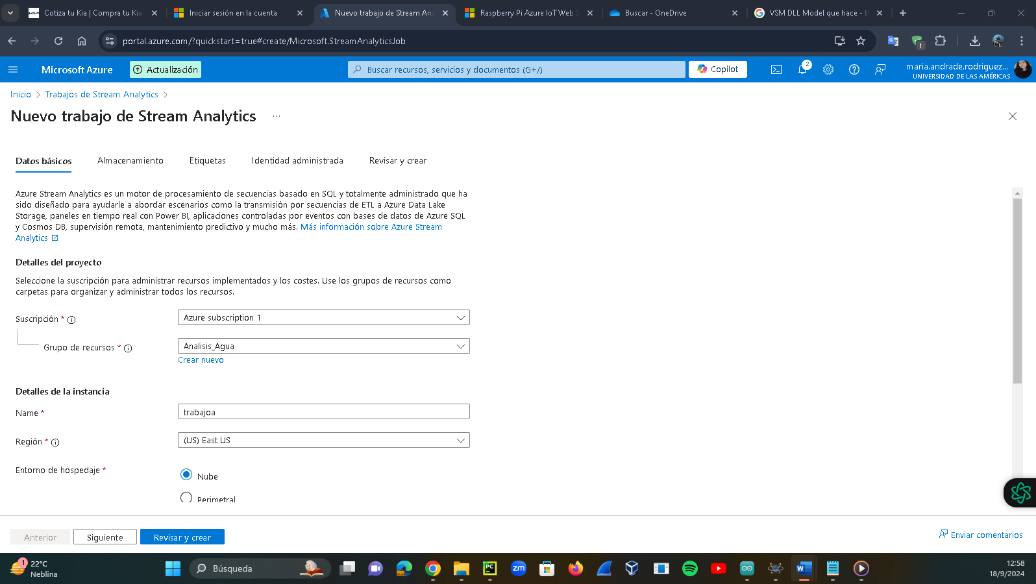
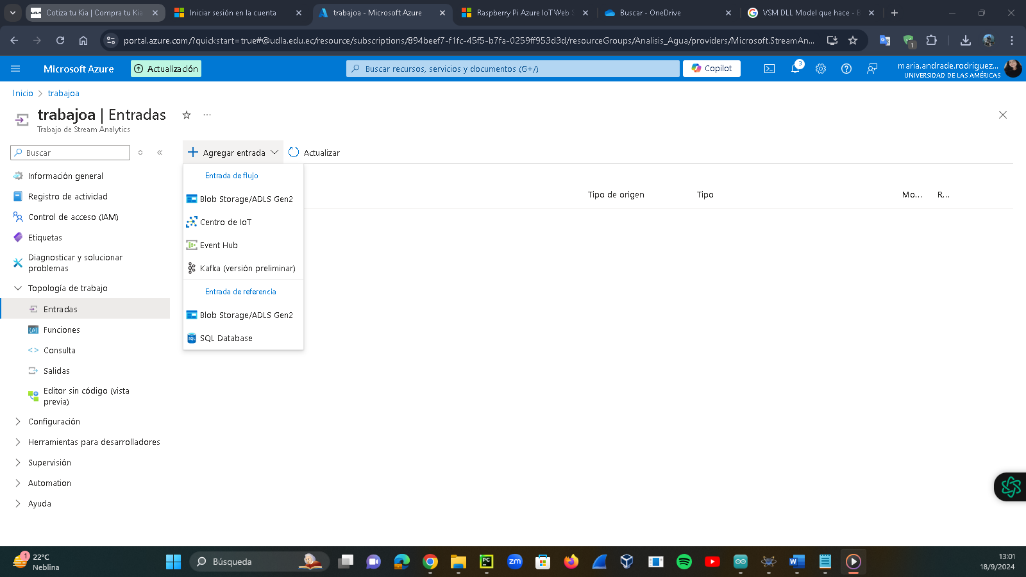


Interfaz de usuario gráfica, Texto

Descripción generada automáticamente





Simulación y propuesta en proteus

CODIGO DE PLACA ARDUINO EN PROTEUS (código\_placa) PARA CORRERLO EN ARDUINO IDLE

Tu código de Arduino parece estar bien estructurado para leer los datos de los sensores de pH y turbidez, y para tomar decisiones basadas en esas lecturas. A continuación, te explico cómo este código se integra con el script Python que actualizamos anteriormente para enviar datos a Azure IoT Hub.

**Resumen del Código Arduino**

1. **Lectura de Sensores**:
   * El código lee los valores de los sensores de pH y turbidez conectados a los pines analógicos A0 y A1, respectivamente.
2. **Procesamiento de Datos**:
   * convertToPh(int value): Convierte el valor del sensor de pH a un valor real de pH.
   * convertToTurbidity(int value): Convierte el valor del sensor de turbidez a un valor de NTU.
3. **Decisiones Basadas en Datos**:
   * Determina si el agua está limpia o turbia y muestra el estado a través del puerto serie.

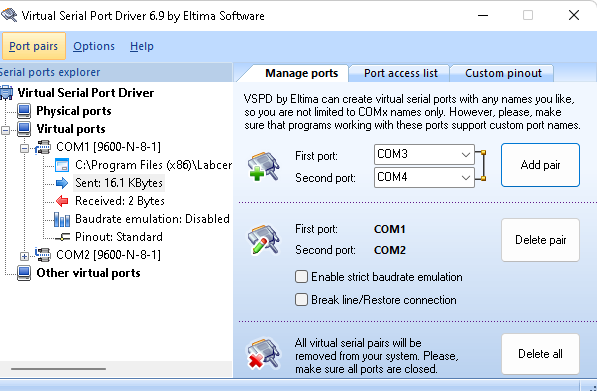
Anexo compilación y ejecución sin ningún inconveniente

Interfaz de usuario gráfica, Texto

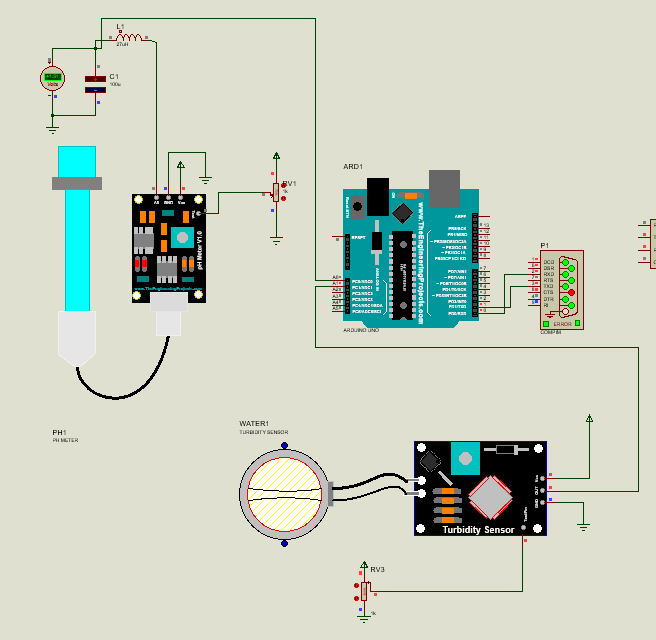
Descripción generada automáticamente

Puertos Serial

El puerto serial en el cual se enviará la información será el puerto serial 1 Y 2



ANEXO PROTEUS ESQUEMATICO



INTEGRACION DE DATOS PROTEUS PYTHON PARA SER ENVIADOS A AZURE (CODE PYTHON)

**Descripción del Código**

1. **Configuración de Azure IoT Hub**:
   * **CONNECTION\_STRING**: Contiene la cadena de conexión necesaria para conectar con Azure IoT Hub.
   * **client**: Se crea un cliente de Azure IoT Hub utilizando la cadena de conexión.
2. **Funciones para Azure IoT Hub**:
   * **enviar\_a\_azure(mensaje)**: Toma un mensaje, lo convierte en formato JSON y lo envía a Azure IoT Hub. Si ocurre un error durante el envío, lo captura y muestra un mensaje de error.
   * **mensaje\_recibido\_handler(message)**: Función de manejo de mensajes que se llama cuando se recibe un mensaje desde Azure IoT Hub. Decodifica el mensaje y lo imprime.
3. **Configuración del Manejador de Mensajes**:
   * **client.on\_message\_received**: Configura el manejador para que mensaje\_recibido\_handler se llame cada vez que se recibe un mensaje desde Azure IoT Hub.
4. **Conexión con Arduino**:
   * **iniciar\_conexion\_arduino()**: Intenta abrir una conexión con el puerto serial (COM2) a 9600 baudios. Espera 2 segundos para asegurarse de que la conexión se establezca correctamente.
   * **cerrar\_conexion\_arduino()**: Cierra la conexión con el puerto serial si está abierta.
5. **Conexión con Azure IoT Hub**:
   * **cerrar\_conexion\_azure()**: Cierra la conexión con Azure IoT Hub cuando el programa termina, capturando y mostrando cualquier error que pueda ocurrir durante el proceso.
6. **Interacción del Usuario**:
   * **print("Presione 'q' para empezar a leer los datos.")**: Imprime un mensaje en la consola para que el usuario presione 'q' cuando quiera comenzar la lectura de datos.
   * **input()**: Lee la entrada del usuario. Si el usuario presiona 'q', se inicia la conexión con Arduino.
7. **Hilo para Mensajes de Azure IoT Hub**:
   * **hilo\_mensajes\_azure()**: Ejecuta un hilo que duerme indefinidamente, permitiendo que el manejador de mensajes (mensaje\_recibido\_handler) maneje los mensajes recibidos.
8. **Bucle Principal**:
   * **while True**: Lee los datos del puerto serial si hay datos disponibles y envía estos datos a Azure IoT Hub. Si ocurre un error durante la lectura o el envío de datos, se captura y muestra un mensaje de error.
9. **Finalización del Programa**:
   * **finally**: Asegura que, al finalizar el programa, se cierren tanto la conexión con Arduino como con Azure IoT Hub.

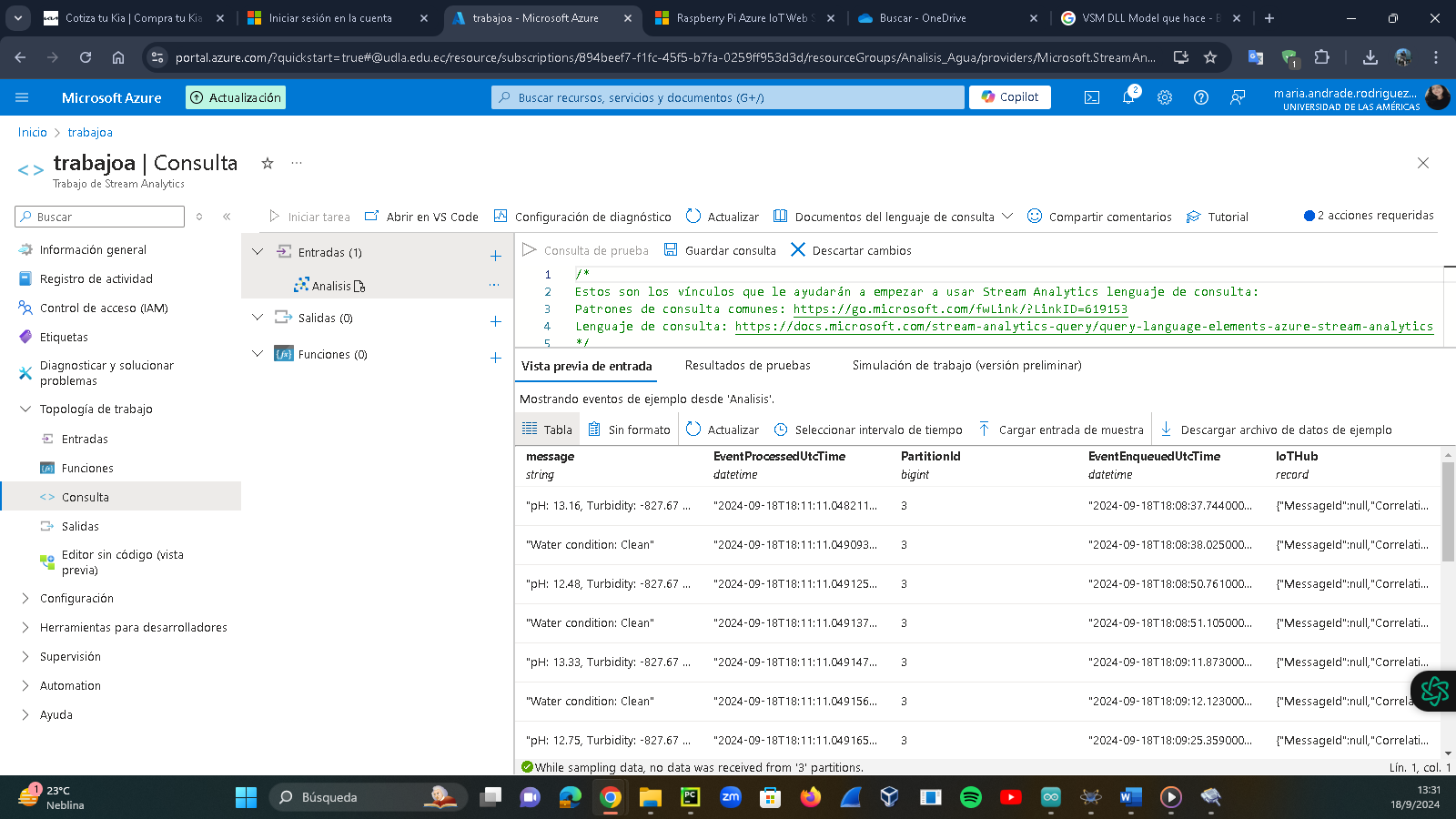
**Resumen**

El código establece una conexión con un dispositivo Arduino a través del puerto serial, lee datos del Arduino, y envía estos datos a Azure IoT Hub. Además, configura un manejador para recibir mensajes desde Azure IoT Hub y maneja cualquier error que pueda surgir durante la conexión y el envío de datos. También incluye manejo de hilos para recibir mensajes desde Azure IoT Hub en segundo plano, permitiendo que la lectura de datos del Arduino y el envío a Azure IoT Hub se realicen simultáneamente.

CAPTURAS PRUEBAS DE DATOS EN PYTHON Y AZURE

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente



Conclusiones sobre la Implementación de IoT

Integración de Sensores con Azure IoT Hub:

Recolección de Datos en Tiempo Real: El código demuestra cómo integrar sensores (pH y turbidez) con una plataforma IoT (Azure IoT Hub). Permite la recolección continua de datos del entorno (agua en este caso) y el envío de estos datos a la nube para su análisis.

Comunicación Bidireccional: Implementa una comunicación bidireccional entre el dispositivo (Arduino) y Azure IoT Hub. Esto incluye el envío de datos desde el dispositivo al cloud y la recepción de comandos o mensajes desde el cloud hacia el dispositivo.

Manejo de Datos del Sensor:

Calibración y Procesamiento de Datos: El código de Arduino se encarga de la calibración y procesamiento de los datos de los sensores. Los valores analógicos se convierten en datos útiles (pH y turbidez), los cuales se envían a Azure IoT Hub para su almacenamiento o análisis posterior.

Condición del Agua: Se realiza una evaluación de la calidad del agua basado en los valores del pH y turbidez, determinando si el agua está limpia o turbia. Esta lógica de decisión puede ser utilizada para activar alertas o tomar acciones automáticas en una aplicación IoT.

Manejo de Mensajes y Errores:

Robustez en la Comunicación: El código implementa manejo de errores tanto en la comunicación con el puerto serial (Arduino) como con Azure IoT Hub. Esto asegura una mayor robustez y estabilidad en el sistema IoT.

Manejo de Mensajes Asíncronos: El uso de un hilo para manejar los mensajes recibidos desde Azure IoT Hub permite que la aplicación pueda recibir mensajes en paralelo a la lectura de datos desde el sensor, optimizando la eficiencia del sistema.

Configuración y Pruebas:

Configuración Flexible: El código permite configuraciones flexibles, como cambiar el puerto serial para el Arduino o actualizar la cadena de conexión de Azure IoT Hub. Esto facilita la adaptación a diferentes entornos y dispositivos.

Validación de Datos: La lectura y envío de datos se realizan de manera continua, y se imprime información en la consola para validación y monitoreo en tiempo real.

Escalabilidad:

Escalabilidad en la Nube: Azure IoT Hub ofrece escalabilidad y gestión de dispositivos a gran escala. Esto permite que el sistema se expanda para integrar más dispositivos y sensores en el futuro sin necesidad de grandes modificaciones en la arquitectura.

Análisis de Datos en la Nube: Una vez que los datos están en Azure IoT Hub, pueden ser utilizados para análisis más avanzados, almacenamiento a largo plazo, y generación de informes, aprovechando las capacidades de Azure.

Recomendaciones para Mejoras

Seguridad: Implementar medidas de seguridad como cifrado de datos, autenticación robusta y autorización para asegurar que las comunicaciones entre el dispositivo y Azure IoT Hub sean seguras.

Optimización de Recursos: Evaluar la eficiencia del código para minimizar el consumo de recursos, especialmente si se planea usar múltiples sensores o dispositivos.

Escalabilidad y Mantenimiento: Considerar la posibilidad de usar un diseño más modular para facilitar el mantenimiento y la expansión del sistema. Implementar un sistema de monitoreo y alertas para detectar y solucionar problemas de manera proactiva.

Documentación y Monitoreo: Asegurarse de tener una buena documentación del código y del sistema en general. Implementar herramientas de monitoreo para seguir el rendimiento del sistema y los datos recibidos.

Este enfoque en la implementación de IoT no solo te proporciona una base sólida para el desarrollo, sino que también te prepara para escalar y mejorar el sistema en función de tus necesidades futuras.

Software Utilizado

Proteus 8 Professional

Librerias Predeterminadas Instaladas

-Ph Meter v1.0

-Turbidity Sensor

Configure Virtual Serial Port Driver

Arduino IDE 2.3.2

Pycharm PyCharm Community Edition 2023.2.3 con version de Pyhton 3.12

Microsoft Azure

