Stream Processing para IOT en la agricultura usando MQTT

Trabajo Final Internet de las Cosas (IoT)

Autor: Antonio Gómez Giménez
Email: i72gogia@uco.es

Córdoba
(2022/2023)

UNIVERSIDAD D CÓRDOBA

Índice:

1. Introducción:

Hoy en día, en pleno siglo XXI, podemos apreciar la continua evolución de las tecnologías que nos rodean y el continuo desarrollo que estas llevan a pasos agigantados. Desde ordenadores cuánticos hasta pequeñas placas con cierta capacidad de cómputo.

Todos los tipos de tecnologías tienen su relevancia, pero el concepto de IoT(internet de las cosas) ha obtenido bastante relevancia debido a las mejoras que este ha introducido, siendo muy importante para ciertos conceptos como automatización de empresas, el uso casi necesario en big data, etc.

IoT no es un concepto definido como tal, pero se puede llegar a entender cómo la digitalización de cosas y su respectiva conexión a internet. Por ejemplo una serie de sensores que reciben información y la traspasan entre otros dispositivos o internet para tomar una serie de decisiones automatizadas y sin intervención humana, todo ello teniendo en cuenta que los datos se generan en tiempo real.

IoT se aplica en distintos ámbitos como transporte, minería de datos, domótica, automatización, etc. En nuestro caso, nos centraremos en IoT para la automatización y recolección de información, concretando más en el ámbito de la agricultura.

Como bien se sabe, la agricultura es un campo muy importante (existiendo desde el comienzo del ser humano) ya que genera alimento para gran cantidad de personas, pero en los tiempos actuales, teniendo en cuenta que la cantidad de seres humanos ha aumentado, es necesario añadir ciertas mejoras para mejorar la productividad.
Por ello, IoT es perfecto para este motivo, ya que permite recolectar información para mejorar la productividad (mejorando calidad y cantidad) y mejora la eficiencia, por ejemplo en recolección.

En definitiva, el IoT en la agricultura permite mejorar la calidad de vida, tanto para los propios agricultores, como el resto de personas, permitiendo facilidades al agricultor (control de forma remota) y mejorando calidad y cantidad del producto.

Por todo lo explicado anteriormente, se plantea crear un sistema IoT en un entorno agrícola, para adquirir estos conocimientos y añadir ciertas tecnologías como MQTT para que sea posible.

Como IoT en agricultura se encuentra en desarrollo y se derivan muchas ramas de investigación sobre la misma, se pretende introducir Stream Processing, ya que es un concepto muy interesante y que actualmente no hay mucho desarrollo en esta dirección en este campo.

Por último, se realizará un despliegue simulado y posteriormente, dentro de las capacidades posibles, un despliegue en campo, ya que el despliegue a nivel de campo es más costoso.

2. Motivación:

Como bien se explicó en el apartado de introducción, IoT se encuentra en continuo desarrollo y ha generado gran cantidad de beneficios para distintos campos, como por ejemplo la agricultura. Por lo tanto, aprovechando algunos conceptos vistos en la asignatura, se planteó la creación de un sistema IoT, primeramente simulado y posteriormente en campo, aprovechando la disponibilidad de un huerto propio.
Para, de esta forma, comprobar la utilidad de un sistema IoT, y las ventajas que este proporciona.

Aprovechando que la creación de un sistema IoT es relativamente sencillo, como este es un campo con muchas ramas de investigación con distintas orientaciones, se pretende ver un nuevo punto de vista donde se aplica el Stream Processing a un sistema IoT, buscando dar un nuevo punto de vista al IoT en la agricultura.

Por último, se pretendía con este proyecto comprender y ser capaz de utilizar aquellas herramientas que se han visto a lo largo del curso como pueden ser:

- El protocolo MQTT.
- Python para un entorno de simulación.
- MicroPython para el uso de distintas Esp32.
- El uso de ubidots para conectar el sistema con una plataforma integradora en la nube.

Así pues, teniendo en cuenta todos los puntos explicados anteriormente, este proyecto es ideal para tener un primer contacto con esta tendencia tecnológica de Iot en la agricultura, ya que engloba la creación de un sistema Iot en la agricultura añadiendo distinta herramientas englobando todo el proceso, y además, añade un cierto concepto de investigación que puede ser muy interesante.

3. Objetivos:

3.1. Objetivo principal:

El principal objetivo del proyecto que se pretende llevar a cabo, es crear un sistema IoT orientado para la agricultura, teniendo en cuenta el concepto de Stream Processing en el diseño de la misma, y basado en Mosquitto para la interconexión entre dispositivos, ya sean simulados o reales.

El sistema tiene como idea ser implantado en un huerto propio como ejemplo, para entender mejor los distintos problemas de despliegue que puedan encontrar. OBJ-01.

3.2. Objetivos específicos:

Los objetivos específicos que se pretenden lograr para este proyecto son los siguientes:

- Montar un servidor Mosquitto que permita la conexión de distintos dispositivos que se encuentren en la misma red local. OBJ-02.
- Aplicar Stream Processing a un sencillo nivel como prueba, para futuras mejoras (sistema de control). OBJ-03.
- Aplicar el sistema en un entorno simulado para su posterior adaptación en campo. Incluyendo pasar código de python a micropython. OBJ-04.
- Crear conexión con ubidots para comprobar la correcta recepción de datos en una plataforma integradora en la nube. OBJ-05.
- Plantear cómo se realizaría el despliegue IoT a nivel de campo. OBJ-06.

3.3. Objetivos didácticos:

Los objetivos didácticos que se pretenden lograr para este proyecto son los siguientes:

- Comprender mejor los distintos problemas existentes al intentar desplegar un sistema IoT. OBJ-07.
- Afianzar los conocimientos adquiridos con Mosquito. OBJ-08.
- Afianzar los conocimientos adquiridos con Ubidots. OBJ-09.
- Aprender a crear un entorno simulado para el desarrollo de un entorno IoT. OBJ-10.

4. Revisión bibliográfica:

4.1. Antecedentes y estado actual:

Para poder hablar de la idea que se pretende llevar a cabo, es necesario realizar una búsqueda bibliográfica donde se comente el estado actual del campo del proyecto propuesto y los antecedentes posibles del mismo. Como el tema a tratar es el Stream Processing para IOT en la agricultura usando MQTT, vamos a explorar cada elemento por separado.

IoT, también conocido como internet de las cosas, es un concepto que lleva bastante tiempo existiendo, y de hecho, la relación de este con la agricultura ha sido un objeto de estudio por bastante tiempo. Aunque ha sido estudiado durante muchos años, en la actualidad sigue siendo un campo con bastante relevancia y con bastante cantidad de gente cuyo objetivo es encontrar mejoras sobre dicho campo. Parece un campo con bastante proyección a futuro.

De hecho se pueden encontrar bastantes artículos de resumen como por ejemplo, **[1]** donde se realiza un resumen más general del estado de la agricultura respecto al uso de IoT.

También se pueden encontrar artículos más concretos que tratan el desarrollo de esta tecnología y la solución a algunos problemas hasta la actualidad. Algunos ejemplos pueden ser: el artículo **[2]** donde se nos explica el uso de la nube para el almacenamiento de los datos para los sensores, el artículo **[3]** que se centra más en la seguridad y privacidad de los datos recogidos por los sensores dando soluciones de los mismos, o incluso, el artículo **[4]** que nos comenta el porqué de la necesidad y creación de este campo concreto explicando la necesidad de crear una agricultura inteligente que permita mejorar la productividad permitiendo alimentar a mucha más cantidad de personas (entre otros beneficios), o incluso una idea propuesta de robot multipropósito (AGRIBOT 1.0) **[5]** que gracias al iot, fuera capaz de adaptarse a terreno inadecuado como es el campo agrícola.

Como podemos ver, hay mucha información sobre la evolución de este campo, sobre todo, relacionado con las distintas tecnologías utilizadas, tanto para los sensores, envío de información o el almacenamiento de datos. Como vimos anteriormente aparecieron distintas tecnologías como la nube para el almacenamiento de los datos o incluso la blockchain que añadía cierta seguridad sobre los datos recibidos por los sensores y el almacenamiento de los mismos, pero también aparecen distintos tipos de protocolos de red como LoraWan **[6]** que usa tecnología Lora para zonas de área amplia con redes de poca potencia permitiendo conectar sensores en zonas complicadas sin capacidad de instalar WiFi.
También se comenta la aparición del 5G **[7]** y el uso del mismo en este campo y mejoras producidas por el mismo.

Lógicamente, al hablar de tecnologías emergentes en IoT (independientemente de la agricultura), aparece el protocolo Mqtt (Message Queue Telemetry Transport) **[8]** que brilla por su sencillez, permitiendo una comunicación sencilla entre dispositivos, tanto sensores como intermediarios con la nube. Se basa en el concepto de publicación/suscripción, donde los dispositivos pueden publicar tópicos o suscribirse a los mismos, permitiendo así acceder a la información de forma sencilla gracias al broker, que es una entidad que se encarga de gestionar todo el proceso de publicación/suscripción entre dispositivos.

Por último, antes de comentar el estado actual del tema tratado y hacia dónde se dirige, es necesario explicar el concepto de Stream processing. Para la explicación de este concepto nos encontramos artículos como **[9]** o **[10]**, donde se nos comenta que el Stream processing es una técnica de gestión de datos donde se divide del procesamiento, es decir, existen ciertos tipos de datos que, sí son un flujo continuo, se permite analizar, transformar, filtrar e incluso mejorar, rápidamente dichos datos en tiempo real, en distintos procesadores.

Una vez explicados estos conceptos, se comenta el estado actual de la agricultura con IoT. Existen artículo como **[11]** donde se da un resumen más general del estado actual sobre este campo, pero podemos encontrar otros donde por ejemplo el artículo **[12]** nos comenta los grandes beneficios que se obtienen con una agricultura smart basándose en IoT, dando recomendaciones sobre estrategias basadas en IoT centrándose en herramientas de monitorización del riego usadas en Indonesia.

También encontramos otros artículos muy interesantes como el artículo **[13]** donde se comenta el big data y ciertos métodos de inteligencia artificial que permiten utilizar todos los datos extraídos de los sensores y darle valor a los mismos.

Sin entrar en mucho detalle se comentan también los siguientes artículos donde, en el artículo **[14]** se realiza una serie de reviews sobre aplicaciones actuales de IoT en agricultura, en el artículo **[15]** se explican avances actuales en IoT en la agricultura, y el artículo **[16]** donde se comenta una forma de mejorar el funcionamiento de IoT basándose en la información de los sensores para mejorar la estructura del sistema, lo más similar al concepto que se pretende explicar.

Como podemos apreciar, tras haber realizado este informe bibliográfico, podemos decir que la idea que se pretende llevar a cabo es puntera, ya que no hay ningún artículo que trate esta idea directamente, y de hecho, hemos encontrado buenas bases con información relevante para dicho proyecto.
Resumiendo el proyecto, no hay ningún otro artículo que se centre en el uso de MQTT para comprobar la estructura del sistema IoT (metadatos) al realizar stream processing. Ya que como se explicó anteriormente, si usamos Stream Processing para procesar los datos en los sensores y en el Fog, evitaremos sobrecargar la red y la nube.

Por último, tras analizar el campo de IoT en relación con la agricultura, se puede decir que es un campo en continuo desarrollo, donde nos encontramos distintas ramificaciones de la misma como puede ser el proyecto que se pretende llevar a cabo.

5. Restricciones:

5.1. Factores Dato

Los factores datos son todas aquellas restricciones que están adheridas al problema de forma implícita y que no son modificables o elegibles, es decir, son impuestas por la naturaleza del problema. Para nuestro problema en concreto nos encontramos las siguientes restricciones:

- El **tiempo**. Al ser un proyecto de una asignatura del máster, el proyecto tiene una fecha límite que no puede sobrepasarse, siendo la mayor prioridad del proyecto. Por lo tanto, si es necesario rechazar posibles mejoras del proyecto como adquisición de material a largo plazo (ciertos sensores), se prescindirá para cumplir esta restricción.

- El **dinero**. Los sensores son muy útiles pudiendo ser con pantallas integradas para ver la información captada, u otro tipo de mejoras, etc. Pero su precio es bastante elevado dependiendo del sensor y las mejoras que este conlleve. Por ello, se pretende buscar el material disponible en la uco para satisfacer este apartado, ya que se realizará en un entorno simulado, pero para el apartado de campo, será necesario disponer de sensores, esp32 y otros materiales para poder realizar la adquisición de datos.

5.2. Factores estratégicos

Respecto a los factores estratégicos, son aquellas restricciones que pueden ser modificables. Son alternativas a tener en cuenta a la hora de resolver el problema, de tal forma, que se escogerán aquellas que sean más beneficiosas para el proyecto. Las restricciones escogidas para este proyecto son:

- **Sensores.** Solo se usarán sensores de ejemplo que tengan algún tipo de relación con la agricultura, para sacar aquellos parámetros que sean interesantes.

- **Plataforma de desarrollo.** La plataforma que se pretende usar para el desarrollo del sistema es linux, ya que para el broker de MQTT podemos utilizar este SO, el cual nos brinda ciertas mejoras y sencillez frente a otros.

- **Lenguajes y librerías.** Respecto al lenguaje, se pretende usar python respecto a otros, por su sencillez en su uso y por la cantidad de librerías existentes que permiten aumentar la velocidad en el desarrollo del proyecto. Sobre todo por la existencia de la librería de MQTT para broker y cliente. Se pretende usar micropython para la implementación a través de thonny para los dispositivos reales en el despliegue en campo.

6. Especificación de requisitos:

En este apartado se pretende realizar un análisis donde se describen los requisitos que debe cumplir el proyecto, cumpliendo con los **factores Dato** y **estratégicos** comentados en puntos anteriores. Estos requisitos se dividen en 4 tipos: **requisitos de usuario, requisitos funcionales, requisitos no funcionales y requisitos de información**.

6.1. Requisitos de usuario

Los **requisitos de usuario** son aquellos requisitos que vienen dados por el usuario y condicionan el proyecto, de tal forma que, estos requisitos deben coincidir con la funcionalidad del proyecto que se está desarrollando. Para mayor claridad, a cada requisito se le nombrará con la nomenclatura **RU** añadiendo el número del requisito.

RU - 1: El sistema debe ser capaz de almacenar los datos en conjuntos de 50 para realizar el procesamiento

RU - 2: Una vez se dispongan de todos los datos (los 50), el sistema debe de realizar todo el procesamiento antes de introducir un nuevo dato, para evitar perder información

RU - 3: El sistema debe de enviar la información tras procesarla a la plataforma integradora en la nube.

RU - 4: El sistema debe ser capaz de realizar todo lo anterior teniendo en cuenta distintos tipos de sensores.

RU - 5: La estructura del sistema se deberá dividir en Edge, Fog y Cloud.

RU - 6: Se realizará la estructura necesaria para el correcto funcionamiento de MQTT.

6.2. Requisitos funcionales

Los **requisitos funcionales** son aquellos que hacen referencia a la funcionalidad concreta que realiza cada parte del sistema a desarrollar. Para este caso, la nomenclatura será **RFUN** seguido del número del requisito.

RFUN - 1: Se usará MQTT para los intercambios de información entre máquina a máquina.

RFUN - 2: Se usará ubidots como herramienta integradora en la nube, usando la conexión necesaria con MQTT.

RFUN - 3: Se usarán esp32 como el dispositivo base de comunicación y donde se alojarán los sensores.

RFUN - 4: El sistema recogerá información del sensor cada minuto, para cada tipo de sensor.

RFUN - 5: El sistema contará mínimo con 3 dispositivos con sensores y una mínima cantidad de dispositivos en el fog que permitan el procesamiento de los datos..

6.3. Requisitos no funcionales

Los **requisitos no funcionales** hacen referencia a aquellos requisitos que no tienen que verse implicados con el funcionamiento general del sistema, ni de cada una de sus partes o de la información que maneja el sistema. Para este caso, la nomenclatura será **RNFUN** seguido del número del requisito.

RNFUN - 1: El sistema debería en la medida de lo posible enviar los datos con cierta sincronía a la nube.

RNFUN - 2: En ubidots, se debería apreciar de forma sencilla e intuitiva los valores de los sensores escogidos.

RNFUN - 3: La estructura del sistema debe permitir que a futuro se pueda aplicar un sistema basado en reglas para modificar su estructura.

6.4. Requisitos de información

Por último, se comentan los **requisitos de información**, estos hacen referencia a la información que manejara el sistema que se pretende desarrollar. Estos requisitos especificarán qué tipo de información necesitará el sistema para funcionar correctamente. Para este caso, la nomenclatura será **RINF** seguido del número del requisito.

RINF - 1: De los sensores se extraerán los datos en bruto, como por ejemplo temperatura, etc.

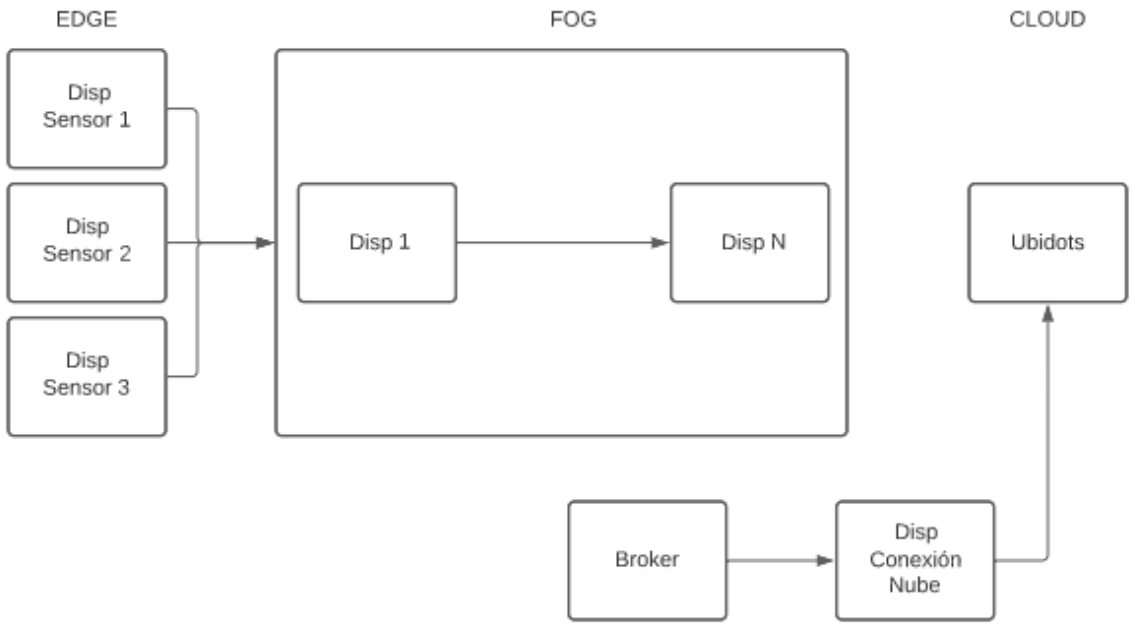
RINF - 2: Durante el procesamiento en el fog, estos datos en bruto se le aplicará un procesamiento donde obtendremos otros datos, como por ejemplo la media de los 50 datos.

RINF - 3: Los datos resultantes se podrán observar en el dashboard de ubidots de forma sencilla.

7. Diseño:

Tras especificar los requisitos que el sistema debe satisfacer, se puede empezar a realizar la etapa de diseño. Para ello, se explicará el diseño general del sistema y se especificará el diseño específico de ciertos apartados que se consideran de relevancia.

Para mayor entendimiento del diseño en general, se creó el siguiente esquema, donde se puede apreciar la estructura general que se pretende implementar de manera simulada (y a posterior en físico)



Como se puede apreciar en el diagrama, nos encontramos los siguiente 3 apartados generales:

- **Edge o borde.** En este segmento se encontrarán los dispositivos, simulados o reales, teniendo estos sensores que enviarán información a las etapas posteriores. Se limitan exclusivamente a recolectar y enviar información cada cierto tiempo a etapas posteriores.

- **Fog o niebla.** Este segmento, realiza el procesamiento de forma distribuida para evitar así que tanto los dispositivos en el edge o en cloud sean demasiado costosos por su procesamiento.
  Para ello, en este segmento, encontraremos tantos dispositivos como sean necesarios para distribuir el procesamiento de tal forma, que estos no necesiten demasiado cómputo y evitar aumentar el coste de los mismos.

- **Cloud o nube.** En este segmento se recolectan todos los datos procesados en el segmento de Fog y se realiza la conexión con la nube. En algún punto es necesario la unificación de todos los datos, por ello se realiza en el borde del sistema, justo cuando se envía la información a la nube (por si fuera necesario añadir algún protocolo de seguridad de los datos hacia el exterior).

Cabe destacar que se podrían realizar ciertas especificaciones de diseño al servidor de mosquito, pero por mayor comodidad y sencillez, se utilizará con su configuración por defecto. Se podría modificar a través del archivo *configuration.conf.*

Por último, cabe destacar que se utilizará como plataforma en la nube ubidots y una vez se reciban los datos y se realice un pequeño procesamiento para su mejor entendimiento, se realizará un dashboard. Dicho dashboard, tendrá una estructura sencilla donde se mostrarán distintas gráficas con información de los datos relevantes, y se da cierta libertad en el diseño para añadir otras opciones que se consideren relevantes o interesantes para añadir al proyecto.

8. Implementación:

En este apartado, se comentará la implementación realizada sobre el diseño explicado en el apartado anterior, en concreto sobre la implementación simulada.

8.1. Implementación simulada.

En este apartado se explicará detalladamente todos los pasos realizados para llevar a cabo la simulación del sistema que se planteó en el apartado de diseño.

8.1.1. Servidor MQTT.

Para poder trabajar con mosquito, es necesario tener un servidor que haga de broker, para ello se usó el ordenador de uso personal de Antonio Gómez para dicho propósito. El sistema operativo utilizado es ubuntu 20.04 y los pasos para su instalación fueron los siguientes:

- sudo apt-add-repository ppa:mosquitto-dev/mosquitto-ppa
- sudo apt-get update
- sudo apt-get install mosquitto
- sudo apt-get install mosquitto-clients
- sudo apt clean

Como se utilizó la configuración por defecto, ya se podía utilizar el servidor y arrancarlo o pararlo según se desee. Los comandos para parar y arrancar mqtt broker son los siguientes:

```
antoniogg@antoniogg-SATELLITE-L50D-C:~/Escritorio/iot/TrabajoFinal/SimulacionFin
al$ sudo service mosquitto stop
antoniogg@antoniogg-SATELLITE-L50D-C:~/Escritorio/iot/TrabajoFinal/SimulacionFin
al$ sudo mosquitto -v -c new_mosquitto.conf
```

8.1.2. Scripts python Edge.

Para poder simular los dispositivos, se utilizaron hilos que lo simulaban, siendo estos scripts de python (simulan el código de micropython usado en dispositivos reales). Para este segmento, se crearon tres script similares que consisten en la generación de números aleatorios y el mensaje de estos a etapas posteriores. Estos datos son la temperatura, la humedad del suelo y la humedad del aire. Para los dos tipos de humedades si son datos completamente aleatorios, pero para la temperatura se usó un código proporcionado por Fernando León García que simula una tendencia, para apreciar mejor un caso más real.

Por lo tanto, cada script almacena 60 datos y cuando dicho buffer está lleno, envía el paquete de datos a la siguiente etapa. Esto simula, que cada minuto se almacena un dato, y cuando pasa una hora, se empiezan a enviar datos a las siguiente etapas. Por comodidad, los primeros 60 datos se simulan directamente.

Cuando el buffer se encuentra completo, este se comporta como una cola, de tal forma que si entra un nuevo dato, el dato con mayor longevidad desaparece y se envía de nuevo al buffer. Esto se realiza para añadir mayor sobrecarga al sistema.

Se muestra el siguiente pequeño ejemplo del código del dispositivo 1.

```
92      #si el tamaño del buffer es 60 elimina el primero que entro
93      if len(buffer) >= 60:
94          buffer.pop(0)
95          buffer.append(round(generar_asintota(buffer, v_final), 2))
96          data_out=json.dumps(buffer)
97          conexion_cliente.publish("/DATA/OUT/TEMPERATURA/STAGE0/VALUE", data_out)
98          sleep(60)
99      else:
100         buffer.append(round(generar_asintota(buffer, v_final), 2))
```

Como podemos observar el dispositivo 1 publica sus datos en el topic:
-> /DATA/OUT/TEMPERATURA/STAGE0/VALUE

Cabe destacar que para el dispositivo 2 (humedad del aire) y dispositivo 3 (humedad suelo) serían de la siguiente forma:
->/DATA/OUT/HUMEDAD_AIRE/STAGE0/VALUE
->/DATA/OUT/HUMEDAD_SUELO/STAGE0/VALUE

Por lo tanto, si quisiéramos añadir un nuevo dispositivo sensor, se publicaría con la estructura anterior de topic pero cambiando el nombre del dispositivo.

8.1.3. Scripts python Fog.

Igual que en el caso anterior, para poder simular los dispositivos, se utilizaron hilos que lo simulaban, siendo estos scripts de python (simulan el código de micropython usado en dispositivos reales). Para este segmento, se crearon tres script similares para cada procesamiento, de tal forma que cada uno consiste en un pequeño procesamiento de los datos de etapas anteriores, y en el envío a etapas posteriores.

En el segmento de Fog encontramos 3 etapas, para cada conjunto de datos de temperatura, la humedad del suelo y la humedad del aire, se realiza la media, la varianza y la desviación típica.

Para mayor entendimiento, se realiza un ejemplo para los datos de temperatura (para los dos restantes sería el mismo proceso).

Tras recibir los datos del Edge (por suscripción al topic), se realizaría la media de los datos, y se enviaría un paquete con todos los datos recibidos de la etapa anterior pero añadiendo al conjunto de datos la media. El código sería el siguiente:

```
89              print (f"Dato recibido: {mensaje.topic}={mensaje.payload}")
90              data_in=json.loads(mensaje.payload)
91              #procesamiento de este dispositivo
92              media = 0.0
93              for n in data_in:
94                  media += n
95              media=media/len(data_in)
96              data_in.append(round(media, 2))
97              data_out=json.dumps(data_in)
98              # fin procesamiento de este dispositivo
99              conexion_cliente.publish("/DATA/OUT/TEMPERATURA/STAGE1/VALUE", data_out)
100
```

El topic tendría la siguiente estructura:
->/DATA/OUT/TEMPERATURA/STAGE1/VALUE

La siguiente etapa se suscribió el topic anterior, y realizaría la varianza de los datos recibidos, basándose en los datos y la media del paquete recibido. Tras realizar el procesamiento, como para la desviación típica los datos no son necesarios, estos se eliminan para eliminar la carga de datos, aliviando la carga de red. Por lo tanto, solo se enviarán al nuevo topic la media y la varianza. El código es el siguiente:

```
90              print (f"Dato recibido: {mensaje.topic}={mensaje.payload}")
91              data_in=json.loads(mensaje.payload)
92              #procesamiento de este dispositivo
93              n = len(data_in) -1
94              media = data_in[n]
95              varianza = 0
96              data_in.pop(n)
97              for dato in data_in:
98                  varianza += math.pow((dato - media), 2)
99              varianza = varianza/(n-1)
100
101             data_in.clear()
102             print (media)
103             data_in.append(media)
104             print (round(varianza, 2))
105             data_in.append(round(varianza, 2))
106             print(data_in)
107             data_out=json.dumps(data_in)
108             # fin procesamiento de este dispositivo
109             conexion_cliente.publish("/DATA/OUT/TEMPERATURA/STAGE2/VALUE", data_out)
```

El topic tendría la siguiente estructura:
->/DATA/OUT/TEMPERATURA/STAGE2/VALUE

Por último, en la última etapa de este segmento, este dispositivo simulado se suscribió el topic anterior, y realizará la desviación típica de los datos recibidos, basándose en la varianza del paquete recibido. Tras realizar el procesamiento, se enviarán al nuevo topic la media, la varianza y la desviación típica de los datos recibidos del segmento de Edge. El código es el siguiente:

```
90              print (f"Dato recibido: {mensaje.topic}={mensaje.payload}")
91              data_in=json.loads(mensaje.payload)
92              #procesamiento de este dispositivo
93              desv_tipica = data_in[len(data_in) -1]
94              desv_tipica = math.sqrt(desv_tipica)
95
96              data_in.append(round(desv_tipica, 2))
97              data_out=json.dumps(data_in)
98              # fin procesamiento de este dispositivo
99              conexion_cliente.publish("/DATA/OUT/TEMPERATURA/STAGE3/VALUE", data_out)
```

El topic tendría la siguiente estructura:
->/DATA/OUT/TEMPERATURA/STAGE3/VALUE

Para los otros dos sensores el proceso es similar, pero con la variación de los topic donde se publican y suscriben, cambiando el nombre de TEMPERATURA por la media que se esté usando (HUMEDAD_AIRE o HUMEDAD_SUELO).

8.1.4. Script python Conexión a la nube.

Igual que en el caso anterior, para poder simular el dispositivo, se utiliza un hilo que lo simula, siendo este un script de python (que simula el código de micropython usado en dispositivos reales). Para este segmento final, se creará un script, de tal forma que se suscribe a todos los topic publicados en la etapa final del segmento de Fog, y a su vez, realiza una conexión con el broker de ubidots para poder publicar en un topic en la nube.

Respecto a la suscripción de todos los topic publicados en la etapa final del segmento de Fog, se puede apreciar mejor con el siguiente código:

```
147  def procesar_eventos():
148      global conexion_cliente, lock, conectado, mensaje
149      while True:
150          lock.acquire()
151          if evento == CONEXION_FALLIDA:
152              print ("Conexion Fallida")
153          elif evento == CONEXION_ESTABLECIDA:
154              conectado = True
155              #conexion_cliente.subscribe("temperatura")
156              MQTT_TOPIC = [("/DATA/OUT/TEMPERATURA/STAGE3/VALUE",0),("/DATA/OUT/HUMEDAD_AIRE/STAGE3/VALUE",0),
157              ("/DATA/OUT/HUMEDAD_SUELO/STAGE3/VALUE",0)]
158              conexion_cliente.subscribe(MQTT_TOPIC)
159              print("Conexion Establecida")
160          elif evento == PUBLICACION_REALIZADA:
161              print ("Publicacion realizada")
162          elif evento == SUSCRIPCION_ACTIVA:
163              print ("Suscripción activa")
164          elif evento == DATO_RECIBIDO:
165              print (f"Dato recibido: {mensaje.topic}={mensaje.payload}")
166              data_in=json.loads(mensaje.payload)
167              #procesamiento de este dispositivo
168
169              if mensaje.topic == "/DATA/OUT/TEMPERATURA/STAGE3/VALUE":
170
171                  media = data_in[len(data_in) -3]
172                  varianza = data_in[len(data_in) -2]
173                  desv_tipica = data_in[len(data_in) -1]
174                  Dict = {
175                  'TEMPERATURA_MEDIA': media,
176                  'TEMPERATURA_VARIANZA': varianza,
177                  'TEMPERATURA_DESVTIPICA': desv_tipica
178                  }
```

```
            elif mensaje.topic == "/DATA/OUT/HUMEDAD_AIRE/STAGE3/VALUE":

                media = data_in[len(data_in) -3]
                varianza = data_in[len(data_in) -2]
                desv_tipica = data_in[len(data_in) -1]
                Dict = {
                'HUMEDAD_AIRE_MEDIA': media,
                'HUMEDAD_AIRE_VARIANZA': varianza,
                'HUMEDAD_AIRE_DESVTIPICA': desv_tipica
                }

            elif mensaje.topic == "/DATA/OUT/HUMEDAD_SUELO/STAGE3/VALUE":

                media = data_in[len(data_in) -3]
                varianza = data_in[len(data_in) -2]
                desv_tipica = data_in[len(data_in) -1]
                Dict = {
                'HUMEDAD_SUELO_MEDIA': media,
                'HUMEDAD_SUELO_VARIANZA': varianza,
                'HUMEDAD_SUELO_DESVTIPICA': desv_tipica
                }

            # fin procesamiento de este dispositivo
            main(mqtt_ubidots_client, Dict)
            #conexion_cliente.publish("/DATA/OUT/STAGEFINAL/VALUE", data_out)
```

Como podemos apreciar en las imágenes anteriores, en la primera se realiza la suscripción a los distintos tópicos, y entre la primera y la segunda, cómo se almacenan los datos de cada tipo para su posterior subida a la nube.

Respecto a la conexión con el broker de ubidots para poder publicar en un topic en la nube, se puede apreciar en la siguiente imagen mejor como se realizaría este proceso:

```
BROKER_ENDPOINT = "industrial.api.ubidots.com"
TLS_PORT = 1883  # Secure port
MQTT_USERNAME = "BBFF-m54xl604oKGS6Z3lMqNVGrXLfkENIJ"  # Put here your Ubidots TOKEN
MQTT_PASSWORD = "BBFF-m54xl604oKGS6Z3lMqNVGrXLfkENIJ"  # Leave this in blank
TOPIC = "/v1.6/devices/iot-huerto"

def on_connect_ubidots(client, userdata, flags, rc):
    global conectado_ubidots  # Use global variable
    if rc == 0:

        print("[INFO] Conectando a ubidots")
        conectado_ubidots = True
    else:
        print("[INFO] Error, conexion fallida")


def on_publish_ubidots(client, userdata, result):
    print("Publicado correctamente")

def connect(mqtt_ubidots_client, mqtt_username, mqtt_password, broker_endpoint, port):
    global conectado_ubidots

    if not conectado_ubidots:
        mqtt_ubidots_client.username_pw_set(mqtt_username, password=mqtt_password)
        #eventos de mqtt para ubidots
        mqtt_ubidots_client.on_connect = on_connect_ubidots
        mqtt_ubidots_client.on_publish = on_publish_ubidots
        mqtt_ubidots_client.connect(broker_endpoint, port=port)
        mqtt_ubidots_client.loop_start()
```

8.1.5. Script auxiliar.

Este es un script sencillo que se creó por comodidad, para poder lanzar todos los archivos.py del sistema en terminales, para evitar tener que lanzar cada script de python a mano en una nueva terminal.

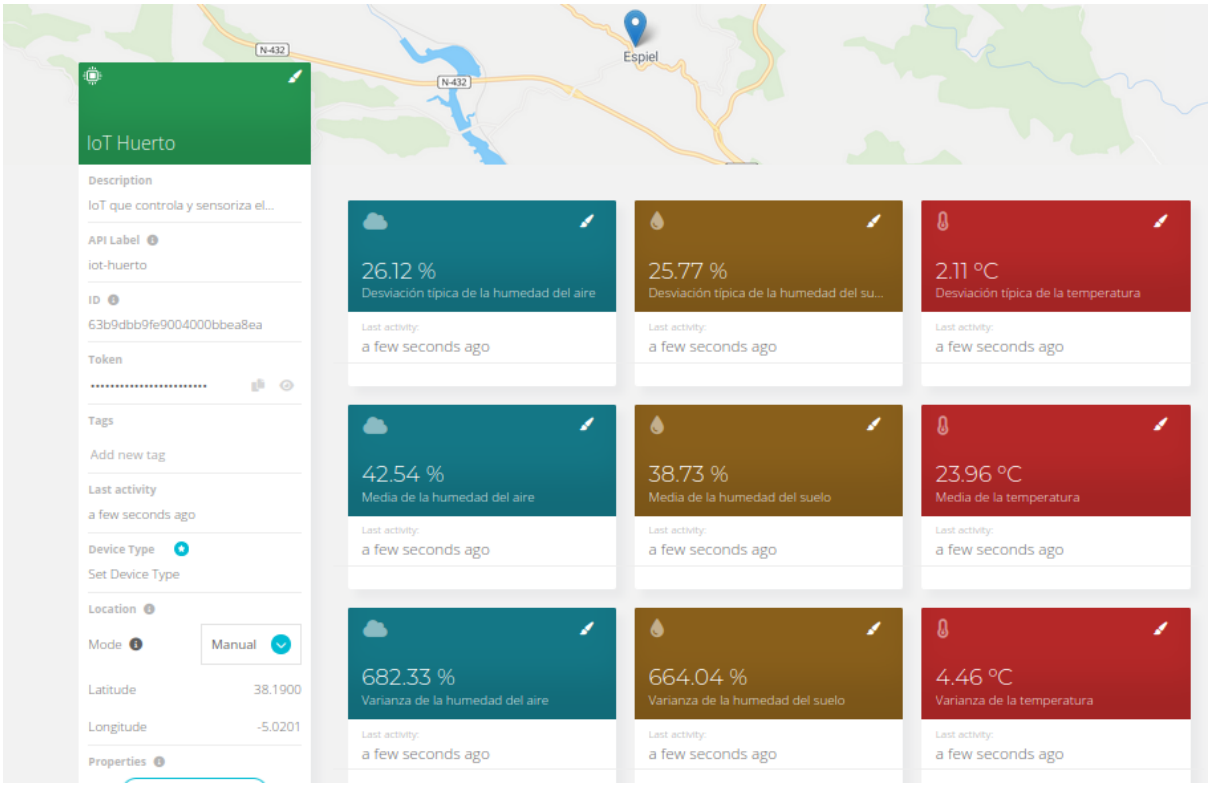
El código es el siguiente:

```
#!/bin/bash

#for py_file in $(find *.py)
#do
    #echo "$py_file"
    #gnome-terminal --execute python3.8  "$py_file"

#done
gnome-terminal --execute python3.8 DispStage1Sensor1.py
sleep 1
gnome-terminal --execute python3.8 DispStage1Sensor2.py
sleep 1
gnome-terminal --execute python3.8 DispStage1Sensor3.py
sleep 1
gnome-terminal --execute python3.8 DispStage2Sensor1.py
sleep 1
gnome-terminal --execute python3.8 DispStage2Sensor2.py
sleep 1
gnome-terminal --execute python3.8 DispStage2Sensor3.py
sleep 1
gnome-terminal --execute python3.8 DispStage3Sensor1.py
sleep 1
gnome-terminal --execute python3.8 DispStage3Sensor2.py
sleep 1
gnome-terminal --execute python3.8 DispStage3Sensor3.py
sleep 1
gnome-terminal --execute python3.8 DispConexNube.py
sleep 1
gnome-terminal --execute python3.8 DispSensor1.py
sleep 1
gnome-terminal --execute python3.8 DispSensor2.py
sleep 1
gnome-terminal --execute python3.8 DispSensor3.py
```

9. Resultados:

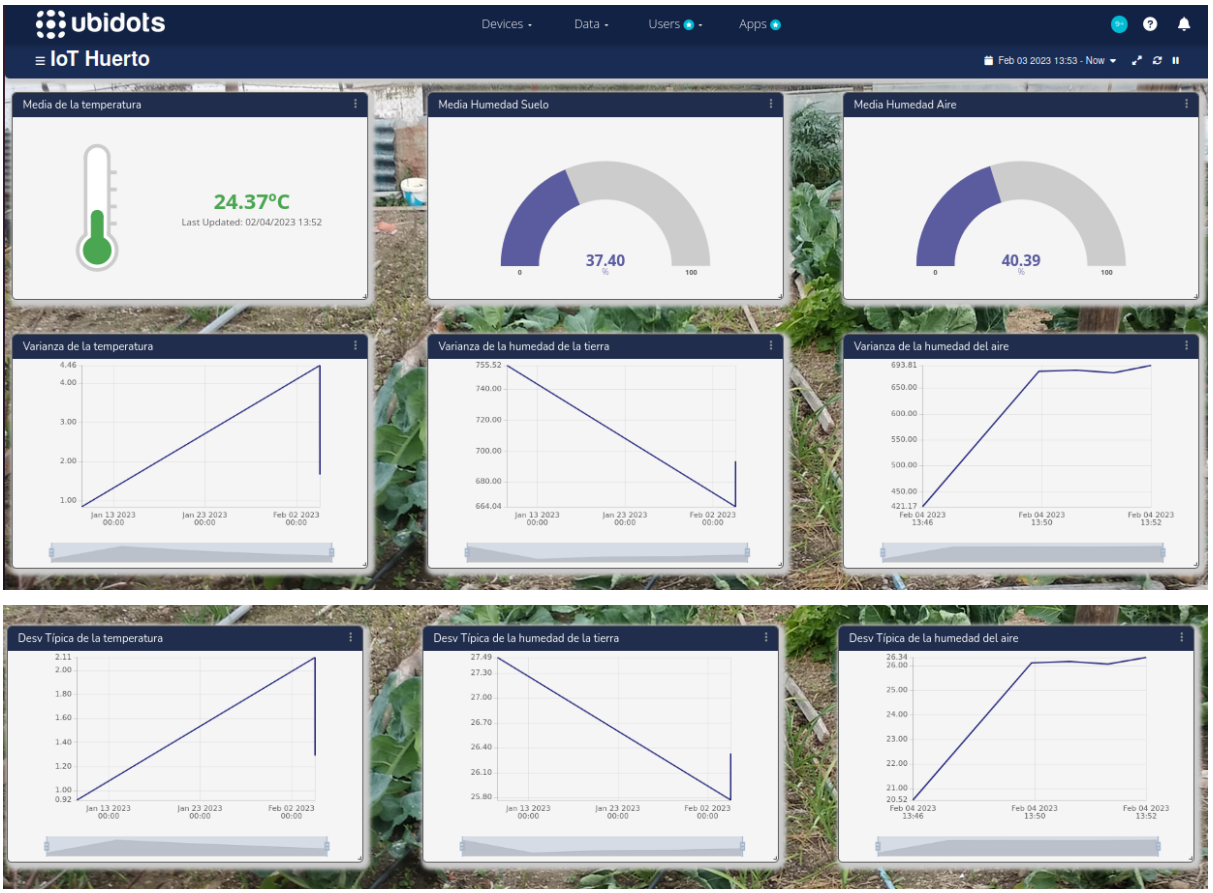
Tras realizar la implementación anteriormente explicada se probó a lanzar todos los archivos.py creados para comprobar su correcto funcionamiento y ver los resultados obtenidos. Para comprobar si se ha realizado correctamente, en ubidots deben aparecer los datos ya procesados. Y si todo funciona correctamente, se deberían poder ver las gráficas con los datos.

Los resultados obtenidos son los siguientes:



Como se puede observar, los datos llegan correctamente a ubidots, dando la media, varianza y desviación típica para los datos de temperatura, humedad del suelo y humedad de aire.

Solo falta comprobar el dashboard, los resultados son los siguientes:



Como podemos observar en las imágenes anteriores, el programa funciona correctamente. Cabe destacar, que en los gráficos se ven picos porque entre ejecución y ejecución hay un mes donde no se simularon los datos, por ende hay cambios bruscos en los mismos.

Respecto a la varianza excesiva que nos encontramos, se debe a que al ser datos aleatorios la varianza se dispara, por ello, en el caso de la temperatura, al no ser aleatoria, la varianza es mucho menor, hasta el punto de que si pasará mucho tiempo, esta sería 0 o casi 0.

10. Conclusiones y futuras mejoras:

Como se ha podido apreciar durante la documentación de este proyecto y más concretamente en el apartado anterior. Se podría decir que se han conseguido casi todos los objetivos marcados, de hecho, se van a revisar los objetivos de uno en uno para comprobar si se han cumplido.

En la siguiente tabla se muestra un resumen de los objetivos y, de si se han conseguido lograr, añadiendo unas observaciones de los mismos.

Objetivo	Consecución	Observaciones
OBJ-01	Sí	Se ha conseguido simular un sistema IoT orientado para la agricultura, teniendo en cuenta el concepto de Stream Processing en el diseño de la misma, y basado en Mosquitto para la interconexión entre dispositivos.
OBJ-02	Sí	Se ha conseguido montar un servidor Mosquitto que permite la conexión de distintos dispositivos que se encuentran en la misma red local (en el mismo pc).
OBJ-03	Sí	Se ha conseguido aplicar Stream Processing a un sencillo nivel como prueba.
OBJ-04	Parcialmente	Se ha conseguido crear un sistema en un entorno simulado para su posterior adaptación en campo. Pero no se consiguió implementarlo en un entorno real

OBJ-05	Sí	Se consiguió crear una conexión con ubidots para comprobar la correcta recepción de datos en una plataforma integradora en la nube.
OBJ-06	No	No se ha conseguido plantear cómo se realizaría el despliegue IoT a nivel de campo.
OBJ-07	Parcialmente	Se han conseguido comprender mejor los distintos problemas existentes al intentar desplegar un sistema IoT, solo a nivel de simulación.
OBJ-08	Sí	Se afianzaron los conocimientos adquiridos con Mosquito, aplicando el modelo de publicación y suscripción de topics.
OBJ-09	Sí	Se afianzaron los conocimientos adquiridos con Ubidots, creando la conexión con MQTT y la creación de un dashboard.
OBJ-10	Sí	Se ha conseguido aprender a crear un entorno simulado para el desarrollo de un entorno IoT.

Respecto al objetivo **OBJ-01**, se puede decir que se ha cumplido sin ningún problema, de hecho, se puede comprobar con el apartado "**7. Diseño**" y el apartado "**8. Implementación**", como se comenta de forma extensa todo este proceso.

Respecto al objetivo **OBJ-02**, se puede apreciar cómo se cumple si nos fijamos en el apartado "**8.1.1. Servidor MQTT**", donde se explica cómo se configura y se lanza el servidor de Mosquito.

Respecto al objetivo **OBJ-03**, podemos asegurar que se ha completado, ya que en el apartado "**8.1.3. Scripts python Fog**", se explica cómo se realizó esta distribución del procesamiento y cómo se aplicó este concepto.

Respecto al objetivo **OBJ-04**, podemos decir que se realizó parcialmente, ya que se consiguió implementar un sistema simulado pero, por falta de tiempo, y recursos, no se realizó en un entorno físico. Esto afectó directamente al **OBJ-06**, ya que al no realizarse finalmente a nivel de campo, se obvió este objetivo para centrarse en la mejora de otros objetivos como por ejemplo el objetivo **OBJ-09.** Cabe destacar que también influyó en el objetivo **OBJ-07**, ya que se vieron las dificultades a nivel de simulación pero no a nivel de campo, como pueden ser, las redes a usar, o el terreno donde se pretende trabajar y cómo proteger los dispositivos, o incluso como estos tienen energía durante el tiempo que se vayan a usar.

Finalmente, faltan por nombrar los objetivos **OBJ-05, OBJ-08 y OBJ-10,** donde el primer objetivo se consigue y se explica en el apartado "**8.1.4. Script python Conexión a la nube**" y los dos últimos, que se pueden apreciar como se han logrado a medida que se explica el diseño y la implementación.

Por lo tanto, podríamos llegar a la conclusión de que se ha conseguido llegar a lograr un sistema que aplica Stream Processing para IOT en la agricultura usando MQTT.

Aun así cabe destacar que el sistema no es perfecto y que tiene mejoras a futuro, como pueden ser:

- Mejora de cantidad de dispositivos.
- Procesamiento en fog más complejo.
- Sistema de control a futuro para cada dispositivo.
- Mejora del dashboard de ubidots y mayor cantidad de personalización.
- Añadir más cantidad de datos de sensores distintos.
- Intentar controlar la llegada de los datos, respecto al tiempo en el que dichos datos se obtuvieron
- Añadir capa de seguridad con la conexión en la nube.
- Etc.

O mejoras a nivel físico:

- Aplicar el sistema en un entorno real.
- Aplicar sistema de reserva de energía para los dispositivos.
- Crear un sistema de comunicación de los dispositivos entre sí y el exterior.
- Crear una estructura segura para preservar los dispositivos.
- Utilizar distintos tipos de sensores.
- Añadir actuadores.
- Etc.

Finalmente solo me falta hablar de las **conclusiones a nivel personal**, personalmente me encuentro bastante satisfecho con el trabajo conseguido, lógicamente soy consciente de que siempre se puede llegar a lograr un poco más, pero los objetivos personales impuestos por mí han sido cumplidos. Pudiendo aprender de un campo que no tenía experiencia previa (el campo de MQTT), aprendiendo también el uso de Ubidots y comprendiendo la relevancia y potencia que tiene el protocolo MQTT para el intercambio de información entre dispositivos IOT.

11. Bibliografía

[1] M. Abbasi, M. H. Yaghmaee and F. Rahnama, “Internet of Things in agriculture: A survey,” IEEE, pp. 17–28, 2019.

[2] M. S. Mekala and P. Viswanathan, “A Survey : Smart Agriculture IoT with Cloud Computing,” and IEEE, 2017.

[3] M. A. Ferrag, L. Shu, X. Yang, A. Derhab, and L. Maglaras, “Security and Privacy for Green IoT-Based Agriculture: Review, Blockchain Solutions, and Challenges,” IEEE Access, vol. 8, pp. 32 031–32 053, 2020.

[4] B. Acharya, K. Garikapati, A. Yarlagadda, and S. Dash, “Internet of things (IoT) and data analytics in smart agriculture: Benefits and challenges,” in AI, Edge and IoT-based Smart Agriculture. Elsevier, 2022, vol. 5, no. 5, pp. 3–16. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/B978012823694900013X

[5] L. J. MENA CAMARE, V. FELIX AVIÑA, and R. OSTOS ROBLES, “MARCO PARA AUTOMATIZAR LA SELECCIÓN DE UN MECANISMO DE COORDINACIÓN ENTRE SOCIEDADES DE AGENTES,” DYNA INGENIERIA E INDUSTRIA, vol. 92, no. 1, pp. 486–486, 2017. [Online]. Available: http://www.revistadyna.com/Articulos/Ficha.aspx?IdMenu=a5c9d895-28e0-4f92-b0c2-c0f86f2a940bCod=8396Idioma=es-ES

[6] D. Davcev, K. Mitreski, S. Trajkovic, V. Nikolovski and N. Koteli, “IoT agriculture system based on LoRaWAN,” IEEE, 2018.

[7] I. Mistry, S. Tanwar, S. Tyagi, and N. Kumar, “Blockchain for 5G-enabled IoT for industrial automation: A systematic review, solutions, and challenges,” Mechanical Systems and Signal Processing, vol. 135, p. 106382, 2020. [Online]. Available: https://doi.org/10.1016/j.ymssp.2019.106382

[8] B. Mishra and A. Kertesz, "The Use of MQTT in M2M and IoT Systems: A Survey," IEEE Access, vol. 8, pp. 201 071–201 086, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9247996/

[9] R. J. W. Housden, "On string concepts and their implementation," The Computer Journal, vol. 18, no. 2, pp. 150–156, feb 1975. [Online]. Available: https://academic.oup.com/comjnl/article-lookup/doi/10.1093/comjnl/18.2.150

[10] F. Braße, P. Collins, and M. Ziegler, "Computer Science for Continuous Data," Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 13366 LNCS, pp. 62–82, 2022. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85137006565doi=10.1007https://link.springer.com/10.1007/978-3-031-14788-35

[11] B. B. Sinha and R. Dhanalakshmi, "Recent advancements and challenges of Internet of Things in smart agriculture: A survey," Future Generation Computer Systems, vol. 126, pp. 169–184, jan 2022. [Online]. Available: https://doi.org/10.1016/j.future.2021.08.006 https://linkinghub.elsevier.com/retrieve/pii/S0167739X21003113

[12] T. Setiaji, C. Budiyanto, and R. A. Yuana, "The contribution of the Internet of Things and smart systems to agricultural practices: A survey," IOP Conference Series: Materials Science and Engineering, vol. 1098, no. 5, p. 052100, mar 2021. [Online]. Available: https://iopscience.iop.org/article/10.1088/1757-899X/1098/5/052100

[13] N. N. Misra, Y. Dixit, A. Al-Mallahi, M. S. Bhullar, R. Upadhyay, and A. Martynenko, "IoT, Big Data, and Artificial Intelligence in Agriculture and Food Industry," IEEE Internet of Things Journal, vol. 9, no. 9, pp. 6305–6324, may 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9103523/

[14] J. M. Talavera, L. E. Tobón, J. A. Gómez, M. A. Culman, J. M. Aranda, D. T. Parra, L. A. Quiroz, A. Hoyos, and L. E. Garreta, “Review of IoT applications in agro-industrial and environmental fields,” Computers and Electronics in Agriculture, vol. 142, no. 118, pp. 283–297, nov 2017. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0168169917304155

[15] A. Tzounis, N. Katsoulas, T. Bartzanas, and C. Kittas, “Internet of Things in agriculture, recent advances and future challenges,” Biosystems Engineering, vol. 164, pp. 31–48, 2017. [Online]. Available: https://doi.org/10.1016/j.biosystemseng.2017.09.007

[16] A. Kamilaris, F. Gao, F. X. Prenafeta-Boldu, and M. I. Ali, “Agri-IoT: A semantic framework for Internet of Things-enabled smart farming applications,” in 2016 IEEE 3rd World Forum on Internet of Things (WF-IoT). IEEE, dec 2016, pp. 442–447. [Online]. Available: http://ieeexplore.ieee.org/document/7845467/

Recursos bibliográficos sobre IoT (tecnologías como Lora o MQTT), todos ellos están en la librería digital de la Universidad de Córdoba: (https://ebookcentral.proquest.com/lib/bibliocordoba-ebooks/home.action):

[17] P. Waher, “Learning Internet of Things”, Packt Publishing, Limited, 2015

[18] S. Cirani, et al, “Internet of Things : Architectures, Protocols and Standards” John Wiley & Sons, Incorporated, 2018

[19] A. McEwen, and C. Hakim, “Designing the Internet of Things”, John Wiley & Sons, Incorporated, 2013

[20] N. Bouhaï, and I. Saleh, “Internet of Things : Evolutions and Innovations”, John Wiley & Sons, Incorporated, 2017

Anexos

Anexo I. Manual de código.

ANEXO I

MANUAL DE CÓDIGO

Anexo I. Manual de Código.

Se adjunta en el siguiente enlace el github donde se encuentra alojado todo el código del proyecto:

https://github.com/Witiza99/Stream-Processing-MQTT

Se añade también el enlace a ubidots, por si se quiere utilizar esta herramienta, para tratar información en la nube sobre los dispositivos iot.

https://ubidots.com/