Politecnico di Milano

Pervasive Systems

Project

Aqua$_{2.}$O

Authors:
Emanuele Falzone
Davide Molinelli

Supervisor:
Fabio Salice
Andrea Masciadri

December 18, 2017

1 Introduction

Aqua$_2$O$_{\text{💧}}$ is an Academic Project commissioned in the context of the Pervasive Systems course. The project arise from the need to create a real-time system able to monitor the domestic water consumptions, through inexpensive and wireless water-flow sensors and a micro-controller. Aims of the project are to analyze the sensor in order to extract the best possible information from its low cost and imprecise measurements and to send the data on a server with the intent to generate a database of the consumptions for analytical and sensitization purposes.

2 Problem Description

2.1 The water-flow sensor

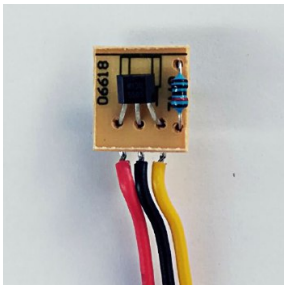
According to the requirement to build a system with the essential properties to be low cost and scalable so that it was possible to easily replicate itself in an economical way, we have been equipped with a YF-S201C sensor-flow model. This sensor consists of a magnetized helix inside it that rotating, crossed by a stream of water, generates a magnetic field. The intensity of the latter is registered by a Hall sensor placed near the helix and a corresponding signal is periodically sent to a micro-controller that converts the received impulses in a flow rate (L/min). The supplied sensor presents crucial problems:

- the measurements are not accurate
- the sensor does not register flow rate less than 1L/min and greater than 30 L/min
- its data-sheet is very poor in the explanation of its properties so that we did not know how the sensor behaved in different real case like, for example, at distinct values of temperatures and pressure

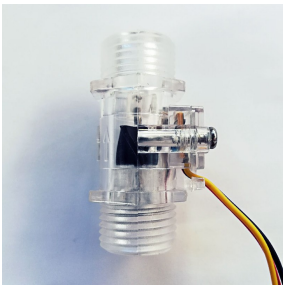
2.2 Functional Requirements

- The system has to acquire the consumptions data in real time, elaborating through the micro-controller, the impulses coming from the Hall sensor installed within the water-flow sensor

(a) The magnetic helix



(b) The hall sensor



(c) The whole sensor

Figure 1: The water-flow sensor

- The micro-controller has to transmit the elaborated data to a web server, through a Wi-Fi communication system

2.3 Non-functional Requirements

- The customer must have the possibility to easily replicate the system in order to install it in an economical and scalable way
- For the same reason, the system has to be inexpensive
- The system has to be reliable and record with good approximation the real consumptions
- The system has to be auto-configurable in order to be installed in an easy way from everyone and reduce the maintenance costs.

3 Data Acquisition

3.1 Hall Sensor Analysis

One of the most significant priority of the project is the analysis of the behavior of the supplied sensor. Initially we hypothesized that the sensor could not be very sensible to the variation of the magnetic fields, so that the corresponding measurements could be characterized by a non-negligible error. Motivated by this suspicion, we decided to install two additional hall sensors in order to compare the measurements.

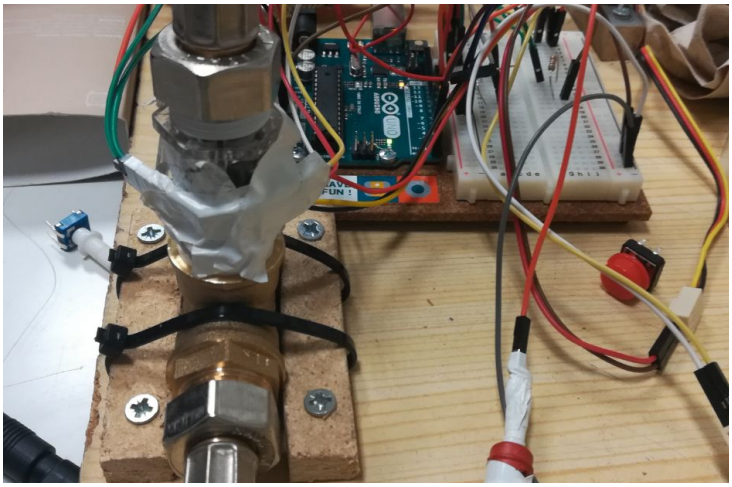


Figure 2: The water-flow sensor with two additional hall sensors

We implemented a simple sketch that use the interrupts to acquire the impulses coming from the sensor and increment a corresponding variable. The following is the principle according to which the impulses are acquired by the Arduino micro-controller and elaborated in order to extract the flow rate information.

In Figure 3a, the magnetized propeller blades are located far away from the sensor, that perceive a vacuum in the magnetic field. Therefore, the value registered by the sensor is under an established threshold and the impulse is not sent to the micro-controller. On the contrary, when a blade is in front of the sensor, the latter detect an intense magnetic field, as shown in Figure 3b. The corresponding value registered by the sensor exceed the threshold and an impulse is sent to the micro-controller. Calculating how many impulses are registered by the sensor in a predefined time interval, it is possible to estimate the speed of the helix and the water-flow rate inside the tube. In our specific case, elaborating and analyzing the impulses coming from the sensors, we inferred that all the sensors behave in the same way (Figure 4) and we could not obtain a significant gain that would justify the use of the additional ones.

(a) Low magnetic field (b) High magnetic field

Figure 3: Hall sensors behavior

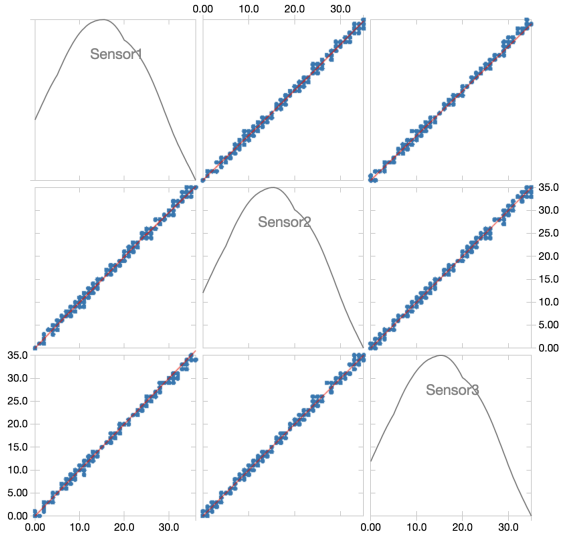


Figure 4: Scatter Plot comparing the three hall sensors

3.2 Water-flow sensor analysis

3.2.1 The supplied water-flow sensor

The data-sheet of the supplied water-flow sensor contains a graph representing how the water-flow rate and the pulses detected from the hall sensor are related. During our experiment we collected a set of 85 points and we used them in a least square algorithm to obtain a regression line that represents how the flow and the pulses are correlated. The results of our analysis are represented in the graph reported in the Figure 5.

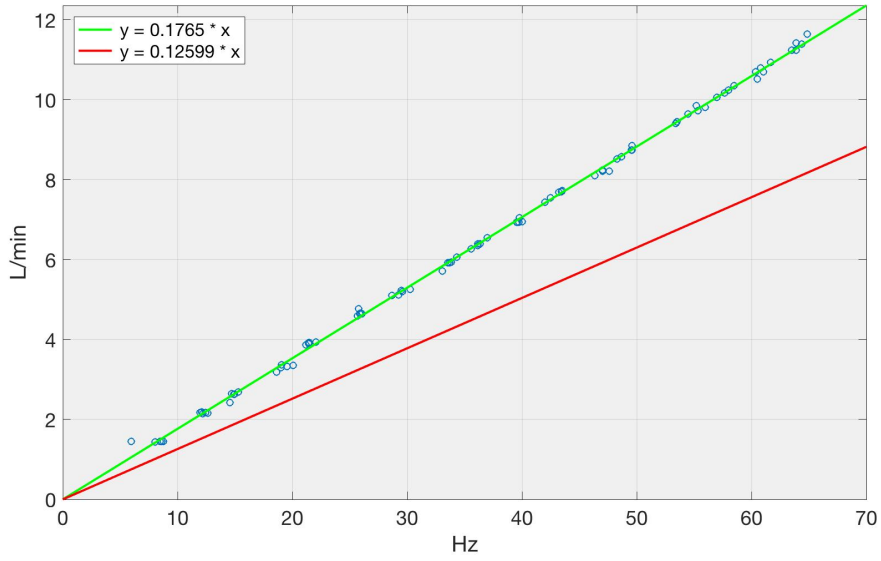


Figure 5: Analysis on the supplied water-flow sensor

3.2.2 The second water-flow sensor

Driven by the curiosity to test the characteristics of the supplied sensor and perform a comparison, we bought a new different water flow sensor that works with the same principle of the first one, but whose propellers rotate in a different way (Figure 6).

(a) The different magnetic helix

(b) The new hall sensor

(c) The whole new sensor

Figure 6: The water-flow sensor

This sensor is more accurate and can measure a lower flow rate than the first one thanks to the different position of the helix. We suppose that in this case the static friction is lower than in the first case. So its propellers start turning before than the propellers of the first sensor.

Also in this case, we collected a set of 85 points and we plot the data to see if the obtained results is similar to the data written in the corresponding data-sheet. In this case the real line has a slope that is more o less the same of the one reported in the data-sheet. The results are shown in Figure 7.

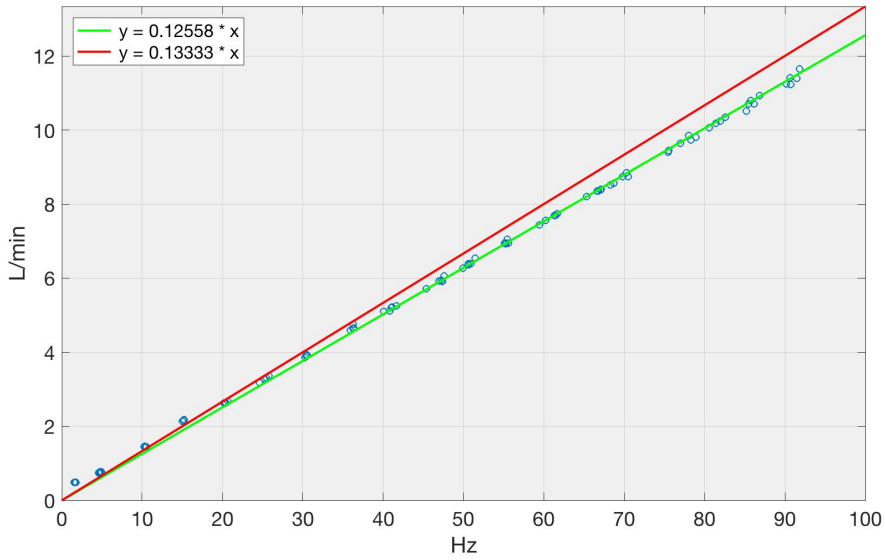
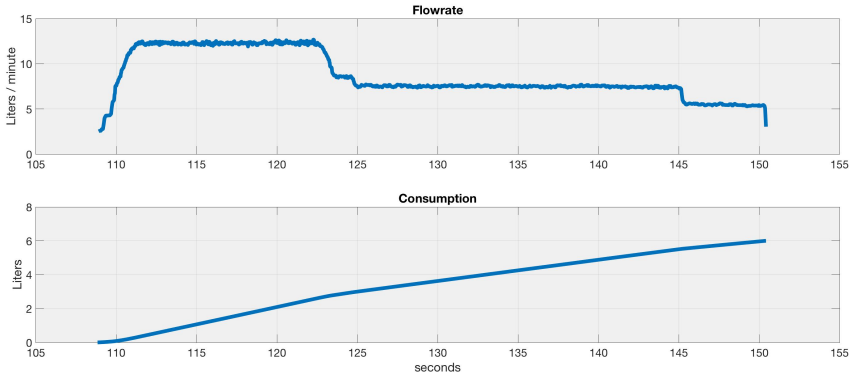


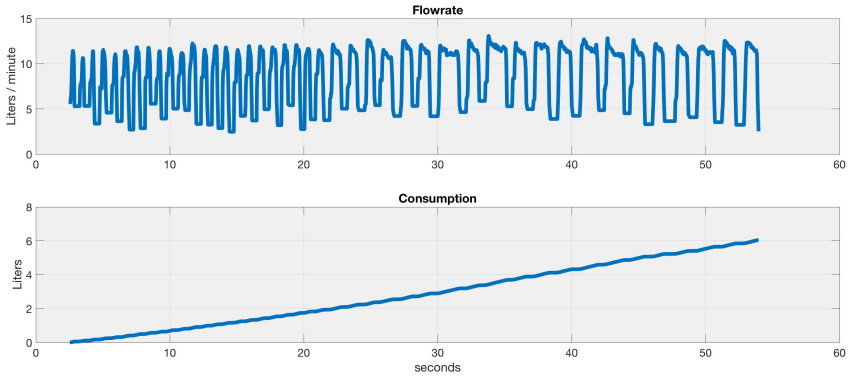
Figure 7: Analysis on the second water-flow sensor

3.2.3 The test phase

Consequently to the analysis of the sensors, performed through the Matlab platform, we executed varied tests in order to verify that the calibration factors extracted from the previous step (corresponding to the angular coefficients of the plotted lines) were accurate. We try to stress the sensors through different typology of simulations. We simulated transitory and steady behaviors (like in Figure 8a) and dynamic behaviors (like in Figure 8b).



(a) Transitory and steady behavior example test



(b) Dynamic behavior example test

Figure 8: Example of tests

The purpose of the tests was to verify that the total quantity of water measured by the sensors in a interval of time was as much as possible equal to the real quantity passed through the sensors themselves. According to the fact that 1mL of water corresponds to a 1g of weight, to realize this simulations we equipped ourselves with a timer and a precision balance, in order to weigh the flowed water collected in a bin and measure the quantity present on it, comparing the effective quantity with the quantity registered by the algorithm implemented on the micro-controller (see Figure 9a and 9b).

(a) A simulation (b) A weigh

Figure 9: Test phases

The simulations returned good results (with an average error of less than 1%). So we decided to attach the sensor to a different pipe and repeat the tests. Not surprisingly, we notices that in different conditions (of pressure, for example) the simulations returned less accurate results (an average error of 2%). We concluded that before installing the sensor in an apartment, it is recommended to analyze the sensor in the specific conditions of pressure and diameter of the pipe in which the sensor will be connected in order to adjust the calibration factor of its behavior and obtain the best measurements.

4 Data Transfer

After having acquired the impulses from the sensor and elaborated them through the algorithm implemented on the Arduino micro-controller, we had the necessity to transfer the data on a platform able to store the information and perform analysis on them.

Figure 10: The Wi-Fi shield

4.1 First approach: saving the data on a SD card

Initially, we decided to memorize the data in a csv format, into a SD card. This solution allowed to have a ready-to-use csv file. The capability of the SD card was not a problem: each data acquisition occupies a trifling memory space so that to exhaust the entire capability of a card with 1 GByte of memory space was necessary to store information for more than 900 days, with a sampling rate of 1sec. On the contrary, this choice was not portable and projecting the academic assignment towards a real case of possible realization (where the system could be installed in each apartment of a residence), we decided to move through a different solution.

4.2 Second approach: trying to send the data on the ThingSpeak web platform

According to the necessity to have the data (coming from an undefined number of different replied systems) ready-to-use in any moment, for analytical purpose, we evaluate the possibility to use a web platform that would allow us to store the data, plot graphs and perform analysis at the same time, in the same place. Our choice was soon directed towards ThingSpeak, an open IoT platform with Matlab integrated on it. To establish a connection between the system and ThingSpeak we equipped the Arduino Uno microcontroller with a Wi-Fi shield (Figure 10).

After configuring a WPA2 Wi-Fi connection, we try unsuccessfully to send http requests to ThingSpeak through the API exposed by the platform itself.

We soon discovered that the free version of ThingSpeak allowed to send data to the platform with a limited rate of 1 call every 20 seconds. Consequently, we attempted to save the data in a json file so that we could send all the consumption information collected in 20 second with a single http request. But this solution presented two non-negligible problems:

- The system lost its real-time acquisition property (at least from the server side point of view). Therefore, it was possible to perform plotting and analysis operations only after the data had been collected;
- The micro-controller has a limited memory capacity: the final sketch loaded on the board occupied more than 90% of its random access memory so that it resulted unstable in its behaviors.

This complications force us to evaluate another different solution.

4.3 Final approach: Node-Red, MySQL Database Server and AWS network

Node-Red is an open IoT-oriented programming tool for wiring together hardware devices, APIs and online services through its browser-based flow editor. We built a flow able to receive multiple http requests from the Arduino micro-controller and save the consumption data on a MySQL Database Server. Finally, we decided to move the architecture on a Linux Virtual Machine instantiated on Amazon AWS.

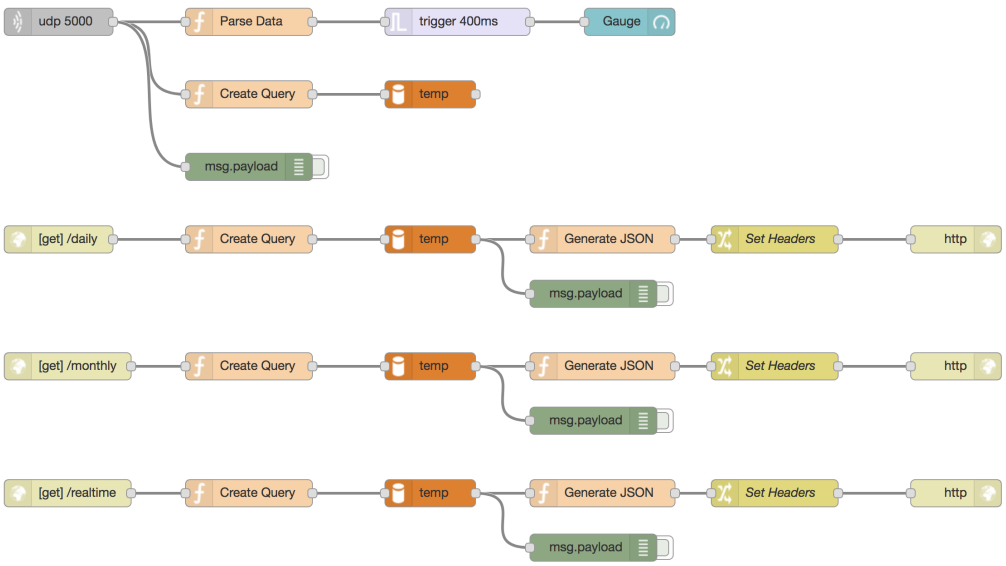


Figure 11: The Node-Red flow

This approach seemed to be a great solution but soon we noticed that incrementing the calls rate, many requests failed. The reason was attributable to the low computational potentialities of the micro-controller in sending multiple requests: the loop time of Arduino was too high to open a socket connection to the server, execute the three-way handshake phase, send the data and close the connection, using a TCP protocol. In this way, we could not get below the threshold of 2 second per call, a period that although acceptable, did not allow a real-time system to be obtained. Furthermore, the system was still rather unstable. For this reason, we were forced to give up the reliability of the requests slightly, in order to have a fair compromise with the efficiency of the system: using a UDP connection instead of a TCP protocol, the calls rate could now grow up to 200ms per request. We had finally got a system that would guarantee a still good reliability and would transmit data in real time. The database server allowed to download the csv file of the data saved on it. Furthermore, the acquisition timestamp registered by the database server granted to solve in an easy way, another non-negligible problem. As a matter of fact, we could use the Wi-Fi connection to have a clock in the micro-controller and use it to memorize the sampling time directly inside the sketch. But, once again, the choice to implement this functionality directly on the Arduino would have resulted in an

excessive use of the dynamic memory and the computational power of the microcontroller itself, generating instability. On the contrary, the database server saved automatically the instant in which any entry was inserted on it.

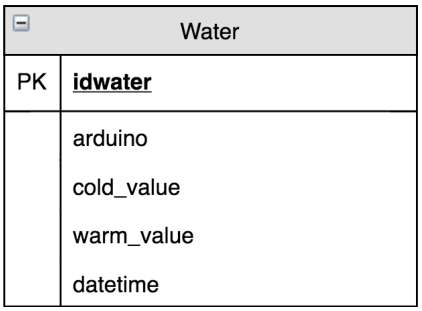


Figure 12: The ER Diagram

Observation: The final solution continued to present the opportunity to save the data on a SD card. This choice was motivated by the possibility to have failures on the Wi-Fi connection (albeit remote). A priori, not being aware of the latency time from a disconnection to the consequent repair of the connection, offering a way to store the data without the need of a connection kept the system away from losing a non-negligible amount of data.