DIT-XX/18

Universidad Católica Boliviana "San Pablo"
Facultad de Ingeniería
DIPLOMADO EN INTERNET OF THINGS (1RA VERSION)

PROYECTO PRESENTADO A LA FACULTAD DE INGENIERÍA PARA LA OBTENCIÓN DEL GRADO ACADÉMICO DEL DIPLOMADO EN INTERNET OF THINGS

SISTEMA DE MONITORIZACIÓN DE FLUJO DE AGUA DOMICILIARIO BASADO EN IOT

Realizado por:
ING. DAVY ALFONSO ROJAS YANA

2018

SISTEMA DE MONITORIZACIÓN DE FLUJO DE AGUA DOMICILIARIO BASADO EN IOT

1. ÍNDICE GENERAL

1.1. ÍNDICE DE FIGURAS

1.2. ÍNDICE DE TABLAS

2. RESUMEN DEL PROYECTO

El presente documento contiene el compendio de la metodología implementada para el desarrollo de un prototipo de dispositivo para monitorización de flujo de agua basado en Internet of things. El prototipo del sistema cuenta con dos elementos principales, un NODO GATEWAY y un NODO SENSOR. El NODO GATEWAY se encarga de recopilar la información de los NODOS SENSOR y a través de su puerto Ethernet se encarga de enviar la información mediante el protocolo MQTT hacia un broker definido. Se plantea además la implementación de una interfaz web básica que sirva para visualizar el estado de los sensores, visualizar una alarma en caso de fuga en la red de agua y poder controlar de manera remota los actuadores de la red.

Se hace énfasis en un diseño de bajo consumo eléctrico, por lo cual se plantean las elecciones tanto del actuador ideal para el sistema como la tecnología de comunicación de la misma, esto debido a que se desea diseñar un dispositivo que pueda ser instalado en cualquier parte de una vivienda, que muchas veces no podrá tener al alcance una toma de corriente domiciliaria y deberá trabajar con baterías.

Ya que se plantea el bajo consumo de los dispositivos y su versatilidad en cuanto al montaje en cualquier parte de una vivienda. Se propone la construcción de una WSN (Wireless Sensor Network). Evaluando así cuál sería la mejor tecnología de comunicación en este tipo de aplicaciones. Finalmente se ha realizado la construcción de un prototipo de prueba de concepto del sistema propuesto en el presente proyecto.

3. PROBLEMÁTICA GENERAL

En la última época se ha producido en nuestra ciudad y en diversas partes del mundo una inminente escasez de agua debido a factores meteorológicos y climáticos. Siendo el agua uno de los recursos vitales para nuestra existencia, la mala administración de dicho recurso junto con la falta de políticas sostenibles para su cuidado ponen en riesgo el abastecimiento del líquido elemento en nuestra ciudad y por qué no en todo el mundo.
Un informe [1] realizado por la OCDE en 2012 (Organización para la Cooperación y el Desarrollo Económico) acerca del desperdicio de agua que tomó en cuenta 48 ciudades alrededor del mundo, dió a conocer en qué porcentajes se desperdicia el agua en dichas urbes. El siguiente gráfico muestra el desperdicio de agua por ciudad de algunas de las estudiadas.

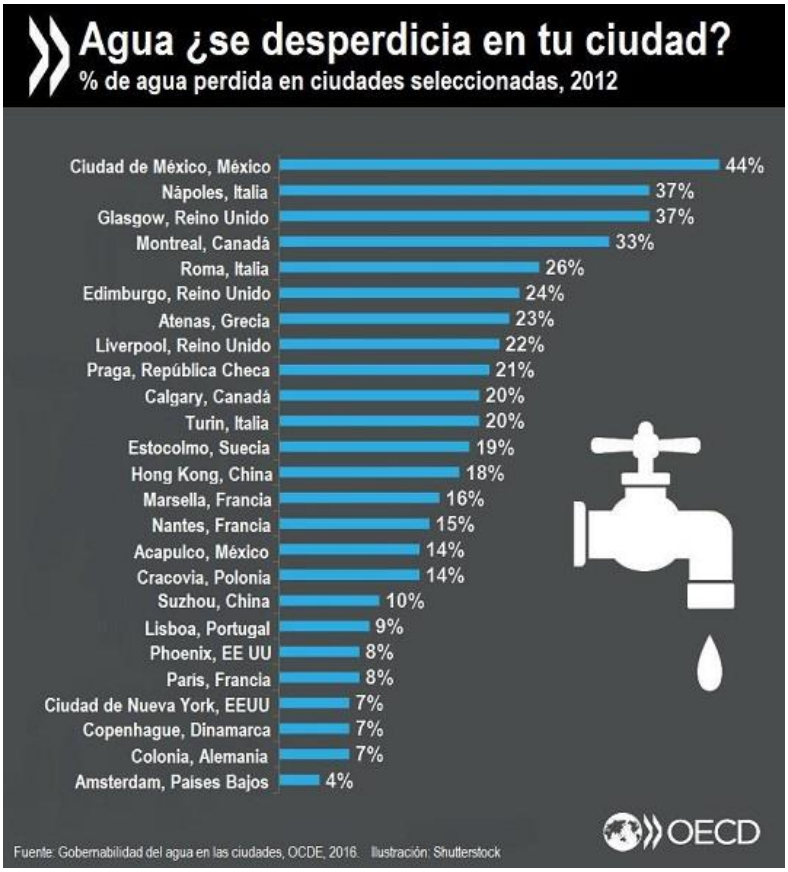


Figura 1: Porcentaje de agua desperdiciado Fuente: OECD
Fuente: Desperdicio de agua en las ciudades [1]

Dicho estudio afirma además que de este porcentaje desperdiciado, en promedio el 60% del mismo se debe a la mala administración doméstica por parte del usuario final y el 40% por problemas en las redes de distribución de la ciudad. Si bien aún no se cuenta con un estudio propio en Bolivia, el problema es latente a nivel mundial, se ha podido notar la falencia de medidas en noviembre de 2016 donde la ciudad de La Paz ha sufrido después de alrededor de 25 años una escasez del líquido elemento.
Otro problema que existe actualmente es el uso regular de recursos humanos de parte de las empresas que facturan el suministro de agua potable, cuyos trabajadores realizan la toma de datos de agua consumida manualmente, la llevan a la central y a partir de esa información se realiza la facturación correspondiente. Si bien el presente proyecto no hace énfasis en la

solución de este problema, (dado que la arquitectura IoT necesaria es diferente a la planteada) puede servir como un paso necesario para el inicio de la automatización de este tipo de mediciones que se realizan actualmente.

4. LÍNEA DE INVESTIGACIÓN

El presente proyecto de diplomado sigue la línea de investigación determinado como Desarrollo superior de Ingeniería. Esto debido a que se propone el diseño de un dispositivo mecatrónico para la monitorización y control de flujo de agua domiciliaria con características referentes al uso de la tecnología IoT.

5. OBJETIVOS GENERALES Y ESPECÍFICOS

5.1. OBJETIVO GENERAL

Diseñar un prototipo para la monitorización de flujo de agua domiciliaria basada en Internet of Things.

5.2. OBJETIVOS ESPECÍFICOS

- Identificar el tipo de sensor adecuado para implementar en la aplicación.
- Elegir el tipo de actuador adecuado para el cierre de flujo de agua del sistema.
- Determinar qué dispositivos estarán involucrados en el sistema IoT.
- Diseñar la arquitectura de los componentes Hardware y Software del sistema IoT.
- Establecer la tecnología óptima de comunicación del dispositivo para su interacción con el sistema.
- Definir la topología de red adecuada para los dispositivos.
- Establecer las configuraciones de conexión para el envío de datos a la nube utilizando el sistema microcontrolado.
- Desarrollar una interfaz Front End básica para el manejo del sistema

.

6. JUSTIFICACIÓN

a) Social

Dado que la problemática que se trata de abarcar con el presente proyecto está relacionado a uno de los recursos básicos para la vida como es el agua. Su cuidado es uno de los aspectos más importantes que se deben tomar en cuenta, más aún en estos tiempos en los que el cambio climático ha afectado en gran medida el suministro de agua en nuestra ciudad y en diversas partes del mundo. Con este proyecto se plantea un sistema que pueda monitorizar e incluso detectar fluctuaciones del líquido elemento anormales, de manera que el sistema pueda accionar válvulas para la conservación del mismo.

b) Técnica

La justificación técnica viene dada por la propuesta de un sistema IoT que simplifica la monitorización del consumo de agua de un domicilio, que puede ayudar al usuario a administrar su consumo de agua mensual, observar indicios de fugas de agua debido a fluctuaciones anormales e incluso corroborar que la empresa que se encarga del suministro de agua esté facturando justamente su consumo.

c) Económica

Respecto al aspecto económico se justifica debido a que el prototipo podría llegar a detectar fugas o consumos anormales de agua en una vivienda, lo que haría que el usuario pueda tomar medidas correctivas en tiempos más cortos que en los que se puede detectar una fuga normalmente. Lo que reduce el desperdicio del líquido que a su vez se verá reflejado en el pago por el servicio mensual.

d) Académica

En la justificación académica se abarcan la implementación de soluciones utilizando principios de sistemas embebidos, como también el uso de tecnologías de comunicación como principal fuerte. Se abarcan conceptos y contenidos de prácticamente los cinco módulos que abarca el diplomado. Todo esto para el diseño y desarrollo del prototipo IoT.

7. LÍMITES Y ALCANCES

Como se menciona el presente proyecto abarca el diseño de un sistema IoT para la medición de flujo de agua, teniendo esto en mente, los aspectos que limitan el presente trabajo son los siguientes:

- Solo se plantea la medición del flujo de agua de un pequeño sistema, no así otros parámetros como ser la presión en las tuberías o la temperatura del líquido.
- El sistema IoT está planeado para ser implementado en una vivienda simple.
- En el presente documento sólo se planteará el uso de 3 nodos de una WSN.
- Diseñar los nodos sensores del sistema para que puedan funcionar de manera autónoma con baterías.
- El servicio Web planteado es básico de manera que solo sirva para telemetría y manipulación de los actuadores.
- El prototipo es de tipo invasivo, es decir que requiere cierta modificación en la red de tuberías de la vivienda.
- No se abarca en profundidad el tópico de seguridad de la información.

En los alcances del trabajo se destacan:

- Se plantea el uso de sensores y actuadores fácilmente accesibles en el mercado para reducir costos de implementación.
- Se hace énfasis en que los nodos tengan un bajo consumo de energía debido a la localización de los mismos.
- Se plantea una arquitectura domótica sencilla para el fácil añadimiento de más nodos a la WSN.
- Diseño de una interfaz simple fácilmente accesible para el usuario.
- En el software se plantea un procesamiento de la información que permita diferenciar al sistema cuando se está produciendo un flujo anormal en las redes de agua de la vivienda.

8. DISEÑO Y DESARROLLO DEL PROYECTO

8.1. SELECCIÓN DEL SENSOR

Como parte principal del proyecto el fin es medir el flujo de agua que circula por una tubería de un diámetro determinado. Para ello debemos hacer uso de un caudalímetro. Existen diversos tipos de caudalímetros, muchos de ellos diseñados específicamente para ciertas funciones, aunque la mayoría para entornos industriales, debido a que es en ese sector donde más se hace uso de la medición de este tipo de variables. Por lo tanto se verá un breve resumen de algunos tipos de sensores existentes para seleccionar eficientemente cuál es el más adecuado para la presente aplicación. Como premisas básicas para la selección de un dispositivo adecuado debemos tomar en cuenta que cuente con voltajes compatibles y tipo de salida compatible con un microcontrolador, que es el tipo de sistema que se tiene planteado utilizar.

8.1.1. TIPOS DE CAUDALÍMETROS

Figura 2: Clasificación de caudalímetros de acuerdo a su principio de funcionamiento
Fuente: Libro Instrumentación industrial, Creus Sole Antonio [2].

8.1.2 CRITERIOS DE SELECCIÓN

Los criterios que se consideran para la selección del sensor son los siguientes:

Rango de caudales: Para conocer el rango de caudales que debe cubrir el sensor, debemos tener una noción del consumo de agua pico de los aparatos que consumen el líquido en las

viviendas, en la **Tabla 1** se despliegan los valores de consumo de aparatos de agua domiciliarios.

Aparato	Caudal (L/s)
Lavamanos	0,04
Inodoro con tanque	0,1
Urinarios con grifo temporizado	0,15
Ducha	0,2
Tina L<1,4m	0,2
Fregadero doméstico	0,2
Lavadora doméstica	0,2
Tina L>1,4m	0,3

Tabla 1: Caudales instantáneos típicos por aparato según Norma Española Canaria 119
Fuente: Criterios para definir el diámetro de la acometida y el medidor a instalar en urbanizaciones y edificios, de EPM [3]

Se ha considerado aparatos que típicamente se encuentran en viviendas de nuestro país.

Precisión requerida: Al ser un proyecto prototipo y aplicado en cierta manera a la domótica, este no es un factor muy crítico. Para una precisión aceptable consideraremos que se toma en cuenta por lo general un error de no más del 5% en la medición.

Tipo de señal requerida: Como mencionamos anteriormente, este tipo de salida debe ser de una interfaz sencilla para sistemas con microcontroladores. Lo que quiere decir que sea deseable que sea un sensor con salida analógica o un sensor con salida de tren de pulsos digitales.

Accesibilidad: Debido a los fines de prototipado del presente proyecto, de manera que pueda ser escalable a precisiones y rangos mayores en una implementación real, se busca que el instrumento de medición sea de un presupuesto bajo y que sea fácilmente adquirible en el mercado local.

Siendo estos los criterios de selección más relevantes, se presenta una lista de dispositivos para esta función en la Tabla 2 donde se observa un resumen de las características de acuerdo con el tipo de dispositivo. Donde observando las características se resalta el medidor de flujo volumétrico (caudalímetro) cicloidal, ya que estos son los más comunes en sistemas de uso doméstico y por lo observado son los que poseen las características más adecuadas para mediciones de caudales bajas como lo son los de tipo residencial.

Tipo	Relación de Caudal	Precisión en toda la escala	Escala	Presión Máxima (bar)	Temperatura máxima °C	Pérdida de carga**	Servicio	materiales	Costo relativo	Ventajas	desventajas
Placa	3:1	1-2 %	No lineal	400	500	20 m	Liq/Vapor/Gas	Metal/plástico	Bajo	Simple, económico	Alta dP, fluidos limpios
Tobera	3:1	0.9-1.5 %	No lineal	400	500	16 m	Liq/Vapor/Gas	Metal/plástico	Medio	Simple, preciso	Alta dP, fluidos limpios, caro
Tubo Ventury	3:1	0.75 %	No lineal	400	500	4 m	Liq/Vapor/Gas	Metal/plástico	Muy alto	Preciso, poca dP*	Alta dP, fluidos limpios, muy caro
Tuvo Pitot	3:1	1.5-4 %	No lineal	400	500	-	Liq/Vapor/Gas	Metal/plástico	Bajo	Simple, económico	Poca precisión
Tubo Annubar	3:1	1%	No lineal	400	500	-	Liq/Vapor/Gas	Metal/plástico	Bajo	Simple, económico	Poca precisión
Rotámetro	10:1	1-2 %	Lineal	400	250	5 m	Liq/Vapor/Gas	Vidrio/Cerámica	Bajo	Mayor precisión	Golpe de ariete puede dañarlo
Vertedero	3:1	1-2 %	Especial	Atmósfera	60	-	Líquidos	Metal	Bajo	coste medio	Voluminoso, caro
Turbina	15:1	0.3 %	Lineal	200	250	0.7 b	Líquidos/Gas	Metal	Alto	Preciso, margen amplio	Difícil de calibrar, fluidos limpios
Sónico	20:1	2%	Lineal	100	250	nula	Líquidos	Metal/plástico	Alto	cualquier líquido, poca dP	Caro, difícil de calibrar, sensible a la densidad
Placa de impacto	10:1	1%	No lineal	100	400	0.5 b	Líquidos	Metal	Medio	Fluidos viscosos	Poca capacidad
Magnético	100:1	0.5-1 %	Lineal	20-200	150	nula	Líquidos	Teflón/Vidrio	Alto	poca dP	Caro, sólo para líquidos conductores
Disco oscilante	5:1	1-2 %	Lineal	10-150	120	0.3 m	Líquidos	Metal	Bajo	barato	Par pequeño
Pistón oscilante	5:1	0.2-0.5 %	Lineal	25	150	10 b	Líquidos	Metal	Medio	Líquidos Viscosos, corrosivos	Alta dP
Pistón alternativo	5:1	0,20%	Lineal	25	100	0.2 m	Líquidos	Metal	Alto	Preciso	Voluminoso, caro, alta dP
Cicloidal	10:1	1%	Lineal	100	150	0.3 b	Líquidos/Gas	Metal/plástico	Medio	poca dP	Poca precisión en caudales bajos
Birrotor	5:1	0.2%	Lineal	100	60-200	0,4 b	Líquidos	Metal/plástico	Medio	Preciso	Margen pequeño
Oval	10:1	0.5%	Lineal	100	180	1 b	Líquidos	Metal/plástico	Medio	No afecta la viscosidad	Alta dP
Paredes deformables	10:1	0.3%	Lineal	-	-	-	Gas	Metal/plástico	Medio	Preciso	Voluminoso, alta dP
Torbellino	100:1	0.2%	Lineal	50	100	0.4 b	Líquidos/Gas	Metal/plástico	Medio	margen amplio, poca dP	Caro
Vórtex	10:1	1%	Lineal	50	400	-	Líquidos/Gas	Metal/plástico	Medio	soprota vibraciones	Insensible a bajo caudal

Tabla 2: Características de instrumentos medidores de caudal
Fuente: Libro Instrumentación industrial, Creus Sole Antonio [2].

El tipo seleccionado en verde corresponde al tipo de medidor que se utilizará. *dP hace referencia a la diferencia de presión. **La pérdida de carga se puede medir en metros cúbicos de área (m) o en bares (b).

8.1.3. CAUDALIMETRO TIPO TURBINA

Denominado también medidor cicloidal consisten en un rotor que gira al paso del fluido con una velocidad directamente proporcional al caudal [2]. La velocidad del fluido ejerce una fuerza de arrastre en el rotor generando una diferencia de presiones debida al cambio de área entre el rotor y el cono posterior, quien ejerce una fuerza igual y opuesta. La medición del caudal en este tipo de aparatos se logra con base en la proporcionalidad que existe entre el número de revoluciones o vueltas que dan las aspas del dispositivo, y la velocidad del flujo que es transportada a través del conducto. La velocidad que adquieren las aspas al contacto con el flujo, se transmite a un sistema de relojería o se transduce en pulsos o niveles eléctricos, y de este modo se obtiene información referente a volúmenes y registro del caudal. El transductor utilizado por lo general para obtener la señal eléctrica a partir del movimiento son los de efecto hall. En donde el rotor lleva un imán permanente que se aproxima a un sensor de efecto hall, el incremento de la intensidad del campo magnético por la proximidad del imán genera un pulso en el sensor de modo que la cantidad de pulsos generados en un intervalo de tiempo son directamente proporcionales al flujo a medir.

Tomando en cuenta todos los aspectos observados, el sensor seleccionado para el presente proyecto es un medidor de flujo volumétrico de paletas construido para tubería de ½ pulgada. Este se compone de una válvula de plástico, un rotor (parte giratoria), y un sensor de efecto Hall.

8.1.4. CARACTERÍSTICAS TÉCNICAS DEL MEDIDOR

Las características técnicas del caudalímetro son las que se ven a continuación:

• Modelo: SEN_0394
• Fabricante: Adafruit.
• Compacto, de fácil instalación.
• Enroscado hermético.
• Sensor de efecto Hall de alta calidad.
• Cumple con la norma RoHS.
• Voltaje de operación: 5 a 24V
• Máximo consumo de corriente: 15mA a 5V
• Caudal de trabajo: 1 a 30 Litros/Minuto (cubre todos los aparatos consumidores de agua seleccionados en la tabla 1).
• Rango de temperatura de trabajo: -25 a 80ºC
• Rango de humedad de trabajo: 35%-80% HR
• Presión de agua máxima: 2.0 MPa
• Ciclo útil de salida: 50% +- 10%
• Tiempo de subida: 0.04 µs
• Tiempo de bajada: 0.18 µs
• Características del pulso: Frecuencia (Hz) = 7.5 * Caudal (L/min)
• Pulsos por litro: 450
• Conexión de 1/2" nominal, 0.75" de diámetro externo y rosca de 1/2"
• Tamaño: 63.5 mm x 35mm x 35mm.

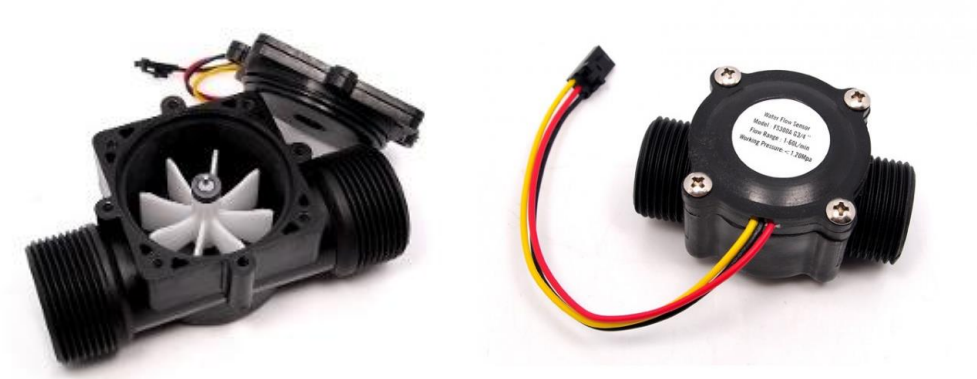


Figura 3: Sensor de flujo de efecto hall
Fuente: fabricantehttps://www.adafruit.com/products/828

8.1.5. CONEXIÓN ELECTRÓNICA

La conexión hacia el microcontrolador es bastante sencilla, puesto a que básicamente solo requiere los dos puntos de alimentación Vcc y GND y una entrada digital, preferentemente de tipo interrupción por parte del microcontrolador, el procesamiento de la información se verá

más adelante para determinar cómo obtener la medición del caudal de acuerdo a la señal generada por el sensor. La conexión a una tarjeta arduino se muestra en la siguiente figura:

Figura 4: Conexión sensor de flujo de efecto hall a arduino Fuente: http://www.naylampmechatronics.com/blog/47_tutorial-sensor-de-flujo-de-agua.html

8.1.6. CALIBRACIÓN

Para la calibración del sensor seleccionado se han tomado en cuenta las siguientes consideraciones:

• El medidor de flujo es calibrado con el líquido a emplear.
• Se han evitado vibraciones que pudieran afectar el sistema.
• El número de valores de flujo seleccionados se encuentra entre 2 y 5 flujos diferentes dentro del alcance del medidor.

Los datos tomados se muestran en la Tabla 3. El microcontrolador despliega el tiempo en segundos y la cantidad de pulsos que se utilizó en cada prueba para llenar la medida de volumen; a partir de estos tres datos es posible aproximar el valor de volumen por pulso (que es constante en la región más lineal del sensor), el valor de la frecuencia en pulsos por segundo y el flujo en diferentes unidades (centímetros cúbicos por segundo, litros por minuto y litros por hora).

Medida Volumétrica (L)	Tiempo de llenado (seg)	Total Pulsos	vol/pulso (CC)	pulsos/seg (Hz)	CC/seg	litros/min	litros/hora
2	124	860	2,33	6,94	16,13	0,97	58,06
1	49	472	2,12	9,63	20,41	1,22	73,47
2	65	928	2,16	14,28	30,77	1,85	110,77
2	42	949	2,11	22,60	47,62	2,86	171,43
2	33	949	2,11	28,76	60,61	3,64	218,18
2	32	925	2,16	28,91	62,50	3,75	225,00
2	31	925	2,16	29,84	64,52	3,87	232,26
2	24	836	2,39	34,83	83,33	5,00	300,00
0,745	10	358	2,08	35,80	74,50	4,47	268,20
2	24	916	2,18	38,17	83,33	5,00	300,00
2	22	920	2,17	41,82	90,91	5,45	327,27
2	21	922	2,17	43,90	95,24	5,71	342,86
2	19	893	2,24	47,00	105,26	6,32	378,95
2	18	881	2,27	48,94	111,11	6,67	400,00
2	18	895	2,23	49,72	111,11	6,67	400,00
2	18	918	2,18	51,00	111,11	6,67	400,00
4	33	1813	2,21	54,94	121,21	7,27	436,36
4	33	1993	2,01	60,39	121,21	7,27	436,36
2,06	16	983	2,10	61,44	128,75	7,73	463,50

Tabla 3: Datos obtenidos en el ejercicio de calibración, la mayor parte de los datos se tomó utilizando un volumen de referencia de 2 litros modificando la velocidad del flujo regulando la misma con la válvula (llave de paso).
Fuente: Elaboración Propia

Se determinó que la forma más conveniente para hacer la calibración de este sensor es efectuar una linealización por tramos. Se han considerado cuatro regiones diferentes de acuerdo a los datos obtenidos y donde a cada una corresponde a una recta del tipo $Y = Ax + B$ Para los valores superiores e inferiores de la escala del sensor estas rectas prácticamente coinciden con los datos del fabricante, en la parte intermedia de la medición las rectas difieren ligeramente de la curva característica entregada.

Una vez efectuado el cálculo matemático de la regresión, las ecuaciones que corresponden al comportamiento del sensor son las siguientes:

$$F(p) = \begin{cases} 7.592p & si \;\; 0 \leq p < 27 \\ 8.6p - 27.2 & si \;\; 27 \leq p < 52 \\ 3.1p + 258{,}8 & si \;\; 52 \leq p < 62 \\ 7.111p + 10.11 & si \;\; p \geq 62 \end{cases}$$

Donde *p* es la cantidad de pulsos por segundo en Hertz (frecuencia) y *F* es el flujo medido en Litros/hora. En la **Figura 8** se muestra el resultado de la linealización respecto a la dispersión medida.

8.2. SELECCIÓN DE ACTUADOR

En este apartado planteamos el uso de dos posibles soluciones en cuanto al actuador de proceso que requiere el nodo sensor. Como primer actuador posible para el control de flujo de agua se tiene a la que es utilizada típicamente para este tipo de aplicaciones, es una válvula de tipo solenoide. Sin embargo al ser el dispositivo planteado para su funcionamiento con baterías, es necesario analizar el consumo de energía eléctrica de dicho actuador. Para ello tomaremos como objeto de análisis la válvula SKU 311070004.

Figura 5: Electroválvula solenoide 2/2 normalmente abierta modelo SKU 311070004.
Fuente: https://www.seeedstudio.com/G1%26amp%3B2-Electric-Solenoid-Valve-%28Normally-Open%29-p-1298.html

Para realizar el análisis de consumo nos basaremos en los datos presentes en su hoja de datos.

- Voltaje de operación 12VDC
- Corriente máxima: 450mA
- Modo de operación normalmente abierto
- Conexiones: G1/2”
- Tipo de válvula: Diafragma
- Máxima temperatura de trabajo: 120°C
- Presión de trabajo: 0.02 – 0.8 MPa
- Número de operaciones: 200,000+

Haciendo un simple análisis del consumo de corriente en una hora de trabajo tenemos que su consumo máximo es de 450 mA, planteamos la activación y desactivación de la válvula por 3 horas, siendo la misma accionada por 15 minutos y desactivada por 15 minutos de manera intermitente. Por lo tanto tenemos que la válvula tipo solenoide, cumple un trabajo en su estado de accionamiento de 1 y ½ horas. por lo tanto, su consumo de corriente es:

$$Consumo\ = 1.5h * 450mA$$
$$Consumo\ = 675mAh\ (\mathbf{1})$$

Esto es el consumo de corriente de dicha válvula realizando el accionamiento intermitente en 3 horas de funcionamiento con intervalos de 15 minutos. Lo que es demasiado para un dispositivo que debe funcionar a baterías.
Para el diseño de un dispositivo de un menor consumo de energía se plantea la siguiente idea, que es el uso de un motor eléctrico DC para el accionamiento de una válvula de bola **(4)** convencional de plomería por medio de acoplamientos mecánicos directamente a la palanca de dicha válvula. Para ello en primera instancia se selecciona el tipo de válvula que se desea utilizar en el sistema, la válvula más comúnmente usada y de mayor facilidad para el diseño de un acople mecánico es la válvula de bola con palanca que se muestra en la **Figura 6:**

Figura 6: Válvula de paso de bola con palanca
Fuente: https://www.ecured.cu/Llave_de_Paso

Este tipo de válvula es de dos posiciones, para el cambio de posición es necesario el giro de la palanca en 90 grados, por lo que realizar el acople de un servo motor es factible. Para la selección del servomotor la característica principal que se requiere es saber el torque con el que se mueve la palanca de la válvula, al ser este elemento una pieza la cual no está pensada para ser accionada con motores o actuadores similares, no se encuentra en las hojas de datos del mismo el torque, por lo que nos vimos en la obligación de con la ayuda de un torquímetro medir el torque necesario. El torque obtenido es de 46 Nm. por lo que se ha considerado el servomotor Tower Pro SG5010.

Figura 7: Servomotor Tower Pro SG5010
Fuente: http://botscience.net/store/index.php?route=product/product&product_id=58
Cuyas características principales son las siguientes:

- Voltaje: 4.8V - 6V DC máx (5V trabaja bien)
- Velocidad promedio: 0.2seg/60grados (@ 4.8V), 0.16seg/60grados (@ 6V)
- Corriente con máxima carga 250mA/5V

- Peso: 39g (1.37 oz)
- Torque: A 5V, 5.5kg-cm, a 6V 6.5kg-cm
- Dimensiones mm: (L x W x H) 40 x 20.0 x 38 mm

Para realizar el análisis del consumo de energía se utiliza la misma hermenéutica que en el caso de la válvula solenoide, asumiendo que se hará un accionamiento intermitente cada 15 minutos durante 3 horas, y tomando en cuenta el consumo máximo de corriente del servomotor. Tomando en cuenta que el servomotor solo necesita ser accionado cuando se requiera un cambio en la posición de la palanca de la válvula de bola. Se tiene que:
En primer lugar se considera el tiempo de trabajo total del servomotor durante el periodo de prueba planteado. Se sabe que para el cambio de posición de la válvula se requiere un giro de 90 grados en la palanca, por lo que el tiempo que le toma al servomotor realizar este movimiento, de acuerdo a su velocidad en sus especificaciones viene dada por:

$$tiempo\ de\ acción\ = velocidad\ angular * ángulo\ de\ giro$$
$$tiempo\ de\ acción\ = \frac{0.2\ seg}{60\ grados} * 90\ grados$$
$$tiempo\ de\ acción\ = 0.3\ segundos$$

El tiempo de real, que necesita el motor para girar 90 grados con un voltaje de 5V es de 0.3 segundos, sin embargo por la experiencia se sabe que este giro debe ser escalado, es decir debe ser con intervalos de tiempo y no así de un solo golpe, por lo que se estima triplicar dicho tiempo para tener un movimiento más suave y seguro de parte del motor para activar la válvula. Por lo tanto consideramos que el tiempo de acción real será de 0.9 seg. Ahora bien, se deben hacer 4 activaciones por hora y en total 12 por las 3 horas de ejemplo planteadas. por lo tanto el tiempo de acción del motor en total es de:

$$tiempo\ de\ acción\ total\ =\ 12 * 0.9seg$$
$$tiempo\ de\ acción\ total\ = 10.8seg$$

Por lo que el consumo de corriente del motor viene a ser:

$$Consumo\ motor\ = tiempo\ de\ acción\ total * Corriente\ máxima$$
$$Consumo\ motor = 10.8seg * 250mA$$
$$Consumo\ motor = 0.003h * 250$$
$$Consumo\ motor = 0.75mAh$$

Se toma en cuenta que una vez accionado el motor, el mismo se desenergiza hasta el próximo cambio de posición para no tener que gastar innecesariamente en la corriente de enclavamiento del motor.
Tomando en cuenta de manera relevante el consumo de energía de ambos dispositivos, queda claro que la alternativa óptima para el desarrollo del dispositivo IoT es la del servomotor. Por lo que para continuar con el desarrollo

8.3. ARQUITECTURA DEL SISTEMA

El sistema de monitorización de flujo de agua basado en IoT, en esta fase de prototipo, comprende dos tipos de dispositivos en cuanto a la red se refiere. El nodo sensor, que es el dispositivo encargado de realizar la medición del flujo de agua y también cuenta con la válvula

motorizada. Y el nodo Gateway, que será el encargado de recolectar la información de los diferentes nodos sensores del sistema y disponer de la información a través de internet. Dado que es una red de sensores ambientado en cierta medida a la domótica, lo que se plantea es el uso de una topología tipo estrella para la comunicación entre los nodos sensores y el nodo gateway. Esto por las características que tiene este tipo de topologías, al ser planteadas para redes sencillas en donde cada dispositivo tiene una comunicación independiente con el nodo central, en donde si un nodo presenta una falla, la comunicación con los demás nodos no se ve afectada, agregar o eliminar dispositivos de la red no afecta en el rendimiento de la comunicación con los otros nodos y debido al direccionamiento, la gestión de energía es mejor.

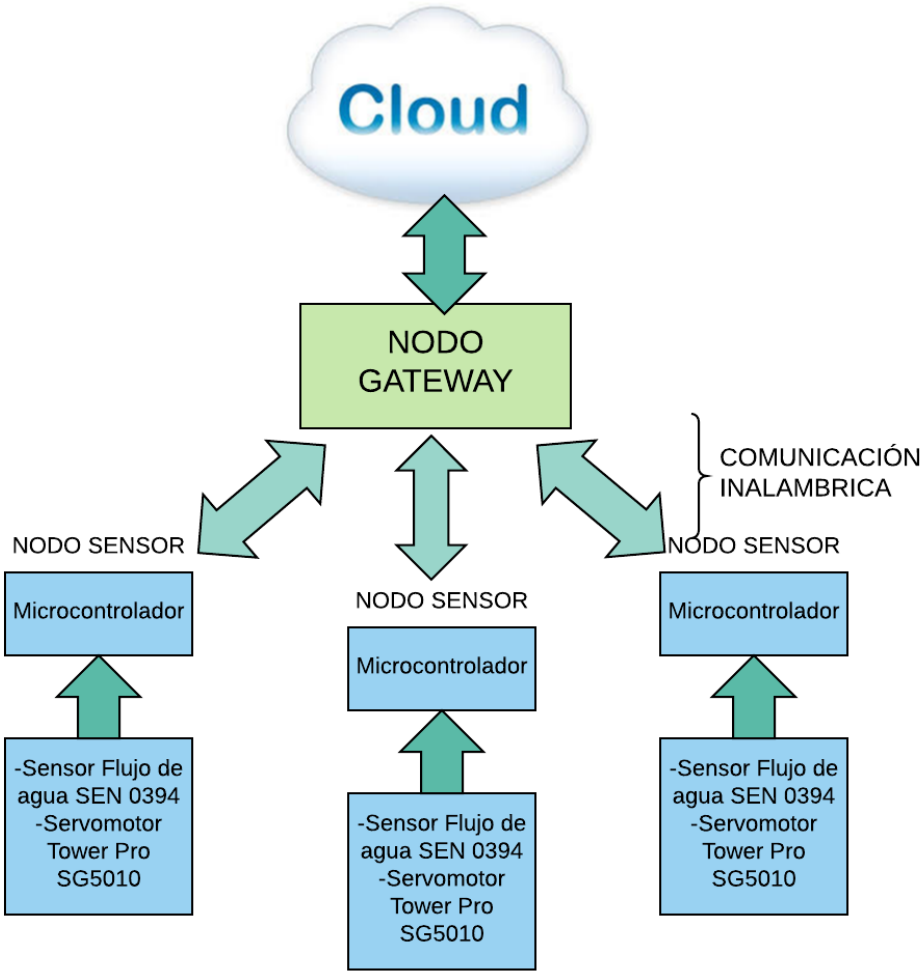


Figura 8: Arquitectura IoT propuesta
Fuente: Elaboración propia

8.4. TECNOLOGÍA DE COMUNICACIÓN

Con la arquitectura propuesta anteriormente, es necesario determinar cuál es la mejor alternativa para la tecnología de comunicación de la red de sensores. Debido a las características inalámbricas que la red requiere, se realizará un breve análisis y comparación de las 3 tecnologías más utilizadas en este tipo de aplicaciones, WiFi, Zigbee y BLE. En la

siguiente tabla podemos observar algunas características técnicas de las tecnologías de comunicación inalámbricas propuestas. Tomando en cuenta que se plantea la construcción de una WSN debido a la distancia a la que se encuentran los dispositivos, que frecuentemente son espacios donde no se cuenta con suministro de energía eléctrica, las prioridades para seleccionar la tecnología vienen a ser alcance y potencia demandada. Por lo que en una primera instancia se descarta la posibilidad de usar BLE, a pesar de ser una tecnología bastante reciente, su alcance está hecho para ser soportada por parte de PAN (Personal Area Network), para dispositivos de uso individual como pueden ser auriculares, mouses, smartwatches, etc. Debido a esto, enfocaremos el tema de la tecnología de comunicación entre WiFi y Zigbee.

Estándar	Bluetooth	UWB	ZigBee	Wi-Fi
Norma IEEE	802.15.1	802.15.3a *	802.15.4	802.1 1a/b/g
Bandas de frecuencia	2.4 GHz	3.1-10.6 GHz	868/915 MHz; 2.4 GHz	2.4 GHz; 5 GHz
Tasa máxima de la señal	1 Mb/s	110 Mb/s	250 Kb/s	54 Mb/s
Alcance nominal	10m	10 m	10 - 100 m	100 m
Potencia	0 - 10 dBm	-41.3 dBm/MHz	(-25) - 0 dBm	15 - 20 dBm
Número de canales de RF	79	(1-15)	1/10; 16	14 (2.4 GHz)
Ancho de banda del canal	1 MHz	500 MHz - 7.5 GHz	0.3/0.6 MHz; 2 MHz	22 MHz
Tipo de modulación	GFSK	BPSK, QPSK	BPSK (+ ASK), O-QPSK	BPSK, QPSK, COFDM, CCK, M-QAM
Extensión de modulación	FHSS	DS-UWB, MB-OFDM	DSSS	DSSS, CCK, OFDM
Mecanismo de coexistencia	Saltos adaptables de frecuencia	Saltos adaptables de frecuencia	Selección dinámica de Frecuencia	Selección dinámica de frecuencia, control de potencia de transmisión (802.1 1 h)
Celda básica	Piconet	Piconet	Estrella	BSS
Ampliación de la celda básica	Scatternet	Igual a Igual (P2P)	Árbol de clúster, Mesh	ESS
Número de nodos de la celda	8	8	> 65000	2007
Encriptación	Cifrado de flujo EQ	Cifrado de clave AES (CTR, modo contador)	Cifrado de clave AES (CTR, modo contador)	Cifrado de flujo RC4 (WEP), Cifrado de clave AES
Autentificación	Secreto compartido	CBC-MAC (CCM)	CBC-MAC (ext. of CCM)	WPA2 (802.11i)
Protección de datos	16-bit CRC	32-bit CRC	16-bit CRC	32-bit CRC

Tabla 4: Comparación entre las diferentes características de los estándares de comunicación inalámbrica más usados.
Fuente: A comparative study of Wireless Communications [4]

Las diferencias de más importancia para el proyecto entre las tecnologías de WiFi y Zigbee se encuentran en la velocidad de transmisión de la información y el consumo de energía eléctrica. Una de las desventajas de los dispositivos y redes basados en Zigbee es que los

mismos no pueden conectarse de manera directa a internet, es necesaria la existencia de un gateway que recolecta los datos de la red Zigbee y los convierte del protocolo Zigbee a IP, algo que implementando una red en WiFi no sería necesario. Sin embargo, observando la velocidad de transmisión de WiFi en la Tabla 4, podemos claramente notar que es una velocidad muy grande, podríamos decir que muy sobredimensionada para la presente aplicación, ya que esa velocidad puede ser usada para transmitir audio y video sin inconvenientes. Y, lo que busca el proyecto, es simplemente la monitorización de estados de los nodos sensores, además como podemos ver en la Tabla 5, la velocidad de transmisión sobre WiFi es acorde a su consumo de corriente. Por lo que no es suficientemente apto para su uso en un dispositivo planteado para trabajar con baterías. Como se viene a recalcando al largo del trabajo una de las prioridades del proyecto es otorgar autonomía energética a los denominados nodos sensores.

Standard	Bluetooth	UWB	ZigBee	Wi-Fi
Chipset	BlueCore2	XS110	CC2430	CX53111
VDD (volt)	1.8	3.3	3.0	3.3
TX (mA)	57	~227.3	24.7	219
RX (mA)	47	~227.3	27	215
Bit rate (Mb/s)	0.72	114	0.25	54

Tabla 5: Comparación energética entre las diferentes características de los estándares de comunicación inalámbrica más usados.
Fuente: A comparative study of Wireless Communications [4]

Observando a grandes rasgos la potencia consumida por las diferentes tecnologías **Figura 9**, podemos concluir que el tipo de tecnología más adecuado para la aplicación de este trabajo es sin duda la tecnología Zigbee, por su característica principal de bajo consumo de energía.

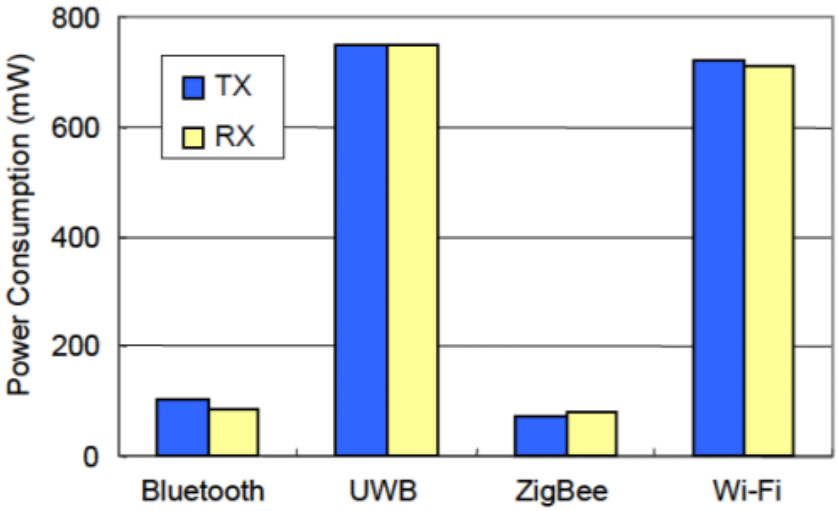


Figura 9: Relación consumo de potencia tecnologías Wireless
Fuente: A comparative study of Wireless Communications [4]

8.4.1. MÓDULO ZIGBEE

El módulo que cuenta con protocolo Zigbee más ampliamente utilizado para el desarrollo de proyectos es el módulo Xbee serie 2, mismo que se adoptará para el presente trabajo debido a su amplia disponibilidad comercial, como a su disponibilidad en cuanto a recursos software como ser programas de configuración y librerías para diferentes plataformas.

Figura 10: Módulo Xbee Serie 2 modelo XB24-Z7WIT-004
Fuente: www.tecbolivia.com

Algunas de sus especificaciones son:

- Interfaz de datos con velocidad de 1200 bps - 1 Mbps
- Banda de 2,4 GHz de frecuencia (aceptada en todo el mundo)
- Rango de temperatura de funcionamiento industrial (-40C a 85C)
- Potencia de transmisión de 1,25 mW (1 dBm)
- Funciona con voltaje de 2,1 a 3,6 VCC; corriente de transmisión de 35 mA, corriente de recepción de 38 mA
- Corriente en power-down
- Nivel de interfaz UART de 3.3V CMOS

8.4.2. CONFIGURACIÓN RED ZIGBEE

Dada la naturaleza de los módulos Xbee, es necesario configurar los parámetros de comunicación de los mismos, esto para crear la red inalámbrica de sensores. Para realizarlo se utiliza el software XCTU, en donde debemos especificar algunos parámetros que serán detallados para configurar la red en topología estrella. Dado que lo que se plantea en el proyecto es la construcción de un prototipo de prueba de concepto, se utiliza una de las configuraciones básicas para el módulo XBee, siendo este conocido como el modo transparente, en donde el módulo funciona como una extensión de un cable serial pero inalámbrica. Se usará en este prototipo debido a que toda la información enviada y recibida por los microcontroladores en los NODOS SENSORES será transmitida por el puerto serial.
En la construcción se implementan dos NODOS SENSORES y un NODO GATEWAY, por lo que en esta red de tres puntos se define al XBee que trabajará con el NODO GATEWAY como Coordinador y a los otros dos que funcionan con los nodos sensores como End Points, todos trabajando con la misma PAN ID y con direcciones propias, dichas configuraciones realizadas en XCTU se pueden ver en el **ANEXO 2**.

8.5. DESARROLLO NODO SENSOR

El nodo sensor consiste básicamente en el control de un servomotor y el sensor de flujo de agua, además de la comunicación en el modo transparente con el módulo XBee, como se puede ver, estas funciones no son muy rigurosas, y las características principales que debe tener el microcontrolador a seleccionar para que pueda cumplir con estas tareas son las siguientes:

- Contar con al menos una salida PWM
- Contar con interrupción por hardware, para el conteo de pulsos del sensor de caudal.
- Tener un puerto serial por software, que funcione a base de interrupciones, esto para el desarrollo de la comunicación con el módulo XBee.

Tomando en cuenta los anteriores aspectos y ya que el fin del proyecto es de desarrollar un prototipo de prueba de concepto del sistema, se ha elegido como la opción de microcontrolador, la tarjeta de desarrollo Arduino Micro. En el siguiente esquema **Figura 12** podemos ver el diagrama general de conexiones del microcontrolador y los diferentes componentes que abarcan el NODO SENSOR:

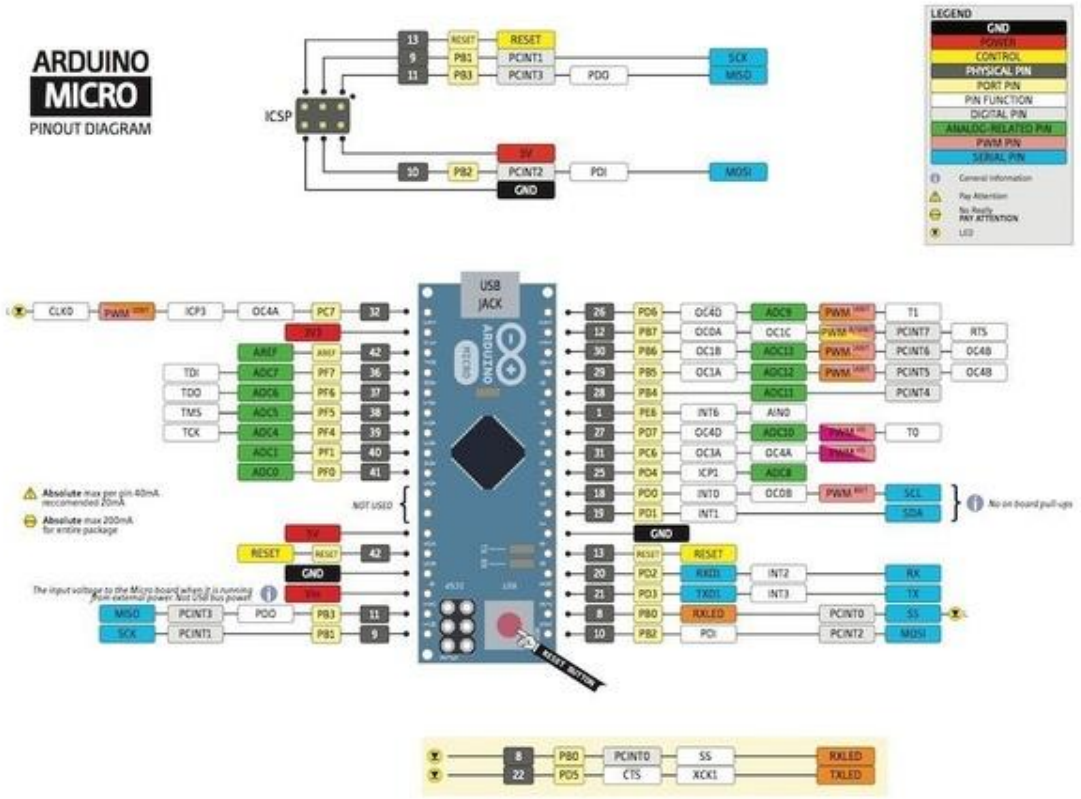


Figura 11: Configuración de pines Arduino Micro
Fuente:https://www.arxterra.com/tlc5940-and-the-arduino-micro/

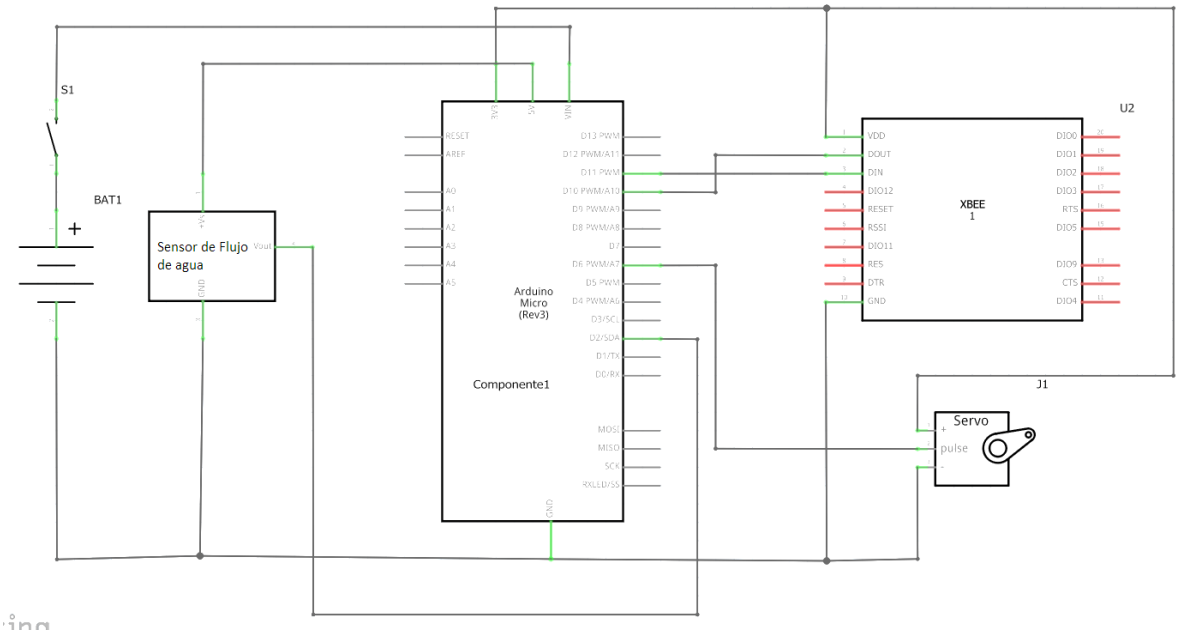


Figura 12: Conexión electrónica NODO SENSOR
Fuente: Elaboración propia

El presente prototipo ha sido armado con la ayuda de un protoboard y con los componentes mencionados anteriormente. A continuación se muestra la construcción del prototipo físico:

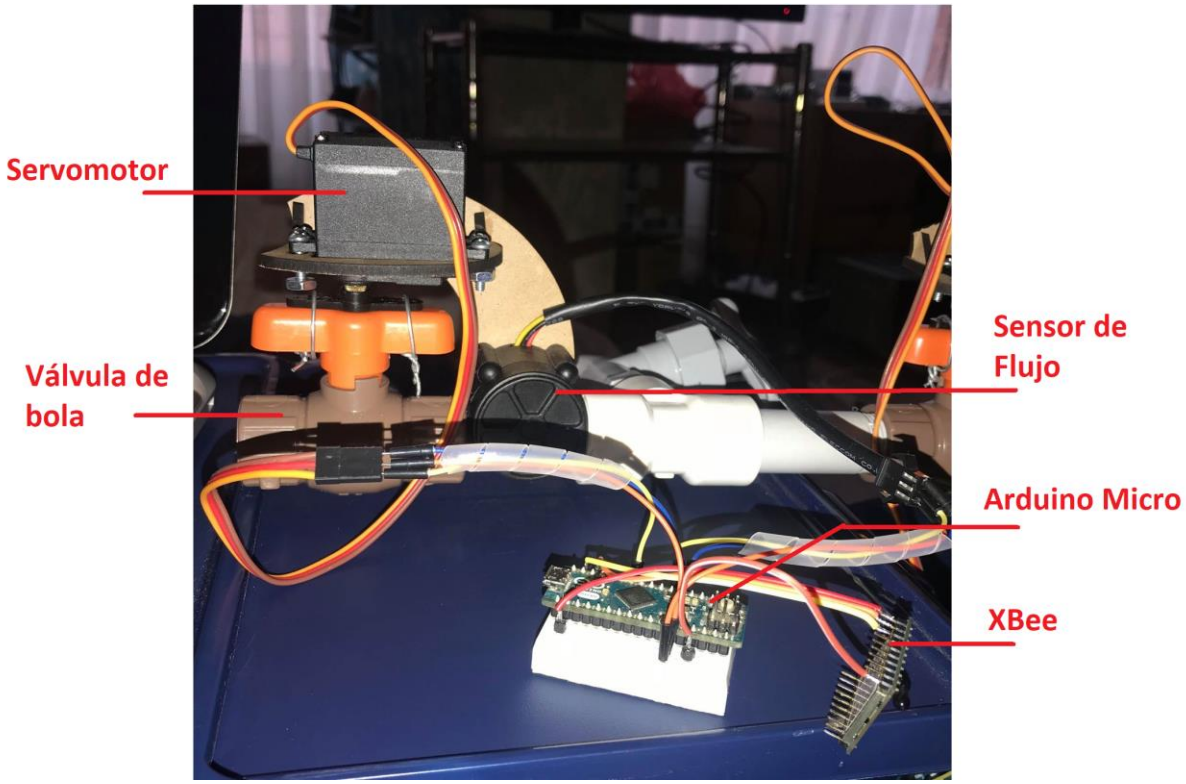


Figura 13: Conexión Física NODO SENSOR
Fuente: Elaboración propia

Ya planteada la construcción de dicho nodo, se muestra a continuación el diagrama de flujo correspondiente, para ilustrar la lógica de funcionamiento de dicho nodo.

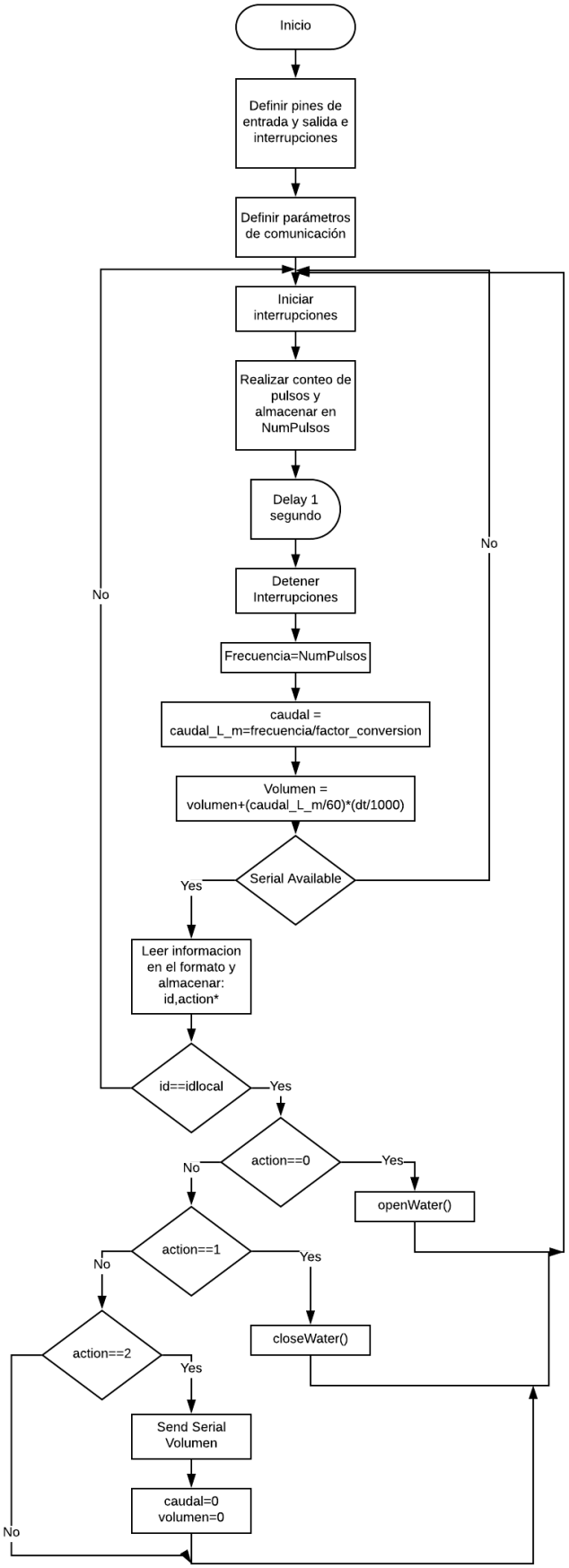


Figura 14: Diagrama de flujo NODO SENSOR
Fuente: Elaboración propia

El programa realizado para el NODO SENSOR se ha desarrollado en el software Arduino IDE 1.8.5, y se despliega en el **Anexo 3**

8.6. DESARROLLO NODO GATEWAY

Dicho nodo como su nombre lo indica debe ser el acceso de la WSN hacia internet, por lo que se ha planteado en el presente prototipo el uso de una placa de desarrollo Raspberry pi B+, en un inicio debido a que se cuenta con dicha placa de desarrollo, pero además tiene un poder de procesamiento mayor a la siguiente alternativa más económica de una gama similar, como viene a ser el shield Ethernet de arduino. Además de cumplir con las dos siguientes características fundamentales con las que debe contar dicho nodo para el correcto funcionamiento del prototipo:

- Contar con puerto Ethernet
- Contar con al menos un puerto serial para la conexión con el módulo XBee en el modo transparente.

Por lo que la elección de la placa Raspberry pi es una elección acertada. La estructura de los protocolos de comunicación se desarrolla más adelante. A continuación se muestra el diagrama de conexión electrónica del NODO GATEWAY

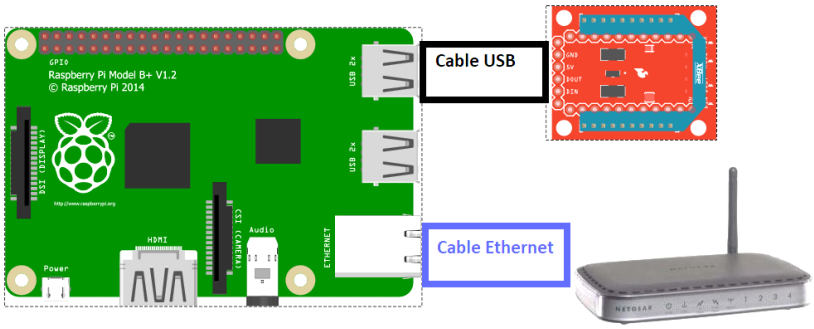


Figura 15: Diagrama de conexión NODO GATEWAY
Fuente: Elaboración propia

Como ya se ha mencionado las únicas conexiones que requiere la placa de desarrollo para la aplicación es la conexión Ethernet para el enlace internet y un puerto USB, a donde se conecta el módulo XBee, mediante su módulo de programación Sparkfun a través de USB.
En la siguiente imagen se observa la conexión física de dicho nodo:

Figura 16: Conexión física NODO GATEWAY
Fuente: Elaboración propia

Dada la conexión propuesta para el NODO GATEWAY, a continuación se presenta el diagrama de flujo bajo el que funciona el respectivo nodo:

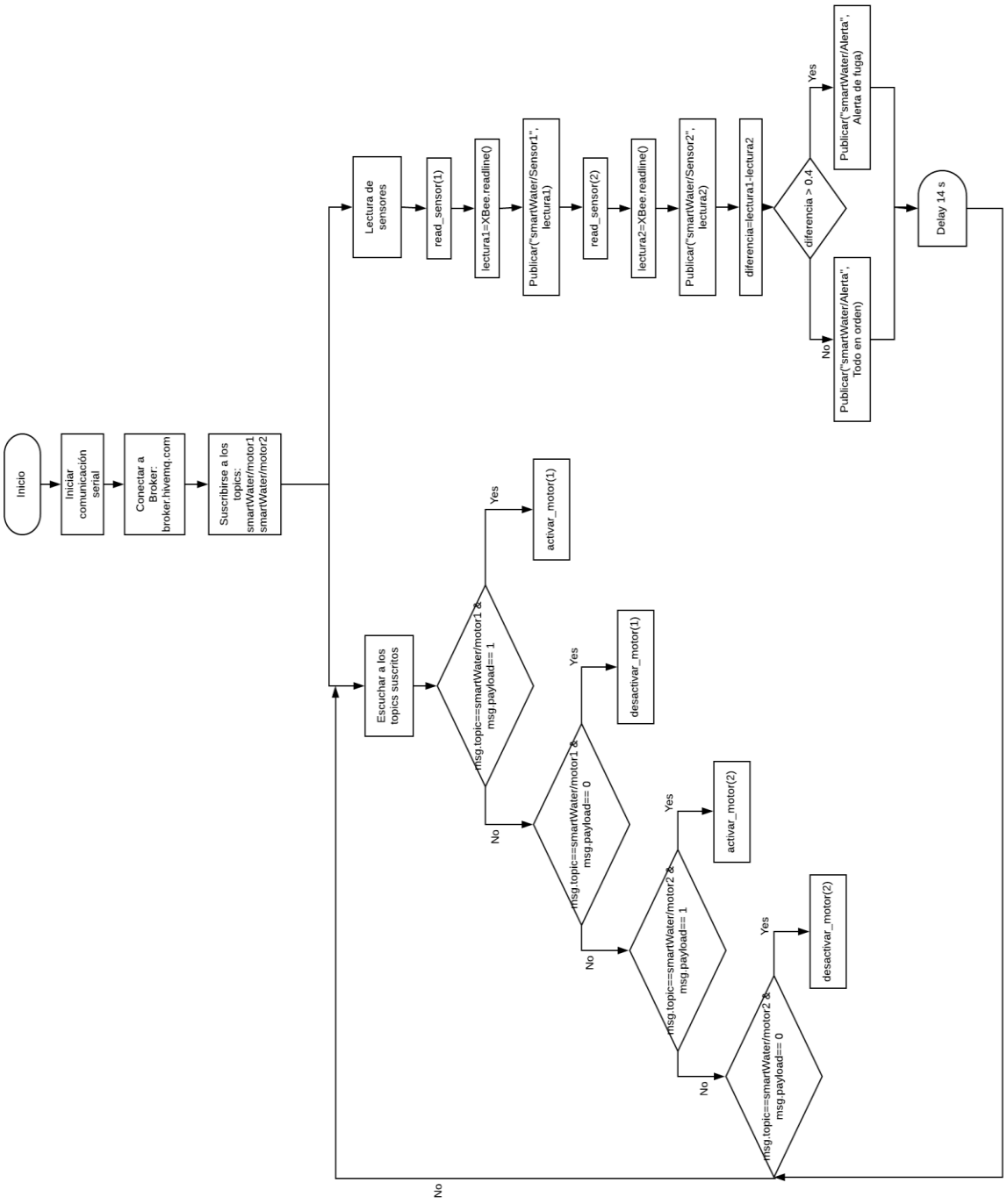


Figura 17: Diagrama de flujo NODO GATEWAY
Fuente: Elaboración propia

El programa para el NODO GATEWAY fue desarrollado en Python 2.7. Dicho programa se ve desplegado en su totalidad en el **Anexo 4.**

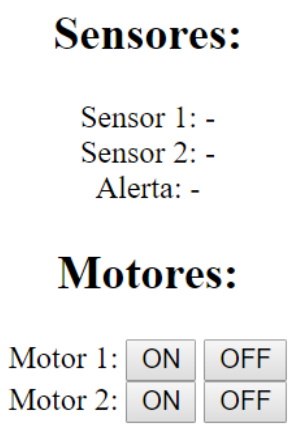
8.7. CONEXIÓN CON INTERNET

El desarrollo de la comunicación con internet fue realizado bajo el protocolo MQTT, en la plataforma Raspberry pi que corresponde al NODO GATEWAY, se ha hecho uso de una librería de eclipse conocida como Paho MQTT para python [5]. A continuación se muestran los tópicos con los que cuenta el sistema IoT en su fase de prototipo y la descripción de su funcionamiento relacionado:

Tópico	Función
smartWater/motor1	Activar Desactivar Motor NODO SENSOR 1
smartWater/motor2	Activar Desactivar Motor NODO SENSOR 2
smartWater/Sensor1	Enviar medición Sensor NODO SENSOR 1
smartWater/Sensor2	Enviar medición Sensor NODO SENSOR 2
smartWater/Alerta	Enviar mensaje en caso de alerta del sistema

Tabla 6: Descripción de Tópicos MQTT
Fuente: Elaboración propia

Para la manipulación del presente proyecto solamente se ha desarrollado una interfaz HTML potenciada por JavaScript, la interfaz desarrollada tiene la siguiente apariencia:



En la parte superior de lo que es la interfaz Web se tiene unos labels con el nombre de Sensor 1 y Sensor 2, dichos elementos despliegan la última lectura de los sensores publicadas por el NODO GATEWAY a través de MQTT, suscritas a los tópicos smartWater/Sensor1 y smartWater/Sensor2 respectivamente. El label Alarma cumple la función de desplegar mensajes de cuando el sistema está funcionando correctamente y muestra una alarma en caso de que se produzca una fuga en el prototipo IoT. Seguidamente tenemos las etiquetas llamadas Motor 1 y Motor 2, las cuales cuentan con dos botones cada uno, dichos botones

funcionan como publicadores sobre los tópicos smartWater/motor1 y smartWater/motor2 respectivamente.

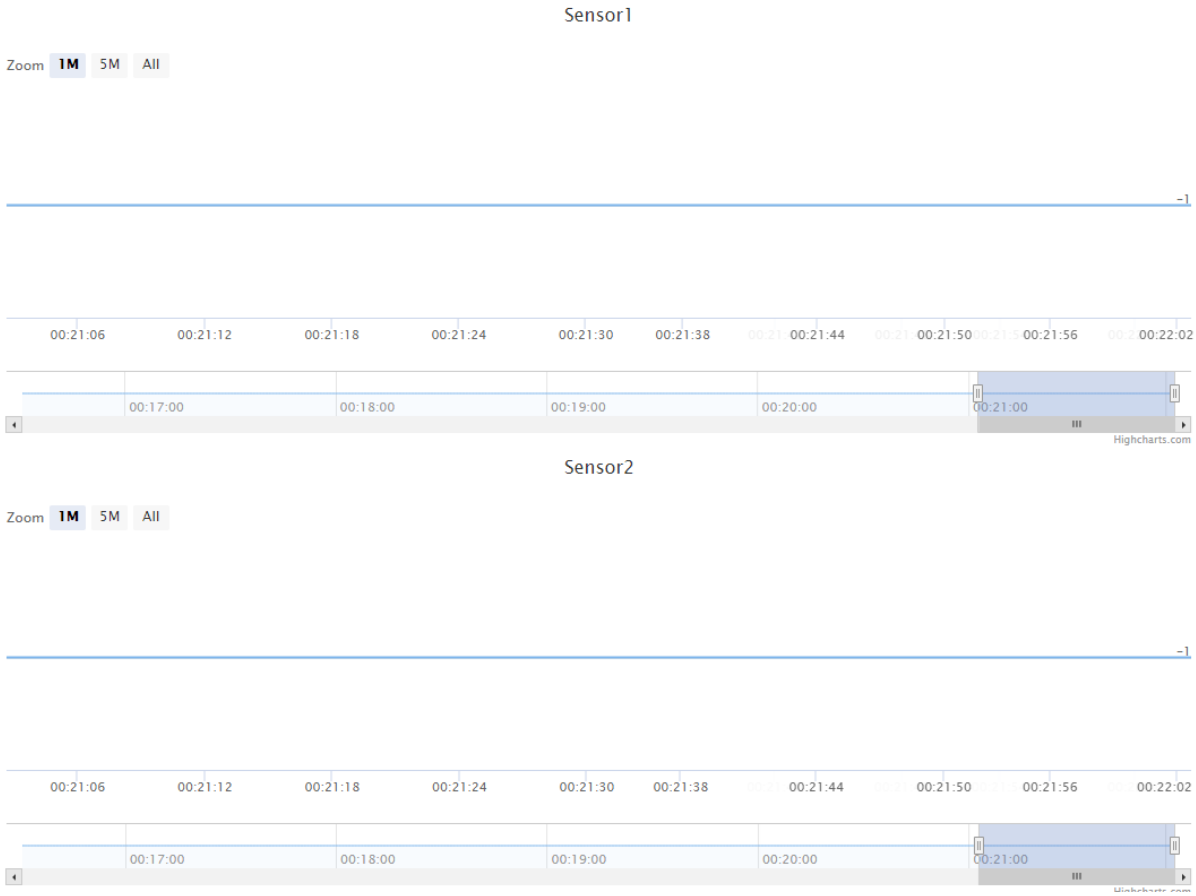


Figura 18: Interfaz Web
Fuente: Elaboración propia

Finalmente en la parte más baja de la interface se despliega una gráfica en tiempo real de la lectura de los sensores 1 y 2, con la ayuda de un complemento de JavaScript proporcionado por Highcharts [6] nos ahorra bastante complejidad a la hora de desarrollar esta clase de interfaces. Finalmente para una mejor comprensión del funcionamiento de la interfaz Web, se despliega en el **Anexo 5** el código HTML acompañado de JavaScript de dicha interfaz. Para poder acceder desde cualquier parte a la interfaz principal, se ha utilizado un servicio de host gratuito conocido como: 000webhost. La cual nos sirve para alojar la página web prototipo. La página Web se puede visitar en [7].

8.8. DESCRIPCIÓN GENERAL FINAL DE FUNCIONAMIENTO

Se describe el funcionamiento del prototipo de manera más explícita, para lo cual haremos uso del diagrama planteado en la **Figura 19:**

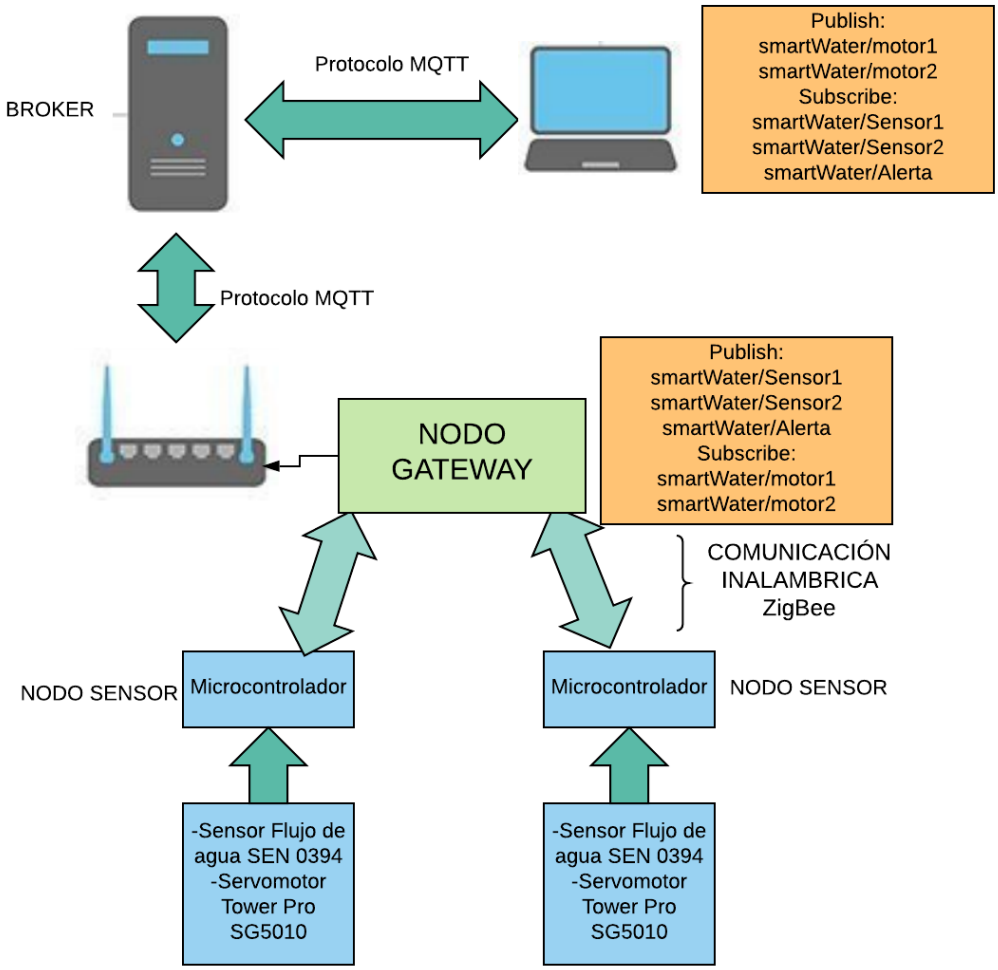


Figura 18: Descripción general de funcionamiento
Fuente: Elaboración propia

Se va a repasar lo ya mencionado en anteriores puntos con la adición de que en este punto se aclaran los distintos elementos de comunicación a través de MQTT que han sido adoptados por el sistema.

El centro del sistema y el nodo más importante es el NODO GATEWAY, ya que este es el encargado de recopilar la información de los sensores y publicarlos, asi como tambien de suscribirse a los tópicos de los motores y direccionar el accionamiento hacia el nodo de destino. En primer lugar, para recopilar la información de los sensores, el funcionamiento general del sistema es el siguiente: El NODO GATEWAY envía a través del puerto serial y este a su vez por medio del módulo XBee una cadena de la forma:

$$\boldsymbol{a, b} *$$

donde:

a es el identificador local de cada nodo

b es apuntador de la acción que deberá realizar cada nodo, pudiendo tomar los siguientes valores: 0 para desactivar la válvula de paso; 1 para activar la válvula de paso y 2 para obtener la lectura actual del sensor del nodo seleccionado. Como se observa en la **Figura 18** el NODO GATEWAY se suscribe a los tópicos smartWater/motor1 y smartWater/motor2 como se ve en el diagrama de flujo de la **Figura 16,** Cuando se recibe un mensaje de uno de estos tópicos, se compara el payload (0 o 1) de los mismos para decidir qué acción realizará el NODO GATEWAY, si por ejemplo, se recibe un mensaje de payload igual a 0 en el tópico smartWater/motor2 el NODO GATEWAY realiza la acción de cerrar la válvula (motor) del NODO SENSOR 2 esto lo hace escribiendo en el puerto serial que será transmitido por Zigbee la siguiente cadena:

$$\mathbf{2,0}*$$

Cuando el NODO GATEWAY envía esta cadena, los dos nodos sensor la reciben, sin embargo dado que el primer dato es el identificador, cada NODO SENSOR compara este primer número con su identificador local, y en caso de coincidir es cuando realiza la acción dada por la segunda cifra. Todo esto ocurre cuando desde la interfaz web se ha presionado sobre uno de los botones accionadores de los motores, ya que la interfaz Web se encarga de que al presionar los botones, los mismos publiquen sobre los tópicos smartWater/motor1 y smartWater/motor2 los valores de 1 o 0.

En cuanto a la lectura de los sensores. De igual forma quien lleva todo el sincronismo para la captura de datos es el NODO GATEWAY, que, cada 14 segundos, envía la petición de lectura de los sensores a ambos nodos, de igual manera con la escritura de las siguientes cadenas a través de su puerto serial:

$$\mathbf{1,2}*$$

$$\mathbf{2,2}*$$

Una vez recibidos los valores en los NODOS SENSOR, los mismos realizan la misma operación que en el anterior caso, en primer lugar verifican el identificador de nodo y realizan la tarea de envío de la información del sensor si es que les corresponde. Una vez recibida dicha información en el NODO GATEWAY este se encarga de publicar estos valores en los tópicos smartWater/Sensor1 y smartWater/Sensor2, los cuales pueden ser visualizados en la interfaz web **Figura 17,** que cuenta con gráficos históricos donde se ha hecho uso de la información para usar los payloads de dichos tópicos y representarlos gráficamente.

9. CONCLUSIONES Y RECOMENDACIONES

Finalizando el presente proyecto, se han cumplido con los objetivos específicos propuestos, y se pueden dar las siguientes conclusiones:

- El uso del módulo XBee ha sido elegido para que el sistema sea del menor consumo de electricidad posible, ya que se plantea que los NODOS SENSORES puedan ser localizados en cualquier parte de una vivienda y no requieran de una fuente de tensión cableada. Sin embargo no se ha enfatizado mucho más allá en este tema con estos módulos, punto que puede ser mejorado más aún, ya que dichos módulos pueden ser configurados en modos Sleep o ultra ahorro de energía, pero que requieren configuraciones especiales dentro del mismo módulo, como en las órdenes enviadas por el microcontrolador.
- Para optimizar más aún el tema del consumo de corriente eléctrica, se puede plantear un circuito adicional para el accionamiento de los servomotores, de manera que

cuando los mismos lleguen a la posición deseada por el usuario, los mismos se desconecten de la alimentación a través de un pequeño circuito de potencia con transistores

- La construcción del prototipo a una escala mediana es posible, debido a que en el mercado local se cuentan con todos los dispositivos electrónicos al alcance, un diseño superior abarcaría incluso el diseño de PCB´s integrando todos los chips SMD en una sola placa.
- Dado que el sensor con el que se cuenta es más enfocado al sector educativo, el mismo no posee una alta precisión a la hora de realizar mediciones, sin embargo habría que analizar cuál sería el fin principal del sistema para considerar el cambio del sensor a uno de uso más industrial, esta precisión debe ser considerada a más detalle si es que se requiere la detección de fugas en las redes de una precisión más considerable.
- El protocolo MQTT es ampliamente utilizado, y el desarrollo de librerías para diferentes plataformas de programación reduce enormemente el tiempo de desarrollo de prototipo de la índole presentada. Además de las grandes ventajas que se a observado en la práctica, siendo la principal la baja latencia de la misma.
- El desarrollo Web propuesto en el proyecto es demasiado básico, para un uso más útil del proyecto y el prototipo sería interesante el planteamiento de una implementación de una base de datos, de la lectura de los sensores.

10. BIBLIOGRAFÍA

[1] "Desperdicio de agua en las ciudades", 2016. [En línea]. Disponible en: https://www.sostenibilidad.com/agua/desperdicio-de-agua-en-las-ciudades/ [Accedido: 23-mar-2018]

[2] A. Creus Sole, *Instrumentacion Industrial Séptima Edición.* España: AlfaOmega-Marcombo, 2005 [En línea]. Disponible en: https://books.google.com.co/books?id=cV6ZOqQ0ywMC&pg=PA156&lpg=PA156&dq=Los+medidores+de+turbina#v=onepage&q=Los%20medidores%20de%20turbina&f=false [Accedido: 28-mar-2018]

[3] Centro de documentos de EPM de Colombia, "Criterios para definir el diámetro de la acometida y el medidor a instalar en urbanizaciones y edificios", *EPM,* Ver. 3, 2011 [En línea]. Disponible en: https://www.epm.com.co/site/Portals/0/centro_de_documentos/clientes_y_usuarios/personas/aguas/vinculacion/Criterios%20para%20definir%20el%20diametro%20de%20acometida%20y%20medidor.pdf [Accedido: 6-abr-2018]

[4] "A Comparative Study of Wireless Protocols: Bluetooth, UWB, ZigBee, and Wi-Fi", Information & Communications Research Labs, Industrial Technology Research Institute (ITRI)

[5] "paho-MQTT eclipse project", 2018. [En línea]. Disponible en: https://www.eclipse.org/paho/ [Accedido: 12-may-2018]

[6] "Highcharts: interactive charts for JavaScript", 2018 [En línea]. Disponible en: https://www.highcharts.com/ [Accedido: 18-may-2018]

[7] D. Rojas "Monitor IoT Davy", 2018 [En línea]. Disponible en: http://iotprojectin20minsdry.000webhostapp.com/ [Accedido: 1-jun-2018]

[8] G. Guacaneme y D. Pardo, "Diseño e implementación de un sistema de medición de consumo de energía eléctrica y agua potable remoto con interacción al usuario basado en el concepto de internet de las cosas ", Universidad distrital Francisco José de Caldas, 2016. [En línea]. Disponible en: http://repository.udistrital.edu.co/bitstream/11349/4315/1/GuacanemeValbuenaGerardo2016.pdf [Accedido: 7-abr-2018]

ANEXOS

Anexo 1: Configuraciones Xbee

CONFIGURACION COORDINATOR

Networking

Change networking settings

ID PAN ID	1234		
SC Scan Channels	40C1	Bitfield	
SD Scan Duration	3	exponent	
ZS ZigBee Stack Profile	0		
NJ Node Join Time	FF	x 1 sec	
NW Network Watchdog Timeout	0	x 1 minute	
JV Channel Verification	Enabled [1]		
JN Join Notification	Enabled [1]		
OP Operating PAN ID	1234		
OI Operating 16-bit PAN ID	4B55		
CH Operating Channel	19		
NC Number of Remaining Children	14		
CE Coordinator Enable	Enabled [1]		
DO Device Options	0	Bitfield	
DC Device Controls	0	Bitfield	

Addressing

Change addressing settings

SH Serial Number High	13A200	
SL Serial Number Low	4163151D	
MY 16-bit Network Address	0	
MP 16-bit Parent Address	FFFE	
DH Destination Address High	0	
DL Destination Address Low	FFFF	
NI Node Identifier	COORDINATOR	
NH Maximum Hops	1E	
BH Broadcast Radius	0	
AR Many-to-One Route Broadcast Time	FF	x 10 sec
DD Device Type Identifier	A0000	
NT Node Discovery Backoff	3C	x 100 ms
NO Node Discovery Options	0	
NP Maximum Number of Transmission Bytes	54	
CR PAN Conflict Threshold	3	

Anexo 2: Programa Arduino NODO SENSOR

```
//prueba final configuraciones arduino Micro
#include <SoftwareSerial.h>
#include <Servo.h>
SoftwareSerial XBee(10,11);
Servo myservo;
int pos = 0;
int servoActivator=7;
String readString; //main captured String
String idlocal="2";
String id; //data String
String action;

volatile int NumPulsos; //variable para la cantidad de pulsos recibidos
int PinSensor = 2;    //Sensor conectado en el pin 2
float factor_conversion=7.11; //para convertir de frecuencia a caudal
float volumen=0;
long dt=0; //variación de tiempo por cada bucle
long t0=0; //millis() del bucle anterior

int ind1; // , locations
int ind2;
void ContarPulsos ()
{
  NumPulsos++;  //incrementamos la variable de pulsos
}
int ObtenerFrecuecia()
{
  int frecuencia;
  NumPulsos = 0;   //Ponemos a 0 el número de pulsos
  interrupts();    //Habilitamos las interrupciones
  delay(1000);   //muestra de 1 segundo
  noInterrupts(); //Deshabilitamos  las interrupciones
  frecuencia=NumPulsos; //Hz(pulsos por segundo)
  return frecuencia;
}

void setup() {
  Serial.begin(9600);
  XBee.begin(9600);
  myservo.attach(6); //senal servo pin 6
  pinMode(PinSensor, INPUT);
  pinMode(servoActivator, OUTPUT);//Activador servo
  attachInterrupt(1,ContarPulsos,RISING);///depende del arduino
  t0=millis();
}

void loop() {
```

```
  //expect a string like 1,2*
  float frecuencia=ObtenerFrecuecia(); //obtenemos la frecuencia de los pulsos en Hz
  float caudal_L_m=frecuencia/factor_conversion; //calculamos el caudal en L/m
  dt=millis()-t0; //calculamos la variación de tiempo
  t0=millis();
  volumen=volumen+(caudal_L_m/60)*(dt/1000);

  if (XBee.available())  {
    char c = XBee.read();  //gets one byte from serial buffer
    if (c == '*') {
      //do stuff

      Serial.println();
      Serial.print("captured String is : ");
      Serial.println(readString); //prints string to serial port out

      ind1 = readString.indexOf(',');  //finds location of first ,
      id = readString.substring(0, ind1);   //captures first data String
      //ind2 = readString.indexOf(',', ind1+1 );   //finds location of second ,
      //action = readString.substring(ind1+1, ind2);   //captures second data String
      action = readString.substring(ind1+1);
      Serial.print("id = ");
      Serial.println(id);
      Serial.print("action = ");
      Serial.println(action);
      Serial.println();
      accion();
      clearValues();
    }
    else {
      readString += c; //makes the string readString
    }
  }


}

void accion(){

if(id==idlocal && action=="0")
{openWater();
}


if(id==idlocal && action=="1")
{closeWater();
}
```

```
if(id==idlocal && action=="2")
{sensorWater();
}

  }

void openWater(){

  digitalWrite(servoActivator, HIGH);
  delay(50);
   for (pos = 0; pos <= 90; pos += 1) {
    // in steps of 1 degree
    myservo.write(pos);
    delay(10);
  }
  delay(50);
  digitalWrite(servoActivator, LOW);

  }


void closeWater(){

   digitalWrite(servoActivator, HIGH);
  delay(50);
   for (pos = 90; pos >= 0; pos -= 1) {
    // in steps of 1 degree
    myservo.write(pos);
    delay(10);
  }
  delay(50);
  digitalWrite(servoActivator, LOW);

  }

void sensorWater(){
  XBee.println(volumen);
  //XBee.println (" L,"+idlocal);
  volumen=0;
  ;}

void clearValues(){
  readString=""; //clears variable for new input
      id="";
      action="";
  }
```

Anexo 3: Programa Python NODO GATEWAY

```
import serial
import time
import paho.mqtt.publish as publish
import paho.mqtt.client as mqtt

XBee = serial.Serial('/dev/ttyUSB0',9600)
# The callback for when the client receives a CONNACK response from the server.
def on_connect(client, userdata, flags, rc):
    print("Connected with result code "+str(rc))
    # Subscribing in on_connect() - if we lose the connection and
    # reconnect then subscriptions will be renewed.
    client.subscribe("smartWater/motor1")
    client.subscribe("smartWater/motor2")

# funciones de los motores

def read_sensor(motor):
    identidad=str(motor)
    XBee.write(identidad+",2*")
    return


def activar_motor(motor):
    identidad=str(motor)
    XBee.write(identidad+",1*")
    return

def desactivar_motor(motor):
    identidad=str(motor)
    XBee.write(identidad+",0*")
    return




# The callback for when a PUBLISH message is received from the server.
def on_message(client, userdata, msg):
    print(msg.topic+" "+str(msg.payload))

    if msg.topic == "smartWater/motor1" and msg.payload == "1":
        print("Activar motor 1")
        activar_motor(1)

    if msg.topic == "smartWater/motor1" and msg.payload == "0":
        print("Desactivar motor 1")
        desactivar_motor(1)

    if msg.topic == "smartWater/motor2" and msg.payload == "1":
```

```
        print("Activar motor 2")
        activar_motor(2)

    if msg.topic == "smartWater/motor2" and msg.payload == "0":
        print("Desactivar motor 2")
        desactivar_motor(2)

# Create an MQTT client and attach our routines to it.
client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message

client.connect("broker.hivemq.com", 1883, 60)
client.loop_start()
while True:
        #client.publish("CoreElectronics/test", "Hello")
    #publish.single("CoreElectronics/topic", "World!", hostname="broker.hivemq.com")

        time.sleep(4)
        read_sensor(1)
        print("se envio peticion 1")
        lectura1=XBee.readline()
        print("Sensor1 "+lectura1)
        client.publish("smartWater/Sensor1", lectura1)

        read_sensor(2)
        print("se envio peticion2")
        lectura2=XBee.readline()
        print("Sensor2 "+lectura2)
        client.publish("smartWater/Sensor2", lectura2)


        diferencia=float(lectura1)-float(lectura2)

        if diferencia>0.4:
                print("Alerta de fuga")
                client.publish("smartWater/Alerta", "Alerta de fuga")
        else:
                print("Todo en orden")
                client.publish("smartWater/Alerta", "todo en orden")
        time.sleep(10)

client.loop_stop()

#client.loop_forever()
```

Anexo 4: HTML desarrollo interfaz

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Monitor IoT Davy</title>
    <!-- <script src='mqttws31.js' type='text/javascript'></script> -->
    <script src="https://cdnjs.cloudflare.com/ajax/libs/paho-mqtt/1.0.1/mqttws31.js"
type="text/javascript"></script>
    <script src="https://code.highcharts.com/stock/highstock.js"></script>
    <script src="https://code.highcharts.com/stock/modules/exporting.js"></script>
    <script src="https://code.highcharts.com/stock/modules/export-data.js"></script>

    <!-- https://api.cloudmqtt.com/sso/js/mqttws31.js -->
  </head>
  <body>
      <div align="center">
        <h2>Sensores:</h2>
      </div>
      <div align="center">
        <a>Sensor 1: </a>
        <a id ="Sensor1">-</a>
      </div>
      <div align="center">
        <a>Sensor 2: </a>
        <a id ="Sensor2">-</a>
      </div>
      <div align="center">
        <a>Alerta: </a>
        <a id ="Alerta">-</a>
      </div>
      <div align="center">
        <h2>Motores:</h2>
      </div>
      <div align="center">
        <a>Motor 1: </a>
        <button type='button' onclick='Encendermotor("1")'>ON</button>
        <button type='button' onclick='Apagarmotor("1")'>OFF</button>
      </div>
      <div align="center">
        <a>Motor 2: </a>
        <button type='button' onclick='Encendermotor("2")'>ON</button>
        <button type='button' onclick='Apagarmotor("2")'>OFF</button>
      </div>
      <div>
       <!--<a>Salida Analógica: </a>
        <input type="range" id="myRange" min="0" max="100"
onmouseup="enviarSalidaAnalogica()">
```

```
  </div>-->
  <div id="container" style="height: 400px; min-width: 310px"></div>
  <div id="container2" style="height: 400px; min-width: 310px"></div>
<script>
  usuario = 'placa1';
  contrasena = '12345678';
  consumo1=-1;
  consumo2=-1;

  function Encendermotor(motor){
    message = new Paho.MQTT.Message("1");
    message.destinationName = 'smartWater/motor'+motor;
    client.send(message);
  }
   function Apagarmotor(motor){
    message = new Paho.MQTT.Message("0");
    message.destinationName = 'smartWater/motor'+motor;
    client.send(message);

  }
  function OnOff(dato){
    message = new Paho.MQTT.Message(dato);
    message.destinationName = '/' + usuario + '/salidaDigital'
    client.send(message);
  };

  function enviarSalidaAnalogica(){
    var dato = document.getElementById("myRange").value;
    message = new Paho.MQTT.Message(dato);
   // message.destinationName = '/' + usuario + '/salidaAnalogica'
    message.destinationName = 'smartWater/Sensor2'////////////////////////////////////////////////
    client.send(message);
  };

  // called when the client connects
  function onConnect() {
    // Once a connection has been made, make a subscription and send a message.
    console.log("onConnect");
    client.subscribe("smartWater/Sensor1");
    client.subscribe("smartWater/Sensor2");
    client.subscribe("smartWater/Alerta");
  }

  // called when the client loses its connection
  function onConnectionLost(responseObject) {
    if (responseObject.errorCode !== 0) {
      console.log("onConnectionLost:", responseObject.errorMessage);
      setTimeout(function() { client.connect() }, 5000);
```

```
  }
 }

 // called when a message arrives
 function onMessageArrived(message) {
  if (message.destinationName == 'smartWater/Sensor1') { //acá coloco el topic
     document.getElementById("Sensor1").textContent = message.payloadString  + " L";
     consumo1=parseFloat(message.payloadString);
     showTime();

  }
  if (message.destinationName == 'smartWater/Sensor2') { //acá coloco el topic
     document.getElementById("Sensor2").textContent = message.payloadString + " L";
     consumo2=parseFloat(message.payloadString);
  }
  if (message.destinationName == 'smartWater/Alerta') { //acá coloco el topic
     document.getElementById("Alerta").textContent = message.payloadString;
  }

 }
  function showTime() {
   var timeNow = new Date();
   var hours   = timeNow.getHours();
   var minutes = timeNow.getMinutes();
    var seconds = timeNow.getSeconds();
   var timeString = "" + ((hours > 12) ? hours - 12 : hours);
   timeString  += ((minutes < 10) ? ":0" : ":") + minutes;
    timeString  += ((seconds < 10) ? ":0" : ":") + seconds;
   timeString  += (hours >= 12) ? " P.M." : " A.M.";
   //document.getElementById("Alerta").textContent =timeString;

}

  function onFailure(invocationContext, errorCode, errorMessage) {
    var errDiv = document.getElementById("error");
    errDiv.textContent = "Could not connect to WebSocket server, most likely you're
behind a firewall that doesn't allow outgoing connections to port 39627";
    errDiv.style.display = "block";
  }

  var clientId = "ws" + Math.random();
  // Create a client instance
 // var client = new Paho.MQTT.Client("broker.hivemq.com", 8000, clientId);
  var client = new Paho.MQTT.Client("broker.hivemq.com", 8000,clientId);
  // set callback handlers
  client.onConnectionLost = onConnectionLost;
  client.onMessageArrived = onMessageArrived;
```

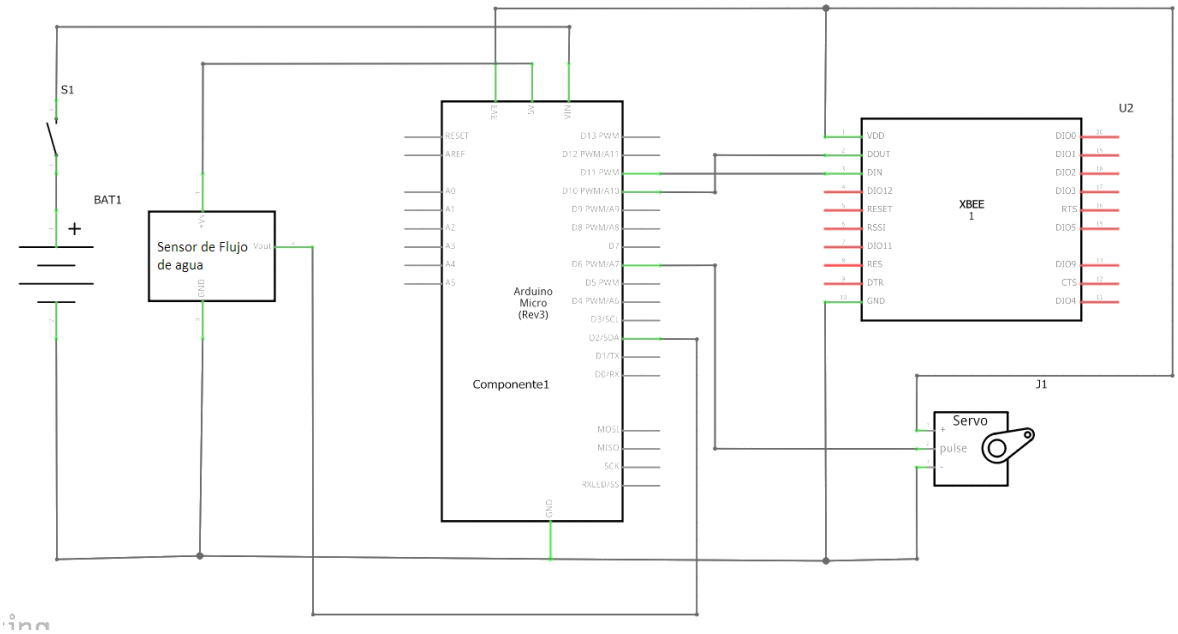
```
        // connect the client
        client.connect({
          useSSL: false,
          userName: usuario,
          password: contrasena,
          onSuccess: onConnect,
          onFailure: onFailure
        });


        Highcharts.setOptions({
    global: {
        useUTC: false
    }
});

// Create the chart
Highcharts.stockChart('container2', {
    chart: {
        events: {
            load: function () {

                // set up the updating of the chart each second
                var series = this.series[0];
                setInterval(function () {
                    var x = (new Date()).getTime(), // current time
                        y = consumo2;
                    series.addPoint([x, y]);
                }, 1000);
            }
        }
    },

    rangeSelector: {
        buttons: [{
            count: 1,
            type: 'minute',
            text: '1M'
        }, {
            count: 5,
            type: 'minute',
            text: '5M'
        }, {
            type: 'all',
            text: 'All'
        }],
        inputEnabled: false,
        selected: 0
```

```
    },

    title: {
        text: 'Sensor2'
    },

    exporting: {
        enabled: false
    },

    series: [{
        name: 'Consumo',
        data: (function () {
            // generate an array of random data
            var data = [],
                time = (new Date()).getTime(),
                i;

            for (i = -300; i <= 0; i += 1) {
                data.push([
                    time + i * 1000,
                    -1
                ]);
            }
            return data;
        }())
    }]
});



// Create the chart
Highcharts.stockChart('container', {
    chart: {
        events: {
            load: function () {

                // set up the updating of the chart each second
                var series = this.series[0];
                setInterval(function () {
                    var x = (new Date()).getTime(), // current time
                        y = consumo1;
                    series.addPoint([x, y]);
                }, 1000);
            }
        }
    },
```

```
    rangeSelector: {
        buttons: [{
            count: 1,
            type: 'minute',
            text: '1M'
        }, {
            count: 5,
            type: 'minute',
            text: '5M'
        }, {
            type: 'all',
            text: 'All'
        }],
        inputEnabled: false,
        selected: 0
    },

    title: {
        text: 'Sensor1'
    },

    exporting: {
        enabled: false
    },

    series: [{
        name: 'Consumo',
        data: (function () {
            // generate an array of random data
            var data = [],
                time = (new Date()).getTime(),
                i;

            for (i = -300; i <= 0; i += 1) {
                data.push([
                    time + i * 1000,
                    -1
                ]);
            }
            return data;
        }())
    }]
});

    </script>
  </body>
</html>
```

Anexo 5: Diagrama electrónico

Nodo Sensor:



Nodo Gateway: