Licenciatura en Sistemas

Desarrollo de estación de monitoreo de un dispositivo híbrido de termotanques solar y a gas natural instalado en la Sede Atlántica de la Universidad Nacional de Río Negro

Alumno: Valsecchi, Javier Jesús

Director: Martinez Luquez, Juan Cruz

Co-director: Malpeli, Guillermo

Índice

Introducción

En la actualidad vivimos en una sociedad altamente dependiente de los combustibles fósiles[1]. Esta dependencia genera una gran contaminación que afecta tanto la salud general como el equilibrio del ecosistema del planeta.

Uno de los combustibles fósiles que más se consume en la Argentina es el gas natural, utilizado por las industrias y por los usuarios domésticos. Según datos aportados por el estudio BP Statistical Review of World Energy 2017, publicado por la consultora Invenomica, y otros del Ministerio de Energía de la Nación, ascendió al 53% dentro de nuestra matriz energética, que resulta la más alta en la región.

Teniendo en cuenta los futuros desafíos ecológicos y económicos, es necesario plantear alternativas viables. Distintas organizaciones y gobiernos apuntan a la implementación y utilización de recursos renovables, de baja contaminación y fácil acceso. Con el objetivo de conseguir una mejor gestión de los recursos y un menor impacto en el medio ambiente, las opciones renovables deberían implementarse en las ciudades con grandes aglomeraciones de personas y usuarios finales para que se mejore la eficiencia energética.

Una tecnología viable para la reducción del consumo de gas natural en los hogares son los termotanques solares. Estos permiten utilizar la radiación solar en reemplazo de gas para llegar a la temperatura requerida; o bien dotar de agua caliente a hogares que no cuenten con el acceso a gas natural.

A partir del surgimiento del concepto de Internet de las Cosas (IoT), es posible, mediante la utilización de dispositivos electrónicos como microcontroladores, obtener en tiempo real datos sobre variables específicas de un dispositivo físico, como puede ser la temperatura de entrada y salida de una red de agua, el caudal de agua que se utiliza, etc. Estos datos pueden ser mostrados en tiempo real o ser almacenados en una base de datos para futuros análisis. Es importante contar con datos sobre el funcionamiento para obtener estadísticas, con el fin de ayudar al ciudadano en la toma de decisiones y en la planificación a largo plazo. En este sentido, el objetivo de este trabajo es, además, colaborar con la promoción y el otorgamiento de subsidios para las tecnologías asociadas al Internet de las Cosas. Una mayor divulgación de los servicios vinculados con IoT conduciría a un desarrollo activo de estas soluciones, que ya están aportando ahorros de recursos no renovables, mejorando la calidad de vida de los ciudadanos o incrementando la seguridad de la población.

Teniendo en cuenta todo lo planteado anteriormente, surge la propuesta de diseñar e implementar un sistema que permita monitorear el desempeño de una instalación híbrida de termotanques solar y a gas natural. Esto nos permitirá obtener datos de

distintas variables de importancia, como la temperatura en distintos puntos del sistema, el caudal de agua utilizado, el tiempo de funcionamiento del termotanque a gas, y mostrarlos de manera histórica. Se busca desarrollar el software requerido y realizar el despliegue del hardware necesario sobre el termotanque solar instalado en la Universidad de Río Negro, en la Sede Atlántica, en el marco del proyecto “Uso de Energías Renovables en la Universidad Nacional de Río Negro”. Se trata, además, de que los datos generados por el proyecto puedan ser de acceso público a los ciudadanos de la región, entidades privadas y al Estado, para obtener información del rendimiento de un termotanque solar en la región con el objetivo adicional de difundir la adopción y desarrollo de tecnologías basadas en IoT.

Motivación

En los años recientes hay un aumento en el interés por utilizar energías renovables, tanto solar como eólica, junto con una innovación en el desarrollo e implementación de tecnologías que permiten aprovechar estos recursos estratégicos.
El calentamiento del agua tiene un rol importante dentro de las actividades cotidianas de las personas. Además de permitir cocinar distintos alimentos, tiene una importancia fundamental en la higiene, lo que ayuda en la prevención de múltiples enfermedades y aporta confort en la vida de las personas. Lamentablemente en varios países, Argentina entre ellos, no toda la población tiene acceso al gas natural o la electricidad para calentar el agua[2], lo que la deja únicamente con la posibilidad de utilizar leña o carbón para hacerlo, algo costoso y de difícil acceso. Por estas razones, utilizar termotanque solares ayuda tanto a las personas que viven en zonas no favorecidas, con carencia de recursos, permitiéndoles el acceso a un recurso básico e indispensable, así como también colabora con la posibilidad de reducir la dependencia de los combustibles fósiles.

En Argentina, de forma paulatina, se está produciendo un aumento en la demanda en el mercado de termotanque solares[3][4]. En su mayoría las instalaciones son realizadas de forma casera utilizando materiales de fácil acceso[5][6]. La mayor parte de la población desconoce esta tecnología por la poca información que se le proporciona y por la falta de una adecuada comunicación, por eso se desconocen los posibles beneficios tanto individuales como sociales que pueden aportar.

Existen muchos estudios y publicaciones que aportan información[7][8][9] sobre los beneficios de utilizar termotanques solares en las viviendas. Sin embargo, los kits de productos que se pueden encontrar en el mercado proporcionan solo el termotanque solar, informan sobre su efectividad y posibles ahorros que genera, pero no agregan ninguna facilidad a los clientes para que puedan monitorear constantemente el rendimiento o valor que aporta el producto en funcionamiento y a lo largo del tiempo. Así como cuando el usuario tiene acceso en tiempo real al consumo de energía este

regula y modifica sus hábitos para mejorar la utilización de ese recurso[10], tener acceso en tiempo real a los beneficios aportados por un termotanque solar permitirá una mayor comprensión de esta tecnología y con ello que cada vez más personas la utilicen y obtengan sus beneficios.

Problemática en cuestión

En el marco del desarrollo del proyecto de extensión “Uso de Energías Renovables en la Universidad Nacional de Río Negro”[11] se instaló un termotanque solar en el comedor del campus de la UNRN en la sede Atlántica (Viedma). Esta instalación tuvo entre sus objetivos obtener datos de su rendimiento, permitir el acceso de forma pública a estos datos, almacenarlos para su posterior análisis y verificar si en un futuro es factible instalar múltiples termotanques solares para reducir el consumo de gas natural del establecimiento.

Partiendo de los objetivos planteados, se propuso el desarrollo de un sistema que permitiera la recolección de datos de los parámetros físicos, su visualización y su almacenamiento.

Solución propuesta

En el trabajo publicado “Desarrollo de sistema de monitoreo para termotanque solar ubicado en el comedor de la Universidad Nacional de Río Negro sede Atlántica”[12] se analizó la problemática planteada y se describió la implementación de un prototipo que cumplía con las funcionalidades requeridas.
El presente trabajo busca expandir la propuesta ya realizada, ampliando en mayor detalle la metodología usada para planificar y estimar el trabajo realizado, así como la arquitectura desarrollada y las diferentes tecnologías involucradas para el desarrollo del prototipo.

A partir del análisis de los requisitos funcionales presentados en el trabajo publicado [12], se propuso que el prototipo monitoree las siguientes variables:

- **Caudal**: permite obtener la cantidad de agua utilizada en el sistema. Cuanta más agua se utiliza en un instante menor será la temperatura en la que se encuentra. Este parámetro permite determinar si está bien dimensionado el equipamiento instalado en cuanto a las necesidades del establecimiento.
- **Temperatura del agua entrante**: permite obtener la temperatura a la que entra el agua al sistema.
- **Temperatura del ambiente**: temperatura general de ambiente.
- **Temperatura de salida del termotanque solar**: variable que determina la temperatura a la cual el termotanque solar calienta el agua.

- **Temperatura de salida del termotanque a gas**: marca la temperatura de salida del termotanque a gas para comparar con la salida del termotanque solar y analizar el trabajo que tuvo que realizar para llegar a esa temperatura.
- **Tiempo de actividad del termotanque a gas**: permite determinar cuánto tiempo estuvo encendido el termotanque a gas.

Las variables seleccionadas son fundamentales a la hora de determinar el rendimiento de la instalación del termotanque solar. Los datos recolectados de las diferentes variables de entrada obtenidas de los sensores serán enviados, de manera local con posibilidad de enviarlos a través de Internet, a un servidor de almacenamiento. Al mismo tiempo, en este servidor se dispondrá de un sistema de visualización amigable de dichos datos.

Además, el prototipo contempla la posibilidad de agregar en un futuro otras variables para recolectar más datos sobre el rendimiento y permitir la posibilidad en el futuro de analizar los datos recolectados para la consiguiente toma de decisiones.

Metodología de trabajo y planificación.

Para llevar a cabo las tareas de planificación, gestión y desarrollo de la solución, se utilizó una metodología de trabajo encuadrada en el marco ágil denominada SCRUM, teniendo en cuenta que todas las personas involucradas en el trabajo poseían conocimientos y experiencia en dicha metodología.

El objetivo es entregar piezas de software funcionales en cortos intervalos de tiempo para implementar de forma incremental los requerimientos evaluados. Para ello, se divide el desarrollo en distintos Sprints autoconcluyentes que, gracias a la metodología seleccionada, no es necesario documentar de forma extensa. Es suficiente con declarar el requerimiento en una forma breve y pasar directamente a la implementación. Al finalizar, el resultado obtenido es una pieza de software funcional que fue previamente probada y cumple con las especificaciones dictadas por el dueño del producto.

Para llevar a cabo la metodología planteada es necesario definir dos artefactos: las historias de usuario y la pila del producto.
Las historias de usuario permiten documentar los requisitos funcionales del producto. Son redactadas por el cliente en conjunto con los miembros del equipo del proyecto. Se utiliza una terminología amigable, de forma que pueda ser comprendida por cualquier persona, requisito indispensable para la validación de la implementación. Estás HU se escribirán siguiendo el siguiente modelo:

Título:

Número:	Prioridad:	Modificación de historia N°...
Descripción		

De acuerdo con este esquema se definen todos los requerimientos como HU. A cada una de ellas se le asigna una prioridad así como el esfuerzo requerido, que son las métricas necesarias para diseñar los Sprints.

En cuanto a la pila de producto, en ella se integran todas las HU ordenadas por su prioridad y estado de desarrollo. La pila es necesaria para que todo el equipo de trabajo tenga presente los requisitos pendientes y las tareas que se encuentran en desarrollo. Una de las mejores herramientas para implementar este seguimiento es un tablero Kanban que cuenta con columnas que representan el estado actual de cada HU. Esas columnas se van desplazando de izquierda a derecha conforme cambian de estado.

Planificación

Para llevar a cabo la planificación y gestión del proyecto se utilizaron herramientas gratuitas disponibles en Internet, de forma que todos los integrantes pudieran acceder a ellas con facilidad. Se decidió utilizar Trello, la cual permite crear tableros Kanban e historias de usuario. Para cada Sprint se generó un nuevo tablero y se volcaron en él las HU planificadas.

Siguiendo con las pautas de la metodología se definieron los siguientes roles dentro del proyecto:

- Dueño del producto: Martinez Luquez, Juan Cruz
- Scrum Master: Malpeli, Guillermo
- Scrum Team: Valsecchi, Javier Jesús

En conjunto se realizó el análisis del problema definiendo las siguientes historias de usuario:

Título: Ver temperatura del agua		
Número: 1	Prioridad: 40	Modificación de historia N°...
Como usuario de la aplicación, quiero acceder desde una página web a la temperatura del agua en tiempo real en los puntos: ● entrada de agua antes del termotanque solar		

- salida de agua del termotanque solar, antes de entrar al termotanque a gas
- salida de agua del termotanque a gas

Además esta información debe estar presentada en grados Celsius.

Título: Ver temperatura del ambiente		
Número: 2	Prioridad: 60	Modificación de historia N°...
Como usuario de la aplicación, quiero acceder desde una página web a la temperatura del ambiente en tiempo real Además esta información debe estar presentada en grados Celsius.		

Título: Ver caudal de agua		
Número: 3	Prioridad: 45	Modificación de historia N°...
Como usuario de la aplicación, quiero acceder desde una página web el caudal de agua que es utilizada y pasa por el sistema de termotanques. Además esta información debe estar presentada en litros por minutos (l/m).		

Título: Ver tiempo de actividad termotanque a gas		
Número: 4	Prioridad: 50	Modificación de historia N°...
Como usuario de la aplicación, quiero acceder desde una página web al estado actual del quemador de termotanque a gas. Además esta información debe estar presentada como activo/inactivo.		

Título: Ver tiempo de actividad termotanque a gas		
Número: 5	Prioridad: 55	Modificación de historia N°...
Como usuario de la aplicación, quiero acceder desde una página web al histórico de los datos obtenidos de la estación de monitoreo.		

Teniendo las HU definidas para el proyecto, es posible realizar la pila del producto, donde las historias de usuario son ordenadas según su prioridad asignada:

1 - Ver temperatura del agua
3 - Ver caudal de agua

4 - Ver tiempo de actividad termotanque a gas
5 - Ver histórico de los datos
2 - Ver temperatura del ambiente

Ya definida la pila del producto se organizaron los Sprints para el desarrollo del producto tomando en cuenta los tiempos estimados para cada historia de usuario y los tiempos de instalación además de los de desarrollo. Se asignó un tiempo de tres meses para el desarrollo e implementación. Por lo tanto las historias generadas fueron asignadas de la siguiente forma en los sprints:

Sprint 1	Inicio:	Fin:
1 - Ver temperatura del agua		

Sprint 2	Inicio:	Fin:
3 - Ver caudal de agua 4 - Ver tiempo de actividad termotanque a gas		

Sprint 3	Inicio:	Fin:
2 - Ver temperatura del ambiente 5 - Ver histórico de los datos		

Arquitectura de la solución.

La arquitectura desarrollada tiene como objetivo ser escalable, es decir, fácilmente adaptable a otros escenarios, estable y de fácil mantenimiento. Para ello se diseñó utilizando software libre y priorizando el bajo acoplamiento de las partes. El proyecto se diseñó para ejecutar todas las partes en un mismo dispositivo, pensando en una zona que no tiene acceso a Internet. En ese contexto los datos puedan ser accedidos en forma local. Sin embargo, es posible separar fácilmente la arquitectura para que funcione en un esquema tipo cloud donde el sistema envía los datos a un servidor en Internet.

La implementación de la arquitectura se puede separar en dos partes que interactúan entre sí: el software de recolección y envío de los datos de los sensores y el servidor. En este proyecto esta arquitectura está instalada en el mismo dispositivo, donde corren los distintos software para recolectar los datos, almacenarlos y mostrarlos de forma amigable al usuario. Sin embargo, existe la

posibilidad de que la instalación sea en Internet, sin necesidad de ninguna modificación en la arquitectura.

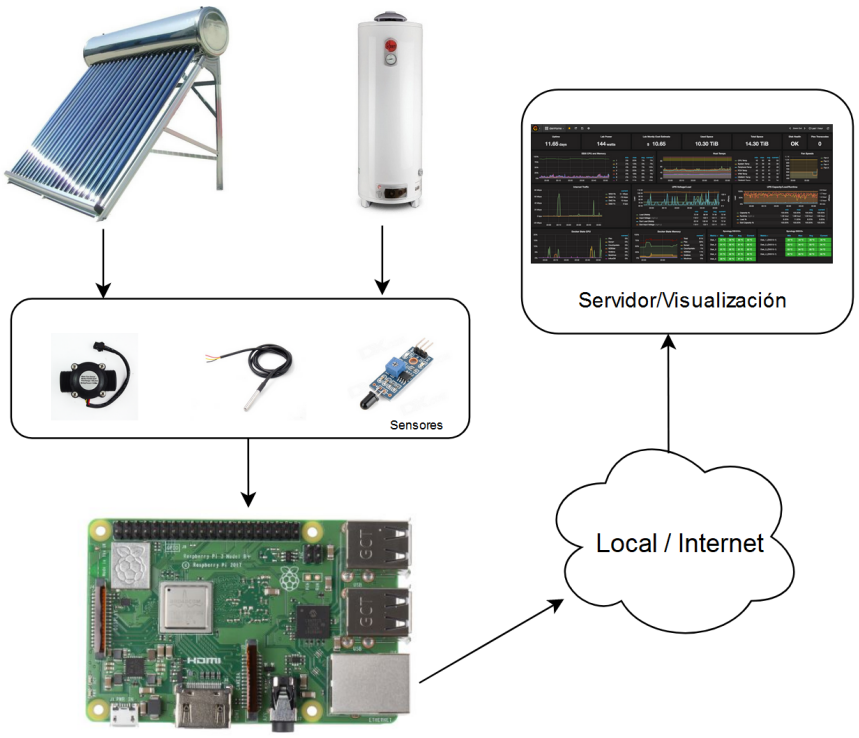


Fig 1. Esquema de componentes y conectividad. Elaboración propia.

En la figura 1 se muestra el esquema simplificado del diseño de la estación. Se utilizan los sensores de caudal de agua, de temperatura y de flama. Además cuenta con conectividad a Internet que le da la posibilidad de enviar los datos recolectados al servidor, donde luego se almacenan y se visualizan.
En la primera etapa de diseño y prototipo se decidió utilizar Telegraf como servidor de recepción de datos, Influxdb como base de datos y Grafana para la visualización de la información. Esta selección permite una rápida configuración y utilización del servicio para las pruebas así como también una arquitectura libre que no implica un costo agregado y puede ser fácilmente modificable para cumplir con requerimientos futuros.

Los datos son obtenidos mediante sensores desplegados en la instalación del termotanque. Se utilizaron sensores DS18B20 para obtener datos de temperatura del agua y ambiente; sensor Ky-026 para monitorear el encendido del termotanque a gas y el sensor FS300A para contabilizar el caudal de agua utilizado.

Las interfaces de conectividad de estos sensores están disponibles en la Raspberry Pi, con lo cual no es necesario adquirir ningún dispositivo adicional para conectarlos. A continuación se los describe brevemente:

- Sensor de caudal FS300A: es un instrumento para la medición de caudal o gasto volumétrico de un fluido. El caudal es la cantidad de líquido o fluido (volumen) que circula a través de una tubería por unidad de tiempo, por lo general se expresa en: litros por minutos (l/m). Funciona mediante el efecto Hall a medida que el agua pasa por el sensor hace girar un un mecanismo el cual emite un pulso eléctrico por cada cierta cantidad de líquido que pasa, este pulso puede ser leído por una entrada digital en el Raspberry Pi.
- Sensores de temperatura DS18B20: es un sensor digital de temperatura que utiliza el protocolo 1-Wire (protocolo tipo bus) para la comunicarse, dicho protocolo necesita un solo pin de datos y permite conectar más de un sensor en el mismo bus teniendo cada uno un ID único de identificación de 64 bits. Este sensor tiene la capacidad de medir temperatura desde los -55°C hasta los 125°C con una resolución programable desde 9 hasta 12 bits.
  La presentación comercial más utilizada por conveniencia y robustez es la del sensor dentro de un tubo de acero inoxidable resistente al agua, lo que permite estar en contacto directo con el agua sin fallar.
- Sensor de flama Ky-026: detecta longitudes de onda de 760 nm y 1100 nm. Este rango de onda corresponde a la luz infrarroja emitida por una llama. Cuenta con una salida digital para detectar si está en on/off y una salida analógica que permite medir la intensidad de la luz infrarroja.

Como se mencionó anteriormente, la Raspberry Pi permite el uso de muchos lenguajes de programación. Para este desarrollo se utilizó Python, un lenguaje de programación interpretado, multiparadigma y de tipado dinámico cuya filosofía hace hincapié en una sintaxis muy limpia y un código legible.

En la figura 2 se muestra la lógica del algoritmo desarrollado para la estación de monitoreo.

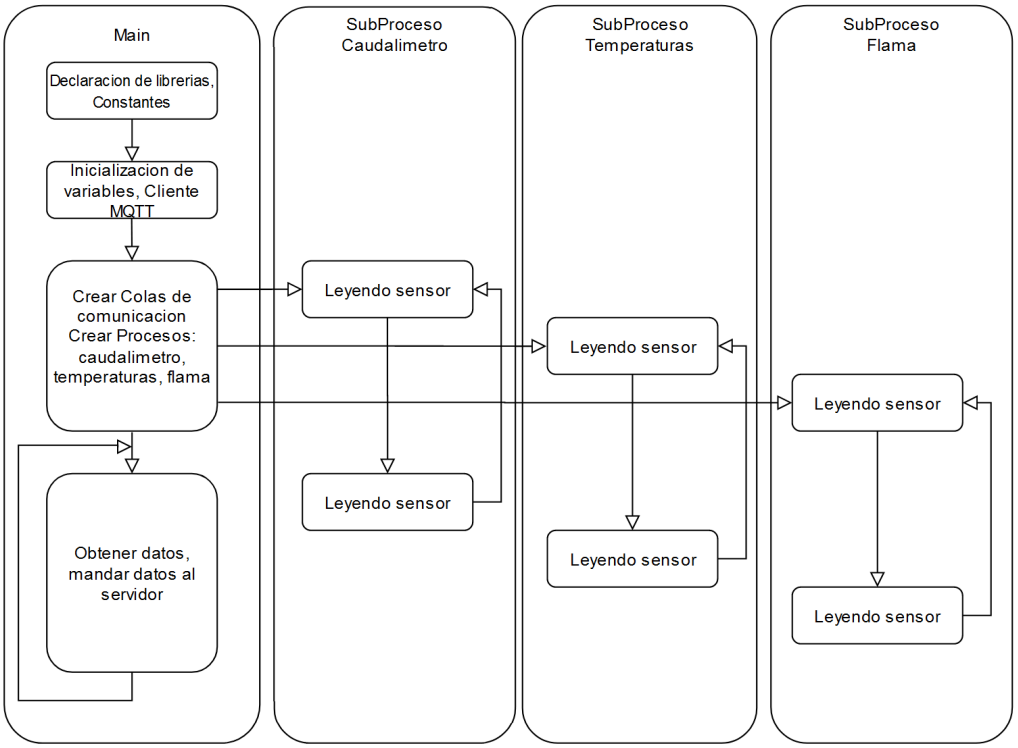


Fig. 2. Diagrama de programación. Elaboración propia.

Se decidió que para la implementación del algoritmo se utilizaría la librería “multiprocessing” que permite la creación de subprocesos, además del proceso main, para que realicen tareas independientes. En la figura 3 se especifica de manera gráfica la secuencia de mensajes entre procesos. Cada subproceso se encarga de manejar un tipo de sensores específicos, el proceso main recolecta los datos obtenidos y los envía al servidor para almacenar los datos y posteriormente permitir visualizarlos.

Implementación de la Solución.

En este capítulo se desarrollará el código de la implementación en python de los distintos subprocesos planteados para la ejecución del algoritmo. Además, se definirán las interconexiones físicas entre el Raspberry Pi y los sensores. Finalmente, se desarrollará la arquitectura de almacenamiento y visualización de los datos.

Software de recolección de datos

Como se mencionó en el capítulo anterior se decidió la implementación en forma de subprocesos como se observó en la figura 2. En la figura 3 se detalla con más profundidad el diagrama de secuencia con el detalle de la interacción entre los distintos sub procesos.

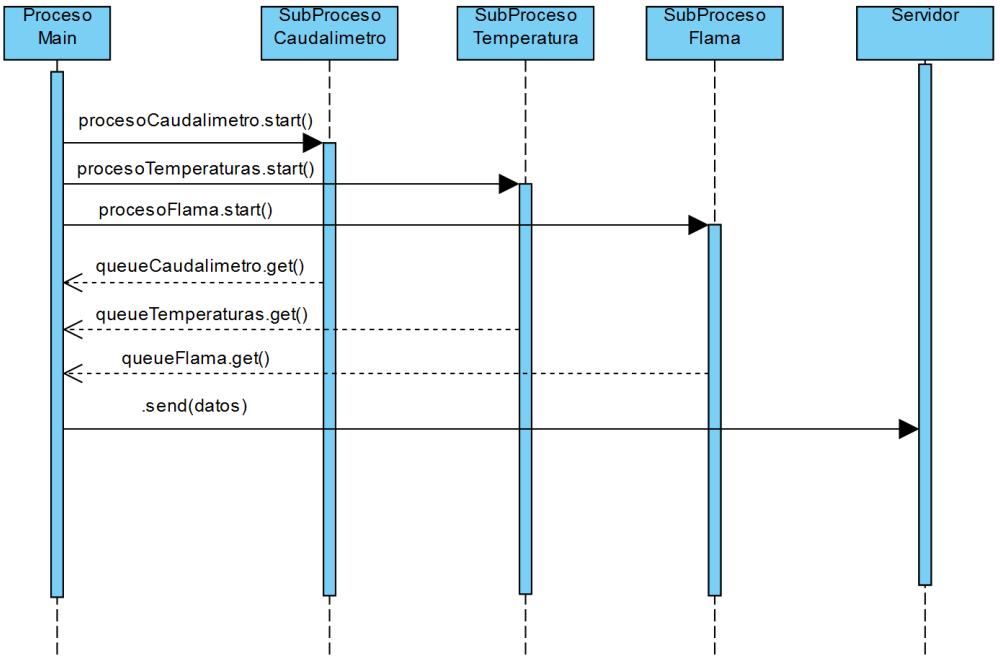


Fig. 3. Diagrama de secuencia entre los distintos procesos. Elaboración propia.

Proceso Main

El proceso mail es el encargado tanto de la creación y la comunicación entre subprocesos como de publicar los datos obtenidos de los sensores.

Para crear los subprocesos se hace uso de la librería <multiprocessing> que permite crear y gestionar los siguientes subprocesos:

- procesoCaudalimetro: proceso destinado a ejecutar el algoritmo de recolección de datos del caudalímetro.
- procesoTemperaturas: proceso destinado a ejecutar el algoritmo de recolección de datos de los sensores de temperatura.
- procesoFlama: proceso destinado a ejecutar el algoritmo de recolección de datos del sensor de flama.

Simultáneamente además crea las colas de mensajes que permiten la comunicación con los distintos subprocesos para su publicación en el servicio MQTT Broker.

Implementación

Se muestra a continuación el código correspondiente a la implementación del proceso Main.

```python
if __name__ == "__main__":
    GPIO.setup(PIN_FLAMA, GPIO.IN, pull_up_down=GPIO.PUD_UP)
    GPIO.setup(PIN_CAUDALIMETRO, GPIO.IN)

    procesos = []
    queues = []

    queueCaudalimetro = multiprocessing.Queue()
    queueTemperaturas = multiprocessing.Queue()
    queueFlama = multiprocessing.Queue()
    queues.append(queueCaudalimetro)
    queues.append(queueTemperaturas)
    queues.append(queueFlama)

    procesoCaudalimetro = multiprocessing.Process(
        target=caudalimetro, args=(queueCaudalimetro,)
    )
    procesos.append(procesoCaudalimetro)

    procesoTemperaturas = multiprocessing.Process(
        target=temperatura, args=(queueTemperaturas,)
    )
    procesos.append(procesoTemperaturas)

    procesoFlama = multiprocessing.Process(target=flama, args=(queueFlama,))
    procesos.append(procesoFlama)

    for pro in procesos:
        pro.start()

    time.sleep(1)

    #inicializar mqqt client
    mqttClient = mqtt.Client()
    mqttClient.connect('localhost', 1883)
    mqttClient.loop_start()

    try:
        while(True):
            for que in queues:
                while(not que.empty()):
                    dato = que.get()
                    mqttClient.publish(topic=dato[0], payload=dato[1])
    except:
        for pro in procesos:
            pro.terminate()()
        # limpio la interfaz GPIO
        GPIO.cleanup()
```

Subproceso caudalímetro

Para la implementación de la medición de caudal de agua, se utilizó el sensor FS300A. El sensor funciona mediante el principio de turbina, en el que el flujo de agua hace girar una turbina interna que está conectada a un imán. El imán genera un campo magnético que es detectado por un sensor de efecto Hall. El sensor de

efecto Hall emite un pulso eléctrico cada vez que el imán pasa por él. La frecuencia de los pulsos es proporcional al caudal de agua.

La salida del sensor es un pulso que se puede tomar como una salida digital, cada pulso a “1” representa 1 pulso. Para calcular cuánto caudal paso por el sensor se utiliza la siguiente fórmula:

caudal = pulsos / factor de conversión

En la implementación se fue sumando pulsos por 60 segundos y el factor de conversión el fabricante especifica que es 5.5 .

Implementación

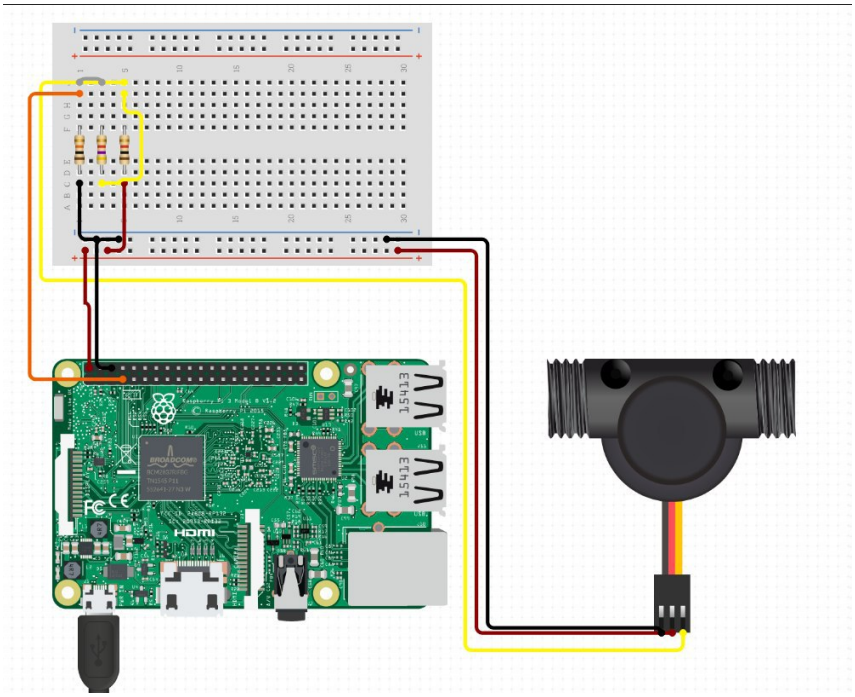
Se muestra a continuación el código correspondiente a la implementación de la función caudalimetro ejecutada por procesoCaudalimetro.

```python
# Varaibles, Constantes de uso para el sub.proceso caudalimetro
PIN_CAUDALIMETRO = 7  # Pin de Data Imput del sensor Caudalimetro
TIEMPO_CAUDALIMETRO = (
    5  # tiempo de espera para leer pulsos del caudalimetro, expresado en segundos
)
FACTOR = 13.2  # Factor de convercion de pulsos a L/min

# Funcion implementa el algoritmo para manejar el caudalimetro
def caudalimetro(queue):
    topic="termotanque/caudal"
    caudalimetro_gpio_ultimo = 2  # Ultimo valor obtenido, variable para detectar cambio en los pulsos
    time_stop = time.time() + TIEMPO_CAUDALIMETRO
    pulsos = 0
    while True:
        pulsos = 0
        while time.time() <= time_stop:
            caudalimetro_gpio_actual = GPIO.input(PIN_CAUDALIMETRO)
            if caudalimetro_gpio_actual != caudalimetro_gpio_ultimo:
                pulsos += 1
            caudalimetro_gpio_ultimo = caudalimetro_gpio_actual
        queue.put([topic,pulsos])
        time_stop = time.time() + TIEMPO_CAUDALIMETRO
```

Diagrama de conexión

En el siguiente diagrama se detalla la conexión entre la Raspberry pi y el sensor de caudal utilizado.



Elaboración propia.

Subproceso Temperatura

Para medir la temperatura del agua en los distintos puntos planteados se utilizó el sensor DS18B20. Este sensor mide la temperatura mediante un termistor. Un termistor es una resistencia que cambia su valor de resistencia en función de la temperatura. El sensor DS18B20 tiene un circuito integrado que convierte la resistencia del termistor en una señal digital. Además utiliza el protocolo 1-Wire para comunicarse. Este protocolo solo necesita un pin de datos para comunicarse y permite conectar más de un sensor en el mismo bus, simplificando así la instalación dado que se pueden conectar los 4 sensores necesarios a un único pin de datos.

Implementación

Para poder utilizar correctamente el sensor se tiene que agregar la librería OneWire para la utilización del bus. En la siguiente imagen se muestra el código correspondiente a la función temperatura ejecutada por procesoTemperaturas:

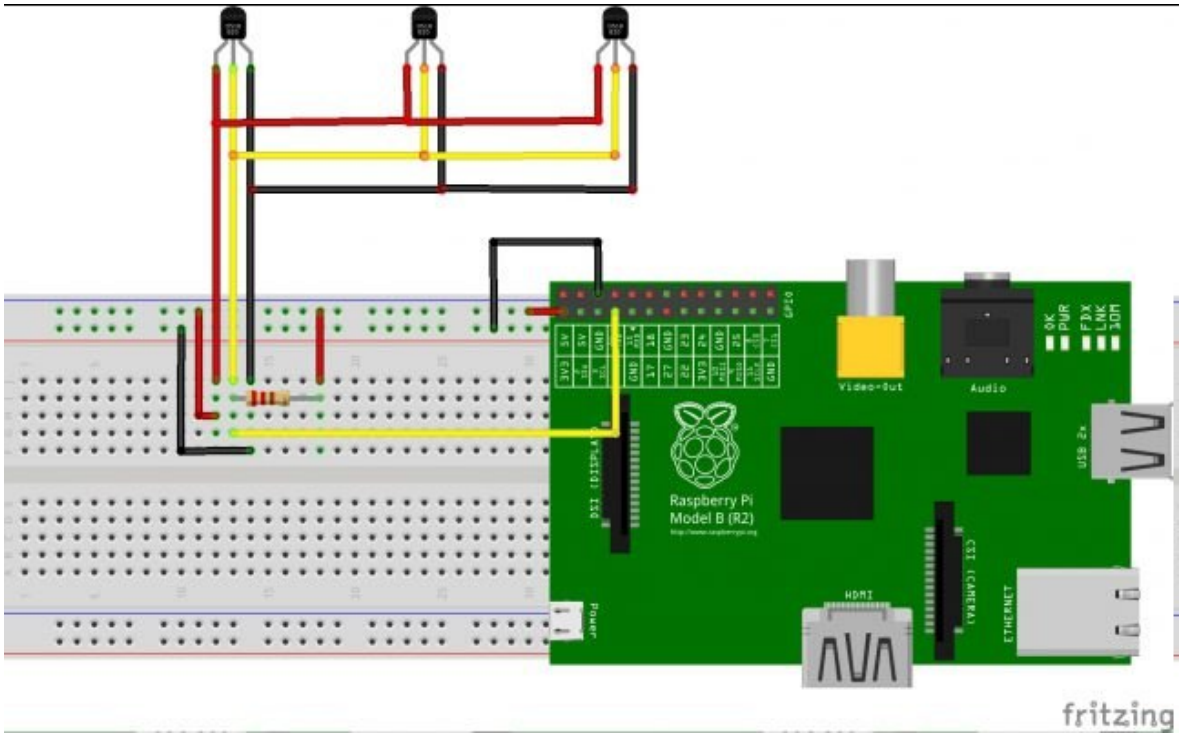
```python
# Los sensores de temperatura utilizan el bus 1-Wire
TIEMPO_TEMPERATURA = 5  # Tiempo de espera para volver a leer los valores de temperatura, expresado en segundos

# Funcion que implementa el algoritmo para obtener datos de los sensores de temperatura
def temperatura(queue):
    # Lista de ID unicos de los sensores dentro del bus
    lista_sensores_temperatura = ["28-03119779DFBE",
                                  "28-03119779E78D",
                                  "28-03119779C423",
                                  "28-031197793750"]

    lista_topics = ["termotanque/temperaturaambiental",
                    "termotanque/entradaagua",
                    "termotanque/salidaaguasolar",
                    "termotanque/salidaaguagas"]
    while True:
        for i in range(4):
            archivo_dato = open("/sys/bus/w1/devices/" + lista_sensores_temperatura[i] + "/w1_slave")
            datos = archivo_dato.read()
            archivo_dato.close()
            secondline = datos.split("\n")[1]
            sensor_temperatura_data = secondline.split(" ")[9]
            sensor_temperatura = float(sensor_temperatura_data[2:]) / 1000
            queue.put([lista_topics[i],round(sensor_temperatura, 1)])
        time.sleep(TIEMPO_TEMPERATURA)
```

Diagrama de conexión

En el siguiente diagrama se detalla la conexión entre la Raspberry Pi y los sensores de temperatura utilizados.



Elaboración propia.

Subproceso flama

Para detectar el tiempo de actividad del termotanque a gas auxiliar en el sistema, se utilizó el sensor KY-026 que funciona al captar la luz infrarroja (IR) emitida por el fuego, que se encuentra en un rango de longitud de onda de 760 nm a 1100 nm. El sensor convierte esta luz IR en una señal eléctrica tanto en salida digital o analógica.

Implementación

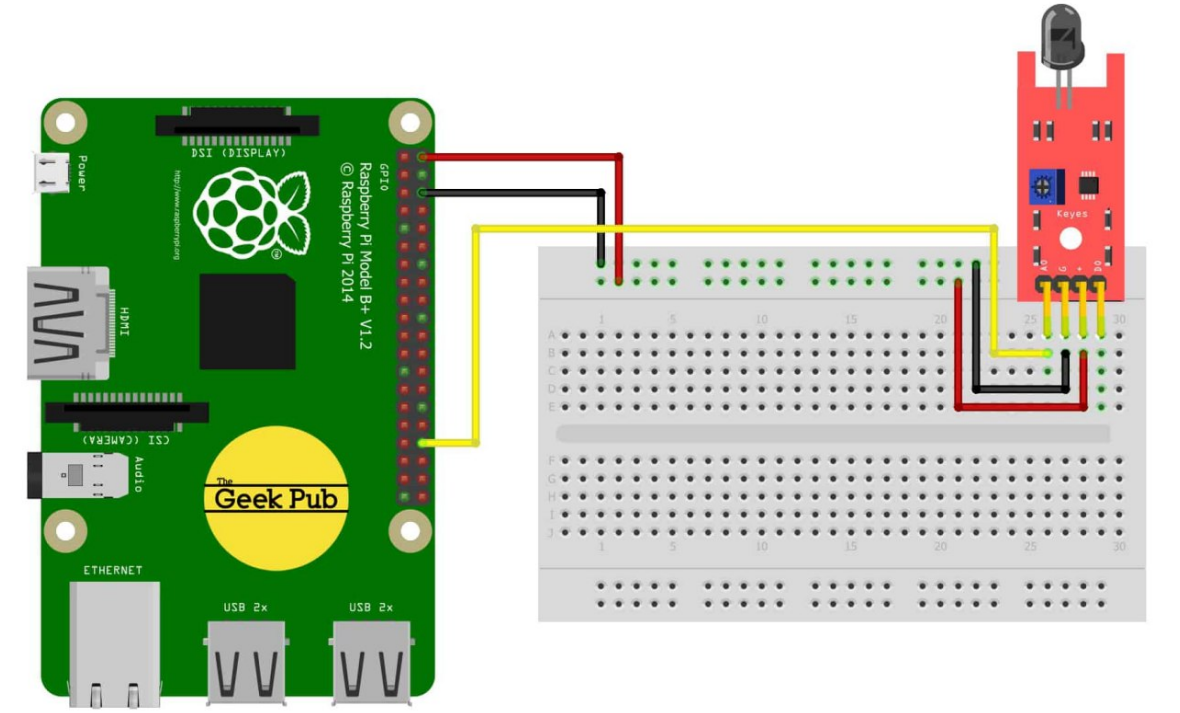
Se muestra a continuación el código correspondiente a la implementación de la función flama ejecutada por procesoFlama.

```
PIN_FLAMA = 11  # Pin de Data Imput del sensor Flama
TIEMPO_FLAMA = 5  # Tiempo de espera para volver a leer los valores del estado del sensor de flama

# Funcion que implementa el algoritmo para obtener datos de los sensores de flama
def flama(queue):
    topic="termotanque/flama"
    while True:
        queue.put([topic,not GPIO.input(PIN_FLAMA)])
        time.sleep(TIEMPO_FLAMA)
```

Diagrama de conexión

En el siguiente diagrama se detalla la conexión entre la Raspberry Pi y el sensor de flama utilizado.



Elaboración propia.

Servicio BD y visualización

A diferencia de la implementación que se presentó en el trabajo publicado[12] mencionado, para la que se utilizó el servicio ofrecido por Adafruit, en esta ocasión se buscó expandir las posibilidades a través de otra opción. Si bien es posible seguir utilizando un servicio en la nube como el mencionado, se diseñó e implementó una arquitectura que permite almacenar los datos y visualizarlos sin ninguna configuración adicional, tanto de forma local como a través de Internet a una nube privada.

Con esta finalidad, se utilizó la herramienta Docker que permite simplificar la implementación y gestión de aplicaciones mediante la virtualización de contenedores. De esa manera se facilita la portabilidad y escalabilidad de sistemas, lo que permite tener un control de entorno fácil y cómodo.

Además para el pasaje de información se utilizó el protocolo MQTT, protocolo de mensajería ligero y eficiente, especialmente utilizado en entornos de IOT.

En la figura 4 se puede observar las interacciones entre las distintas partes de la arquitectura que está compuesta por:

- Software de recolección de datos: algoritmo desarrollado en el presente trabajo para recolectar los datos de los distintos sensores.
- MQTT Broker: permite independizar toda la lógica de publicación/suscripciones del protocolo MQTT del software de recolección de datos, para una simplificación de la implementación de dicho protocolo y una alta escalabilidad de la arquitectura.
- Telegraf: es una herramienta de recopilación y procesamiento de datos, permite suscribirse a los tópicos MQTT y almacenarlos directamente en la BD.
- Influxdb: gestor de bases de datos diseñado para almacenar bases de datos de series temporales.
- Grafana: servicio web que nos permite visualizar en un panel de control los datos almacenados en InfluxDB.

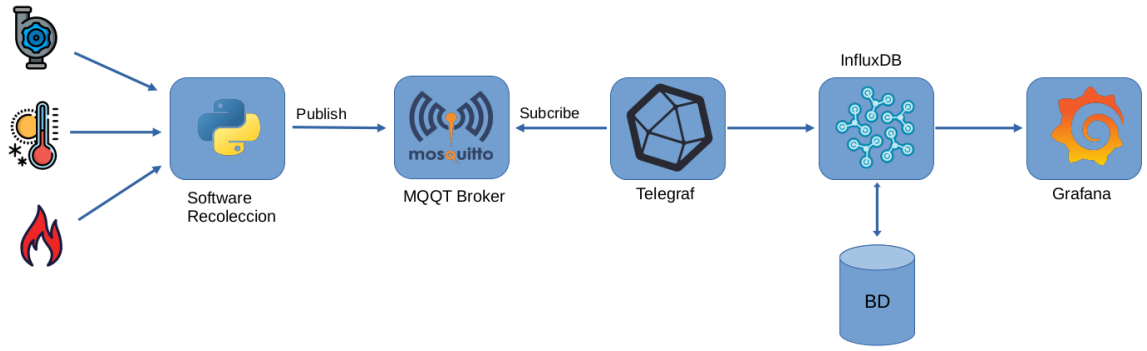


Fig. 4. Diagrama de arquitectura. Elaboración propia.

A continuación se describirán en detalle las partes, en forma individual, así como su implementación.

MQTT Broker

El propósito de esta herramienta es simplificar el desarrollo del software de recolección. Toda la lógica de publicación/suscripción de tópicos de MQTT se implementó sobre esta herramienta, lo que permite una arquitectura débilmente acoplada y altamente escalable.

El software de recolección publica los datos de los distintos sensores lo que posibilita que dispositivos o software se suscriban a los mismos. Además, permite la publicación de diferentes dispositivos para agregar en un futuro más estaciones de monitoreo de forma fácil sin ningún tipo de cambio en la arquitectura.

Eclipse Mosquitto es el MQTT Broker que se utiliza en este proyecto. La configuración del dockerfile para este servicio es la siguiente:

```
mosquitto:
    image: eclipse-mosquitto:2
    ports:
      - 1883:1883
    volumes:
      - ./mosquitto/mosquitto.conf:/mosquitto/config/mosquitto.conf
      - mosquitto_data:/mosquitto/data
      - mosquitto_log:/mosquitto/log
```

Se genera un archivo de configuración, mosquitto.conf, que contiene la configuración específica del servicio.

Telegraf

Permite suscribirse a los tópicos generados por el software de recolección de datos y almacenarlos en la base de datos influxdb.

En la configuración se va a suscribir a los tópicos:

- tsolar/caudal
- tsolar/entradaagua
- tsolar/salidaaguasolar
- tsolar/salidaaguagas
- tsolar/temperaturaambiental
- tsolar/flama

La configuración del dockerfile para este servicio es la siguiente:

```
telegraf:
    image: telegraf:1.18
      volumes:
        - ./telegraf/telegraf.conf:/etc/telegraf/telegraf.conf
      depends_on:
        - influxdb
```

Se genera un archivo de configuración, telegraf.conf, que contiene la configuración específica del servicio.

Influxdb

Es un sistema gestor de bases de datos diseñado para almacenar bases de datos de series temporales (TSBD - *Time Series Databases*). Se suele utilizar en aplicaciones de monitorización, donde es necesario almacenar y analizar grandes cantidades de datos con marcas de tiempo, como pueden ser datos de uso de cpu, uso memoria, datos de sensores de IoT, etc.

No es necesario generar ningún archivo de configuración ya que los parámetros se pasan directamente por docker. La configuración del dockerfile para este servicio es la siguiente:

```
 influxdb:
    image: influxdb:1.8
    ports:
      - 8086:8086
    volumes:
      - influxdb_data:/var/lib/influxdb
    environment:
      - INFLUXDB_DB=termotanque_db
      - INFLUXDB_ADMIN_USER=root
      - INFLUXDB_ADMIN_PASSWORD=root
      - INFLUXDB_HTTP_AUTH_ENABLED=true
```

Grafana

Es el servicio web que nos permite visualizar en un panel de control los datos almacenados en InfluxDB, con acceso en tiempo real o de forma histórica, de manera intuitiva, a los datos.

Para poder utilizar los beneficios brindados por Docker y no tener que reconfigurar constantemente el entorno, Grafana permite la autoconfiguración tanto del origen de los datos como de los dashboard que muestran los mismos.

Para ello se generan los siguientes archivos:

- datasource.yml que mantiene los datos requeridos para conectar con la BD.
- dashboard.yml y termotanque.yml que generan los dashboard para visualización de los datos.

La configuración del dockerfile para este servicio es la siguiente:

```
grafana:
    image: grafana/grafana:7.4.0
    ports:
      - 3000:3000
    volumes:
      - grafana_data:/var/lib/grafana
      - ./grafana-provisioning/:/etc/grafana/provisioning
    depends_on:
      - influxdb
```

El dashboard creado tiene el diseño que muestra la figura 5.

Además de lo planteado, se configuró la posibilidad de acceder al dashboard de forma pública o local un formato de url como el siguiente:

```
http://<ip o url de acceso>/grafana/termotanque
```

Esta funcionalidad permite que en caso de exponer el servicio a Internet, se pueda acceder de manera pública a los datos obtenidos del termotanque, tanto en tiempo real como el historia, sirviendo así como fuente de información y promoción.

Fig. 5. Diseño del Dashboard. Elaboración propia.

Prototipo Final

Como se comentó en capítulos anteriores, el prototipo fue instalado en el comedor de la Sede Atalntica de la Universidad Nacional de Río Negro. Para dicha instalación se utilizaron caños corrugados de pvc para aislar los cables y una caja estanca para aislar la Raspberry PI. Se agregaron además enchufes adicionales para el suministro eléctrico.

La disposición de los sensores instalados fue la que se indica en la figura 6, que incluye el diagrama del sistema de agua caliente con el cual se instaló el termotanque solar, especificando dónde serán colocados los distintos sensores utilizados:

- C1: sensor de caudal destinado a medir la cantidad de agua en L/min que pasan por el sistema.
- T1: temperatura del agua al ingresar al termotanque solar.
- T2: temperatura del agua al salir del termotanque solar e ingresar al termotanque a gas.
- T3: temperatura del agua al salir del termotanque a gas.
- T4: temperatura del ambiente cercano al termotanque solar.
- F1: sensor de flama para determinar en qué momentos se prende el mechero del termotanque a gas.

En cuanto a la conectividad, para acceder al dispositivo se lo conectó por medio de un cable UTP a la red lan del edificio, la que le suministra conectividad a Internet y brinda la posibilidad de acceso remoto.

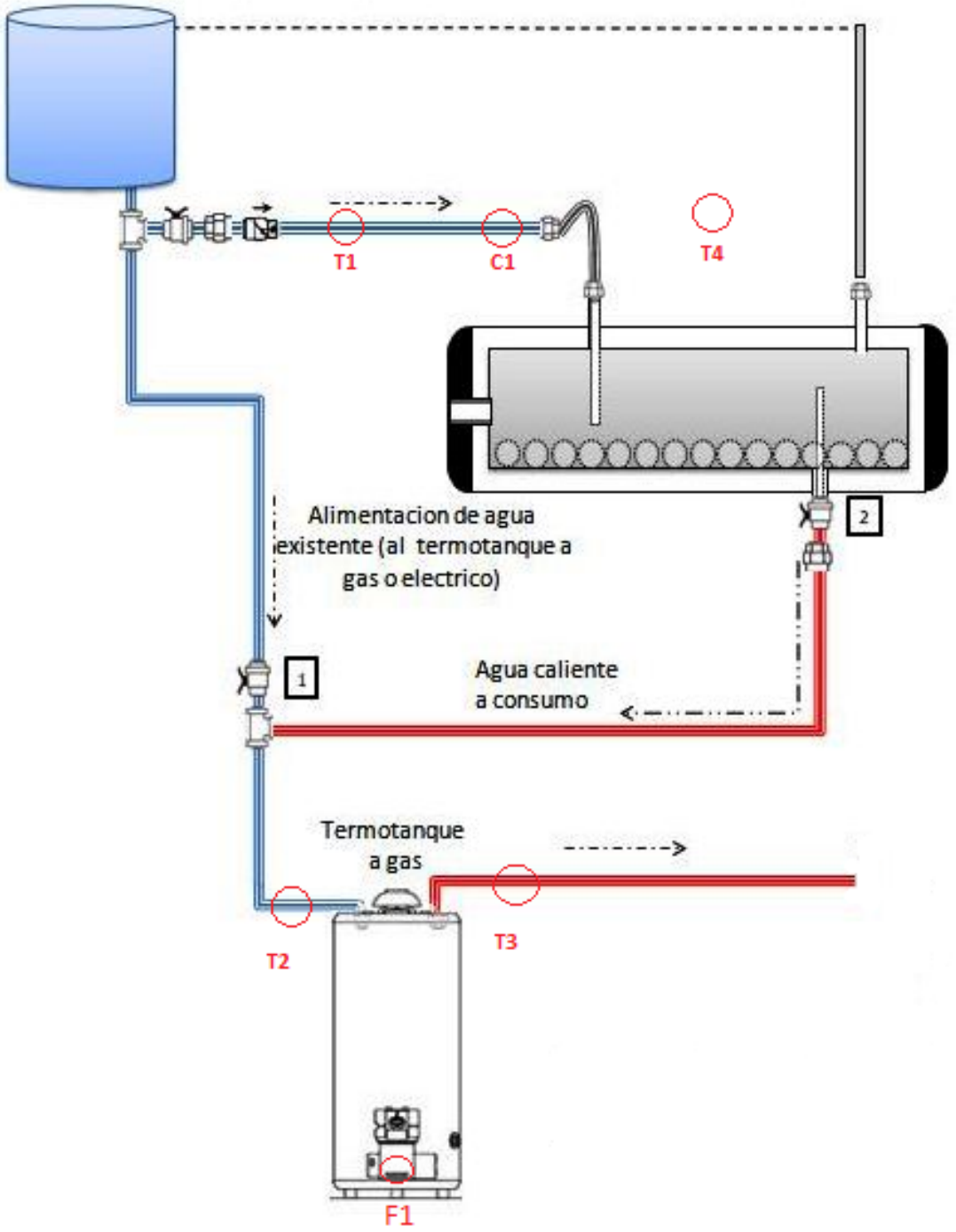


Fig. 6. Diagrama de la red de agua caliente con la ubicación de los sensores. Elaboración propia.

En las siguientes imágenes se muestra el montaje de los distintos sensores y el montaje final de todas las partes:

Fig. 7. Caja estanca que resguarda la Raspberry Pi junto a todos los terminales de conexión de los sensores. Elaboración propia.

Como se muestra en la figura 7, se montó una caja estanca para proteger los dispositivos tanto de las incidencias de clima como de posibles vandalismos que puedan causar la pérdida o daño del equipamiento instalado. La caja estanca es también el punto de unión entre la Raspberry Pi y los cables que conectan tanto los sensores como la red lan del dispositivo.

En la figura 8 se muestra la instalación de 3 sensores que se encuentran en el exterior:

- 1: Sensor tipo sonda de temperatura destinado a medir la temperatura de entrada de agua al sistema, marcado en la figura 6 como T1.
- 2: Sensor tipo sonda de temperatura destinado a medir la temperatura ambiental, marcado en la figura 6 como T4
- 3: Sensor de caudal que media la cantidad de agua que pasa por el sistema, marcado en la figura 6 como C1.

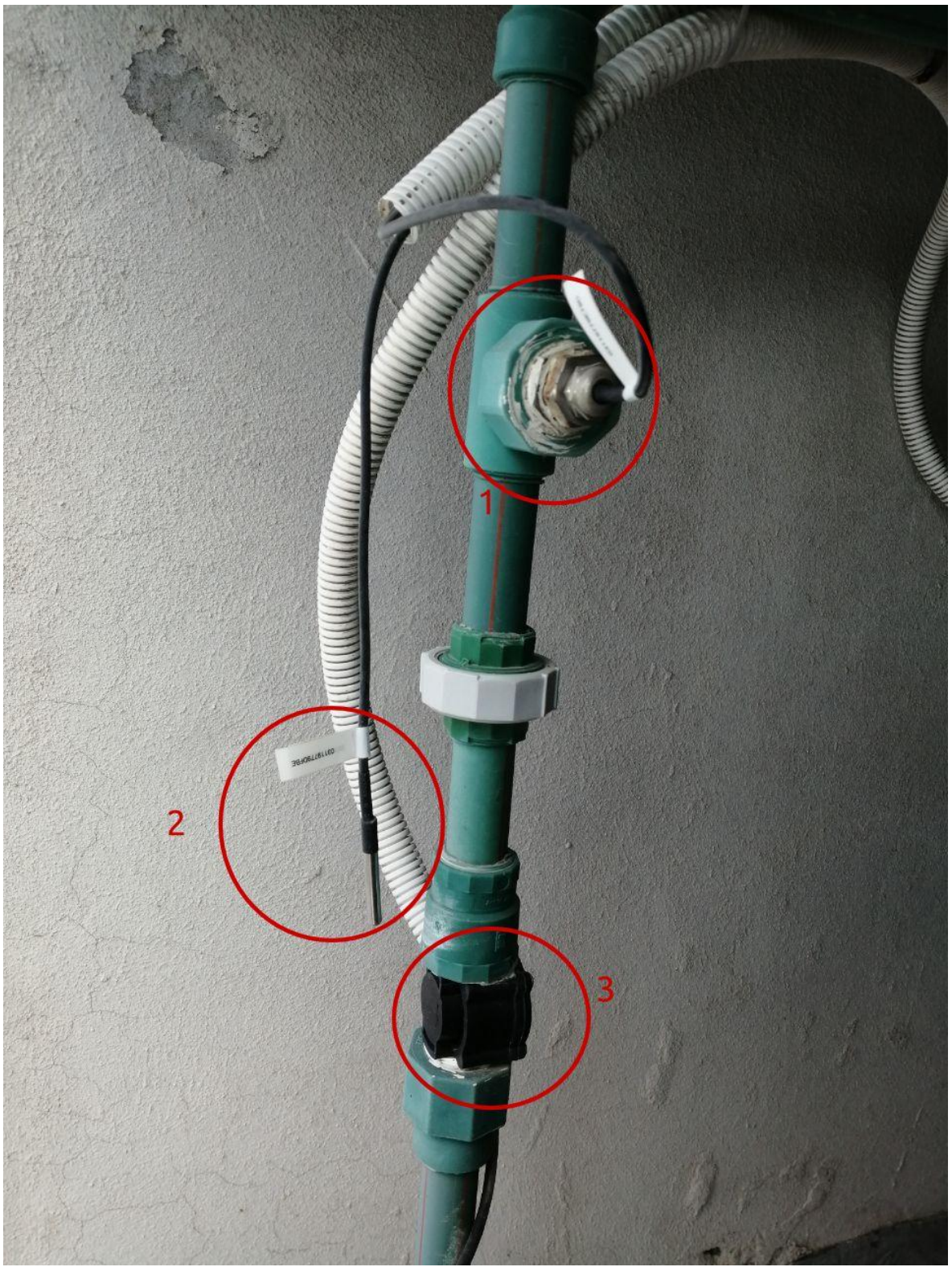


Fig. 8. Ubicación de sensores instalados, 1: sensor de temperatura de entrada de agua, 2: sensor de temperatura ambiental, 3: sensor de caudal. Elaboración propia.

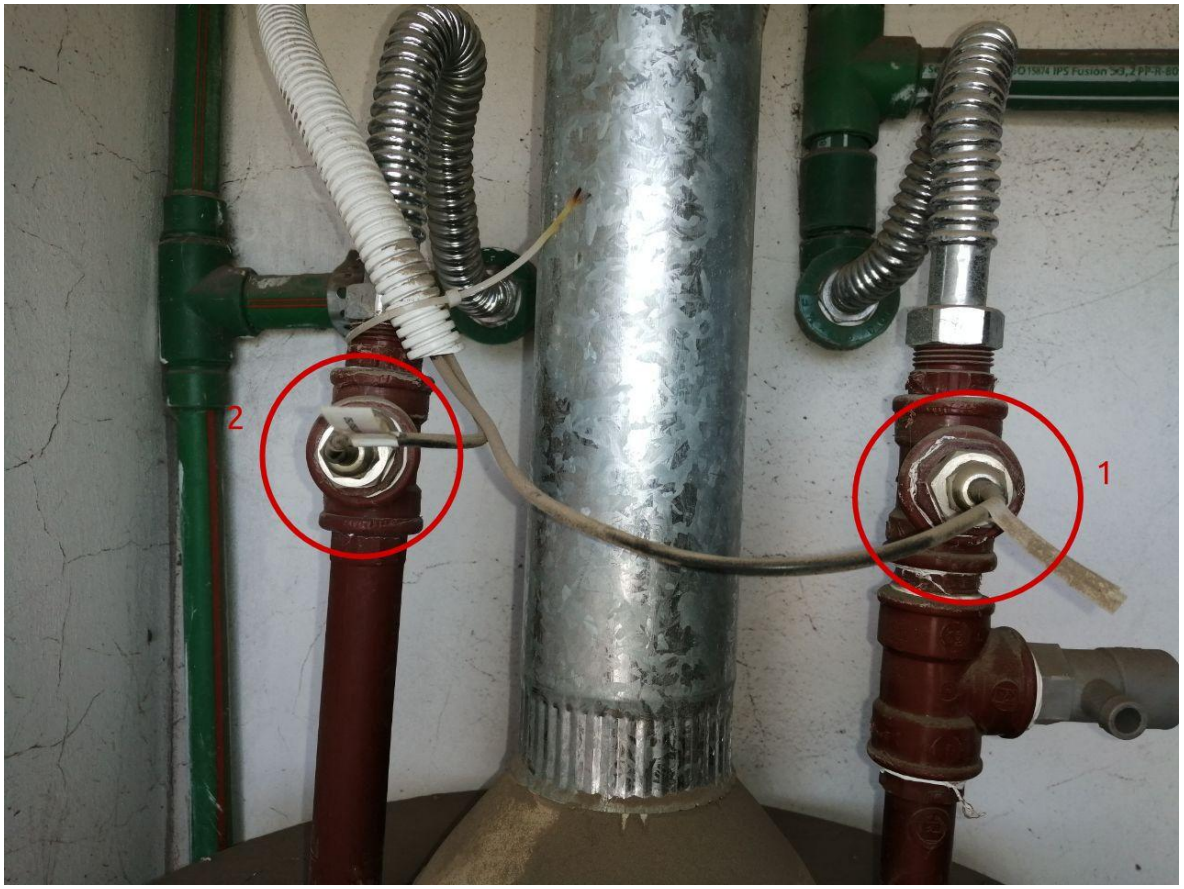


Fig. 9. Ubicación de instalación de 1: sensor de temperatura de salida de agua del termotanque solar, 2: sensor de temperatura de salida de agua del termotanque a gas. Elaboración propia.

El termotanque a gas de la instalación se encuentra ubicado en un lugar cerrado. En la figura 9 se detalla la ubicación de la instalación de los siguientes sensores:

1. Sensor tipo sonda de temperatura destinado a medir la temperatura de salida de agua del termotanque solar, marcado en la figura 6 como T2.
2. Sensor tipo sonda de temperatura destinado a medir la temperatura de salida de agua del termotanque a gas, marcado en la figura 6 como T3.

Además de los sensores anteriores, en la misma ubicación se encuentra el sensor de flama, como se muestra en la figura 10. Este sensor está marcado en la figura 6 como F1.

Fig. 10. Ubicación de la instalación del sensor de flama. Elaboración propia.

Conclusiones

En función de lo expuesto a lo largo de este trabajo se desarrolló exitosamente un prototipo que permite el monitoreo de las variables básicas de un sistema de agua caliente que cuenta con un termotanque solar y un termotanque a gas. Esta estación de monitoreo permite recolectar datos sobre el funcionamiento de dicho sistema, con la posibilidad de visualizarlos en tiempo real o en un registro histórico.

Una de las mejoras con respecto a la situación presentada en el trabajo publicado [12] fue la incorporación de la tecnología Docker, lo que permite acceder al entorno de almacenamiento y visualización de datos de manera local o en una nube privada. Se cumplió así con una de las mejoras planteadas ya que no se depende de un servicio exclusivo o de conectividad a Internet.

En cuanto a la metodología ágil utilizada para el desarrollo, la misma pudo llevarse a cabo adecuadamente, enfocándose en el desarrollo del prototipo y generando la menor cantidad posible de documentación.

Futuras líneas de trabajo

Utilizar Raspberry Pi como soporte tecnológico permite la incorporación de nuevos requisitos sin tener que cambiar drásticamente el hardware o modificar la tecnología subyacente. Esto hace factible enfocarse en aumentar la cantidad de parámetros de la instalación a monitorear, como por ejemplo, incorporar la radiación solar, la humedad del ambiente, etc. Esto permitiría reunir aún más datos del comportamiento del sistema donde está instalado.

Al implementar la nueva arquitectura de almacenamiento y visualización de datos, es posible pensar en un escenario de uso masivo de la solución. Para esto resulta útil disponer de funcionalidades que permitan realizar un análisis en mayor profundidad de los datos. Si se aplicara minería de datos sobre la información sería posible generar reportes analíticos con datos relevantes, definir las variaciones de consumos dependiendo de los sectores geográficos, establecer comparativas de uso diario, etc.

Anexo 1: Archivos de configuración

MQTT Broker

Archivo de configuración necesario para iniciar el servicio de Eclipse Mosquitto.

mosquitto.conf

Archivo de configuración para el MQTT Broker Eclipse Mosquitto.

```
listener 1883
allow_anonymous true
```

Telegraf

Configuración necesaria para iniciar el servicio de telegraf.

telegraf.conf

Configuración necesaria de telegraf que se debe de agregar al archivo telegraf.conf por defecto.

```
[[inputs.mqtt_consumer]]
   servers = ["tcp://mosquitto:1883"]

   topics = [
     "termotanque/#"
   ]

  data_format = "value"
  data_type = "float"

[[outputs.influxdb]]
  urls = ["http://influxdb:8086"]

  database = "termotanque_db"

  skip_database_creation = true

  username = "root"
  password = "root"
```

Grafana

Archivos necesarios para el inicio correcto del servicio de visualización Grafana.

datasource.yml

Archivo de configuración que indica los datos de acceso para que Grafana tenga acceso a la información guardada en la base de datos.

```
apiVersion: 1
datasources:
  - name: InfluxDB
    type: influxdb
    access: proxy
    database: termotanque_db
    user: root
    password: root
    url: http://influxdb:8086
    isDefault: true
    editable: true
```

dashboard.yml

Archivo de configuración necesaria para que Grafana cree automáticamente el acceso al dashboard con los datos del termotanque.

```
apiVersion: 1
providers:
- name: termotanque
  folder: ''
  type: file
  disableDeletion: false
  editable: true
  options:
    path: /etc/grafana/provisioning/dashboards
```

termotanque.yml

Archivo de configuración del dashboard que presenta los datos obtenidos del termotanque.

```
{
  "annotations": {
          "list": [
          {
          "builtIn": 1,
          "datasource": "-- Grafana --",
          "enable": true,
          "hide": true,
```

```
        "iconColor": "rgba(0, 211, 255, 1)",
        "name": "Annotations & Alerts",
        "type": "dashboard"
        }
        ]
},
"editable": true,
"gnetId": null,
"graphTooltip": 0,
"id": 2,
"links": [],
"panels": [
        {
        "datasource": null,
        "fieldConfig": {
        "defaults": {
        "color": {
        "mode": "palette-classic"
        },
        "custom": {
        "axisLabel": "",
        "axisPlacement": "auto",
        "barAlignment": 0,
        "drawStyle": "line",
        "fillOpacity": 10,
        "gradientMode": "none",
        "hideFrom": {
        "graph": false,
        "legend": false,
        "tooltip": false
        },
        "lineInterpolation": "linear",
        "lineWidth": 1,
        "pointSize": 5,
        "scaleDistribution": {
        "type": "linear"
        },
        "showPoints": "never",
        "spanNulls": true
        },
        "mappings": [],
        "thresholds": {
        "mode": "absolute",
        "steps": [
        {
                "color": "green",
                "value": null
        },
        {
                "color": "red",
                "value": 80
        }
        ]
        },
```

```
"unit": "short"
},
"overrides": []
},
"gridPos": {
"h": 8,
"w": 6,
"x": 0,
"y": 0
},
"id": 10,
"options": {
"graph": {},
"legend": {
"calcs": [],
"displayMode": "list",
"placement": "bottom"
},
"tooltipOptions": {
"mode": "single"
}
},
"pluginVersion": "7.4.0",
"targets": [
{
"groupBy": [
{
"params": [
        "$__interval"
],
"type": "time"
},
{
"params": [
        "null"
],
"type": "fill"
}
],
"measurement": "mqtt_consumer",
"orderByTime": "ASC",
"policy": "default",
"refId": "A",
"resultFormat": "time_series",
"select": [
[
{
        "params": [
        "value"
        ],
        "type": "field"
},
{
        "params": [],
```

```
                "type": "mean"
            }
            ]
            ],
            "tags": [
            {
            "key": "topic",
            "operator": "=",
            "value": "termotanque/temperaturaambiental"
            }
            ]
            }
            ],
            "title": "Temperatura ambiental",
            "type": "timeseries"
            },
            {
            "datasource": null,
            "fieldConfig": {
            "defaults": {
            "color": {
            "mode": "palette-classic"
            },
            "custom": {
            "axisLabel": "",
            "axisPlacement": "auto",
            "barAlignment": 0,
            "drawStyle": "line",
            "fillOpacity": 10,
            "gradientMode": "none",
            "hideFrom": {
            "graph": false,
            "legend": false,
            "tooltip": false
            },
            "lineInterpolation": "linear",
            "lineWidth": 1,
            "pointSize": 5,
            "scaleDistribution": {
            "type": "linear"
            },
            "showPoints": "never",
            "spanNulls": true
            },
            "mappings": [],
            "thresholds": {
            "mode": "absolute",
            "steps": [
            {
                "color": "green",
                "value": null
            },
            {
                "color": "red",
```

```
                "value": 80
            }
            ]
            },
            "unit": "short"
            },
            "overrides": []
            },
            "gridPos": {
            "h": 8,
            "w": 6,
            "x": 6,
            "y": 0
            },
            "id": 4,
            "options": {
            "graph": {},
            "legend": {
            "calcs": [],
            "displayMode": "list",
            "placement": "bottom"
            },
            "tooltipOptions": {
            "mode": "single"
            }
            },
            "pluginVersion": "7.4.0",
            "targets": [
            {
            "groupBy": [
            {
            "params": [
                "$__interval"
            ],
            "type": "time"
            },
            {
            "params": [
                "null"
            ],
            "type": "fill"
            }
            ],
            "measurement": "mqtt_consumer",
            "orderByTime": "ASC",
            "policy": "default",
            "refId": "A",
            "resultFormat": "time_series",
            "select": [
            [
            {
                "params": [
                "value"
                ],
```

```
                "type": "field"
            },
            {
                "params": [],
                "type": "mean"
            }
            ]
            ],
            "tags": [
            {
            "key": "topic",
            "operator": "=",
            "value": "termotanque/entradaagua"
            }
            ]
            }
            ],
            "title": "Temperatura entrada Termotanque Solar",
            "type": "timeseries"
            },
            {
            "datasource": null,
            "fieldConfig": {
            "defaults": {
            "color": {
            "mode": "palette-classic"
            },
            "custom": {
            "axisLabel": "",
            "axisPlacement": "auto",
            "barAlignment": 0,
            "drawStyle": "line",
            "fillOpacity": 10,
            "gradientMode": "none",
            "hideFrom": {
            "graph": false,
            "legend": false,
            "tooltip": false
            },
            "lineInterpolation": "linear",
            "lineWidth": 1,
            "pointSize": 5,
            "scaleDistribution": {
            "type": "linear"
            },
            "showPoints": "never",
            "spanNulls": true
            },
            "mappings": [],
            "thresholds": {
            "mode": "absolute",
            "steps": [
            {
                "color": "green",
```

```
                "value": null
            },
            {
                "color": "red",
                "value": 80
            }
            ]
            },
            "unit": "short"
            },
            "overrides": []
            },
            "gridPos": {
            "h": 8,
            "w": 6,
            "x": 12,
            "y": 0
            },
            "id": 6,
            "options": {
            "graph": {},
            "legend": {
            "calcs": [],
            "displayMode": "list",
            "placement": "bottom"
            },
            "tooltipOptions": {
            "mode": "single"
            }
            },
            "pluginVersion": "7.4.0",
            "targets": [
            {
            "groupBy": [
            {
            "params": [
                "$__interval"
            ],
            "type": "time"
            },
            {
            "params": [
                "null"
            ],
            "type": "fill"
            }
            ],
            "measurement": "mqtt_consumer",
            "orderByTime": "ASC",
            "policy": "default",
            "refId": "A",
            "resultFormat": "time_series",
            "select": [
            [
```

```
{
		"params": [
		"value"
		],
		"type": "field"
},
{
		"params": [],
		"type": "mean"
}
]
],
"tags": [
{
"key": "topic",
"operator": "=",
"value": "termotanque/salidaaguasolar"
}
]
}
],
"title": "Temperatura agua salida Termotanque Solar",
"type": "timeseries"
},
{
"datasource": null,
"fieldConfig": {
"defaults": {
"color": {
"mode": "palette-classic"
},
"custom": {
"axisLabel": "",
"axisPlacement": "auto",
"barAlignment": 0,
"drawStyle": "line",
"fillOpacity": 10,
"gradientMode": "none",
"hideFrom": {
"graph": false,
"legend": false,
"tooltip": false
},
"lineInterpolation": "linear",
"lineWidth": 1,
"pointSize": 5,
"scaleDistribution": {
"type": "linear"
},
"showPoints": "never",
"spanNulls": true
},
"mappings": [],
"thresholds": {
```

```
"mode": "absolute",
"steps": [
{
        "color": "green",
        "value": null
},
{
        "color": "red",
        "value": 80
}
]
},
"unit": "short"
},
"overrides": []
},
"gridPos": {
"h": 8,
"w": 6,
"x": 18,
"y": 0
},
"id": 8,
"options": {
"graph": {},
"legend": {
"calcs": [],
"displayMode": "list",
"placement": "bottom"
},
"tooltipOptions": {
"mode": "single"
}
},
"pluginVersion": "7.4.0",
"targets": [
{
"groupBy": [
{
"params": [
        "$__interval"
],
"type": "time"
},
{
"params": [
        "null"
],
"type": "fill"
}
],
"measurement": "mqtt_consumer",
"orderByTime": "ASC",
"policy": "default",
```

```
"refId": "A",
"resultFormat": "time_series",
"select": [
[
{
        "params": [
        "value"
        ],
        "type": "field"
},
{
        "params": [],
        "type": "mean"
}
]
],
"tags": [
{
"key": "topic",
"operator": "=",
"value": "termotanque/salidaaguagas"
}
]
}
],
"title": "Temperatura Agua Salida Termotanque Gas",
"type": "timeseries"
},
{
"datasource": null,
"description": "caudal de agua que pasa por el sistema en L/m",
"fieldConfig": {
"defaults": {
"color": {
"mode": "palette-classic"
},
"custom": {
"axisLabel": "",
"axisPlacement": "auto",
"barAlignment": 0,
"drawStyle": "line",
"fillOpacity": 10,
"gradientMode": "none",
"hideFrom": {
"graph": false,
"legend": false,
"tooltip": false
},
"lineInterpolation": "linear",
"lineWidth": 1,
"pointSize": 5,
"scaleDistribution": {
"type": "linear"
},
```

```
"showPoints": "never",
"spanNulls": true
},
"mappings": [],
"thresholds": {
"mode": "absolute",
"steps": [
{
        "color": "green",
        "value": null
},
{
        "color": "red",
        "value": 80
}
]
},
"unit": "short"
},
"overrides": []
},
"gridPos": {
"h": 8,
"w": 16,
"x": 0,
"y": 8
},
"id": 2,
"options": {
"graph": {},
"legend": {
"calcs": [],
"displayMode": "list",
"placement": "bottom"
},
"tooltipOptions": {
"mode": "single"
}
},
"pluginVersion": "7.4.0",
"targets": [
{
"groupBy": [
{
"params": [
        "$__interval"
],
"type": "time"
},
{
"params": [
        "null"
],
"type": "fill"
```

```
}
],
"measurement": "mqtt_consumer",
"orderByTime": "ASC",
"policy": "default",
"refId": "A",
"resultFormat": "table",
"select": [
[
{
        "params": [
        "value"
        ],
        "type": "field"
},
{
        "params": [],
        "type": "mean"
}
]
],
"tags": [
{
"key": "topic",
"operator": "=",
"value": "termotanque/caudal"
}
]
}
],
"timeFrom": null,
"timeShift": null,
"title": "Caudal",
"type": "timeseries"
},
{
"datasource": null,
"fieldConfig": {
"defaults": {
"color": {
"mode": "palette-classic"
},
"custom": {
"axisLabel": "",
"axisPlacement": "auto",
"barAlignment": 0,
"drawStyle": "line",
"fillOpacity": 10,
"gradientMode": "none",
"hideFrom": {
"graph": false,
"legend": false,
"tooltip": false
},
```

```
"lineInterpolation": "linear",
"lineWidth": 1,
"pointSize": 5,
"scaleDistribution": {
"type": "linear"
},
"showPoints": "never",
"spanNulls": true
},
"mappings": [],
"thresholds": {
"mode": "absolute",
"steps": []
},
"unit": "short"
},
"overrides": []
},
"gridPos": {
"h": 8,
"w": 8,
"x": 16,
"y": 8
},
"id": 12,
"options": {
"graph": {},
"legend": {
"calcs": [],
"displayMode": "list",
"placement": "bottom"
},
"tooltipOptions": {
"mode": "single"
}
},
"pluginVersion": "7.4.0",
"targets": [
{
"groupBy": [
{
"params": [
        "$__interval"
],
"type": "time"
},
{
"params": [
        "null"
],
"type": "fill"
}
],
"measurement": "mqtt_consumer",
```

```
        "orderByTime": "ASC",
        "policy": "default",
        "refId": "A",
        "resultFormat": "time_series",
        "select": [
        [
        {
                "params": [
                "value"
                ],
                "type": "field"
        },
        {
                "params": [],
                "type": "mean"
        }
        ]
        ],
        "tags": [
        {
        "key": "topic",
        "operator": "=",
        "value": "termotanque/flama"
        }
        ]
        }
        ],
        "timeFrom": null,
        "timeShift": null,
        "title": "Actividad Termotanque a Gas",
        "type": "timeseries"
        }
  ],
  "schemaVersion": 27,
  "style": "dark",
  "tags": [],
  "templating": {
        "list": []
  },
  "time": {
        "from": "now-15m",
        "to": "now"
  },
  "timepicker": {},
  "timezone": "",
  "title": "Termotanque_Solar",
  "uid": "p4COZCNIk",
  "version": 2
}
```

Referencias

[1] B. Metz, O.R. Davidson, P.R. Bosch, R. Dave, L.A. Meyer (eds). 2007. CAMBIO CLIMÁTICO 2007 Informe del Grupo de Trabajo III - Mitigación del Cambio Climático. Cambridge University Press, Cambridge, United Kingdom and New York, NY, USA.

[2] Fundacion Colsecor. (2023). Estudio sobre consumo y percepciones en torno a la energía. fundacioncolsecor. https://www.fundacioncolsecor.org.ar/informes/mientras-mas-chica-localidad-menor-acceso-gas-natural-n12398

[3] enertik. (2023). Categoria Solar-termica. enertik. https://enertik.com/ar/categoria/solar-termica/

[4] Ypfsolar. (2023). Termotanques. Ypfsolar. https://ypfsolar.com/soluciones/termotanques/

[5] Somoza, J. I., Tarditti, O., Christophersen, B., Belogi, A., Sivo, M., Soto, A., Gutiérrez, O., Prezzo, M., Canto Trione, D., & Sánchez, F. (2021). Termotanques solares comunitarios para mitigar necesidades energéticas. *INNOVA UNTREF. Revista Argentina De Ciencia Y Tecnología*, *1*(1). Recuperado a partir de https://www.revistas.untref.edu.ar/index.php/innova/article/view/956

[6] Cyrulies, Ernesto Enrique; Sartarelli, Salvador Andrés; Echarri, Rodolfo Manuel; Vera, Sergio Gustavo; Construcción de termotanques solares de bajo costo: un proyecto de voluntariado universitario; Asociación Argentina de Energía Solar; Avances en Energías Renovables y Medio Ambiente; 16; 10-2012; 9-17

[7] Lanson, A. and Bianchi, A. (2015) *Estimación del ahorro energético que podria obtenerse del uso de sistemas termosolares híbridos en distintos puntos de Argentina.* Energías Renovables y Medio Ambiente, 36 . pp. 67-74. ISSN 2684-0073

[8] Lanson, A. | Bianchi, A.. (2016, octubre). Análisis y proyección del modelo de ahorro que representa el uso de un sistema híbrido solareléctrico para calentamiento de agua domiciliaria. XXXIX Reunión de Trabajo de la Asociación Argentina de Energías Renovables y Medio Ambiente (ASADES) (La Plata, 2016), 978-987-29873-0-5, pp. 57-68

[9] Lanson, Anahí | Righini, Raúl | Aguerre, R. J. . (2017). Mapa de ahorro energético obtenible con sistemas solares híbridos de calentamiento de agua domiciliaria. Avances en Energías Renovables y Medio Ambiente; , 21, pp. 83-94

[10] Putra, Michael, Yudishtira y Kanigoro. 2015. Design and Implementation of Web Based Home Electrical Appliance Monitoring, Diagnosing, and Controlling System

[11] Dr. Pablo Fabián CARRANZA. (2017). “Uso de Energías Renovables en la Universidad Nacional de Río Negro”. UNIVERSIDAD CULTURA Y SOCIEDAD – SPU.

[12] Valsecchi, J., Malpeli, G., Martinez Luquez, J. C., & Vivas, H. L. (2020). Desarrollo de un prototipo para estación de monitoreo de un dispositivo híbrido de termotanques solar ya gas natural instalado en la Sede Atlántica de la Universidad Nacional de Río Negro. In *XXVI Congreso Argentino de Ciencias de la Computación (CACIC)(Modalidad virtual, 5 al 9 de octubre de 2020)*. Recuperado a partir de http://sedici.unlp.edu.ar/handle/10915/114471