REAL TIME MONITORING OF ANALOG AND DIGITAL SENSORS

Cristina Gabriela SĂRĂCIN[1], Dan BELIBOV[2]

The paper presents the way of designing a real-time monitoring system of the information received from analog and digital sensors. The system consists of a microcontroller, various sensors, a network adapter and a cloud server. The measurements from sensors are saved on the server for further data processing. The main feature of the system is scalability. This function is presented throughout the paper through examples and options for further development of the system. The advantage of implementing this system is the possibility of using it in any field.

Keywords: real-time monitoring, microcontroller Arduino, server cloud, Internet of Things

1. Introduction

This paper proposes a method of building a system based on a microcontroller, sensors, network adapter (ESP8266) and a cloud server. The originality of the paper consists in the implementation of the cloud server application and its communication with any type of microcontroller that is possible made through simple adaptations of the C language libraries. The server allows saving data such as real-time monitoring of the information received from the sensors and its processing. More recently, cloud concepts (cloud storage and cloud server) [1, 2, 3] and IoT (**I**nternet **o**f **T**hings) [4] have become increasingly popular. The proposed system can be used for didactic purposes and subsequently developed for commercial purposes.

Obtaining data from sensors (analog and digital) is a priority in many industrial and non-industrial branches. Sensors are present in factories, warehouses, shops, cars and, last but not least, in homes. In this context, there will always be a need for optimization, continuous development, and scalability of systems containing these sensors [5].

In this paper, we have developed a system based on an Arduino Uno microcontroller [6, 7, 8], sensors (temperature, sound and motion) and an ESP8266 serial Wi-Fi adapter [9]. The system described in this paper can be

[1] Associate Professor, Dept. of Electrical Engineering, University POLITEHNICA of Bucharest, Romania, e-mail: cristina.saracin@upb.ro

[2] Student, Automatic Control and Computer Science, University POLITEHNICA of Bucharest, Romania, e-mail: belibovd@gmail.com

constructed with any type of microcontroller and any type of sensors. Also, a cloud server based on HTTP protocol (or HTTPS for secure connection) has been created to receive measurement data from the system, having several such systems connected in parallel. The server can also inform the microcontroller about the need to perform a particular action.

2. Description of the system

To read and process data from the sensors, any microcontroller can be used. Primary data processing means the transformation of results into international measurement units. Data storage for monitoring or subsequent processing involves connecting to a network with internet access or a closed network for data security.

The DHT11 sensor was used to measure the temperature and humidity and 2 digital sensors (LM393 or PIR HC-SR501) to determine the presence of sound and motion.

Aiming for the smallest possible size on the system (4x3x3 cm), a serial Wi-Fi adapter (ESP8266-01) was used.

Most Wi-Fi modules that use a serial interface to communicate with the microcontroller, do so at a 115,200 bps baud rate. This speed is not available on the Arduino microcontroller and therefore no data was received or lost at the first attempt. For this reason, it was necessary to rewrite the ROM memory of the ESP8266 card to communicate at 9600 bps. In this case, even if due to low transmission bandwidth we lose processing speed, we greatly gain data integrity, with low error rates. The communication speed between the microcontroller and the ESP8266 module depends on the microcontroller used and it can be obtained through a change in the driver used. To install the package, specialized utility software will be needed. It is available for Windows, Linux and Mac OS (ESPFlashDownloadTool and XTCOM_UTIL). There are other ways to install the necessary software on the Wi-Fi card, such as the direct use of the Arduino card as an USB modulator and install the driver directly through Arduino Studio.

The production size of the system may be much lower, depending on the resources allocated to design it and the availability of technologies to reduce the dimensions of the sensors and the motherboard. It is obvious that in the case of a system used in production, an Arduino microcontroller will not be used, but much smaller variants will be chosen, or another board will be built with only the desired functionality.

One of the goals pursued in the building of this system was to minimize costs and maximize utility. Another goal is the availability of the system for anyone. Getting a lower price determines the possibility of building a new system

later and integrating it into the network without great expense or effort on the part of the administrator.

Prices for the built system are shown in Table 1:

Table 1

Prices of system components

Component	***Price (EUR)***
Arduino Uno	5
ESP8266 Serial Wi-Fi adapter	1,6
DHT11 Temperature and humidity sensor	1,2
LM393 Sound Sensor	2,7
PIR HC-SR501 Motion Sensor	1,4
Conductive elements and resistors	0,6
TOTAL	12,5

The price becomes much lower if components are ordered directly from the manufacturer or if much cheaper comparable components are used. The price shown in the above table is the price of the devices in specialized stores in Bucharest.

Sensor connection and ESP-8266 mode was made using a breadboard and connecting conductors elements..

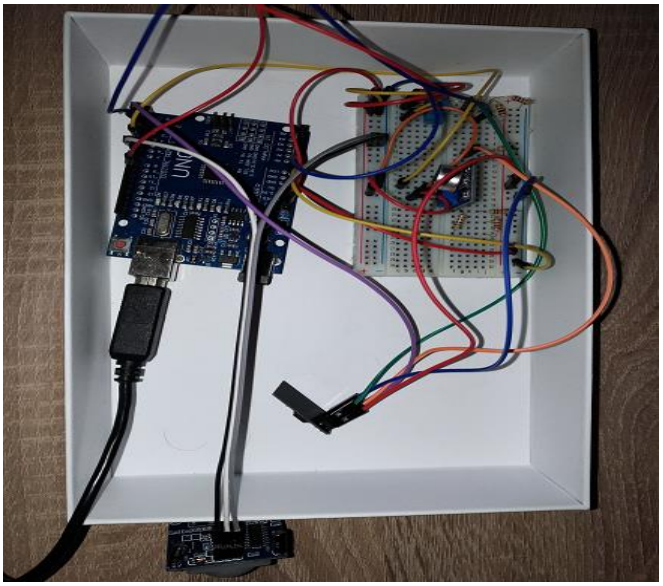


Fig. 1. The proposed hardware structure of the system

To connect the DHT11 temperature sensor, a 10 kΩ resistor was used between the Vcc pin and the output signal. There is no need for resistance when using the 3-output version of the sensor.

The PIR sensor and the sound sensor being digital, the setting of the signal pins in the source code loaded in the microcontroller must be digital.

To connect the ESP8266 module to the microcontroller, a voltage divider scheme for the RX and TX pins of the module was used because it has positive logic at 3.3 V and the Arduino Uno microcontroller at 5 V. If we use a 3.3 V logic microcontroller, it would not be necessary to use the voltage divider.

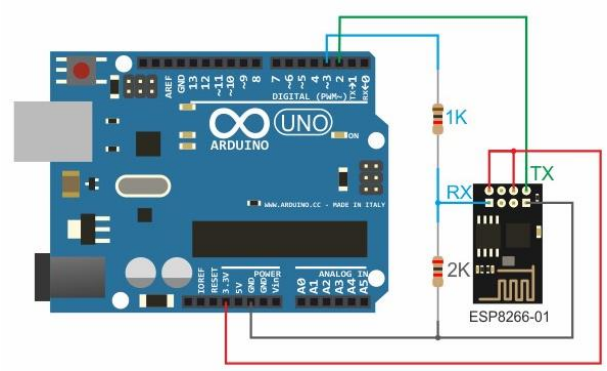


Fig. 2. Connection scheme between ESP 8266 mode and microcontroller

The physical connection of the sensors is shown in Fig. 3 for the temperature sensor, in Fig. 4 for the motion sensor, respectively Fig. 5 for the sound sensor.

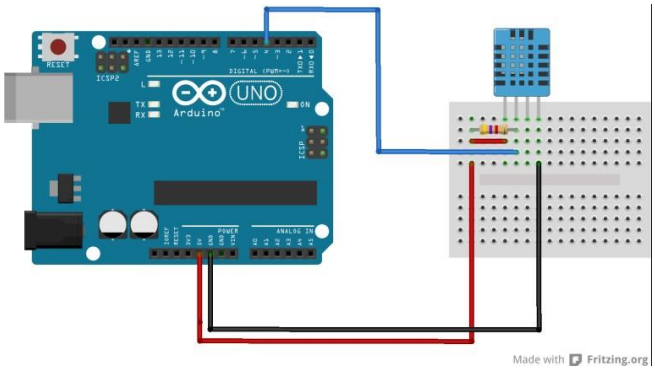


Fig. 3. Connection diagram between DHT11 temperature sensor and microcontroller

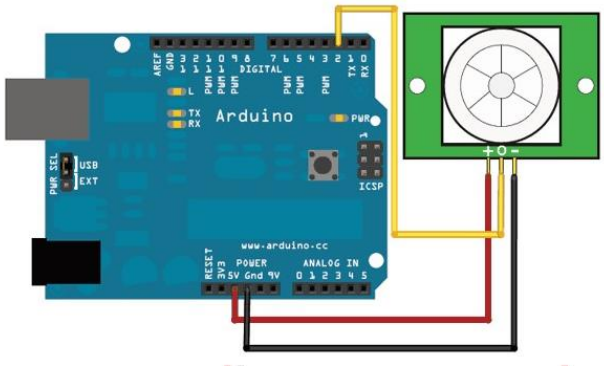


Fig. 4. Connection scheme between PIR motion sensor and microcontroller

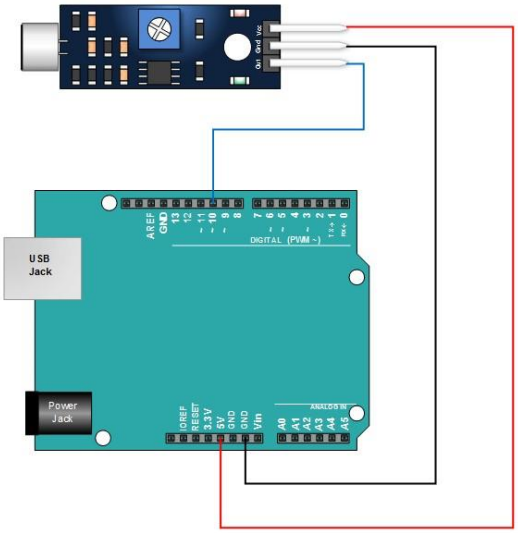


Fig. 5. Connection scheme between the sound sensor and the microcontroller

The code used to run the system is shown at: https://drive.google.com/file/d/0Bx1BJ3tJprfUOXQtTEJUc2NIczg/view?usp=sharing.

The code is written in Wired, a special language for Arduino. It is very simple to translate it into C or C ++ for use on any other controller, the libraries used for sensors are written in C. The only change to be made is to change the pins setting without using the functions in Wired language.

It is also necessary to change the character limit in the Arduino Software Serial Library. It is recommended to set a value of 256, and the old header should be kept for any eventuality.

3. Server Cloud

Creating a cloud server in the context of this paper aims to be as scalable and inexpensive as possible for later maintenance. The cloud server must be able to run for both public and closed networks.

It was chosen to build the cloud server using the Symfony 3.2.8 [10] framework and the PHP7.0 language for the logic behind it. As cloud storage, it was chosen to use MongoDB 3.2 [11] because of its speed and the availability of interaction libraries with the database. The system used was NoSQL which allows a large amount of data to be stored.

The server holds information about the metering systems connected to it, such as the name, the access key, but also certain data such as the minimum temperature and maximum temperature that is desired, and the system enabling the transmission of certain commands or actions back to the microcontroller. In turn, the microcontroller is able to process and execute the required actions (i.e. by using an IR module it is possible to operate the air conditioning).

The server, as well as the cloud database, can be scaled to multiple hardware or virtual servers. At the same time, it is possible to transmit the data to two different servers and to back up the already recorded measurements. In the case of large quantities of data, the decision to archive measurements over 30 days or any other requested interval can be made.

For connection and parallel operation of about 20-25 measurement systems described in this paper, an Amazon cloud server with minimal configurations, which would cost Euro 4 per month is more than adequate. Likewise, any other server, including a Raspberry Pi or a home server, can be used. To demonstrate the functionality of this project, three sensors were connected to a laptop assigned as server, and it worked smoothly.

The Symfony Framework also makes it easy to secure the connection by creating users for each microcontroller using FosUserBundle, but also for every customer who wants to obtain the data. Similarly, Symfony allows creating an API to serve collected data to customers. Examples are systems that control temperature, humidity or alarms, or for example, mobile applications that can display real-time data on the sensor.

For maximum security, the HTTPS connection can be used, supported by both the server and the measurement system. It is important to note that in this case the ESP8266 module will work more slowly due to the need to process the SSL certificate. TCP communication with secure data transmission and reception can be used.

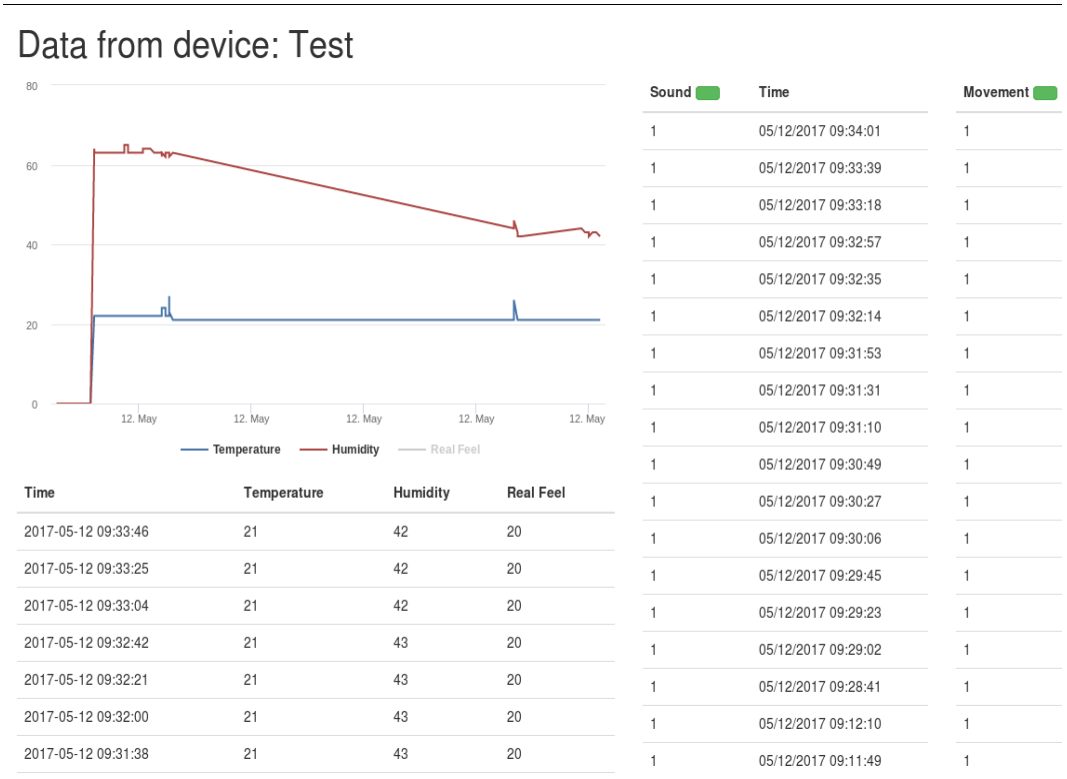
The source code for the server can be found at :
https://github.com/Belibov/arduinoReport

As mentioned earlier, the Symfony framework allows quick introduction of changes, making it an optimal solution for development. This framework can also be used for a production server providing a stable connection and low system resource consumption.

Data transmission to the server

The most important activity undertaken by the microcontroller is the transmission of data to the server.

Collected data is transmitted separately in 3 different requests for each sensor (eg temperature, motion and sound). This allows us to scale the number of sensors and assures us that if the sensor fails or is lost, the others will reach the server. After measuring the data on each sensor, they are immediately transmitted by serial interfacing to the ESP8266 board via an AT command, the data represents an HTTP or HTTPS request with the POST method and the data together with the security key in the content of the request.



Time	Temperature	Humidity	Real Feel
2017-05-12 09:33:46	21	42	20
2017-05-12 09:33:25	21	42	20
2017-05-12 09:33:04	21	42	20
2017-05-12 09:32:42	21	43	20
2017-05-12 09:32:21	21	43	20
2017-05-12 09:32:00	21	43	20
2017-05-12 09:31:38	21	43	20

Sound	Time	Movement
1	05/12/2017 09:34:01	1
1	05/12/2017 09:33:39	1
1	05/12/2017 09:33:18	1
1	05/12/2017 09:32:57	1
1	05/12/2017 09:32:35	1
1	05/12/2017 09:32:14	1
1	05/12/2017 09:31:53	1
1	05/12/2017 09:31:31	1
1	05/12/2017 09:31:10	1
1	05/12/2017 09:30:49	1
1	05/12/2017 09:30:27	1
1	05/12/2017 09:30:06	1
1	05/12/2017 09:29:45	1
1	05/12/2017 09:29:23	1
1	05/12/2017 09:29:02	1
1	05/12/2017 09:28:41	1
1	05/12/2017 09:12:10	1
1	05/12/2017 09:11:49	1

Fig. 6. Data received from sensors

On the server side, the microcontroller with the name specified in the request is checked, and then it is verified that the secret key sent to the request is valid. If these conditions are not met, the request is ignored. It is important that

the microcontroller is registered with its key on the Symfony server side. If a valid microcontroller has been identified, it will identify the type of sensor from which the request came and the data will be inserted into the MongoDB database. Prior to insertion, the data can be processed. A processing example implemented on the server side is setting certain limits for the temperature sensor. Thus, if the temperature is below or above the set limits, an action can be taken, such as connecting the air conditioner or calling an external system and notifying it. In case of movement or presence of the sound, the arming functionality was implemented. Thus, if the system is armed, when motion or sound is detected, the alarm signal can be started. These settings are set at the microcontroller level and can be changed at any time.

Sensor data can be monitored either on the server by going to server_host / data / controller_name, where server_host is the address of the server, and controller_name is the name of the controller set in the server interface (Figure 6). At the same time, this data can be sent to a mobile client or other systems or services inside or outside the network by using a REST API with the FOS bundle: RestAPI from Symfony.

4. Benefits of the system

The given system has several benefits that have been listed and described above. The main objective of the paper was to obtain a cheapest system in relation to the benefit. We believe that this goal was achieved by reducing the prototyping cost to 12,5 Euro, the production cost being reduced to about 10 Euro. Maintenance of the server is 4 Euro per month, and a local server can be run on a Raspberrry Pi card, its price is about 40 Euro.

Another benefit of the system is its scalability. Scalability is present in the following aspects:

- The cloud server can be run both on public and within closed networks. It can be used to monitor a home by serving collected data to the user's devices.
- The server can be scaled horizontally and vertically, as can the cloud database. In some contexts, it is possible to introduce intermediate servers to collect data from multiple devices and serve them to the base server. Database replication systems can be built just as simply, with a backup available, but also a possibility to serve data even when the main system is no longer working.
- As many sensors and as many modules as needed can be connected to the microcontroller (subject to availability). Any microcontroller can be used due to its programming in C language. Thus, the limit of sensors is dictated by the correct choice of the microcontroller.

- Any type of microcontroller, Wi-Fi adapter and sensors can be used.
- Wi-Fi can be replaced with an Ethernet connection with the RJ-45 cable microcontroller. Thus, in the absence of a Wireless signal or to ensure a safer connection, direct data cable connection can be used.
- The system can be used for any purpose: educational, industrial, security, home monitoring. It can be improved to serve any purpose, scaling either the sensors used or the number of supported devices that can be integrated into the system.

Availability is another benefit of the system, as building it, reprogramming from existing sources and code is very simple, and system components can be found in any electronics store. For example, only around the Polytechnic University of Bucharest are at least 5 stores selling the components necessary for the system. In the case of use for production purposes, it is possible to choose to purchase the components directly from the manufacturer or to manufacture the necessary components, thus ensuring the lowest price of the components, but also the size of a device as small as possible.

5. Conclusions

This paper demonstrates the optimal use of a cloud server for monitoring the read data obtained from the sensors. The scientific importance of the work consists in building a cloud server able to monitor the real-time received information from sensors and rewriting it into the ROM memory of the Wi-Fi module (ESP8266-01). The Wi-Fi module makes possible communication at 9600bps speed and without any loss of information. The cost and effort required for it is not very high, being available to anyone.

From a teaching point of view, the hardware structure of the system will be improved by creating printed circuits that will allow the use of as many sensors as possible. Interconnecting them with the microcontroller will allow real-time monitoring of all analog and digital sensors connected. From the point of view of industrial scale usage, depending on the requirements of the customer, this system can be developed on dedicated printed circuits. These printed circuits will be made using specific programs such as EAGLE, DipTrace, OrCAD.

The main benefits of such a system are scalability, low costs and availability. Thus, it can become a learning resource, but also a powerful tool that can be used in any field of activity.

The system can be further developed by adding compatibility with boards such as Raspberry Pi, with a much larger application development power.

Similarly, developing compatibility with various modules can lead to the application of the system within automotive, home automation, warehouse monitoring systems.

Theoretically, the cloud system has no limit to further development.

It is also possible to make a mobile application for system monitoring, data being served by the cloud server. Such a change on the server side is very fast and does not require advanced knowledge.

In the context of the development of the IoT segment in the world and the increasing interest in the automation of housing and in general of any buildings, such a system for real-time monitoring of sensor data is practically necessary. The difference of the system outlined in this paper over the existing alternatives is its scalability and its easy adaptability irrespective of the final purpose for which it is built.

R E F E R E N C E S

[1]. *G. Suciu, G. Todoran*, Cloud M2M Platform for renewable energy tele-monitoring, , Scientific Bulletin, University POLITEHNICA Bucharest, Series C: Electrical Engineering, Vol. 76, Iss. 1, 2014

[2]. *G. Mateescu, V. Sgârciu*, Cloud computing audit, Scientific Bulletin, University POLITEHNICA Bucharest, Series C: Electrical Engineering, Vol. 77, Iss. 3, 2015

[3]. *R. Marcu, D. Popescu, I. Dănăilă*, Healthcare integration based on cloud computing, Scientific Bulletin, University POLITEHNICA Bucharest, Series C: Electrical Engineering, Vol. 77, Iss. 2, 2015

[4]. *M. A. Moisescu, I.Ş Sacală, A. M. Stănescu*, Towards the development of Internet of Things oriented intelligent systems, Scientific Bulletin, University POLITEHNICA Bucharest, Series C: Electrical Engineering, Vol. 72, Iss. 4, 2010

[5]. *G. Manea, S. Popa*, Integration of sensor networks in cloud, Scientific Bulletin, University POLITEHNICA Bucharest, Series C: Electrical Engineering, Vol. 78, Iss. 2, 2016

[6]. Arduino Uno documentation, disponibilă online: https://www.arduino.cc/en/main/arduinoBoardUno

[7] *Y. Wang, Z. Chi*, System of Wireless Temperature and Humidity Monitoring Based on Arduino Uno Platform, Sixth International Conference on Instrumentation & Measurement, Computer, Communication and Control (IMCCC), 21-23 July 2016, Harbin, China

[8]. *L. Chen, J. Zhang, Y. Wang*, Wireless Car Control System Based on ARDUINO UNO R3, 2nd IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), 25-27 May 2018, Xi'an, China

[9]. ESP8266 documentation, disponibilă online: https://espressif.com/en/support/download/documents

[10]. Symfony 3.2 Documentation, disponibilă online: http://symfony.com/doc/current/index.html.

[11] MongoDB documentation, disponibilă online: https://docs.mongodb.com

[12] *C.G. Sărăcin, M. Bizineche*, Educational platform for monitoring and reconfiguring of electric power distribution network, Scientific Bulletin, University POLITEHNICA Bucharest, Series C: Electrical Engineering, Vol. 73, Iss. 2, 2011