The Journal of Open Source Software

TEGAT: A Lightweight and Reusable Gateway for Scientific Sensor Networks

Lars Frölich [1*], **Patrick Aigner** [1*¶], **and Jia Chen** [1¶]

1 Professorship of Environmental Sensing and Modeling, Technical University of Munich (TUM), Munich, Germany ¶ Corresponding author * These authors contributed equally.

DOI: 10.21105/joss.10887

Software

- Review
- Repository
- Archive

Editor: Abhishek Tiwari

Reviewers:

- @Puneetha-Ramachandra
- @Ahanaf-Aziz
- @brpat07

Submitted: 04 June 2026
Published: 24 August 2026

Summary

TEGAT (Telemetry Edge Gateway) is lightweight, general-purpose open source software built for managing and monitoring networks of IoT devices. It is built to integrate with the ThingsBoard platform and was designed specifically for operating distributed sensor networks for scientific research. Originally developed for and validated in the ACROPOLIS urban CO2 sensor network, it is network-agnostic and can be used with a wide range of sensor hardware. TEGAT offers a stable and reusable architectural baseline for distributed sensor networks, enabling users to focus on application-specific logic while relying on a field-tested software solution for communication, data persistence, and remote management of sensor devices. It is designed to be robust against network and power outages, crashes, and other failures, thus reducing the risk of data loss, system downtime, or the need for physical intervention. Application-specific and hardware-interfacing logic is delegated to a user-provided Controller Software which is managed and deployed by TEGAT, thus separating infrastructure and application logic.

Statement of need

Distributed sensor networks are a critical tool in scientific research and widely used across disciplines, enabling long-term, continuous sensor measurements. While they vary in the size and type of sensor hardware and data processing protocols used, such networks often face common infrastructure challenges: Physical access to sensor devices is often limited or costly, which can be exacerbated by challenges in scaling networks to large numbers of deployed sensors. Sensor networks often need to provide continuous measurements while operating unattended for extended periods of time, all while network connectivity and system power can be intermittent or unreliable. Although these challenges vary across deployments, they translate into a common set of infrastructure requirements:

- Reliable bidirectional communication
- Local buffering of data during network outages
- Remote configuration and maintenance capabilities
- Safe remotely initiated software updates
- Ability to recover from failures without physical intervention
- Seamless addition and removal of devices from the network

TEGAT addresses these infrastructure requirements to significantly reduce the engineering overhead associated with deploying and maintaining sensor networks. This enables network operators to focus on application-specific software instead. TEGAT leverages the ThingsBoard IoT platform (ThingsBoard contributors, 2025), a robust open source software, which was chosen for its maturity (more than 10 years in development) and scalability. The flexible design of TEGAT, combined with the ThingsBoard IoT platform, enables users to configure

customized sensor networks, seamlessly integrate additional sensors into existing networks, as well as making it possible to reuse infrastructure across multiple research projects.

State of the field

Other existing solutions already cover a variety of the features provided by the combination of TEGAT and ThingsBoard, though there are different tradeoffs and limitations to consider with each approach: Some solutions are designed around an on-site centralized "edge" server which collects data from multiple connected sensor devices. Examples include thin-edge.io (Thin-edge.io contributors, 2026) and ThingsBoard's own product ThingsBoard Edge (ThingsBoard Edge contributors, 2026). These architectures benefit from sensor network layouts where many sensor devices share a local network, such as large factory or office buildings, but face limitations when sensors are deployed individually in remote locations. A subset of our features can be covered with commercial solutions: For example, Amazon's AWS IoT (Amazon Web Services, 2026) and Microsoft Azure IoT Edge (Azure IoT Edge contributors, 2026) products are IoT cloud-platforms similar to ThingsBoard, and Balena Cloud (Balena, 2025) offers reliable device management and software updates of IoT device fleets similar to TEGAT's OTA and RPC functionality. However, projects building on top of such products are dependent on their future pricing and availability, and require continuous funding. Alternatively, a combination of open source solutions can offer a similar feature set: Examples are the Eclipse Foundation's Kura (Eclipse Kura contributors, 2025) and Kapua (Eclipse Kapua contributors, 2024) projects, as well as the Linux Foundation's Fledge (Fledge contributors, 2025) and KubeEdge (KubeEdge contributors, 2026) projects. In both cases, these unfortunately lack data visualization dashboards and software maturity. Finally, the Ivy project (Makowski & Chen, 2025) does not separate application and infrastructure logic, making software updates brittle. For example, any crashes not covered by the test suite may result in permanent downtime requiring on-site fixes.

Software architecture

The software is based on a three-component architecture (see Figure 1):

1. TEGAT (this software)
2. Controller Software (user provided)
3. ThingsBoard IoT Platform

Both TEGAT (1) and the Controller Software (2) are deployed on the same IoT sensor device, with TEGAT acting as an intermediary between the Controller Software and the ThingsBoard platform which runs on a remote server. This design strictly separates the infrastructure and application logic, and divides responsibilities among all three components: TEGAT (1) is designed to be lightweight and robust, performing only essential functions like forwarding telemetry to ThingsBoard and managing the deployment of the Controller Software. It communicates with the ThingsBoard platform via a secure MQTT (MQTT.org, 2024) connection. The Controller Software (2) is provided by the user and is responsible for handling application-specific logic such as controlling actuators and collecting and processing sensor data. It is deployed inside a Docker (Merkel, 2014) container environment and communicates with TEGAT via an intermediary database. Finally, the ThingsBoard platform (3) is deployed remotely and acts as a centralized data storage and network management system. It is built to be highly scalable, both in the number of connected devices and in the amount of data received and stored. It is also highly customizable, supporting arbitrary sensor data formats and protocols. Decoupling TEGAT from the Controller Software ensures that corrective actions, such as reverting to a stable software version or adjusting configurations, can be performed remotely without risking system connectivity or requiring on-site intervention. In case a newly deployed Controller Software version fails to start or contains errors, TEGAT remains

operational and continues to communicate with the ThingsBoard platform.

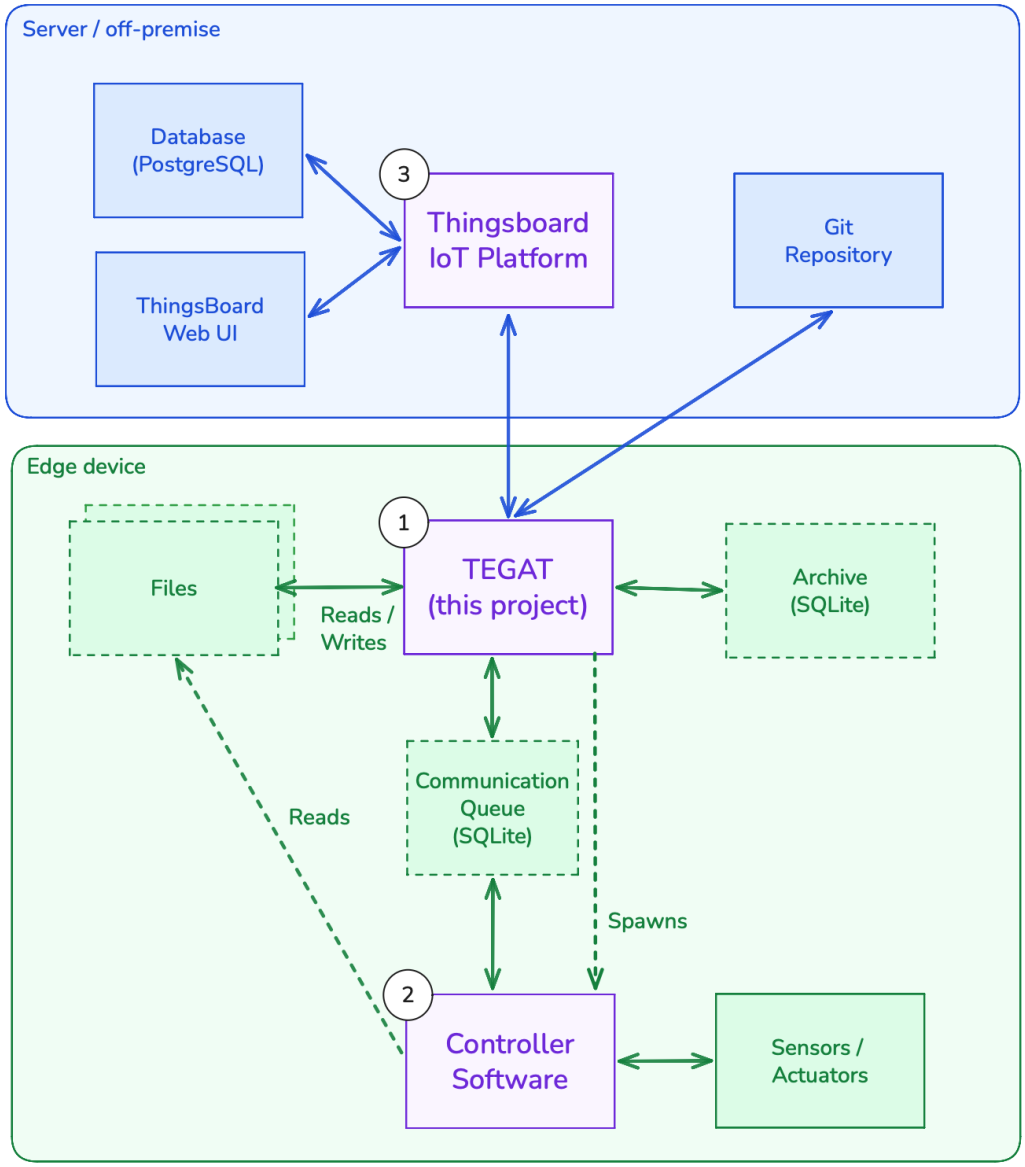

Figure 1: Software architecture for on-device (green) and off-premise (blue) components. Purple boxes show the three main architecture components TEGAT (1), Controller Software (2), and the ThingsBoard IoT Platform (3). Dashed boxes show local files.

Software design and implementation

The TEGAT software is written in Python (Python Software Foundation, 2026). It follows a modular design, encapsulating independent functionality such as logging, database access, or communication into separate software modules. During an initial setup phase, communication is established with the ThingsBoard platform using MQTT via TLS, and the device is provisioned in the ThingsBoard platform if needed. The software subsequently enters a steady-state main loop which performs one task per iteration, with higher priority tasks, such as processing incoming MQTT messages, executed first. This design ensures operational reliability and efficiency. TEGAT receives telemetry data from the Controller Software via a local SQLite database (Hipp & SQLite contributors, 2026), which is used to buffer messages between the

two software components for additional fault tolerance. TEGAT then forwards the telemetry data to the ThingsBoard platform via MQTT, and stores a copy of the data in a local database for additional redundancy (e.g., to backfill data gaps on demand). TEGAT also manages the deployment of the Controller Software through the host system's Docker daemon: If the Controller Software's Docker container is not running or has not provided a recent heartbeat, TEGAT attempts to start it using an exponential backoff strategy. Besides managing the Controller Software and forwarding telemetry data, TEGAT provides the following core features:

1. Remote procedure calls (RPC)
2. Over-the-air (OTA) updates of the Controller Software
3. Remote file management

RPCs (1) enable users to invoke one of several predefined commands on TEGAT, e.g., remotely rebooting the sensor device. This mechanism is primarily intended for operational control and diagnostics that must be executed on demand without direct access to the device. The OTA update feature (2) enables users to remotely deploy new versions of the Controller Software to the device, for example to fix bugs or add new features. By the same mechanism, users can also easily downgrade the Controller Software. This feature leverages the Git (Torvalds et al., 2026) version control system to manage the software version history: Users can specify a specific commit hash or tag. TEGAT then builds a Docker image based on the corresponding source code. TEGAT also provides a mechanism for directly creating, reading, and writing files on the sensor device using the remote file management feature (3). As Linux (Torvalds & Linux kernel contributors, 2026) systems provide extensive access to operating system functionality through files, this feature has a particularly wide range of applications. Typical use cases are managing software configuration files for the Controller Software and configuring on-device drivers and system daemons. More technical details can be found in the documentation[1], which is built on Sphinx (Komiya et al., 2025). To make TEGAT's source code more robust, its codebase is statically typed. Developers can perform local type checks using mypy (Mypy contributors, 2026), which also runs in a continuous integration pipeline using GitHub Actions. For integration testing, a demo application is provided which deploys a ThingsBoard server alongside TEGAT and an example implementation of the Controller Software. This example implementation also serves as a starting point for developers.

Research impact statement

TEGAT has been validated in the ACROPOLIS urban CO2 sensor network (Aigner et al., 2026) within the ICOS Cities project. ACROPOLIS-edge (Aigner et al., 2025) serves as an example of a successful deployment of TEGAT in a real world use case. During this deployment, it continuously ran for over 18 months across 17 devices, transmitted more than 100 million messages to ThingsBoard, deployed over 250 OTA updates across the network, and processed more than 1000 triggers from remote procedure calls and remote file management operations.

Author contributions

LF and PA designed the software architecture, implemented the software, wrote documentation and user guides, deployed and validated the software as part of the ACROPOLIS sensor network, and wrote the manuscript. JC is the principal investigator and scientific lead of the ICOS Cities project in Munich. All authors reviewed the manuscript.

[1] https://tum-esm.github.io/TEGAT/user-guide/

Acknowledgements and funding

This work has been funded by PAUL, Pilot Applications in Urban Landscapes – Towards integrated city observatories for greenhouse gases (ICOS Cities), a project under the European Union's Horizon 2020 Research and Innovation Programme (grant agreement no. 101037319). Furthermore, the work is partly supported by the Horizon Europe European Research Council (ERC) Consolidator Grant CoSense4Climate (grant no. 101089203, PI: Jia Chen).

AI usage disclosure

Generative AI tools were used to assist with language refinement, formatting, and editorial support during manuscript preparation, and to aid documentation of the codebase and online materials. All AI-assisted outputs were reviewed and approved by the authors.

References

Aigner, P., Chen, J., Böhm, F., Chariot, M., Emmenegger, L., Frölich, L., Grange, S., Kühbacher, D., Kürzinger, K., Laurent, O., Makowski, M., Rubli, P., Schmitt, A., & Wenzel, A. (2026). ACROPOLIS: Munich urban CO_2 sensor network. *Atmospheric Measurement Techniques*, *19*(2), 745–773. https://doi.org/10.5194/amt-19-745-2026

Aigner, P., Frölich, L., & Chen, J. (2025). *ACROPOLIS-edge* (Version 0.0.118). https://doi.org/10.5281/zenodo.17913263

Amazon Web Services. (2026). *AWS IoT: Unlock your IoT data and accelerate business growth*. https://aws.amazon.com/iot

Azure IoT Edge contributors. (2026). *Azure IoT Edge: The Azure IoT Edge product* (Version 1.5.39). https://github.com/azure/azure-iotedge

Balena. (2025). *Balena Cloud: The container-based platform for deploying IoT fleets*. https://www.balena.io/cloud

Eclipse Kapua contributors. (2024). *Eclipse Kapua* (Version 1.6.12). https://github.com/eclipse-kapua/kapua

Eclipse Kura contributors. (2025). *Eclipse Kura: The extensible open source Java/OSGi IoT edge framework* (Version 5.6.1). https://github.com/eclipse-kura/kura

Fledge contributors. (2025). *Fledge: An open source platform for the Industrial Internet of Things* (Version 3.1.0). https://github.com/fledge-iot/fledge

Hipp, D. R., & SQLite contributors. (2026). *SQLite source repository: Official Git mirror of the SQLite source tree* (Version a2116f86b09b23220eb9dd94ba6a54e1450f0c67). https://github.com/sqlite/sqlite

Komiya, T., Brandl, G., Turner, A., Jean-François B., Shimizukawa, T., Andersen, J. L., Neuhäuser, D., Finucane, S., Lehmann, R., jacobmason, Kampik, T., Addison, J., Magin, J., Dufresne, J., Waltman, J., danieleades, Rodríguez, J. L. C., Ronacher, A., Geier, M., … Freitag, F. (2025). *Sphinx-doc/sphinx: Sphinx 9.1.0* (Version v9.1.0). Zenodo. https://doi.org/10.5281/zenodo.18109073

KubeEdge contributors. (2026). *KubeEdge: Kubernetes native edge computing framework* (Version 1.23.0). https://github.com/kubeedge/kubeedge

Makowski, M., & Chen, J. (2025). Ivy: A data acquisition system for distributed sensor networks supporting remote configuration and software updates. *Journal of Open Source Software*, *10*(114), 8862. https://doi.org/10.21105/joss.08862

Merkel, D. (2014). Docker: Lightweight Linux containers for consistent development and deployment. *Linux Journal*, *2014*(239), 2.

MQTT.org. (2024). *MQTT specifications: MQTT is an OASIS standard. The specification is managed by the OASIS MQTT Technical Committee*. https://mqtt.org/mqtt-specification

Mypy contributors. (2026). *Mypy: Optional static typing for Python* (Version 1.20.1). https://github.com/python/mypy

Python Software Foundation. (2026). *Python 3.12.13*. https://www.python.org

Thin-edge.io contributors. (2026). *Thin-edge.io: The open edge framework for lightweight IoT devices* (Version 2.0.0). https://github.com/thin-edge/thin-edge.io

ThingsBoard contributors. (2025). *ThingsBoard: Open-source IoT platform - device management, data collection, processing and visualization.* (Version 4.2.1). https://github.com/thingsboard/thingsboard

ThingsBoard Edge contributors. (2026). *ThingsBoard Edge: Open source IoT edge computing* (Version 4.3.1.1). https://github.com/thingsboard/thingsboard-edge

Torvalds, L., Hamano, J. C., & Git contributors. (2026). *Git* (Version 2.53.0). https://git-scm.com

Torvalds, L., & Linux kernel contributors. (2026). *Linux kernel: Linux kernel source tree* (Version 7.0). https://github.com/torvalds/linux