DOI 10.5162/SMSI2023/P49

Using established microservices for data collection of distributed sensor systems

Henrik Lensch[1], Johannes Amann[1], Christian Bur[1], Andreas Schütze[1]
[1]Saarland University, Lab for Measurement Technology, Saarbrücken, Germany
h.lensch@lmt.uni-saarland.de

Summary:
In this work, a scalable capture and visualization setup for time series data of distributed field test systems is presented. As a data source various different commercially available metal oxide semiconductor (MOS) gas sensors with their on-chip sensor-integrated manufacturer evaluation models are used as well as own calibration models

Keywords: gas sensor, data storage, field test, calibration, IoT

Introduction

Recent trends in IoT technology and big data handling led to easy-to-use open source microservices that do not require extensive knowledge in programming. These services claim a high rate of automatization for handling data, especially long-term experiments [1]. It is only necessary to connect the needed services over preconfigured APIs and attach the experiment data output, e.g. files, automated scripts, to the data input of the database. Further interconnectivity is handled automatically once this is set up. This automatization enforces a standardized data format for different experiments, since uploading and querying the database must happen in a standardized format. This is a fundamental requirement for achieving findable and interoperable data [2].

As an example, to implement and test such a software stack, we chose a long-term measurement campaign, investigating the performance of different MOS sensors for indoor air quality monitoring and their manufacturer calibration against our own, more selective, calibration.

Hardware

To test the suggested software stack, time series data of four distributed field test systems consisting of various gas sensors is used. Each system holds ten sensors operating in manufacturer mode and additional five sensors using temperature cycled operation. All sensors are operated with a self-made hardware platform [3] connected to a Raspberry Pi [4]. Each sensor system was calibrated under lab conditions [5]. Based on this data, regression models are built by means of machine learning. Due to the various functionalities of a single sensor, several environmental variables (this highly depends on the sensor but such as humidity, total VOC, air quality index, ozone level, NOx, CO2 equivalent, etc.) are calculated already on chip and are provided together with the electrical resistance, i.e. raw data [6].

A total of four systems were set up in different locations. One is located at the Lab of Measurement Technology, one in a city flat, one in a flat in the countryside, and one in an office room of a local company. Specific events, like everyday processes in the flats such as cleaning or cooking, are recorded manually to correlate them to the sensor signals.

System Design

Since the systems are distributed across multiple locations and generating a large amount of data, a high degree of automatization is needed to transfer, store, and visualize data at a centralized location. This software stack should provide the following features:

- Display significant data instantly after initial setup with focus on a user with no expertise in databases.
- Annotation option to relate events at the site (cooking, cleaning, etc.) to a gas sensor signal.
- Queryable API to download data selectively for further evaluation with MATLAB, python, etc.
- Retroactively add, remove, and identify models based on lab-calibration data to the database and compare them with already existing models and manufacturer mode.
- Option to increase the number of field test systems easily.

- Automatic alerting mechanism if a field test system or the database malfunctions.

Results

The sensor data is saved and compressed in HDF5-files every hour. For transferring the raw data file to our storage server Syncthing is used. Syncthing is a peer-to-peer file synchronization tool avoiding transferring and storing data on a 3rd party's server. Additionally, no advanced configuration on the IT infrastructure is needed. The most important feature is that it automatically reconnects if the connection to the server is interrupted without disturbing the measurement software.

On our storage server, the HDF5 files of each system are extracted and reshaped for database upload and applying the machine learning models by a python script.

The raw data is fed through one or multiple automated machine learning toolchains based on the pre-trained calibration models. The model estimates together with on-chip calculated manufacturer models are pushed to an Influx timeseries database. This is one of the most popular databases for storing timeseries data because it is easy to set up and use without any knowledge of query languages thanks to its clickable query constructor [1].

This database is connected to a visualization dashboard, which can query the database and visualize the data in a browser. For the visualization system Grafana is used due to its popularity and excellent capability for handling timeseries data. Since it is impossible to constantly track the dataflow manually, an alerting system was configured to automatically send an email to notify the administrator if any system failure occurs. Thus, downtime of the software stack and the field test systems is minimized.

To add additional machine learning models, parameter files, containing the trained model parameters and a model name, can be placed in the synchronization folder by any client. This action is recognized by the python script which automatically applies these parameters retroactively to the raw data in the backup HDF5 files. Then, the new model estimates are uploaded to the database.

Fig 1 shows the dataflow from each site to the visualization system. The blue arrows represent internal dataflow whereas the yellow arrows represent standardized API calls to an interface. The asterisk on the python script indicates that this part of the toolchain is excluded from the "no expertise" policy because a basic understanding of python might be needed.

Conclusion

The suggested software stack was easy to set up and requires almost no knowledge in programming. The server operates in this form since June 2022 without failures, whereas connectivity losses to the clients through bad connection or software crashes of the client get reported immediately.

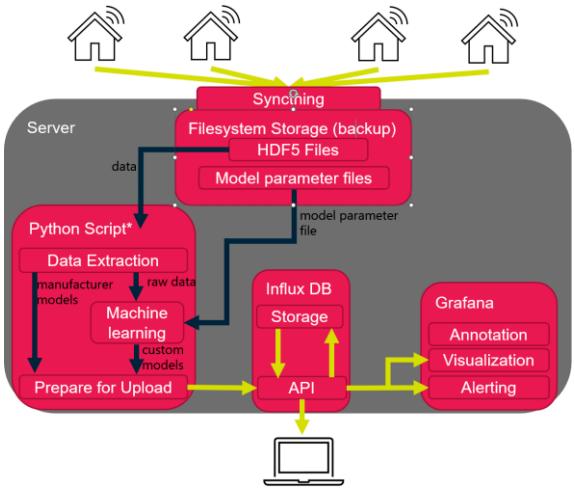

Figure 1 Schematic of the dataflow. Blue arrows indicate self-written code, whereas yellow arrows indicate standardized data transfer maintained by the tools.

References

[1] Nasar, M., et al.: "Suitability Of Influxdb Database For Iot Applications" International Journal of Innovative Technology and Exploring Engineering (2019).
[2] Wilkinson, M., et al.: "The FAIR Guiding Principles for scientific data management and stewardship." Sci Data 3, 160018 (2016).
[3] Fuchs C., et al.: "Concept and realization of a modular and versatile platform for metal oxide semiconductor gas sensors: A Versatile Platform to Measure Analog and Digital Gas Sensors" tm - Technisches Messen (2022).
[4] Amann J., et al.: "Monitoring Indoor Air Quality with low-cost Sensor Systems", Proceedings 16. Dresdner Sensor-Symposium (2022)
[5] Baur, T., et al.: "Random as mixtures for efficient gas sensor calibration" Journal of Sensors and Sensor Systems (2020).
[6] Baur, T., et al.: "Field study of metal oxide semiconductor gas sensors in temperature cycled operation for selective VOC monitoring in indoor air." Atmosphere (2021)

Acknowledgement

This research was partly performed within the project "VOC4IAQ" funded by the German Federal Ministry for Economic Affairs and Climate Action (BMWK) through the program Industrial Collective Research (AiF-iGF) under the grant number 22084N/1. The authors also thank the European Regional Development Fund (ERDF) for supporting our research within the project number 14.2.1.4-2019/1 and the German Research Foundation (DFG) for funding within project number 442146713.