# FINAL PROJECT EMBEDDED SYSTEMS

Breayann Ortiz Aldana, breayanortiz@javeriana.edu.co , *Member, GREEN VIBES,*

***Abstract*—This article presents the development of a system focused on measuring water flow, temperature, and proximity to deliver reports to a possible end-user, this data is captured by a Raspberry Pi 3B + and processed to be sent to an IoT server.**

***Index Terms*—Systemd, python, raspberry pi.**

## I. Introduction

IN this document, the proposed development for a water measurement system will be presented in a general way, which aims to establish the foundations of a project that seeks, through the data captured by the sensors, to generate awareness in users seeking to guide them to what makes a sustainable consumption of water.

### A. Summary

The name given to the developed system is ONE DROP, this system is composed of a series of sensors focused on the measurement of flow, temperature, and object detection, it also has actuators, specifically solenoid valves, which can be actuated remotely. The project was developed using mainly an SBC, the Raspberry Pi, which allows the use of high-level languages to program the tasks to be carried out. The language that was used the most in this project was Python, although it should be mentioned that some system configurations were made with Linux commands. The code developed in general makes use of several modules or libraries that allow performing tasks with ease, for example, the handling of threads of the processor to perform tasks in parallel. The system, ONE DROP, captures the information of interest, flow, temperature, and detection of objects, and later, through MQTT, send them to the internet, punctually to an IoT server, in this case, it is Thingspeak, which is free and very easy to use. use. For remote activation, there is a web page that allows you to interact with Thingspeak services and remotely activate or deactivate the actuators.

### B. Motivation and justification

An average person consumes 100 liters of water per day (direct consumption), and a water tap with a leak can waste more than 30 liters per day [1]. The above allows us to see how a simple leak can waste up to 30The problem is that if actions are not taken, the water resource will not be enough for the growing demand, which is why implementing technologies that raise people's awareness is essential since as users of this resource we are the key piece to generate a change. Current water prices are often too low to limit excessive use by wealthy households or industries [2], but the solution is not to raise the costs of the resource or create policies that charge more to those who use more, the The solution is in each one of us, if we learn how important this resource is for everyone and we have knowledge of the consumption that we carry out, it will be possible to change the way of thinking and thus generate a viable future.

### C. Article structure

The structure proposed for this document focuses on the following sections, first the related work is addressed, where the most relevant aspects of the research carried out are highlighted and the comparison with the current project is made, after this the proposed solution is presented , where it is intended to expose and explain the development carried out, in the experimental configuration section a very simple assembly is presented, which tries to evaluate the most relevant aspects to consider the application as functional and viable to continue working. As a final development section, the results and the analysis carried out are presented, where concrete results will be presented that will contribute to the justification or evidence of the functionality of the system. Finally, the section on conclusions and future work will address aspects to improve and possible ways to continue in the development of the project.

## II. Related work

In [3], a water consumption measurement system is presented, to generate reports (through emails) that expose the user to their consumption and generate alerts when an established limit is exceeded. This document allows us to observe one of the shortcomings in the majority of systems and the fact is that there is no main purpose related to water consumption, only the system is developed to provide the user with consumption data, but this information does not present more than a report, which as we know, a company of aqueduct does it easily and for free. The way in which the flow measurement is carried out is by using the YS-S201 sensor, which will be used in the ONE DROP system, the platform or development board used is Arduino and the user interface to present the data is developed in Thingspeak. Something to highlight from this work [3] is that the individual consumption generated by the kitchen faucet and the bathroom faucet is presented, this allows us to observe how, by integrating more devices that measure the flow in a home, it is possible to get to identify behaviors that are sectorized or classified by areas of the home. In [4] the development of a system focused on

the detection and alert of leaks in piping systems is presented, to generate alerts on time, this document focuses on one of the most significant problems in terms of the correct use of the water, since the losses of the resource due to the delay in detecting leaks are very high. The development of the system has as its main microcontroller an ATmega328, which captures the signals of interest (flow, flow direction) to later send them to an ES8266, which sends the information to the internet. To make sense of the search carried out, a brief

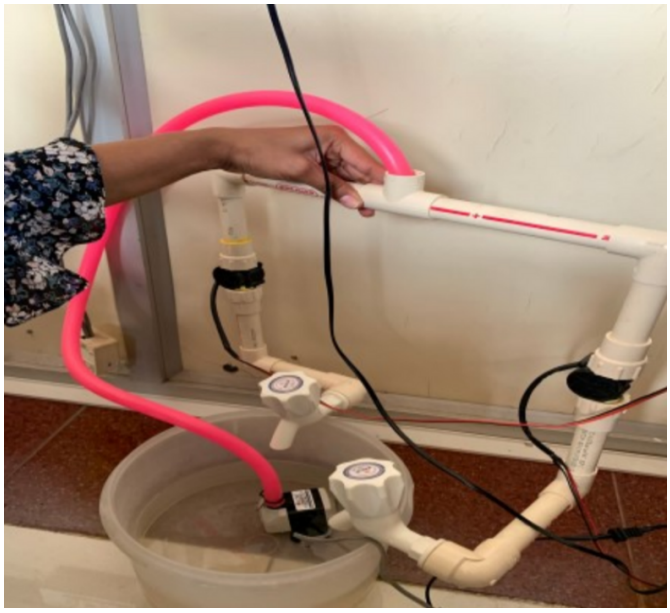

Fig. 1. experimental assembly water consumption measurement system

comparison is made between the works found and ONE DROP, the most significant characteristic that differentiates our project is that most documents present a system that focuses on only delivering the consumption data to the user unlike ONE DROP that raises the possibility of performing an analysis on this data to deliver a clear value proposition, with a simple objective, to generate awareness by showing the user their consumption profile and allowing them to interact with the tap to generate better habits.

## III. Proposed solution

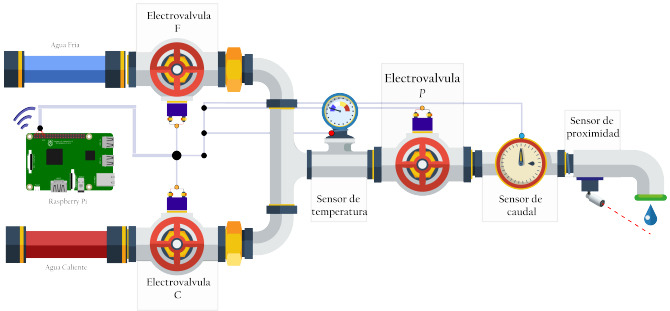


Fig. 2. General diagram of the proposed solution

To solve the problem raised in the motivation and justification section, a system is proposed that has as its main processing unit a Raspberry Pi 3B + board, which will be in charge of directing each of the sensors and actuators, in addition to uploading data to the internet, since this board has several features, including connectivity to a network through WIFI. Regarding the measurement of the physical variables of interest, there are 3 sensors, which are very commercial and have very good support, which allows accelerating the development of the code. As for the actuators, the use of 3 solenoid valves is proposed, these require voltages higher than those of the system operation, it is for this reason that it is also necessary to use devices that allow handling these voltage levels, for example, a relay. In the present development, the design of the control circuit of these solenoid valves is not considered since their control lies in a binary control signal, which can be suitable in many ways to directly manipulate the solenoid valves.

### A. Solution architecture

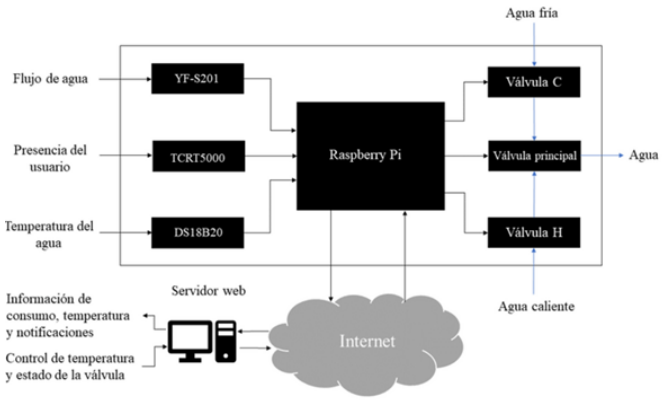


Fig. 3. High-level schematic of the proposed system

The diagram presented in figure 3, exposes the proposed architecture for the system, in this diagram you can see the integration of the sensors and actuators to the Raspberry Pi, and how it interacts with the internet to give global access to the information collected.

**YF-S201**: The water flow sensor, which measures the flow coming out of the tap and collects consumption data. Its output is a digital pulse signal with which the flow is calculated according to the frequency.

**TCRT5000**: Proximity sensor, which is responsible for identifying the presence of the user to supply the water. Its output is a digital signal.

**DS18B20**: Temperature sensor with which the temperature in the water will be measured. Its output is a sequence of bits using the 1-Wire protocol to indicate temperature. Raspberry Pi 3: The central control unit, where the reading and interpretation of the sensors and the management of the valves would be carried out. This block must connect to the Internet to perform information processing and additionally interact with the user through the webserver.

**Valve C**: The solenoid valve that connects to the cold water tank, through which the user can control the flow of cold water. It receives a signal from the Raspberry Pi 3 to activate or deactivate it.

**Valve H**: Like the solenoid valve C, unlike that, it connects with hot water. Main valve: The outlet valve of the entire system, which is responsible for completely managing the water outlet. It receives a signal from the Raspberry Pi 3 to activate or deactivate it

### *B. Modular theoretical development*

As a modular development, the following scheme is proposed, focused mainly on the code: As the first module, **communication** is developed, focused on the transmission of data under the **MQTT** protocol, the above looks for the correct integration with an IoT service. For the above, the use of Thingspeak as an IoT service is proposed, for this, it is only enough to create an account and a channel to store the data. Thingspeak has a lot of documentation that allows you to configure the communication very easily. To **section the**

```
import RPi.GPIO as GPIO
import time, sys
import _thread
import requests
#from __future__ import print_function #Modulo para utilizar caracteristicas de python 3 en versiones inferior$impo
rt RPi.GPIO as GPIO
import paho.mqtt.publish as publish
import time,sys
from w1thermsensor import W1ThermSensor


## CONFUGURACION MQTT #-------------------------------------------------------------------------------------------
---------------------------------------$#  The Hostname of the ThinSpeak MQTT service
mqttHost = "mqtt.thingspeak.com"
#  Replace this with your Channel ID
channelID = "1520495"
# The Write API Key for the channel
apiKey = "87T7PC6C1FTTHAJJ"
#MQTT Connection Methods
#  Set useUnsecuredTCP to True to use the default MQTT port of 1883: This type of unsecured MQTT connection uses th
e least amount of system resources.
useUnsecuredTCP = False
#  Set useUnsecuredWebSockets to True to use MQTT over an unsecured websocket on port 80.: Try this if port 1883 is
blocked on your network.
useUnsecuredWebsockets = False
#  Set useSSLWebsockets to True to use MQTT over a secure websocket on port 443.: This type of connection will use
slightly more system resources, but th$#  will be secured by SSL.
useSSLWebsockets = True
##End of user configuration   ### -------------------------------------------------------------------------------
---------------------------------------$
```

Fig. 4. Section of the code referring to a fragment of configuration of the communication through the MQTT protocol

**main code**, the following 2 sections are presented, where the topics related to threads and sensor reading are addressed. The

```
#           ------      HILOS       ------
# Electrovalvula Agua Caliente
def SET_EV():
    while True:
        time.sleep(3.5)
        msg=requests.get("https://thingspeak.com/channels/1520495/field/5")
        msg=msg.json()['feeds'][-1]['field5']
        if (str(msg)=='1'):
            STATE_E = int(1)
           #print("Dentro del IF"+str(STATE_E))
        else:
            STATE_E = int(0)
        #print("hilo SET_EV, electrovalvula agua caliente"+str(msg))
        GPIO.output(PIN_Elec_V, STATE_E)

_thread.start_new_thread(SET_EV, ())

# Electrovalvula Principal
def SET_ELEC_VAL():
    while (True):
        STATUS_S_IF = GPIO.input(S_IF)
        GPIO.output(A_ELV,not(int(STATUS_S_IF)))

_thread.start_new_thread(SET_ELEC_VAL, ())
```

Fig. 5. Section of the code responsible for implementing and executing the threads

justification for using threads in the development of this project lies in the fact of the limitations of the IoT server and the time requirements of the application. As for the limitations of the IoT server, the most critical is the update speed that it allows in its free version, 15 seconds. Regarding the time requirements of the project, it is expected that the delay in the reading of the proximity sensor and the opening of the solenoid valve will be minimal. As for the second thread that was used, this is responsible for updating the value of an actuator, it can be the hot water or cold water solenoid valve, the important thing here is that this update must be carried out as soon as possible, once the user requires the change, although as already mentioned there is a limitation in the update speed of the IoT server, with which this thread intends to minimize this time looking for the speed of the state change to fall on the IoT server that is available. **The last module** is responsible

```
while(True):
## ----- READ
  #Sensor de Flujo
    print("Lectura de sensores")
    #time.sleep(4)
    start_counter = 1
    time.sleep(15)
    start_counter = 0
    FLW = (count /(7.5*15))
  #Consumo
    C = round((C+(FLW)*1/60),2)
  #Temperatura
    TEMP = S_TEM.get_temperature()
    time_tuple = time.localtime() # get struct_time
    time_string = time.strftime("%m/%d/%Y, %H:%M:%S", time_tuple)
    F = open("R_Log.txt", "a")
    F.write("Consumo Total: "+str(round(C,2))+"   Litros""Flujo :" +str(round(FLW,2)) + " Litros/min"+
"   Temperatura: " + str(round(TEMP,2))+"ºC   " + "Activacion remota electrovalvula_C: "+ str(STATE_E) +
"    Activacion local electrovalvula Principal: "+ str(STATUS_S_IF) +"  " + str(time_string) + chr(10))
    #F.write("hola"+chr(10))
    F.close()
    print("ya escribí"+chr(10))
   #Build the payload string
    #print("Inicio_Enviar_MQTT"+chr(10))
    tPayload = "field1=" + str(TEMP) + "&field2=" + str(FLW) + "&field3=" + str(C)

    try:
        publish.single(topic, payload=tPayload, hostname=mqttHost, port=tPort, tls=tTLS, transport=tTransport)
#print("Fin_Enviar_MQTT"+chr(10))
        count=0
    except (KeyboardInterrupt):
        F.close()
        GPIO.cleanup()
        sys.exit()
        break
    except:
        print ("There was an error while publishing the data.")
        GPIO.cleanup()
```

Fig. 6. Code section referring to while loop

for reading the sensors, performing a simple preprocessing, recording the data locally in a file with its respective time stamp, and finally transmitting it through MQTT to the IoT server every 15 seconds.

## IV. Experimental setup

To carry out the tests, a very simple assembly is proposed figure 7, in which the flow, temperature, and proximity sensors are integrated, and 2 LEDs are incorporated as actuators, which allow the theoretical status of each of the solenoid valves to be observed.

This basic configuration allows capturing the signal from the sensors and processing it, to later send it to the IoT server, in the same way, the LEDs allow observing the behavior of the solenoid valves, it should be mentioned that in the architecture and schematic 3 solenoid valves are proposed, but the Final code I only contemplate 2, to speed up the final result, in any case, it is clear that the code made for a solenoid valve is the same, you just have to select and modify the PIN to which the solenoid valve will be connected.

## V. Results

With the help of the experimental setup, some sensors were stimulated to obtain values and in this way to be able to visualize the changes in the data in the IoT server, thus for the flow sensor it can be seen how when the flow changes the consumption changes (the water flow or flow was

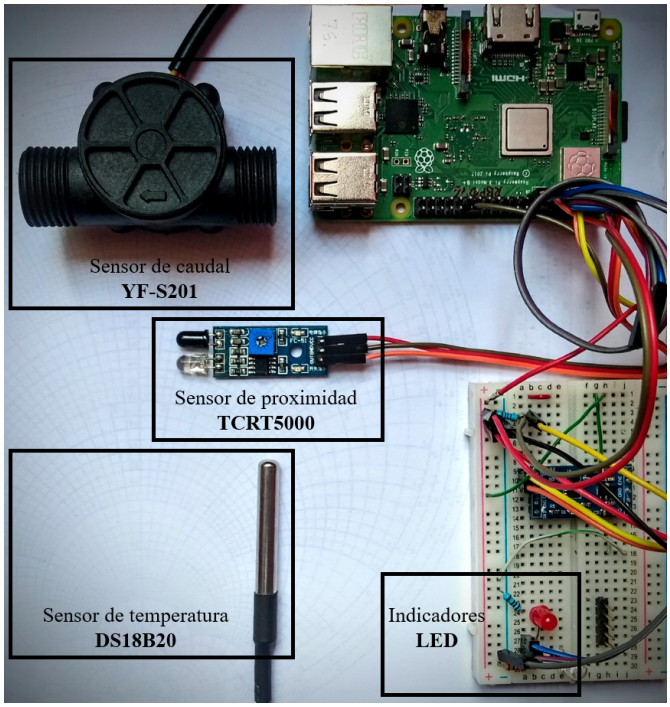


Fig. 7. Proposed experimental setup to perform the measurements

simulated by blowing the sensor to rotate the blades, thus a water flow can be simulated), occasionally this consumption always increases. As for the activation of the solenoid valves

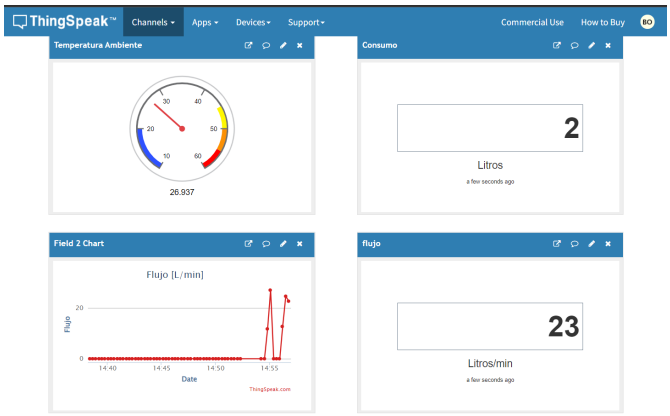


Fig. 8. Graphs and indicators displayed on the Thingspeak page

remotely, it can be done in two ways, the first is by writing the channel using the facilities provided by Thingspeak, the other way is to use the website developed for ONE DROP (http: //onedrop.125mb.com/), this page is for testing and is for testing only. On this page, you can register and access a dashboard, which captures the data recorded by the sensors and can also turn the solenoid valve on or off remotely. It is important to mention that this power button implements a functionality very similar to the one described above, where through the utilities provided by Thingspeak an action to update the value is performed in a channel, it is for this reason that the response of this button is not immediate as it mainly depends on the response speed of the IoT server. Finally, the

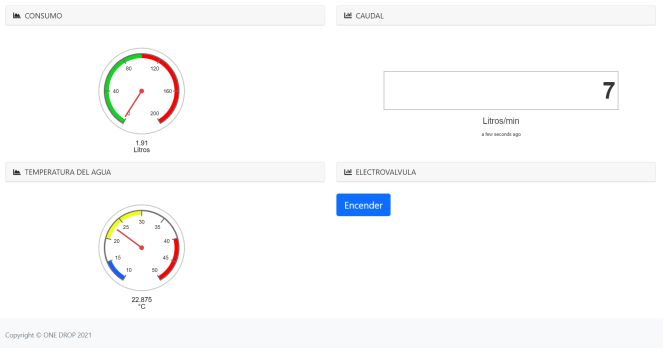


Fig. 9. Dashboard implemented on the ONE DROP test page

tests carried out to allow us to observe the correct operation of the two implemented wires, since when an object is brought closer to the proximity sensor the LED indicator lights up, in the same way, the wire in charge of updating the status of the solenoid valve works, and as It was said before with a considerable delay in the update time, this thread incorporates a delay, since making many requests to the Thingspeak page can saturate it, and after a few tests, it was determined that this is an adequate value.

## VI. Conclusions y future work

Using a board like the Raspberry Pi greatly reduces the development time of a beta application, where a feasibility analysis is carried out on a proposed solution. The Python language libraries allow to speed up the interaction of the Raspberry with the sensors or actuators, in addition, some modules help with very specific tasks, such as the module used to read the temperature sensor. Although the Code developed was not discussed in-depth, it should be mentioned that it contains some aspects that require a little more detailed analysis, for example, how the payload is created to transmit the messages through MQTT. In addition, the file that contains all the Code was configured to be called after each restart through service, with the help of systemd, aspects like the previous one are not explained in this work, but understanding the objective to be achieved is clear that going into a detailed explanation of each configuration would make this document unnecessarily long. In conclusion, it is clear that working with a free IoT service represents quite a few limitations, which is why using these IoT platforms is viable for a testing stage, but it is not up to a commercial solution. Analyzing the problem to be solved and the limitations when implementing the solution, it is determined that the next step is to hire an IoT service that allows for the speed that the solution requires. Also as a step to follow, a model must be made that contains all the sensors and actuators, preferably on a real scene, where data can be obtained from the consumption of a person. Finally, I see it very important to perform an analysis on this data, which provides the user with easy to understand information, this data analysis can be the differentiating factor concerning the other projects analyzed since a consumption data is easy to deliver, but an analysis of user behavior and habits allows the generation of tips and alarms that correctly guide the user.

## VII. Appendices

The code developed for this project can be access form: https://github.com/BreayannOrtiz1.

## References

[1] ControlPublicidad.com. 2021. , . *[online] Available at:* https://controlpublicidad.com/archivo/un-grifo-que-gotea-10-veces-por-minuto-desperdicia-2000-litros-de-agua-al-ano Accessed 24 November 2021.

[2] G. L. Harika, H. Chowdary and T. S. Kiranmai , *Cloud-based Internet of things for Smart Water Consumption Monitoring System," 2020 5th International Conference on Communication and Electronics Systems (ICCES), 2020, pp. 967-972, doi: 10.1109/ICCES48766.2020.9138074.*

[3] J. Fikejz and J. Roleček, *"Proposal of a smart water meter for detecting sudden water leakage," 2018 ELEKTRO, 2018, pp. 1-4, doi: 10.1109/ELEKTRO.2018.8398316.*

[4] H. Mohapatra and A. K. Rath, *"Nub Less Sensor Based Smart Water Tap for Preventing Water Loss at Public Stand Posts," 2020. doi: 10.1109/MTTW51045.2020.9244926.*