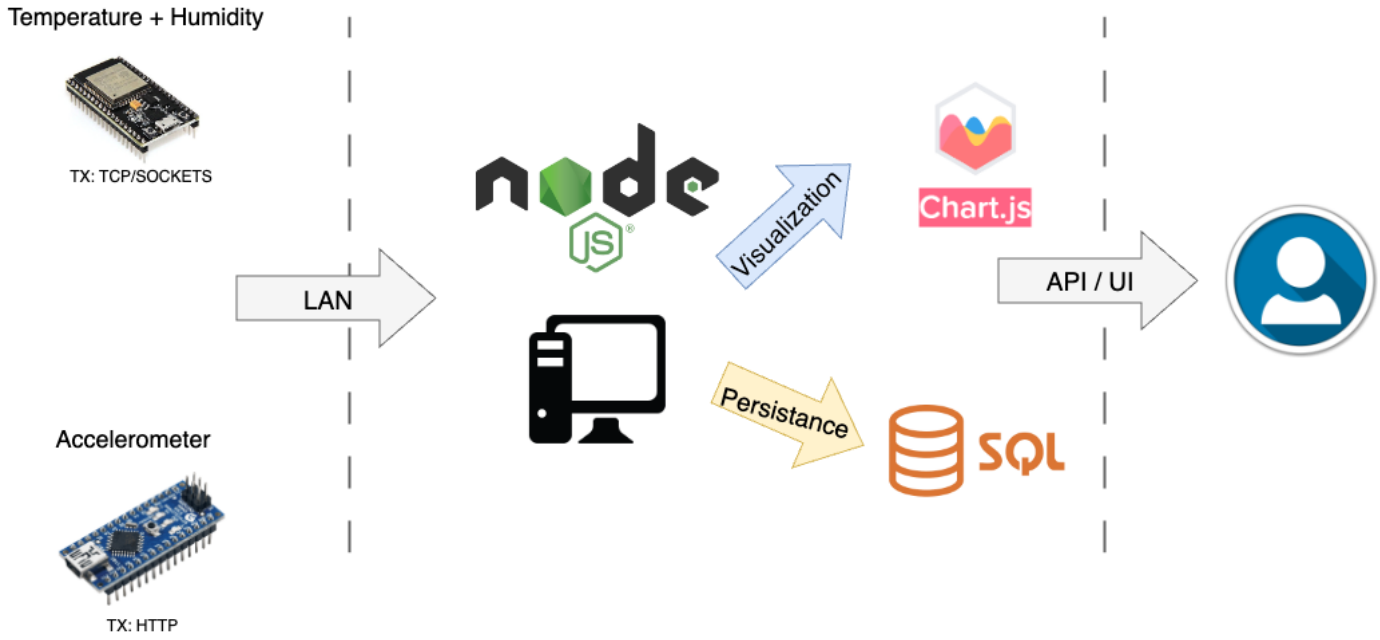
Real-time Sensor Data Acquisition and Visualization



Objective:

In this lab, you will create a system that collects sensor data from two different microcontrollers, the Arduino Nano 33 IoT and ESP32, and sends it to a Node.js server via different communication protocols (HTTP for the Arduino Nano 33 IoT, and TCP sockets for the ESP32). The server will save the incoming data into an SQLite database and provide real-time visualization of the data using Chart.js.
An API endpoint will also be available to retrieve the stored data in JSON format.

At the end of this lab, you should be familiar with the principles of IoT device communication with a server, database management, and real-time data visualization. You should also understand the functional differences between HTTP and TCP protocols in the context of IoT.

Materials and Tools:

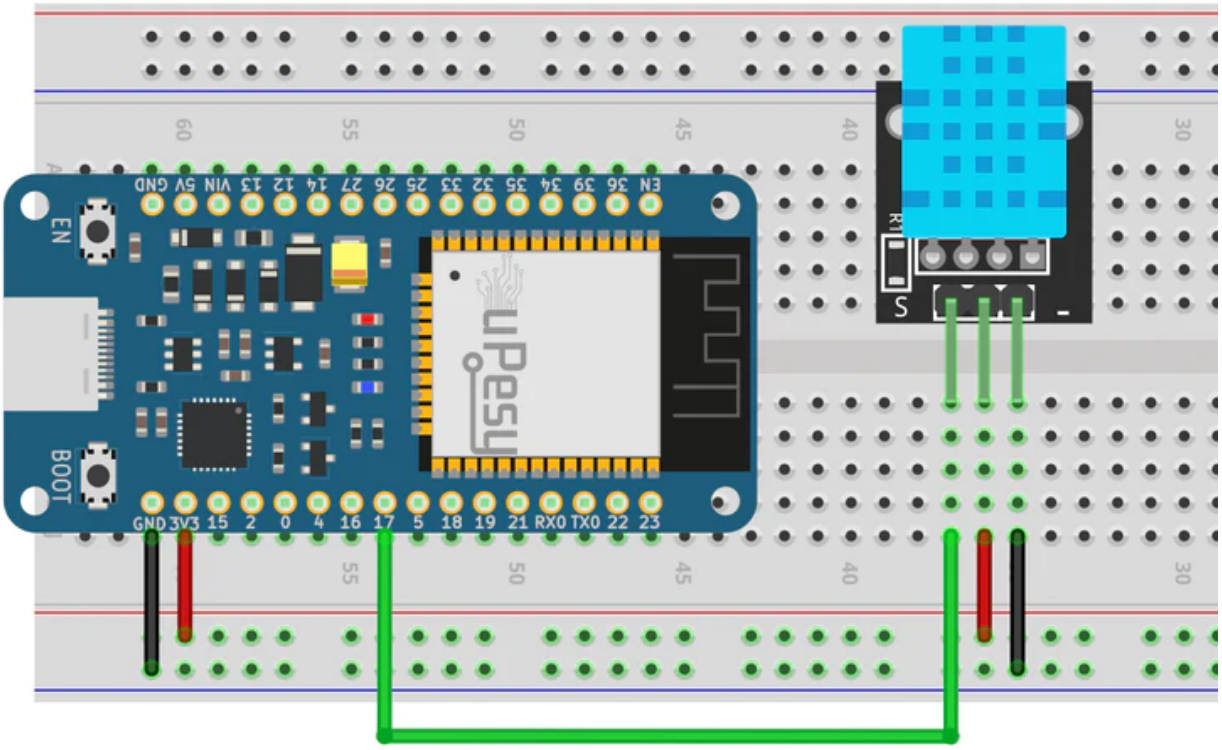
- Arduino Nano 33 IoT
- ESP32 Development Board
- DHT11 Temperature and Humidity Sensor
- Breadboard and jumper wires
- Computer with Internet access
- Node.js and npm installed
- SQLite

Part 1: Setting up the Hardware

Task 1: Connect the DHT11 sensor to the ESP32

Start by installing the ESP32 board in Arduino IDE whether you're using Windows, Mac OS X or Linux:
https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/

Pin connections should be clearly specified.

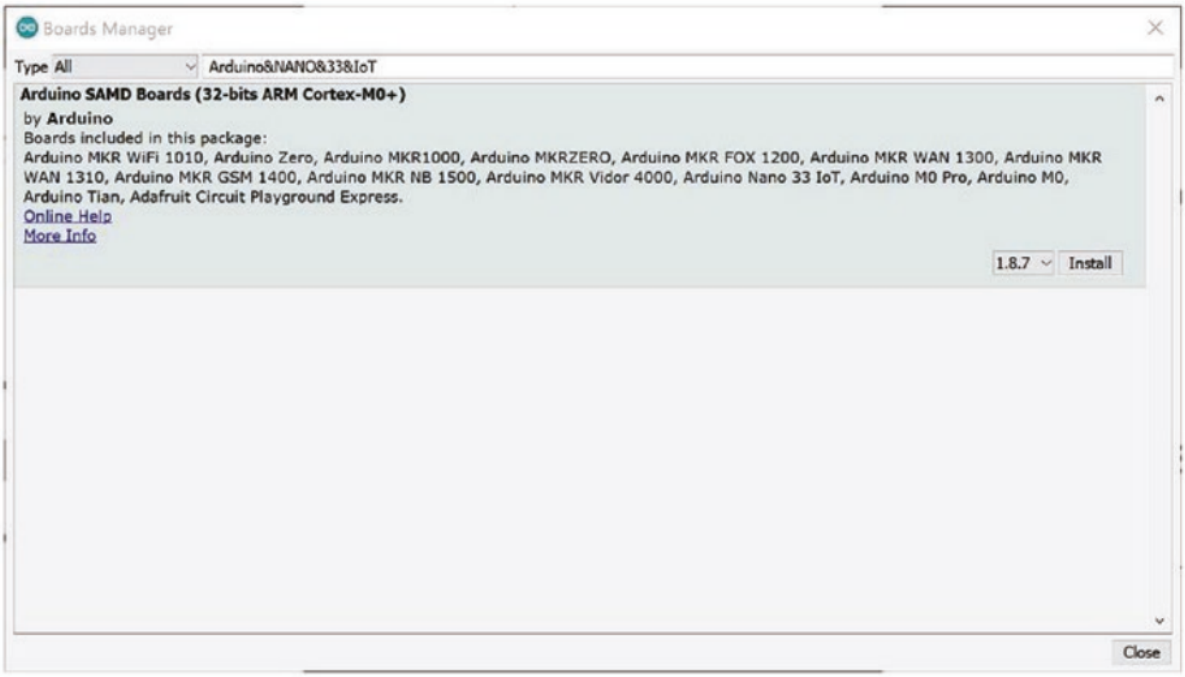


Refer to this tutorial:
https://www.upesy.fr/blogs/tutorials/dht11-humidity-temperature-sensor-with-arduino-code-on-esp32-board

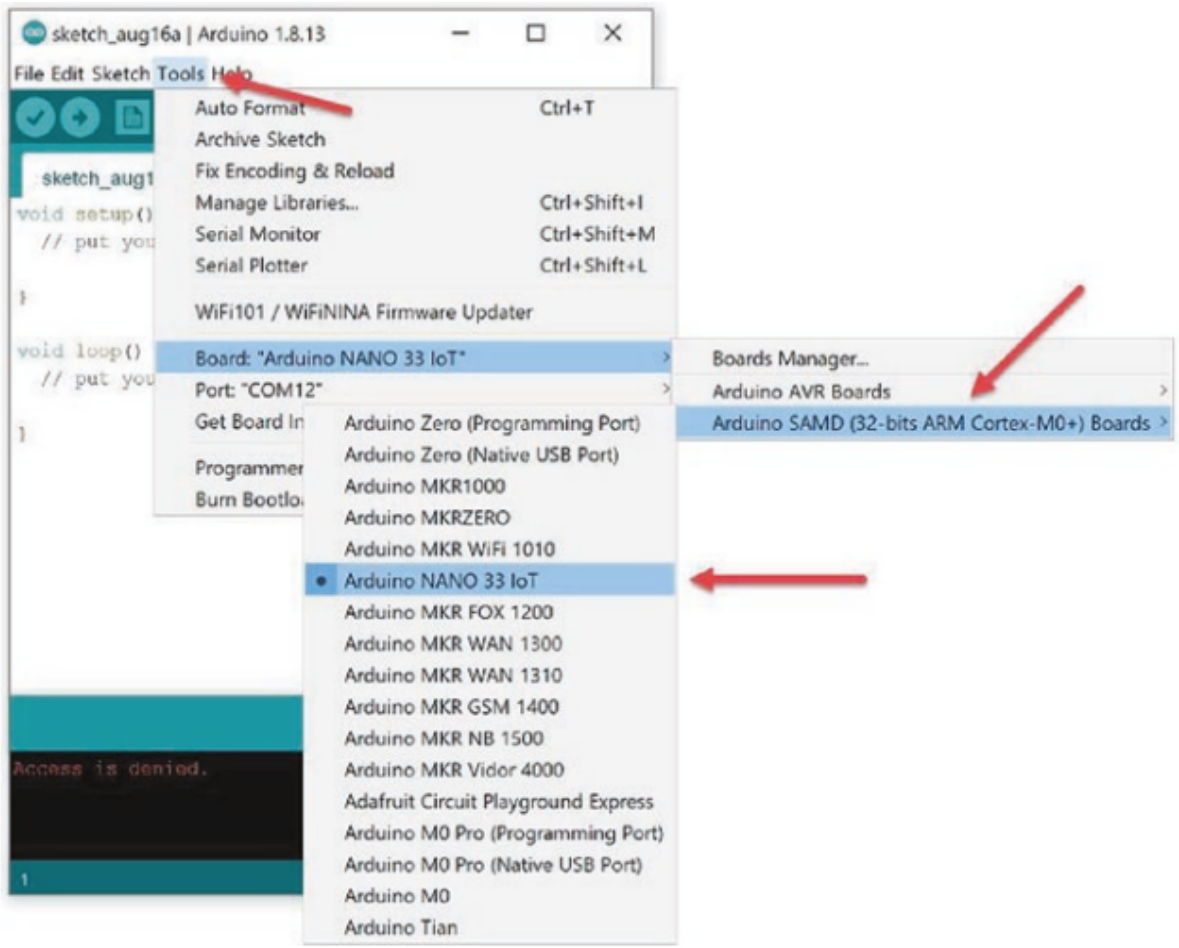
Task 2: Connect the Arduino Nano 33 IoT

Ensure the onboard accelerometer is functional and accessible.

To work with the Arduino Nano 33 IoT board, we need to configure Arduino software. First, we add Arduino SAMD Boards so the Arduino software will recognize our Arduino Nano 33 IoT board. You can open a menu on Arduino software by clicking the menu **Tools ➤ Board ... ➤ Boards Manager…**
After clicking the Board Manager menu, we will obtain the Boards Manager dialog, as shown below. Select All on the Type menu from Boards Manager. Then, type Arduino&NANO&33&IoT in the textbox. You will see Arduino SAMD Boards. Click and install this package. Make sure your computer is connected to an Internet network.



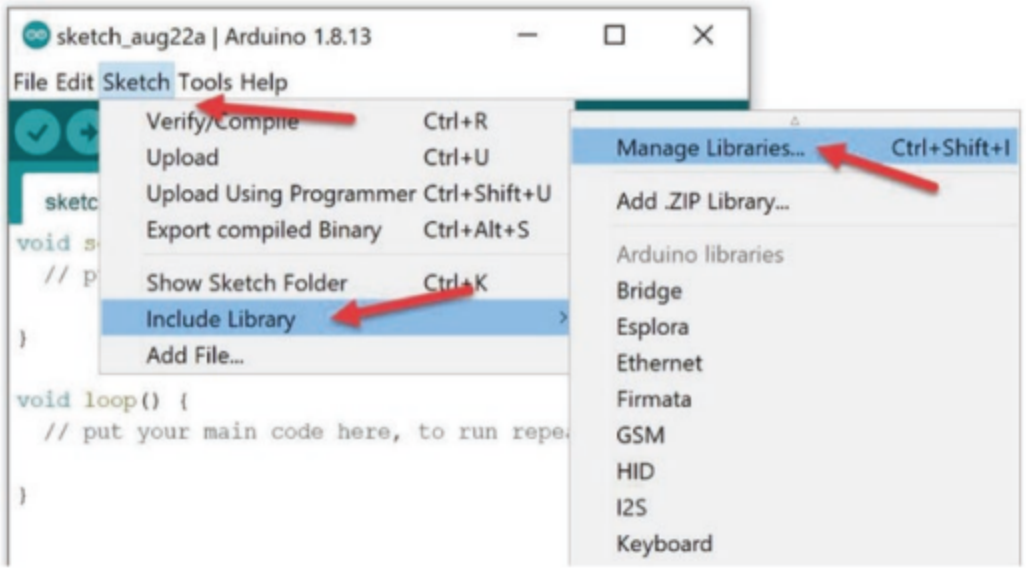
This installation takes several minutes to complete. After the installation, you can see the Arduino Nano 33 IoT board on the targeted board. You can verify it by clicking the menu Tools ➤ Board ... ➤ Boards Manager…on Arduino software. You will see your board list. The figure shows Arduino Nano 33 IoT on Arduino software.

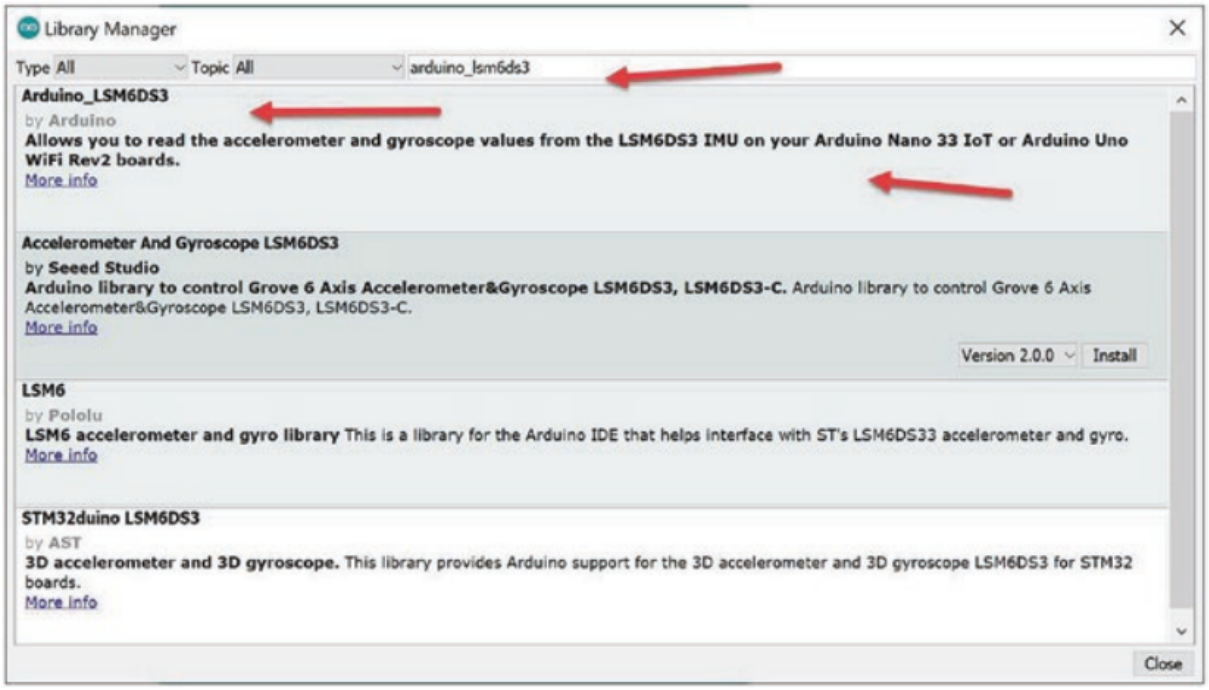


Set Up LSM6DS3 Library

To access the IMU sensor-based LSM6DS3 chip on Arduino Nano 33 IoT, we need to install the Arduino LSM6DS3 library. This library can be used to access the IMU sensor for accelerator and gyroscope sensors. We will use this library in this chapter. Details about the LSM6DS3 library can be read at this link, https://www.arduino.cc/reference/en/libraries/arduino_lsm6ds3/

To install the Arduino LSM6DS3 library, you can open Arduino software. Then, click the menu Sketch ➤ Include Library ➤ Manage Libraries, as shown below:





Check this documentation to learn about how to work with accelerometer data:
https://docs.arduino.cc/tutorials/nano-33-iot/imu-accelerometer

Part 2: Programming the Microcontrollers

Task 1: Program the ESP32

- Write a sketch that reads data from the DHT11 sensor.
- Establish a TCP socket connection to the Node.js server.
- Send the sensor data over the TCP connection.

Task 2: Program the Arduino Nano 33 IoT

- Write a sketch that reads the onboard accelerometer data.
- Use the HTTP protocol to send data to the Node.js server.
- For this task, use the following library:
    - WiFiNINA: https://docs.arduino.cc/tutorials/nano-33-iot/wifi-connection
    - ArduinoHttpClient : https://github.com/arduino-libraries/ArduinoHttpClient

Part 3: Creating the Node.js Server

Task 1: Set up the Node.js environment

- Initialize a new Node.js project.
- Install necessary npm packages (express, sqlite3, ws, etc.).

Task 2: TCP and HTTP Communication

- Create a TCP server to receive data from the ESP32.
- Set up an HTTP endpoint to receive data from the Arduino Nano 33 IoT.

Task 3: Database Management

- Initialize SQLite.
- Create tables to store temperature, humidity, and accelerometer data.

Part 4: Data Visualization and API

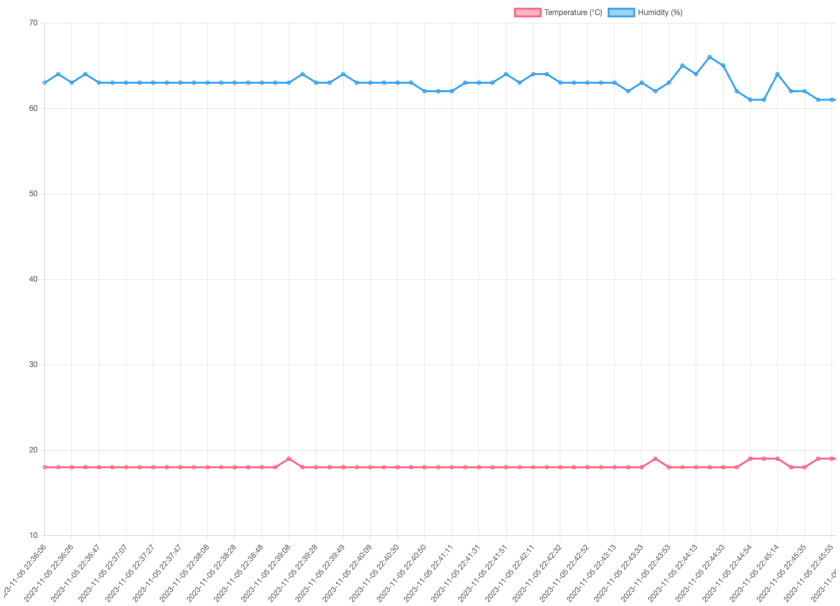
Task 1: Serve the Visualization Page

- Create an HTML page that includes Chart.js for plotting.

- Serve this page with the Node.js server.
- Implement a mechanism to update the charts in real-time (using long polling or WebSockets).

Below is an example of a web page that shows the accelerometer data sent from Arduino Nano 33 in real time using web sockets and chart.js.

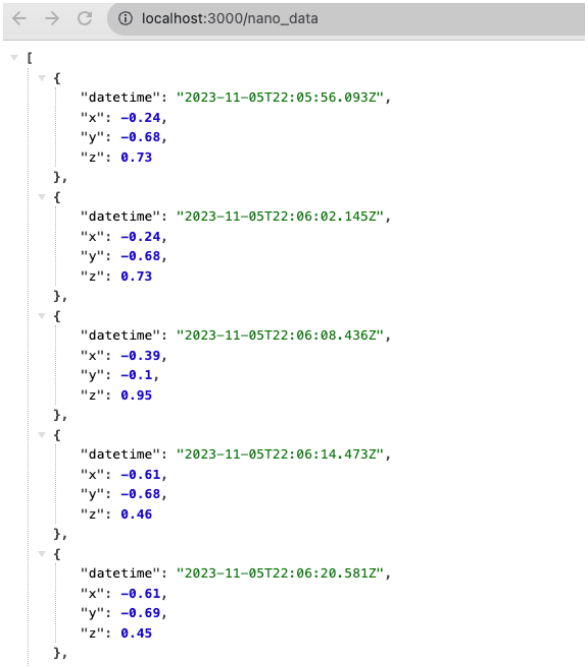
Below is an example of a web page that shows the temperature/humidity data sent from ESP32 in real time using web sockets and chart.js.

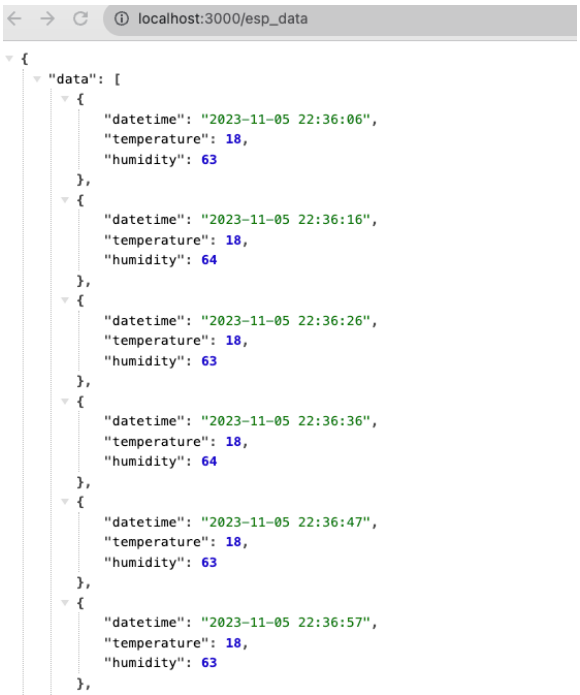


Task 2: API Endpoint

- Create a GET endpoint to serve the data as JSON.

Examples:

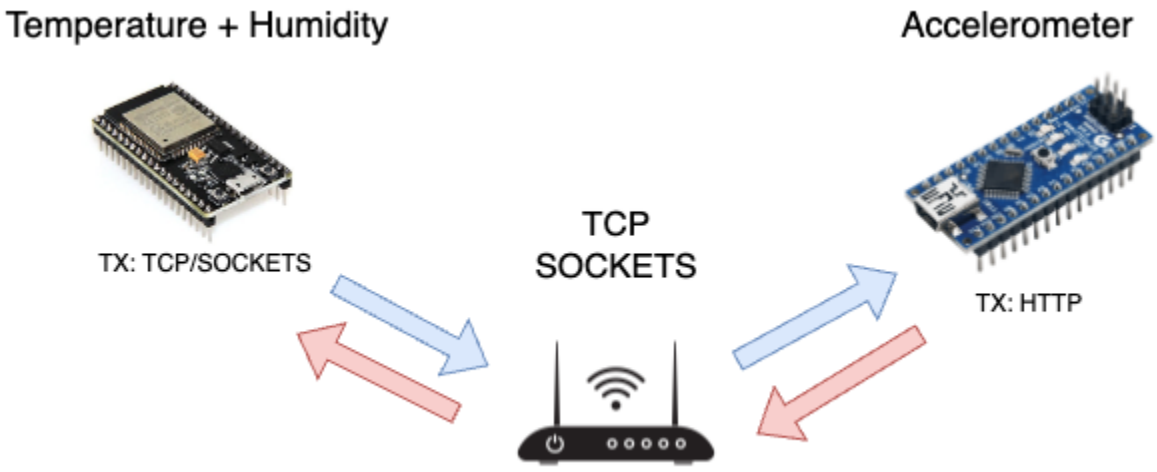




Part 5: Testing and Validation

- Verify the TCP and HTTP communications are working.
- Confirm data is correctly stored in the database.
- Ensure the visualization reflects the sensor data accurately.

Part 6: ESP32 and Arduino Nano Chatting



This task aims to establish a TCP socket communication between the ESP32 and the Arduino Nano 33 for data exchange. This is a high level overview of the tasks to be done:

1. Setup Both Devices to Connect to the WiFi Network

- ESP32 and Arduino Nano IoT:
  - Program both devices to connect to your WiFi network (you can use your cellphone as a hotspot).
  - Use the appropriate WiFi library for each (e.g., WiFi.h for the ESP32 and WiFiNINA.h for the Arduino Nano IoT).

2. Establish the ESP32 as a TCP Server

- ESP32:
  - Program it to act as a TCP server.
  - Use the WiFiServer class to create a server on a specific port.
  - The ESP32 will listen for incoming connections on this port.

3. Configure Arduino Nano IoT as a TCP Client

- Arduino Nano IoT:
    - Program it to act as a TCP client using the WiFiClient class.
    - It will connect to the ESP32's IP address and the port number on which the ESP32 server is listening.

4. Coding and Connection Logic

- Server (ESP32):
    - Waits for the client to make a connection.
    - Upon connection, it can receive and send data to the client.
- Client (Arduino Nano IoT):
    - Initiates the connection to the server.
    - Once connected, it can send and receive data.

5. Handle Data Exchange

- Both devices can now exchange data using TCP sockets. The ESP32 sends the temperature and humidity to the Arduino Nano, and the Nano sends accelerometer data to the ESP32.