Chair of Computing in Civil and Building Engineering
TUM School of Engineering and Design
Technical University of Munich

Software Lab:

GraphQL-Fleet-Managment-Robotics

Description

On-site robotics construction often suffers from data silos: IoT sensors publish time series, robots come with URDF models and datasheets, and semantic knowledge exists in ontologies, yet applications still need custom glue code for each new robot or sensor. This project builds a ***GraphQL API as a unified developer interface*** *for a robotics fleet plus sensor infrastructure.*

The core idea is to design a **GraphQL schema grounded in existing robotic and sensor ontologies** (e.g., an existing "Ontology Sensor" model) and expose consistent, frontend-friendly queries for robots, components, sensors, measurements, frames, capabilities, and runtime status. In addition, the project develops a pipeline to **automatically map robot datasheets and URDF files** to the schema, enabling fast, repeatable, and error-free onboarding of new robots. The final result is a working prototype that integrates real sensors and demonstrates end-to-end queries and UI integration.

Task

GENERAL INSTRUCTIONS:

- Develop GraphQL Schema based on existing Robotic **Ontologies**
- Research about **Ontologies** for IoT Sensors
- Integrate Sensors in Frontend and GraphQL Schema
- Map Robot datasheets and URDF files automatically in the schema

WP1: Requirements & Research (Semantics → API)
Collect concrete frontend and requirements (typical queries such as "which sensors are mounted on robot X", "which measurement types does sensor Y provide", "where is sensor Z mounted", "which robots are available and compatible"). Conduct a focused review of relevant IoT/sensor and robotics ontologies and identify the subset required for a practical API.

WP2 – GraphQL Schema Design (stable and developer-friendly)
Design a clean GraphQL schema that models core concepts (Robot, Link/Joint, Sensor, Measurement, Unit, Frame, Capability, Deployment, Status). Ensure the schema is stable, extensible, and does not "leak" ontology complexity into the client.

WP3 – Backend Implementation (Resolvers + Data Adapters)
Implement a GraphQL server (Node/TypeScript or Python) with resolvers that connect to ontology/data sources. Build an adapter layer so the schema remains independent of the underlying representation (e.g., RDF/SPARQL store, existing services, databases).

WP4 – Sensor Integration (IoT in the Software Lab)
Integrate real lab sensors: identities, measurement channels, units, sampling metadata, and provenance. Ensure sensor data and metadata are queryable via GraphQL, including example queries for the demo.

WP5 – Automatic Mapping (URDF + Datasheets → Schema)
Develop an ingestion/mapping pipeline that transforms URDF and robot datasheets into structured schema objects (robot structure, frames, physical limits, mounted sensors, capabilities). Heuristics are acceptable if the pipeline is robust, documented, and reproducible.

WP6 – Frontend Demo & Quality (Docs, Testing, Demo Scenario)
Integrate the API into a simple frontend (or an existing UI) to visualize the fleet and sensor setup and to execute example queries. Add basic tests (schema/resolvers) and deliver clear documentation. Prepare a final demo scenario that showcases the integration of "fleet + sensors."

Deliverables

- GraphQL schema with documentation and example queries
- Running a GraphQL server with a clean adapter layer
- URDF + datasheet ingestion/mapping pipeline, including sample inputs
- Integration of at least 1–2 real sensor setups from the lab
- A small frontend demo or integration into an existing UI
- Short report: design decisions, mapping approach, limitations, and outlook

Optional / Stretch Goals

- GraphQL **Subscriptions** for live sensor streams
- Role-based access / multi-user lab setup
- Modular schema design (e.g., federation-style decomposition)
- Mapping quality metrics (coverage, confidence, validation rules)

Recommended Skills
Solid programming skills (TypeScript/Node or Python), basic API engineering, and interest in data modeling. Ontology knowledge is helpful but not required; the key skill is translating semantic models into a usable developer API.

Supervisor

Wolf Nepomuk, Chair of Computing in Civil and Building Engineering, nepomuk.wolf@tum.de
Wrabel Tamira, Chair of Computing in Civil and Building Engineering, tamira.wrabel@tum.de

References

GraphQL specification: https://spec.graphql.org/September2025/

URDF specification: https://wiki.ros.org/urdf/XML

Ardunio: https://docs.arduino.cc/?_gl=1*1v76nlr*_up*MQ..*_ga*MTEwODMwOT-kwNS4xNzY4OTg4OTA4*_ga_NEXN8H46L5*czE3Njg5ODg5MDYkbzEkZzEkdDE3Njg5ODg5Mjg-kajM4JGwwJGgxNDA3MTYxNDEy

IoT ontologies: https://arxiv.org/abs/1707.00112