Proyecto DAD

Este proyecto, concebido en colaboración entre mi compañero Julio y yo, se origina en nuestra decidida aspiración por profundizar en el campo de la ciberseguridad, con el objetivo de aplicar nuestros conocimientos en entornos prácticos. Para ello, hemos desarrollado un prototipo que comprende una protoboard, diversos sensores y placas cuya funcionalidad específica detallaremos más adelante.

En consonancia con el nombre de la asignatura que abordamos, este proyecto se concibe con la capacidad inherente de ser implementado de manera distribuida. Nos propusimos encontrar un enfoque que permitiera la aplicabilidad del sistema de seguridad en escenarios reales, lo que implicaba la necesidad de un servidor para su gestión, así como la integración de múltiples placas y sensores, junto con actuadores. La inclusión de estos últimos es de vital importancia, dado que sin ellos el sistema carecería de la capacidad de respuesta necesaria para actuar ante situaciones críticas. En última instancia, un sistema de seguridad que no pueda prevenir incidentes o salvaguardar vidas resultaría insuficiente para nuestro propósito.

La premisa fundamental de nuestro diseño se centra en la instalación de sensores de calidad del aire, temperatura y un relé actuador que simula la apertura de puertas para facilitar la ventilación. Asimismo, hemos considerado la posibilidad de incorporar alarmas visuales mediante LEDs, lo que añadiría una capa adicional de alerta y notificación.

La implementación de este sistema a través de un servidor centralizado para la gestión de los datos adquiere una importancia significativa. En situaciones donde se detecte una calidad de aire perjudicial para el personal de la planta o un incremento alarmante de la temperatura, combinado con la presencia de humo detectado por el sensor correspondiente, el sistema activará los actuadores para abrir las puertas y facilitar la evacuación del aire contaminado y, en consecuencia, del personal de la planta.

Considerando el alcance de nuestro proyecto, podríamos explorar la integración de sistemas de alerta remota y la implementación de medidas de seguridad adicionales para fortalecer la integridad y la eficacia del sistema en su totalidad.

Después de la fase inicial de diseño, hemos optado por la utilización de dos placas de desarrollo ESP32 WROOM-32. Estas placas representan una solución integral para proyectos de Internet de las cosas (IoT), caracterizadas por su conectividad WiFi y Bluetooth de baja energía. Además de contar con un flash SPI integrado que proporciona un almacenamiento rápido y fiable, las placas ESP32 facilitan la comunicación inalámbrica directa a través de WiFi, lo que permite la transferencia de datos a largas distancias.

Cada placa está equipada, como mencionamos anteriormente, con dos sensores específicos. En primer lugar, utilizamos el sensor de humedad DHT11, un dispositivo compacto que proporciona mediciones precisas de la humedad relativa y la temperatura del aire. A pesar de su diseño de bajo costo, el DHT11 es altamente confiable y fácil de usar. Aunque su aplicación principal suele ser en el monitoreo ambiental y el control del clima, en nuestro caso, lo empleamos de manera diferente, como se discutió anteriormente. Su interfaz digital de un solo cable simplifica su integración con microcontroladores, lo que lo convierte en una opción popular para proyectos de IoT y automatización.

El otro sensor que hemos seleccionado para medir la calidad del aire es el MQ-135. Este dispositivo de detección de gases es altamente sensible y versátil, capaz de detectar una variedad de gases nocivos en el aire, como dióxido de carbono (CO2), monóxido de carbono (CO), amoníaco (NH3) y compuestos orgánicos volátiles (COVs). Diseñado para ofrecer precisión y confiabilidad, el MQ-135 se utiliza ampliamente en aplicaciones de calidad del aire interior y exterior, control de la contaminación, seguridad laboral y detección de fugas de gas. Su interfaz analógica permite una fácil integración con microcontroladores y sistemas de monitoreo, lo que lo convierte en una herramienta invaluable para proyectos de IoT y dispositivos de seguridad ambiental.

Empezaremos por ver como se ha desarrollado este proyecto mediante diferentes puntos que representaran los entregables en los que se han ido trabajando.

1. El primer entregable nos acerca a una breve introducción de como hoy en día gestionar webs.

   Los servlets en Java son componentes independientes de la plataforma que pueden ejecutarse en cualquier servidor compatible con su aplicación. Esta independencia proporciona flexibilidad tanto en el desarrollo como en la implementación. Además, los servlets son reutilizables, lo que permite a los desarrolladores encapsular la lógica de la aplicación en componentes fáciles de invocar. Esto conduce a la reutilización del código en diferentes proyectos, a la escalabilidad de las aplicaciones y al mantenimiento simplificado. Todos los servlets en Java se implementan mediante la interfaz ‘javax.servlet.Servlet’, que define los métodos necesarios para su funcionamiento y proporciona una estructura estandarizada para su desarrollo. Estos servlets manejan solicitudes HTTP, procesan parámetros de entrada y generan respuestas dinámicas, lo que permite crear aplicaciones web interactivas y dinámicas.

   A la hora de programar el servlet la única complicación que tuvimos fue que nos costo un poco adaptarnos a la nueva tecnología debido a que antes ya habíamos montado una pagina web pero nunca nos enfretamos a montar nuestra propia API pero se pudo llevar a cabo sin apenas complicaciones.

2. Empezamos a usar Vertx, un framework basado en eventos que corre en la máquina virtual de Java. Esta hecho para que sea fácil escalar y soporte la tolerancia a fallos.

   Primeramente, empezamos a tener como el entregable anterior unos datos en listas o mapas los cuales los usamos para poder comprobar que nuestras operaciones si funcionaban correctamente.

   Nos enfrentábamos a hacer funciones llamadas por endpoints, donde lo que haríamos es lo mismo que obtener los datos con un filtro cualquiera

o devolviendo todos los datos específicos de algo (en nuestro caso, por ejemplo, sería de todos los sensores de humedad o de todos los actuadores).

Este entregable no fue complicado debido a la ayuda de otros compañeros y la aportación de ejemplos del profesor.

3. En este momento fue la hora de dar el mayor paso de todos, conectar nuestro Vertx con una base de datos, aquí cambiaba todo, debido que a los endpoints ejecutarían código SQL para hacer las solicitudes GET, POST, DELETE.

   En este no hubo grandes complicaciones debido a que quisimos experimentar con una nueva herramienta de trabajo (XAMP) que funciona perfectamente para crear de primera mano las tablas que necesitábamos del actuador y del sensor sin tener que escribir mínimamente código, pero a la hora de gestionar la conexión con la XAMP tuvimos bastantes complicaciones, de forma que optamos por movernos a herramientas ya conocidas por nosotros como HeidiSQL o MySQL.

4. El desarrollo del firmware fue la entrega mas complicada de todas. Programar el ESP32 fue un reto debido a que era la primera vez que conectábamos una placa a la red wifi y seguidamente conectarla al servidor, si que nos facilitó mucho programar con las librerías de Arduino, y gracias a platformio no tuvimos mucha complicación de encontrar las librerías del DTH11 y a pesar de un pequeño error en la conexión de la fuente de alimentación del relé, pudimos comprobar con que el código funcionaba correctamente al solucionar dicho error.

5. MQTT es un protocolo de mensajería basado en estándares, o un conjunto de reglas, que se utiliza para la comunicación de un equipo a otro y Mosquitto es un broker MQTT ampliamente utilizado debido a su ligereza lo que nos permite, fácilmente, emplearlo en gran número de ambientes.

Este entregable esta intrínsicamente relacionado con los dos últimos, debido a que debemos de agregar código a en nuestro Vertx y nuestro firmware, en el Vertx establecemos la conexión en con MQTT y dentro del post del sensor, según un umbral establecido mandaremos con un topic un mensaje en el buffer, el cual será ON u OFF, dando a entender que el actuador se deberá de activar o no, dentro del firmware debemos de suscribirnos al topic y usar el buffer para el funcionamiento del actuador como hemos mencionado anteriormente.

A la hora de implementar MQTT y mosquitto tuvimos diferentes complicaciones que hoy en día no entendemos del todo porque se dan, pero para que pueda funcionar correctamente debemos de desactivar el cortafuegos de Windows en la red y usar a parte una red privada.

Gracias a este proyecto hemos aprendido un poco de como funciona un servicio web gestionado por nuestro servidor y como implementar varias placas como sistema distribuido mediante una pequeña columna en nuestra base de datos.