Desarrollo

**EMQX**

**Creación de cuenta:**

Para poder hace uso de esta plataforma debemos de crear un cuenta, para ello debemos ir al siguiente link, <https://www.emqx.com/en/cloud>

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Estando en la pagina web, debemos dar clic el botón que decie “Start Free”, esto nos redirigirá a otra página donde podremos crear una nueva cuenta desde 0, o asociar un cuenta de Google o GitHub en su lugar.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Para simplificar el proceso, se usará la opción de asociar una cuenta de Google.

**Creación de un “Deploymenyt”**

Después de crear la cuenta, nos pedirá elegir un modo de despliegue o “Deployment”, que en este caso para fines educativos se eligira la opción gratuita

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Al seleccionar esto nos mandara a otra pagina de mostraran las características del plan del despliegue, y podremos crearlo o desplegarlo.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Después de crearlo, si queremos acceder a él debemos ir al apartado de “Deployments”, donde se podrán ver todos los despliegues que tenemos. Para el caso de desplegue gratuito unciamente se podrá crear uno.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

**Configuración del “Deployment”**

Una vez creado nuestro servicio MQTT debemos configurarlo para poder hacer uso de el. Al entrar en nuestro servicio lo primero que podemos ver es una descripción breve de este:

Interfaz de usuario gráfica, Aplicación, Sitio web

Descripción generada automáticamente

Lo destacable de esta vista es que aquí tenemos las características de la conexión, las cuales son:

* Dirección: URL con el cual podremos hacer uso del servicio.
* Puerto MQTT: número del puerto para acceder al servicio mediante MQTT
* Puerto WebSocket: número del puerto para acceder al servicio mediante WebSockets
* CA Certificate: opción de descargar de un certificado de identificación, que se usara más adelante para permitir el acceso al servicio.

Como primer paso para la configuración debemos crear un usuario con el cual podremos acceder al servicio, para ello vamos a apartado de “**Authentication & CLA**”, y después vamos a “**Authentication**”. Allí damos clic en el botón que dice “+ Add”, esto desplegara un formulario donde se debe ingresar un nombre y contraseña para el usuario. Para el caso de esta práctica el usuario será “mqtt”

Interfaz de usuario gráfica, Aplicación, Teams

Descripción generada automáticamente

Después debemos irnos al apartado de “**ACL**” y después donde dice “All users”, allí lo que haremos será crear un “**Topic**”. Un topic es algo similar a una sección o un canal de comunicación el cual se podrán crear mensaje. Se pueden tener varios canales o Topic para seccionar o separar funciones al crear implementaciones o separa aplicaciones, o se pueden crear sub-topics dentro de un topic para realizar lo mismo. Por ejemplo, para los mensajes que usaran “n” servicios dentro de una aplicación

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Para crear un Topic debemos dar clic en “+ Add”, esto desplegara un formulario donde debemos darle un nombre, que en este caso será “monitores /#”, debemos darle un tipo de acción, que en este caso será “Publish an Suscribe” y el tipo de permiso, que este caso será “Allow”.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto, Correo electrónico

Descripción generada automáticamente

El uso del “/#” al final del nombre es una forma de nombrar al Topic que nos permitirá usar o implementar “sub-topic” dentro de este. Nos permitirá acceder a estos de forma similar a los “endpoints” de HTTP. Por ejemplo, “monitore/servidor”, “monitores/esp32”, etc.

**ESP32**

El ESP32 el dispositivo que nos permitirá recolectar datos e interacciones del usuario, junto a ello nos permitirá enviar esta información a nuestro servidor de Python o a nuestra Web. Entre las funcionalidades que tiene están:

* Recolectar y enviar las lecturas de temperatura y humedad proveída por el sensor DHT11
* Almacenar el valor actual del contador, controlar su aumento y decremento en base a las veces que el usuario presioné los botones físicos, e enviar el valor de este a la página web cada que este sufra algún cambio
* Recibir el estado del led, enviado desde la página web, para así realizar la acción de encenderlo o apagarlo

**Explicación del codigo:**

Es esta primer sección del código de declaran las librerías que se usuran para el programa:

* WiFi: nos permitirá poder acceder a una red Wi-FI
* PubSubClient: es la librería que nos permitirá acceder al servicio MQTT que habilitamos en EMQX
* WiFiClienteSecure: librería que en conjunto con “PubSubClient” permite un acceso seguro mediante certifidos al servicio de MQTT.
* ArduinoJson: librería que nos permite trabajar con “JSON”, funcionalidad que usara para el envia y recepción de mensajes
* DHT: librería usada para acceder a las funcionalidades del DHT11

**Texto

Descripción generada automáticamente**

En la siguiente sección de código se definen e inicializan ciertas variables necesarios para la funcionalidad del código. Por una parte, se definen dos variables que identificar el pin al que se conecta el DHT11 y el tipo de DHT que estaremos usando, posteriormente estas dos variables se usan para crear un objeto que permitirá acceder a las funcionalidades de la librería DHT.

Después definimos tres variables para establecer los pines donde se conectara el led, y los botones para el aumente y decremento del contador. Por ultimo, creamos una variable entera para guardar el valor del contador, la cual comenzara en 0.

Imagen que contiene Forma

Descripción generada automáticamente

Seguidamente creamos dos variables que contendrán el nombre y contraseña de la red Wi-Fi que usaremos para conectarnos a internet.

Texto

Descripción generada automáticamente

Posteriormente creamos las variables que usaran para hacer uso del servicio de MQTT.

* mqtt\_broker: guarda el URL con el cual se acceder al servicio
* topic: guarda el topico al cual nos suscribiremos para enviar y/o recibir mensajes
* mqtt\_username: guarda el nombre del usuario que creamos para el servicio
* mqtt\_password: guarda la contraseña del usuario que puede usar el servicio
* mqtt\_port: guarda el numero de puerto por cual se acceder al servicio
* ca\_cert: guarda el valor String del certificado que usara para conectarnos al servicio. (Este certificado se puede descargar desde EMQX).

Texto

Descripción generada automáticamente

Por último, se crear dos variables que se usaran para crear la conexión al servicio y realizar la publicación o recepción de mensajes.

Texto

Descripción generada automáticamente

Terminando la de claración de variables lo que sigue es configurar todo aquello que debe inicializarle al enceder el ESP32, para ello se usa la funcipon “setup”.

En esta primera sección del “setup” se encarga de iniciar la comunicación serial, inicializar el objeto que usa para recibir las lectura del DHT11, y después se realiza la conexión a la red Wi-Fi con las credenciales establecidas previamente.

Texto

Descripción generada automáticamente

Después, se comienza a establecer la conección al servicio MQTT, para ello primero establecer cual es el certificado que usaremos, después establecer cual será el URL y puerto de coneción y estableces una acción callback (una función que se ejecutara cada vez que se reciba un mensaje en el topico al que estemos suscrito).

Seguido de los anterior, después de establecer los elementos de conexión se se realiza un ciclo while que estará ciclado hasta conectarnos al servicio. Una vez que nos conectemos enviaremos un mensaje al topic que establecimos previamente y nos suscribiremos a este para poder escuchar/recibir los mensajes que se envíen dentro de este.

Texto

Descripción generada automáticamente

Por último, establecemos el modo en que trabajaran los pines de los botones y del led. El led será un pin de salida, y los pines de los botones serán de entrada.

Texto

Descripción generada automáticamente

Después de definir la función “setup”, sigue la función “callback” que la que manejara las acciones a realizar cuando se reciba un mensaje dentro de topic al que nos suscribimos. Aquí la única función relevante es la siguiente:

Esta se encarga de recibir el mensaje y deserializarlo como un JSON, en caso de que surja un error entonces significa que el mensaje recibido no tiene el formato adecuado, si esto pasa se manda un mensaje de debug y se termina el método.

En caso de tener el formato adecuado, verificamos que el mensaje sea para la ESP32 y vemos si el mensaje tiene involucra la acción del “led”, sino es así la función termina. En caso de que si cumpla con la condición recuperamos el valor del estado del led, si este valor el 1 lo encendemos, si el valor es 0 lo apagamos, y en caso de no coincida con ninguno mandamos un mensaje de debug que dice que el valor no esta entre los valores esperados.

Texto

Descripción generada automáticamente

La siguiente función “reconnect” se trata de una función que usara en caso de que se pierda la conexión con servicio, la en forma resumida se trata un ciclo que se realizara una y otra vez hasta establecer una conexión de forma correcta.

Texto

Descripción generada automáticamente

Después de la función anterior se definen dos variables que usaran dentro del loop, estas variables se usaran para controlar la frecuencia del envio de las lecturas del sensor de humedad y temperatura.



Como ultimo función, tenemos la función “loop” la realiza las acciones de lecturas de sensores, actualización del contador y el envio de estos parámetros a través del servicio.

Primeramente, dentro de del loop tenemos un código de verificar si estamos conectados al servicio, sino se esta conectado se ejecuta la función “reconnect” para recuperar la conexión. Si estamos conectados llamamos al método “loop” de cliente, que nos permitirá escuchar constamente en la espera de recibir mensajes dentro del topic.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Después sigue el código que manejara la interacción con led, si detecta que pin “ascPin” este en HIGH (fue presionado) entonces incrementamos el valor del contador en 1 y mandamos un mensaje JSON a nuestro tópico, dicho mensaje indica quien lo envía “from: esp32”, quien debe recibirlo “to: broadcast” (para todos), la acción a realizar “UPDATE\_COUNTER” (actualizar el valor del contador), establecemos cual el valor actual “value: contador”, y publicamos este mensaje

Texto

Descripción generada automáticamente

En caso de el pin “desPin” este en HIGH (fue precioado), entonces decrementamos el valor de contador en 1, y mandamos un mensaje JSON al topico con el mismo formato que en el código anterior.

Texto

Descripción generada automáticamente

Por último, dentro de loop establecemos la forma en la que se realizara la lectura de la temperatura. Primero verificamos si paso el lapso necesario de tiempo (30 segundos), si esto se cumple lo primero que se realiza es la lectura de la temperatura y humedad, posteriormente verificamos que ambos valores sean valores numéricos válidos. De ser así creamos un JSON, este indicara quien lo envía “from: esp32”, quien debe recibirlo “to: server” (nuestro servidor), la acción a realizar “SEND\_DATA” (envió de valores), establecemos los valores de temperatura y humedad que enviaran, después publicamos este mensaje.

Justo después de enviar este mensaje, actualizamos el valor de “to” por “web” en lugar de “server” y volvemos a enviar este mensaje.

Esto se realiza porque el primer mensaje que se envía es para que el servidor almacene el valor dentro de nuestra base datos, y el segundo es para que nuestra web muestre el valor actual de temperatura y humedad en el entorno gráfico.

Texto

Descripción generada automáticamente

**Servidor de Python**

**AQUÍ VA LA EXPLICACIÓN DEL SERVIDOR**

**Página web**

**Escala de tiempo

Descripción generada automáticamente**

Esta pagina web se encarga de ser la interfaz grafica que permite que cualquier usuario pueda interactuar con nuestro servicio. Las acciones que puede realizar son:

* Encender/apagar el led
* Ver una grafica que contiene los valores de la temperatura y humedad en la ultima hora
* El valor actual de la temperatura
* El valor actual de la humedad
* El valor actual del contador

Esta pagina funciona a partir del siguiente código:

Este primer fragmento de código se encarga de establecer los parametros necesarios para mostrar los valores de humedad y temperatura en formato grafico, primero importamos el cdn de “chart.js” la librería que nos permite dibujar y trabajar con la grafica. Después establecemos los valores predeterminados que tendrá la grafica, entre ellos están:

* Los labels o etiquetas de tiempos
* Los datasets, que son los valores que se mostraran en la grafica, en este caso, los valores de humedad y temperatura.

Por ultimo, establecemos un función que nos permitirá actualizar los valores de las etiquetas de tiempos y valores de temperatura y humedad cada vez que esta sea llamada.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamentePosteriormente el siguiente fragmento de código se usar para establecer la conexión con el servicio MQTT, para ello primero importamos la librería “mqtt”, librería que nos permitirá realizar dicha conexión.

Después creamos una variable para establecer el Topic que se usara para escuchar y recibir mensajes y la URL que usaremos para la conexión. Seguidamente creamos un objeto llamados “options” que tendrá, entre otras cosas, el nombre de usuario y contraseña que puede usar el servicio y un ID para identificarlo. Teniendo este objeto pasamos a crear una variable que almacenara la conexión con el servicio, dicha variable es “client”.

Seguidamente establecemos la acción o función que se realziara al establecer la conexión, que en este caso únicamente se mandara un mensaje que diga “Kirby Web”.

Texto

Descripción generada automáticamenteDespués se estable la acción o función que se realizara cuando se reciba un mensaje.

Al recibir un mensaje primero trataremos de convertir ese mensaje a un JSON, si esto no es posible desplegamos un mensaje en consola para señalar que no se recibió el formato adecuado.

Si el mensaje tiene el formato adecuado lo primero que se hará será verificar si el mensaje proviene de la ESP32, si va dirigido a la nuestra Web, y si la acción es “SEND\_DATA”, de ser así actualizamos los valores de la humedad y temperatura en nuestra interfaz.

Si por otro lado, el mensaje viene del servidor y esta dirigido a la nuestra web, entonces mandamos llamar a la función “updateChart” y le enviamos los nuevos valores de que tendrá la grafica.

Por ultimo, si el mensaje proviene de la ESP32, esta dirigido a “broadcast” y la acción es “UPDATE\_COUNTER”, entonces actualizamos el valor del contador que se muestra en la interfaz.

Después, en el siguiente fragmento de código, se crean dos variables que hacen referencia a los botones usados para encender y apagar el led.

Texto

Descripción generada automáticamenteAl tener esta variables lo primero que hacemos es agregar un “eventListener” para la acción clic del botón de encendido, cuando este botón se precione se realizaran unas acciones que cambiaran el estilo del pagina y se enviara un mensaje a JSON el cual indica que lo envia la web “from: web”, que va dirigido a la ESP32 “to: esp32”, que la acción corresponde al led “action: led”, y que el estado del led será encendido “data: { state: 1 }”.

Texto

Descripción generada automáticamente

Después se agregara un “eventListener” similar al anterior pero para el botón de apagado, cuya única diferencia será que el JSON enviado indicara que el estado del será apagado “data: { state: 0 }”.

Por ultimo, agregamos también agregarmos un “eventListener” para la acción clic del botón de “Solicitar datos”, que se encarga de enviar un mensaje al servidor para solicitar los datos de la base datos que corresponden a las lecturas de humedad y temperatura de la ultima hora.

Texto

Descripción generada automáticamente

**Conclusión:**

Cambiar de utilizar un servicio MQTT preconfigurado, como el proporcionado por Arduino Cloud, a la creación y configuración de un servicio desde cero resultó ser una tarea sumamente compleja y exigente. La configuración de este servicio implicó inicialmente la definición y establecimiento de reglas de comunicación entre diversos servicios, equivalente a la creación de un protocolo simple de comunicación. Además, conllevó la creación e implementación de varios servicios, incluida la configuración del servicio de la ESP32. También fue necesario crear un servidor en Python para asegurar la persistencia de los datos de temperatura y humedad. Finalmente, se desarrolló un servicio o página web que actuó como interfaz gráfica.

Comparando esta implementación con las otras dos realizadas a lo largo del semestre, podría considerarse como la segunda mejor. Aunque fue construida en gran medida desde cero, proporciona amplias capacidades de configuración y presenta tiempos de respuesta mejores en comparación con la implementación HTTP, siendo relativamente similares a los del Arduino Cloud. La única desventaja significativa radica en el incremento del nivel de complejidad que conlleva.

En términos generales, esta implementación se puede calificar como positiva. Desde el punto de vista del conocimiento, proporcionó una comprensión más profunda del funcionamiento de implementaciones reales de MQTT. Ofrece una gran versatilidad y, con la ayuda de brokers como EMQX, posibilita medios de comunicación rápidos y gratuitos para implementaciones de menor escala.