

Data Engineering & Streaming

Data Engineering

Data engineering involves the processes and techniques for collecting, processing, and transforming raw data into a structured and usable format for analysis and decision-making.



- | | | | |
|------------------------------------|----------------|----------------|--------------|
| • Problem definition | • Data sources | • Processing | • Storage |
| • Permissions & rights | • People | • Validation | • Security |
| • Machine & human
usable format | • Collection | • Augmentation | • Errors |
| | • Labeling | | • Versioning |

Data Collection (Getting Raw Data)

• Sources of Data

- Sensors (IoT, IMU, environmental, medical, industrial).
- Logs (system, application, clickstream, telemetry).
- APIs (financial, weather, social media, etc.).
- Databases (SQL/NoSQL exports).
- User-generated data (images, text, labels, crowdsourcing).
- Public datasets (UCI, Kaggle, OpenML).

• Collection Methods

- Batch data capture (e.g., CSV/JSON dumps).
- Streaming data capture (Kafka, MQTT, RabbitMQ).
- Edge/Embedded collection (TinyML devices, mobile apps).

• Challenges

- Data labeling & annotation (manual, semi-automated, active learning).
- Data privacy & ethics (consent, anonymization).
- Data quality (missing, noisy, duplicated, biased).

Data Collection Pipeline (MQTT)

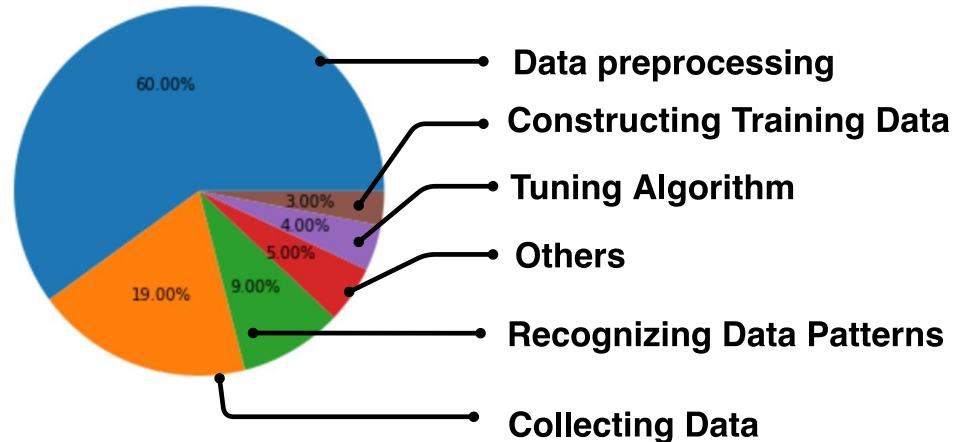
- Real-Time Data Collection with MQTT (Pub/Sub Model)
- **The Core Concept: Publish/Subscribe** Unlike traditional "Polling" (asking devices for data), MQTT uses a "**Push**" model. Devices send data only when they have it, and the server receives it instantly. It decouples the *Data Source* from the *Data Consumer*.
- **The Data Flow Pipeline**
 - **Step 1: Collection (The Publishers)**
 - Sensors (End-device) wake up and read data (e.g., Temperature, humidity).
 - They **Publish** this payload to a specific **Topic** (e.g., factory/machine_01/temp).
 - **Step 2: Transmission (The Broker)**
 - A central **MQTT Broker** (e.g., Mosquitto) receives the message.
 - It filters and routes the data to anyone listening to that specific topic.
 - **Step 3: Processing (The Subscribers)**
 - Backend systems (Databases, Dashboards, AI Models) **Subscribe** to topics of interest.
 - They instantly receive the data for storage or real-time analysis.

M5core2 to Local RPI using MQTT



Data pre-processing

- Data scientists spend more than 50% of their time on Data preprocessing.
- Collecting data is the second most time-consuming component.
- Tuning algorithm occupies a small part.



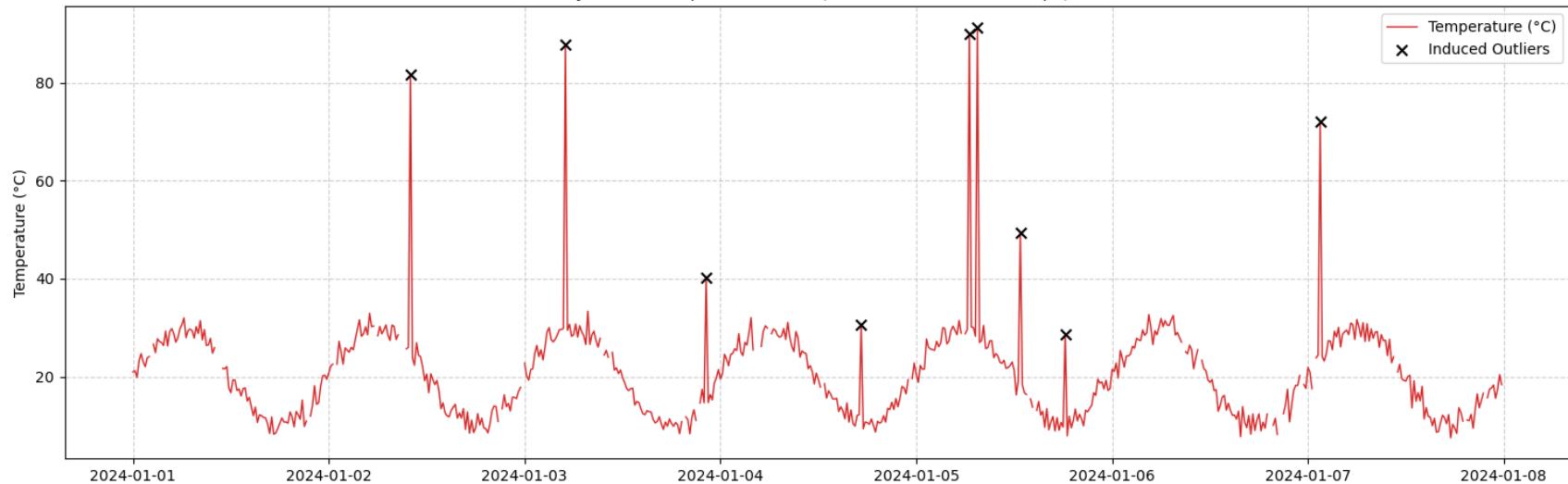
Why data pre-processing?

- **Incomplete:** data lacks attributes or contains missing values.
- **Noisy:** data contains incorrect records such as outliers.
- **Inconsistent:** data contains conflicting records or discrepancies.
- **Missing values:** some attributes in the collected data would have blank or NULL values.
- **Invalid Values:** some well-known attributes such as gender may have incorrect values.
- **Uniqueness:** repeated values of the same identifiers.
- **Misspellings:** incorrectly written values

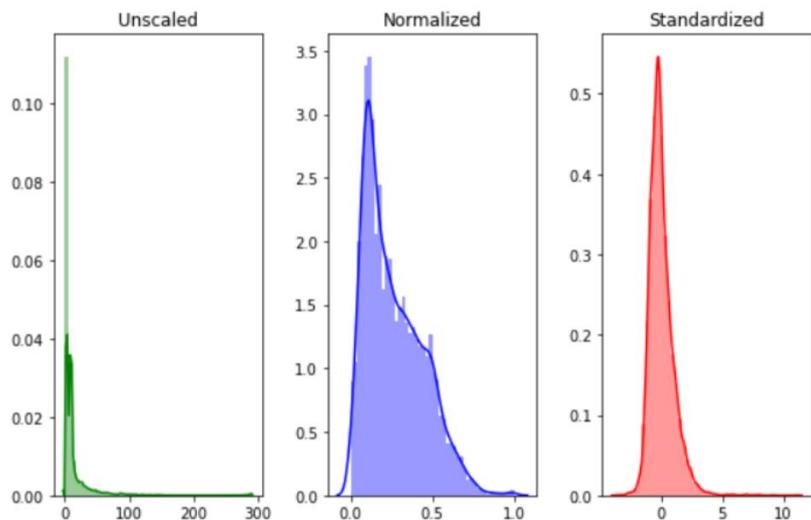
Data Pre-Processing

- **Missing Values** – Handle absent data points by removal or imputation (mean, median, interpolation, or model-based).
- **Outliers** – Detect and treat extreme values that deviate significantly from normal behavior to avoid skewed models.
- **Data Scaling** – Normalize or standardize features so all variables contribute equally to model learning.
- **Normalization** – Rescale data to a fixed range (e.g., 0–1) for algorithms sensitive to magnitude.
- **Standardization** – Transform data to zero mean and unit variance (z-score normalization).
- **Down-sampling** – Reduce data rate or sample count to lower computation, storage, or bandwidth requirements.
- **Up-sampling** – Increase sample rate or balance class distribution by duplicating or synthesizing data.
- **Noise Filtering** – Remove unwanted high-frequency or random variations from sensor or signal data.
- **Smoothing** – Apply moving average or low-pass filters to reduce short-term fluctuations.
- **Encoding** – Convert categorical data into numerical representations (e.g., one-hot, label encoding).

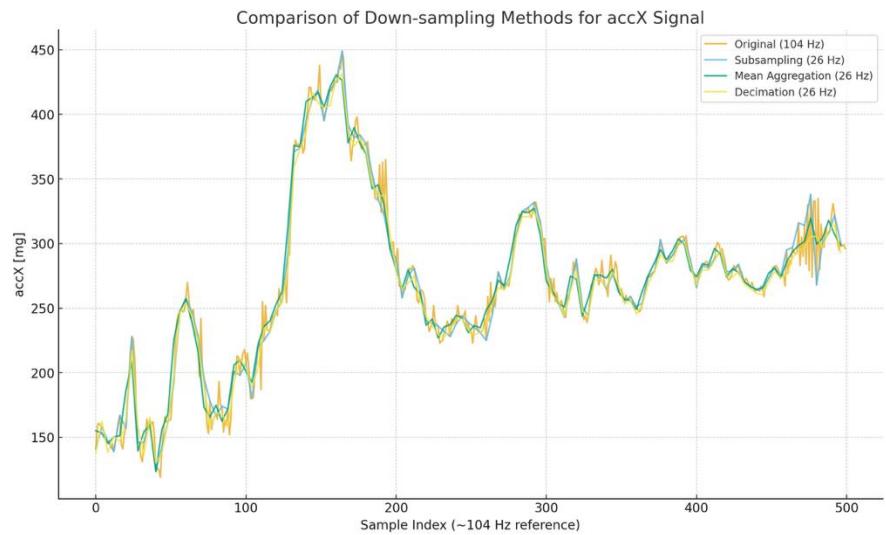
Synthetic Temperature Data (Raw: Noise, Outliers, Gaps)



Normalize & Standardize

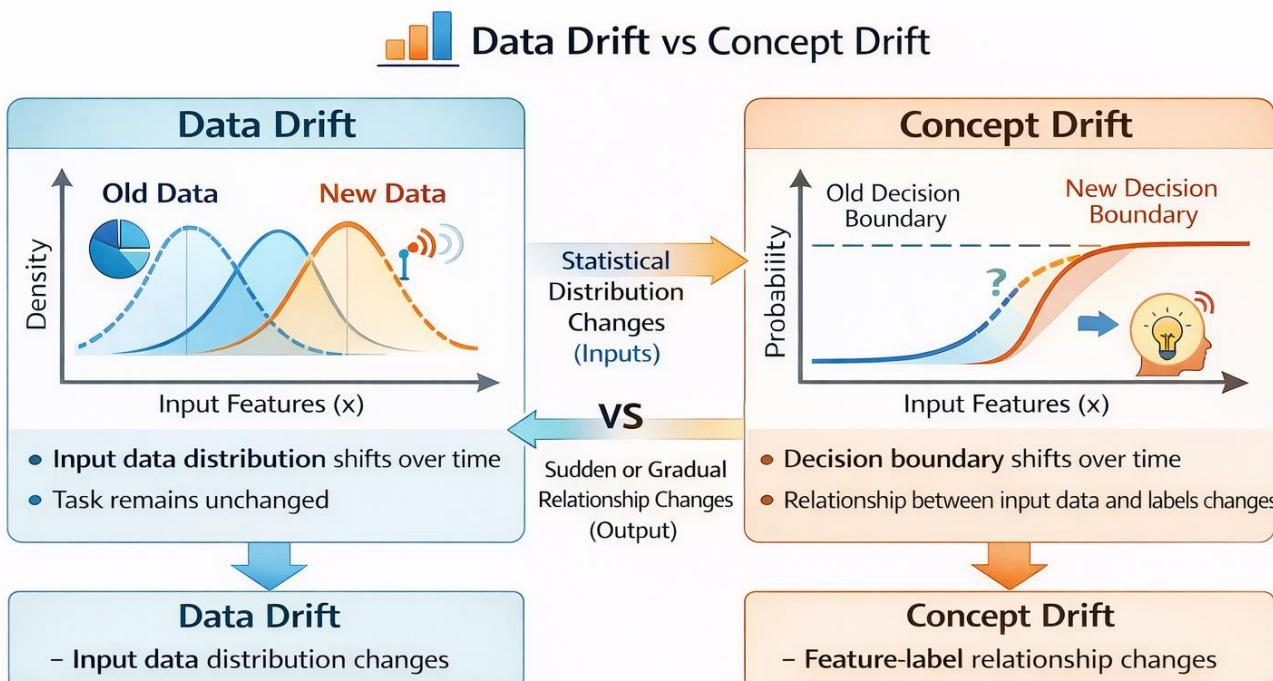


Down Sampling



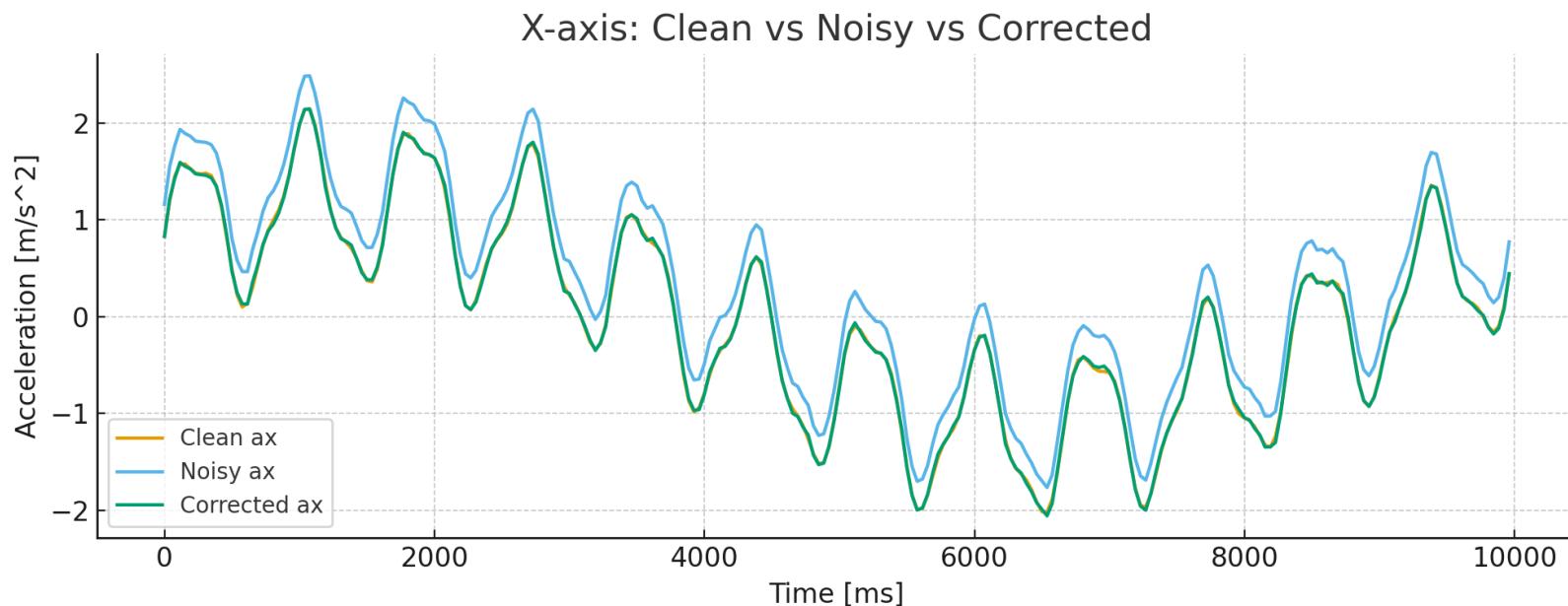
Data and Concept Drift

- **Data Drift** – When the statistical distribution of input data changes over time, even though the task remains the same.
- **Concept Drift** – When the underlying relationship between input data and target labels changes over time.



Noise Removal

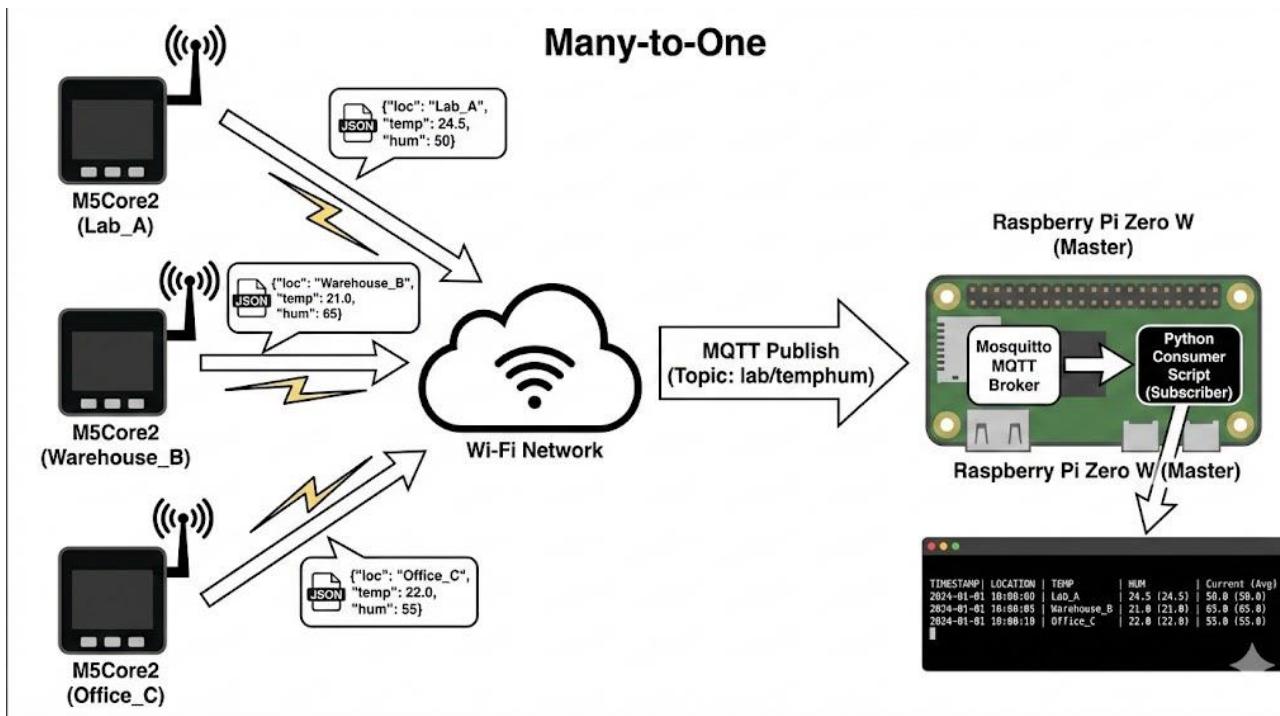
- **Apply filtering:**
 - Short-window moving average to smooth random jitter.
 - Long-window exponential smoothing to track slow bias.
 - Subtract estimated bias from raw signal.
- **Steps**
 - **Remove obvious artifacts** (e.g., spikes, out-of-range values).
 - **Denoise high-frequency jitter** (using filters or smoothing).
 - **Correct bias/drift** (subtract slow-moving average).
 - **Normalize or scale** (e.g., clip to ± 4 g if sensor range is known).



Multiple M5core2 to Master RPI - MQTT



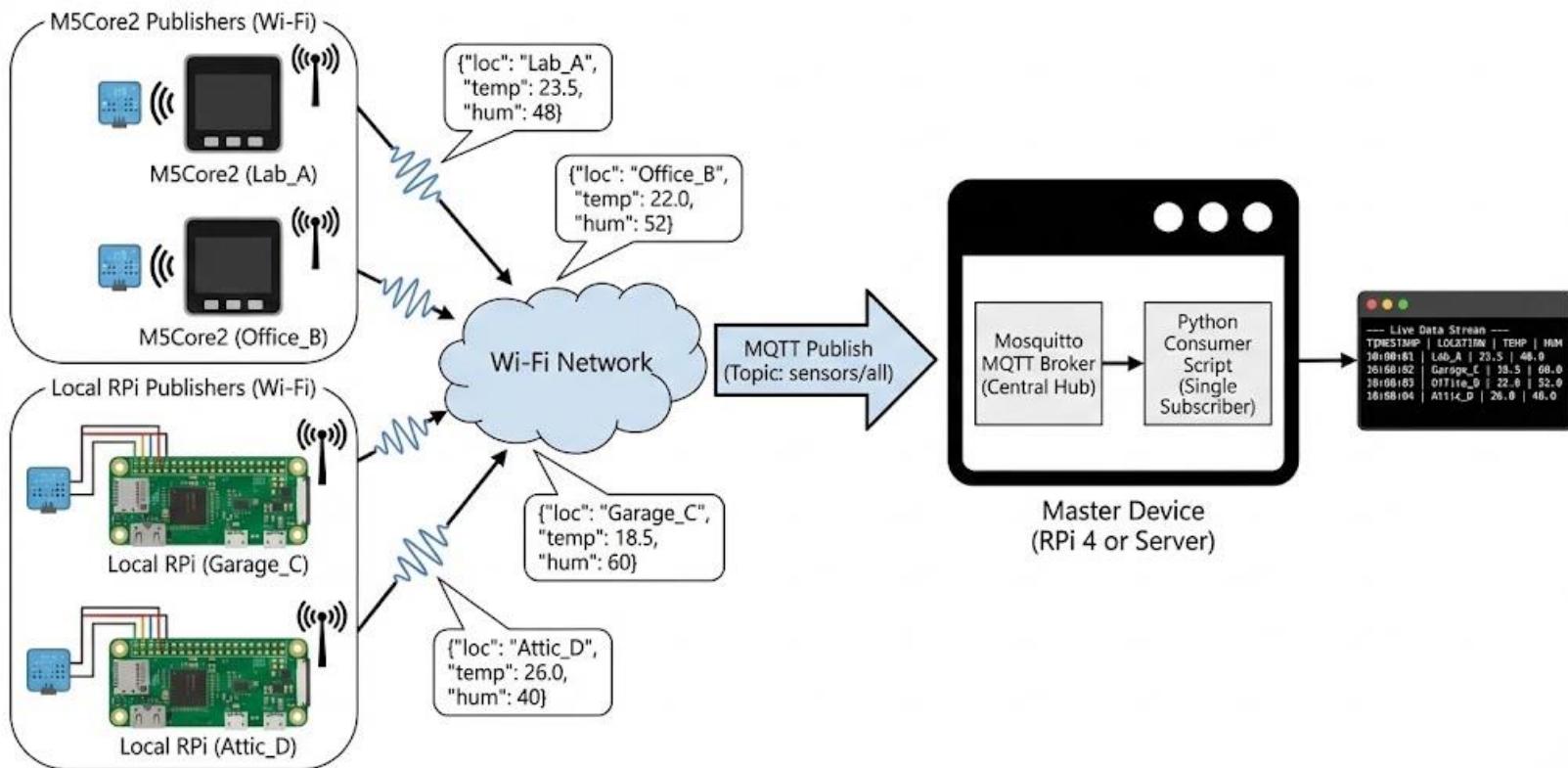
- **Left Side (Publishers):** You see multiple M5Core2 devices (labeled "Lab_A", "Lab_B", etc.). Each one is independent and sends its own unique data stream over Wi-Fi.
- **Middle (Broker):** The Raspberry Pi Zero W running the **Mosquitto** software acts as the central hub. It receives messages from all devices on port 1883.
- **Right Side (Consumer):** The Python script running on the same Raspberry Pi (or a different computer) subscribes to the topic. It receives the aggregated stream of data, filters it by the "loc" tag, and calculates the averages you see in the terminal.



Multiple M5core2 & RPIs to Master RPI

- This system represents a **Many-to-One** distributed IoT architecture. It is designed to centralize environmental monitoring from a heterogeneous network of devices (different hardware types like Microcontrollers and Single Board Computers) into a single analytical dashboard.

Many-to-One: Distributed Sensors to Master Hub



HandsOn