Final Project

Assessing the Sensitivity of a Physical Sensor

DAT-230 Data Visualization & Storytelling with AI

Instructor: Dr. Vikas Thammanna Gowda Semester: Fall 2025
Contact: vthammannagowda@champlain.edu Office Location: West Hall 100

Title: Assessing the Sensitivity of a Physical Sensor through Visualization on Web
Assigned Date: 11/10/2025
Due Date: 12/05/2025
Presentation: Finals week Live in-class

1 Overview

This four-Stage project combines hardware experimentation with web-based data visualization. Students will interface a DHT22 digital temperature/humidity sensor and a 20×4LCD display with a Raspberry Pi, creating a simple digital thermometer system. They will then collect sensor data under varying environmental conditions to observe how the sensor's readings respond to changes (i.e., assess the sensor's sensitivity to its environment). Finally, students will build an interactive web dashboard to visualize the collected data, using modern JavaScript libraries, thereby linking the physical experiment with data analysis and presentation.

In summary, this project will give hands-on experience in integrating hardware with software, data logging, and front-end visualization. By the end, you will not only have a working sensor setup displaying live readings, but also an online dashboard to analyze and present how the sensor performed across different conditions. This end-to-end process mirrors real-world IoT (Internet of Things) data workflows and highlights how sensor sensitivity can be studied and communicated effectively.

2 Learning Objectives

By completing this project, students will be able to:

- **Sensor Integration:** Connect and interface a DHT22 sensor and an I2C-enabled 20×4 LCD with a Raspberry Pi, and read real-time temperature and humidity data using Python.
- **Data Handling in Python:** Utilize variables and data types to store sensor readings, performing any necessary computations (e.g., converting temperature units from Celsius to Fahrenheit). (Optional extension: display temperature in both Celsius and Fahrenheit to practice unit conversions.)
- **Data Logging:** Implement a method to record sensor readings over time (e.g., writing to a CSV file) while respecting the sensor's limitations (the DHT22 cannot be read more frequently than about once every 2 seconds to ensure reliable data.
- **Data Consolidation and Analysis:** Consolidate collected data from multiple experimental conditions into a single dataset (using tools like CSV or Excel) and perform basic analysis to compare conditions (e.g., observing how quickly or significantly the sensor's readings change with the environment).
- **Web Data Visualization:** Set up a modern web development environment and parse a CSV dataset within a React application. Build interactive charts to visualize the sensor data using the Recharts library – for example, plotting temperature vs. time – thereby learning to present data in an engaging dashboard format.

- **Dashboard Design and Interpretation:** Apply good visualization practices (labels, legends, tooltips, etc.) to ensure the dashboard is clear and informative. Develop skills in interpreting the resulting graphs to assess the sensor's sensitivity, and communicate findings in a brief report or presentation.

3 Required Materials

Hardware and Components

- **Raspberry Pi:** A Raspberry Pi (any model with a 40-pin GPIO header) with Raspberry Pi OS installed (Raspbian). Ensure the Pi has internet connectivity for installing libraries if needed.
- **DHT22 Sensor:** A DHT22 (AM2302) temperature and humidity sensor. *Note:* You will need a 10 kΩ pull-up resistor to connect between the DHT22 data pin and 3.3V power for stable readings.
- **20×4 Character LCD Display:** A 20×4 character LCD with an I2C interface adapter (often sold attached to the LCD). The I2C interface significantly reduces wiring complexity by using only four connections (power, ground, SDA, SCL).
- **Breadboard and Jumper Wires:** A breadboard and an assortment of male-to-female jumper wires for making the necessary connections between the Raspberry Pi and the components (sensor and LCD).
- **Power Supply:** A reliable 5V power supply for the Raspberry Pi (e.g., the official Raspberry Pi power adapter) to ensure the Pi and attached devices have stable power.

Software and Tools

- **Python 3 and Libraries:** Python 3 (which comes with Raspberry Pi OS). You will also need the appropriate Python library to read the DHT22 sensor (e.g., the Adafruit DHT sensor library) and potentially a library to interface with the I2C LCD (if using a library for the LCD, or you can use SMBus/I2C directly).
- **Node.js and npm:** Node.js (with npm) for setting up the web dashboard development environment. This will be used to create a React application.
- **React and Create-React-App:** A React development setup (you can use `create-react-app` to bootstrap a new React project). This will serve as the foundation for building the data visualization dashboard.
- **Recharts and PapaParse Libraries:** The project will use Recharts (for charts) and PapaParse (for CSV parsing) in the React app. These can be added via npm (e.g., by running `npm install recharts papaparse` in your project).
- **Spreadsheet Software (optional):** A spreadsheet program like Microsoft Excel or Google Sheets. This can be useful for examining or combining data logs (CSV files) and performing any quick manual clean-up or analysis of the data outside of Python.
- **Text Editor/IDE:** A code editor of your choice (VS Code, PyCharm, etc.) for writing Python and JavaScript code.

4 Project Phases (Stage -by-Stage)

Over the course of four Stage s, the project is divided into the following phases. Each Stage builds on the previous, so it is important to stay on schedule. By the end of each Stage, you should achieve the listed milestones:

4.1 Stage 1: Hardware Setup and Initial Testing

In Stage 1, you will assemble the hardware and get the basic sensor system working.

- Circuit Assembly: **Ensure the Raspberry Pi is powered off before wiring any components.** Connect the DHT22 sensor and the LCD to the Raspberry Pi as described below:

  DHT22 Sensor Wiring:

  The DHT22 has three pins: VCC (power), Data out, and GND. We will use a 10 kΩ resistor as a pull-up on the data line (connecting data to VCC) to ensure reliable communication. Wire the DHT22 to the Raspberry Pi as follows:

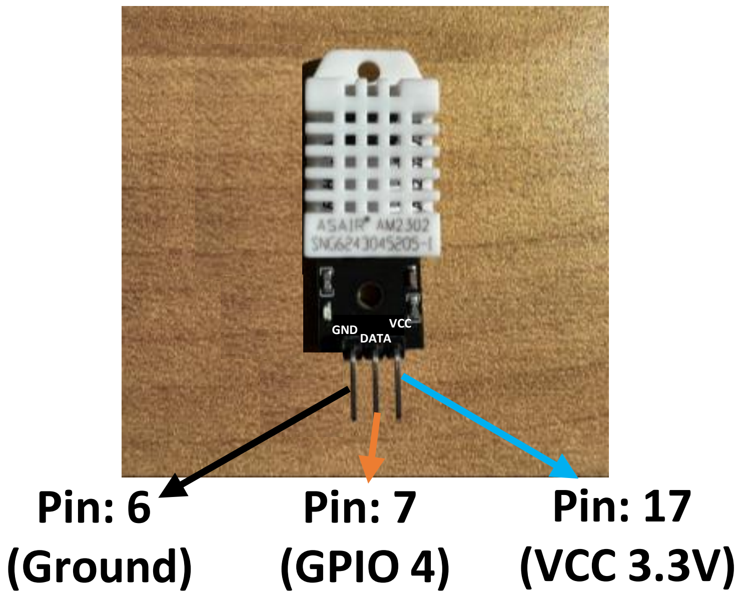

Figure 1: DHT22 Sensor connection to Raspberry Pi's GPIO pins

1. Place a 10 kΩ resistor between the DHT22's VCC pin (pin 1) and Data pin (pin 2) on the breadboard. This resistor acts as a pull-up, tying the data line to 3.3V when idle. (If you had a DHT22 module on a breakout board, this resistor would typically be built-in.)
2. Connect DHT22 pin 3 (VCC) to the Raspberry Pi's 3.3V power pin (e.g. physical pin 17 on the Pi)
3. Connect DHT22 pin 2 (Data) to a Raspberry Pi GPIO input pin. In our example, we'll use GPIO4 (physical pin 7) which is a common default for DHT sensors.
4. Connect DHT22 pin 1 (GND) to a ground pin on the Raspberry Pi (physical pin 6 or any other GND)

Note: The DHT22 should be powered at 3.3V (not 5V) when used with the Pi, to keep the data signal at Pi-safe voltage levels. Also, the sensor cannot be read more often than once every 2 seconds (approx.), as it needs time to take a measurement.

LCD Display Wiring:

For the 20×4 character LCD, we assume it has an I^{2}C interface adapter attached. This adapter greatly simplifies the wiring by using only four connections.

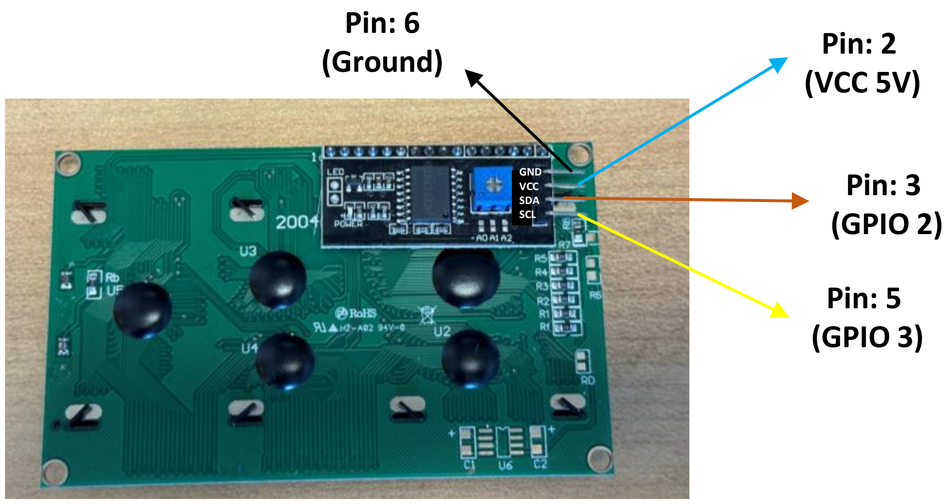

Figure 2: LCD connection to Raspberry Pi's GPIO pins.

1. Connect the LCD module's VCC pin to the Pi's 5V power (physical pin 2 or 4), and the LCD GND to a Pi GND pin 6.
2. Connect the LCD's SDA pin to the Pi's SDA line. On Raspberry Pi, SDA corresponds to GPIO 2 (physical pin 3)
3. Connect the LCD's SCL pin to the Pi's SCL line, which is GPIO 3 (physical pin 5).

- Install Sensor Libraries: Power on the Raspberry Pi. Install any required Python libraries (for example, use `pip` to install the DHT sensor library if not already available). Also install any library or enable interfaces needed for the LCD (for instance, ensure I2C is enabled on the Pi via `raspi-config`, and install an LCD library if provided).
- Python Testing: Write or use a provided Python script to read data from the DHT22 and display the readings on the LCD. The script should initialize the sensor (using the DHT library), read the temperature (in °C) and humidity (%) values, and then print these values to the LCD screen. Aim to update the readings roughly every two seconds (in accordance with the DHT22's read interval limit).
- Verify Operation: Run the script and observe the LCD. You should see the temperature (°C) and humidity values updating periodically. Confirm that the readings make sense (e.g., around room temperature and reasonable humidity). If the LCD displays garbled text or the sensor reads as `None`/`NaN`, double-check your wiring and library installation.
- (Optional) Extension - Unit Conversion: For an extra challenge, extend your Python code to also calculate and display the temperature in Fahrenheit (and/or Kelvin) on the LCD. This can be done by using the formula $F = C \times \frac{9}{5} + 32$ for Fahrenheit. This step is not required for the core project, but it helps you practice variable usage and arithmetic operations in Python.

4.2 Stage 2: Data Collection under Varying Conditions

In Stage 2, you will design and execute experiments to test the sensor's sensitivity. This involves collecting data with the sensor in different environmental conditions and recording the data for later

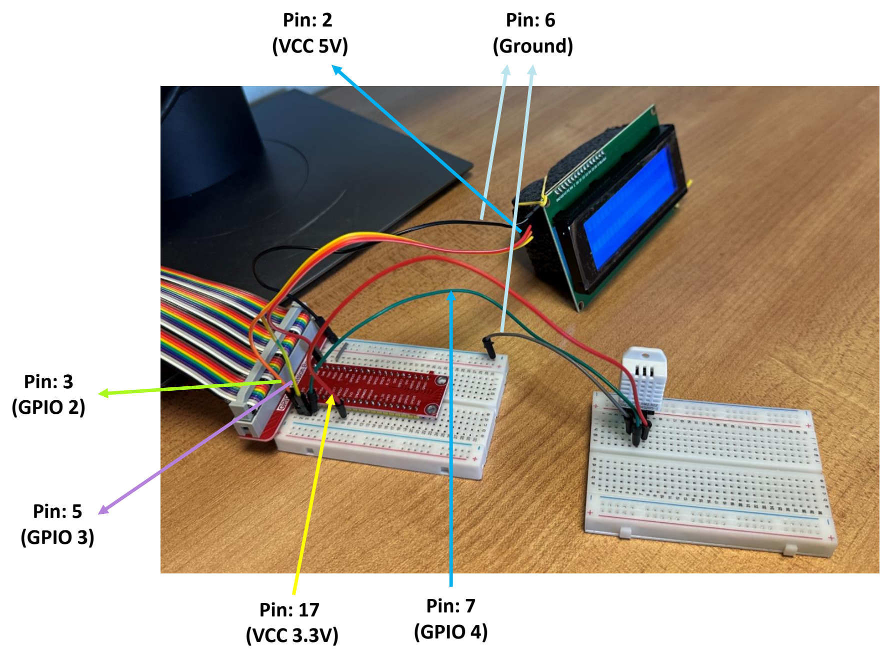

Figure 3: Wiring set-up.

analysis.

- Plan Two Conditions: Identify at least two distinct conditions to test the sensor. For example, one condition could be *"Normal room environment"* (ambient temperature and humidity), and another could be *"Heated environment"* (e.g., placing the sensor near a gentle heat source or in a warmer location). You might also consider a *"High humidity"* test (e.g., placing the sensor near a source of moisture) depending on the sensor's capabilities. The goal is to see how much the sensor's readings change and how quickly they respond when the environment changes.
- Logging Data: For each condition, run the DHT22 sensor for a sustained period and log the data. Modify your Python script to record readings to a file (CSV format is recommended) at regular intervals. Each data entry should include a timestamp, the temperature, and the humidity. Ensure your sampling interval is reasonable – around once every 2 seconds or slower (the DHT22 won't update faster than this). You might run the sensor for, say, 10 minutes for each condition to collect a good amount of data points (this could yield on the order of a few hundred readings per condition at a 2 s interval).
- Conduct Experiments: Start with the first condition (e.g., room environment). Run your logging script and allow the sensor to collect data. Monitor the setup to ensure everything is recording properly. Then change to the second condition (e.g., move the sensor to the new environment or alter the environment around it) and record another set of data. Be sure to label or separate the data from each condition (for instance, using separate CSV files or adding a "condition" column to your data).
- Ensure Data Quality: As you collect data, verify that the readings are plausible (for example, if you warmed the sensor, the temperature values should rise compared to the room baseline). Also check that timestamps are recorded correctly and consistently (preferably in a format like `YYYY-MM-DD HH:MM:SS` for clarity).
- Example Data Structure: The logged data should be structured, as illustrated in Table 1.

Having consistent columns will make it easier to merge and analyze the data later.

Column	**Description**
Timestamp	Date and time of the reading (e.g., 2025-09-01 14:30:00)
Temperature (°C)	Measured temperature in Celsius from the DHT22
Humidity (%)	Measured relative humidity (percentage) from the DHT22
Condition	Label for the test condition (e.g., *RoomTemp*, *Heated*)

Table 1: Suggested format for logging sensor data. Each reading's timestamp, temperature, humidity, and a condition identifier are recorded.

By the end of Stage 2, you should have at least two sets of logged data (in CSV files or Excel). For example, one file for each condition, or one combined file with a column distinguishing conditions. Keep these data files safe – you will use them in the next stages for visualization.

4.3 Stage 3: Data Consolidation and Dashboard Setup

Stage 3 focuses on preparing your data for visualization and starting to build the web dashboard.

- Consolidate Data: If you recorded data in separate files for different conditions, merge them into a single dataset (this can be done using a script or manually in a spreadsheet). Ensure the final CSV has all readings and includes a column indicating the condition for each row (as noted in Table 1). Clean the data if necessary (remove clearly erroneous data points, fill in missing entries if any, etc.). The result should be one comprehensive CSV file (e.g., `sensor_data.csv`) containing all your sensor readings.
- Set Up React App: Initialize a new React project for your dashboard. Ensure Node.js and npm are installed on your development machine. You can use `npx create-react-app` (or a similar tool) to scaffold a React application (for example, you might create an app called *sensor-dashboard*). This will generate the basic project structure for you.
- Install Charting Libraries: Inside your React project directory, install the necessary libraries for data visualization. Specifically, add Recharts and PapaParse to your project by running `npm install recharts papaparse`. This will make Recharts (for rendering charts) and PapaParse (for parsing CSV files) available in your React app.
- Import Data into the App: Add your consolidated CSV data file to the React app's public folder (for example, you can create a `public/data` directory and place `sensor_data.csv` inside it). By keeping the file in the public directory, you can easily fetch it in your app by reference (e.g., using the path `/data/sensor_data.csv`).
- Parse CSV Data in React: In your React code, use the PapaParse library to load and parse the CSV file when the application loads. For instance, you might use `Papa.parse()` in a React effect hook to read the CSV asynchronously. Configure PapaParse with headers (if your CSV has a header row) and ensure it interprets numeric values correctly. Once parsing is complete, you should have your data accessible as a JavaScript array of objects (each object representing a row of your CSV).
- Verify Data Loading: To confirm that everything is set up correctly so far, you can temporarily use `console.log` in your React app to print out some of the parsed data, or even render a simple list of readings in the browser. This sanity check ensures that the React app is successfully reading your sensor data file.

By the end of Stage 3, your data should be neatly consolidated into one file, and your React application should be capable of loading that data. You will have laid the groundwork for visualization by preparing the dataset and setting up the development environment with the required libraries.

4.4 Stage 4: Dashboard Implementation and Finalization

In Stage 4, you will create the visualizations and finalize the project. The aim is to build an informative dashboard that showcases the sensor data and to prepare for presenting your findings.

- Build Data Visualizations: Using Recharts, implement charts to display your sensor readings. A primary chart could be a **time-series line chart** that plots Temperature (°C) vs. Timestamp, so you can see how temperature changed over time during your experiments. Similarly, you could create another line chart (or use the same chart with a second y-axis) for Humidity vs. Time. If you included multiple conditions in one dataset, consider using different colored lines or markers to differentiate the conditions on the chart, or even separate charts per condition for clarity.
- Implement the Chart in React: Within your React app, utilize Recharts components such as `LineChart`, `XAxis`, `YAxis`, `Tooltip`, `Legend`, and `Line` to construct your chart. Feed it the parsed data from PapaParse. For example, you might have an array of data points and use a `Line` for temperature (with `dataKey` pointing to the temperature field in your data objects). Ensure that the x-axis is properly formatted to show time (you may use the timestamps as strings, or convert them to JavaScript Date objects for better handling).
- Dashboard Features and Polish: Add a title and labels to your chart(s) (e.g., ”Temperature vs Time” with axes labeled in degrees Celsius and time units). Enable the tooltip in Recharts so that when a user hovers over the chart, the exact values and time are displayed. If you have multiple data series (like humidity and temperature, or multiple conditions), include a legend to distinguish them. Make sure the design is tidy: for instance, if the data range is large, adjust the axis ranges or formatting for readability. Basic CSS or inline styling can be used to ensure the dashboard looks presentable (fonts, alignment, etc.).
- Testing the Dashboard: Run your React app (usually with `npm start`) and verify that the charts render correctly. Check that all data is represented (scroll through the time axis if needed to see all points). Test interactive features: hovering should show tooltip details, legend toggling (if enabled) should work, etc. If you notice any performance issues with a large number of data points, you might down-sample the data or use React optimizations, but this is unlikely to be a problem for moderate data sizes.
- Interpretation of Results: As you finalize the dashboard, take note of what the data is showing you. How did the temperature and humidity values differ between your test conditions? Did the DHT22 sensor respond immediately to changes or was there a noticeable delay? Were the changes in readings significant or relatively minor? These observations will be important when you compile your report and discuss the sensor's sensitivity.
- Project Finalization: Ensure your code (both Python and JavaScript) is clean and well-documented with comments. It's a good idea to save plots or screenshots of your dashboard, which can be included in your report or used during a presentation.

By the end of Stage 4, you should have a fully functional system: the Raspberry Pi collects data from the DHT22 sensor, and an interactive React-based dashboard visualizes this data for viewers. You will also consolidate your findings and prepare to demonstrate or submit your project.

5 Deliverables & Submission

By the conclusion of the project (end of Stage 4), students must submit/deliver the following items:

- **Hardware Demonstration:** In the lab or via a short video, demonstrate the working sensor and LCD setup. Show that the DHT22 is correctly wired and the LCD displays live temperature and humidity readings.
- **Data Files:** All collected data in CSV format. This should include the raw data logs (e.g., the CSV file with all sensor readings) that were used for the dashboard. Ensure that the data is labeled and clean, as these files may be reviewed for completeness.
- **Source Code:** Provide the code written for both parts of the project. This includes the Python script used on the Raspberry Pi for data collection, and the source code of the React application for the dashboard (all relevant files: components, etc.). You can upload these to a version control repository (such as GitHub) or submit them as a zip file. The code should be well-organized and commented.
- **Dashboard Access:** If possible, deploy the web dashboard or provide instructions for running it. For example, you might host it on a service (GitHub Pages, Netlify, etc.) or simply ensure that we can run it with `npm start`. At minimum, include screenshots of your dashboard in your report (especially the charts) in case we cannot run the live version.
- **Project Report:** A brief report (approximately 2-4 pages) summarizing your project. This report should include:
  - An introduction describing the project goals and the sensor setup.
  - A short description of how you conducted the experiments (conditions tested, how data was collected).
  - At least one visualization (chart) from your dashboard with an explanation. You can paste a screenshot of your chart into the report.
  - An analysis of the results: discuss what the data shows about the sensor's sensitivity. For instance, how did the temperature change when you moved the sensor from room temperature to the heated environment? How quickly did it stabilize? Similarly, what happened with humidity? Comment on any sources of error or uncertainty (e.g., DHT22 accuracy limits, response lag).
  - A conclusion reflecting on what you learned and any suggestions for further investigation (for example, how might you test sensor sensitivity more rigorously, or what you would do differently next time).
- **Submission Format:** Submit the report as a PDF. The code can be submitted via a link or archive as mentioned. Data files should be included as CSVs. Make sure to name your files clearly (e.g., `SensorProject_Report.pdf`, `sensor_data.csv`, `sensor_dashboard_code.zip`, etc.).
- **Due Date:** All project materials are due by [Due Date] (end of Stage 4). Late submissions may be subject to the course's late policy.

Note: This is an individual project (each student must submit their own work) unless otherwise specified by the instructor. You are encouraged to discuss ideas and troubleshoot with peers, but all data collection, code, and analysis should be your own. Academic integrity policies apply.

6 Grading Rubric

Table 2 outlines how the project will be assessed. The criteria reflect both the process (hardware setup, data gathering, etc.) and the final outputs (dashboard and report). Ensure you meet each component to maximize your score.

Evaluation Criteria	**Points**
Hardware Setup & Wiring: All components (DHT22 sensor and LCD) are correctly wired and functioning. The sensor readings display on the LCD as expected (proof of working setup).	15
Data Collection & Logging: Sufficient data was collected under the different conditions. Data logging is implemented correctly (appropriate format, timestamps, no significant gaps or errors in data).	15
Data Consolidation & Preparation: All experimental data have been combined into a well-structured single dataset. Data is labeled by condition and cleaned (ready for use in the dashboard).	10
Dashboard Functionality (React App): The web dashboard loads the data and presents interactive charts without errors. The use of PapaParse and Recharts is correctly implemented (e.g., data parsing occurs, chart renders with proper data points).	30
Visualization Design & Clarity: The charts are well-designed and easy to interpret. Axes are labeled (with units), legends or annotations distinguish different variables or conditions, and any interactive features (tooltips, etc.) enhance understanding.	10
Report & Analysis: The written report is clear and well-organized. It accurately describes the project methodology and presents the results. The analysis of the sensor's sensitivity (differences between conditions, response behavior) is insightful and supported by the data.	20
Total	100

Table 2: Grading rubric for the project. Meeting each criterion will ensure full credit for that component. Instructors may adjust point values or criteria details, but the total will remain 100 points.