This full text paper was peer-reviewed at the direction of IEEE Instrumentation and Measurement Society prior to the acceptance and publication.

A Wireless Instrumentation Control System Based on Low-Cost Single Board Computers

D. Enériz
Group of Electronic Desing (GDE)
University of Zaragoza
Zaragoza, Spain
eneriz.daniel@gmail.com

N. Medrano
Group of Electronic Desing (GDE)
University of Zaragoza
Zaragoza, Spain
nmedrano@unizar.es

B. Calvo
Group of Electronic Desing (GDE)
University of Zaragoza
Zaragoza, Spain
becalvo@unizar.es

J. Pérez-Bailón
Group of Electronic Desing (GDE)
University of Zaragoza
Zaragoza, Spain
jorgepb@unizar.es

***Abstract*— Automated instrumentation allows monitoring complex processes synchronizing the acquisition of physical magnitudes measured from different instruments. For this, standard communication protocols are used to send and receive both messages and data between the different components of the system. Most instrument communication protocols are based on wired standard buses, so that their application is restricted to relatively small environments, and the instruments mobility is limited. This work presents a Raspberry-based gateway which enables the wireless communication of an instrument, providing an IEEE 802.11 wireless link to Virtual Instrument Software Architecture (VISA) compatible instruments having a USB control bus. The compatibility of this proposal with the USB standard, available in a vast majority of the commercial instrumentation, allows its application in most of automated measurement systems. In addition, the use of Wi-Fi (Wireless Fidelity) as wireless protocol allows take advantage of a Wi-Fi infrastructure already deployed in the environment, or the use of a specific wireless network using a dedicated router if required. The proposed communications system has been successfully tested in a real scenario, measuring the signal propagation in a coaxial cable.**

Keywords*— *Wireless instrumentation control, single-board computer, SCPI, VISA, remote measurement

I. INTRODUCTION

In most of the automated measurement systems, the communications between the instruments and the host computer that manages the measurement process makes use of either specific instrumentation buses as GPIB (General-Purpose Instrumentation Bus) or VXI (VME extensions for Instruments), or general purpose standard buses as USB (Universal Serial Bus) or Ethernet. Nevertheless, when the monitored processes cover large areas it may be necessary to distribute the instrumentation along the whole measurement region, so that complex wired infrastructure can be required. In addition, the need of keeping the instruments connected to these busses greatly limits the instrument mobility and, hence, the flexibility in the use of the instrumentation system.

Taking advantage of the great development wireless communications standards arised in the last decades, some proposals have been made to replace the instrumentation wired buses by wireless solutions: a wireless connection provides instrument mobility, in addition to affording a more comfortable environment in the measuring area due to the absence of wired nests. In this way, in [1] a wireless communication based on IEEE 802.15.1 standard (Bluetooth) is proposed, connecting the instrumentation and the host computer by means of ad hoc interfaces to actively control up to seven RS-232-port instruments, which limits both scalability (due to the master-slave limited architecture of Bluetooth) and generality (only instruments with a limited and obsolete RS-232 port interface can be controlled). Additionally, at least two specific interfaces are needed, one to be in the host computer and the other in each instrument.

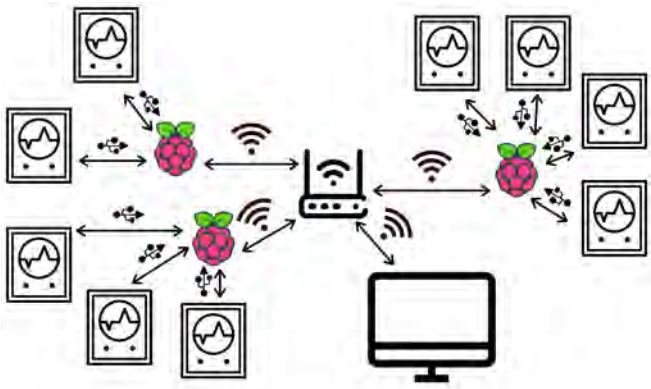

Fig. 1. Hardware connection scheme for the wireless control: Both the host computer and the Raspberry-based gateways are connected to a Wi-Fi network (previously available in the measurement area, or ad hoc). Gateways send commands and queries received from the host to the instruments through their USB buses, returning the corresponding instrument messages to the computer.

This work presents a wireless instrumentation control system using a low-cost open-source platform based on a single-board computer (SBC). The selected SBC is a Raspberry Pi Zero W including a Linux based operating system. The application running on the Raspberry to interface between the host computer and the instruments and manage the wireless communications has been developed in Python, thus keeping the open-source philosophy. Wireless communication is performed through the IEEE 802.11 standard (Wi-Fi – Wireless Fidelity), what allows managing as many instruments as necessary without the limitation imposed by other wireless standards, while keeping a good timing performance in data transfer [2]. Due to the choice of the Raspberry SBC as gateway, instruments with a USB port interface complying the VISA (Virtual Instrument Software Architecture) standard can be controlled. Due to the wide implementation of the USB standard in instruments with the

This work has been supported by projects TEC2015-65750-R (MINECO-FEDER, UE) and UZ2019-TEC-08 (University of Zaragoza)

978-1-7281-4460-3/20/$31.00 ©2020 IEEE

Authorized licensed use limited to: Universidad de Zaragoza. Downloaded on September 21,2020 at 08:13:02 UTC from IEEE Xplore. Restrictions apply.

option of computer control, its implementation in a previously deployed measurement system becomes simple and easy.

The paper is organized as follows: Section II introduces the key technologies used in this work, both at hardware and software level. Section III describes in detail the proposed SBC gateway, with special attention to data transfer between the host computer and the instruments. Section IV evaluates the use of the proposed wireless gateway powered by batteries, experimentally determining the discharge profile in realistic conditions and developing a numerical model for its estimation in real time operation. In Section V, the proposal is tested in a real scenario consisting on the measurement of the propagation velocity of an electrical signal in a coaxial cable at several distances. Finally, some conclusions are drawn in Section VI.

II. Technologies Selection

In order to design the wireless instrumentation control system (Fig. 1), several technologies must be selected, attending to its adequacy with the proposed objectives of low-cost, scalability and ease of use. These technologies can be classified as communications protocol, gateway hardware and software.

A. Communications protocol

In order to select a suitable wireless communications protocol, it is necessary to choose both the wireless standard and the transmission protocol. Concerning the wireless standard, attending to its main features the IEEE 802.11 (Wi-Fi) standard complies with the proposed specifications. Wi-Fi is based on a point-centered architecture which enables multipoint communications and larger distance transmissions compared to other widely extended communication protocols, such as the master-slaves limited architecture of the IEEE 802.15.1 (Bluetooth) standard. Due to its wide availability, the use of Wi-Fi as communications standard allows the instruments system to be connected to a wireless network already available in the measurement area. In addition, Wi-Fi allows local communications by means of hotspots (e.g. a household router), establishing a bi-directional communications channel between the instruments gateways and the host computer in charge of the measurement process. Moreover, even when the area where the instrumentation is deployed exceeds the maximum network reach (as in outdoor applications), it is possible to extend the signal range using a Wi-Fi repeater or a Virtual Private Network (VPN) when the wireless network is connected to the Internet allowing, in addition, cloud storage and the capability of distributed processing. Finally, a Wi-Fi connection allows the user to remotely control the measurement system just by knowing the IP numbers of the gateways.

Concerning the transmission protocol, the TCP [3][4] allows an easy and safe data exchanging between devices connected to the same network. Its ACK (acknowledgement) feature guarantees error-free data transmissions, so that it is widely used in most widespread network protocols, from Hyper Text (HTTP), Simple Mail (SMTP) to File (FTP) transfer.

B. Gateway hardware

The gateway is in charge of managing the operation of the instrument where it is connected, establishing the wireless link to the host computer, addressing the received commands to the instrument and sending the response data to the host using the VISA standard in the bi-directional gateway-to-instrument communication via USB. For this, a low-cost small form-factor (65mm×30mm×5mm) SBC from Raspberry [5] has been selected: the Rasperry Pi Zero W (Fig. 2, left), which main hardware features include a 1 GHz CPU, 512 MB of RAM, a 2.4 GHz Wi-Fi transceiver and antenna set. This SBC is powerful enough for our requirements, and its mini-HDMI port allows monitoring the hardware behavior in the setup stage and tracking some debugging operations in the application development phase.

Fig. 2. (left) Raspberry Pi Zero W top view. (right) Zero4U coupled to the Raspberry Pi Zero W.

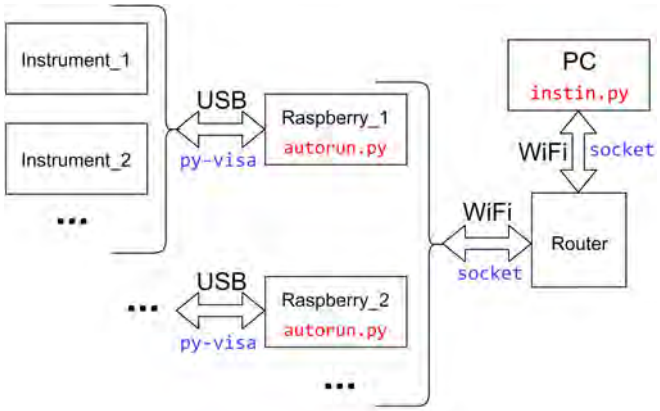

Fig. 3. Hardware connection scheme for the wireless control. The Python packages used for each task are shown in blue and the developed code is in red. The Router block represents an already deployed Wi-Fi network (if available in the measurement area), or a custom Wi-Fi using a dedicated Router (otherwise). In the first case, the full system control and operation monitoring can be performed from a computer connected to Ethernet.

The number of USB devices that can be connected to a single SBC is increased including a Zero4U USB hub (Fig. 2, right) [6] connected to the gateway. This hub expands the micro-USB port in the Raspberry module to four additional A-type USB ports. In this way, the full gateway hardware cost (Raspberry Pi Zero W and hub Zero4U) is around 30 €, so we line up with the low-cost target.

C. Software

Both the control software running on the host computer and that executed on the gateways connected to the instruments have been developed in Python, an interpreted, high-level, general purpose programming language that eases design portability and scalability, and has the required packages for developing the communication protocols and instrumentation management.

1) Host control

For the wireless communication via Wi-Fi, TCP has been selected for exchange data between the measurement host system and the Raspberry gateways. Python has the *socket* package which allows accessing to the TCP features, enabling to create connections between two Internet Protocol (IP) addresses, one acting as server and the other as client [7].

Authorized licensed use limited to: Universidad de Zaragoza. Downloaded on September 21,2020 at 08:13:02 UTC from IEEE Xplore. Restrictions apply.

In this work, in the communications process the host is configured as the client device while the gateways are configured as servers. In this way, gateways without a current open communication to the host are listening the communications channel waiting for a connection request and the host can request opening a communication channel to a specific instrument through the corresponding gateway server.

2) Instrument control

The Raspberry-based gateway runs Raspbian, a Linux-based operating system especially engineered for these devices, that includes an updated Python version. Due to the incompatibility between commercial VISA standard drivers and some Linux distributions, a customized free version, *PyVISA-Py* has been specifically developed in Python. This backend software, together the *PyVISA* package [8], enables the use of the VISA standard features in the proposed platform.

III. System Description

Fig. 3 presents a basic scheme of the system architecture. A function set including the required instrument control functions has been developed in Python for its application in the host computer. It is based on the standard commands used in measurement control, but adapting their operation to the requirements derived from the use of a TCP communication protocol under wireless technology and distributed control gateways. Functions and the required parameters have been gathered in the developed *instin.py* package.

Concurrently, a Python autorun program (*autorun.py*) has been developed to be executed into the Raspberry gateways to monitor the communications channel in search of connection requests from the host, receiving the commands, executing the instructions and sending the corresponding replies to the host queries.

A. Host management and control

The host computer controls the measurement process by means of the custom Python package *instin.py*. It has been developed to include the functions required for the wireless management of an instrument-based measurement system: sending of a communications channel open request to a specified gateway, opening of an instrument connection, sending commands and queries, receiving the corresponding replies from the instrument, and finally closing the channel. Depending on each specific task, the arguments of the *instin.py* functions will vary. For instance, opening an instrument in gateway requires the IP address of the gateway, the communications port number for the wireless link and the instrument VISA identifier. IP address and port are also needed for sending commands to an instrument, accompanied by the instruction to be sent. In this way, the host software sets up the message to be sent as a character string consisting of a 1-byte header followed by the *rest* of the message. The header act as function flag in the *autorun.py* function running on the gateways, which will use the *rest* of the received message as the argument of the function to be executed.

TABLE I. Message format

Header (1 byte)	Rest

In summary, the package *instin.py* has the same functions as any instrument control package but two extra arguments must be added to all functions for the adaption to the wireless protocol: the gateway IP address and the TCP port connection.

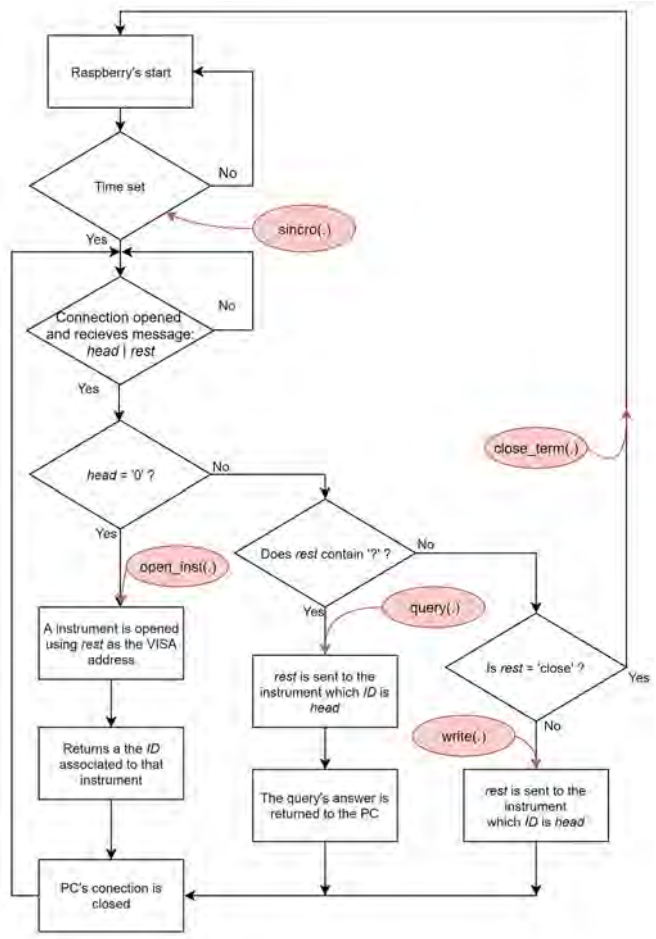

Fig. 4. Flowchart of *autorun.py*. Also, in red, the *instin.py* functions that connect to those points on the flowchart when they are called.

B. Gateway management and control

The gateway operation is controlled by the Python program *autorun.py*. This software starts its execution at the time the Raspberry is powered on. Its first task is to create a TCP server and wait for a connection request from the host. Once a connection request is received, the next operation before receiving the commands from the host is a time synchronization. After the time-set message has been received by the selected gateway, the header is separated before sending the instruction to the instrument. Depending on the character in the header, the gateway performs different operations, as it is shown in Fig. 4. If the message header is the character '0', the gateway executes the function *open_resouce(·)* from the PyVISA package, using the *rest* of the message as the VISA address of the instrument whose communication is opened, assigning it an identifier that is sent to the host for the following communications. In all the following messages with the header equal to this identifier, the *rest* of the message will be used as the command to be executed by this instrument. When the instruction includes a question mark (in the case of SCPI compatible instruments), *autorun.py* identify the command as a query and waits for an answer to be sent back from the instrument to the host. For instruments non-SCPI compliant, the system can be easily adapted modifying the *instin.py* and *autorun.py* functions: to send a query statement to the instrument, *instin.py* is modified to insert a '?' character at the end of *rest*, while *autorun.py* is

Authorized licensed use limited to: Universidad de Zaragoza. Downloaded on September 21,2020 at 08:13:02 UTC from IEEE Xplore. Restrictions apply.

configured to identify the query statement, using *rest* (without the '?' character) as the instrument command, and replying the answer to the host.

In addition, a close communication option is also implemented. This allows us to reboot the gateway when it is required.

IV. Gateway Power Consumption Analysis

The capability of controlling an instrumentation process by means of a wireless protocol can facilitate its application in environments where no mains infrastructure is available and battery-powered devices are required. Therefore, the estimation of the power consumption of the Raspberry-based gateway is of interest to allowing an accurate determination of the lifetime of the device powered by a portable battery. In this section an experimental model of power consumption for the proposed gateway is developed, next analyzing its application in the estimation of the maximum lifespan when powered by a 3.7 V-2000 mAh battery.

Power consumption is measured by periodically monitoring the remaining voltage of a battery powering a gateway that controls the measurements of an Agilent 34461A digital multimeter (DMM). By means of sending periodically query statements to the instrument from the host, the gateway performs all the operations required in a measurement process, so that a complete consumption profile can be obtained. In addition, taking advantage of the use of a DMM as a test instrument, the measurement requests correspond to the value of the remaining voltage in the battery, thus self-monitoring the process, and providing the voltage discharge curve (Fig. 5). In this way, it is possible to model the mean energy consumption of each query statement as:

$$E_i = (I_b(\nu^{-1} - t_q) + I_q t_q)\bar{V} \quad (1)$$

Where I_b is the average current biasing the gateway when it is in passive mode (i.e. waiting for a connection from the host); I_q is the average current in active mode (during the query statement process); ν is the number of queries per time unit; t_q is the duration of the peaks of current between the I_b and I_q levels; and $\bar{V}$ is the nominal voltage of the battery. The values for I_b, I_q, t_q have been experimentally acquired using a Tektronix DPO4104 oscilloscope with a Tektronix TCP0030 current probe (Fig. 6) controlled by means of the developed wireless system. However, $\bar{V}$ is a battery characteristic and ν is defined in the host control program. Thus, with the average energy consumption of a query statement, E_i, the maximum operating time can be estimated as:

$$t_{max} = \frac{E_0}{E_i \cdot \nu} \quad (2)$$

Where E_0 is the maximum energy stored in the battery. Using the voltage discharge curve (Fig. 5) and the values of I_q and I_b, it is possible to obtain the power discharge curve which, once integrated in time, will provide the energy consumed by the gateway during the measurement process. Assuming the battery is fully charged at the start, the consumed energy at the end of this process should be equal to the maximum energy stored by the battery, so that it can be experimentally estimated the value E_0 = 6.735 Wh, very close to its nominal value, namely 7.4 Wh.

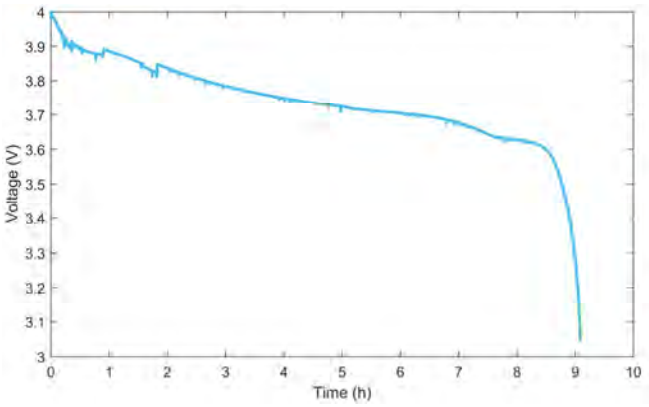

Fig. 5. Experimental battery discharge curve.

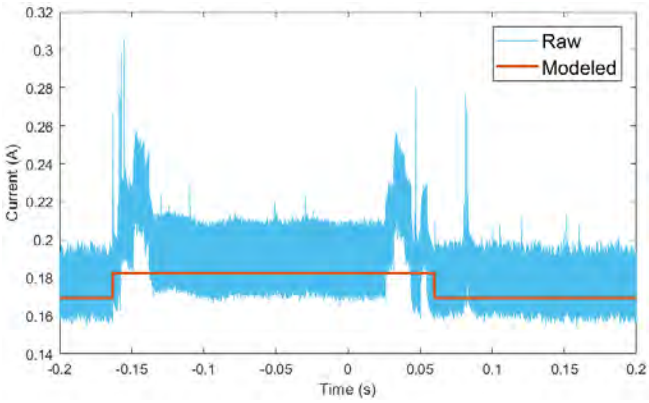

Fig. 6. In blue, the current consumption measured in a query statement. In orange, the modeled average value.

Finally, using the value for E_0, the estimated lifetime of the gateway in this measurement process is t_{max} = 9.17 h, very close to the 9.08 h experimentally measured when acquiring the voltage discharge curve (Fig. 5).

A. Generic lifespan model

In this subsection, a general model to calculate the maximum operating time of the battery is presented. Once the chosen battery for biasing the Rasperry gateway is characterized (i.e. its maximum stored energy is determined), the lifetime of the battery-powered gateway can be estimated using (2). The only parameter that must be generalized in this equation is E_i, the energy consumed by a query statement (1). In a generic measurement process, most of the time the instruments are used to sweep some physical magnitudes (either by setting or by measuring), thus E_i is the energy consumed each epoch in the measurement process. Hence, if the frequency of the loop is ν, the number of query statements per epoch is n_q, the number of write orders per epoch is n_w, the time of a query process is t_q and the time for a write is t_w, (1) is generalized to:

$$E_i = \left[\left(\nu^{-1} - n_q t_q - n_w t_w\right) I_b + \left(n_q t_q + n_w t_w\right) I_q\right] \bar{V} \quad (3)$$

V. System Operation Test

In order to verify the operation of the wireless instrumentation prototype a measurement process has been tested. The proposed solution can be useful in monitoring a distributed system since there is no need of wiring the instruments with the control host. A realistic measuring problem can be to characterize the velocity of propagation in a coaxial line at different access points. To represent this, a 11 m coaxial cable was deployed along the laboratory (Fig. 7).

Authorized licensed use limited to: Universidad de Zaragoza. Downloaded on September 21,2020 at 08:13:02 UTC from IEEE Xplore. Restrictions apply.

The input signal is provided by a Tektronix AFG3252 arbitrary function generator connected to one end of the coaxial cable. Measurements are carried out by four Agilent DSOX2002A oscilloscopes probing the cable in four different positions. The operation of the five instruments has been managed using the proposed wireless control system, connecting each of the instruments to a different wireless gateway.

Assuming the distances are known, the velocity of propagation should be easily derived from the delay between the forward and backward signals in the same point. Aimed with this, in a coaxial cable, the reflection coefficient can be described as:

$$\Gamma = \frac{Z_L - Z_0}{Z_L + Z_0} \quad (4)$$

Where Z_0 is the characteristic impedance of the wire and Z_L is the load impedance at the end [9]. Thus, when the end is an open circuit the backward signal is equal to the forward (Γ=+1, Fig. 8). Otherwise, a short circuit at the end inverts the phase of the backward signal.

Then, the delay between the forward and backward signals can be simply measured and knowing the distance from the probing points to the reflection end, the average velocity of signal propagation through the cable can be obtained. The measured values in the four probing points are about the 70 % of the light speed in the vacuum, a typical value as it is shown in [9].

VI. Conclusions

This work presents a wireless instrument control system based on the IEEE 802.11 standard for VISA compatible instrumentation. Scalability, generality and low-cost were three main goals stated in its development. The use of Wi-Fi standard and TCP protocol in the communications processes allow to easily add gateways to a previously existing system, thus increasing the number of instruments, the covered area and instruments location. The proposed system does not require to deploy a dedicated Wi-Fi infrastructure if a general purpose Wi-Fi network connected to internet is available in the area where instruments are located. In addition, it allows relocate the host computer on any site that has internet access.

Concerning the control software, it has been developed in Python, using the available *PyVISA* instrument control package and *PyVISA-Py* drivers, as well as the available Wi-Fi and *socket* package functions.

A lifespan model for a battery-powered case has been provided which should be of interest in the application of the system in environments where no mains infrastructure is available. Moreover, the proposed system has been tested in an actual example: the characterization of the velocity of propagation in a coaxial cable, controlling the measurement of a system comprising several instruments with control USB port, distributed in a large area.

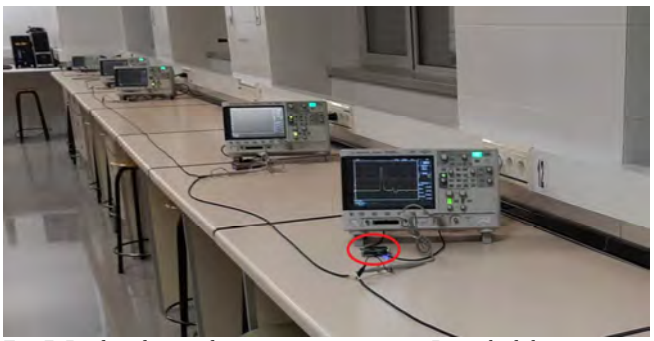

Fig. 7. Deployed coaxial wire measuring system. Remarked the gateway in the first measuring location.

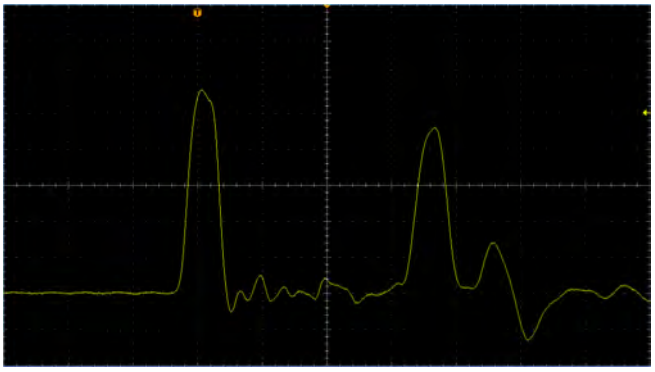

Fig. 8. Screenshot of a probing oscilloscope. It shows both forward and backward pulses.

The selection of the hardware components for this system, together the use of open-source software resources ensures the low cost of the proposed solution, currently below 30 € per gateway, but providing fully functionality for VISA compatible USB controlled instruments, widely used in laboratories and measurement environments.

References

[1] L. Ferrigno, V. Paciello and A. Pietrosanto, "A Bluetooth-based proposal of instrument wireless interface," *IEEE Transactions on Instrumentation and Measurement*, vol. 54, *(1)*, pp. 163-170, 2005. DOI: 10.1109/TIM.2004.840245.

[2] D. Raychaudhuri and N. B. Mandayam, "Frontiers of Wireless and Mobile Communications," *Proceedings of the IEEE*, vol. 100, *(4)*, pp. 824-840, 2012. DOI: 10.1109/JPROC.2011.2182095.

[3] W. R. Stevens, *TCP/IP illustrated*, vol. 1. Reading, Massachusetts etc.: Addison-Wesley, 1994.

[4] W. R. Stevens and G. R. Wright, *TCP/IP illustrated*, vol. 2. Reading, Massachusetts etc.: Addison-Wesley, 1995.

[5] Sparkfun, *Getting Started with the Raspberry Pi Zero Wireless*; 2017.

[6] UUGear, *Zero4U. 4-Port USB Hub for Raspberry Pi Zero. User Manual.* 2017.

[7] *socket-Low-level networking interface-Example*, Accessed on: Oct. 22, 2019. [Online]. Available: https://docs.python.org/3.8/library/socket.html#module-socket

[8] *PyVISA: Control your instruments with Python*; Accessed on: Oct. 22, 2019 [Online]. Available: https://pyvisa.readthedocs.io/en/latest/.

[9] W. C. Johnson, *Transmission lines and networks*, Tokyo etc.: McGraw-Hill Kogakusha, 1950

Authorized licensed use limited to: Universidad de Zaragoza. Downloaded on September 21,2020 at 08:13:02 UTC from IEEE Xplore. Restrictions apply.