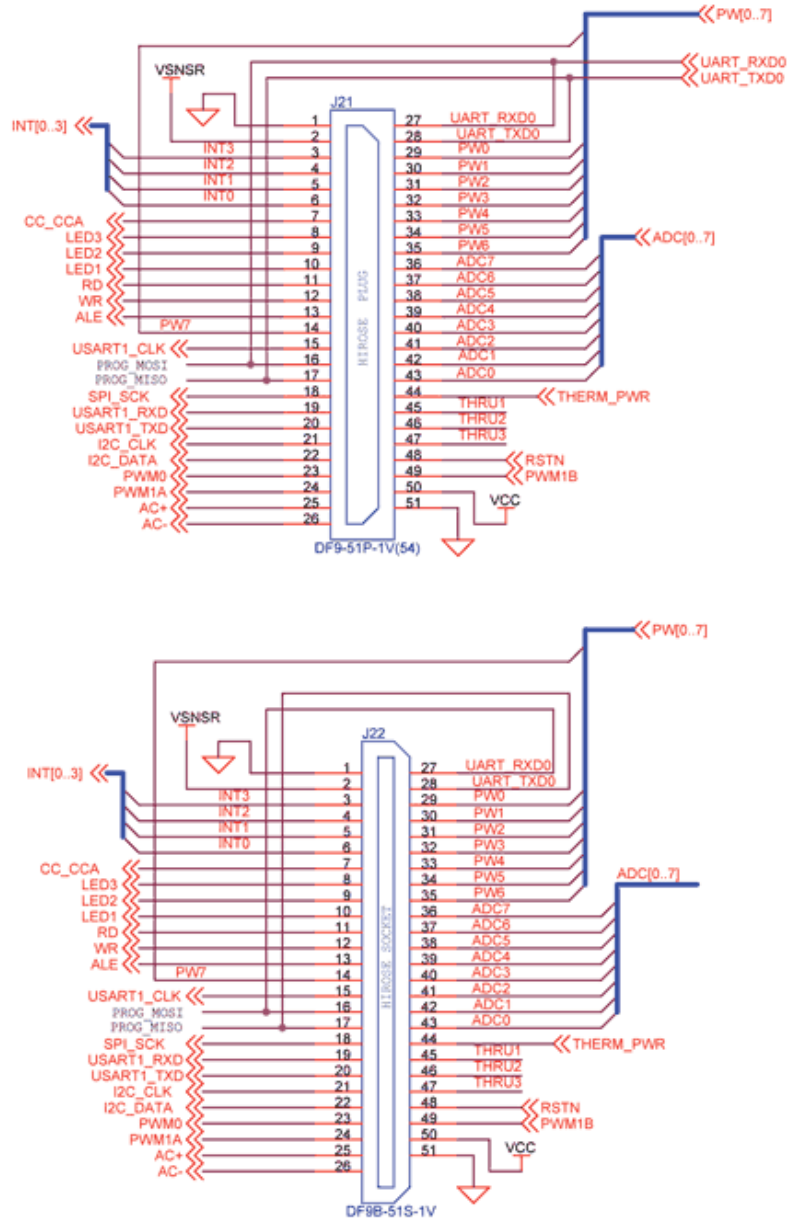


Figure 69 MICAz's 51 pins expansion connector schematic [20]



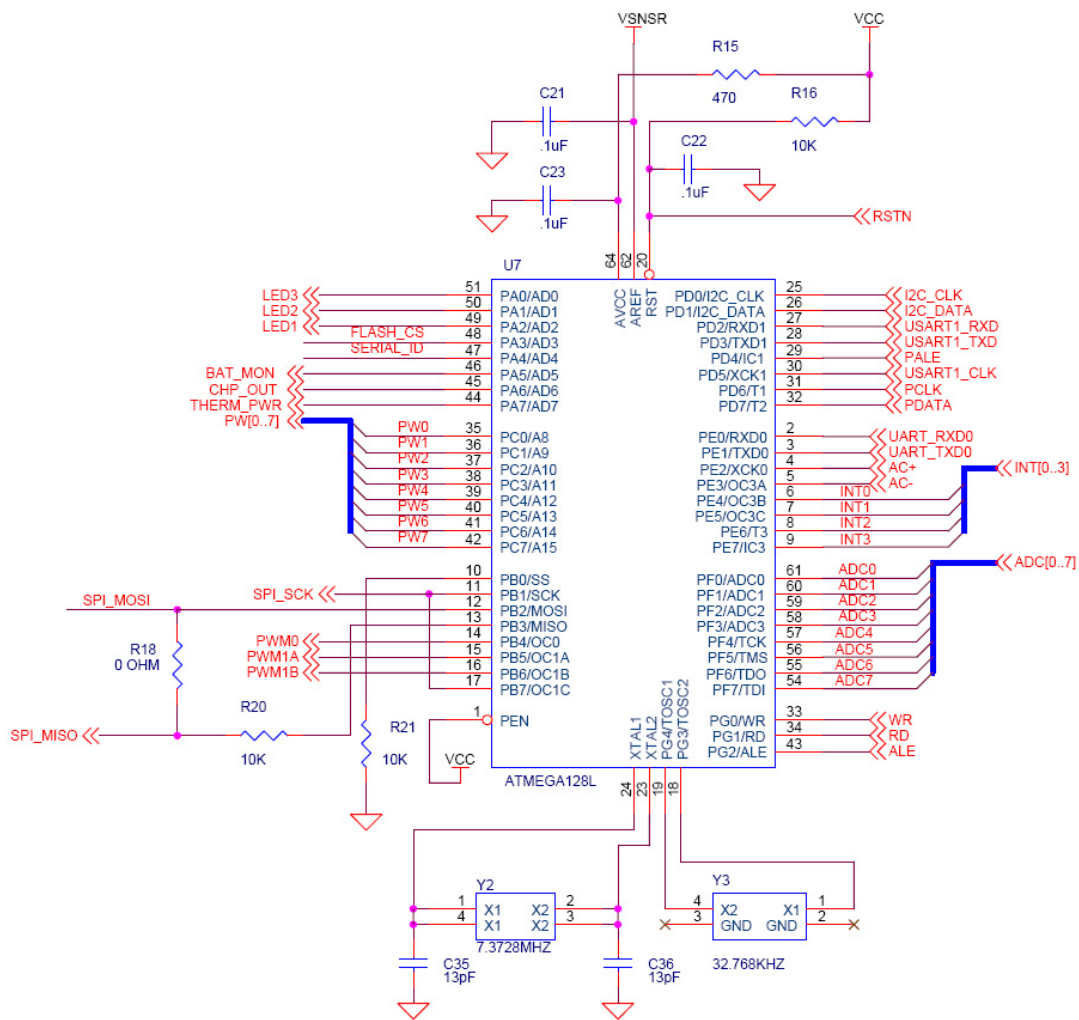


Figure 70 MICAz's ATmega128 configuration schematic [20]

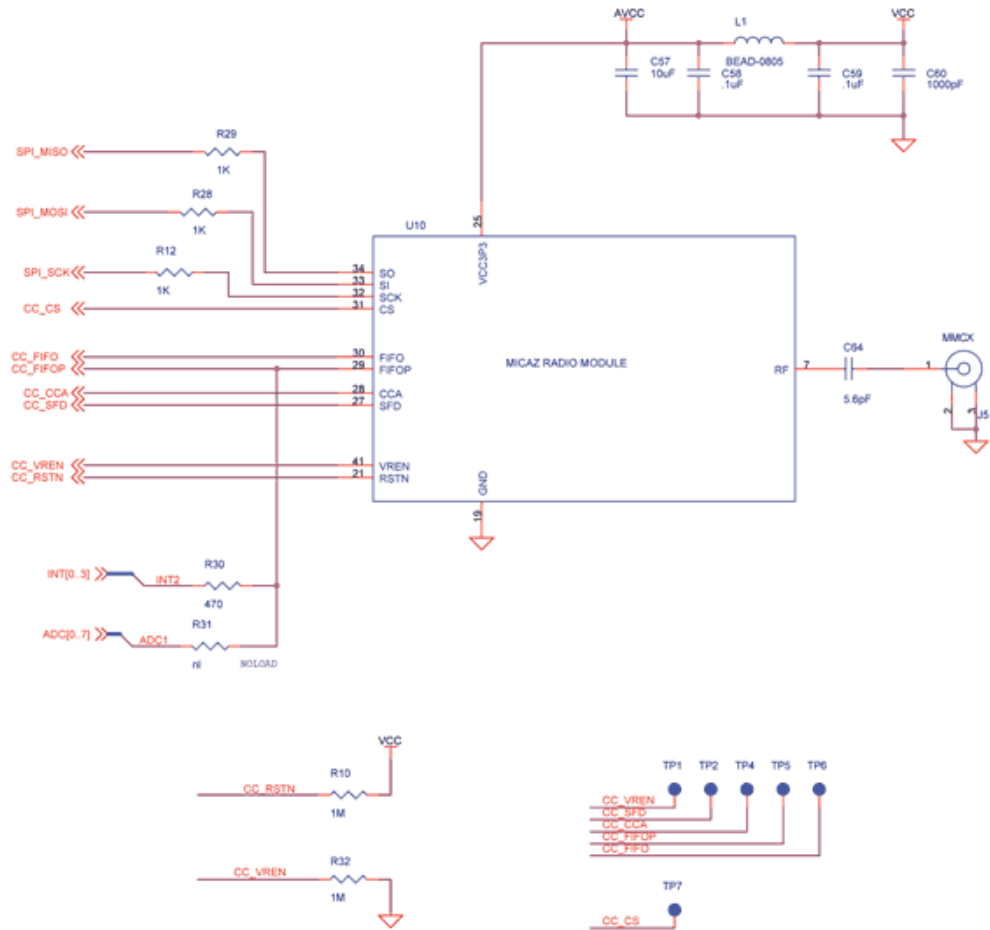


Figure 71 MICAZ's CC2420 radio transceiver configuration schematic [20]

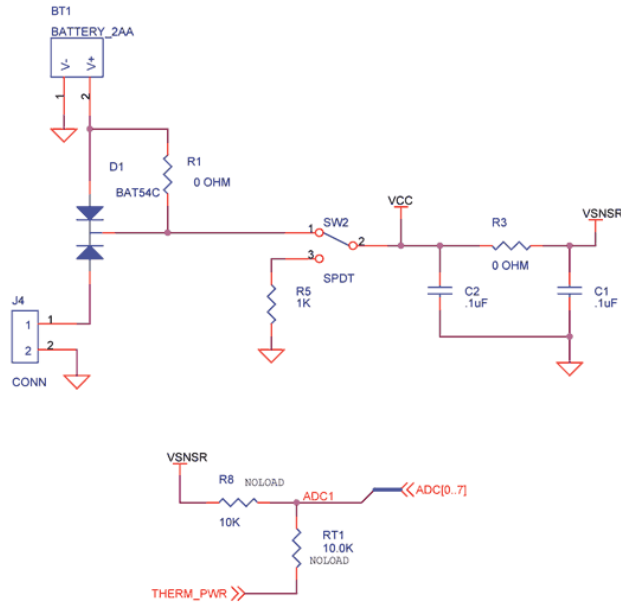


Figure 72 MICAz's battery connections, ON/OFF button and ADC1 schematics [20]

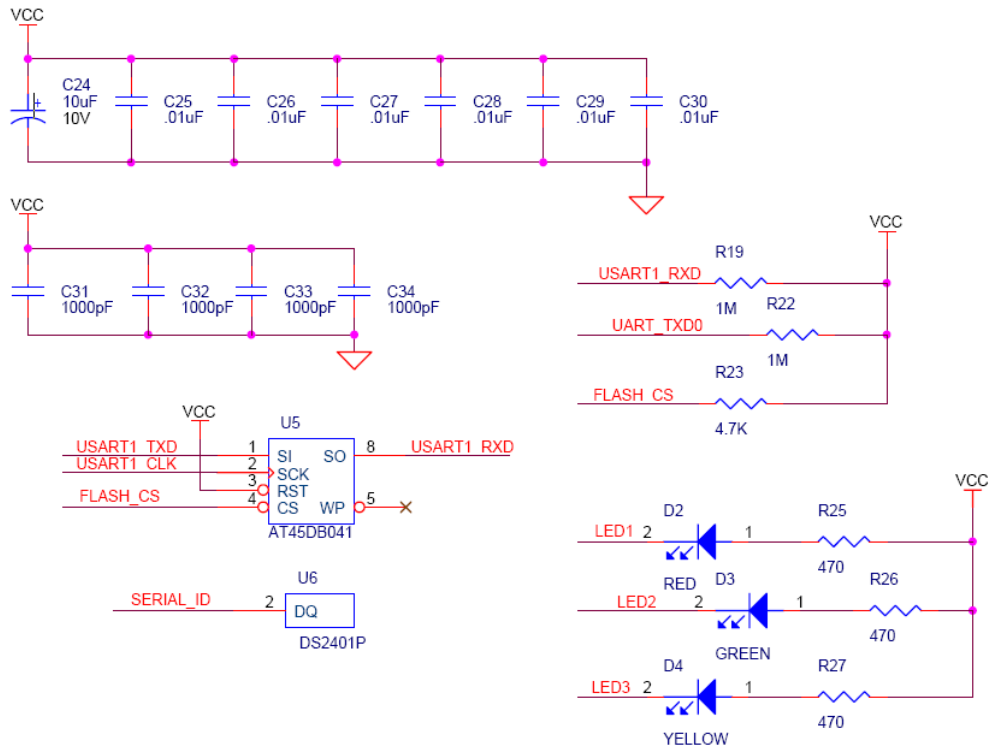


Figure 73 MICAz's flash memory, serial ID, LEDs and USART, configuration schematics [20]

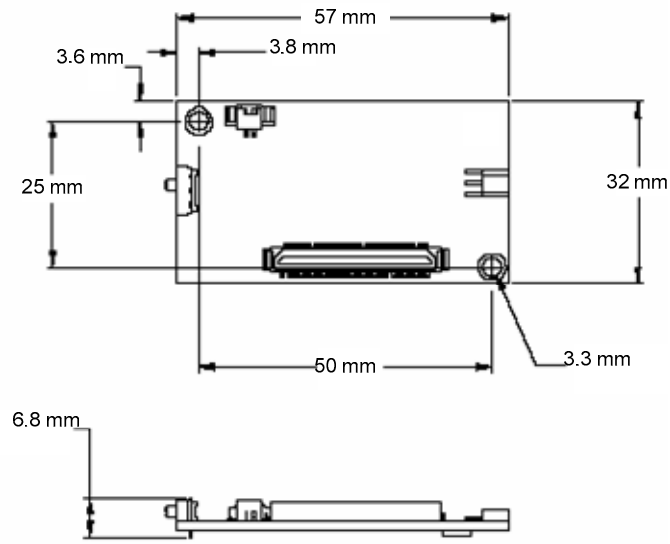


Figure 74 MICAz's dimensions [20]

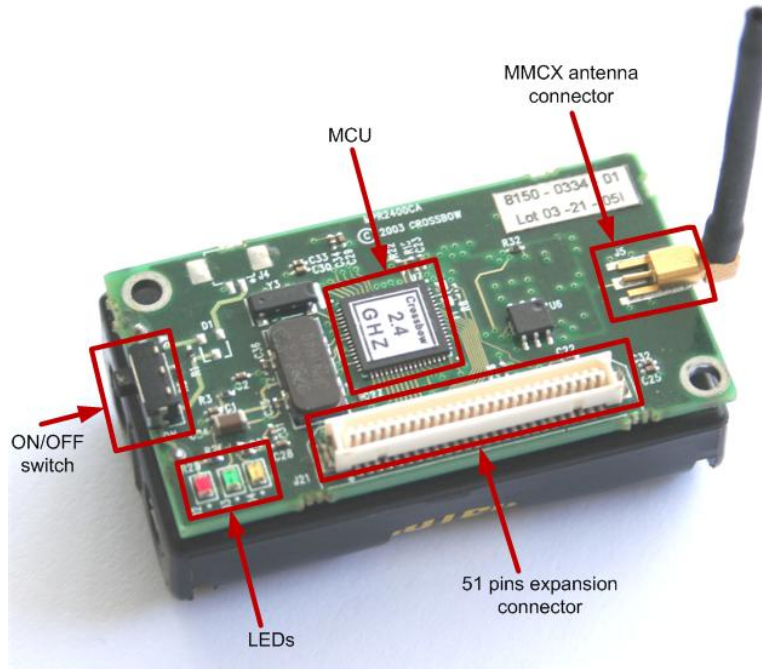


Figure 75 MICAz's main components (top layer only)



# Appendix B. CMU-FireFly's schematic and detailed information

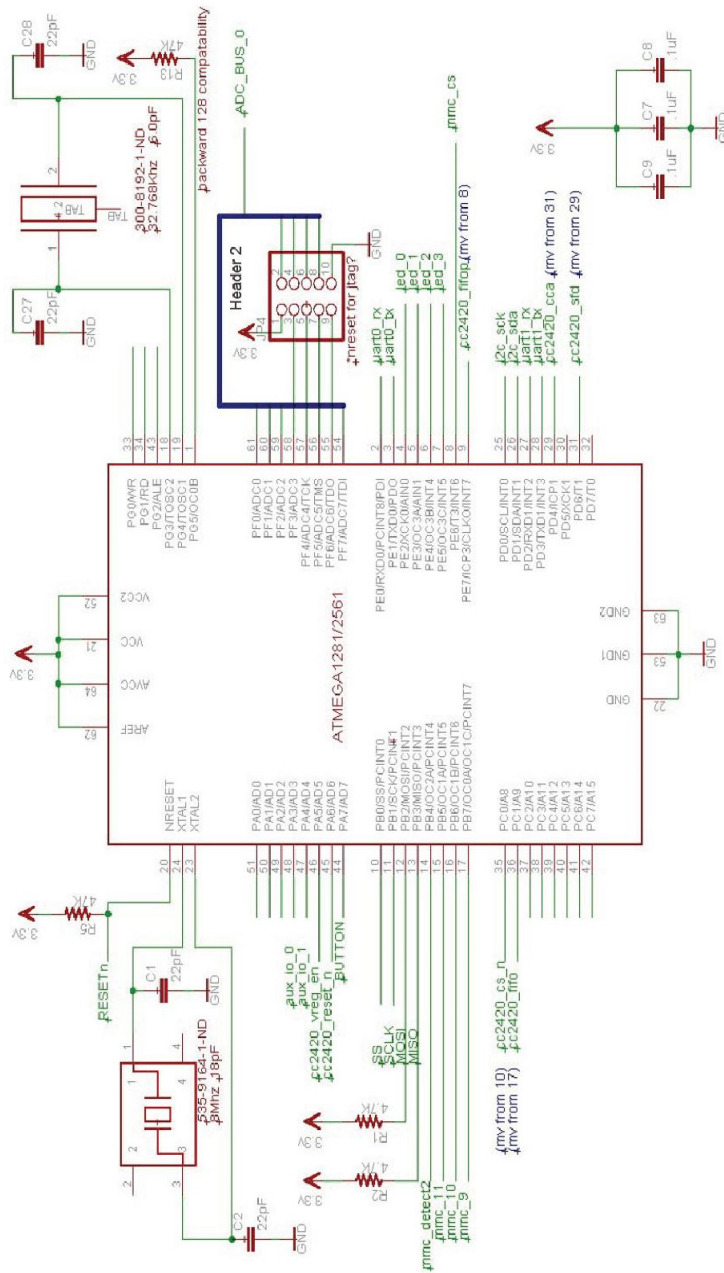


Figure 76 CMU-FF's ATmega128L configuration and 10 pin expansion connector, for ADC purposes (Header 2), schematic [57]

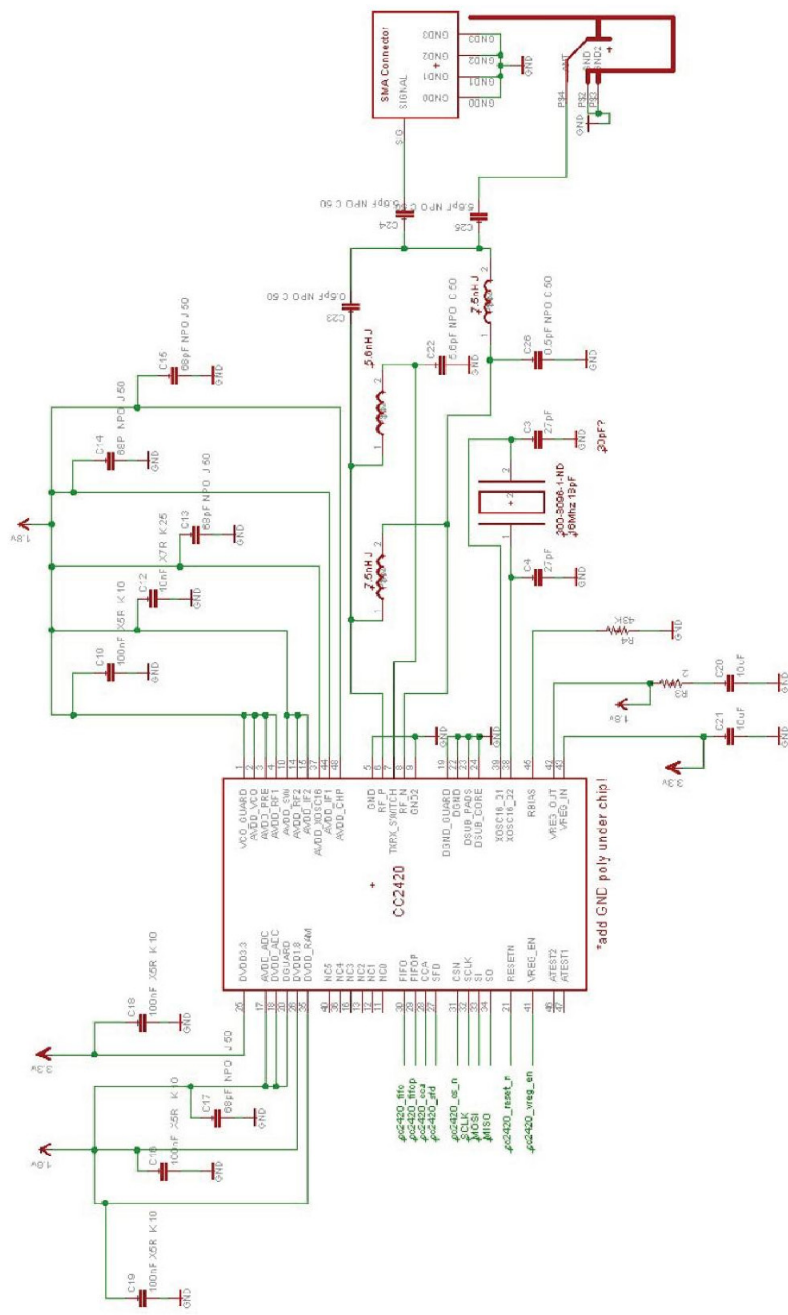


Figure 77 CMU-FF's CC2420 radio transceiver configuration schematic [57]

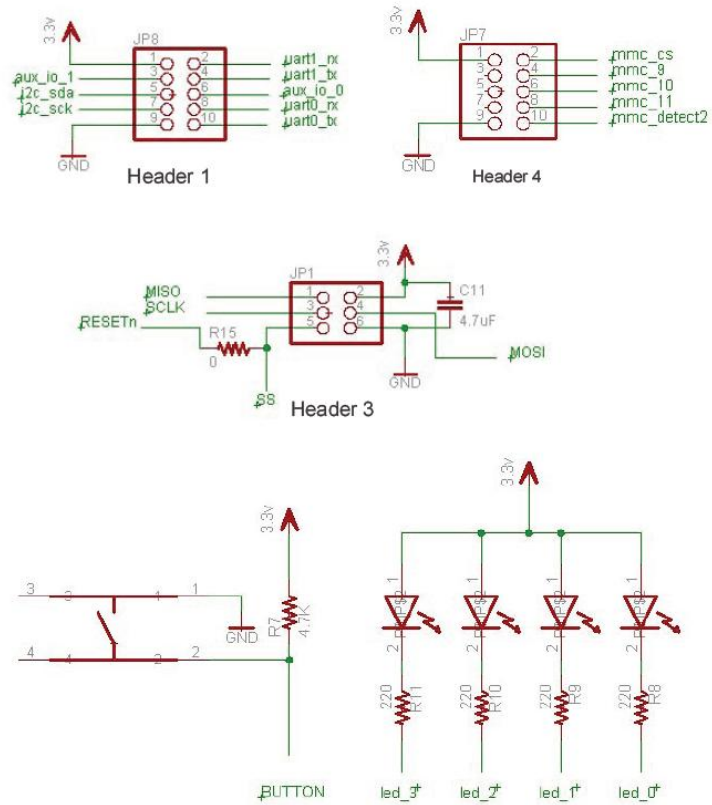


Figure 78 CMU-FF's expansion connectors for UART (Header 1), ISP programming (Header 3), mini-SD memory card or GPIO (Header 4), push-button and LEDs schematics [57]

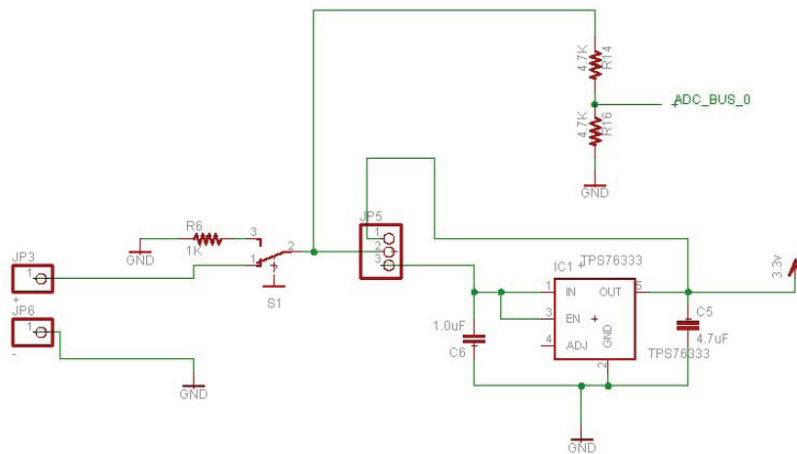


Figure 79 CMU\_FF's battery connections, voltage regulator, ON/OFF button and ADC\_BUS\_0 schematics [57]



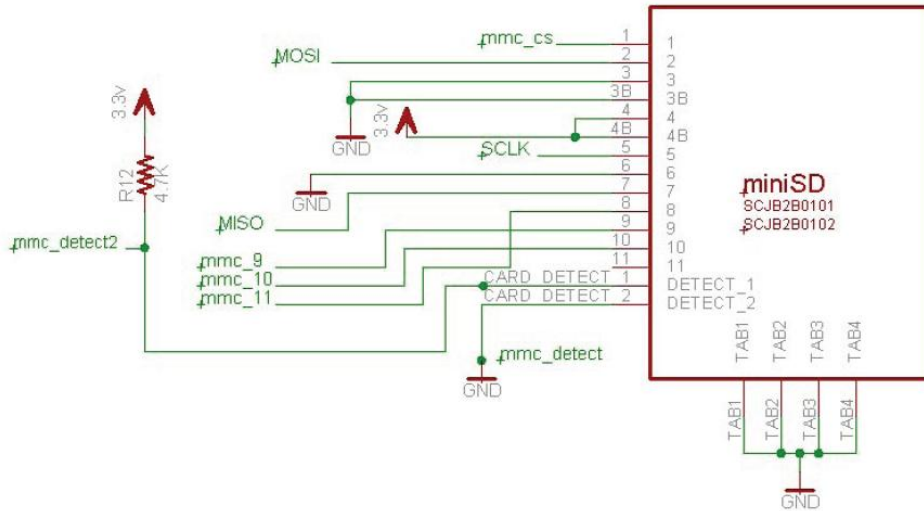


Figure 80 CMU-FF's mini-SD card slot schematic [57]

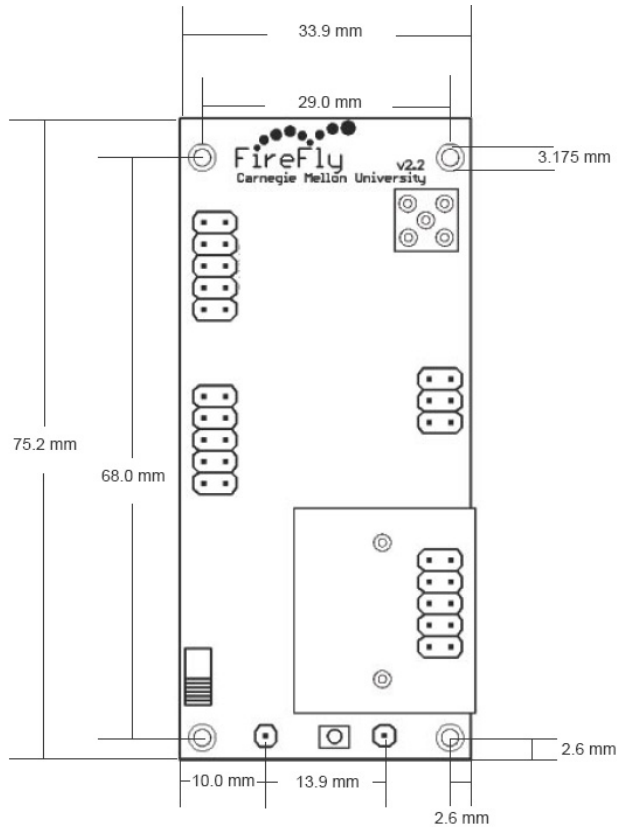


Figure 81 CMU-FF's dimensions [57]

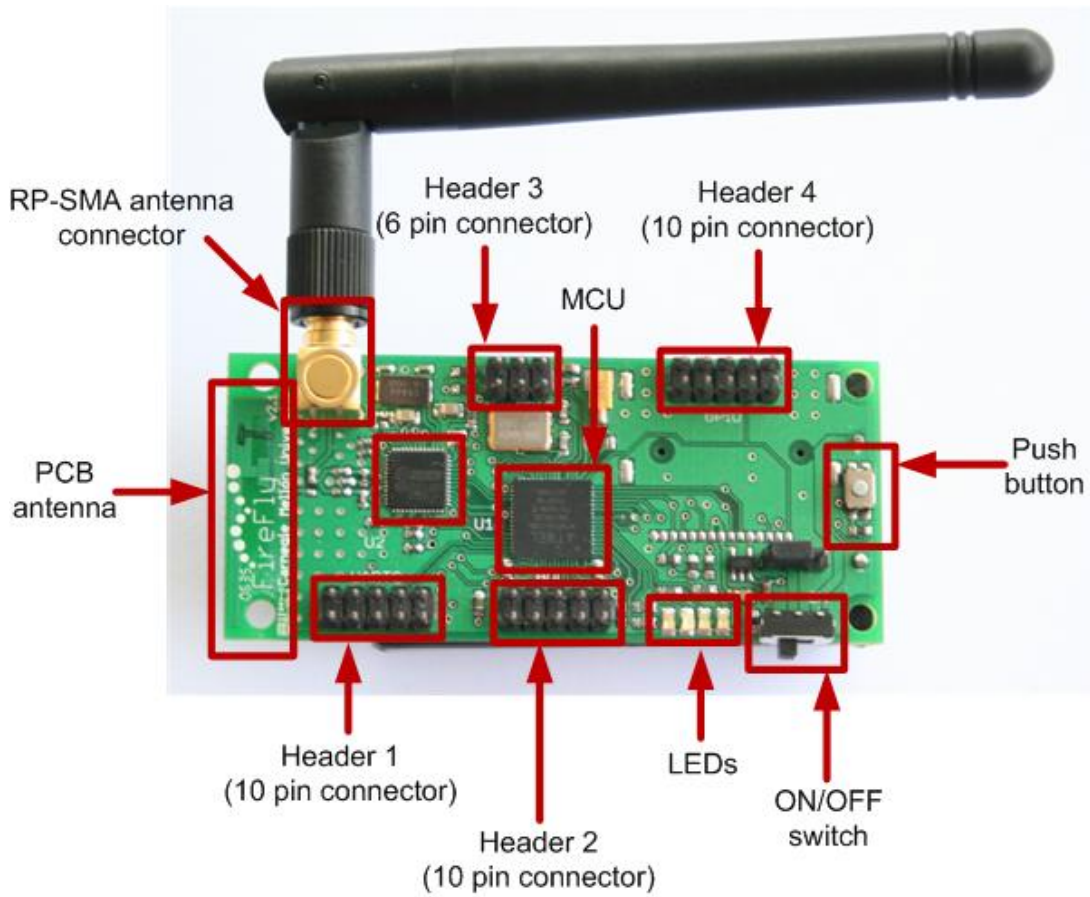


Figure 82 CMU-FF's v2.1, without mini-SD card slot, main components overview



## Appendix C. Receiving radio devices' digital output

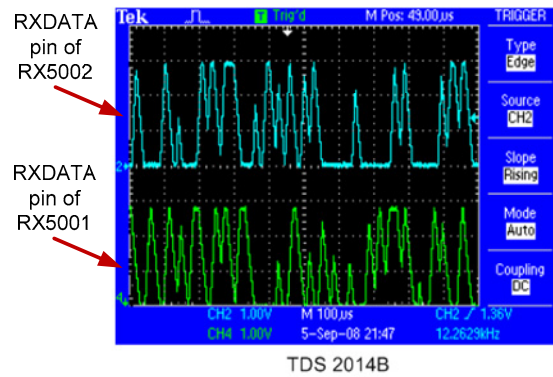


Figure 83 Instance of the RXDATA pin of the RX5001 (green) and RX5002 (blue) radio devices

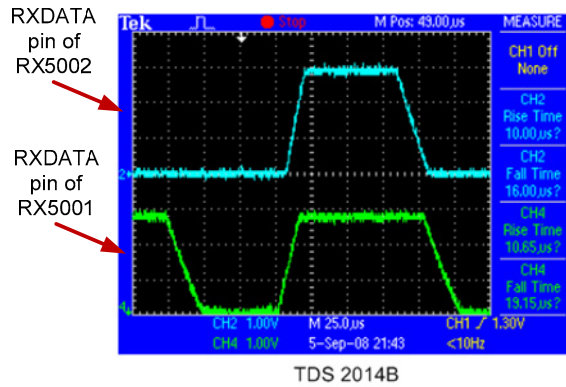


Figure 84 Response of the RXDATA pin to carrier wave transmission.



## Appendix D. Radio devices' external circuitry dimensioning

Table 7 Fundamental components for TX5002 OOK configuration

Component	Value	Observations	Description
$C_{DCB}$	4.7 $\mu$ F	Polarized; Tantalum type, 10 % tolerance	Bypass capacitor; Power supply ripple attenuation
$C_{RFB1}$	100 pF	High frequency NPO type, 5 % tolerance	RF bypass capacitor
$C_{RFB2}$	100 pF	High frequency NPO type, 5 % tolerance	RF bypass capacitor
$L_{AT}$	56 nH	$Q > 50$	Matching coil for antennas with impedance within 35 $\Omega$ to 72 $\Omega$
$L_{ESD}$	220 nH	Resistance < 0.1 $\Omega$	DC path from RFIO to GND for ESD protection
$L_{RFB}$	RF-bead	Fair-Rite 2506033017YO or equivalent	RF decoupling coil; Eliminates the possibility of RF feedback from antenna to pin 2 (positive supply voltage pin for the output amplifier and the transmitter base-band circuitry)
$R_{TXM}$	8.2 k $\Omega$	5 % tolerance	Output RF power limiting resistor

Table 8 **Fundamental components for RX5002 OOK configuration**

Component	Value	Observations	Description
C <sub>DCB</sub>	4.7 $\mu$ F	Polarized; Tantalum type, 10 % tolerance	Bypass capacitor; Power supply ripple attenuation
C <sub>BBO</sub>	3.3 nF	Ceramic; 10 % tolerance	Coupling capacitor for internal data slicer operation
C <sub>RFB1</sub>	100 pF	High frequency NPO type, 5 % tolerance	RF bypass capacitor
C <sub>RFB2</sub>	100 pF	High frequency NPO type, 5 % tolerance	RF bypass capacitor
L <sub>AT</sub>	56 nH	Q > 50	Matching coil for antennas with impedance within 35 $\Omega$ to 72 $\Omega$
L <sub>ESD</sub>	220 nH	Resistance < 0.1 $\Omega$	DC path from RFIO to GND for ESD protection
R <sub>LPF</sub>	56 k $\Omega$	5 % tolerance	Low-pass filter bandwidth adjust
R <sub>PR</sub>	330 k $\Omega$	5 % tolerance	Set the interval time between a falling edge of an on pulse to first RF amplifier and the rising edge of the next on pulse
R <sub>PW</sub>	270 k $\Omega$	5 % tolerance	Set the duration of the on pulse applied to the first RF amplifier
R <sub>REF</sub>	100 k $\Omega$	1 % tolerance	External reference resistor
R <sub>TH1</sub>	22 k $\Omega$	1 % tolerance	Set the threshold for the first data slicer

Table 9 **Fundamental components for RX5001 OOK configuration**

Component	Value	Observations	Description
$C_{DCB}$	4.7 $\mu$ F	Polarized; Tantalum type, 10 % tolerance	Bypass capacitor; Power supply ripple attenuation
$C_{BBO}$	3.9 nF	Ceramic; 10 % tolerance	Coupling capacitor for internal data slicer operation
$C_{RFB1}$	100 pF	High frequency NPO type, 5 % tolerance	RF bypass capacitor
$C_{RFB2}$	100 pF	High frequency NPO type, 5 % tolerance	RF bypass capacitor
$C_{PKD}$	1 nF	Ceramic; 10 % tolerance	Coupling capacitor for internal data slicer operation
$L_{AT}$	82 nH	$Q > 50$	Matching coil for antennas with impedance within 35 $\Omega$ to 72 $\Omega$
$L_{ESD}$	33 nH	Resistance < 0.1 $\Omega$	DC path from RFIO to GND for ESD protection
$R_{LPF}$	56 k $\Omega$	5 % tolerance	Low-pass filter bandwidth adjust
$R_{PR}$	330 k $\Omega$	5 % tolerance	Set the interval time between a falling edge of an on pulse to first RF amplifier and the rising edge of the next on pulse
$R_{PW}$	270 k $\Omega$	5 % tolerance	Set the duration of the on pulse applied to the first RF amplifier
$R_{REF}$	100 k $\Omega$	1 % tolerance	External reference resistor
$R_{TH1}$	22 k $\Omega$	1 % tolerance	Set the threshold for the first data slicer
$R_{TH2}$	100 k $\Omega$	1 % tolerance	Set the threshold for the second data slicer



Table 10 **Fundamental components for TX5001 OOK configuration**

Component	Value	Observations	Description
$C_{DCB}$	4.7 $\mu$ F	Polarized; Tantalum type, 10 % tolerance	Bypass capacitor; Power supply ripple attenuation
$C_{RFB1}$	100 pF	High frequency NPO type, 5 % tolerance	RF bypass capacitor
$C_{RFB2}$	100 pF	High frequency NPO type, 5 % tolerance	RF bypass capacitor
$L_{AT}$	82 nH	$Q > 50$	Matching coil for antennas with impedance within 35 $\Omega$ to 72 $\Omega$
$L_{ESD}$	33 nH	Resistance < 0.1 $\Omega$	DC path from RFIO to GND for ESD protection
$L_{RFB}$	RF-bead	Fair-Rite 2506033017YO or equivalent	RF decoupling coil; Eliminates the possibility of RF feedback from antenna to pin 2 (positive supply voltage pin for the output amplifier and the transmitter base-band circuitry)
$R_{TXM}$	8.2 k $\Omega$	5 % tolerance	Output RF power limiting resistor

## Appendix E. WiFLEX\_main and WiFLEX\_txsync boards hardware mapping

Table 11 WiFLEX\_main board hardware mapping

Label	Port	Pin number	Description
MOSI	B.3	15	SPI Master Out – Slave In
MISO	B.4	16	SPI Master In – Slave Out
SCK	B.5	17	SPI serial clock
UART_RXD	D.0	30	UART receive data line
UART_TXD	D.1	31	UART transmit data line
TX418_CNTRL	B.1	13	CNTRL0 line of the TX5002 transmitter; “0” – sleep mode, “1” – active in OOK mode
TX418_DATA	C.0	23	TXMOD line of the TX5002 transmitter; “0” – carrier off, “1” – carrier off
RX418_CNTRL	C.2	25	CNTRL0 and CNTRL1 lines of the RX5002 receiver; “0” – sleep mode, “1” – active mode
RX418_DATA	C.3	26	RXDATA line of the RX5002 receiver; “0” – no carrier sensed, “1” – carrier sensed
RX315_CNTRL	C.4	27	CNTRL0 and CNTRL1 lines of the RX5001 receiver; “0” – sleep mode, “1” – active mode
RX315_DATA	C.5	28	RXDATA line of the RX5001 receiver; “0” – no carrier sensed, “1” – carrier sensed
SW_CNTRL	C.1	24	CTRL line of the HF switch; “0” – antenna path set to RX5002 input, “1” – antenna path set to TX5002
LED	D.5	9	Controls the LED; “1” – LED on, “0” – LED off
FF_GPIO1	D.2	32	Connected to port B.5 of CMU-FF
FF_GPIO2	D.3	1	Connected to port B.4 of CMU-FF
FF_GPIO3	D.4	2	Connected to port B.6 of CMU-FF
MZ_GPIO1	B.0	12	Connected to port B.5 of MICAz
MZ_GPIO2	D.6	10	Connected to port E.6 of MICAz
MZ_GPIO3	D.7	11	Connected to port B.4 of MICAz

Table 12 **WiFLEX\_txsync board hardware mapping**

Label	Port	Pin number	Description
MOSI	B.3	15	SPI Master Out – Slave In
MISO	B.4	16	SPI Master In – Slave Out
SCK	B.5	17	SPI serial clock
UART_RXD	D.0	30	UART receive data line
UART_TXD	D.1	31	UART transmit data line
TX315_CNTRL	C.0	23	CNTRL0 line of the TX5001 transmitter; “0” – sleep mode, “1” – active in OOK mode
TX315_DATA	C.2	25	TXMOD line of the TX5001 transmitter; “0” – carrier off, “1” – carrier on
LED	D.5	9	Controls the LED; “1” – LED on, “0” – LED off
GPIO1	D.2	32	GPIO 1
GPIO2	D.3	1	GPIO 2
GPIO3	D.4	2	GPIO 3

## Appendix F. Determining pulse width and switching time experimental results

Table 13 **Winning node alone**

Silence time ( $\mu\text{s}$ ) \backslash Pulse width ( $\mu\text{s}$ )	40	45	50	55	60	65	70	75	80	85
	30	0	953	1000	1000	1000	1000	1000	1000	1000
35	0	233	1000	1000	1000	1000	1000	999	1000	1000
40	0	0	1000	1000	1000	1000	1000	1000	1000	1000
45	0	0	1000	1000	1000	1000	1000	1000	1000	1000
50	0	0	1000	1000	1000	1000	1000	1000	1000	1000
55	0	0	2	1000	1000	1000	1000	1000	1000	1000
60	0	0	382	1000	1000	1000	1000	1000	1000	1000
65	0	0	0	1000	1000	1000	1000	1000	1000	1000
70	0	0	0	1000	1000	1000	1000	1000	1000	1000
75	0	0	0	1000	1000	1000	1000	1000	1000	1000

Table 14 **Losing node alone**

Silence time ( $\mu\text{s}$ ) \backslash Pulse width ( $\mu\text{s}$ )	40	45	50	55	60	65	70	75	80	85
	30	0	1000	1000	1000	1000	1000	1000	1000	1000
35	0	804	1000	1000	1000	1000	1000	1000	1000	1000
40	0	0	1000	1000	1000	1000	1000	1000	1000	1000
45	0	0	1000	1000	1000	1000	1000	1000	1000	1000
50	0	0	1000	1000	1000	1000	1000	1000	1000	1000
55	0	0	956	1000	1000	1000	1000	1000	999	999
60	0	0	1000	1000	1000	1000	1000	1000	1000	1000
65	0	0	2	1000	1000	1000	1000	1000	1000	1000
70	0	0	982	1000	1000	1000	1000	1000	1000	1000
75	0	0	0	1000	1000	1000	1000	1000	1000	1000

Table 15 **Winning node at 0 m distance**

Silence time ( $\mu\text{s}$ ) \backslash Pulse width ( $\mu\text{s}$ )	40	45	50	55	60	65	70	75	80	85
	30	0	988	1000	1000	1000	1000	1000	1000	1000
35	0	47	1000	1000	1000	1000	1000	1000	1000	1000
40	0	0	1000	1000	1000	1000	1000	1000	1000	1000
45	0	0	637	1000	1000	1000	1000	1000	1000	1000
50	0	0	744	1000	1000	1000	1000	1000	1000	1000
55	0	0	1000	1000	1000	1000	1000	1000	1000	1000
60	0	0	19	1000	1000	1000	1000	1000	1000	1000
65	0	0	1000	1000	1000	1000	1000	1000	1000	1000
70	0	0	1000	1000	1000	1000	1000	1000	1000	1000
75	0	0	1000	1000	1000	1000	1000	1000	1000	1000

Table 16 **Losing node at 0 m distance**

Silence time ( $\mu\text{s}$ ) \backslash Pulse width ( $\mu\text{s}$ )	40	45	50	55	60	65	70	75	80	85
	30	16	12	0	0	0	0	0	0	0
35	0	385	0	0	0	0	0	0	0	0
40	0	142	0	0	0	0	0	0	0	0
45	0	0	329	0	0	0	0	0	0	0
50	0	0	251	0	0	0	0	0	0	0
55	0	0	406	0	0	0	0	0	0	0
60	0	0	505	0	0	0	0	0	0	0
65	0	0	230	0	0	0	0	0	0	0
70	0	0	397	0	0	0	0	0	0	0
75	0	0	32	0	0	0	0	0	0	0

Table 17 **Winning node at 1 m distance**

Silence time ( $\mu\text{s}$ ) \n Pulse width ( $\mu\text{s}$ )										
	40	45	50	55	60	65	70	75	80	85
30	0	999	1000	1000	1000	1000	1000	1000	1000	1000
35	0	0	1000	1000	1000	1000	1000	1000	1000	1000
40	0	0	1000	1000	1000	1000	1000	1000	1000	1000
45	0	0	545	1000	1000	1000	1000	1000	1000	1000
50	0	0	771	1000	1000	1000	1000	1000	1000	1000
55	0	0	0	1000	1000	1000	1000	1000	1000	1000
60	0	0	0	1000	1000	1000	1000	1000	1000	1000
65	0	0	0	997	1000	1000	1000	1000	1000	1000
70	0	0	1	1000	1000	1000	1000	1000	1000	1000
75	0	0	0	1000	1000	1000	1000	1000	1000	1000

Table 18 **Losing node at 1 m distance**

Silence time ( $\mu\text{s}$ ) \n Pulse width ( $\mu\text{s}$ )										
	40	45	50	55	60	65	70	75	80	85
30	0	1	0	0	0	0	0	0	0	0
35	0	855	0	0	0	0	0	0	0	0
40	0	0	0	0	0	0	0	0	0	0
45	0	0	455	0	0	0	0	0	0	0
50	0	0	229	0	0	0	0	0	0	0
55	0	0	830	0	0	0	0	0	0	0
60	0	0	999	0	0	0	0	0	0	0
65	0	0	3	6	0	0	0	0	0	0
70	0	0	795	0	0	0	0	0	0	0
75	0	0	0	0	0	0	0	0	0	0

Table 19 **Winning node at 5 m distance**

Silence time ( $\mu\text{s}$ ) \n Pulse width ( $\mu\text{s}$ )										
	40	45	50	55	60	65	70	75	80	85
30	13	622	956	1000	997	999	1000	999	999	1000
35	0	992	995	1000	1000	1000	1000	999	987	979
40	1	154	1000	1000	1000	1000	1000	1000	1000	1000
45	0	0	1000	1000	1000	1000	1000	1000	1000	1000
50	0	0	1000	1000	1000	1000	1000	1000	1000	1000
55	0	0	999	1000	1000	1000	1000	1000	1000	1000
60	0	0	1000	1000	1000	1000	1000	1000	1000	1000
65	0	0	617	1000	1000	1000	1000	1000	1000	1000
70	0	0	1000	1000	1000	1000	1000	1000	1000	1000
75	0	0	85	1000	1000	1000	1000	1000	1000	1000

Table 20 **Losing node at 5 m distance**

Silence time ( $\mu\text{s}$ ) \n Pulse width ( $\mu\text{s}$ )										
	40	45	50	55	60	65	70	75	80	85
30	0	378	44	0	2	1	0	1	1	0
35	0	5	5	0	0	0	0	1	5	0
40	1	0	0	0	0	0	0	0	0	0
45	0	0	0	0	0	0	0	0	0	0
50	0	0	0	0	0	0	0	0	0	0
55	0	0	1	0	0	0	0	0	0	0
60	0	0	0	0	0	0	0	0	0	0
65	0	0	0	0	0	0	0	0	0	0
70	0	0	0	0	0	0	0	0	0	0
75	0	0	0	0	0	0	0	0	0	0

Table 21 **Winning node at 10 m distance**

Silence time ( $\mu\text{s}$ ) \n Pulse width ( $\mu\text{s}$ )										
	40	45	50	55	60	65	70	75	80	85
30	100	369	893	999	1000	999	998	1000	1000	1000
35	0	957	987	1000	1000	1000	999	998	1000	1000
40	0	743	1000	999	1000	1000	1000	999	1000	1000
45	0	0	1000	1000	1000	1000	1000	1000	1000	1000
50	0	0	1000	1000	1000	1000	1000	1000	1000	1000
55	0	0	1000	1000	1000	1000	999	1000	1000	1000
60	0	0	1000	1000	1000	1000	1000	1000	1000	1000
65	0	0	996	1000	1000	1000	1000	1000	1000	1000
70	0	0	1000	1000	1000	1000	1000	1000	1000	1000
75	0	0	966	1000	1000	1000	1000	1000	1000	1000

Table 22 **Losing node at 10 m distance**

Silence time ( $\mu\text{s}$ ) \n Pulse width ( $\mu\text{s}$ )										
	40	45	50	55	60	65	70	75	80	85
30	0	627	107	1	0	1	2	0	0	0
35	0	43	13	0	0	0	1	2	0	0
40	0	12	0	1	0	0	0	1	0	0
45	0	0	0	0	0	0	0	0	0	0
50	0	0	0	0	0	0	0	0	0	0
55	0	0	0	0	0	0	1	0	0	0
60	0	0	0	0	0	0	0	0	0	0
65	0	0	0	0	0	0	0	0	0	0
70	0	0	0	0	0	0	0	0	0	0
75	0	0	0	0	0	0	0	0	0	0



Table 23 **Winning node at 15 m distance**

Silence time ( $\mu\text{s}$ ) \backslash Pulse width ( $\mu\text{s}$ )	40	45	50	55	60	65	70	75	80	85
	30	102	276	852	985	1000	994	998	1000	995
35	0	881	952	1000	1000	1000	998	1000	1000	1000
40	0	969	1000	1000	1000	1000	1000	1000	1000	994
45	0	37	998	1000	998	987	997	1000	1000	1000
50	0	0	1000	1000	1000	1000	1000	1000	1000	1000
55	0	0	1000	1000	1000	1000	1000	1000	1000	1000
60	0	0	1000	1000	1000	1000	1000	1000	1000	1000
65	0	0	1000	1000	1000	1000	1000	1000	1000	1000
70	0	0	1000	1000	1000	1000	1000	1000	1000	1000
75	0	0	1000	1000	1000	1000	1000	1000	1000	1000

Table 24 **Losing node at 15 m distance**

Silence time ( $\mu\text{s}$ ) \backslash Pulse width ( $\mu\text{s}$ )	40	45	50	55	60	65	70	75	80	85
	30	150	714	128	5	0	6	2	0	5
35	0	119	48	0	0	0	2	0	0	0
40	0	29	0	0	0	0	0	0	0	3
45	0	8	2	0	1	7	3	0	0	0
50	0	0	0	0	0	0	0	0	0	0
55	0	0	0	0	0	0	0	0	0	0
60	0	0	0	0	0	0	0	0	0	0
65	0	0	0	0	0	0	0	0	0	0
70	0	0	0	0	0	0	0	0	0	0
75	0	0	0	0	0	0	0	0	0	0

Table 25 **Winning node at 20 m distance**

Silence time ( $\mu\text{s}$ ) \backslash Pulse width ( $\mu\text{s}$ )	40	45	50	55	60	65	70	75	80	85
	30	59	163	789	993	995	996	1000	1000	998
35	0	774	902	1000	1000	999	1000	988	987	1000
40	0	991	1000	999	996	1000	1000	1000	1000	1000
45	0	294	998	1000	1000	1000	1000	1000	1000	1000
50	0	0	1000	1000	1000	1000	1000	1000	1000	1000
55	0	0	1000	1000	998	999	1000	985	1000	1000
60	0	0	1000	997	1000	1000	1000	1000	1000	1000
65	0	16	1000	1000	1000	1000	1000	1000	1000	999
70	0	0	1000	1000	1000	1000	1000	1000	1000	1000
75	0	0	1000	1000	1000	1000	1000	1000	1000	1000

Table 26 **Losing node at 20 m distance**

Silence time ( $\mu\text{s}$ ) \backslash Pulse width ( $\mu\text{s}$ )	40	45	50	55	60	65	70	75	80	85
	30	7	883	199	7	5	4	0	0	2
35	0	226	98	0	0	1	0	11	4	0
40	0	6	0	1	3	0	0	0	0	0
45	0	0	2	0	0	0	0	0	0	0
50	0	0	0	0	0	0	0	0	0	0
55	0	0	0	0	2	1	0	0	0	0
60	0	0	0	3	0	0	0	0	0	0
65	0	0	0	0	0	0	0	0	0	1
70	0	0	0	0	0	0	0	0	0	0
75	0	0	0	0	0	0	0	0	0	0

Table 27 Combined probability of winning node success from 0 m to 20 m

Silence time ( $\mu\text{s}$ ) \diagdown Pulse width ( $\mu\text{s}$ )	40	45	50	55	60	65	70	75	80	85
	30	3.914	62.429	92.714	99.671	99.886	99.829	99.943	99.986	99.886
35	0.000	55.486	97.657	100.000	100.000	99.986	99.957	99.771	99.629	99.700
40	0.014	40.814	100.000	99.971	99.943	100.000	100.000	99.986	100.000	99.914
45	0.000	4.729	88.257	100.000	99.971	99.814	99.957	100.000	100.000	100.000
50	0.000	0.000	93.071	100.000	100.000	100.000	100.000	100.000	100.000	100.000
55	0.000	0.000	71.443	100.000	99.971	99.986	99.986	99.786	100.000	100.000
60	0.000	0.000	62.871	99.957	100.000	100.000	100.000	100.000	100.000	100.000
65	0.000	0.229	65.900	99.957	100.000	100.000	100.000	100.000	100.000	99.986
70	0.000	0.000	71.443	100.000	100.000	100.000	100.000	100.000	100.000	100.000
75	0.000	0.000	57.871	100.000	100.000	100.000	100.000	100.000	100.000	100.000

Table 28 Combined probability of losing node success from 0 m to 20 m

Silence time ( $\mu\text{s}$ ) \diagdown Pulse width ( $\mu\text{s}$ )	40	45	50	55	60	65	70	75	80	85
	30	111.814	62.643	93.171	99.814	99.900	99.829	99.943	99.986	99.886
35	114.286	79.471	97.657	100.000	100.000	99.986	99.957	99.800	99.871	100.000
40	114.271	111.586	100.000	99.971	99.957	100.000	100.000	99.986	100.000	99.957
45	114.286	114.171	88.743	100.000	99.986	99.900	99.957	100.000	100.000	100.000
50	114.286	114.286	93.143	100.000	100.000	100.000	100.000	100.000	100.000	100.000
55	114.286	114.286	82.957	100.000	99.971	99.986	99.986	100.000	100.014	100.014
60	114.286	114.286	78.514	99.957	100.000	100.000	100.000	100.000	100.000	100.000
65	114.286	114.286	110.929	99.914	100.000	100.000	100.000	100.000	100.000	99.986
70	114.286	114.286	83.229	100.000	100.000	100.000	100.000	100.000	100.000	100.000
75	114.286	114.286	113.829	100.000	100.000	100.000	100.000	100.000	100.000	100.000

Table 29 Combined probability of both nodes success probability from 0 m to 20 m

Pulse width ( $\mu\text{s}$ ) \backslash Silence time ( $\mu\text{s}$ )										
	40	45	50	55	60	65	70	75	80	85
30	4.377	39.107	86.383	99.486	99.786	99.657	99.886	99.971	99.772	99.971
35	0.000	44.095	95.369	100.000	100.000	99.971	99.914	99.572	99.500	99.700
40	0.016	45.543	100.000	99.943	99.900	100.000	100.000	99.971	100.000	99.871
45	0.000	5.399	78.322	100.000	99.957	99.714	99.914	100.000	100.000	100.000
50	0.000	0.000	86.689	100.000	100.000	100.000	100.000	100.000	100.000	100.000
55	0.000	0.000	59.267	100.000	99.943	99.971	99.971	99.786	100.014	100.014
60	0.000	0.000	49.363	99.914	100.000	100.000	100.000	100.000	100.000	100.000
65	0.000	0.261	73.102	99.871	100.000	100.000	100.000	100.000	100.000	99.971
70	0.000	0.000	59.461	100.000	100.000	100.000	100.000	100.000	100.000	100.000
75	0.000	0.000	65.874	100.000	100.000	100.000	100.000	100.000	100.000	100.000



## Appendix G. Reliability versus distance experimental results

The first group was composed by nodes 1, 2, and 3, and the second group was composed by the remaining nodes.

Table 30 Failure to lose the tournament counting

Distance (m) \ Node	1	5	10	15	20
1	1	0	12	967	1152
2	0	6	9	821	1209
3	3	2	8	7	1765
4	0	12	16	189	23
5	5	2	26	54	142
6	2	3	5	3	5
7	0	7	4	20	104
8	6	3	17	58	52
9	4	11	9	79	18
10	1	3	13	68	119
Total	22	49	119	2266	4589
Error ratio	0.001467	0.003267	0.007933	0.151067	0.305933

Table 31 Failure to win the tournament counting

Distance (m) \ Node	1	5	10	15	20
1	0	1	0	104	192
2	0	2	0	30	330
3	0	0	4	102	27
4	1	0	5	0	4
5	3	0	0	1	392
6	0	0	1	1337	1680
7	3	1	7	7	1945
8	0	5	0	7	0
9	2	2	0	1	519
10	1	1	1	602	60
Total	10	12	18	2191	5149
Error ratio	0.000000	0.000800	0.001200	0.146067	0.343267

