Case Study 3: Indoor Climate Monitoring and Visualization with Raspberry Pi

DAT-230 Data Visualization & Storytelling with AI

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

Case Study 3: Weather and Climate Patterns
Assigned Date: 10/09/2025
Due Date: 10/22/2025

Overview

This case study engages student teams in an end-to-end exercise combining embedded sensing, data collection via Raspberry Pi, critical visualization design, and bias identification/mitigation. Students will build a weather and indoor climate monitoring system using a Raspberry Pi and multiple environmental sensors, collect continuous time-series data, and produce a suite of visualizations. Two starter visualizations will be provided: one with poor design and another with an intentional bias. Students must diagnose and improve those, create four additional visualizations (including one with an introduced bias), and reflect on how visual choices influence interpretation. The project develops skills in ethical data storytelling, sensor-based data engineering, and effective visual communication.

Learning Goals

- Acquire, log, and preprocess real-time indoor climate data using Raspberry Pi and sensors.
- Visualize temporal patterns (trends, seasonality, anomalies) in environmental data.
- Compare environmental conditions across spatial or temporal contexts.
- Construct composite indices (e.g., comfort index) combining multiple sensor metrics.
- Detect and articulate sources of bias (sensor placement, sampling choices, encoding).
- Apply visual design principles to improve clarity and correct misleading representations.
- Use LLMs responsibly to assist with coding, debugging, and interpretive prompts, documenting the interaction and verification process.

Assessment Focus

The project assesses:

- Ethical awareness in data visualization and bias introduction/mitigation.
- Technical correctness and reproducibility of data collection and processing code.
- Design quality of visualizations and rationale for improvements.
- Reflection on how choices (both good and biased) affect audience perception.
- Transparent and critical use of AI/LLM assistance.

1 Project Setup and Experimental Design

1.1 Hardware Components

- **Raspberry Pi 3/4** with WiFi (OS installed, SSH enabled)
- **DHT22** sensor for temperature and humidity

- **MQ135** gas sensor for air quality indicators (e.g., CO_2, NH_3, benzene)
- **Photoresistor (LDR)** or **TSL2561** for light intensity
- **Optional**: Additional sensors for comparison (e.g., second DHT22), SD card for logging, breadboard, jumper wires, pull-up resistors, enclosure for fixed placement

1.2 Data Collection Objectives

1. Continuously sample indoor environmental readings: temperature, humidity, air quality proxy, and light intensity. Continuous sampling means the system should autonomously take readings at regular intervals without manual intervention. For each sensor:

- **Temperature & Humidity (DHT22):** Read both values together since they're provided in one transaction. Watch out for occasional failed reads; implement retry logic and flag dropped samples.
- **Air Quality Proxy (MQ135):** This sensor output drifts and is non-linear. You'll typically read an analog voltage (via an ADC) and map it to a relative air quality metric. Record the raw value and, if possible, apply a simple smoothing (e.g., rolling average) in post-processing to reduce noise, but also preserve the raw for bias analysis.
- **Light Intensity (LDR / TSL2561):** LDRs require a resistor divider and ADC reading; TSL2561 gives lux directly over I2C. Log in consistent units (e.g., lux) and note saturation (too bright) or floor effects (too dark).

Implementation notes: Buffer writes so that sensor read latency doesn't block the next scheduled sample; use lightweight daemons (Python script with `while` loop + `sleep`) or `cron` with careful timing.

2. Timestamp each reading for time-series analysis: Use accurate, consistent timestamps:

- Use ISO 8601 format with timezone (e.g., `2025-08-03T14:32:00-04:00`) or UTC to avoid ambiguity.
- Prefer getting time from the system clock synchronized via NTP. Verify the Raspberry Pi's clock drift before long runs.
- Record the timestamp immediately upon acquiring sensor values, not after processing, to minimize jitter.
- Include both the intended sample time and actual acquisition time if there's any delay logic (helps in post hoc latency analysis).

If multiple devices are deployed (see optional below), ensure their clocks are synchronized (e.g., all using NTP to the same server) so cross-comparison isn't corrupted by skew.

3. Operate the system continuously for a minimum of 3–5 days to capture temporal variation: Short bursts miss diurnal cycles, weekday/weekend effects, and transient anomalies. Continuous multi-day operation allows:

- Capturing daily periodicity (temperature rise/fall, human activity patterns).
- Observing slow drifts (e.g., HVAC cycling, sensor baseline shifts).
- Identifying anomalies that might occur only occasionally (e.g., sudden humidity spikes when a door opens).

Practical concerns:

- Ensure power stability (use UPS or stable supply).
- Log system health (uptime, script restarts, failed samples) alongside sensor data.
- Rotate or archive logs to avoid filling the SD card (e.g., daily rollover, compress older batches).

4.(Optional) Deploy two setups in different room locations to compare spatial variability (e.g., near a window vs interior corner)

- Spatial variability exposes environmental heterogeneity and helps surface bias caused by placement.
- Example contrast: a sensor near a window might show higher temperature fluctuation and light intensity; one in a corner might show delayed responses or lower ventilation (affecting humidity/air quality).
- Ensure both systems are calibrated similarly and their clocks aligned.
- Label each dataset with location metadata; keep a "placement log" describing physical context (distance to window, height above floor, proximity to vents/heat sources).
- Use this to answer questions like: "Does proximity to external light sources correlate with temperature variability?" or "Do indoor air quality readings lag in poorly ventilated corners?"
- When comparing, account for micro-environment confounders — e.g., if one location gets sunlight during part of day, that's a systematic difference you must document.

1.3 Sampling Design

- **Duration**: At least 3–5 consecutive days.
- **Frequency**: Every 1–5 minutes (trade off data resolution vs storage/processing).
- **Placement**: Fixed sensor mounting; avoid direct airflow, sunlight glare, or heat sources unless intentionally experimenting with bias due to placement.
- **Calibration Notes**: Record any known offsets or environmental effects (e.g., sensor warm-up time, MQ135 baseline drift).

2 Student Activities

2.1 Bias Identification and Design Enhancement

Students will be provided with two starter visualizations derived from the collected (or sample) weather/indoor climate dataset:

1. **Design-flawed visualization**: Contains issues such as poor labeling, ambiguous color scales, cluttered presentation, or ineffective use of visual encoding.
2. **Biased visualization**: Manipulates perception via selective axis truncation, omitted context, misleading filtering, or other encoding choices that could misinform a viewer.

Tasks:

1. Analyze each starter plot, identify the specific flaws or biases, and document why they mislead or obscure insight.
2. Create improved versions: correct design issues, clarify encoding, and remove bias in the biased version while preserving the intended data story accurately.
3. Develop four additional visualizations using the same dataset:
   - At least one of these must intentionally introduce a new form of bias (e.g., reversing time order, truncating axes, selective sampling, deceptive color mapping).
   - For the biased one, explicitly describe the bias, its potential impact, and then provide a corrected, unbiased counterpart for comparison.

2.2 Visualization Questions (Guided)

Students should address and, where applicable, implement the following in their visualization suite:

1. Plot daily temperature (min/max/average) with rolling averages; annotate significant events (e.g., heatwaves, cold snaps).

2. Compare light intensity or air quality across different times or locations using small multiples or distribution plots.
3. Scatter plot of temperature vs humidity, highlighting periods with poor air quality or significant anomalies.
4. Define and plot a *comfort index* derived from temperature, humidity, and possibly air quality/light exposure.
5. Visualize anomalies relative to a baseline (e.g., deviation from historical averages or expected comfort thresholds).
6. Build an interactive dashboard (e.g., Shiny, Python/Plotly Dash) enabling user filtering by time range, location, and variable combinations.
7. Optional: Correlate poor indoor air quality (as indicated by MQ135) with environmental patterns such as temperature inversion, low ventilation times, or lighting conditions.

2.3 LLM Prompting and Documentation

Students are expected to use LLM assistance for parts of the workflow (coding, design suggestions, data interpretation), with the following requirements:

- Provide example prompts used, including the original prompt and any refinements.
- Record the model output versions and what modifications were made to adapt to the actual data/context.
- Explain how each prompt informed an analysis or visualization decision, and describe how outputs were verified or corrected.
- Reflect on what was learned from using the LLM, including limitations or errors encountered.

2.4 Collaboration and Academic Integrity

- Peer discussion and feedback are encouraged, but all submitted code and narrative reflections must be individual unless explicit group approval is given.
- AI assistance is permitted but must be transparent: any code or insight sourced from an LLM must be cited in the prompt log and adapted with student understanding.
- Copy-pasting without comprehension or attribution is considered academic dishonesty under institutional policy.

3 Deliverables and Grading

3.1 Deliverables

1. **Code** (20%): Reproducible scripts (R, Python, or Shiny app) that ingest the sensor data, process it, and generate all visualizations discussed. Should include data cleaning, rolling computation, composite index calculation, and bias-introduced visual generation.
2. **Report** (50%): PDF document adhering to formatting guidelines. Must include:
   - Analysis of starter visualizations and improvements.
   - Four new visualizations (with bias introduced in one) plus unbiased correction.
   - Narrative insights, interpretation, and discussion of environmental patterns and anomalies.
   - Discussion of bias sources (sensor placement, sampling, encoding) and how they were surfaced or mitigated.
3. **Prompt Log** (30%): PDF capturing LLM prompts, output versions, modifications, verification procedures, and reflections.

3.2 Rubric Summary

3.2.1 Code Expectations (20%)

- Correctness and reproducibility of processing and visualization.
- Organization, naming clarity, comments, and application of appropriate libraries.
- Effective use of tools and avoidance of redundant code.

3.2.2 Report Breakdown (50%)

Criterion	Points
Enhanced visualizations from starter code, including bias introduction	10
At least 4 complete visualizations with clear captions	20
Interpretation and insights for each visualization	20
Narrative coherence linking goals, data, and visuals	15
Discussion of biases, confounders, and outliers	15
Clarity, formatting, and adherence to template	20
Total	100

3.2.3 Prompt Log Breakdown (30%)

Criterion	Points
Summary of LLM prompts for at least two visualizations	40
Versions and modifications of outputs documented	10
Explanation of how prompts informed analysis	15
Reflection on LLM learning value	15
Clarity and formatting adherence	20
Total	100