Bidirectional integration of a KNX equipped energy efficiency test bench building with the OGC SensorThings standard

Johannes Hurka, Pavel Paulau, Jan Middelberg, Sascha Koch

Jade University of Applied Sciences Wilhelmshaven/Oldenburg/Elsfleth

Department of Civil Engineering Geoinformation and Health Technology

johannes.hurka@jade-hs.de

KNX SCIENTIFIC CONFERENCE 2023

Abstract: In order to develop and test control strategies for energy efficient building system technology, we organize the complete data flow and data storage according to the SensorThings API by the Open Geospatial Consortium (OGC). This allows an abstract, protocol-agnostic structuring of sensor data and control commands.
Beside other buildings with different systems, one test bench building with photovoltaics, heat pump and heat storage tanks will be fully equipped with KNX components.
We present the hardware and software architecture which connects the KNX installation with the central server running FROST, an implementation of the OGC SensorThings API.

Keywords: SensorThings API, Energy Efficiency, Control Strategies

Introduction: The thermal conditioning of buildings is a major constituent of the final energy consumption and contributes to resource depletion, air pollution and climate change. Due to the facts that a building typically stays in service much longer than e.g. a car, that many different building types, use cases, and heating technologies are involved, and that the feasibility of a given measure often depends on local factors such as enough space for a heat pump, the availability of a local heat distribution network, or the preparedness of the building for insulation, the necessary decarbonization of the heating sector is a highly differentiated and complex problem which calls for a large bundle of different approaches intelligently applied and combined. Accordingly, digitalization will play an important role in analysing situations and potential actions, better matching supply and demand, and predicting the most efficient approaches.
In the research project “Wärmewende Nordwest” (Heat transition north-western [Germany] [1]) we investigate the potential applications of digitalization within the heating sector on different scales from single buildings up to regional thermal energy planning. The basis for different research activities and applications of research field 3 “Digitaler Experimentalcampus Bauphysik” (Digital experimentation campus [for] Building Physics) of the project is the implementation of a bidirectional data infrastructure in order to organize sensor and status data coming from campus buildings as well as command data aimed at

the different systems in the buildings. In contrast to processing and storing all data on-site, we use a data lake approach to collect all incoming and outgoing data of several buildings in one place, allowing for multiple access and filter options regarding time, space, and type of data.
The central topic of this article covers the ongoing design considerations and development of tools to connect a KNX installation with the FROST server which is used as the central data lake. After a short overview of the building and its devices we describe the use cases defining the requirements on the data infrastructure and those parts of the OGC SensorThings standard relevant to our work as well as the specifics of the FROST Server in use. The actual solutions for data transfer in both directions are then presented and evaluated. The article concludes with a summary and outlook.

Funding acknowledgement: The work presented in this article is part of the research project "Wärmewende Nordwest" (WWNW) funded by the German Federal Ministry of Education and Research (Bundesministerium für Bildung und Forschung, Förderkennzeichen 03SF0624).

The energy efficiency test bench building (EEP): Located on the Oldenburg Campus of the Jade University of Applied Sciences, a recently built teaching and laboratory building (Figure 1) is being equipped with photovoltaics, an air-water heat pump with water storage tanks, controlled ventilation with heat recovery, controlled shading and LED lighting, as well as indoor and outdoor sensors.

Figure 1: The energy efficiency test bench building

Building materials are mostly three-pane glass and wood, with additional insulation in the outer layer.
KNX components in the building include:

- Presence and light sensors
- Temperature, humidity and CO_2 sensors
- Interfaces for electricity meters
- Interfaces for heat flow meters
- Binary input sensors
- EnOcean gateway
- Ventilation gateway
- Binary and shading actuators
- Glass push button panels
- GPS weather station
- DALI gateway

For automation and visualization, a Gira X1 (https://partner.gira.com/en/systeme/smart-home/x1.html) server is included. Furthermore, to cover the need for detailed building physics data, several sensors for temperature and humidity, air pressure, and light are attached to ESP32 wireless controllers and distributed in the building. The temperature and humidity sensors of these devices will be subject to calibration measurements in a climate chamber.
Within the research project, at least two other buildings on the campus will be connected, but as of now, the EEP is the only building with KNX components.

Use cases defining the requirements: Within research field 3 and cross-section activity 2 of the project "Wärmewende Nordwest", there are three main use cases that depend on the availability of time series of building physics data, weather data, sensor data and actuator status information, as well as actual and

predicted room occupancy information and protocolled user interactions (e.g. turning a thermostat or filing a request for more ventilation).
These use cases are:

1. Long-term monitoring and building data stories. Time series over multiple years will be used to analyse interdependencies between overall building quality, energy efficiency, and health aspects of the room conditions. For example, the energetic renovation of older buildings often drastically alters the air exchange in the rooms which can result in higher levels of CO_2 and humidity, influencing the well-being of the occupants and in the case of humidity also the substance of the building itself. The long-term monitoring will be assisted by graphical dashboards that condense the data of different time series in an interactive visualization.
   For the most significant constellations of start conditions and development of parameters, the time series will be extracted, visualized and enriched with textual information to form "building data stories" which can be used in teaching and for public information.
2. "Digital Janitor". For optimised control of the heating, shading, lighting and ventilation systems under the influence of usage patterns and weather conditions, a machine learning approach is being developed which will be fed with all available information and predictions and which is trained via reinforcement algorithms to find the best control strategies for the different systems in different buildings. The most important step will be to de-couple the software from the training buildings in order to achieve a system which can adapt to an arbitrary building and which will learn how to interpret the data coming from its sensors and how to control the technical systems. Because training cycles in real time would be too slow and could render the buildings temporarily unusable, we will also implement "digital twins" of the buildings (see [2]).
3. Education formats for sustainable development. Similar to building data stories, data subsets typical of certain situations or concepts will be curated and prepared for immersive experience in a 3-dimensional virtual representation of the energy efficiency test bench building. This will enable students, professionals, or interested citizens to explore the situation in their own pace and perspective.

For all three use cases, a reading access to the stored data is necessary. Since there are many different data types from several buildings, it is mandatory to supply an interface that lets the users and applications select and filter the data. Each data point has to be identifiable in the time and space domains and it must be documented which quantity was measured and what unity applies. To avoid manual hassles, all steps of the data access have to be easily automated.
Moreover, use case 2 also depends on the possibility to write data back into the building technology systems. For example, to efficiently utilize the PV-powered heat pump at day and switch to stored heat at night, the "digital janitor" must be able to control heat pump, 3-way valve, and circulation pump.
All of these requirements are met by the FROST Server implementation of the OGC SensorThings API, which will be described in the following section.

The OGC SensorThings API: The Open Geospatial Consortium (OGC) has defined the OGC SensorThings API as "a standard way to manage and retrieve observations and metadata from heterogeneous IoT sensor systems" ([3]). It is aimed at resource-constrained devices and makes use of JSON encoding and the MQTT protocol. It provides a structure for storing and transferring data, metadata, and geospatial information. While part 1 of the standard concerns sensor data, part 2 defines additional functionality for parameterizing actuators ([4]).

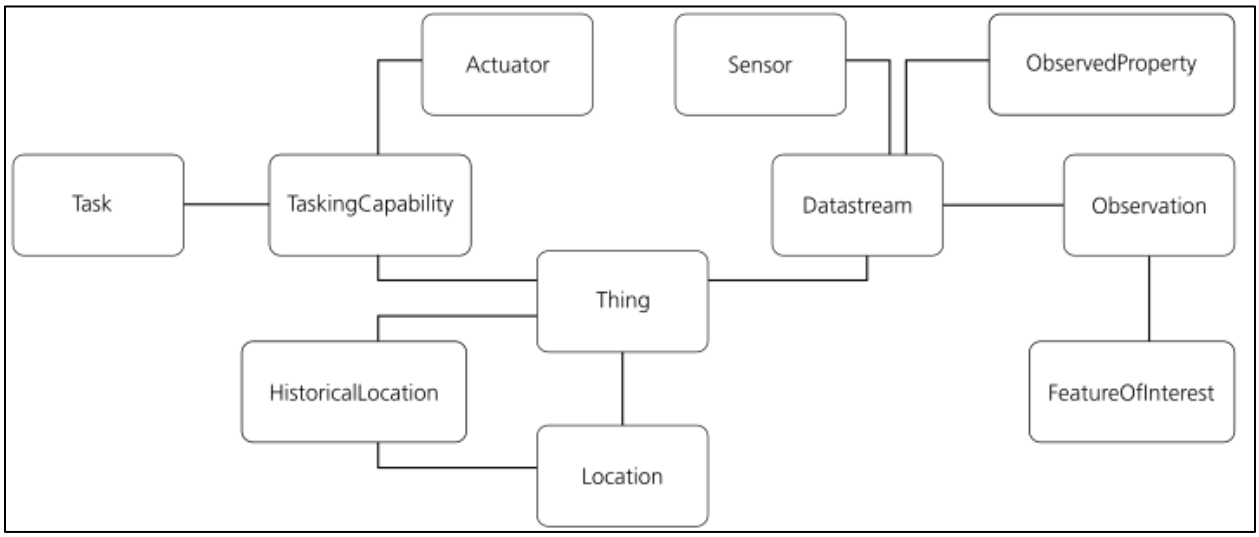

Figure 2: Simplified data schema of the OGC SensorThings API for sensing and tasking

A simplified version of the complete data structure of the SensorThings API is shown in Figure 2. The most central entities are "Things", devices that have a location and that can communicate. A thing can possess any number of "Datastreams" that each accumulate the "Observations" of a sensor. The "TaskingCapability" entity is associated with the "Actuator" part of the thing and any command sent will be stored in the "Task" table. For our work, we do not use the "HistoricalLocation" (intended for moving objects) and the "FeatureOfInterest" entities yet.

Each entity has a unique ID within its table and several mandatory fields but "Thing", "Location", "Datastream", "TaskingCapability", "Actuator", "Sensor", "ObservedProperty", and "FeatureOfInterest" allow freely defined key and value pairs in the "properties" section. This functionality is helpful to store LoRaWAN device ID or KNX group address with the entities. To avoid confusion between similar devices in the different rooms and buildings, we use the "name" fields of "Sensor" and "Actuator" for internally defined unique identifiers. For example, "Nr.042.EEP.EG.innenluft08.KNX-AQS-TH" denotes a KNX indoor sensor for temperature, humidity and CO_2 in the EEP building. It is connected to three Datastream entities with the names "Nr.042.EEP.EG.innenluft08.KNX-AQS-TH.T", "[…].H" and "[…].CO2", respectively. These identifiers are used for the data transfer solutions described in the next sections.

We use the FROST server implementation from Fraunhofer Institute of Optronics, System Technologies and Image Exploitation IOSB ([5]) of the OGC SensorThings API together with the "Actuation"-Plugin providing the tasking functionality. The server allows for HTTP communication in JSON format and for MQTT subscriptions and publishing. As of now, we use basic authentication with user name and password.

Data Transfer from KNX bus to FROST Server: Since a server component for visualization and smartphone control of the KNX system was already in demand, we use the Gira X1 server and Gira IoT REST API ([6]) to forward status and sensor values to the FROST server. However, direct communication between the systems is not possible due to different keywords, but an intermediary software component on a Revolution Pi Compact (https://revolutionpi.de/revpi-compact) acts as a callback server and translates the "event" messages coming from X1 into "observation" messages for the FROST server.

Figure 3 shows the overall setup for both sensing and tasking. For the sensing part, only the black connections are necessary.

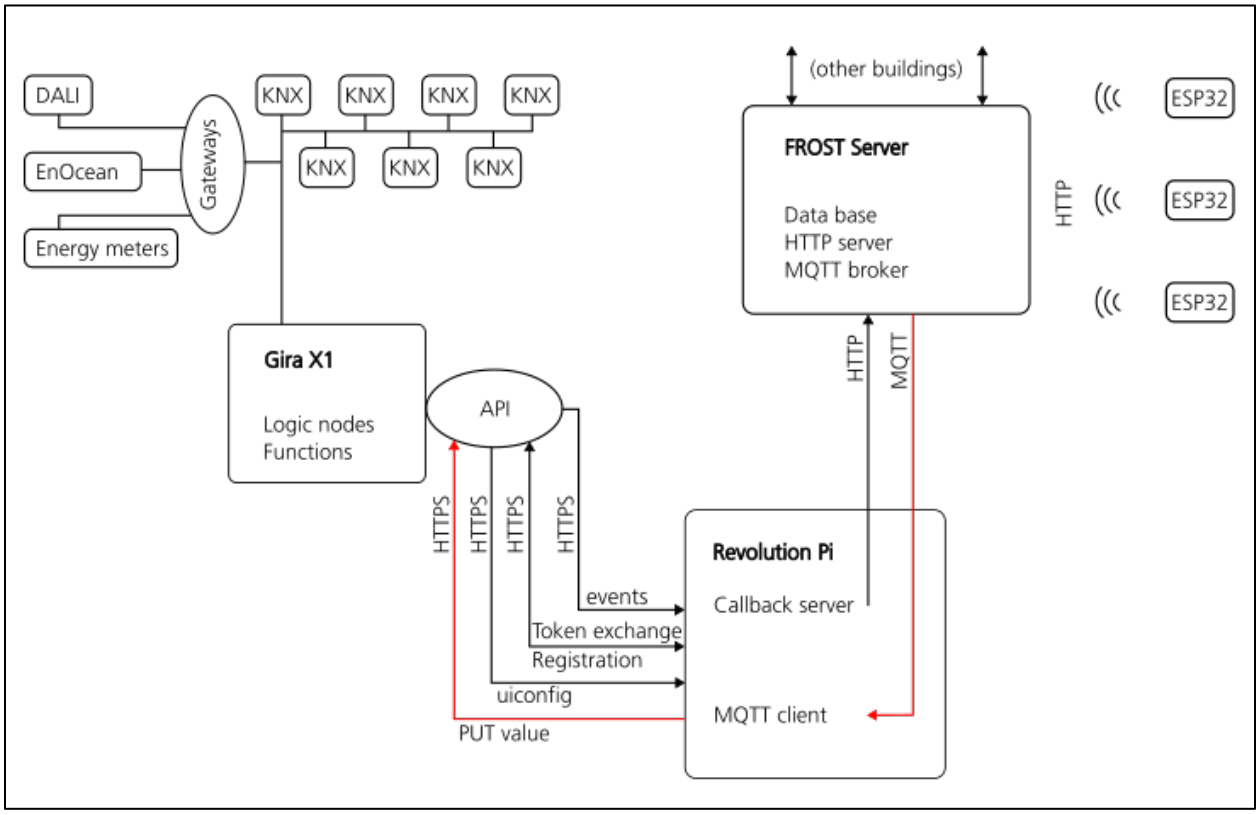

Figure 3: Hardware and software architecture for sensing and tasking. Connections in red are exclusively for tasking.

For each desired Datastream of a KNX sensor or status, the corresponding communication object is connected with a group address that gets imported into the X1 configuration and is assigned a X1 function of the type status display named exactly like the Datastream name tag. Internally, the X1 function also is given a short unique ID, for example “a003”.
The Gira IoT REST API exclusively uses HTTPS communication but accepts self-signed TLS certificates. We use the Python modules “http.server”, “json”, and “ssl” together with an own “frost” library for a Python program that

- … connects to the X1 with user name and password in order to identify itself as an API client and to receive a communication token. All subsequent communication uses this token instead of user name and password.
- … requests the latest “uiconfig.xml”, a complete description of the X1 configuration which contains among others all function names and the corresponding unique identifiers (uid).
- … parses the uiconfig and saves uid / name pairs in a Python dictionary.
- … connects to FROST server and for each name in the dictionary retrieves the ID of the belonging Datastream. These pairs also are stored in a dictionary.
- … registers itself as a “valueCallback” server for the X1. Subsequently, X1 send a JSON message to the callback server whenever it receives a KNX telegram. The message contains an “event” block consisting of a “uid” and a “value” field.

- … uses the dictionaries to match the uid to the name, and the name to the Datastream ID, for each incoming message. The "value" is then embedded into an Observation Message, which is sent to the specific Datastream of the FROST server.

This solution works in principle, but a severe drawback has to be noted:
Whenever there are several telegrams in fast succession, only one of the messages from X1 is completely received by the Revolution Pi. The other messages get truncated within the SSL handshake process and are not processed. Since many KNX devices provide more than one relevant Datastream, this problem is quite serious, as can be seen in Figure 4, where temperature, humidity, and CO2 measurement from the same device are shown over the course of eight hours:

Figure 4: Gaps in the plots of temperature, humidity, and CO2 sensor values, all from the same KNX device

This problem still has to be investigated in detail. First tests with the callback server running on much faster hardware did not show any improvement.
Because the function name is the only individually assignable field in the Gira X1 configuration process, these long and unhandy names show up on top of the functions in the smartphone interface. Together with

the fact that the interface gets cluttered with many similar functions, this makes the smartphone control of the KNX installation slightly inconvenient.

Command Transfer from FROST Server to KNX: For the tasking communication part, the FROST server does not provide a HTTP mechanism, but MQTT has to be used. However, commands can be sent to the X1 with a simple HTTPS PUT call, with the uid of the targeted function appended to the server URL. In our setup, another Python program on the Revolution Pi is dedicated to the message transfer. On top of the modules mentioned above, here also the "paho.mqtt.client" module is imported. Each KNX actuator together with the name of the corresponding "TaskingCapability" previously has to be defined in a settings file. The software

- … connects to the X1 with user name and password in order to identify itself as an API client and to receive a communication token. All subsequent communication uses this token instead of user name and password.
- … requests the latest "uiconfig.xml", a complete description of the X1 configuration which contains among others all function names and the corresponding unique identifiers (uid).
- … parses the uiconfig and saves name / uid pairs in a Python dictionary. Different from above, now the corresponding uid for a given name can be looked up easily.
- … registers itself as a client with the FROST MQTT broker.
- … subscribes to each MQTT "TaskingCapability" topic defined in the settings files.
- … parses each incoming MQTT message for the unique name of the "TaskingCapability" and matches this name with the corresponding function uid on X1, using the Python dictionary.
- … generates a HTTPS PUT request to the X1, with the correct uid in the URL and the command value in the payload.

The described solution for command transfer is not thoroughly tested yet, but first test runs worked without flaws. However, for the shading control used in the tests, a complete X1 function for smartphone access needs two group addresses attached (up / down and stop / slat). But it only gets one uid which is in our case connected with the stop / slat group address. Therefore, a "dummy" function has to be created to bind a uid to the up / down group address that will be visible but not working on the smartphone.

Conclusions and outlook: The bidirectional connection of a KNX installation with the FROST server is possible. But it can be assumed that the chosen approach is not necessarily the best solution. The decision to let the X1 serve two purposes at once – provide smartphone connectivity and the interface to FROST – leads to problems in both areas. On the smartphone side, the large number of functions (some of them not working properly) and the need for identifiable function names significantly degrade the user experience.
On the other side, the callback server mechanism over HTTPS seems to be too unreliable for the transfer of telegram series which occur quite regularly.
Also, the need for an intermediary station which translates between the X1 and the FROST server introduces additional failure potential and management overhead.
Hence, a different approach is already in the early stages of planning. It will consist of a dedicated Raspberry Pi with a "kberry" HAT (https://weinzierl.de/en/products/knx-baos-modul-838/), enabling the Raspberry Pi to be directly connected to the KNX bus. Both directions will be using MQTT communication, and there will be no need for an extra set of identifiers, since direct matches between group addresses and FROST IDs can be established.

1. References

[1] C. Rosinger and S. Lehnhoff. “Wärmewende Nordwest - Project homepage: Mit Digitalisierung die Wärmewende im Nordwesten schaffen.” https://www.waermewende-nordwest.de/ (accessed Sep. 27, 2023).

[2] E. Lesnyak *et al.,* "Applied Digital Twin Concepts Contributing to Heat Transition in Building, Campus, Neighborhood, and Urban Scale,", *BDCC*, vol. 7, no. 3, p. 145, 2023, doi: 10.3390/bdcc7030145.

[3] S. Liang, T. Khalafbeigi, and H. (. van der Schaaf. “OGC SensorThings API Part 1: Sensing Version 1.1.” https://docs.ogc.org/is/18-088/18-088.html (accessed Sep. 27, 2023).

[4] S. Liang and Khalafbeigi, Tania (Editors). “OGC SensorThings API Part 2 – Tasking Core.” https://docs.ogc.org/is/17-079r1/17-079r1.html (accessed Sep. 27, 2023).

[5] H. van der Schaaf. “FROST®-Server: An open source implementation of OGC SensorThings API.” https://www.iosb.fraunhofer.de/en/projects-and-products/frost-server.html (accessed Sep. 27, 2023).

[6] Gira Giersiepen GmbH & Co. KG. “Gira IoT REST API Dokumentation.” https://partner.gira.de/data3/Gira_IoT_REST_API_v2_DE.pdf (accessed Sep. 29, 2023).