20th IMEKO TC4 International Symposium and
18th International Workshop on ADC Modelling and Testing
Research on Electric and Electronic Measurement for the Economic Upturn
Benevento, Italy, September 15-17, 2014

Architecture of the Monitoring System designed for an Active Guard Rail

Giuseppe Caliano[1], Luca De Vito[2], Francesco Paolo Di Candia[2],
Gianluca Mazzilli[2], Francesco Picariello[2]

[1]*TechCon, Via F. Iandoli 18, 82100, Benevento, Italy, giuseppe.caliano@techcon.it*
[2]*University of Sannio, Corso Garibaldi 107, Palazzo Bosco Lucarelli, 82100, Benevento, Italy, {devito, dicandia, mazzilli, fpicariello}@unisannio.it*

***Abstract* – This paper describes the architecture of a monitoring system developed for a Wireless Active Guardrail System (WAGS). A WAGS is a Wireless Sensor Network (WSN) embedded on guardrails along the road infrastructure. The proposed WAGS architecture consists of four levels: *Road, Concentrator, Storage and Monitoring*. In this paper, an architecture overview of the proposed WAGS is carried out. Furthermore, a wide description of the last three levels is reported. At the road level, the measurements are performed by sensor nodes. Instead, at the *Concentrator level*, the *Concentrator* configures each sensor node and collects the measured values. The storage and the presentation of all the measurements have been realized through a multilayer server infrastructure written in Java language. In order to verify the communication protocol and the node configuration, several experimental tests have been performed and the obtained results reported.**

I. INTRODUCTION

The main target of a guardrail is to avoid that vehicles go off road. An interesting application is to embed sensors and wireless communication technologies on guardrails. In this way, the guardrail becomes an active system, which can provide information about the road infrastructure. An intelligent system for active protection mounted on guardrails has been designed in the project called "Barriera Attiva" - active guardrail [1]. The "Barriera Attiva" objective is the design and realization of an innovative guardrail based on a new concept of road safety. "Barriera Attiva" adds active functionalities to the guardrail. The active functionalities consist of a distributed measurement system for crash prevention and environmental monitoring along the road infrastructure. These functionalities are realized through an innovative Wireless Sensor Network (WSN) placed on the guardrail for monitoring the road infrastructure and for providing safety warnings to drivers. The architecture layers of the proposed WAGS are shown in Fig.1. The first layer, called Road Level, consists of a distributed measurement system, embedding two WSN node typologies: traffic safety node [3] and environmental node [4]. These nodes provide to the monitoring system and to the drivers, information about the road infrastructure by measuring the following physical quantities:

- vehicle-guardrail distance and impact, and vehicle speed, measured by the traffic safety WSN node;
- Carbon Monoxide (CO), Nitrogen Dioxide (NO2), Sulfur Dioxide (SO2) concentrations, ambient temperature and relative humidity, measured by the environmental WSN node.

Each node has an IPv6 [5] address and runs a real time operating system, TinyOS [6]. The communication between nodes is based on a mesh network, as described in [7]. The 6LoWPAN standard [8] implements the transmission of the IPv6 over the IEEE 802.15.4 [9, 10]. The second layer, called *Concentrator level*, performs the following activities: (i) storage of the measurements provided by the nodes, (ii) configuration of each WSN node, and (iii) transmission of commands to the nodes.

In order to realize these activities, a communication protocol and two software systems have been developed. The former is placed on each WSN node and it is written in nesC language. The latter is placed on a Raspberry Pi and consists of an UDP server listening on a IPv6 socket, written in C language. The measurements are periodically sent from the *Road Level* to the *Concentrator Level*. By using a web service, the measured values are sent to a server infrastructure placed on the *Storage and Presentation Level* through Internet. This server infrastructure is called *Service Center*. The *Service Center* stores the data in a database and the *Monitoring Level* shows them to the user through several media supports.

The aim of this paper is to describe the software and the hardware of *Concentrator, Storage and Presentation, and Monitoring levels*. Furthermore, in order to verify the communication protocol and the node configuration, several experimental tests have been performed. The structure of the paper is organized as it follows. In Section II, the hardware and software descriptions of each *Concentrator level* node and the communication protocol are presented. Section III describes the *Service Center*. The results obtained from an experimental validation of the protocol are reported in Section IV. Last section concludes the paper.

ISBN-14: 978-92-990073-2-7

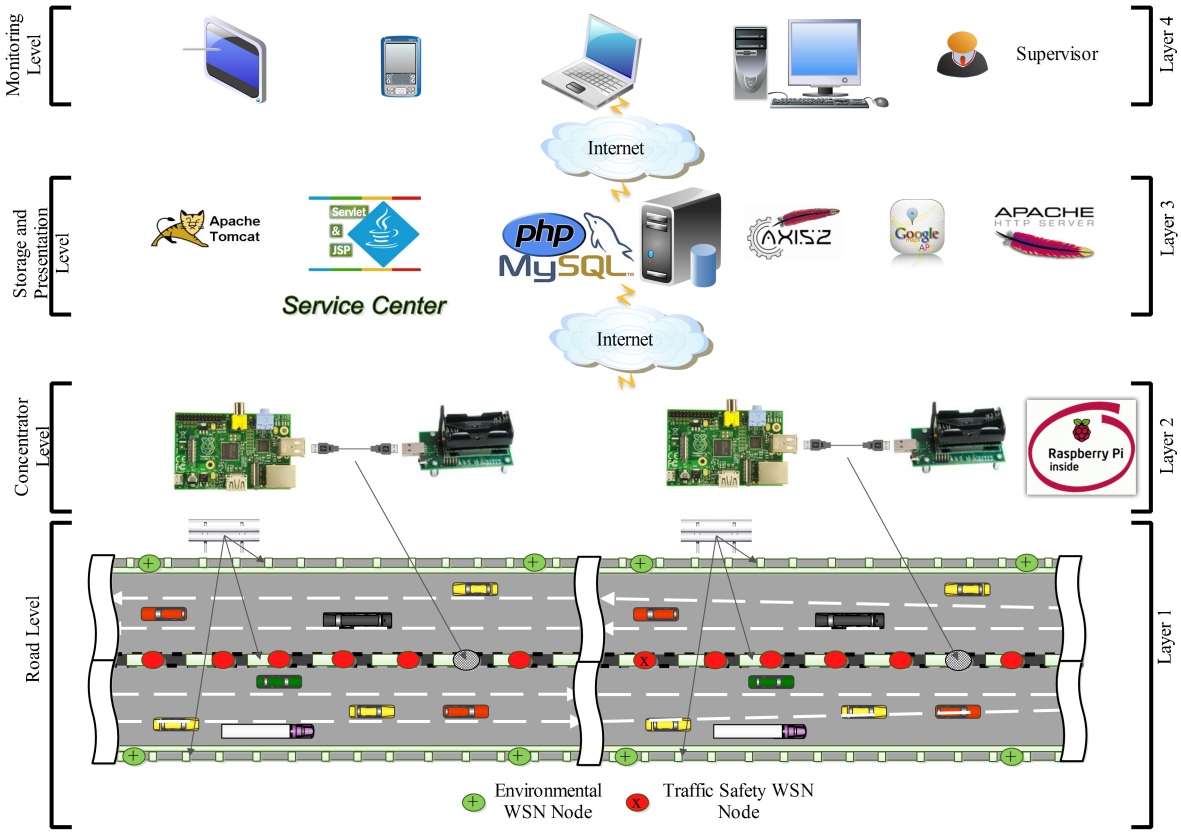

Fig. 1. Architecture layers of "Barriera Attiva".

II. CONCENTRATOR LEVEL

A. Hardware

The *Concentrator Level* consists of a Raspberry Pi [11] connected to an IRIS node via USB [7], (see Fig.2). The Raspberry Pi, is a low cost single board computer (SBC) embedded hardware based on the Broadcom 2835 system on chip (SoC), including the 700 MHz ARM1176JZF CPU and the Broadcom 250 MHz VideoCore IV GPU with a RAM of 512 MB. The system boots from the SD card where the Debian-derived Raspbian [12] is installed.

B. Software

In Fig.3, the interaction between the WSN Node and the *Concentrator* is depicted.

The aims of the *Concentrator Level* are: (i) to configure the WSN Nodes and (ii) to store the measurements obtained from the Road Level. In order to comply those purposes, a distributed software for the *Concentrator* (IP Base Station Application) and for each WSN node has been developed. The IP Base Station Application allows to receive all the configurations and periodic data packets sent by the nodes. It runs on the Raspberry Pi and consists of two software processes:

- a routing driver, listening on the usb port, that forwards the 6LoWPAN packet from and to the WSN nodes;
- an UDP server listening on a IPv6 socket bound to the 7001 port in order to communicate with the WSN node via the routing driver.

When the WSN node boots, it sends a configuration packet request to the *Concentrator* for retrieving its configuration. Once the configuration is retrieved, the WSN node starts to make measurements. In case of speed and environmental measurements, the node sends the measured values to the *Concentrator*.

C. Communication protocol

The packets exchanged between WSN nodes and the *Concentrator* can be divided in two typologies depending

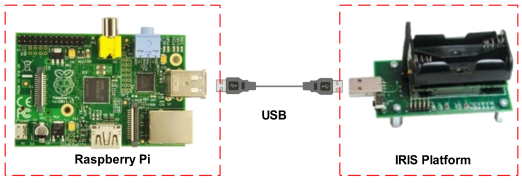

Fig. 2. The Concentrator Level hardware.

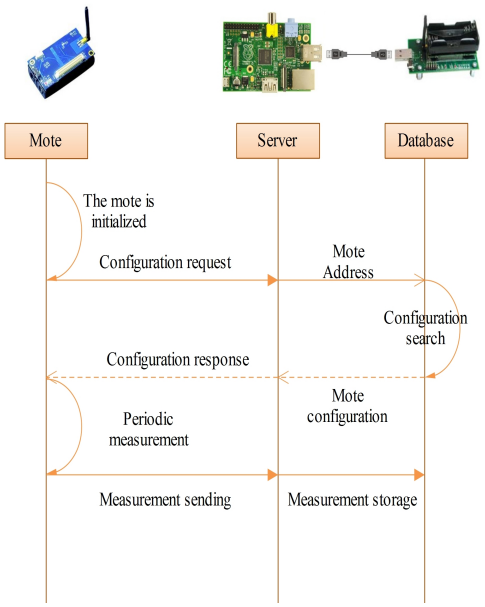

Fig. 3. The Concentrator Level sequence diagram.

on the measurement and the packet priority. For example the transmission of the impact event has a higher priority than the sending of periodic data required by speed and environmental measurements. In Fig.4, the packet structure of the message is reported. It contains three fields:

- the header section, consisting of four bytes to discriminate a configuration packet from a periodic data, to fix the priority of the packet and to distinguish

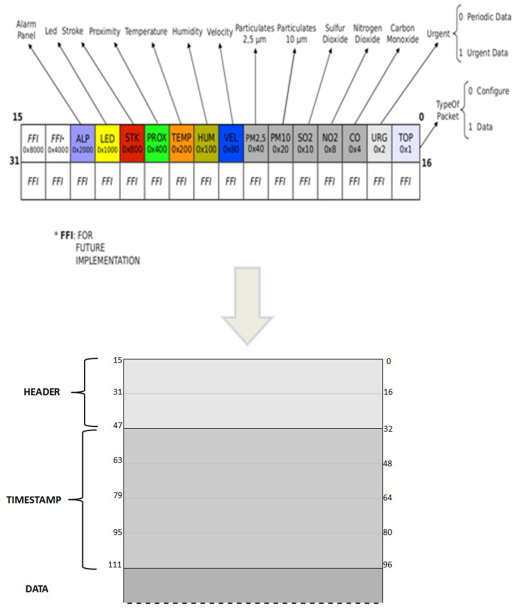

Fig. 4. Protocol structure.

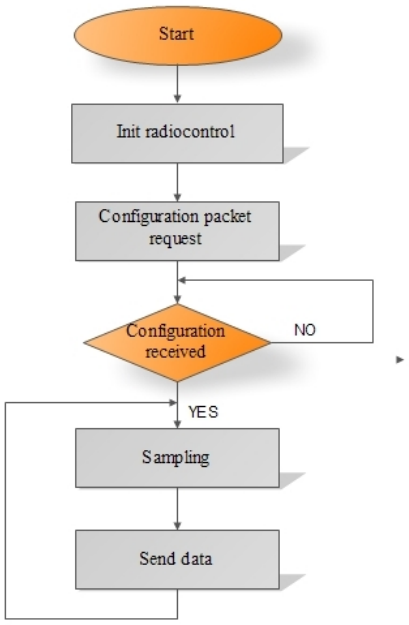

Fig. 5. The flow chart for the sensor software.

the measurement typology;
- the timestamp section, consisting of eight bytes;
- the data payload, which has dynamic dimension depending on the measurement.

The first bit of the header section is the TOP (Type of Packet), if it is set to 1 the packet is a data packet instead if it is set to 0 it is a configuration packet. The second bit is the URG (Urgent), if it is set to 1 the data packet is a periodic one instead if it is set to 0 it is an urgent one (e.g. the impact event). The third bit and the other fourteen bits represent the device measure bits.

D. WSN node: software

The WSN node firmware is written in the nesC language and is based on the TinyOS operating system. The type of WSN node is configured by setting the device measure bit. In order to receive the configuration, an UDP server listening on the 7001 port has been implemented on the node. In Fig.5, the flow chart of the software for each WSN node is shown.

When the WSN node boots, the radio on the chip is initialized and the node sends a configuration packet. This packet has: a TOP bit set to 0, URG set to 0 and the device measure bits set according to the node measurements. Once the configuration packet is sent, the WSN node is listening on the UDP server waiting for the configuration packet. When the configuration packet is received, the WSN node sets the periodic time to send the data, acquires the measurements and, if requested, sends them to the *Concentrator*, reachable through its IPv6 address.

E. Concentrator: software

On the *Concentrator*, a 6LoWPAN driver has been developed. This driver waits to receive on the USB the packets and forwards them to an UDP server. The UDP server, written in C language, is listening on an IPv6 socket bound to the 7001 port. The server accesses a database to retrieve WSN node configuration and stores the measured values. In Fig.6, the flow chart of the *Concentrator* software is depicted.

When the Raspberry Pi boots, the UDP server is automatically started and listens on an IPv6 non-blocking socket. The concurrency is regulated using a select system call [13], bound to the 7001 port. All the packets received from the WSN are inspected in the header section. According to the TOP value, the packet is recognized as *Configuration* or *Data type*. If the packet received is a configuration, the server starts a query on the database to find the node configuration.

The query is based on the WSN node IPv6 address and on the device measure bits present in the packet header section. If the node is found according to its IPv6 address, its settings are retrieved. Otherwise, if the node is not found, the default settings are retrieved and a row is added in the database with the node settings. If the received packet contains data, the measures encapsulated in the packet are stored in a local database, according to the device measure bits.

III. SERVICE CENTER

The interface of the *Service Center* allows to monitor the data received from the *Concentrator Level*. As already said, the measured quantities are environmental parameters, speed and number of transit vehicles and impact events. In order to show these measured quantities on a map, the Google Maps Geolocation API [14] has been used (see Fig.7). By clicking on a map point corresponding to a street section, it is possible to see the sensors ID, the sensors status and the measured quantities. The *Service Center* is based on a multi-layer structure, as depicted in Fig.8. It consists of *Data Layer* and *Presentation Layer*, for implementing the storage and presentation functionalities respectively.

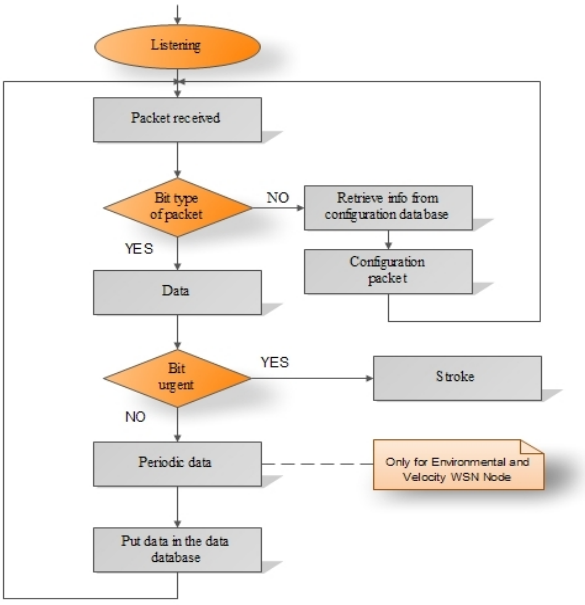

Fig. 6. The flow chart for the Concentrator software.

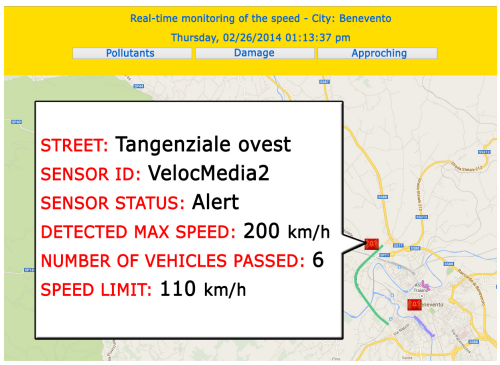

Fig. 7. Visualization of the data acquired by a speed sensor.

A. Data Layer

The *Data Layer* is based on a multi-tier architecture in which one tier is represented by the web service, used to transport data from and to *Concentrator Level*. The second tier is represented by a database system. The web service tier is implemented using the Apache Axis2 module [15]. The database tier, implemented using a MySQL DBMS [16], stores the data received from the *Concentrator Level* through web service tier and it makes them available to the *Presentation Layer*. The *Presentation*

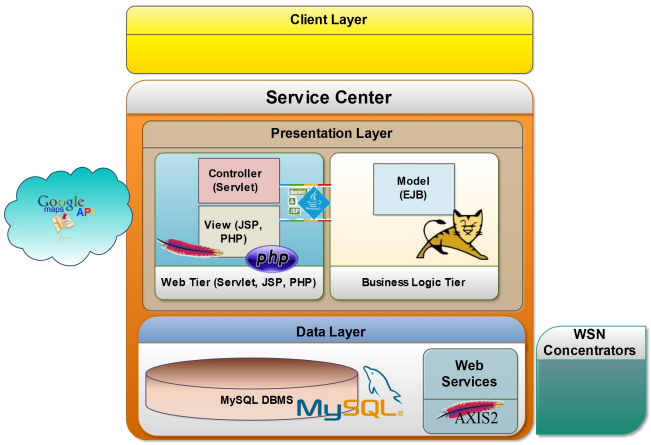

Fig. 8. Data and presentation layer.

Layer is based on J2EE [17] technology and consists of two tiers: the "Web Tier" and "Business Logic Tier". They implement the Model-View-Controller (MVC) [18] architectural pattern.

B. *Presentation layer*

When an user logs on the Service Center, through a web browser, jsp pages show the measured data from the sensors and the sensors status. The jsp page request is processed by the servlet and the data are encapsulated and transported through JavaBeans [19]. It follows a brief description of each *Presentation tier*. The Web Tier was developed through a Web Server Apache Tomcat [20] and contains, referring to the already mentioned MVC pattern, the following software components:

- View: jsp pages that allow the presentation of the data stored on the *Data layer* on cartographic maps using Google Maps Geolocation API. Php pages used for the configuration of the *Service Center* and the access to the configuration data via a MySQL connector.
- Controller: the java servlet is involved in the sorting of the data traffic between the *Presentation Layer* and the *Data Layer* through the Business Logic Tier.

The Business Logic Tier was developed through a Tomcat Application Server and, referring to the MVC pattern, contains the Model implemented by JavaBeans components.

IV. PROTOCOL VALIDATION

In order to validate the software developed on the nodes and *Concentrator*, a benchmark scenario has been designed, as shown in Fig.9.

The experimental setup consists of three WSN nodes placed at a mutual distance of 4 m. The distance between each node and the *Concentrator* is of 8 m. Furthermore, the *Concentrator* communicates with a computer via

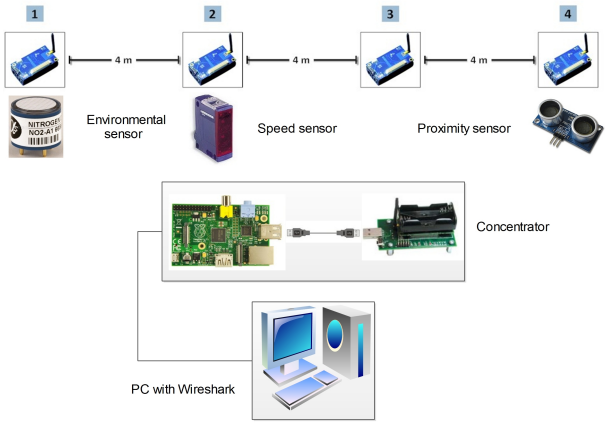

Fig. 9. Test setup.

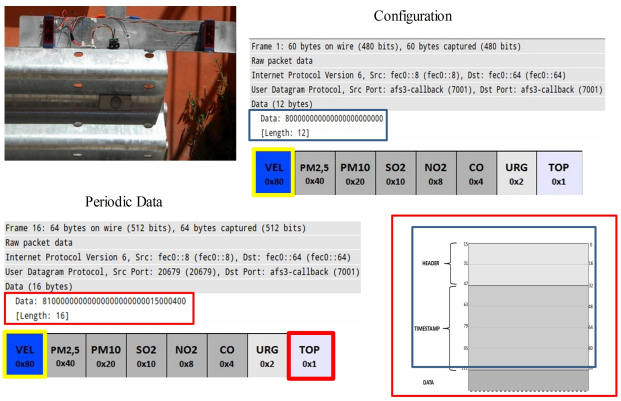

Fig. 10. Speed WSN node Wireshark capture.

ethernet. The first WSN node is an environmental sensor that measures SO2, NO2, CO. The second one is a speed sensor that measures medium speed and the third one consists of a proximity sensor. The first two WSN nodes send and receive both configuration packets and data packets, while the third one requires only the configuration packet. By using the TCPdump tool [21], the packet sent and received from the *Concentrator* is stored on a logging file on the Raspberry Pi SD card. TCPdump is a command line tool that provides to the PC the logging file, via ethernet. By using the Wireshark tool [22], the contents of this file are shown to the user.

In Fig.10, the experimental test results for the speed sensor are reported. As it is possible to see in the Configuration packet, the TOP bit is set to 0 and the VEL bit is set to 1 as expected. In the Periodic Data packet, TOP bit is set to 1 and the data section contains the average value of the vehicle speeds.

The same results was obtained for the environmental sensor, as shown in Fig.11. As it is possible to see in the Configuration packet, the TOP bit is set to 0 and the SO2,

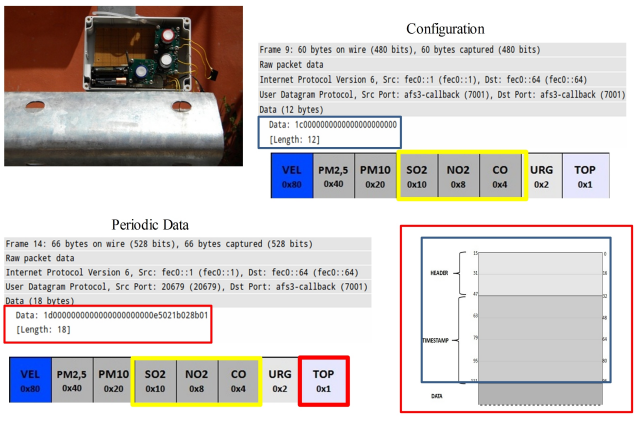

Fig. 11. Environmental WSN node Wireshark capture.

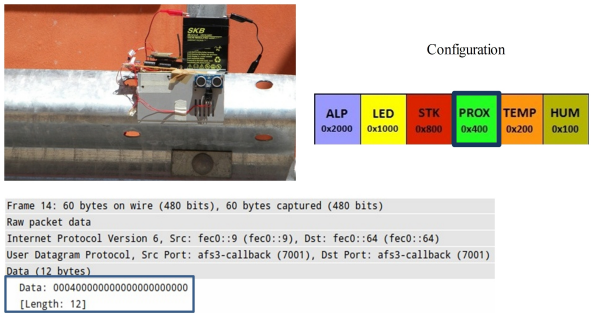

Fig. 12. Proximity WSN node Wireshark capture.

NO2, CO bits are set to 1 as expected. In the Periodic Data packet, the TOP bit is set to 1 and the data section contains the SO2, NO2, CO values.

Fig.12 depicts the test results for the proximity sensor. As it is possible to see in the Configuration packet, the TOP bit is set to 0 and the PROX bit is set to 1 as expected. The data periodic packet is not sent. The reported results comply with the expected behaviour of the protocol.

V. CONCLUSIONS

In this paper the architecture overview of a WAGS was presented. Furthermore, the hardware and software of each WSN node and the communication protocol were described. A description of the *Service Center* that implements the user interface was reported. In order to validate the protocol and the software, experimental tests were performed. Three scenarios were investigated, using a speed, an environmental, and a proximity WSN nodes. All the test results comply with the expected protocol behaviour.

ACKNOWLEDGMENT

The paper has been supported from Grant no. PON01_03100 "Barriera Attiva *proposal target is the ideation, design and realization of an innovative road barrier based on a new concept of safety, in which are combined various technological innovations (active functionalities) with the traditional structural function (passive functionalities) creating an active system for crash prevention and elimination"* financed by the Italian Ministry of Education, University and Research [1].

REFERENCES

[1] Project PON01_03100, "Barriera Attiva". Available:http://www.barrieraattiva.unisannio.it.

[2] P.Daponte, L.De Vito, F.Picariello, S.Rapuano and I.Tudosa, "Wireless sensor network for traffic safety". Proc. of IEEE Workshop on Environmental, Energy, and Structural Monitoring Systems, Perugia, Italy, 28 Sept. 2012, pp. 42-49.

[3] P.Daponte, L.De Vito, S.Rapuano and I.Tudosa, "Wireless active guardrail system for traffic safety: preliminary tests of speed and proximity measurement". Proc. of Int. Instr. and Measurement Technology Conference (I2MTC). Minneapolis, 6-9 May 2013, pp. 737-742.

[4] L.De Vito, V.Cocca, M.Riccio and I.Tudosa, "Wireless active guardrail system for environmental measurements". Proc. of IEEE Workshop on Environmental, Energy, and Structural Monitoring Systems, Perugia, Italy, 28 Sept. 2012, pp. 50-57.

[5] Internet Protocol, Version 6 (IPv6) Specification. Available:http://tools.ietf.org/html/rfc2460.

[6] TinyOS. Available:http://www.tinyos.net.

[7] IRIS mote. Available:http://www.memsic.com.

[8] 6LoWPAN standard. Available: http://www.cs.berkeley.edu/ jwhui/6lowpan.html.

[9] P.Daponte, L.De Vito, F.Picariello, S.Rapuano, I.Tudosa, "Prototype design and experimental evaluation of wireless measurement nodes for road safety", Measurement, vol. 57, Nov. 2014, pp.1-14.

[10] P.Daponte, L.De Vito, G.Mazzilli, S.Rapuano and I.Tudosa, "Network Design and Characterization of a Wireless Active Guardrail System". Montevideo, Uruguay, 12 May 2014.

[11] The Raspberry Pi. Available: http://www.raspberrypi.org/.

[12] The rapbian os. Available:http://www.raspbian.org/.

[13] The man for the C system call select. Available:http://www.unix.com/man-page/freebsd/2/select/.

[14] The Google Maps Geolocation API. Available: http://developers.google.com/ .

[15] The Apache Axis module. Available: http://ws.apache.org/axis/.

[16] The MySql server. Available: http://www.mysql.com/.

[17] The Java 2 Enterprise Edition. Available: http://java.sun.com/j2ee.

[18] The Model View Controller Pattern. Available: http://en.wikipedia.org/wiki/Model-View-Controller.

[19] The Enterprise JavaBeans. Available: http://java.sun.com/products/ejb/docs.html.

[20] The Apache Tomcat module. Available: http://jakarta.apache.org/tomcat/.

[21] The TCPdump sniffing tool. Available:http://www.tcpdump.org/.

[22] The Wireshark sniffing tool. Available:http://www.wireshark.org/.

Index of Authors

D

L

M

W

Y

Z