# IoT Challenge #3

Node-Red

Giuliano Crescimbeni, 10712403 - Arimondo Scrivano, 10712429
Politecnico di Milano

April 2025

# Contents

**1 Node-Red Flow Overview** **3**

**2 Node Description** **4**
2.1 Sender Branch (ID Generation and Publishing) . . . . . 4
2.2 Receiver Branch (Subscription and Processing) . . . . . 4

**3 Temperature Chart** **6**

**4 Ack Counter Chart** **7**

**5 ThingSpeak Channel** **7**

# 1 Node-Red Flow Overview

The developed Node-Red flow addresses the requirements of Challenge 3 by:

- Periodically generating a random ID and timestamp and publishing it to the local MQTT broker under topic `challenge3/id_generator`.
- Saving generated IDs into `id_log.csv`.
- Subscribing to `challenge3/id_generator`, computing $N = 10712403 mod 7711$ , and processing `challenge3.csv`.
- Publishing messages extracted from the CSV with rate limiting (4 messages/minute).
- Filtering and plotting temperature data in Fahrenheit.
- Saving filtered publish messages to `filtered_pubs.csv`.
- Managing MQTT acknowledgment messages and saving them to `ack_log.csv`.
- Updating a ThingSpeak channel with the ACK counter.

# 2 Node Description

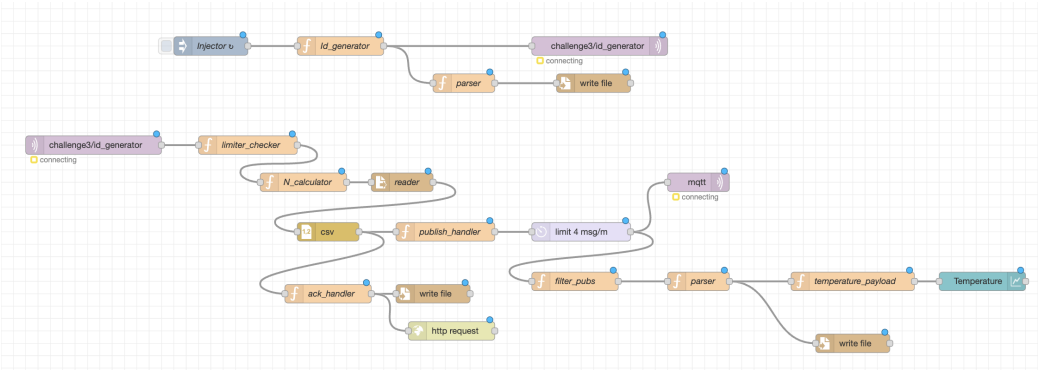


Figure 1: Node-Red Flow

## 2.1 Sender Branch (ID Generation and Publishing)

- **Inject Node (task_1)**: Triggers a message every 5 seconds to start the ID generation process.
- **Function Node (Id_generator)**: Creates a payload with a random ID and current timestamp.
- **MQTT Out Node**: Publishes the generated payload to `challenge3/id_generator`.
- **Function Node (parser)**: Converts the message into a CSV format and saves it into `id_log.csv`.

## 2.2 Receiver Branch (Subscription and Processing)

- **MQTT In Node**: Subscribes to `challenge3/id_generator`.
- **Function Node (limiter_checker)**: Ensures processing of only 80 messages.
- **Function Node (N_calculator)**: Computes $N = ID7711$.
- **File In Node (reader)**: Reads the `challenge3.csv` file.
- **CSV Node**: Parses the CSV content.
- **Function Node (publish_handler)**: Handles publish messages and creates new MQTT publications.

- **Rate Limiter Node**: Limits the publish rate.
- **Function Node (filter_pubs)**: Filters temperature messages in Fahrenheit.
- **Function Node (parser)**: Formats filtered messages for saving.
- **File Node**: Saves filtered publish messages to `filtered_pubs.csv`.
- **Function Node (temperature_payload)**: Extracts temperature values for charting.
- **UI Chart Node**: Displays the temperature values in real-time.
- **Function Node (ack_handler)**: Manages ACK messages, saving them and sending updates to ThingSpeak.
- **HTTP Request Node**: Sends the ACK count to ThingSpeak via HTTP.

## 3 Temperature Chart

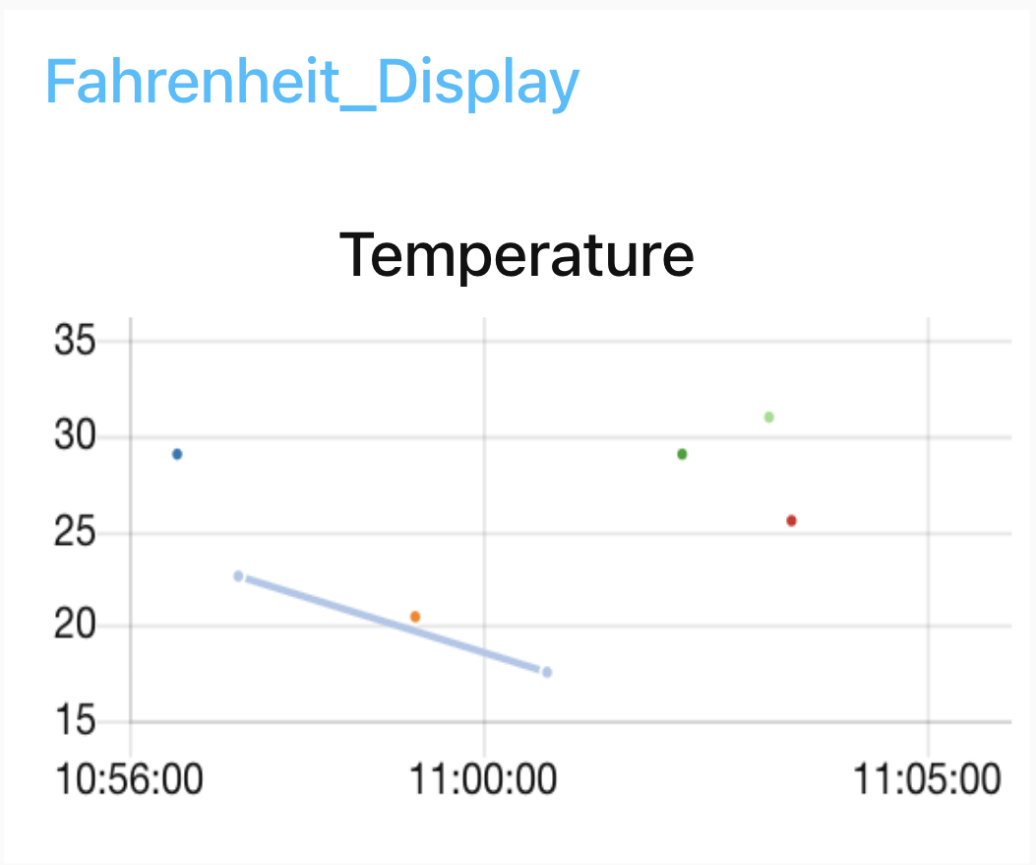


Figure 2: Temperature Chart of filtered fahrenheit values

# 4 Ack Counter Chart

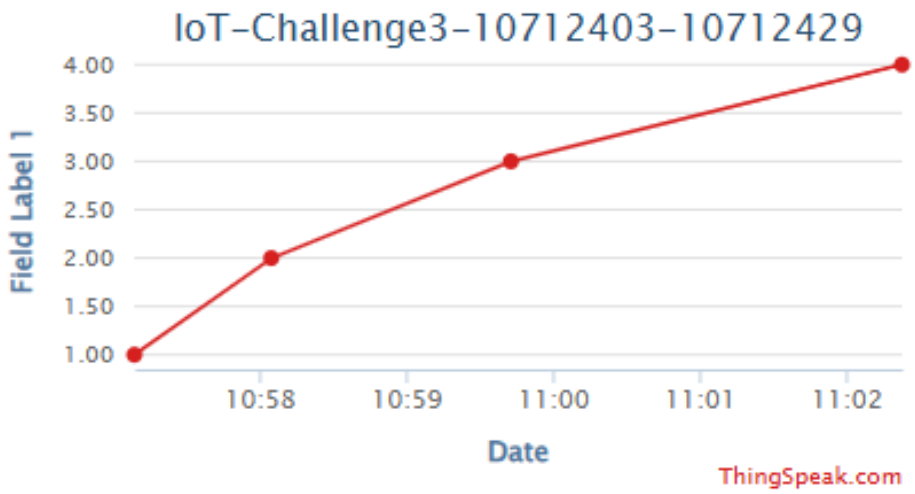


Figure 3: Ack counter chart

# 5 ThingSpeak Channel

The ThingSpeak public channel used for the ACK counter updates is available at:
`https://thingspeak.mathworks.com/channels/2924778`