WSN Architecture for Smart Irrigation System

Trifun Savić, Milutin Radonjić, *Member, IEEE*

***Abstract* — In this paper, an architecture of a ZigBee wireless sensor network for application in smart irrigation system is proposed. After a brief overview of the most important features of ZigBee standard related to Wireless Sensor Network (WSN) operation, Arduino-based sensor node for acquisition of soil moisture and air temperature is described. This node is equipped with XBee S2 communication module in order to be part of the established WSN. Furthermore, we presented the reception of the collected data on the main control unit, designed on the Waspmote microcontroller platform equipped with XBee S2 coordinator. All necessary steps required to establish the communication between Waspmote platform and XBee S2 coordinator device are described, even though Waspmote platform is designed to communicate only with Xbee routers and end devices. Proposed architecture allows flexible implementation with reliable data transfer that provides necessary information for autonomous decision-making in smart irrigation system.**

***Keywords* — Agriculture, irrigation, Waspmote, WSN, ZigBee**

I. Introduction

PROPER irrigation of crops is one of the most important aspects of the yield and product quality. If the plant does not receive enough moisture in the soil, it dries and veins. On the other hand, too much moisture is suitable for the appearance and development of plant diseases. Also, unnecessary irrigation significantly wastes water resources.

Globally, throughout the last 50 years, irrigation development helped alleviate poverty by creating employment opportunities, lowering food prices, and increasing the stability of farm output [1]. However, the growth of the population worldwide, causing larger food demands, is increasing water consumption and puts a lot of pressure on the water management systems. In that context, there is an urgent need for improving available irrigation systems and developing new, more efficient ones.

Therefore, we performed multidisciplinary research in implementation of state of the art microcontroller platforms, realization of communication systems and creation of algorithm for smart irrigation, with the aim of implementing a system that will ensure an increased yield and quality of agricultural products by applying optimal irrigation [2]. Based on the input parameters, which are determined by soil analysis, and obtained by the sensor nodes, the algorithm autonomously determines the optimal periods for irrigation.

In this context, the sensor node for measuring soil moisture and air temperature on the agricultural field has been developed. Measured values from the sensor nodes had to be reliable transferred with minimal power consumption, regardless of their position on the field and distance from the central control unit of the system. The Wireless Sensor Network (WSN) imposed itself as a logical choice, as it provides all the required conditions.

In this paper, we present the WSN architecture implemented in our smart irrigation system. Proposed architecture comprises of Arduino-based sensor nodes for measuring soil moisture tension and air temperature, equipped with ZigBee communication modules. These sensor nodes transfer the measured parameters to the network coordinator implemented on the main control unit of the system, which is based on Libelium Waspmote microcontroller platform [3].

The rest of the paper is organized as follows. In section II we provide a brief overview of ZigBee standard for wireless sensor networks. Architecture of proposed WSN together with implementation detail of sensor nodes and network coordinator is presented in section III. Final conclusions and future directions are given in section IV.

II. ZigBee Wireless Sensor Network

WSN can consist of a variety of sensor nodes with different types of sensors. The concept of micro-measurement, along with wireless communication, provides a huge variety of possible applications.

Among the most popular types of wireless sensor networks, with low consumption and low speed data transfer, are networks based on ZigBee standard. This standard defines higher-layer protocols in the protocol stack and has a very wide application in wireless low-power sensor networks with affordable prices.

ZigBee standardizes higher functional levels in the protocol stack, of both application and network layer [4]. The application layer provides a framework for application development and communication between application objects. The network layer organizes the network and allows packet routing. The network layer of the ZigBee standard relies on the IEEE 802.15.4 standard that defines the link layer and physical layer. A schematic representation of the ZigBee standard protocol stack is shown in Fig. 1.

This work has been supported by the Ministry of Science of Montenegro and the HERIC project through the BIO-ICT Centre of Excellence (Contract No. 01-1001).

Corresponding Trifun Savić is with the University of Montenegro, Faculty of Electrical Engineering, Bul. Džordža Vašingtona bb, 81000 Podgorica, Montenegro (phone: 382-20-245839; e-mail: trifuns@ac.me).

Milutin Radonjić is with the University of Montenegro, Faculty of Electrical Engineering, Bul. Džordža Vašingtona bb, 81000 Podgorica, Montenegro (phone: 382-20-245839; e-mail: m.radonjic@ieee.org).

978-1-5386-3620-6/18/$31.00 ©2018 IEEE

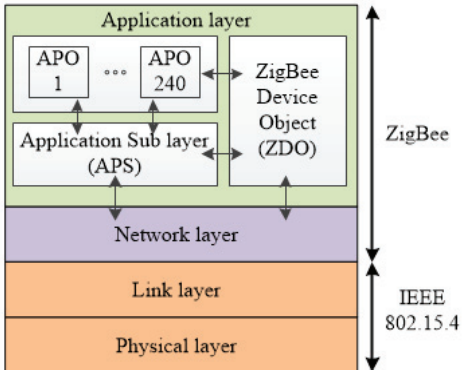

Fig. 1. ZigBee standard architecture of functional layers and protocol stack [4]

The application layer of the ZigBee standard consists of application objects that are deployed on sensor nodes. These objects represent software that controls hardware components available on the device (sensors, actuators, etc.). One ZigBee application can have up to 240 application objects, which can also be represented as end devices. Each application object is assigned locally a unique number, which other objects use as the address extension of that device for easier connection and retrieval in the network. ZigBee device object is a special object that offers services to application objects. It enables them to find devices in the network and services that these devices offer.

The main feature of the network layer is creation of network topology in communication between sensor nodes in the network. At the network level, ZigBee stack protocol defines three types of nodes: *end device*, *router* and *coordinator*. The *end device* is the simplest type of device in the ZigBee network, which can't allow other devices to join the network, nor can help routing data in the network. *Router* has the ability to route packets in the network. A device that creates and manages the network is called the *coordinator*. It can also participate in packet routing in mesh topology networks. *Routers* and *end devices* can join the network only after the coordinator creates it.

The link level of the ZigBee device, together with the physical level, is defined by the IEEE 802.15.4 standard. This standard defines two types of devices: Reduced Function Device (RFD – only *end device*) and Full Function Device (FFD – *end device* or *coordinator*).

The physical layer controls the activation/deactivation of the radio module, receiving and sending data, performs channel frequency selection, checks whether the communication channel is busy, or whether some data is currently being sent to the communication channel.

The construction of a WSN mainly depends on the type of application. For this reason, the choice of topology and network routing protocol must be tailored to its purpose. ZigBee technology provides low-power solutions for many IoT applications [5]. For our application in agriculture, the data transfer speed is not very important, but the energy and lifespan of the network should be given higher importance. Therefore, we chose to develop WSN based on the ZigBee standard.

III. Architecture of Proposed Wireless Sensor Network

The proposed smart irrigation system is designed to autonomously decide on the moment of activation and deactivation of electromagnetic valves for releasing water on the agricultural field. The main control unit manages the process of autonomous decision-making. Thus, in addition to automated control of the irrigation process [6], the user can configure an autonomous operation of the system. In this mode of operation, the appropriate decision on irrigation intervals is made by smart irrigation algorithm, based on the ambient temperature and soil moisture obtained over the ZigBee WSN. Illustration of the proposed smart irrigation system is shown in Fig. 2.

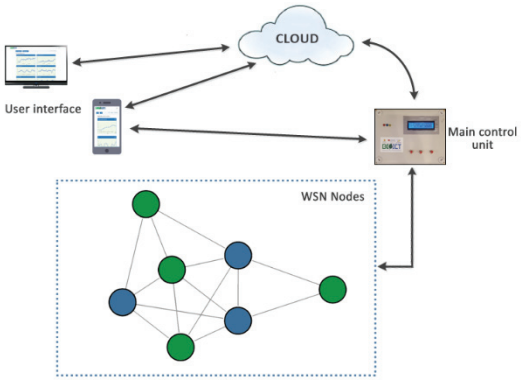

Fig. 2 Illustration of the proposed smart irrigation system

A. *Arduino-based wireless sensor node*

The wireless sensor node for measuring the humidity of the soil and the air temperature consists of the following components:

- Arduino Uno microcontroller platform
- Watermark 200SS soil moisture sensor
- DHT11 air temperature and humidity sensor
- XBee S2 communication module

Prototype of our sensor node is depicted in Fig. 3.

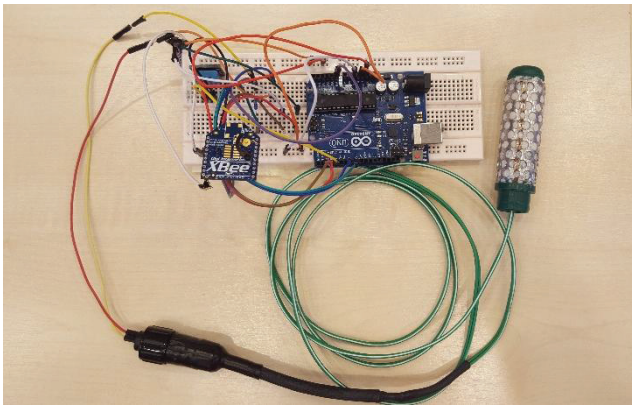

Fig. 3 Prototype of the wireless sensor node for soil moisture and air temperature measurement

The Arduino Uno microcontroller platform is the base of the sensor node. It communicates with sensors, processes measured parameters and sends them to the main control unit via the ZigBee WSN.

First step is to perform measurement of the significant parameters. In our implementation, the Watermark 200SS

sensor has been used to measure soil moisture [7]. This sensor belongs to the type of gypsum sensors, which measure the soil moisture potential, i.e. tension. As tension changes with water content, the resistance of Watermark 200SS sensor changes as well. Since the sensor is part of the electric circuit, the measurement of resistance can be performed as the measurement of the voltage. Sensor is connected to a 10-bit A/D converter, which is part of the Arduino platform. For the correct operation of this sensor, it is necessary to provide an alternate (AC) power supply as explained in [8], to avoid polarization of the sensor during long-term exposure to DC power supply.

In our implementation of the smart irrigation system, information about air temperature is used to avoid the system operation at low temperatures. If the temperature drops below limit, the algorithm prevents the irrigation process from causing damage to the system itself and freezing water in the irrigation pipes. The DHT11 sensor was used to measure air temperature. Reading of the measured values from this sensor is performed using the DHT library [9].

As mentioned above, communication between the sensor node and the main control unit is performed using XBee S2 communication modules. These modules use ZigBee standard, which can allow direct data transfer to the coordinator (star topology), but also provide the possibility of packet routing, and thus the creation of other network topologies, such as tree and mesh topology. In our implementation we chose the mesh topology**.**

The Arduino platform is connected to the XBee S2 communication module via the UART serial interface. The configuration of the XBee S2 module is performed using XCTU software, developed by the DIGI Company that products XBee modules [10]. In our implementation, the XBee S2 communication module on the sensor node is configured to send the data to the coordinator in AT (Transparent) mode [11]. This means that any data Arduino platform sends to the XBee S2 module via the serial interface is automatically forwarded over the communication channel to the coordinator located on the main control unit.

After obtaining required information from aforementioned sensors, data have to be processed and prepared to be sent as a part of the communication message. Next to these parameters, message also has to contain a unique ID of the corresponding sensor node.

In order to create a communication message, it is necessary to convert information about soil moisture and air temperature from the float data type into a string. After the conversion, a message is forwarded to the XBee S2 module, via the built-in UART port on digital pins D0 (Rx) and D1 (Tx) of the Arduino Uno platform. Following code illustrate required steps:

```
/* Arduino serial port baud rate */
Serial.begin(9600);
/* Conversion from float to string */
FloatToString(S, soil_moisture);
FloatToString(T, temperature);
/* Message to be transferred to main unit */
sprintf(message,"%s?%s!%s#",
  id, soil_moisture, temperature);
/* Sending the message to XBee */
Serial.write(message);
```

From the previous code, one can see that message with the measured sensors data contains characters '?', '!' and '#'. These characters are used to ensure easier parsing of the message on the coordinator (receiver) side, i.e. at the main control unit. First block of the message contains the ID of the sensor node and it is followed by character '?'. The character '!' denotes the end of the second block of the message that carries information about the soil moisture, while the character '#' denotes the end of the third block that contains information on the air temperature.

B. Receiving data on the main control unit

In our implementation of the smart irrigation system, it was necessary to implement WSN coordinator within the main control unit to avoid an additional device (gateway) that will collect data and forward it to the control unit. The idea is that the reception of data from the sensor nodes and the smart irrigation algorithm are performed on the same device.

As mentioned before, the main control unit of the smart irrigation system is implemented using Waspmote platform. This platform has an XBee socket that uses the UART0 serial port. This socket was expected to be used for the purpose of establishing communication with the Waspmote platform. However, according to the official Libelium forum, the XBee socket on the Waspmote platform is not intended to communicate with the XBee coordinator device, but only with XBee routers and end devices. Thus, Waspmote devices are essentially envisaged to be client devices, i.e. sensor nodes that are used to collect and send data. Even if it would be possible to use the embedded XBee socket, due to the size of the overall program code implemented on the main control unit, the use of Libelium API libraries to communicate with XBee devices would not be optimal. Due to the limited resources of the microcontroller in terms of free memory space, it was necessary to implement different communication approach. This was performed using directly the UART0 serial port of the Waspmote platform, without using the entire XBee socket and interface, but only the power pins (3.3V and ground) and the serial communication pins (Tx and Rx). To enable this communication, the *XBeeOn()* function, represented by the following code, is implemented:

```
Utils.setMuxSocket0();/* mux output to Skt0 */
beginSerial(9600, UART0); /* baud rate */
/* Setting the output pins for Socket0 */
pinMode(XBEE_PW, OUTPUT);
pinMode(MUX_PW, OUTPUT);
pinMode(MUX_USB_XBEE, OUTPUT);
/* Enable power supply of the multiplexer */
digitalWrite(MUX_PW, HIGH);
digitalWrite(MUX_USB_XBEE, HIGH);
/* Enable 3.3V on the XBee socket */
digitalWrite(XBEE_PW, HIGH);
/* Update the Waspmote register */
WaspRegister |= REG_SOCKET0;
serialFlush(UART0);/* Flush UART0 buffer */
```

The specified *XBeeOn()* function is executed in the

setup() function of the Waspmote program. Then, it is possible to read the UART0 port looking for the message that the XBee coordinator receives from the sensor nodes.

Receiving and parsing the message from the sensor node is performed within the *loop()* function of the main program through the following program code:

```
/*Checking if UART0 serial port is available*/
if (serialAvailable(UART0)){
    /* Reading the serial port */
    node_message = serialRead(UART0);
    if(ind_wsn==0) {
     /* Information about node ID */
        sensor_node_id[a++] = node_message;
  /* Looking for the end of the first block */
        if(node_message == '?') {
            sensor_node_id[a-1]='\0';
            /* String to float conversion */
            node_id =
string_to_float(sensor_node_id);
            /* Reset the index variable */
            a=0;
/* Setting the flag for the second block */
            ind_wsn=1;
    if(ind_wsn==1) {
/* Reading the second block of the message -
soil moisture tension */
        sensor_node_moisture[a++] =
node_message;
 /* Looking for the end of the second block */
        if(node_message == '!') {
            sensor_node_moisture[a-1]='\0';
            /* String to float conversion */
            soil_moisture =
string_to_float(sensor_nod_moisture);
            /* Reset the index variable */
            a=0;
    /* Setting the flag for the third block */
            ind_wsn=2;
        }
    }
    else if(ind_wsn==2) {
/* Reading the third part of the message - air
temperature */
        sensor_node_temperature[a++] =
node_message;
        /* Checking if the end of the third
part of the message is reached */
        if(node_message == '#') {
            sensor_node_temperature[a-1]='\0';
            /* String to float conversion */
            air_temperature =
string_to_float(sensor_node_temperature);
            /* Reset the index variable */
            a=0;
    /* Setting the flag for the first block */
            ind_wsn=0;
 }  }   }
```

First, we check if there is any data at the UART0 serial port, by the *serialAvailable* function, so the message can be received. If the data at serial port is available, its reading is performed. During each iteration through the *loop()* function, one byte of the message is read and written to the corresponding string. As already mentioned, the first block of the message contains information on ID of the sensor node, the second block of the message contains information on the soil moisture, while the third block of the message contains information on the air temperature. In order to track block of the message which is being parsed at a given moment, the flag *ind_wsn* is invoked.

If the flag is equal to zero, the program is reading the ID of the sensor node, until '?' character. Thereafter, string variable is terminated and index is reset for the next parsing. Finally, *ind_wsn* flag is set to one, for the reading of the second block of the message. Similar processing is performed for the second block (delimiter is '!' character) and the third block (delimiter is '#' character). Next to it, the information on the measured soil moisture and ambient temperature is converted to the float data type in order to be further processed by the smart irrigation algorithm.

IV. Conclusion

In this paper we presented the architecture of the wireless sensor network implemented in our smart irrigation system. WSN was designed using communication modules with ZigBee standard. This standard provides low power consumption and it was very suitable in this application, where the scalar data are transferred through the network with low latency. We described the Arduino-based sensor node for soil moisture and air temperature measurement, used in this implementation. We also described implementation of the main control unit related to data acquisition of the sensor nodes. The proposed architecture is experimentally verified. The future work will be oriented to investigation of other features, such as power consumption reduction and minimization of the sensor nodes.

References

[1] C. De Fraiture, D. Molden, and D. Wichelns, "Investing in water for food, ecosystems, and livelihoods: An overview of the comprehensive assessment of water management in agriculture," *Agricultural Water Management,* vol. 97, no. 4, pp. 495-501, April 2010.

[2] T. Savic, B. Bajagic, M. Knezevic, and M. Radonjic, "Method for Measuring Released Amount of Water in Smart Irrigation System," in *Proc. 22nd Conference on Information Technologies IT 17*, 2017, pp. 161 – 164.

[3] "Waspmote datasheet," [Online]. Available at: http://www.libelium.com/downloads/documentation/waspmote_datasheet.pdf [Accessed: 05.11.2017.]

[4] P. Baronti, et al. "Wireless sensor networks: A survey on the state of the art and the 802.15. 4 and ZigBee standards," *Computer communications*, vol. 30, no.7, pp. 1655-1695, May 2007.

[5] Elkhodr, Mahmoud, Seyed Shahrestani, and Hon Cheung. "Emerging wireless technologies in the internet of things: a comparative study." arXiv preprint arXiv:1611.00861 (2016).

[6] T. Savic, and M. Radonjic, "Proposal of Solution for Automated Irrigation System," in *Proc. of 24th Telecommunication Forum TELFOR 2016*, 2016, pp. 647-650.

[7] "200SS Watermark datasheet," [Online]. Available at: http://www.irrometer.com/pdf/sensors/403%20WATERMARK%20Sensor-WEB.pdf [Accessed: 06.11.2017.]

[8] V. Radman, and M. Radonjić, "Arduino-based System for Soil Moisture Measurement," in *Proc. 22nd Conference on Information Technologies IT 17*, 2017, pp. 289 – 292.

[9] "DHT library," [Online]. Available at: https://github.com/adafruit/DHT-sensor-library/blob/master/DHT.h [Accessed: 10.11.2017.]

[10] "XCTU," [Online]. Available at: https://www.digi.com/products/xbee-rf-solutions/xctu-software/xctu [Accessed: 16.11.2017.]

[11] R. Faludi, *Building wireless sensor networks: with ZigBee, XBee, arduino, and processing.* O'Reilly Media, Inc., 2010.