Real time implementation of IoT structure for pumping stations in a water distribution system

Alexandru Predescu
Automatic Control and Computer Science
University Politehnica of Bucharest
Email: mircea.predescu@stud.acs.upb.ro

Mariana Mocanu
Automatic Control and Computer Science
University Politehnica of Bucharest
Email: mariana.mocanu@cs.pub.ro

Ciprian Lupu
Automatic Control and Computer Science
University Politehnica of Bucharest
Email: ciprian.lupu@acse.pub.ro

***Abstract*—This article presents a real-time control system for an experimental model of a water distribution system. The hardware and software implementation follows the latest trends in IoT and proves that such an architecture can be used for real-time process control in distributed systems. The control of flow is implemented using a multiple-model control algorithm for the pumping station. A method which shows the sensitivity of the sensors is used to validate the leak detection and parameter control capabilities resulted from the structure of the water network.**

***Index Terms*—Real Time Applications, Internet of Things, Adaptive Control**

I. Introduction

Accurate flow control in a water distribution system according to the demand given by the consumers has become feasible by implementing a distributed control system. This makes use of new technologies for real-time monitoring of measured parameters and control of actuator systems.

In this article, the problem of controlling water flow and pressure in the network by adjusting the pump output is described from a control systems point of view. The experimental model also demonstrates the integration of wireless sensor networks and mapping applications for real-time control and monitoring of water distribution systems. The multiple-model control strategy is a particular case of adaptive control which can be used for systems with large variations in parameters. [1]

II. Related work

The newly emerging technologies allow for a higher level of integration as part of a smart infrastructure. The growing number of Internet connected devices can be seen as a second revolution of the Internet which will allow for connected objects to play an active role in a smart environment. This is accomplished with open-source solutions which share a common language based on interoperable protocols. [2]

In the case of water networks, GIS (Geographic Information System) and SCADA (Supervisory Control And Data Acquisition) systems are becoming the norm. These are used in the context of IoT (Internet of Things) solutions for remotely managing distributed systems as in the case of recent projects such as the smart irrigation system presented in [3]. Research work has been done on leak detection such as measuring the effects of leaks using a fault sensitivity matrix in [4], control and optimization of energy usage in [5], automation of pumping stations for residential water supply in [6], where the pump output is controlled for increased dynamic using a frequency converter. This method is proved to be superior to on/off electric drives which lead to frequent malfunctions. An accurate control of pressure and flow in a water network also improves the lifetime of the piping systems and contributes to energy savings.

In the area of control systems for complex models with variable parameters there are multiple approaches. The adaptive controller solves some of the problems of parameter uncertainty and the particular case of multiple-model controller is well suited to known parameter variations in the model. The switching solutions that can be used for the multiple-model control algorithm are described in [7].

III. Proposed solution

The water network presented in this article can be used for experimental testing of control algorithms for water distribution systems. First, we describe the construction and the hardware and then we present the software solution for monitoring and controlling the parameters of the network (flow, pressure, pump output). We present the design and validation of control algorithms on the experimental model and we also implement a method for measuring the leak sensitivity of the sensor network as described in [8]. We considered an IoT approach for monitoring and controlling a water distribution system, having a client-server architecture where the sensor and actuator nodes are connected using wireless technologies and the UI (User Interface) is accessible through web services. The performance of this solution can be observed through the experiments presented in this article and we comment on the feasibility of such architecture as a real-time control system.

A. Experimental model

The experimental model is implemented using a small scale water network which is described in this section.

The diagram of the network in figure 1 shows the numbering of the sensors and valves.

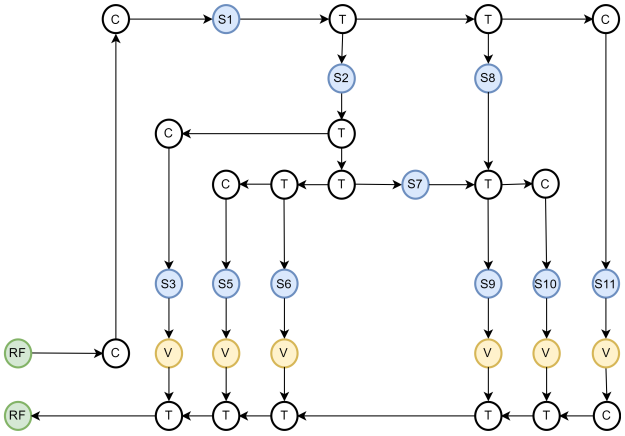

Fig. 1. Schematic of the experimental model

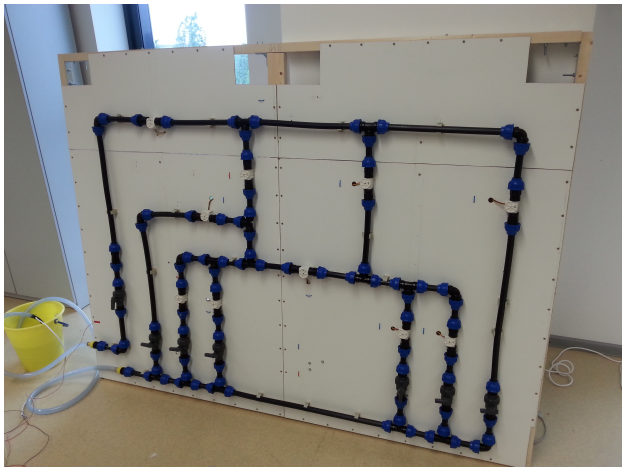

Fig. 2. Experimental model

The water network consists of PVC (polyvinyl chloride) pipes with detachable connecting elements such as couplings, tees, elbows, as shown in figure 2. The water flow is controlled using manual valves to simulate the demands and leak scenarios. Water flow sensors are installed along each section and at every demand or leak from a node. The water from demand and leak nodes is directed back into the supply tank via tee connectors and it is then introduced back into the network by using a DC electrical pump.

B. Reading data from the sensors

The flow sensors contain a rotor and a hall sensor which converts the water flow into electrical pulses. By reading these pulses and accurate timing, water flow can be calculated. We used an Atmega 328p as the application MCU (microcontroller unit) and pin change interrupts for reading the sensor signal. In the case of larger networks, our solution is scalable and there can be one MCU for each sensor, communicating with the central server or gateway using ESP8266 Wi-Fi adapters. In this experimental solution, we used a single board for cost reasons. The Atmega 328p can be configured to use multiple pins for pin change interrupts and is sufficient for the task.

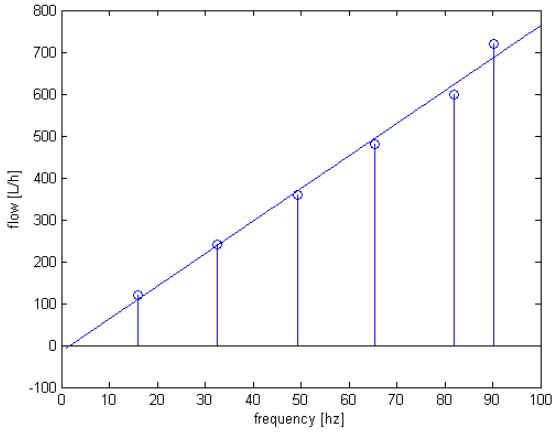

Fig. 3. Flow sensor characteristic

The flow sensor (YF-S201) has a range of $1L/min$ to $30L/min$ with the output characteristic shown in figure 3. We considered the linear approximation of $Q = 7.77f - 14.8605$, where f is the frequency of the square wave generated by the sensor and Q is the calculated flow in L/h.

The time between consecutive pulses is measured using a single MCU timer. Timer 1 is configured with a prescaler of 64 which gives a 4 μs resolution and the pulse frequency is calculated using the following formula:

$$f = \frac{f_{osc}}{N \cdot prescaler} \tag{1}$$

where f_{osc} is the clock frequency of $16MHz$ and N is the number of ticks returned by the timer. We used a single hardware timer for all sensors by using a variable for implementing each software timer and a variable which is incremented when the timer overflows. The code for a single pin change interrupt is shown next:

```
dt[0] = TCNT1 + t_ovf[0]*TIMER_OCR − t0[0];
t0[0] = TCNT1;
t_ovf[0] = 0;
```

In this way, the timer runs continuously without being reset.

We implemented a digital low pass filter to reduce the noise from the measurements, with the sampling time $T_s = 0.1s$ and cutoff frequency $f_c = 10Hz$. This results in a smoothing constant:

$$\alpha = \frac{T_s}{T_s + RC} = 0.86 \tag{2}$$

where

$$RC = \frac{1}{2\pi f_c} \tag{3}$$

In code we implemented the filter using integer arithmetic which results in a faster computation time and therefore higher sampling rates:

```
dtf[i] = (dtf[i]*14 + dt[i]*2)>>4;
```

The smoothing constant is 0.875 in this case, which is close to the calculated value.

C. Monitoring the network

For monitoring the parameters in the water network, we implemented a web mapping application which can handle real-time data from sensors and display the data on the map. The sensors are connected to a Wi-Fi gateway and the connectivity to the application server is implemented using raw sockets (TCP). The sensors transmit data to the server along with the identifier and sensor type. The Wi-Fi adapter used for the sensors is programmed to connect to the gateway and has the functionality of a TCP to Serial (UART) bridge. The downside would be the requirement of a gateway within the range of the sensors. Using a modular design, the solution can be adapted for remote sensors in geographically distant locations by using long range wireless technologies.

The water network is digitized using a GIS application which is according to the INSPIRE directive specifications. [9]

We used ArcGIS to define the layers of the map: nodes, pipes, sensors. [10]

The web application front end (figure 4) is written in AngularJS and uses the ArcGIS API (Application Programming Interface). This allows for a well structured and maintainable application as well as an interactive map-centric application. With AngularJS, the navigation, views, and front end logic are implemented and the map API is integrated into the application. This offers more flexibility in terms of functionality, as compared to the usual approach using Dojo as the main framework. [11]

The libraries are loaded in the following order: the ArcGIS API, the generated JavaScript file for the application and the Angular directive for the ESRI map. The stylesheet for the ESRI theme is also loaded before the application stylesheet. In the application modules we load *esri.map*. Then, in the controller for the map view, we use *esriLoader* to load the ESRI modules: map view, feature layers, widgets. We define the feature layers according to the generated map, in this application we include the nodes, pipes and flow sensors layers having the URL of the ArcGIS server. We use the API to detect the user selection of map elements and request real-time data and history for the selected sensor.

Other functionalities such as navigation, charts and table views are implemented using JavaScript and AngularJS. [12]

The real-time values are retrieved from the server via WebSockets and the time series data via HTTP (Hypertext Transfer Protocol) requests. By selecting an element on the map, real-time sensor data and history is shown.

The back end is written in Python and implements the server for the wireless sensor network, database connection, the HTTP and WebSocket server. The application runs on a server along with the ArcGIS and SQL (Structured Query Language) services.

The Pyhton back end implements the routes for HTTP requests and WebSocket messages. The *socketio* library is

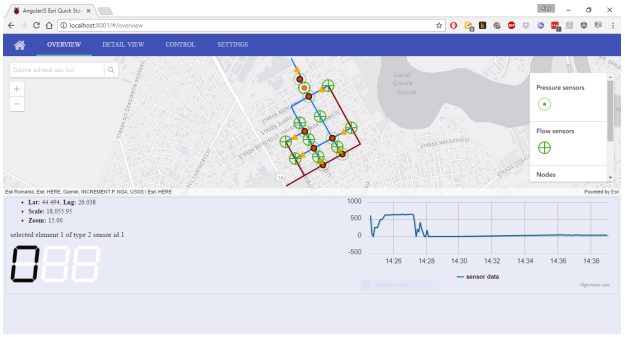

Fig. 4. Web application front end

used. The application uses threads for the main tasks such as collecting data from sensors and control algorithms. It also implements the connection to the database created with MS SQL Server 2012, using the *pyodbc* API.

The sensors are connected to the server by a TCP (Transmission Control Protocol) messaging system. The sensor id, type and data is transmitted in CSV (Comma-Separated Values) format and is parsed by the application to be shown in the GUI (graphical user interface) and stored into the database.

D. Flow and pressure control

The main control problem consists of regulating the flow through a specific pipe, or through multiple pipes. This is achieved by implementing a control algorithm for the pump, using the measured flows as inputs. The controller is implemented on the application server and uses measured data from the wireless sensors. The output is sent to a wireless actuator device which controls the pump speed.

This method allows for a predictive control of flow based on demand and leak estimations, in which the pump output is adjusted to supply the required total flow in the particular section of the water network.

The first objective is to find the transfer function from the pump input to the sensor output, and an algorithm which satisfies the control objectives for every possible scenario.

We considered multiple scenarios which result in a multiple-model approach. The valve corresponding to the demand node for which the flow control is required (sensor 10) is opened for the entire experiment and the valves corresponding to the other nodes (sensor 3, 5, 6, 9, 11) are opened and then closed in sequence for each case.

The supervisor implements the selection of best controller according to the measured parameters as shown in figure 5. The usual method selects the model having an accurate estimation of the process output, or implements a weighted average of the model outputs. In the case of complex water networks, the models are not mutually exclusive and multiple models can have the same output in different scenarios. This problem is solved by using a weighted average of controller outputs at the supervisor level. First, the error of the predicted output is calculated for each model in (4). The errors are

normalized in (5) using the minimum absolute value and the particular case when the error is zero is treated in (6). The multiple-model controller output in (9) is a sum of controller outputs scaled by the calculated coefficient in (8). If $s[k] = 0$ then a random controller is selected.

$$\epsilon_i[k] = y[k] - y_i[k] \tag{4}$$

$$\alpha_i[k] = \frac{\min_{j=1}^{N} |\epsilon_j[k]|}{|\epsilon_i[k]|}, |\epsilon_i[k]| \neq 0 \tag{5}$$

$$\alpha_i[k] = \begin{cases} 1, |\epsilon_j[k]| = 0, j = i \\ 0, |\epsilon_j[k]| = 0, j \neq i \end{cases} \tag{6}$$

$$s[k] = \sum_{j=1}^{N} \alpha_j[k] \tag{7}$$

$$\beta_i[k] = \frac{\alpha_i[k]}{s[k]}, s[k] \neq 0 \tag{8}$$

$$u[k] = \sum_{i=1}^{N} (\beta_i[k] \cdot u_i[k]) \tag{9}$$

When the state of the network can be accurately measured or estimated, we suggest that the selection of the best models can be accomplished by monitoring the flow sensors in the network. In this case, the selection criteria used in the supervisor block are the measured flows.

E. Leak sensitivity

Leak sensitivity is defined using the sensitivity matrix for flow discrepancies (10). This measure is used to visualize the effects of leaks on measured flows in the network. The method is presented in [8] using a simulated model.

$$S_q = \begin{bmatrix} \frac{\partial q_1}{\partial q_{L1}} & \frac{\partial q_1}{\partial q_{L2}} & \dots & \frac{\partial q_1}{\partial q_{Ln}} \\ \frac{\partial q_2}{\partial q_{L1}} & \frac{\partial q_2}{\partial q_{L2}} & \dots & \frac{\partial q_2}{\partial q_{Ln}} \\ \vdots & & & \\ \frac{\partial q_n}{\partial q_{L1}} & \frac{\partial q_n}{\partial q_{L2}} & \dots & \frac{\partial q_n}{\partial q_{Ln}} \end{bmatrix} \tag{10}$$

The partial derivatives represent the difference between the nominal conditions and the leak scenario. The sensitivity matrix is normalized for each row (12) using the maximum value obtained by simulating each leak scenario (11).

$$\sigma_i = max\{s_{i1}, \cdots, s_{in}\} \tag{11}$$

$$\overline{S} = \begin{bmatrix} \frac{s_{11}}{\sigma_1} & \frac{s_{12}}{\sigma_1} & \dots & \frac{s_{1n}}{\sigma_1} \\ \frac{s_{21}}{\sigma_2} & \frac{s_{22}}{\sigma_2} & \dots & \frac{s_{2n}}{\sigma_2} \\ \vdots & & & \\ \frac{s_{n1}}{\sigma_n} & \frac{s_{n2}}{\sigma_n} & \dots & \frac{s_{nn}}{\sigma_n} \end{bmatrix} \tag{12}$$

We considered every possible scenario with two demand nodes for which the valves are always open and the other valves which are opened in sequence. The data measured during this experiment is used to compare the nominal values to the values obtained in the other cases. The flow discrepancies are represented in the sensitivity matrix in section IV. This experiment shows the most sensitive sensors for each case, which can be used to estimate the current test scenario with the highest possible accuracy. This method can also be used for optimizing the location of sensors in order to detect the most leak scenarios.

IV. Results

The experimental model and application described in this article allow for real-time monitoring of the effects which leaks have on the measured flow in the water network.

For obtaining the flow discrepancies, we opened and closed each valve from the leak nodes (we considered nodes 3, 6, 9, 11). The valves from demand nodes (nodes 5, 10) were kept at a fixed position. The measured values were recorded for each case and the sensitivity matrix is used to show the effects of each leak on measured flows in the network. In figure 6, the normalized matrix is represented with the highest sensitivity shown in red and the lowest sensitivity shown in blue. For generating the matrix, the element at position (i,j) represents the flow discrepancy measured in the node i in the case of a leak in node j. The normalization is done for each row (we use the maximum value for the row and then divide each element by this value). For plotting the diagram we used the Matlab function *stem3*. The x-axis is the transposed array of measured nodes, repeated to form a matrix with the number of rows equal to the number of leak nodes.

x3d = repmat(m_nodes,1,length(leak_nodes))'

The y-axis is the array of leak nodes, repeated to form a matrix with the number of columns equal to the number of measured nodes.

y3d = repmat(leak_nodes,1,length(m_nodes))

The z-axis represents the value of the flow discrepancy for each case.

For process identification, we considered multiple scenarios and a multiple-model control algorithm. For each active demand node, we used experimental methods for calculating a first order model. We considered the scenarios with a single demand at node 10 and the ones with an additional demand at nodes 5, 11.

TABLE I
IDENTIFIED MODELS

active demands	K_p	T_p
10	0.98	0.5
10, 5	0.74	0.425
10, 11	0.44	0.4

The control algorithm for regulating the flow through pipes across the water network is designed according to the identified models. Each controller is calculated using the closed loop transfer function which is defined by taking into consideration the desired performance of the system. We

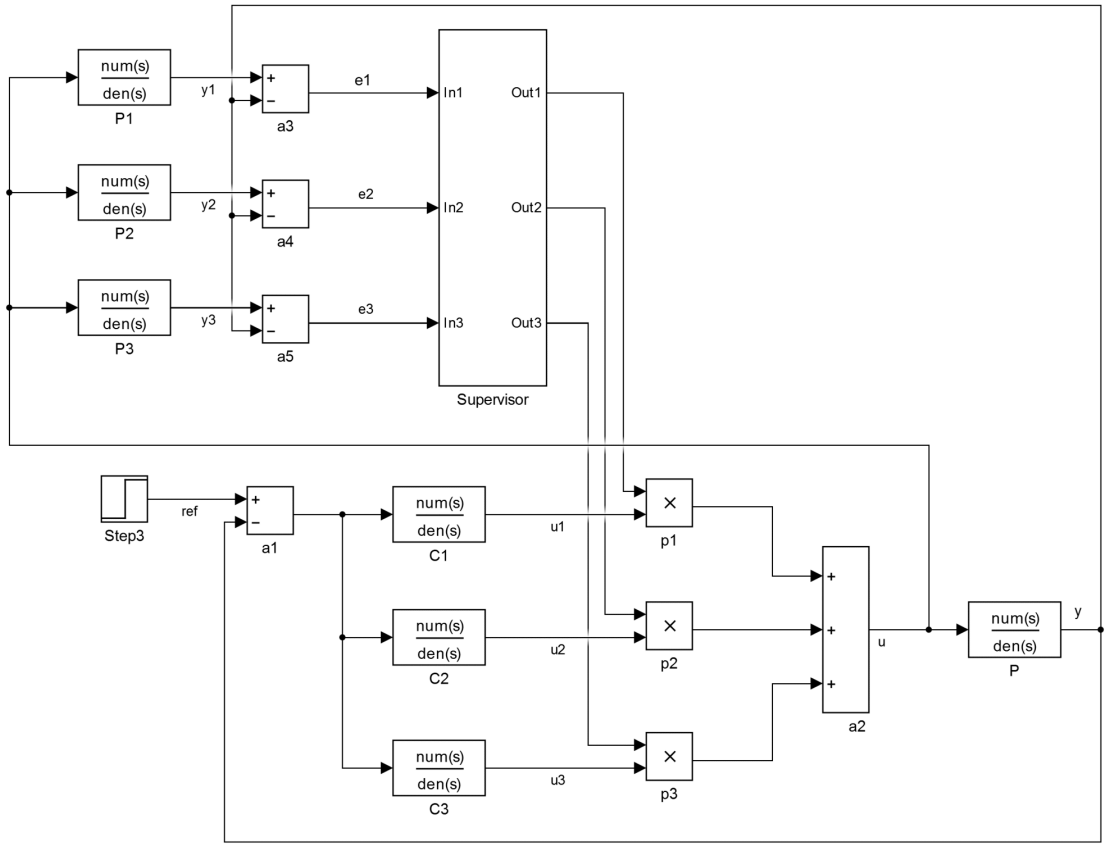

Fig. 5. Multiple-model control diagram

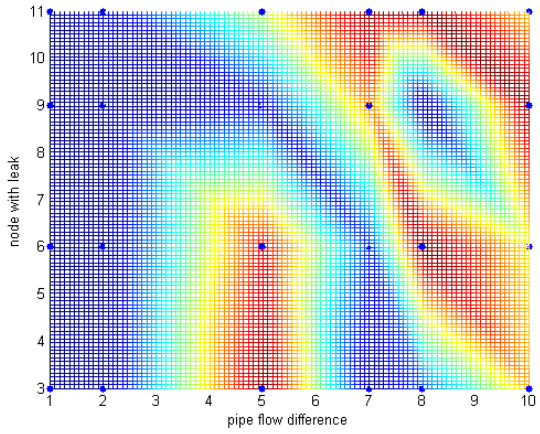

Fig. 6. Leak sensitivity

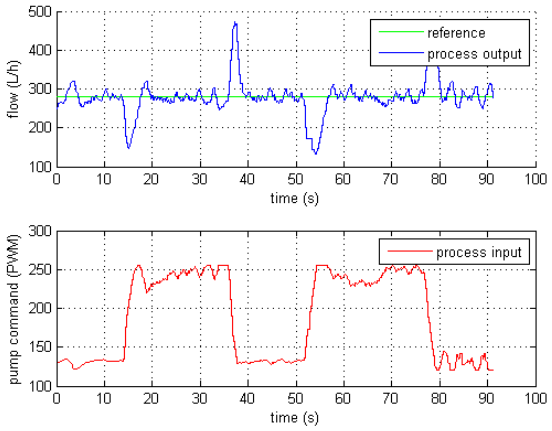

Fig. 7. Disturbance rejection

chose a first order transfer function and this resulted in a PI controller.

In figure 7, the experimental results show the disturbance rejection using the multiple-model controller. The disturbance is in this case the opening and closing of valves from sensor 5 and 11. In this case, the supervisor correctly identifies the model, though in more complex scenarios we suggest that the measured state of the system should be used.

In the first plot, the reference value and the process output are shown. In the second plot, the pump command (8-bit PWM) is shown and represents the process input generated by the control algorithm.

The uncertainties given by the structure of the experimental model and the vertical layout make the results less accurate than expected. The results obtained using the experimental model can be improved by increasing the water pressure, which would reduce the system uncertainties across the input domain and would result in more accurate models.

V. Conclusion

This work presents a method to implement a real-time monitoring and control system using hardware and software solutions which are being used on a large scale in IoT. By integrating the flexibility of web frameworks, open-source hardware, and ArcGIS, we propose a different approach to process control with an experimental model, which can be extended in the case of remote sensors and actuators.

The multiple-model approach can be used in a control strategy optimizing the control signal for the pump and therefore the energy efficiency and lifetime of the equipment is improved. By having flow sensors installed on each pipe, the state of the system can be measured directly, which is important for accurate estimation of the most suitable model in the control algorithm. The requirement for smooth transition between the multiple models in the control system is satisfied using a weighted average of the controller output.

Acknowledgment

The paper presentation has been supported by the Data 4 Water - H2020 Twinning project no. 690900.

References

[1] K. S. Narendra, O. A. Driollet, M. Feiler, and K. George, "Adaptive control using multiple models, switching and tuning," *International Journal of Adaptive Control and Signal Processing*, vol. 17, pp. 87–102, 2003.

[2] O. Vermesan and P. Friess, *Internet of Things - Converging Technologies for Smart Environments and Integrated Ecosystems*. River Publishers.

[3] Jasper. SMART Watering Systems Automates Water Management, Conserving Water and Saving Customers Money. [Online]. Available: http://www.jasper.com

[4] R. Pérez, V. Puig *et al.*, "Methodology for leakage isolation using pressure sensitivity analysis in water distribution networks," *Control Engineering Practice*, vol. 19, no. 10, pp. 1157–1167, 2011.

[5] J. Studzinski and M. Kurowski, "Water network pumps control reducing the energy cost," *Proceedings of the 28th EnviroInfo 2014 Conference, Oldenburg, Germany*, 2014.

[6] K. A.M., "Automation of the residential building water supply system pumping station," *Automation of technological and business-processes*, vol. 8, no. 2, pp. 27–30, 2016.

[7] C. Lupu and C. Petrescu, "Multiple-models control systems - switching solution," *UPB Scientific Bulletin Series C*, vol. 70, no. 1, pp. 105–117, 2008.

[8] A. Predescu, M. Mocanu, and C. Lupu, "Modeling the effects of leaks on water distribution systems," *accepted for CSCS21 Bucharest, Romania*.

[9] European Commission. INSPIRE. [Online]. Available: http://inspire.ec.europa.eu/

[10] ESRI. ArcGIS. [Online]. Available: https://www.arcgis.com/features/index.html

[11] ——. ArcGIS for Developers. [Online]. Available: https://developers.arcgis.com/javascript/

[12] Google. AngularJS: Developer Guide. [Online]. Available: https://docs.angularjs.org/guide