Engineering Report: Integrated Sensor Data Acquisition, Processing, and Visualization System

Abstract

This report provides a comprehensive overview of an integrated system developed to acquire environmental sensor data, extract features from raw measurements, and apply machine learning for classification. The system also offers real-time logging and visualization. It is composed of four major software components that work together to deliver robust and efficient data handling from sensor interfacing to predictive analytics.

Table of Contents

1. Introduction
2. System Overview
3. Component Descriptions
    - Embedded Firmware (Arduino/C++)
    - Model Trainer GUI (Python/Tkinter)
    - Data Logger GUI (Python/Tkinter)
    - Data Processor Module (Python)
4. Integration and Workflow
5. Conclusion

Introduction

In modern sensor-driven applications, integrating data acquisition, processing, and analysis is critical. This report outlines the design and implementation of a system that collects sensor data, processes it for machine learning purposes, and provides tools for both offline and real-time visualization. The system has been developed with modularity and scalability in mind, allowing for future enhancements and adaptation to various sensor types and processing needs.

System Overview

The system is comprised of four key components:

1. **Embedded Firmware (Arduino/C++):** This low-level software interfaces with BME68x sensors, handles sensor initialization and configuration, and streams sensor data.
2. **Model Trainer GUI (Python/Tkinter):** A desktop application for loading, processing, and training machine learning models on sensor data.
3. **Data Logger GUI (Python/Tkinter):** Another desktop application dedicated to real-time logging and dynamic visualization of sensor data.
4. **Data Processor Module (Python):** A backend module that performs feature extraction, windowing of sensor data, and label encoding for machine learning tasks.

Component Descriptions

Embedded Firmware

The embedded firmware is designed to run on an BME688 board and performs the following functions:

- **Sensor Initialization and Configuration:**
  - It initializes BME68x sensors using a communication multiplexer.
  - It configures the sensors with specific temperature, pressure, and humidity settings.
  - It applies one of four predefined heater profiles to each sensor, ensuring optimal sensor operation.
- **Serial Communication and Command Handling:**
  - The firmware listens for commands (e.g., `START`, `STOP`, `SEC_`, and `GETHEAT`) via the serial port.
  - It streams sensor data in a CSV format that includes a timestamp, a label tag (modifiable via button presses), and heater profile details.
- **User Interaction through Buttons:**
  - It implements debouncing logic to accurately capture user input.
  - Single button presses adjust a label value while simultaneous presses cycle through available heater profiles.
- **Core Functions:**
  - `setup()`: Configures the system, including serial communications, sensor initialization, and setting up heater profiles.
  - `loop()`: Continuously processes commands, handles button interactions, and collects sensor data.
  - Additional helper functions manage heater profiles and signal errors via an LED indicator.

Model Trainer GUI

The Model Trainer GUI is built using Python and Tkinter, and it facilitates the following:

- **Data Loading and Processing:**
  - It uses the Data Processor module to read CSV files containing sensor data.
  - The software windows the raw data and extracts relevant features for model training.
- **Machine Learning Model Training:**
  - It trains a RandomForestClassifier on the processed data.
  - It implements sample weighting to handle imbalanced datasets.
  - Training progress is displayed using progress bars and status messages.
- **Prediction and Metrics Display:**
  - The GUI supports making predictions on new data and computes performance metrics such as accuracy, confusion matrices, and classification reports.
  - Real-time predictions can also be made by connecting to the embedded device via a serial port.
- **User Interface Features:**

    - File browsing for selecting raw data and prediction files.
    - Visual feedback through progress bars and status messages.
    - Background processing to ensure a responsive user interface.

Data Logger GUI

The Data Logger GUI provides a real-time interface for monitoring sensor data:

- **Real-Time Data Logging:**
    - It connects to the embedded firmware via a serial port and logs the incoming CSV data into a CSV file.
    - The data is parsed and stored in a pandas DataFrame with a sliding time window to display recent sensor activity.
- **Dynamic Plotting:**
    - The GUI utilizes matplotlib for live plotting of sensor parameters such as temperature, pressure, humidity, and gas resistance.
    - Users can select specific sensors and parameters, as well as adjust the time window for visualization.
- **Heater Profile Retrieval:**
    - A command (`GETHEAT`) can be sent to the firmware to retrieve and display the current heater profile configurations in a formatted popup window.
- **User Interaction:**
    - Serial port management (connect, disconnect, and refresh).
    - Interactive data display and status updates.

Data Processor Module

The Data Processor Module is a critical backend component that prepares sensor data for machine learning. Its responsibilities include:

- **Feature Calculation:**
    - It calculates statistical features such as mean, standard deviation, minimum, and maximum values for sensor data (e.g., gas resistance, temperature, pressure, humidity).
    - Additional dynamic features, including the range and slope of gas resistance values, are also computed.
- **Data Windowing:**
    - The module splits continuous sensor data into fixed-length windows (with a configurable stride), ensuring each window contains a single, consistent label.
    - This windowing process is essential for extracting meaningful features over a period of time.
- **Label Encoding:**

    - It loads an external CSV file that maps raw numeric labels to human-readable class names.
    - If an unseen label is encountered during processing, the module prompts the user to assign a new class name and updates the encoder accordingly.

- **Data Output:**

    - The processed data is output as a list of feature dictionaries or as a processed pandas DataFrame.
    - Functionality is provided to save the processed features to a CSV file for further analysis or model training.

- **Core Functions:**

    - `read_csv()`: Reads sensor data from a CSV file.
    - `process_data()` / `process_batch()`: Perform windowing and feature extraction.
    - `calculate_features()`: Applies feature functions to a given data window.

Integration and Workflow

The overall system workflow is as follows:

- **Data Acquisition:**
  The embedded firmware continuously collects sensor data and streams it over the serial port in CSV format.

- **Real-Time Logging and Visualization:**
  The Data Logger GUI connects to the firmware to log the data and update live plots, providing immediate feedback on sensor performance.

- **Data Processing and Model Training:**
  Both the Model Trainer GUI and Data Logger GUI utilize the Data Processor module to convert raw sensor data into a feature-rich dataset suitable for machine learning. The processed data is then used to train a RandomForestClassifier, and performance metrics are computed and displayed.

- **Predictive Analytics:**
  Once the model is trained, real-time predictions can be made either on previously saved data or live data streaming from the firmware.

Conclusion

This integrated system exemplifies a complete end-to-end solution for environmental sensor data management and analysis. The combination of embedded firmware, Python-based GUIs, and a robust data processing module provides a scalable framework for sensor data acquisition, feature extraction, and predictive analytics. This design not only ensures efficient real-time monitoring but also supports advanced data analysis and machine learning for accurate classification.