Domain Specific Language for Home Automation (several BAs/MAs are possible)

Over several years a big IoT network called SecureWSN was established and continuously expended towards a trustworthy environmental monitoring framework for constrained networks. The network itself was built up through the effort of multiple student theses and currently consists of three parts:

1. Data collection via constrained devices;
2. Gateway component handling incoming data and managing the network called CoMaDa;
3. Framework realizing backend and front-end for the end-user called WebMaDa.

A remaining issue is Home Automation that is highly configurable in a simple manner. While there are already existing solutions via various approaches those solutions are currently limited to a certain preprogrammed behavior. The automation itself is therefore barely extendable and difficult to maintain. One way to solve this issue would be the definition of a Domain Specific Language (DSL) that is specifically tailored to this use case. Using the DSL a user should have more options to create finer grained automation behavior without a need to modify existing source code or a necessity to use a programming language. Creating a DSL would require a domain analysis to define the necessary contained concepts, the definition of an abstract syntax to make it computable and also a concrete syntax to provide humans with a possibility to create such expressions in the DSL. The concrete syntax might be graphical or textual. To help with multiple steps in the development process a language workbench may be deployed. The development process would therefore consist of at least the following steps and will be adapted based on thesis type:

- **Domain analysis**: Identify necessary concepts to integrate into the DSL to enable the full range of actuators depending on environmental changes.
- **Abstract Syntax**: Define a data structure representing one unit of automation which can be modified or processed by a machine.
- **Concrete Syntax**: Create a text based or graphical representation which may be transformed into abstract syntax by a machine.

Finally, the complete solution needs to be evaluated and the report/thesis needs to be written. Further, a detailed documentation on how to use and install everything is required including how it was integrated in running instances. Depending on the results we will try to publish it on high ranked conferences and workshops together with you.

As this work is based on different works and research results, a willingness to familiarize oneself with the existing system/parts is expected. Familiarity with graphs and trees is a clear advantage and necessity for this thesis. The programming language could be anyone as long as you know how to create and manipulate graphs/trees with it. While the choice of programming language is free, a statically typed language with a possibility for explicit typing like Java, C++, Go or Rust is recommended. Functional programming languages like Haskell or Scala have good support for working with tree-like data structures.

If you are interested in this thesis, contact us and let's discuss:

- Thomas Holger, thomas.holger@unibw.de (advisor) and
- PD Dr. Corinna Schmitt (UniBw M/LMU Examiner).