PERFORMING AN EXPERIMENTAL PLATFORM TO OPTIMIZE DATA MULTIPLEXING

Remy Astier[1], Thierry Capitaine[1], Jerome Dubois[1], Valery Bourny[1], Aurelien Lorthois[1] and Jerome Fortin[1]

[1]Laboratoire des Technologies Innovantes (LTI, EA3899), INSSET, Saint-Quentin

ABSTRACT

This article is based on preliminary work on the OSI model management layers to optimized industrial wired data transfer on low data rate wireless technology. Our previous contribution deal with the development of a demonstrator providing CAN bus transfer frames (1Mbps) on a low rate wireless channel provided by Zigbee technology. In order to be compatible with all the other industrial protocols, we describe in this paper our contribution to design an innovative Wireless Device (WD) and a software tool, which will aim to determine the best architecture (hardware/software) and wireless technology to be used taking in account of the wired protocol requirements. To validate the proper functioning of this WD, we will develop an experimental platform to test different strategies provided by our software tool. We can consequently prove which is the best configuration (hardware/software) compared to the others by the inclusion (inputs) of the required parameters of the wired protocol (load, binary rate, acknowledge timeout) and the analysis of the WD architecture characteristics proposed (outputs) as the delay introduced by system, buffer size needed, CPU speed, power consumption, meeting the input requirement. It will be important to know whether gain comes from a hardware strategy with hardware accelerator e.g or a software strategy with a more performing scheduler. At the end, our experimental platform will be a tool for characterizing different WD.

KEYWORDS

Embedded system, Multiplexing, Decision Support Tool, Test bench, Networking management systems

1. INTRODUCTION

Our problematic is about a smart data transmission through a multiplexing channel (Figure 1). We choose to work with a wireless technology because today, wireless technologies are everywhere and multiplexing of wire data on a wireless channel is a major issue.

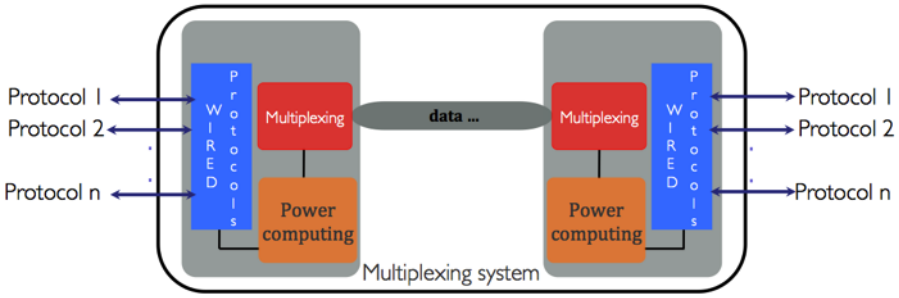

Figure 1. Multiplexing system representation

DOI : 10.5121/ijesa.2013.3401

This study is based on previous work [1] on wireless data transmission from 1*CAN Bus High Speed and 1*serial communication RS232. But due to the number n of possible protocol (RS232/422/485, CAN Bus, Ethernet) and their parameters (start/stop bit, data field, data rate, ID, type of frame), it is difficult to make 1 embedded system which integrates at all issues (Figure 2) which protocols priority? How to send data? It is necessary to consider this information.

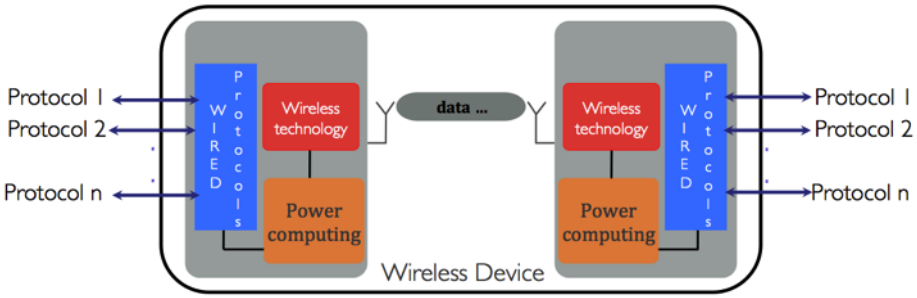

Figure 2. Wireless Device representation

In order to process these different protocols, we have defined 2 types of information:

- S*ervice Information* or *SI*: allow synchronization and correct transmission of information between 2 entities (with synchronization bit, control field like CRC, ACK, and upper layers of OSI Model).
- *Useful Information* or *MSG*: are useful data like CAN Bus ID, IP, Mac address, type of frame, field data... They must be sent with WD.

With this representation, we can work with any protocol. To illustrate this, we take Controller Area Network (CAN Bus) case [2]. Figure 3 shows a 2.0A and 2.0B CAN Bus frame with *SI* framed in blue on yellow background and *MSG* framed in red on grey background.

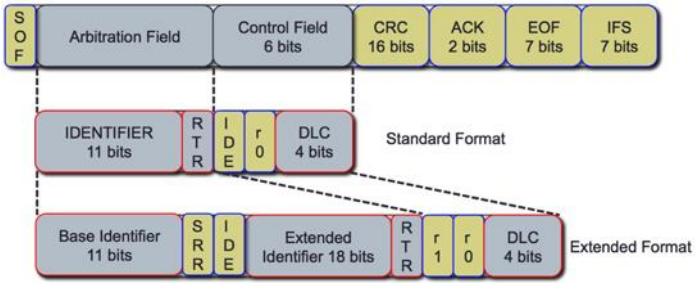

Figure 3. Highlights of SI and MSG fields

Now, with this generic representation, our WD can manage any I/O protocols as show in the Figure 4. We must transmit a *header*, here call "H", in order to identified source of data. Each data transmit has a *header*.

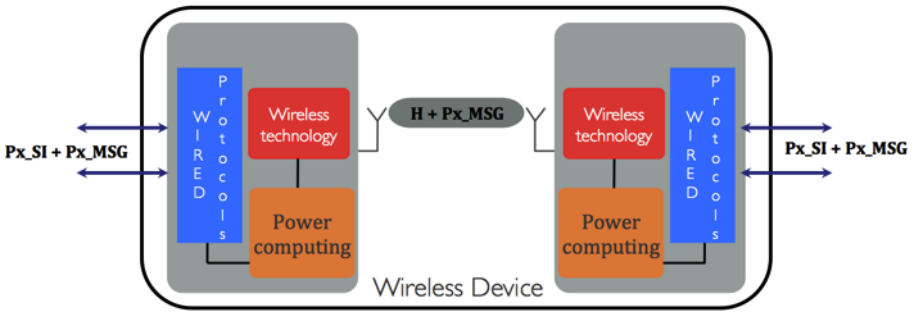

Figure 4. Wireless Device representation with SI and MSG fields

This study builds on preliminary work [1] which has enabled us to identify 4 locks (Figure 5):

- L1 in charge of auto-configuration, this means the automatic setting of protocols.
- L2 will ensure the extraction of useful messages (going back more or less high in OSI layers).
- L3 will handle data storage decoded from different protocols (multiplexing) their optimized coding (compression) and encapsulation within a frame that meets the requirements of the block L4 (Figure 4, "H + Px_MSG").
- L4 is responsible for the transmission of the previous frame in a wireless link. The wireless transmission channel must meet the following requirements:
  - High-speed data rate.
  - Real-time and full duplex.
  - Compatible with the industrial environment.

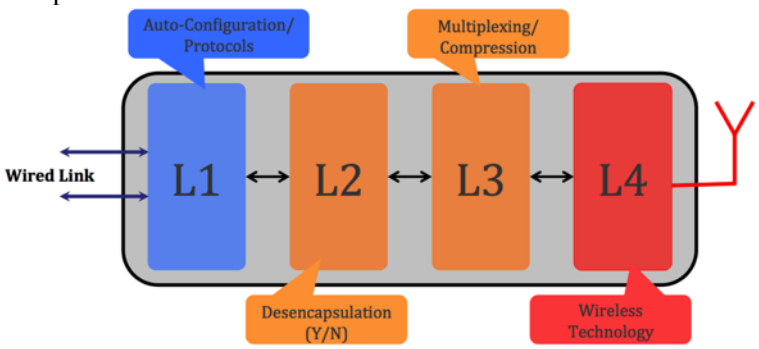

Figure 5. Wireless Device locks

It is important to note that lock L1 is necessary for the proper functioning of our system because we wish to develop a Plug and Play system, that is to say to be able to operate without knowing protocols parameters. Moreover, in order to process locks L3 and L4, it is necessary to work with OSI Model [3], [4].

In previous work, we also identified 3 wireless technologies (L4): Zigbee [5], [6], Bluetooth [5], [7], and UWB for Ultra WideBand [5], [8], [9]. It will also be implemented in our system. Wireless technologies forces us a bottleneck emphasizing an essential constraint that must be considered:

$$\sum_{allprotocols} WireDataRate < WirelessDataRate$$

The WDs aim being to have n protocols in input, it seems obvious that it is difficult to maintain this critical constraint. This is why we decided to develop a tool, call Decision Support Tool or DST, allowing to define hardware and software strategies to optimize operation of WD and thus minimizing wireless bandwidth. It can, for example, integrate hardware mechanism like DMA (Direct Memory Access) in order to make faster data transfer between Peripheral to Memory or Memory to Memory (and vice versa). DST should take into consideration all of 4 locks previously identified (Figure 5).

The purpose of this paper is to show interest and pertinence of development of an experimental platform. This platform will enable us to set up an experimental plan to test our various strategies developed in our WD. They will be qualified and quantified in terms of performance according to various indicators such as workload, data rate, protocols. Platform is divided into 3 parts that will represent 3 parts of 4 main sections of this article.

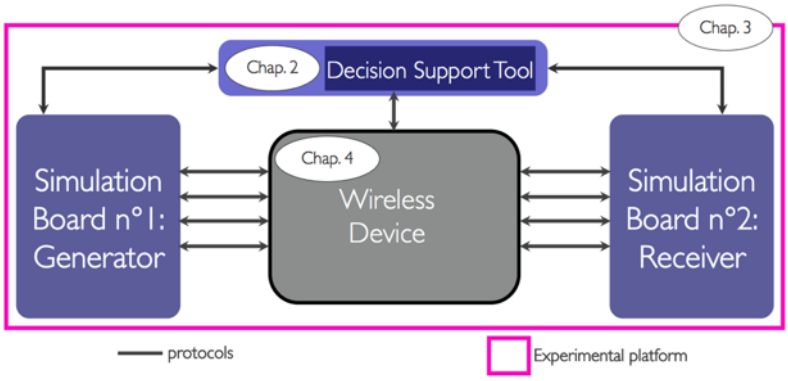

Figure 6. Experimental platform representation

In chapter 2 we will present DST with these hardware and software strategies, for our different architecture, in such a way as to be optimized and generic. In chapter 3, we will speak about our experimental platform whose purpose is to test performance of our WD described in chapter 4. We conclude this paper by presentation of first experimental results (chapter 5) and a conclusion about perspective.

2. DECISION SUPPORT TOOL

2.1. Aim

The purpose of DST is to solve locks L1, L2, L3, L4 (Figure 5) presented in previous paper [1]. The tool must be able to take into account all constraints in order to response to this problematic of wireless transfer:

- Workload: linked to protocols.
- Protocols and their parameters.
- Management of OSI Model layer (encapsulation/des-encapsulation).
- Multiplexing: we must develop a owner protocol in order to identify transmit data by wireless technologies because we have n protocols different in input (Figure 4, “H + Px_MSG”).
- System performance.
- Delay created by system.

DST will allow us to configure our WD in order to responding to a given problem. The purpose is to have a generic tool, that is to say capable of taking into account any type of architecture.

2.2. Working approach

The basic principle of our DST is to allow us to set up our WD in the most optimized way. Therefor, It will generate a *Configuration File* which will describe hardware and software strategies to be implement (Figure 7): it should take into account input protocols, as well as their parameters, priorities levels, buffer size in order to define the best optimization. This corresponds to the static aspect of our tool.

We also want to implement dynamic aspect, that is to say to be able to come to reconfigure our WD on detection of one or more particular events due to an integrated electronic device. Various scenarios can be imagined like increase buffer size, or priority level of protocol in order to adapt the increase of his workload for example. DST will be connected to wire protocols in order to able to make these changes.

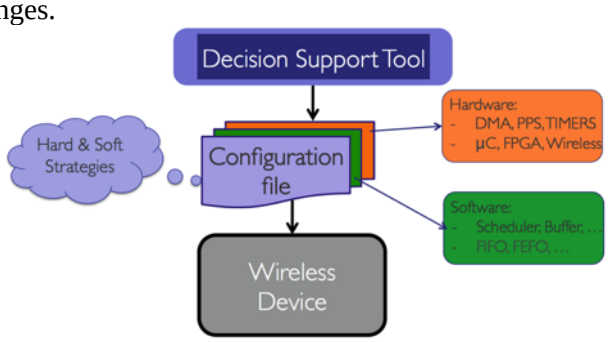

Figure 7. DST and WD

2.3. Architectures

In order that DST can generate *configuration file*, it need to be set. As illustrated in Figure 8, DST composed of several blocks:

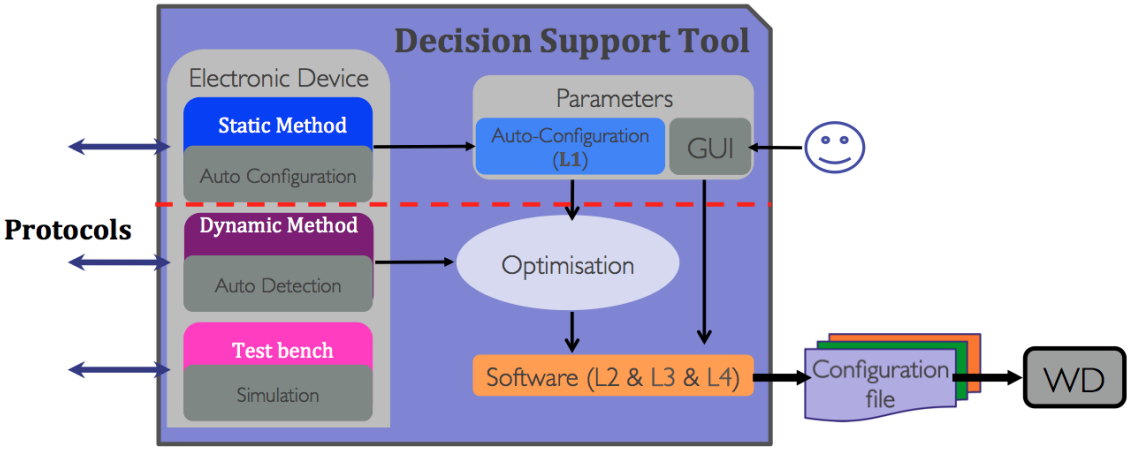

Figure 8. DST architecture

- **Settings**: it is an essential part of our DST. Indeed, it is necessary to properly configure this tool to have the right settings for our system (data rate, priority, ...). The configuration performs statically, that is to say at system initialization. 3 methods can be used:

- Manual: we enter information from a GUI: this involves knowing the parameters.
- Automatic: we do not know parameters, we use the "Auto Configuration" method (L1).
- Manual-Automatic: we use both methods.

- Setting in “dynamic”: we can actually speak of a reconfiguration. The aim is to come regenerate a small part of *configuration file* in order to come to dynamically reconfigure our system on arrival of external events generated by the party “Auto Detect” of the electronics device (increase of workload for example). The interest is to have an automatic system, able to adapt to increase the use of a protocol, allowing more priority to its messages, a buffer size larger, so that they are transmitted more quickly.

- **Electronic Device**: includes 3 essential parts to the proper functioning of our DST.
  - “Auto Configuration”: needed for automatic configuration in static mode.
  - “Auto Detection”: used to detect an event on the protocols and thereby regenerate a portion of the *configuration file* so that the system adapts.
  - “Test bench”: The objective of this part is to manage the experimental aspects in order to describe our various hardware and software optimizations.

- **Optimization**: optimizes our system of previously identified locks (L2 and L3). Different strategies will be implemented to optimize the scheduling (FIFO, FEFO), system priorities taking into account all constraints identified, buffers size? A model queue will be also used [10], [11] and meta-heuristic algorithms for optimization [12], [13].

- **Configuration file**: (Figure 9) it is the file describing the strategy which must be established to configure our WD. It allows defining hardware peripherals and their parameters including wireless technology (L4). It also describes algorithms to be implemented with the use or not of optimization brick present in DST.

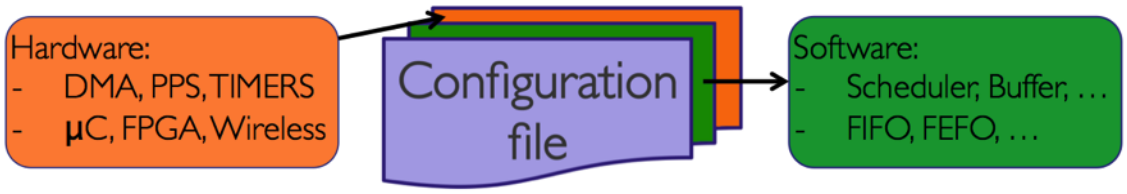

Figure 9. Configuration file

2.4. In summary

DST is a tool, which allow us to configure any WD so that it can respond to a given problem: number and type of protocol in entry, defining their parameters, scheduling system, priority level. It will generate automatically and transparently, a file will be called *configuration file* that will determine the hardware and software strategies to implement within our WD.
The advantage of such system is simple: have a generic tool, able to adapt to different technologies (processor, transceiver, wireless module), with different combinations of protocols, also having different combinations of parameters, which take into account the constraints of workload, ACK, response time, encapsulation.

DST has been introduced, I will now present our experimental platform that will allow experimental validation of our various strategies implemented.

3. EXPERIMENTAL PLATFORM

3.1. Aim

This platform will allow us to validate our different hardware and software strategies implemented in our wireless device. The benefit of linking our DST in this platform is to implement an experimental protocol that will allow us to start the same process to a different hardware and software strategies: it allows us to describe the system in order to determine that they are the best strategies. It will be necessary to develop a real experimental strategy and to define relevant indicators.

3.2. Experimental Strategy

In order to know if a hardware and/or software optimization is more interesting than another, it is necessary to define, in the first instance, indicators to observe the improvement in terms of processing speed, delay, power consumption ... Here are a few indicators identified:

- Delay: integrating a WD to an existing wired system, we will necessarily create a delay in data transmission. So it is important to measure it in order to define what is most relevant optimization: it corresponds to the smallest delay.
- Buffer size: it is the number of messages that we can be stored. But more we store and more we are falling behind.
- Priority level: defines which protocol will take priority. It can be assumed it must transmit faster data from a CAN bus protocol as an Ethernet or the RS232 protocol.
- Capacity system processing: that is to say delimit the lower boundary and the upper boundary. This will be in direct line with the different strategies used.

The second point to take into consideration is the experimental protocol. If you wish to know it is the most appropriate optimization, it is necessary to apply the same experimental protocol that is to say, the same data sets of events on different WD. You just have to compare the indicator results in order to determine what is the best strategy to apply.

Our DST will have the objective of reconfigure our WD to resolve this increased protocols workload. It may for example give more priority to the incoming data to be dealt with more quickly. It may also increase the buffer size to store more data.

The last point concerns the operation of the experimental phase: the transmitted data are known and are saved. Thus, DST will be able to compare the data transmitted and data received, so it will determine the time of transmission, involving the delay caused by the system. For example, comparing different types of strategies implemented in our WD, we can determine which is the most efficient and produces the least delay.

3.3. Platform architecture

This platform will allow us to test the hardware and software strategies implemented in different architectures within our WD. It is composed of our DST (Figure 10.1), electronic board for transmission (Figure 10.2.a) and receiving (Figure 10.2b) data on wired and our WD (Figure 10.3).

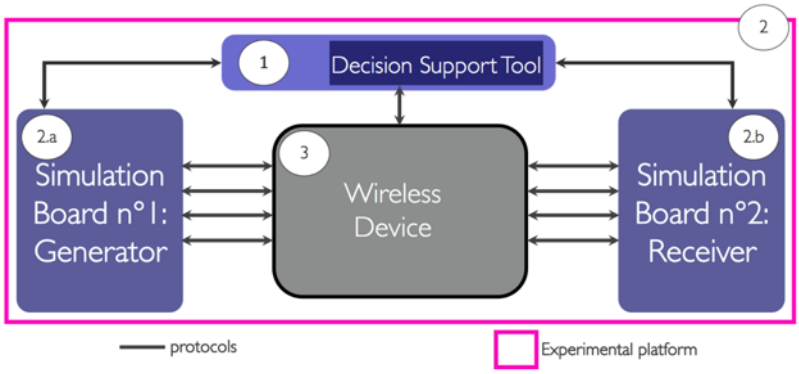

Figure 10. Experimental platform representation

The generator board 2.a will allow the transfer of data communication protocols. It also includes interfaces to dynamically manage constraints such as workload, speed to constrain more and more the system. The receiver board 2.b allows to recover the data sent over the wire channels to sure that the data transmitted by the board 2.a have been received. It is also used to measure various indicators such as delay caused by the system. Both electronic board have been designated and completely realized in our laboratory (see Figure 11). It incorporates a PIC24EP512GU810 microcontroller [14] allows us with its available peripherals (2 * CAN Bus, 4 * UART protocols: 2 * RS232 and 2 * RS485, and Ethernet via ENC424J600 component [15]). This microcontroller integrates PPS (Peripheral Pin Select) functionality, which can easily manage mapping of these different peripherals.

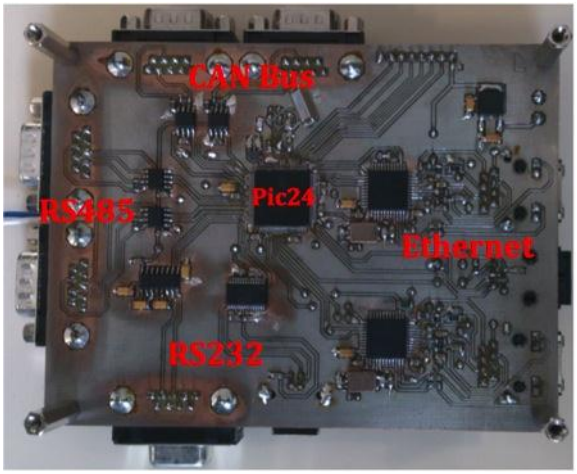

Figure 11. Simulation board

With the method implemented to the development of the platform, we have only come to connect a Wireless Device in order to the test.

4. WIRELESS DEVICE ARCHITECTURES

4.1. Aim

To carry out this study, we consider wireless technologies with their standards, speed, resistance to outside perturbations but also the protocols and their parameters or the different possible hardware architectures. We chose to develop 3 Wireless Devices (WDA, WDB, WDC)

representative of the hardware and software strategies that we wish to put in place with the DST. These solutions will fit perfectly on our experimental platform (Figure 12):

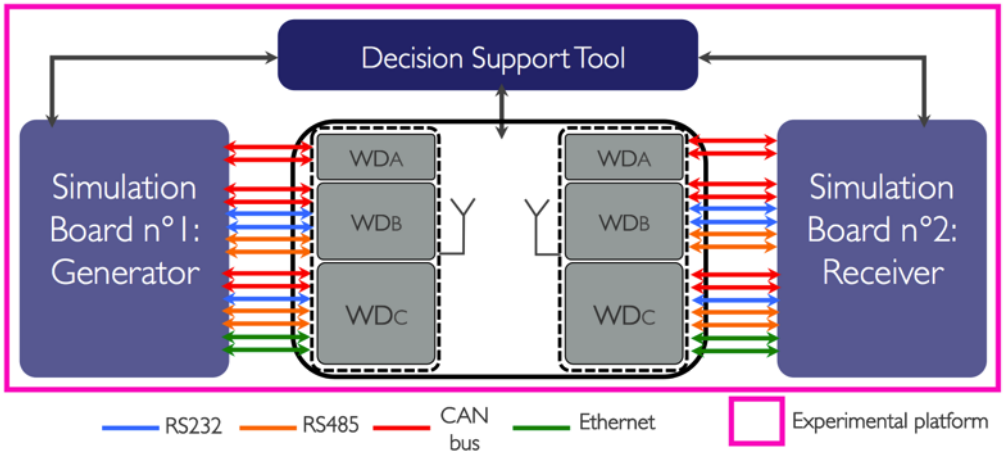

Figure 12. Experimental platform with our WDA/B/C

4.2. Our Strategies

We chose to work with 3 Wireless Devices (WDA, WDB, and WDC) in order to show relevance of our hardware and software optimization approach.

4.2.1. Wireless Device A

The WDA (Figure 13) will be relatively simple architecture to implement with little constraint on wireless throughput and allow us to work on the timing of the reception and transmission of CAN Bus architecture.

$$WireDataRate > WirelessDataRate$$

The target used is a PIC24EP512GU810 [14], the Zigbee module is an XBee Pro [16] and a BT31 Bluetooth module [17].

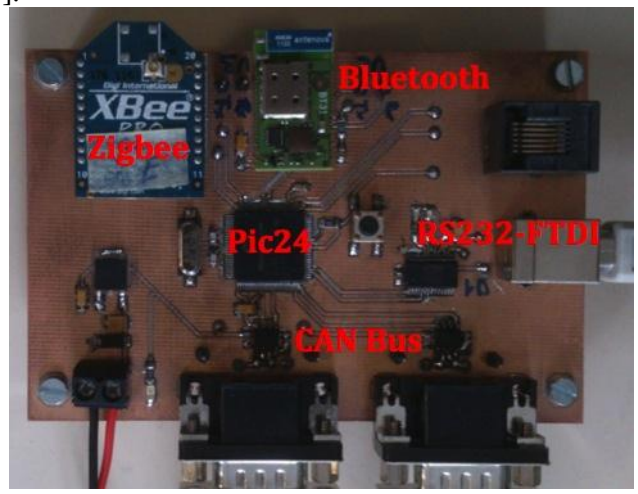

Figure 13. WDA demonstrator

4.2.2. Wireless Device B

The purpose of the device WDB (Figure 14) is to show the limits of a microcontroller architecture compared to the number of peripherals, memory and processing power available. An actual optimization will be made with the DST.

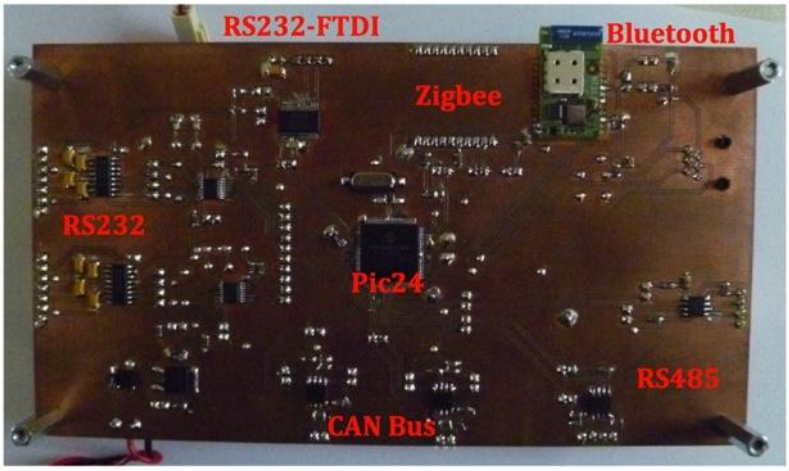

Figure 14. WDB demonstrator

Furthermore, wireless constraint is more important:

$$WireDataRate >> WirelessDataRate$$

4.2.3. Wireless Device C

The last device WDC aims to show dynamic aspect of our DST using an FPGA architecture allowing dynamic reconfiguration [18], [19]. However, we decided to use InES UWB kit [20] as wireless solution (not present in the two first devices) to have no constraint on our bandwidth.

$$\sum_{allprotocols} WireDataRate << WirelessDataRate$$

But in the future, we could have :

$$\sum_{allprotocols} WireDataRate >> WirelessDataRate$$

because we could have n input protocols.

The aim of WDC is to show the relevance of our dynamic approach between DST and a configurable and dynamically reprogrammable architecture [18]. To simplify the hardware implementation, we chose to work with the Industrial Network KIT (INK) [21] in Terasic that integrates wired protocols and hardware architecture (FPGA Cyclone IV [22]) which we need. Below is a representation of our dynamic architecture:

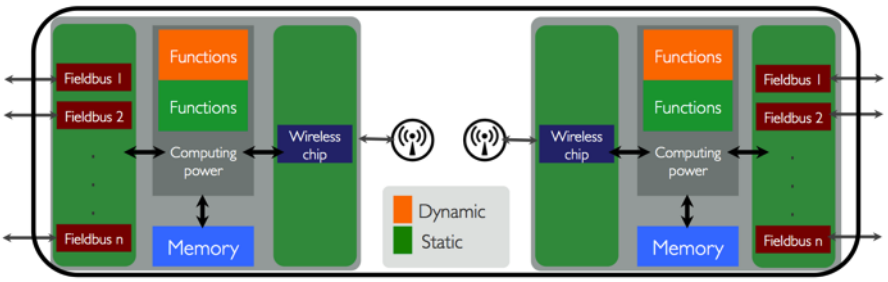

Figure 15. WDC dynamic architecture

As illustrated in Figure 15, the green areas correspond to the static phase, that is to say, the system boots: they come to configure according to these inputs / outputs wired and wireless. Orange area integrated within the processing unit corresponds to features that can be changed dynamically: buffer size, priority level, algorithm compression / decompression ... The objective of this architecture is to have a system capable to deal with different events, with different mechanisms, allowing it to adapt and overcome them.

4.3. Strategies comparison

This table provides a quick comparison between our different Wireless Device A/B/C:

WD	Protocols		Wireless		Target	Constraints	Interest
	type	data rate	type	data rate			
WDA	CAN Bus 2.0A	1Mbps	Zigbee	115kbps	PIC24	WireDataRate > WirelessDataRate	low constraint on wireless data rate
	CAN Bus 2.0B	1Mbps	Bluetooth	921kbps			simple architecture
							works on CAN Bus timing
WDB	CAN Bus 2.0A	1Mbps	Zigbee	115kbps	PIC24	WireDataRate >> WirelessDataRate	high constraint on wireless data rate
	CAN Bus 2.0B	1Mbps	Bluetooth	921kbps			show system and architecture limitation
	RS485	12Mbps					high optimization with DST
	RS485	12Mbps					
	RS232	56kbps					
	RS232	56kbps					
WDC	Ethernet	100Mbps	UWB	480Mbps	Cyclone 4	WireDataRate << WirelessDataRate	no constraint on wireless data rate
	Ethernet	100Mbps					show the dynamic aspect with an FPGA
	CAN Bus 2.0A	1Mbps					using programmable and reconfigurable archi.
	CAN Bus 2.0B	1Mbps					dynamicaly reconfiguration
	RS485	12Mbps					more liberty in architecture
	RS485	12Mbps					peripherals, PINs mapping...
	RS232	56kbps					optimized integration with dynamic DST part

Table 1. Wireless Device comparison

5. EXPERIMENTAL RESULTS

5.1. Delay Calculation

The delay is the time "lost" due to the integration of WD. This study was conducted with the WDA (see Section 4.2.1). Through our platform we were able to determine the delay time caused by the system.

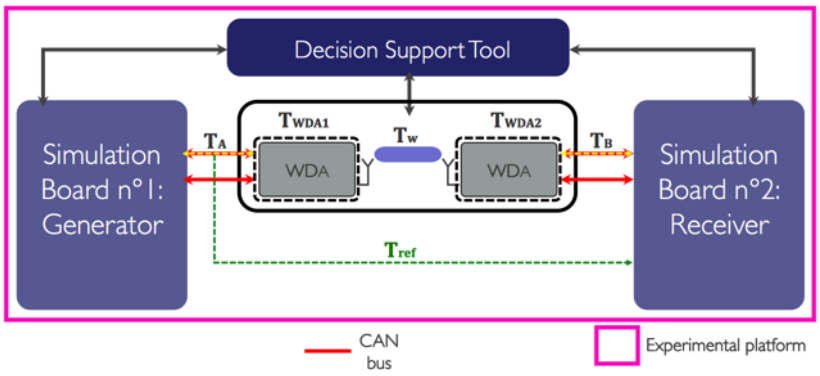

Figure 16. Delay calculation

Here the principle of calculation with:

- *Tref* = wire transmission time.
- *Twd* = $T_A + (T_{WDA1} + T_W + T_{WDA2}) + T_B$

Our DST measures the time reference: *Tref*, and the total transmission time via the wireless device: *Twd*, (see Figure 16). *Tref* allows us to know "normal" time of data transmission. We can calculate the delay:

$$Tdelay = Twd - Tref$$

5.2. Results

This experiment was carried out with Zigbee and Bluetooth technology and CAN Bus 2.0B to 1Mbps with DLC = 2. The experimental protocol used is relatively simple: to illustrate this delay and relevance of material optimization, we wanted to test two strategies: with and without DMA. We make a simple test: 10 CAN Bus frames per second during 8 minutes. The aim here is to see delay transmission and not workload of system.

5.2.1. Bluetooth

In order to compare Zigbee and Bluetooth technologies, we are working with 2 data rate for Bluetooth technology:

- 115 kbps.
- 921 kbps.

Figure 17 shows delay introduce by Bluetooth technology, for both data rate with and without DMA. We can see average of delay for both graphics. We note that delay introduce by Bluetooth is very variable.

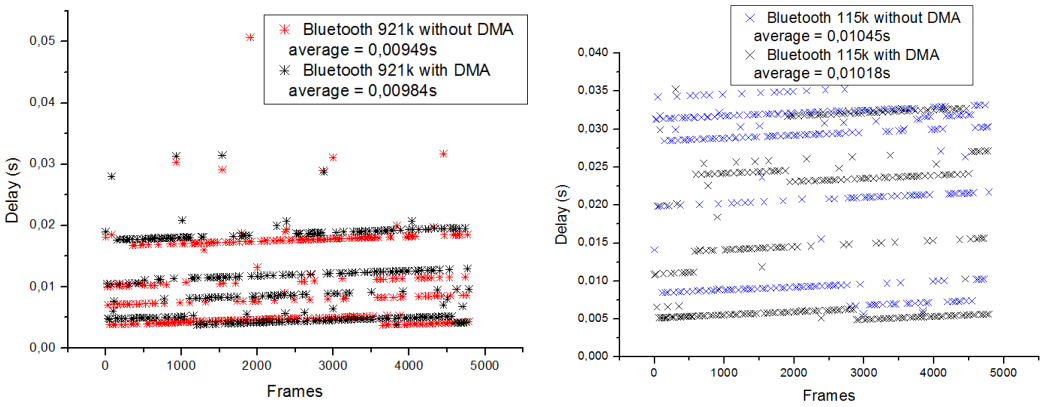

Figure 17. Bluetooth delay

5.2.2. Zigbee

Figure 18 shows delay introduce by Zigbee technology. We can see that average of Zigbee is smaller than Bluetooth.

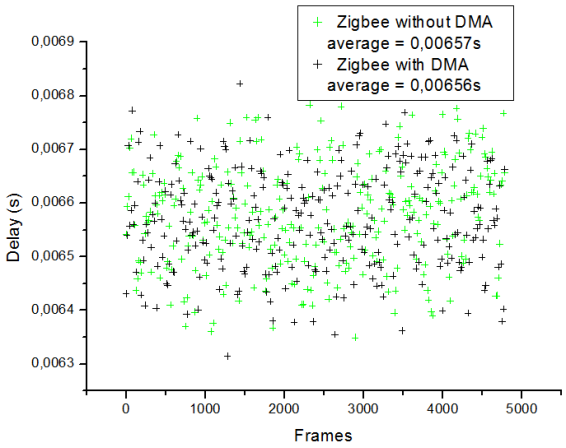

Figure 18. Zigbee delay

5.2.3. Comparison

Aim of this graphic (Figure 19) is to show delay variation introduce by Bluetooth technology whereas delay introduce by Zigbee is constant. We don't show DMA or no DMA strategy on this figure because we focus on delay difference between Zigbee, Bluetooth and wire.

Due to many layer on Bluetooth technology, Figure 19 sees a fluctuation (red and blue plots) in contrary of Zigbee technology (green plot) where all values are concentrated. We also see purple plot, corresponding to wire connection: it corresponds to *Tref (*see section 5.1), that is to say data transmission without Wireless Device.

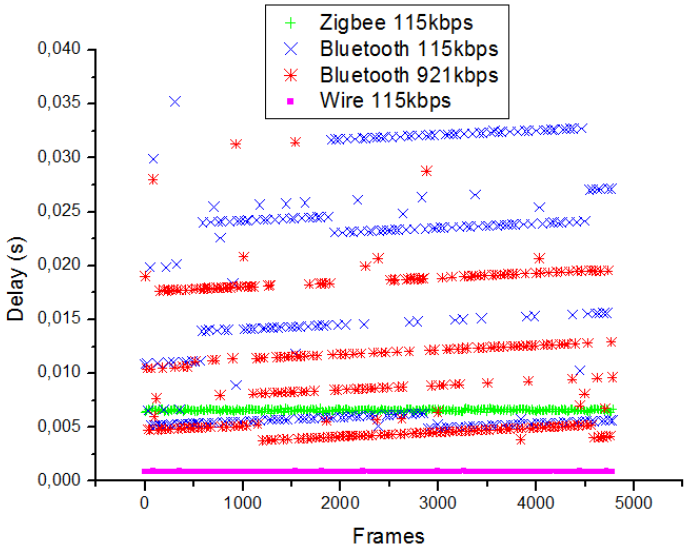

Figure 19. Delay comparison depending on connection types

Colours of both Figure (19 and 20) correspond to the same data. Figure 20 is a column diagram representation of delay average. It is a pertinent figure because we can see difference of delay transmission between types of transmission (wire and wireless). It is obvious that wire connection is more efficient (Figure 20, purple plot). However there is Zigbee technology can be a good wireless solution relative to Bluetooth, despite some values of Bluetooth delay (Figure 19, blue and red plots) below Zigbee delay (Figure 19, green plot).

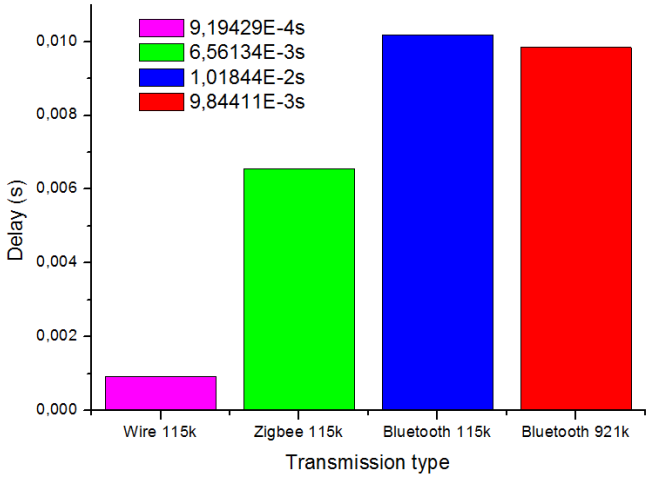

Figure 20. Average of delay in function of transmission types

5.3. DMA optimization

We can see the difference with and without DMA is not obvious (Figure 17 and 18). The reason for this is simple: the management of CAN Bus is already configured with DMA as shown in Figure 21 below taken from the datasheet of the component [14].

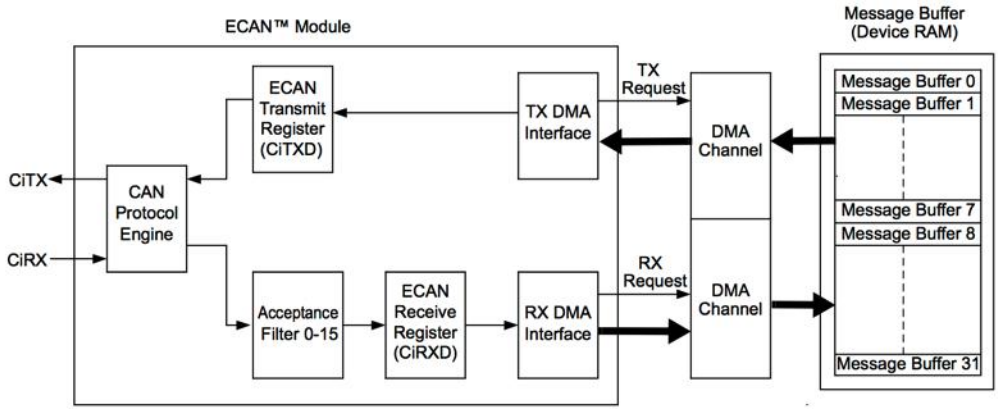

Figure 21. CAN interaction with DMA

Whether the transmission or reception, it is already working with the DMA: data are processed faster. In this context, our optimizations for the DMA transfer MSG to the wireless channel (and vice versa) with DMA as shown in the Figure 22.

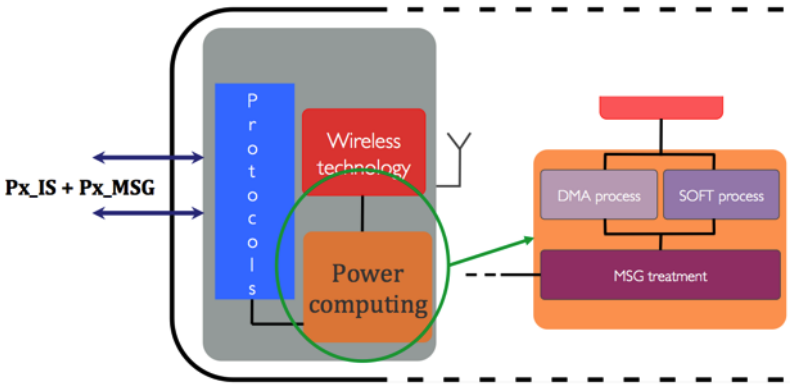

Figure 22. DMA/Software data transfer

One can speak optimized transfer for a ten byte (header multiplexing MSG + CAN bus): the time saved is minimal in comparison with software processing. The impact would be more important on larger architectures in terms of protocols and data traffic. In conclusion, element generating the largest delay is the Bluetooth component. Additional strategies will be developed to minimize it.

5.4. DST

Our DST allows us to see delay transmission due to Wireless Device. We have 2 operating modes:

- File reading: we save data on text file. Our simulation board is connecting to computer with RS232 technology. An example is show with Figure 20.
- Real-time data: due to RS232 connection between simulation board and DST, we can see in real-time delay transmission.

Figure 23 is a Graphic User Interface (GUI) develops with QT. We can see that Zigbee delay (green plot at 115kbps) is smaller that Bluetooth delay (red plot at 921 kbps, blue plot at 115kbps). This tool also allows us to see delay average in real-time.

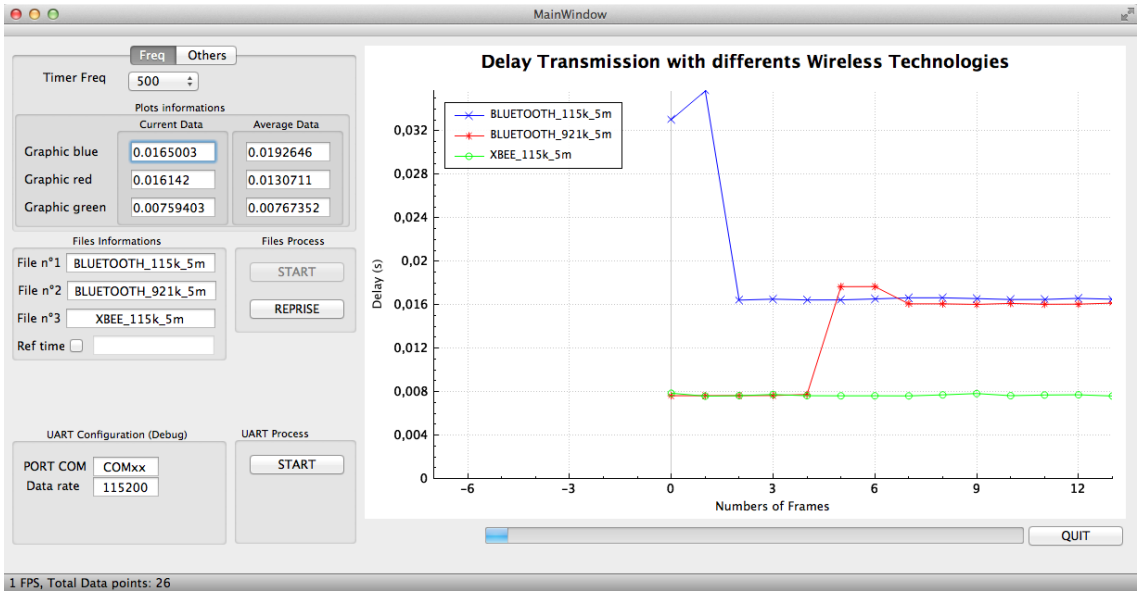

Figure 23. DST data viewer (read file)

5.5. Prospects

Experiments are going to continue with our other WD. The relevance of the DMA will be more significant in the WDB and WDC because the volume of data exchanged will be much higher and that for several reasons:

- More protocols (WDB et WDC) in I/O.
- Higher MSG (Ethernet, WDC).

6. CONCLUSIONS

The development of this experimental platform allows us to test hardware and software strategies defined by our DST in order to optimize our WD. We can test wire protocols like RS232/422/485, CAN Bus and Ethernet with different parameters for different hardware architectures and wireless technologies. More, we can see in real-time delay introduce by WD and compare it with another data from saved files.

This first study has allowed to validate our approach but also our tools. Next step will be to test WDB and to develop our programmable and reconfigurable architecture with RICA (WDC).
The aim of this platform is to test all possible solutions statically but also dynamically highlighting, through our various indicators, the best solutions and optimizations.

ACKNOWLEDGEMENTS

This study has been carried out thanks to the financial support provided by the Region of Picardie.

REFERENCES

[1] R. Astier, T. Capitaine, V. Bourny, J. Dubois, and J. Fortin, "Technological leap of signal transfer system: CONTACTLESS," in *IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society*, 2012, pp. 3238–3243.

[2] R. B. GmbH, "Can specification," Version 2.0, Tech. Rep., 1991.

[3] H. Zimmermann, "OSI reference Model–The ISO model of architecture for open systems interconnection," *IEEE Transactions on Communications*, vol. 28, no. 4, pp. 425–432, 1980.

[4] Y. Li, W. Cui, D. Li, and R. Zhang, "Research based on OSI model," in *2011 IEEE 3rd International Conference on Communication Software and Networks (ICCSN)*, 2011, pp. 554–557.

[5] Y.W. S. Jin-Shyan Lee and C.C. Shen, "A comparative study of wireless protocols: Bluetooth, uwb, zigbee, and wi-fi," in *The 33rd Annual Conference of the IEEE Industrial Electronics Society (IECON)*, 2007.

[6] P. Baronti, P. Pillai, V. W. Chook, S. Chessa, A. Gotta, and Y. F. Hu, "Wireless sensor networks: A survey on the state of the art and the 802.15.4 and zigbee standards," *Computer Communications, ScienceDirect*, p. 1655, 2007.

[7] P. McDermott-Wells, "What is bluetooth?" *IEEE Potentials*, vol. 23, no. 5, pp. 33–35, 2005.

[8] ECMA-368, "High rate ultra wideband phy and mac standard," ECMA International, Tech. Rep., 3rd Edition / December 2008.

[9] ECMA-369, "Mac-phy interface for ecma-368," ECMA International, Tech. Rep., 3rd Edition / December 2008.

[10] B. D. Choi, S. H. Choi, B. Kim, and D. K. Sung, "Analysis of priority queueing system based on thresholds andits application to signaling system no. 7 with congestion control," *ELSEVIER, Computer Networks 32*, 2000.

[11] S.Balsamo, V.D.N.Persone, and P.Inverardi, "Areviewonqueueing network models with finite capacity queues for software architectures performance prediction," *Perform. Eval.*, vol. 51, no. 2-4, pp. 269– 288, Feb. 2003. [Online]. Available: http://dx.doi.org/10.1016/S0166- 5316(02)00099-8

[12] D. Jones, S. Mirrazavi, and M. Tamiz, "Multi-objective meta-heuristics: An overview of the current state-of-the-art," *European Journal of Operational Research*, vol. 137, no. 1, pp. 1 – 9, 2002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377221701001230

[13] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM Comput. Surv.*, vol. 35, no. 3, pp. 268–308, Sep. 2003. [Online]. Available: http://doi.acm.org/10.1145/937503.937505

[14] Pic24ep512gu810. [Online]. Available: http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en554334

[15] Enc424j600: 10/100 base-t ethernet interface controller. [Online]. Available: http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en542414

[16] Xbee-pro 802.15.4. [Online]. Available: http://www.digi.com/fr/products/wireless/point-multipoint/xbee-series1-module

[17] B. module. [Online]. Available: http://ampedrftech.com/modules.htm

[18] S. Khawam, I. Nousias, M. Milward, Y. Yi, M. Muir, and T. Arslan, "The reconfigurable instruction cell array," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 16, no. 1, pp. 75–85, 2008.

[19] T. S. Arslan, S. Khawam, J. M. Millward, I. Nousias, and Y. Yi, "Reconfigurable instruction cell array," The University Court Of The University Of Edinburgh, Tech. Rep., EP1877927 B1.

[20] H. Gelke, "Wireless 480 mbit/s uwb link for embedded system," *InES-Institute of Embedded Systems*, 2009.

[21] Industrial networking kit (ink). [Online]. Available: http://www.terasic.com.tw

[22] Cyclone iv fpga family. [Online]. Available: http://www.altera.com/devices/fpga/cyclone-iv/cyiv-index.jsp

AUTHOR

Remy ASTIER received his Master Degree in Embedded System at INstitue Supérieur des Sciences et Techniques (INSSET), at Saint-Quentin, in France, in 2009. Since, He is a PhD Student, in the LTI laboratory (EA3899). He is working on hardware embedded architecture to be able to plug on every systems which using wires protocols, in order to do smart wireless data transfer.