Actas de las XXXIX Jornadas de Automática, Badajoz, 5-7 de Septiembre de 2018
https://doi.org/10.17979/spudc.9788497497565.0538

A COMPARISON OF WIRED AND WIRELESS TECHNOLOGIES FOR CONTROL APPLICATIONS

Isidro Calvo[1]✉, Steven Abrahams[2], Oscar Barambones[1], Ander Chouza[1], Javi Velasco[3], Idurre Sáez de Ocáriz[3], Jerónimo Quesada[1]
[1]Escuela de Ingeniería de Vitoria-Gasteiz, University of the Basque Country (UPV/EHU), Nieves Cano 12, 01006 Vitoria-Gasteiz.
[2]University of Hasselt / KU Leuven, Hasselt, Belgium
[3]Fundación Centro de Tecnologías Aeronáuticas (CTA), Juan de la Cierva 1, 01510 Miñano.
{isidro.calvo✉, oscar.barambones, ander.chouza, jeronimo.quesada}@ehu.eus
{javier.velasco, idurre.saezdeocariz}@ctaero.com

Abstract

Wireless technologies are already mature in non-critical applications. However, the use of this kind of technologies in control applications is, still, under research. There are several reasons for the reluctance to use this kind of technologies such as the lack of reliability, finding the optimal sampling period not to disturb the control algorithm, the introduction of message delays, higher message dropouts (e.g. caused by interferences) or the power supply at the devices. Most of these issues are related to the Quality of Service (QoS) that wireless technologies provide. This article evaluates the use of wireless technologies to be used in control applications when compared with wired technologies. The article draws some conclusions and recommendations for building control applications. In particular, the problem of building wireless networked control systems (WNCS) is addressed by means of XBee technology for communication and LabVIEW for processing the acquired data, implementing the control algorithm and sending the control signal to the actuators. These tools were selected based on the low power consumption of the XBee devices and the flexibility of LabVIEW for building complex control applications.

Key words: Wireless Communications, Networked Control Systems (NCS), Zigbee, XBee, LabVIEW

1 INTRODUCTION

This paper explores wireless networked control systems (WNCS) for control applications built with XBee and LabVIEW. Nowadays, most control applications are built with wired technologies, due to several issues that have a big influence in control applications such as the reliability (limiting message delays and lowering message dropouts), finding the optimal sampling periods, or feeding power to the wireless devices. Networked control applications require a specific Quality of Service (QoS) in order to be usable [1], and obviously, wired communication technologies are more predictable and reliable than wireless technologies. However, wireless technologies are expected to provide high benefits in terms of flexibility since they avoid long wires that are hard to place or may become an obstruction [2]. These benefits are especially noticeable when mobile devices are involved in the control loop (e.g. sensors or actuators over a moving platform), but in general avoiding (or at least reducing) wires eases the construction and maintenance of any machinery, and also reduces the cost. So, the big research question for the authors is: *Do the benefits of using wireless technologies in control applications get over the drawbacks?*

The authors focus on an approach based on XBee and LabVIEW technologies. On one side, XBee is used for providing wireless communications among sensors, controller and actuators. XBee devices are based on the IEEE 802.15.4 protocol, which is known for its low power consumption [3]. This may become an important issue when building control systems in which some components (sensors or actuators) are powered with batteries. On the other side, LabVIEW is used to implement the controller that gets the information from the wireless sensors, processes the control algorithm, and sends the control signal to the wireless actuators [6]. LabVIEW allows the implementation of complex control algorithms by means of combining several modules. In addition, LabVIEW works well at different sampling periods, provides several alternatives for communication purposes and utilities for saving monitored information. Unfortunately, due to the fact that WNCS are still under research, especially at critical applications, the support of LabVIEW for XBee communications is still a bit limited.

In order to evaluate whether XBee is suitable for control applications, several experiments at different sampling periods and distances were designed to compare wireless communication versus wired communication. Firstly, the Analog to Digital Converter (ADC) included in the XBee device (Fig 2-a) was compared with a NI myDAQ USB-6008 board (with higher resolution, 12 bits) and an

Arduino (same resolution, 10 bits). Secondly, the offset and message dropout were analyzed for different distances and sampling periods. Thirdly, the message delay was analyzed at different distances and sample periods. Finally, the influence of Wi-Fi interferences was evaluated. The results of the study should show if the combination of XBee and LabVIEW may be adequate to build WNCS, or at least to see in which situations they may become a suitable alternative.

The layout of this paper is as follows. Section 2 provides an overview about the XBee technology. Section 3 proposes the approach followed by the authors. Section 4 describes and discusses the experimental tests carried out. Finally, Section 5 draws some conclusions.

2 XBEE TECHNOLOGY

Even though there are several wireless technologies around, Fig. 1 shows the most used for control applications [4]. This figure compares different technologies in terms of data rate, power consumption, cost and complexity. It can be appreciated that Zigbee is considered a low-cost, low-power solution. Even though Zigbee does not allow sending big blocks of data, it seems adequate for connecting simple analog sensors [3]. XBee is an implementation of the IEEE 802.15.4 standard. With regards to distances, XBee allows to communicate up to 100 m (there is also XBee-Pro that improves these figures, not tested in this work).

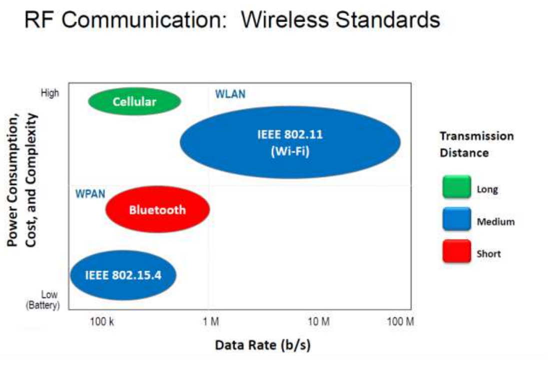
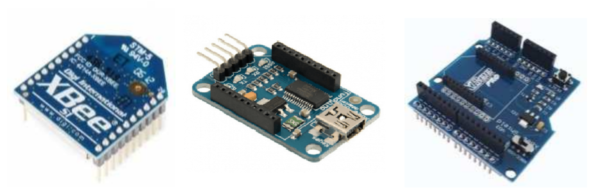


Fig. 1: Wireless technologies available [4]

There are several kind of devices used with XBee technology (see Fig. 2) as follows: (a) XBee antennas: they manage the XBee protocol. Also, they may be configured for measuring/producing digital or analog signals; (b) USB adapter: It is used for configuring the XBee antennas. Also, it may be used to connect several XBee devices to a computer by means of a USB connection; (c) XBee-Arduino shield: This shield allows connecting Arduino boards to XBee antennas. All of these devices were used in the current work.



a) b) c)

Fig. 2: XBeeS1 devices: a) XBeeS1 antenna; b) USB adapter; c) Arduino shield

XBee technology allows complex configurations that may adapt to different situations. For that reason it is necessary a tool that allows configuring the XBee devices. This tool is known as XCTU [9]. It allows configuring a broad number of parameters of the XBee technology. The most relevant parameters are shown in Table 1:

Table 1: XBee main configuration parameters

Param.	Functionality
ID	Set the PAN ID (Personal Area Network Identifier)
DH/DL	Set/Read the Higher/Lower Destination Address (32+32 bits)
MY	Set/Read the 16 bits address of the module
CE	Set/Read the coordinator settings (End Device/Coordinator)
NI	Set/Read Node Identifier String
AP	Enable API mode
D0..8	Configurate different pins of the XBee antenna, for digital/analog inputs outputs
IR	Set/Read sample rate

XBee provides several working modes: AT mode and API mode. The AT mode allows to use the XBee device as a serial modem, establishing a serial communication. However, this is not a flexible approach since the communication parameters must be written in the firmware and this operation takes a long time. For that reason the API mode is provided. This is a lower level communication approach in which there is full access to the frames. The API mode was used in this work.

3 ADOPTED APPROACH

In this section it is proposed a configuration with XBee and LabVIEW that allows creating WNCS. Basically, the authors are aimed to create control applications that follow the layout of Fig. 3.

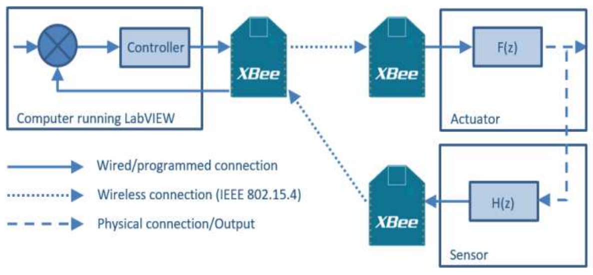

Controller
XBee
XBee
F(z)
Computer running LabVIEW
Actuator
Wired/programmed connection
Wireless connection (IEEE 802.15.4)
Physical connection/Output
H(z)
XBee
Sensor

Fig. 3: Layout of a wireless networked control system

Fig. 3 shows the basic closed control loop with a (1) sensor, (2) a control algorithm (executed by the computer) and (3) an actuator [1]. Even though the figure shows a SISO system, from the architectural perspective, it can be extended for more sensors and actuators. However, in such case the sampling rate of the control system could need to be recalculated, since adding more XBee devices may have an impact on the QoS performance.

As Fig. 3 shows, the sensors connected to the physical system are connected to a XBee antenna. The use of a microcontroller, or similar, allows the execution of any pre-processing algorithm for the acquired values (represented as H(z) in Fig. 3). These measured values will be serialized and sent to the control computer (which runs the control algorithm in LabVIEW). For that reason the computer needs to be connected to the XBee antenna by means of a USB link. The LabVIEW VISA device has been proven to be the best alternative to manage the data received/sent by the XBee module. The control algorithm generates the signal to be serialized and sent to the XBee actuator. Again, the use of a microcontroller on the actuator's side may be necessary in order to execute F(z).

As explained in previous section, the XBee API mode provides higher flexibility. However, it requires a lower programming level, since the programs must dig into the frames to get the information. Anyway, this was considered the best approach and consequently it was used at the sensor, controller and actuator. Afterwards, the format of the API frames is briefly described.

Fig. 4 shows the structure of a typical frame. It always starts with the XBee frame delimiter, 0x7E. It continues with the number of bytes at the entire frame, without the delimiter. Next, it goes the API-specific structure. This structure always starts with an API identifier to determine the kind of message sent or received and is followed by the data corresponding to the message. Finally, a message checksum is calculated [10]. The API-specific structure is where the application data goes (i.e. the value of the sensors and actuators). A simple protocol was created ad hoc, which included the API identifier, the destination address, and the sensor/actuator data to be sent. All the programs executed at sensor, controller and actuator had to dig into this frame to get the data to be sent/received.

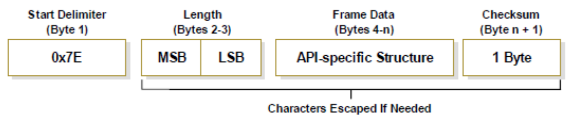


Fig. 4: Structure of a typical XBee API frame

Due to the limitations of LabVIEW dealing with XBee technology, the LabVIEW computer is provided with one XBee-USB adaptor by means of the LabVIEW VISA device, which allows managing easily serial communications from LabVIEW. National Instruments and several other companies designed the NI VISA protocol, aimed at easing the communication with different devices [5]. In particular, the NI VISA protocol allows the connection between the XBee antenna located at the control computer and the LabVIEW application that executes the control algorithm.

4 EXPERIMENTAL WORK

The experimental layout reproduces Fig. 3, for a Single Input Single Output (SISO) Control System. Three components were considered: (1) sensor, (2) actuator, both connected wirelessly with (3) the controller, executed in a computer running LabVIEW. The NI VISA device was used to manage the communication between the XBee and LabVIEW application.

In parallel to the wireless communication, a NI myDAQ USB-6008 acquisition board was used to compare wireless and wired communications. This is a low cost board designed by National Instruments and, consequently, has very good compatibility with LabVIEW by means of ad hoc libraries and drivers. The NI myDAQ USB-6008 also provides higher resolution (12 bits) when compared to the XBee and Arduino ADCs (10 bits). For these characteristics, it was considered as the reference value for the tests carried out.

The test set up is aimed at analyzing the performance of wireless communications in control applications, not the application itself. For that reason, the test sensor component was implemented by means of an Arduino board that sampled the value of a potentiometer (see Fig. 5). The Arduino's acquired values were serialized and sent to a XBee connected module. This module represents, from the communications point of view, any analog sensor with XBee connectivity.

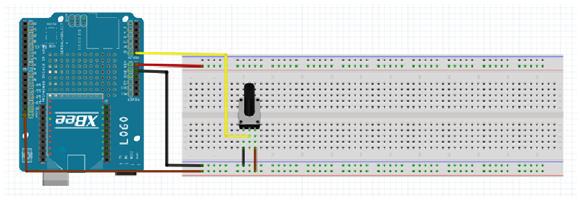


Fig. 5: Sensor module used in the test layout

Regarding the actuator side, an Arduino was also connected to a XBee module (by means of an Arduino XBee shield). This module received the control signal generated by the LabVIEW control algorithm. For the test purposes a DC motor was connected to one of the PWM Arduino board pins. Thus, the authors aimed at representing a generic actuator (see Fig. 6).

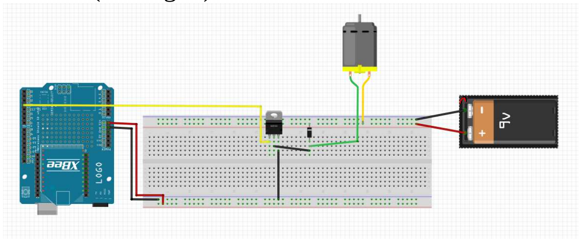


Fig. 6: Actuator module used in the test layout

Several experiments were designed with this layout aimed at testing the behavior of the XBee protocol and modules in LabVIEW-based control applications. Next subsections describe these experiments and discuss briefly the results.

4.1 Comparing different ADCs

This test was aimed at analyzing the influence and precision of the ADC at the XBee module by comparison to other ADCs. The XBee ADC represents the measured analog voltage into a 10-bit value. This value was compared to the value measured by a NI myDAQ USB-6008 (12 bits). These ADCs were also compared to the values measured directly with an Arduino board (which uses a 10-bit ADC) and transmitted to LabVIEW by means of an XBee module. These tests were done at 0.5 meters distance and a sample period of 100 milliseconds.

The behavior of the XBee ADC was quite limited. In the tests, it was proved that the XBee ADC introduced an offset of around 0.4 Volts when 0 volts were measured. The higher the voltage, the difference with the value obtained from the myDAQ was diminished. Due to these poor results, the authors used an Arduino in order to discretise the sensor signal. These values were compared with those acquired by the myDAQ board. Fig. 7 shows the comparison between the Arduino with XBee (wireless) and myDAQ (wired). It can be appreciated (see Fig. 7 below) that the maximum difference between both signals is around 0.05 volts. This seems to happen when the highest increments occur.

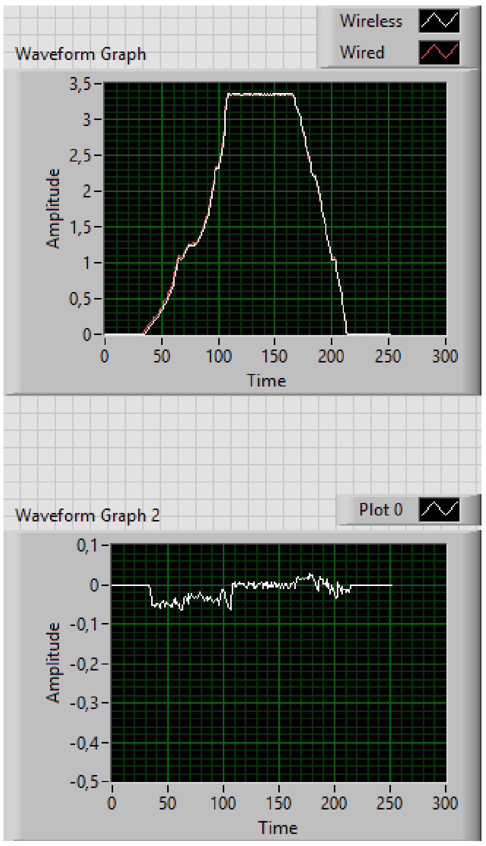


Fig. 7: Wireless (Arduino + XBee) vs. wired (NI myDAQ USB-6008). *Top*: Obtained signals. *Botton*: Difference between the signals

4.2 Comparing offset and message dropout at different distances

The second experiment was aimed at investigating the influence of the following characteristics at the dropout: (1) the sample period and (2) the distance between the computer and the receiving modules. Several tests were done with sample periods of 30, 50 and 100 milliseconds and at a distance of 0, 0.5, 2 and 5 meters. Namely, the test consisted of sending periodically sensor and actuator messages at different frequencies (at 30, 50 and 100 milliseconds) and varying the distance at 0 (very short distance), 0.5, 2 and 5 meters. The following parameters were measured: (1) the message dropout; (2) the average difference between the wired and wireless signals at these frequencies and distances; and (3) the standard deviation of the difference between wired and wireless signals. The measurements were done at different voltages (1 V, 2 V and 3.3 V). Table 2 summarizes the obtained results.

Table 2: Summary of the experiment results at different frequencies and distances

Test	Distance [m]	5			2			0,5			0		
	Sampling period [ms]	30	50	100	30	50	100	30	50	100	30	50	100
Message dropout [%]		1,3	1,2	1,3	1,8	2,1	2,2	0,9	2,3	1,7	1,6	2,2	2,3
Average Diff. at 1V [mV]		-19	-60	-85	-29	-29	-45	-24	-23	-22	-22	-32	-21
Standard deviation ↑ [mV]		15	12	20	12	9,4	16	4,6	5,4	5,4	7,9	10	7,6
Average Diff. at 2V [mV]		-8,5	-25	-39	-16	-12	-15	-14	-11	-12	-12	-13	-5,8
Standard deviation ↑ [mV]		12	16	16	11	23	12	5,0	2,9	4,4	8,3	9,4	14
Average Diff. at 3.3V [mV]		0,9	2,2	0,8	-0,3	1,1	1,2	-1,7	-2,3	-2,4	-1,0	1,3	-2,0
Standard deviation ↑ [mV]		6,2	5,7	8,2	19	7,9	7,2	5,2	4,9	6,2	6,7	5,6	13
Average Diff. total [mV]		-9,9	-32	-40	-14	-13	-18	-14	-13	-13	-14	-16	-12
Standard deviation ↑ [mV]		16	31	40	19	19	25	16	14	14	17	19	16

The first thing to notice in table 2 is that the message dropout measured is low, averaging below 2%. Also it was not found any correlation between message dropout and distance (at least the considered distances) and frequencies. The authors consider that the high values at the standard deviation are mostly due to the effect of the interferences during the test. Also, it can be appreciated that, in general, the offset (see section 4.1) becomes smaller as the voltage gets higher. As happened in section 4.1 when comparing the behavior of the ADC at the XBee, whenever there is a rising edge the difference in voltage increases in the negative direction and for a dropping edge it increases in the positive direction (see Fig. 7). A last remark is that at each different voltage the average stays stable with several outliers. The most remarkable outliers occurred at five meters. The reason for this is still not clear, but it could be due to interferences.

4.3 Message delay

The third experiment was aimed at measuring the full message delay between sending a signal from the sensor to the computer and receiving at the actuator (see Fig. 8). The time difference between sending and receiving was then measured using an oscilloscope. These tests were only done at a distance of 0, 0.5, 2 and 5 meters. Next to the wireless connection the wired connection was tested in the same way.

Fig. 8 implements the more generic view of a SISO WNCS deployed in Fig.3.

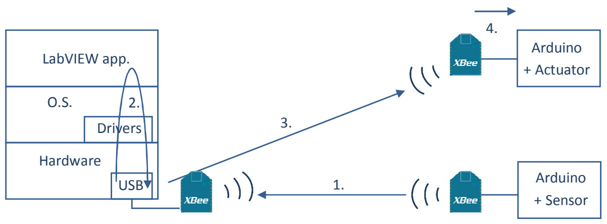


Fig. 8: Layout for message delay

In order to reduce the influence of the size of the data sent, for this experiment the potentiometer was replaced with a pushbutton and the motor by a LED. The sensor and actuator were put at several distances from the computer. The mean and standard deviation of ten measurements are shown in table 3.

Table 3: Message delay for 9600 bps

Test	Distance [m]	Wired	0	0.5	2	5
Mean [ms]		3	45,9	45,5	46,2	47,2
Std. Dev.		0,60	3,93	5,25	4,92	6,39

Two things can be noticed from this experiment: (1) the distance does not seem to have a big impact on the delay (below 5 meters), and (2) the delay is significantly lower in the wired connection. For this experiment the XBees were configured to transmit the data at a baud-rate of 9600 bits per second.

Considering that the frames are around 100 bits and that there are several steps to close the loop (see Fig. 8) it is explained that the mean time is around 46 milliseconds for the wireless connection. These different steps also explain a higher standard deviation. To reduce the time needed to translate the data into a frame a higher baud-rate can be used. A second test was done at two different baud-rates. The results are shown in table 4.

Table 4: Message delay at different baud-rate

Test	Speed [bps]	9600	38400	57600
Mean		45,9	24,1	22,6
Std. Dev.		3,93	4,07	3,95

The table shows the influence of the communication speed configuration (in bits per second) at the delay. For example, at a baud-rate of 38400 the time to send one frame will be considerably shorter, and it will have big impact in the closed loop communication. However, the authors expect a higher dropout rate (not measured). If the time used by the computer to process the data is considered the same for all tests (regardless of the communication speed) the following four steps may help to understand where the data is delayed. These steps were depicted in Fig. 8:

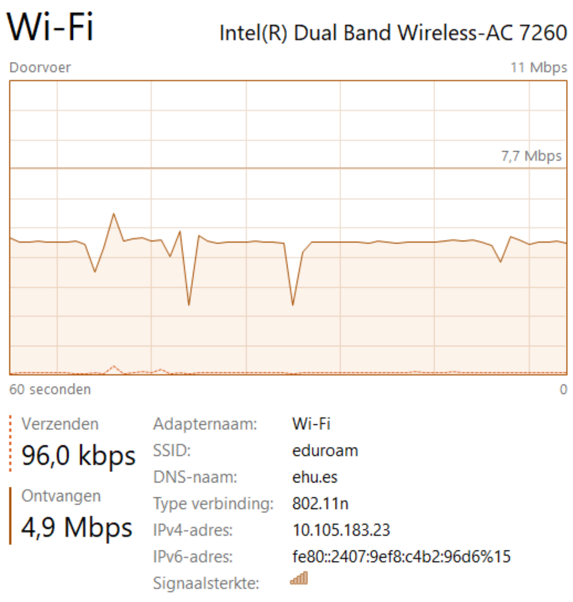
1. The communication between the XBee module at the sensor and the XBee module at the control computer. The frames considered have a 100 bits size.
2. At the control computer, the message is received and passed to the application. As shown in Fig. 8 the influence of several components make difficult measuring this time in a general purpose Operating System.
3. The generated frame is serialized and immediately transmitted by the XBee.
4. After the entire frame is received in the final XBee, the frame is analyzed and sent to the Arduino.

These steps help to understand the delay at the Zigbee communications in control applications, even in the absence of interferences. However, the designers of the control applications must analyze these steps in order to decide whether the delays and jitters may be tolerated or not for every specific application.

4.4 Interferences with WiFi

Several works have dealt with the influence of the behavior of Zigbee due to WiFi interferences [8,9]. For that reason, the last experiment investigated briefly the effect of interferences between Wi-Fi and XBee. Since both technologies work at the radio band of 2.4GHz there is a chance of interferences. To test this potential interference, large files were downloaded to the computer while the XBee application was working. The results were analyzed to check whether the dropout increased due to the interference.

A high-resolution video was downloaded while the controller was acquiring data from the sensor. During this test the message dropout was checked and saved in one file for off-line analysis. The authors measured experimentally at the computer the downloading speed, which, on average, was of 4,9Mbps (see Fig. 8).



Verzenden
96,0 kbps
Ontvangen
4,9 Mbps

Adapternaam:	Wi-Fi
SSID:	eduroam
DNS-naam:	ehu.es
Type verbinding:	802.11n
IPv4-adres:	10.105.183.23
IPv6-adres:	fe80::2407:9ef8:c4b2:96d6%15
Signaalsterkte:	

Fig. 8: Wi-Fi download speed during the test

The authors analyzed the behavior of the XBee communication at the WiFi speed peaks and observed that at the peak, up to the 60% of the messages were not received by the XBee on the computer. This value went far beyond the dropout data obtained in Test 2. The authors believe that this significant dropout was caused by interference since both XBee and Wi-Fi work at 2.4GHz.

This interference could be avoided (or at least reduced) by either using the XBees in a Wi-Fi free zone or using different XBees that work on a different frequency. Some XBee devices are able to work at 868 MHz. However, they were not available during the tests.

5 CONCLUSIONS

Wireless Networked Control Systems (WNCS) is an emergent research topic. The application of wireless technologies to control applications is expected to provide advantages in control applications, such as increasing the flexibility, reducing the wiring or getting information for sensors that are difficult to connect (e.g. those located in moveable devices). However, wireless communications are still under scrutiny since there are several challenges that must be solved. These challenges involve: (1) reliability, i.e. limiting message delays and lowering message dropouts; (2) finding the optimal sampling period, since it may affect the behavior of the control systems, having influence at the stability; (3) feeding power to the wireless devices. So, this work aims at analyzing the benefits and drawbacks of using wireless technologies in control applications.

Namely, this work analyzes the use of XBee wireless technology in control applications programmed with LabVIEW. In order to analyze these technologies a Single Input Single Output (SISO) system was considered, in which both the sensor and actuator were connected wirelessly by means of XBee modules. Wireless communications (with XBee) were compared to a wired connection made with a NI myDAQ USB-6008 board. Several experimental tests were carried out in order to find the benefits and problems of these technologies. Namely, the experiments carried out involved analyzing: (1) the performance of the Analog to Digital Converter (ADC) included at the XBee device; (2) the offset and message dropout for different distances and sampling periods; (3) the message delay at different distances and sample periods; (4) the influence of Wi-Fi interferences. The results of these tests should show if the combination of XBee and LabVIEW is adequate to build reliable WNCS, or at least to see in which situations they may become a viable alternative.

The first conclusion drawn was that the dropout rate was not so high, for the measured distances (up to 55 meters) and sampling periods (up to 30 ms), in the absence of interferences. For the tests carried out the dropout was always around 2%. It was checked that the checksum of every received message was correct. During the tests it was also proven that the flexibility of using XBee antennae is very high, since wireless technologies allow moving the devices easily at different distances.

Nonetheless there were also weak points. A higher delay was created. This delay was very sensitive to the baud-rate of the XBee configuration. However, the authors believe that a higher speed could increase the dropout. In the carried tests a general purpose operating system (Windows 10) was used, so this introduced uncertainty in the layout. The authors aim at reproducing these tests with a true real time operating system. Also, it was detected that WiFi caused serious problems of interference with XBee, since both were using the 2.4 GHz band. For that reason, the authors recommend analyzing the use XBee modules that work at 868 MHz in control systems. Another finding was that the ADC used by the XBee did not provide good results. It could be improved by connecting an Arduino board for sampling the signal. However, for some high precision applications this approach could still be poor. For these cases more precise technologies should be analyzed.

In general, it can be concluded that for non critical systems or only monitoring purposes the analyzed combination may provide a suitable solution, even though it has some limitations. However, for more critical applications several factors must be improved, either by tuning more precisely the XBee configuration, reducing the environmental interferences and using a real-time operating system to reduce and better anticipate the message delay.

Acknowledgements

This work has been supported in part by the Basque Government (GV/EJ), by means of the ELKARTEK Expedient KK-2017/00033 grant, and by means of the CONAVAUTIN agreement, signed by the local institution (Diputación Foral de Álava, DFA/AFA), the University of the Basque Country (UPV/EHU) and Aernnova Aerospace. So the authors are most grateful to these institutions.

References

[1] B. Wittenmark, K. J. Åström, and K. E. Årzen, “Computer Control: An Overview,” IFAC Prof. Br., pp. 1–82, 2002

[2] P. Park, S. C. Ergen, C. Fischione, C. Lu, and K. H. Johansson, “Wireless Network Design for Control Systems: A Survey,” IEEE Commun. Surv. Tutorials, no. c, pp. 1–36, 2017

[3] Farahani, S. Zigbee Wireless Networks and Transceivers, 1st ed.; Newnes: Burlington, MA, USA, 2008; pp. 6–12. ISBN 0750683937

[4] Hans-Petter Halvorsen, “Wireless Data Acquisition in LabVIEW,” 2011 [Online] Available: http://home.hit.no/~hansha/documents/labview/training/Wireless%20Data%20Acquisition%20in%20LabVIEW/Wireless%20Data%20Acquisition%20in%20LabVIEW.pdf [Accessed: 23-May-2018]

[5] National Instruments, “NI-VISA Overview,” 2018. [Online]. Available:

http://www.ni.com/tutorial/3702/en/ [Accessed: 23-May-2018]

[6] National Instruments, “Building a Real-Time System With NI Hardware and Software,” 2018. [Online]. Available: http://www.ni.com/white-paper/4040/en/ [Accessed: 26-May-2018]

[7] M. D. Abrignani, C. Buratti, L. Frost and R. Verdone, "Testing the impact of Wi-Fi interference on Zigbee networks," 2014 Euro Med Telco Conference (EMTC), Naples, 2014, pp. 1-6.

[8] W. Mardini, Y. Khamayseh, R. Jaradatand, R. Hijjawi, "Interference problem between ZigBee and WiFi", International Proceedings of Computer Science and Information Technology, pp. 133-138, 2012

[9] Digi, XCTU - Next Generation Configuration Platform for XBee/RF Solutions, 2018 [Online] Available: https://www.digi.com/products/xbee-rf-solutions/xctu-software/xctu [Accessed: 26-May-2018]

[10] MaxStream, “XBee TM / XBee-PRO TM OEM RF Modules,” MaxStream, no. 801, pp. 1–33, 2005.