Lappeenranta University of Technology

School of Industrial Engineering and Management

PERCCOM Master Program

Adekola Martins Adebayo

Specification of a Smart Meter/Actuator Description Mechanism & Development of a Mobile Application

Examiners: Prof. Karl Andersson

Dr. Jean-Philippe Georges

Supervisors: Professor Olaf Droegehorn

Professor Jari Porras

This thesis is prepared as part of an European Erasmus Mundus Programme PERCCOM – Pervasive Computing & COMmunications for Sustainable Development

Co-funded by the Erasmus+ Programme of the European Union

This thesis has been accepted by partner institutions of the consortium (cf. UDL-DAJ, n°1524, 2012 PERCCOM agreement).

Successful defense of this thesis is obligatory for graduation with the following national diplomas:

- Master in Master in Complex Systems Engineering (University of Lorraine)
- Master of Science Degree in Computer Science and Engineering, Specialization: Pervasive Computing & Communications for Sustainable Development (Lulea University of Technology)
- Master of Science in Technology (Lappeenranta University of Technology)

ABSTRACT

Lappeenranta University of Technology

School of Industrial Engineering and Management

Degree Program in Computer Science

Adekola Martins Adebayo

Specification of a Smart Meter/Actuator Description Mechanism & Development of a Mobile Application

Master's Thesis

71 pages, 25 figures, 2 appendices

Examiners: Professor Olaf Droegehorn

Professor Jari Porras

Keywords: device description, home automation, device abstraction, sensor/actuator description

Home Automation holds the potential of realizing cost savings for end users while reducing the carbon footprint of domestic energy consumption. Yet, adoption is still very low. High cost of vendor-supplied home automation systems is a major prohibiting factor. Open source systems such as FHEM, Domoticz, OpenHAB etc. are a cheaper alternative and can drive the adoption of home automation. Moreover, they have the advantage of not being limited to a single vendor or communication technology which gives end users flexibility in the choice of devices to include in their installation. However, interaction with devices having diverse communication technologies can be inconvenient for users thus limiting the utility they derive from it. For application developers, creating applications which interact with the several technologies in the home automation systems is not a consistent process. Hence, there is the need for a common description mechanism that makes interaction smooth for end users and which enables application developers to make home automation applications in a consistent and uniform way. This thesis proposes such a description mechanism within the context of an open source home automation system – FHEM, together with a system concept for its application. A mobile application was developed as a proof of concept of the proposed description mechanism and the results of the implementation are reflected upon.

ACKNOWLEDGMENTS

"If you want to fast, go alone; if you want to go far, go together". That African proverb summarily describes my experience in the course of the past 2 years studying in this Master's Program. Especially for the final 6 months of working on this research thesis, I would like to thank my supervisors, Professor Olaf Droegehorn and Professor Jari Porras for the support and guidance they availed in the course of the work. Furthermore, the teaching and instruction of the Professors at the various universities of the consortium – Lappeenranta University of Technology (Finland), Universite de Lorraine (France) and Lulea University of Technology (Sweden) have together enhanced my competencies as an ICT graduate. I'm also deeply grateful to my classmates who have now become family and are most cherished by me. The opportunity to undertake my graduate studies financed by the Erasmus Mundus Scholarship is very well appreciated. Finally, my deep gratitude to my ever supporting family as well as my loving girlfriend, Motunrayo. Their love and support saw me through the rough patches.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

HVAC – Heating Ventilation and Air Conditioning

HA – Home Automation

EU – European Union

RFID – Radio Frequency Identification

MCM – Machine to Machine Communications

ETSI - European Telecommunications Standards Institute

IOT – Internet of Things

JSON – JavaScript Object Notation

XML – eXtensible Markup Language

1. INTRODUCTION

In recent years, there has been increased interest in Home Automation (HA) systems, both commercially and within research communities. While the concept of HA isn't entirely new (Ryan 1989), the technologies on which it is built have become more economically viable and also been pushed into mainstream consciousness through the consumer electronics market. Home Automation systems are designed to provide automaton capabilities so that home owners can conveniently control their homes while being able to realize energy efficiency and cost savings. This is achieved by means of devices which either measure (meters/sensors) or control (actuators) some aspect of the home environment. These devices offer diverse functions such as occupancy detection, metering, lighting control, HVAC[1] control, video surveillance, entertainment etc.

1.1 Background

The appeal of HA systems is not far-fetched, considering the benefits for users – comfort/convenience, energy and resource savings, security amongst others. For example, (Sangogboye et al. 2015) demonstrated the return on investment for installing home automation systems in selected countries. This supports the argument in (Sangogboye 2016) that HA systems can realize sustainability objectives such as stipulated by the European Union Directive on mitigating CO_2 emissions from buildings (European Commission 2010). Among other things, the directive aims to improve the EU energy efficiency by 27% while also attaining a 27% EU share of renewable energy by 2030 as well as reduction in greenhouse gas emissions (European Commission 2014).

Despite the benefits which home automation systems offer in energy efficiency, resource conservation and so on, adoption of home automation is still very low. In Germany for example, household penetration of home automation systems is reported to be 1.21% in 2016 and projected to reach 6.23% in 2020 (Statista 2016b). Worldwide, the figures are likewise low – 0.77% household penetration in 2016, projected to reach 2.97% in 2020 (Statista 2016a).

[1] HVAC (heating, ventilation, and air conditioning) is the technology of indoor and vehicular environmental comfort. Its goal is to provide thermal comfort and acceptable indoor air quality.

There are several reasons for the low adoption of home automation systems. For one thing, vendor-supplied HA systems are costly to implement and if users cannot justify such costly investments, they're often hesitant to install a home automation system. Other reasons for the low adoption of home automation systems include the need for expert knowledge to install the systems, lack of plug and play mode for easy setup as well as proprietary devices (sensors and actuators) which make it difficult to mix and match devices from various vendors according to user preferences (Nasrin & Radcliffe 2015).

By themselves, the devices (sensors and actuators) within HA systems do not provide much value. Only when managed by software platforms can they provide valuable services to end users, for example the capability to control devices from mobile phones. In the domain of management platforms for HA, there are vendor-supplied systems such as Samsung (Samsung 2016) and NEST (NEST Labs 2016) which provide users both devices & services. Then, there are open source systems in which users mix and match the devices, gateways and other components of the system by themselves. Examples of these include FHEM, OpenHAB, Domoticz, e-domotics. Such open source platforms are targeted to end users who prefer a DIY approach to setting up their HA system.

Vendor-supplied systems provide the entire HA solution together with their own devices and compatible devices from other vendors. For example the SmartThings platform (Samsung 2016) offers a mix of devices – locks, lamps, switches, etc. while also supporting the Hue lighting bulbs by Phillips and Honeywell thermostats amongst others. However, end-users are limited in the choice of devices that can use with their HA installations. Besides, they are often more expensive to purchase than open source alternatives and the services available to users depend on the range of devices supported by the vendor. Moreover, with vendor supplied systems, users must depend solely on the vendor for updates. Open source HA systems however offer users much greater flexibility in the choice of devices which they can include in their installations.

Open source systems for HA have a general structure comprising a gateway device (which typically runs a middleware platform to handle low-level communication specifics), several smart devices having diverse communication capabilities and a management interface to control and access the home environment (e.g. a wall panel, mobile phone or even voice activated). The devices which make up HA system are sourced from different manufacturers, each with varying data representations and communication technologies.

Such diversity in the HA devices makes it quite cumbersome for application programmers to create HA solutions in a uniform and consistent manner. As it stands, for each device in a given category, they usually have to create software specifically to support that device. This implies that software development efforts are often duplicated (which is not sustainable) and that devices may not be supported in the HA system as soon as they are released to market. It is a moving target problem in which developers of HA systems are unable to keep up with the quick pace of HA device proliferation.

It is therefore imperative to specify a device description technique/mechanism which is able to describe devices in HA systems such that applications can be created in a uniform manner. Such a common description mechanism could enable richer solutions in HA systems, expanding the scenarios and utility they offer consumers and thus increase their adoption. This can be expected to lead indirectly to gains in sustainability by reducing the carbon footprint of domestic energy/resource consumption.

1.2 Statement of the Problem

Extensive work has been done in literature in the aspects of defining common device models or descriptions to foster interoperability between heterogeneous devices in HA systems. Most of the efforts in this space have been geared towards developing or extending middleware systems to realize integration between heterogeneous devices. Efforts in the area of device descriptions have not been centered on creating a common description which is abstract enough to be generalized to diverse vendors and communication technologies. The research problem therefore concerns how to specify a common description mechanism for sensors/actuators that enables their seamless integration into an HA regardless of vendor or technology.

1.3 Motivation

The motivation for this work consists in the premise that removing a common barrier in HA systems for both application developers and end users can lead to increased adoption of HA. With an increased adoption of HA systems, domestic energy consumption would be more efficient, reducing costs for the users and also reducing the impact on the environment in the way of carbon

emissions. The thesis is that if there exists a common description language/mechanism for devices in HA systems, it can be expected that:

a. Application developers can develop richer HA solutions for end users.
b. Adoption of HA systems will increase.
c. Increased adoption will realize the gains in cost-savings and resource/energy conservations.

1.4 Research Questions

This work sets out to explore the question of how a sufficient level of abstraction can be reached to bridge between the specifics of HA sensor and actuator devices from different vendors and using different communications technologies. The foremost research question is

What parameters are common to the devices which can be abstracted to describe them?

In the course of answering the above question, the following questions will be explored.

a) How can a sufficient level of abstraction be reached to bridge between the specifics of smart devices from different vendors and in different countries?

b) What device description mechanisms exist in literature?

c) Can such existing description mechanisms be adapted to this problem? If yes, how?

1.5 Research Objectives

In exploring the answers to the defined research questions, this thesis aims to fulfill the following two objectives.

a. To specify a common description mechanism for smart devices (meters/actuators) which enables seamless integration of devices from different vendors and countries. The description mechanism to be proposed must be such that when implemented eases the nature of end user's interaction with the home automation system and also provides developers a common model/interface to which they can consistently develop home automation solutions.
b. To develop the description mechanism into a computer readable format which will be used to implement a proof of concept that integrates previously unknown devices. The eventual

computer readable format shall be used in the proof of concept which demonstrates the potential of the description mechanism to realize the set out objectives.

1.6 Expected Results & Outcomes

The work is expected to contribute the following in the way of results and outcomes.

1. A description mechanism /technique/method which applies to heterogeneous devices in a smart home environment.
2. A representation of the description technique in computer/machine readable format.
3. A mobile application as a proof of concept which incorporates the description technique and is able to access/integrate previously unknown devices.

1.7 Delimitations

The thesis is set in the context of open source HA systems which have technology specific interactions for devices. In particular, FHEM[2] will be the system of interest given its flat and modular architecture, its support for a wide range of communication technologies as well as the active community of developers and hobbyists who will benefit from a common description mechanism. It is not within the scope of the work to explore issues with usability, information modeling and ontologies for home automation.

1.8 Structure of the Thesis

The rest of the thesis is organized as follows. Chapter 2 presents related literature bordering on communication standards within the home automation domain, existing device descriptions/standards as well as previous approaches of addressing integration of heterogeneous devices. Chapter 3 discusses the methodology for specifying the common description and a system concept for its use, chapter 4 presents the implementation of a proof of concept while Chapter 5 concludes the work and highlights further directions of work.

[2] FHEM is an open source Home automation server.

2. LITERATURE REVIEW

Relevant works have been grouped under three headings – 1) Fundamentals of home automation systems (including common communication technologies) 2) approaches for heterogeneous device integration and 3) existing sensor description standards. The review is intended to highlight a research gap, evaluate previous approaches and where appropriate, extend or take cues from previous work.

2.1 Fundamentals of Home Automation Systems

An Automated Home is one in which technology has been applied to improve living conditions for residents by means a set of services and features such as energy monitoring and conservation, security, convenience and fire prevention amongst other things (Withanage et al. 2014).

A principal use case for home automation systems is in energy saving given that it enables detailed monitoring of energy consumption of devices within the home as well as easy monitoring of the home. Security, convenience and comfort are other compelling benefits of home automation systems (Gonnot & Saniie 2014).

While the architecture of specific HA implementations vary, they all share some critical common elements as depicted in figure 1 below.

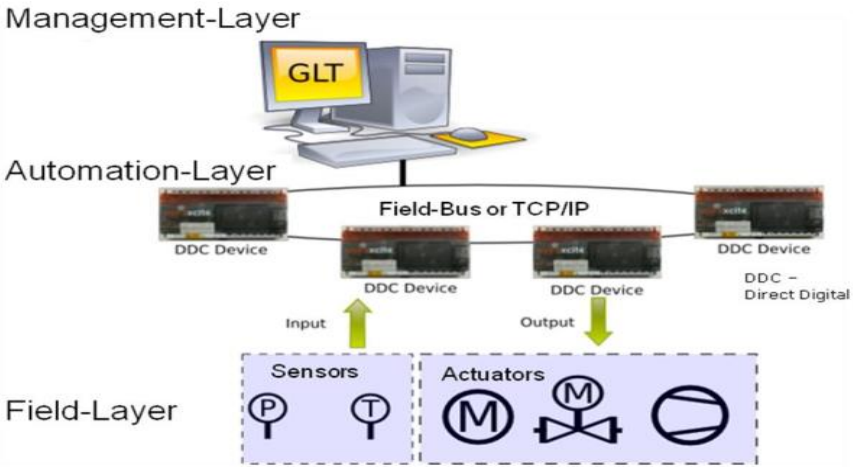

Figure 1: Generic HA Architecture (Sangogboye 2016)

At the field layer, there are sensors and actuators which are chosen according to the requirements of the home automation scenario. The mix of the field-layer devices influence the structure of the automation layer. Within the field layer, there is an enormous variety in the wired and wireless protocols in use and this presents a wide range of possibilities to end users regarding which technologies to adopt for their home automation needs. The communication technologies adopted in home automation applications fall into wired (KNX, X10, LoneWorks, ModBus) and wireless technologies (FS20, HomeMatic, EnOcean, OneWire, Zigbee, X10, KNX/EIB among others).

The role of the Management layer in the HA architecture is to manage this variety and enable end users control devices transparently. Often, vendors such as Siemens, Bosch and others bundle these three layers into their consumer solutions. This has the advantage that the management layer is well tailored to the specifics of the included products. However, challenges arise when the end user wishes to include devices from other vendors or those of a different technology into the system. In such scenarios, users are confronted with the question of how to monitor and control such disparate devices. There are different approaches to this as discussed in (Sangogboye 2016). As they argued, a most suitable solution would be to have a management layer which is able to integrate several automation and field-layer systems into one single and manageable system. Open source solutions/platforms are well suited for this requirement since they stretch beyond the boundaries of a single vendor system (Sangogboye 2016). Amongst the different open source systems for home automation, this thesis was done in the context of the FHEM platform. Among other reasons for choosing it is the fact that it has a simple architecture - a single main core as an event loop and other modules to handle different tasks (Sangogboye 2016). While FHEM supports a varied set of technologies, the most commonly deployed ones are FS20, EnOcean and HomeMatic especially in Germany where the project originated. Hence these technologies are the focus of this work and they are discussed further in subsequent paragraphs.

2.1.1 EnOcean

EnOcean pioneered the idea of energy harvesting for self-powered home automation elements. The solutions enabled by EnOcean include self-powered wireless switches, sensors, and controls. It also supports wireless LED controls with energy harvesting wireless switches & sensors,

controllers and tools. Each EnOcean device has a unique 32-bit ID as part of every transmitted message so as prevent activation of the wrong components (Torbensen et al. 2011). Every datagram is short and transmitted thrice with random delays and timing offset that's intended to minimize chances of 50Hz and 60Hz interference. Besides, EnOcean has communication profiles to enable interoperability such as profiles for temperature, humidity, CO_2 sensors etc. (Torbensen et al. 2011).

2.1.2 FS20

FS20 devices communicate wirelessly in the ISM (Industrial, Scientific & Medical) frequency band using a proprietary communication protocol. In terms of the data format, at the data link layer a message is comprised of between 5 to 6 bytes such that the first 3 bytes contain the address, the 4^{th} and optional 5^{th} byte contain the actual command and then the 6^{th} byte is a checksum. The specific application scenario determines what commands are encoded in the 2 command bytes. FS20 has no cryptographic security built in. Being a simple protocol, most FS20 devices support only unidirectional communications (i.e. they can only send or only receive data but not both). This implies FS20 systems having a low reliability since the devices can't acknowledge receipt or execution of commands (Gauger et al. 2008).

2.2 Integration Middleware/Approaches

Existing attempts in heterogeneous device integration as reported in (Paz-Lopez et al. 2012) can be broadly grouped along the lines of:

a. Abstracting functionalities as interoperable OSGi services.
b. Developing control gateways in charge of providing uniform access to different domotic technologies.
c. Virtual Sensor Definitions (e.g. GSN)

2.2.1 Global Sensors Network (GSN)

Global Sensors Network (GSN) presented in (Aberer et al. 2006) was introduced to address two major challenges in the context of sensor internet (a.k.a Internet of Things). First, it's costly to develop and deploy heterogeneous sensor devices in large scale systems and secondly, data-oriented differences between the various sensors were significant. In spite of these differences, the requirements for processing, storing, querying and publishing data are similar and as such can be abstracted into a common framework. Hence, GSN provides a uniform platform to enable fast and flexible integration & deployment of heterogeneous sensor networks. GSN's key abstraction is the concept of a virtual sensor. A virtual sensor abstracts from the implementation details of accessing sensor data. Such a virtual sensor corresponds either to a stream of data received directly from sensors or a data stream which has been derived from other virtual sensors. From an input/output point of view, a virtual sensor may have an arbitrary number of input streams but it has only one output stream. The specification of a virtual sensor provides all the details required to deploy and use it. Such information includes among other things:

- Metadata used for identification and discovery.
- Structure of the data streams which the virtual sensor consumes and produces.
- An SQL-based specification of the stream processing performed in a virtual sensor.
- Functional properties related to the persistency, error handling, life-cycle management, and physical deployment.

A declarative deployment descriptor which contains these properties of the virtual sensor makes it possible to realize rapid deployment.

A fragment of a virtual sensor description is shown in Figure 2. It contains the definition of a sensor which returns an average temperature value from a remote virtual sensor that's accessed via the internet through another GSN instance (Aberer et al. 2006).

```
<life-cycle pool-size="10"/>
<output-structure>
<field name="TEMPERATURE" type="integer"/>
</output-structure>
<storage permanent-storage="true" size="10s"/>
<input-stream name="dummy" rate="100">
    <stream-source alias="srcl" sampling-rate="1" storage-size="1h">
        <address wrapper="remote">
            <predicate key="type" val="temperature"/>
            <predicate key="location" val="bc143" />
        </address>
        <query> select avg(temperature) from  WRAPPER </query>
    </stream-source>
    <query>select * from srcl</query>
</input-stream>
```

Figure 2: GSN Descriptor File (Aberer et al. 2006)

GSN reduces the deployment of sensors to the task of simply writing an XML file which describes properties that are dependent on the specific sensor (Aberer et al. 2006). It uses the IEEE1451 standard for automatic detection, configuration, and calibration. IEEE1451 compliant sensors have a Transducer Electronic Data Sheet (TEDS) which is stored within the sensor and is what provides a simple semantic description of the sensor including things such as the sensor's properties and measurement characteristics (type of measure, scaling, calibration etc.). This thesis takes a cue from the GSN virtual sensor concept to propose a generic device concept which captures the essential aspects of an HA device.

2.2.2 Cooperative Middleware Platform as a Service – (COMPAAS)

The authors of (Amaral et al. 2015) set out to develop a middleware which doesn't constrain application developers and end users alike by which and how physical devices are implemented or deployed in target environments and provides standard abstractions for both device registration and interoperability as well as for the provision of high-level and cooperative services to applications, thus making IoT programming as platform independent as possible.

Their proposed system - COMPAAS was based on the specifications of the EPCGlobal for high-level service provisioning of RFID middleware interfaces. It also provides a light weight system architecture based on the ETSI[3] specifications for M2M[4] services platform.

COMPAAS is based on two main cooperating systems – Middleware and Logical Resource. Middleware abstracts the interactions between applications and physical devices while hiding the complexity involved in these activities. Logical resources however abstract the functionality of these devices (Amaral et al. 2015). Logical resources are described by means of system profiles. A system profile contains attributes which characterize the physical device. It includes attributes such as name, manufacturer, function, model, data type, URI[5] and so on. These attributes are then used by applications to locate the desired resources. In this thesis, the concept of profiles which define name, manufacturer etc. has been adopted in developing the attributes characteristic of devices

2.2.3 GSN + Firebase + JESS

The authors in (Boman et al. 2014) based their work on GSN + Firebase[6] + JESS (Java-based rule engine), to create two applications - a web application that allows a user to visualize the status of the physical devices monitored by sensors as well as a generic notification application which is powered using the rule engine. The main idea was to empower the IoT consumer to be able to add as many sensors as they desire while being able to consume the resulting data in a flexible manner.

[3] European Telecommunications Standards Institute. A European Standards Organization developing World Class Standards in Europe for global use.
[4] Machine to Machine refers to direct communication between devices using any communications channel, including wired and wireless
[5] Uniform Resource Identifier (URI) is a string of characters used to identify a resource. Such identification enables interaction with representations of the resource over a network, typically the World Wide Web, using specific protocols.
[6] Firebase is a cloud storage service they leveraged to enable the middleware be decoupled and interface with third party applications

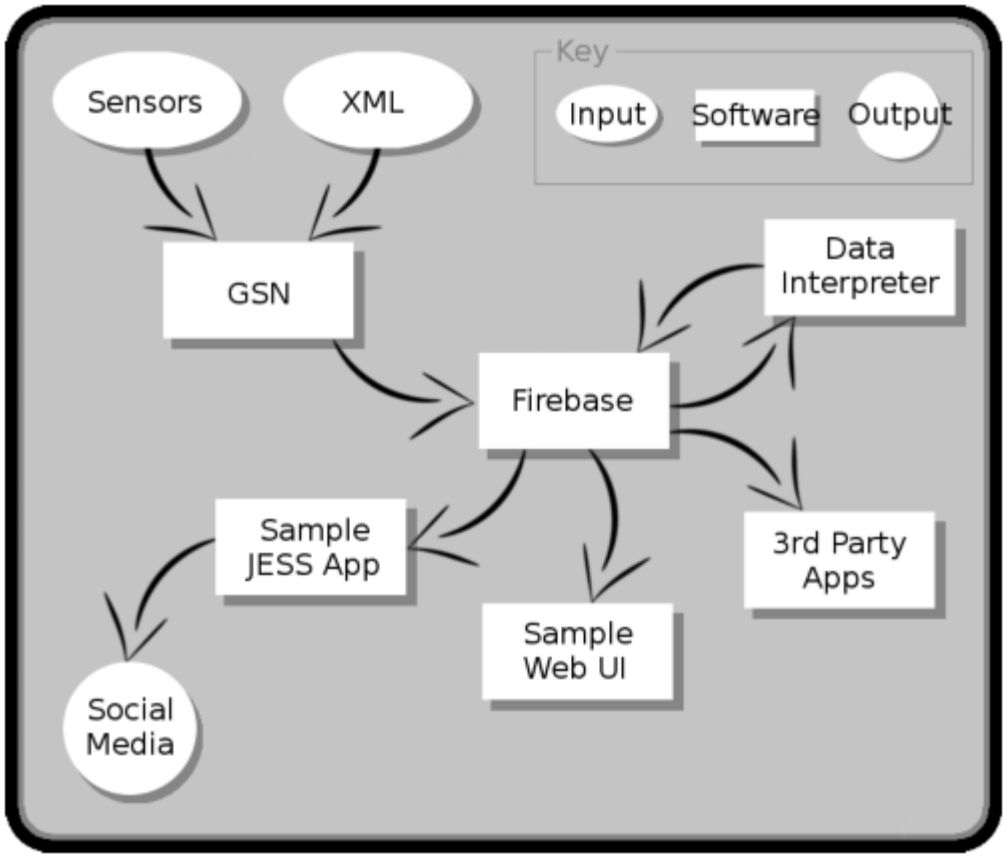

Figure 3: Architecture for GSN + Firebase + JESS

For their implementation, they made use of GSN and Java for building out the middleware, Firebase as the storage system for the IoT data instead of simply storing it in the default SQL database which GSN uses. A data interpreter was developed which pulls in data from Firebase and uses it to create deductions about the state of the sensors. Using GSN as their base device description, the system requires the sensors and an XML file created by the user to describe the main properties of the sensors. In the XML file one would specify the virtual sensor class which GSN will use to represent the sensor as well as the name of the wrapper class it'll require for connecting to the sensor and parsing the data. They also implemented a data interpreter which interfaces with the firebase data store. The data interpreter requires six values for each separate numeric value that will be generated by the sensor. The first field is a unique ID, second is the device name which the sensor is monitoring, be it an actual device (e.g. a microwave or a something in the apartment such as the doorway). Finally, the last four values represent the trigger values for the device i.e. they're used by the data interpreter to determine if the device is on or

off. A cue was taken from these authors to propose in this thesis the use of a mapper layer which is able to provide appropriate commands for particular technology types.

2.2.4 Seamless integration of Heterogeneous Devices and Access Control in Smart Homes

The authors in (Kim et al. 2012) present an architecture for smart homes which is based on the OSGi framework and seamlessly integrates heterogeneous protocols and diverse device types such that end users can add new devices as they wish, irrespective of the discovery mechanisms of the specific communication protocols. They were able to propose an abstraction layer based upon a simple semantic domain model. Compared to existing smart home architectures such as DOG (Bonino et al. 2008) and Hydra (LinkSmart Open Source Team 2014) which similarly try to integrate heterogeneous device types, the authors have proposed an end to end workflow for device discovery which enables plug and play of heterogeneous devices at runtime, extensibility to previously unknown devices, as well as the use of a barcode reading feature with smart phone cameras to improve the system's usability. At the center of their proposed system is an abstraction layer based on a simple semantic model (DogOnt) and implemented using the OSGi framework.

Their prototype consisted of an actual home gateway, end devices and a smartphone as a user interface. It achieved discovery of selected new devices and was able to integrate services based solely on semantic information. The integrated devices were based on various communication protocols such as X10, Insteon, Zigbee, UPnP and they included sensors (motion, water leak), on/off adapters for home appliances, and dimmer lamp adapters.

The flow during use starts with the user connecting X10, Insteon and Zigbee controllers via USB to the home gateway. The gateway detects that new controller devices have been attached and thus creates new controller device objects. The user can proceed to use their smartphone to turn on and off devices of a certain class (Light, Entertainment) located in a certain room. This thesis will attempt to realize a similar usage flow for testing the proof of concept. i.e. accessing the target devices transparently via the gateway.

2.2.5 Unified Data Model for Wireless Sensor Networks

The authors in (Nachabe et al. 2015) present a unified model for data representation in Wireless Sensor Networks and modeled among other things – the evolution over time of both energy availability and energy consumption of sensor units, properties related to the node itself (which change over time), properties related to the node which do not change over time, operational state of the node (transmission, sleep), the process for obtaining the sensor measurements (things like input, outputs, description, calibration, drift, latency, unit of measure precision, and response order attributes).

They re-used basic attributes from previously proposed solutions and by introducing semantic knowledge, their proposed ontology could be integrated with existing technologies to provide a solution for WSNs/Sensors data heterogeneity management. The authors conducted a symbolic pre-validation of the open data model and ontology. In the test scenario which involved a runner with a mobile app that reports vital data to a server, the server creates a new WSN having the user as the contact using the mobile cluster. When the user turns on the H7 polar (heart rate sensor), the mobile detects its presence and informs the server that a new cluster has been created and elected the mobile as the sink. They were able to successfully accept other Bluetooth LE enabled sensors from any vendor without reprogramming mainly due to the custom-built MyOntoSens ontology. Their results show that being able to define a generic ontology covering the most important features of WSNs not only provides a solution for interoperability and heterogeneity management, but also allows the reduction of power consumption of the WSN. The authors justified the use of JSON as a format for data transfer, namely lightness in processing and its non-intensive bandwidth utilization. This thesis will consider the use of JSON versus XML as a format to represent device descriptions.

2.2.6 SENSEI – Modeling of sensor data & context for the real world internet

The authors in (Villalonga et al. 2010) note that the problem of bringing real world information into the internet has been studied from two perspectives, namely Sensor & Actuator networks (middleware as well as frameworks for deploying sensor based geo-dispersed services) and Context awareness. They built the SENSEI platform around the concept of resources – Sensors,

processors or actuators which provide information about the real world or allow interaction with it. This resource concept and resource descriptions provide a basis for homogenous discovery and access to heterogeneous substrate of SANs.

Resource Descriptions (RD) describe the resource in terms of the information it provides, the task it performs and how to access it, hence it consists of Resource Access Interface (RAI) and Resource Endpoints (REPs). SENSEI offers 3 layers of the information model:

a. Raw Data: the lowest layer of the information model which contains the value that has been observed or measured by a resource.
b. Observation & Measurements: this layer interprets the data sensed from the physical world, for example interpreting a temperature sensor value of 25.5 as temperature reading in Celsius. This is realized by introducing metadata. Therefore, this layer contains the sensed value as well as zero or more metadata parameters which define the value. (It is in this way similar to SWE data model and semantic sensor web).
c. Context information: deals with the modelling of context information about real world entities required by context framework.

2.2.7 Parameter-Based Mechanism for Unifying User Interaction, Applications and Communication Protocols

In (Song et al. 2014), the authors introduce the concept of a Parameter as a means to consistently exchange information between users, devices and applications in smart home context. They have outlined the following aspects to be modelled about devices.

a. Basic information such as name, category, unit etc.
b. Status information – last value and last value time. These are used in conjunction with the rule engine to set off relevant triggers.
c. Actions – whether the parameter value can be changed and if so, by who (user or application). This also concerns the appropriate data format that should be passed to the device (e.g. Boolean for switch and integer value for a heating system).
d. Access Type – deciding if the user/application is able to access the real time parameter value. Realized with the ParameterAccessType attribute (Request, Event, Programmable).

e. Request Frequency – sets the time interval between two requests, to prevent the problem of excessive requests.
f. Validation – regex-based means of ensuring that all values set by the user/application/driver follows the appropriate format and are within the correct range.

2.3 Sensor Device Description Standards

In the domain of sensor device descriptions, there are several existing standards developed for different purposes and catering to different aspects of the problem of interfacing the physical world with the digital world.

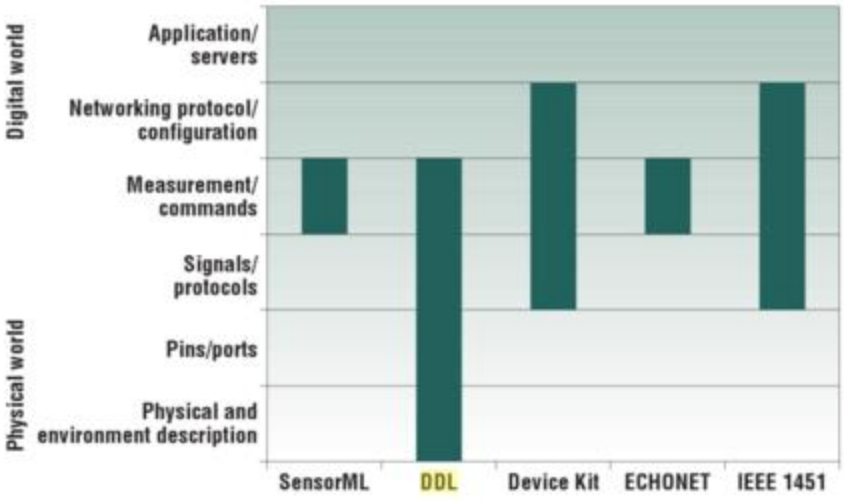

Figure 4: Sensor Description standards (Helal 2010)

At each layer, certain aspects/problems need to be addressed and these are enumerated next, based on Figure 4 above.

a. Physical & environment descriptions explain the physical characteristics of a device such as its form factor and operating environment.
b. Pins & ports: this has to do with the physical presence of the device's interface.
c. Signals & Readings: these are the raw (unprocessed) readings from the interface of a device.

d. Measurements and Commands represent the semantics of signals and protocols.
e. Networking protocol and Configuration is the last layer before data is sent to the middleware.
f. Applications and services are the destination for the sensor data. This layer hasn't been part of any sensor standard. It is at this layer that the description mechanism to be proposed in this thesis will be positioned.

2.3.1 Device Description Language – DDL

DDL was developed by the University of Florida to facilitate the automatic integration of devices - sensors and actuators alike. In DDL, a device is conceived as comprising of three things – a set of attributes covering information about manufacturer, capabilities, an internal mechanism that defines its operations as well as an interface between the external world and the internals of the device. It goes further to classify devices into three categories – sensors, actuators and complex devices (Chattopadhyay 2012b).

Central to the DDL is a device descriptor file which describes a single type of device in XML. The file contains both information for service registration & discovery as well as descriptions of the device's operations. The modeling approach of the DDL is a data-oriented one in which a device's operation is regarded as a collection of input to output processing chains.

The modules within the XML documents i.e. device descriptor file can be parsed by a controller to construct a logical control model which can subsequently be applied to other devices of the same class. DDL isn't tied to one protocol in particular, hence the details of each protocol of interest have been factored into special protocol specific elements. The objective of DDL is to represent devices using 'prime factors' which can be recombined in many different forms to cater to the requirements of new devices or domains of application. In this way, controllers can be developed which can already incorporate features and capabilities of yet unknown devices (Nye 2014).

2.3.2 Device Kit

Device Kit provides a common interface for application code to interact with RFID readers and other devices/actuators (Chattopadhyay 2012b). It models a device in terms of device, transport and connection layers. The application interfaces with hardware devices or sensor by means of Java bundle in an Open Services Gateway Initiative (OSGi) framework (Chattopadhyay 2012a). The device layer provides an interface to hardware, the transport layer parses messages while the connection layer supports Input/Output (I/O) operations to the hardware (Chattopadhyay et al. 2013).

The DeviceKit environment consists of an application, a runtime, and a hardware device as shown in figure 5 below. The runtime is divided into an application agent layer, transport layer and connection layer.

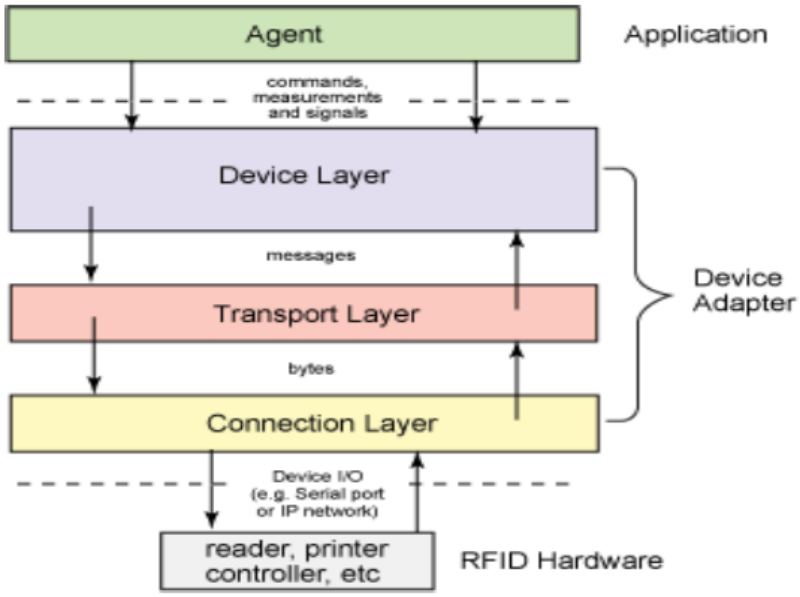

Figure 5: Device Kit Environment (Chattopadhyay 2012b)

a. Connection layer – it supports the read/write of data bytes to the hardware.
b. Transport Layer – this layer, being one layer above the connection layer operates at the level of messages. It understands the format of messages although it doesn't understand what the messages represent or intend.

c. Device layer – this is the layer that's between the hardware and the application. Since it can parse messages as well as the parameters they hold, when an application executes a command, this layer requests that the transport layer send a properly formatted command message.
d. Application agent layer – acting as an adapter, this layer serves to create a transparent interface for the application to access common hardware devices.

2.3.3 IEEE1451

IEEE1451 is a collection of transducer interface standards proposed by the Institute of Electrical & Electronics Engineers (IEEE). The set of standards describe "a set of open, common network-independent communication interfaces for connecting transducers (sensors or actuators) to microprocessors, instrumentation systems and control/field networks" (Chattopadhyay 2012b).

The goal of this standard is to allow the access of transducer data through a common set of interfaces regardless of whether the transducers are connected wirelessly or not. This is achieved by means of a modular design. Signal conditioning and conversion modules are defined in a single entity called Smart Transducer Interface Module (STIM) while application algorithm and network communication modules are grouped into a Network Capable Application Processor (NCAP). From such a function-based modularization, it is possible to realize interoperability between transducer and network by a one to one, one-to-many or many-to-many mappings between STIMs from different sensor vendors NCAPs from one various suppliers.

2.3.4 SensorML & Sensor Web Enablement (SWE)

SensorML is one of the standards proposed within the framework of Sensor Web Enablement (SWE) of the Open Geospatial Consortium (OGC) alongside Observation & Measurement (O&M) Schema and Transducer Modelling Language (TML) (Chattopadhyay 2012b). SensorML provides standard models and an XML encoding for describing any process, including the process of measurement by sensors and instructions for deriving higher-level information from observations. It presents a provider-centric view of information in a sensor web. Processes described in SensorML are discoverable and executable. All processes define their inputs, outputs, parameters,

and method, as well as provide relevant metadata. SensorML models detectors and sensors as processes that convert real phenomena to data. SensorML supports different sensor types by means of a definition of self-contained process models and process chains. SensorML does not encode measurements taken by sensors; measurements can be represented in TransducerML, as observations in Observations and Measurements, or in other forms, such as IEEE 1451.

2.3.5 ECHONET

ECHONET – Energy Conservation and Homecare Network was proposed by Japanese device manufacturers as a set of ISO IEC standards. ECHONET specifies an architecture with which appliances and sensors from multiple vendors can be integrated. This is realized by means of a device specification that defines the devices' properties as well as access methods so that vendors can conform to a uniform interface (Chattopadhyay 2012b). It assumes that other standards are responsible for converting signals and parsing lower layer protocols, hence the description confines itself to only the functional description. It treats devices as self-contained units and specifies the description using object-oriented techniques, with each device having properties and methods as well as validation rules. (Chattopadhyay 2012a)

The ECHONET device object specification defines several classes that represent a device or sensor used in a home network. The properties and methods of each class are enumerated in plain text as well their types and allowed values.

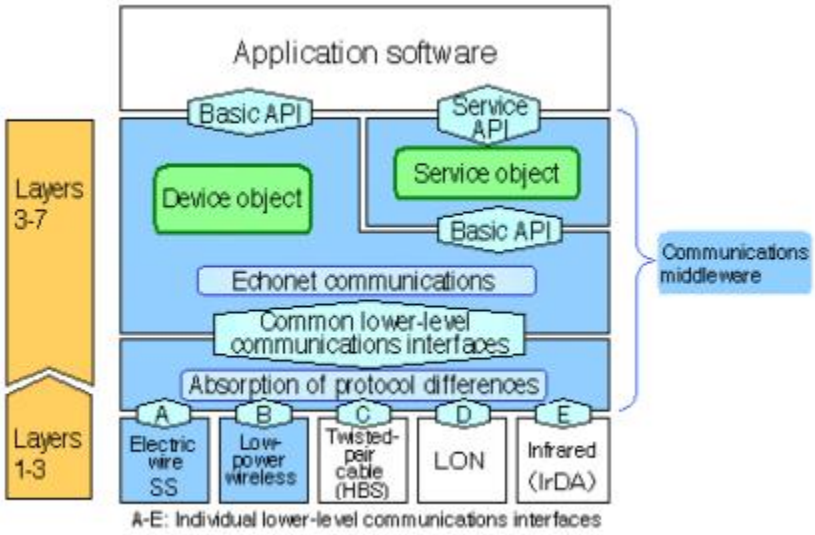

Figure 6: ECHONET Device Specification (Chattopadhyay 2012b)

3. METHODOLOGY

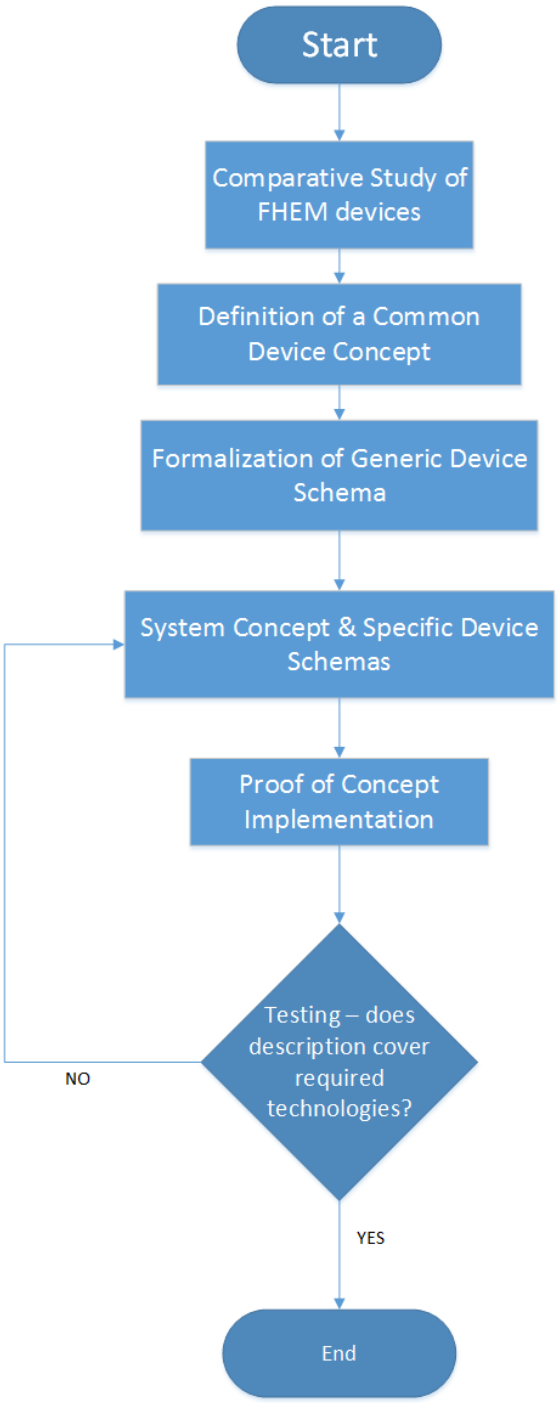

Figure 7: Flow of Steps in Specifying the description mechanism

This chapter will discuss the steps which were followed in the realization of a device description mechanism and in the process, address the research questions raised in chapter 1.

3.1 Comparative Study of FHEM Devices

In order to specify a common description mechanism of similar devices from different vendors and utilizing different communication technologies, information regarding the state of things as they currently are within FHEM has to be gathered. To this end, the documentation available for FHEM was an important information source as well as actual operation of the devices. Although there is a wide range of devices supported by FHEM, only a few categories of sensor and actuator devices were chosen. In the sensors category - Temperature, Humidity and Occupancy sensors while for actuators - Dimmers, Wireless Switches and Heating Controllers. These devices were chosen because they are the most commonly deployed in Home Automation environments.

FHEM is a GPL'd server written in Perl for home automation. It is capable of automating common tasks within the home such as switching lamps, shutters, heating as well to log events and measures such as temperature, humidity, power consumption etc. Being an open-source system, it enjoys the support a broad community of users and developers. Its architecture is flat, having a single main core as an event loop and several modules to implement different tasks (Sangogboye 2016). The field or automation layers in home automation systems as described in chapter 2 are handled by these different modules within FHEM.

FHEM's flat and simple architecture is depicted in figure 8 below.

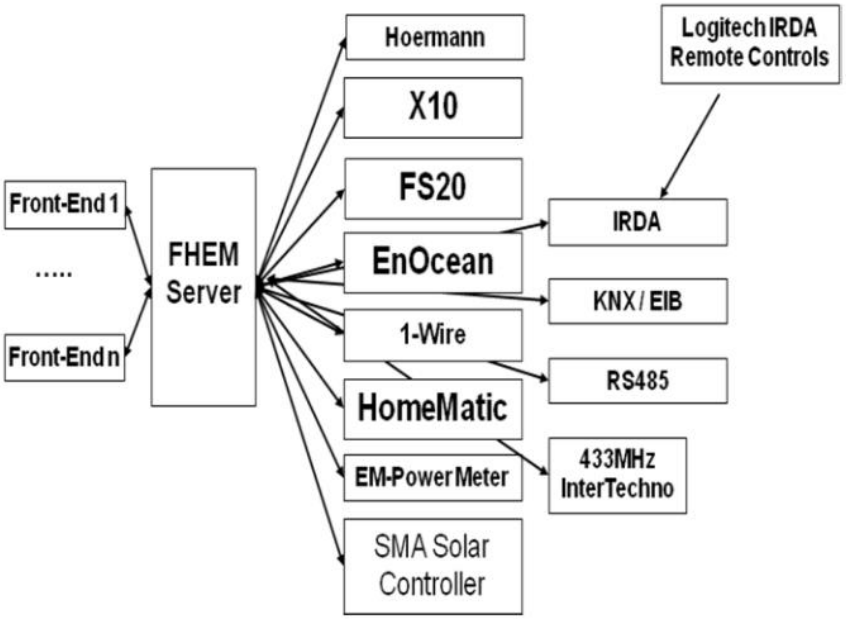

Figure 8: FHEM Server Architecture (Sangogboye 2016)

FHEM as a management layer in HA, there is no common, uniform mechanism for describing the heterogeneous devices which the system attempts to integrate. As shown in figure 8 above, FHEM uses several modules to cater to low level RF communication as well as device specific functions which vary by vendor as well as the communication technology of the device

As effective as FHEM is in handling the specifics of what's required to communicate with and operate an FS20 dimmer or HomeMatic Dimmer, it still requires that separate modules be created for each of these different communication technologies and while that is required to realize such an open source home automation system, it is inconvenient for end users to have to separately manage these discrepancies. How so? For example, to make use of an EnOcean switch in FHEM, a user needs to remember that the switch isn't just a switch, but that it is an EnOcean switch which means that it is must be defined with certain parameters and supports certain actions or behaviours that have to be triggered using particular commands. If the user decides to use an FS20 switch, perhaps because it's cheaper they also have to remember what the specific commands for defining and interacting with FS20 switch is.

Here's an example of this inconsistency. To define an FS20 device, the command is shown in the snippet below:

```
define <name> FS20 <housecode> <button> [fg <fgaddr>] [lm
                    <lmaddr>] [gm FF]
```

Where the values of housecode, button, fg, lm, and gm have meaning explained below.

a. housecode is a 4 digit hex or 8 digit ELV4 number, corresponding to the housecode address.
b. button is a 2 digit hex or 4 digit ELV4 number, corresponding to a button of the transmitter.
c. The optional fgaddr specifies the function group. It is a 2 digit hex or 4 digit ELV address. The first digit of the hex address must be F or the first 2 digits of the ELV4 address must be 44.
d. The optional lmaddr specifies the local master. It is a 2 digit hex or 4 digit ELV address. The last digit of the hex address must be F or the last 2 digits of the ELV4 address must be 44.
e. The optional gm specifies the global master, the address must be FF if defined as hex value or 4444 if defined as ELV4 value.

With such a syntax for defining FS20 devices, valid ways of defining FS20 devices include

```
define roll1 FS20 7777 01
```

```
define otherlamp FS20 24242424 1111 fg 4412 gm 4444
```

Meanwhile, the definition of an EnOcean device is markedly different from the above based solely on the nature of the devices and thus, for EnOcean devices, the syntax for the definition command is:

```
define <name> EnOcean <DEF> [<EEP>]|getNextID|<EEP>
```

where <DEF> is the SenderID/DestinationID of the device (8 digit hex number), for example define switch1 EnOcean FFC54500 and in the case that the EEP (EnOcean Energy Profile) is known, the appropriate device can be created with the basic parameters. An EEP-based definition will be

```
define sensor1 EnOcean FFC54500 A5-02-05
```

or

```
define sensor1 EnOcean A5-02-05
```

For the regular user, these inconsistencies in operating or interacting with devices (of the same category i.e. serve the same function such as dimmers, switches or heating controllers) are at best inconvenient and at worst a burden when all the user wants to do is dim down the bedside lamp at bed time.

Beyond the inconvenience for the end user, such differences in interactions for devices creates an unsustainable practice for software developers who contribute to or maintain such open source Home Automation systems like FHEM. The fact that they have to create and subsequently maintain separate modules for interacting with heterogeneous devices means that for every/any newly released device by vendors, they need to write modules to support interactions with the device. The absence of a uniform model of representing commonly useful devices presents a barrier for application developers in the aspect of developing rich home automation apps which can transparently interface with devices regardless of its communication technology or vendor specific features.

This two-sided problem within FHEM underscores the need for a common description mechanism which presents the end user with a convenient means of interacting with varied devices within a common use case as well as for application developers to create solutions which target devices at a common level of abstraction which shields them from the specific interaction commands with which the devices are driven.

With a common description mechanism for heterogeneous devices, end users will benefit from the ease of operating devices from whichever vendor they purchase and not have to cope with the hassle of different commands for their usage. So, whether they buy a HomeMatic switch or bring home an EnOcean switch, they can rest assured that as far as they are concerned, a switch is a switch. For application developers, a common description mechanism would mean that they have more opportunities to focus on developing Home Automation solutions which enable valuable scenarios for consumers rather than worrying about how to support or adapt to newly released devices.

3.2 Definition of a Common Device Concept

As mentioned in the introduction to this chapter, a limited set of devices were chosen as the focus of this work – sensors (temperature, humidity & occupancy) and actuators (dimmers, switches and heating controllers). In devising a common means of representing the devices, the current representations within FHEM were investigated. Figure 9 below shows how FHEM internally representing the devices.

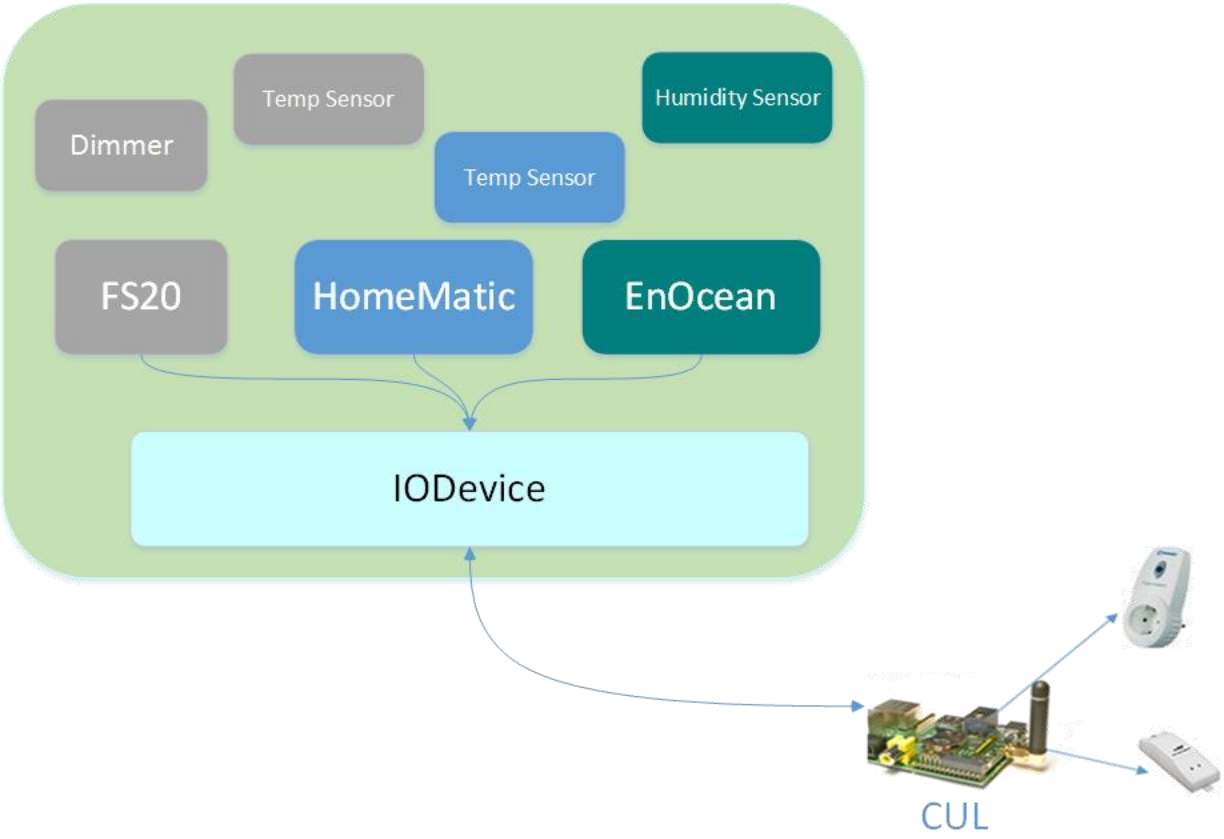

Figure 9: Elements of FHEM Communication

As can be seen in figure 9, within FHEM, the physical hardware device which does the RF communication with field devices is internally represented within FHEM as an IODevice[7]. At a higher level, the various devices – FS20, HomeMatic when defined are assigned/allocated to an IODevice which handles specifics of message parsing and formatting to and from the physical RF transceiver, the CUL[8].

What this means is that the IODevice handles the low level details of messaging, addressing etc as is applicable to protocols such as BidCoS[9] (for HomeMatic) which is different from FS20. Technology specific modules at a higher level then handle issues closer to the functionality

[7] IODevice (Input Output Device) represents the physical device used for sending signals originating fromlogical devices within FHEM

[8] CUL stands for CC1101 USB Lite which is a transceiver for RF communications and supports both AM and FM

[9] BidCoS is short for Bidirectional Communication Service and is the communication protocol used by HomeMatic devices. Because the communication is bidirectional, one can know if a packet is received or not

provided by the devices. In FHEM, these modules are called when commands for defining devices, setting or getting parameters etc. are triggered.

Device modules within FHEM are clustered by their communication technology, for instance the FS20 module handles interactions with all FS20 devices, regardless of their specific functions. Same applies to the CUL_HM which handles interactions for HomeMatic devices while the EnOcean module works for EnOcean devices. Therefore, understanding device specifics across the different technologies required a comparison of the commands/syntax defined by the respective modules for interacting with devices of respective communication types.

Figure 10 below is an excerpt from a comparison of dimmer commands for FS20, HomeMatic and EnOcean.

Comm. Tech / Device Category	FS20	HomeMatic	EnOcean
Dimmers	define <name> FS20 <housecode> <button> [fg <fgaddr>] [lm <lmaddr>] [gm FF]	define <name> CUL_HM <6-digit-hex-code|8-digit-hex-code>	define <name> EnOcean <DEF> [<EEP>]|getNextID|<EEP>
	set <name> dim%	set <name> 0-100 [on-time] [ramp-time]	set <name> dim%
	set <name> dimdown	set <name> on	set <name> teach
	set <name> dimup	set <name> off	set <name> on
	set <name> dimupdown	set <name> press <short|long>	set <name> off

Figure 10: Heterogeneous Device commands in FHEM

The comparison indicated a few patterns in how the interactions were set up. Regardless of which technology is considered, it was observed that they all share a common set of high level interactions.

a. Definition:

```
define [option] <name> <type> <type-specific>
```

This sets up the device within FHEM. It requires specifying the name of the device, its type (indicated by the name of the module which corresponds to its technology) and then type-specific attributes. For example, HomeMatic devices require that things like housecode, button and so on

are set while EnOcean devices make use of an identifier for EEP[10]. Such a valid definition step constitutes a devspec within FHEM.

b. Set:

```
set <devspec> <type-specific>
```

Commands in this category are used to manipulate the device with respect to the actions or commands which it supports. It is also used to change the state of the device by altering the values of some properties or attributes. This is equally realized with the `setstate` or `setreading` commands.

c. Get

```
get <devspec> <type-specific>
```

Get commands are issued to retrieve certain attributes from the specified device.

d. Attr

```
attr <devspec> <attrname> [<value>]
```

Attr commands are used to set/modify values of supported attributes on the device. It is possible also to have user-defined attributes which can in turn be reused in other applications.

From these common set of interactions which are common across devices in different communication technologies, a visual description was realized.

[10] EEP - EnOcean Equipment Profiles are standardized communication profiles that ensure that devices based on EnOcean technology from different vendors can work with gateways from another.

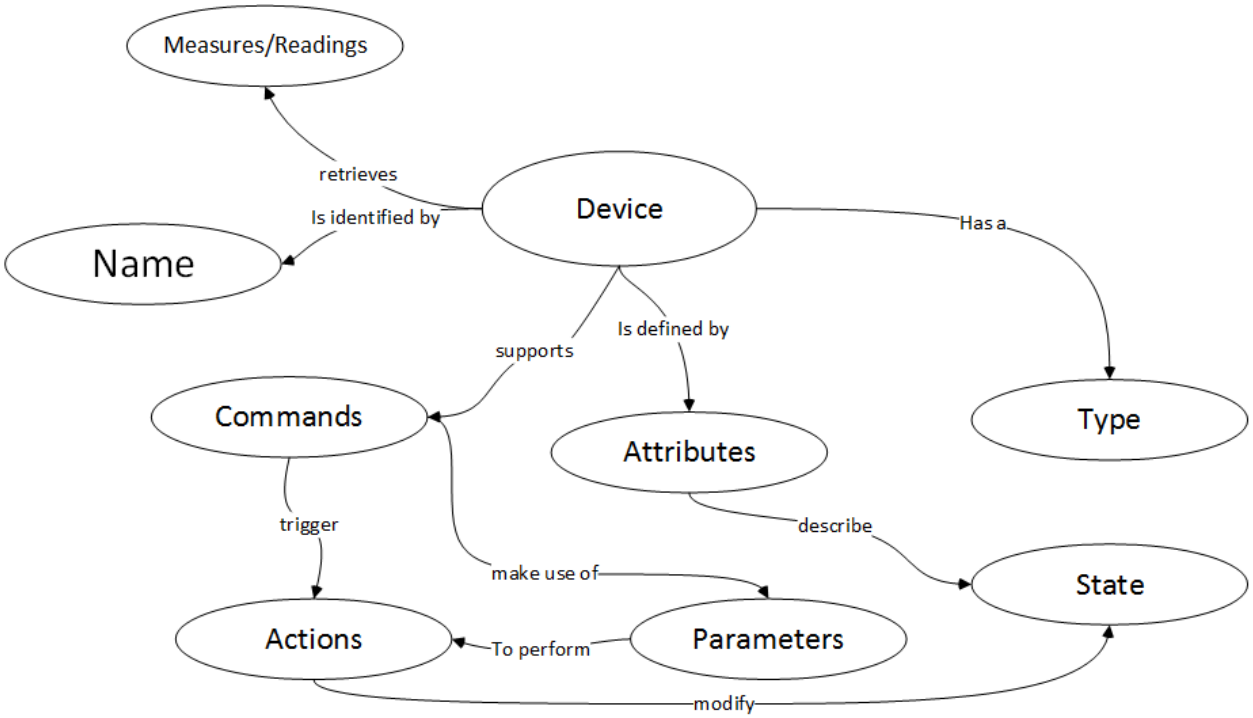

Figure 11: Visual Device Description

The visual device description in figure 11 above is helpful in reasoning about common aspects of a device regardless of the particular technology type within FHEM. Whatever the communication technology is, devices in FHEM are defined by attributes which tell certain details of the device such as its vendor, year of manufacture, type etc. Devices can also support commands that trigger actions which modify its state. Further, a device can retrieve measures/reading s from the environment such as temperature, humidity, concentration of gases etc.

This generic device description highlights what aspects of devices needs to be modelled. This has been the case in similar attempts at device descriptions. For example, the authors in (Park et al. 2013) centred their work around creating an abstract representation of devices based on characteristics. They categorized devices in order to assign them a device ID – Environment Monitoring Devices, Environment Control Devices, Home Appliance/Automation Devices and Miscellaneous Devices. These four are device type set codes that categorize the devices based on their usage purposes. Device set codes are further used to distinguish devices based on their physical attributes or characteristics. Device codes are the most specific means for classifying the

devices, representing more specific attributes of each device such as oxide or carbon dioxide measurement.

The system proposed by the authors in (Song et al. 2014) - BATMP is similar to FHEM in one important aspect. BATMP implements several drivers to convert a physical device into a logical device that contains a set of parameters. FHEM does the same thing by means of modules which handle specifics of interactions to different communication protocols. In order to facilitate a common description of devices, the authors proposed the use of parameters to represent devices. These parameters are based on the underlying data representation and operation of the devices in different protocol categories (KNX, CoAP/6LoWPAN, ModBus) which is then used to determine how to map them to corresponding logical representations.

Having identified what needs to be modelled, it is pertinent to determine how and where such a model can be applied. As discussed in section 2.3 in chapter 2, there are different layers in which device descriptions can be applied. In Figure 5 it was evident that the application/server layer isn't covered by existing descriptions. In FHEM as well, this is the case. Therefore, the device description being proposed will be applied in the application layer of FHEM as an extension to its current architecture.

The works in (Aberer et al. 2006), (Boman et al. 2014), (Pires et al. 2014), (Nachabe et al. 2015) as well as DDL (Nye 2014), DeviceKit (Chattopadhyay 2012a) and SensorML (Chattopadhyay 2012b) all make use of a tag-based approach for representing devices and we have followed a similar approach. A tag based approach means that the essential aspects of devices which are required to represent it are grouped into a set of keywords which taken together can sufficiently represent the devices within FHEM. A proposed set of tags which could apply in describing the devices is given below.

device_category – this describes the category to which the device belongs. For instance temperature sensor, humidity sensor etc.
name/id – this references a unique identity for the device within the system.
state_attrs – here are the attributes/properties that represent the internal state of the device e.g if it on/off.

attr – a name for the attribute.

data_type – an appropriate data type for the attribute value.

measure_attrs – these attributes are for properties in the device environment which the device is measuring e.g. temperature values, humidity values.

attr – as in the case of the *state_attrs*, captures the value of the external measure or property of interest.

data_type – a valid and appropriate data type for the property of interest.

communication_tech – captures the communication technology used by the device.

actions/commands – this group of tags represent the actions or commands which can be performed by the device.

action_name – the name of the action or command.

parameters – these set of keywords capture the inputs required to trigger the command/action.

param_name – name of the parameter.

data_type – a valid and appropriate data type for the parameter.

return_values/output – these set of keywords capture the output from the commands/action

output_name – name of the output.

data_type – valid and appropriate data type.

The data types which apply to devices were determined based on the nature of values captured by the devices. For example, the data type for values from a temperature sensor will be floating point while for an occupancy sensor, a value to represent motion (or otherwise) is best represented as a Boolean (true/false). The aforementioned tags/keywords were used to specify the structure of a generic device which can then be applied to devices in a specific category. For instance, the keywords can be used to create a description for a generic dimmer containing attributes and actions that are typical of a dimmer device regardless of which communication technology it uses. A similar reasoning will apply when extending the tag structure to a sensing device, in which case there will be more attributes than the actions/commands.

3.3 Formalization of a Generic Device Schema

With these keywords that define aspects of device representation, it provides a starting point for formalizing a description mechanism which can be integrated into FHEM. In trying to formalize the description that's been formulated up to this point, a schema-based representation was

considered. A principal requirement for a formal representation of the description mechanism is that it should enable a common interface which allows developers to program applications for home automation without having to know the specific interaction details of each communication technology within the system. A schema representation to formalize the device descriptions is necessary so that the description can be interpreted by applications to create an in-memory representation of a device, which can then be used to interact with it.

Schemas describe the structure and sometimes, the semantics of data (Wang 2010). More importantly, schemas are intended to assert the correct structure of documents (Ml 2009). By means of a schema, it is possible to assure a consistent, unambiguous representation or presentation of some phenomena, in this case, the description of a device. This feature of schemas in asserting a consistent representation was applied to define a base schema which constrains device descriptions at a very generic level. The schema proposed in this work specifies the names, data types and allowed values for the keywords that together make up the description of a device. Figure 12 below shows the derived relationship between the base schema and device specific schemas. The base schema is used to create definitions for devices in specific categories so that when parsed, applications are able to construct a device in memory with its associated set of attributes and supported actions/commands. For instance, a temperature sensor schema will define a characteristic temperature sensor within FHEM and when the schema is interpreted, FHEM is able to correctly create a temperature sensor representation complete with its attributes/properties.

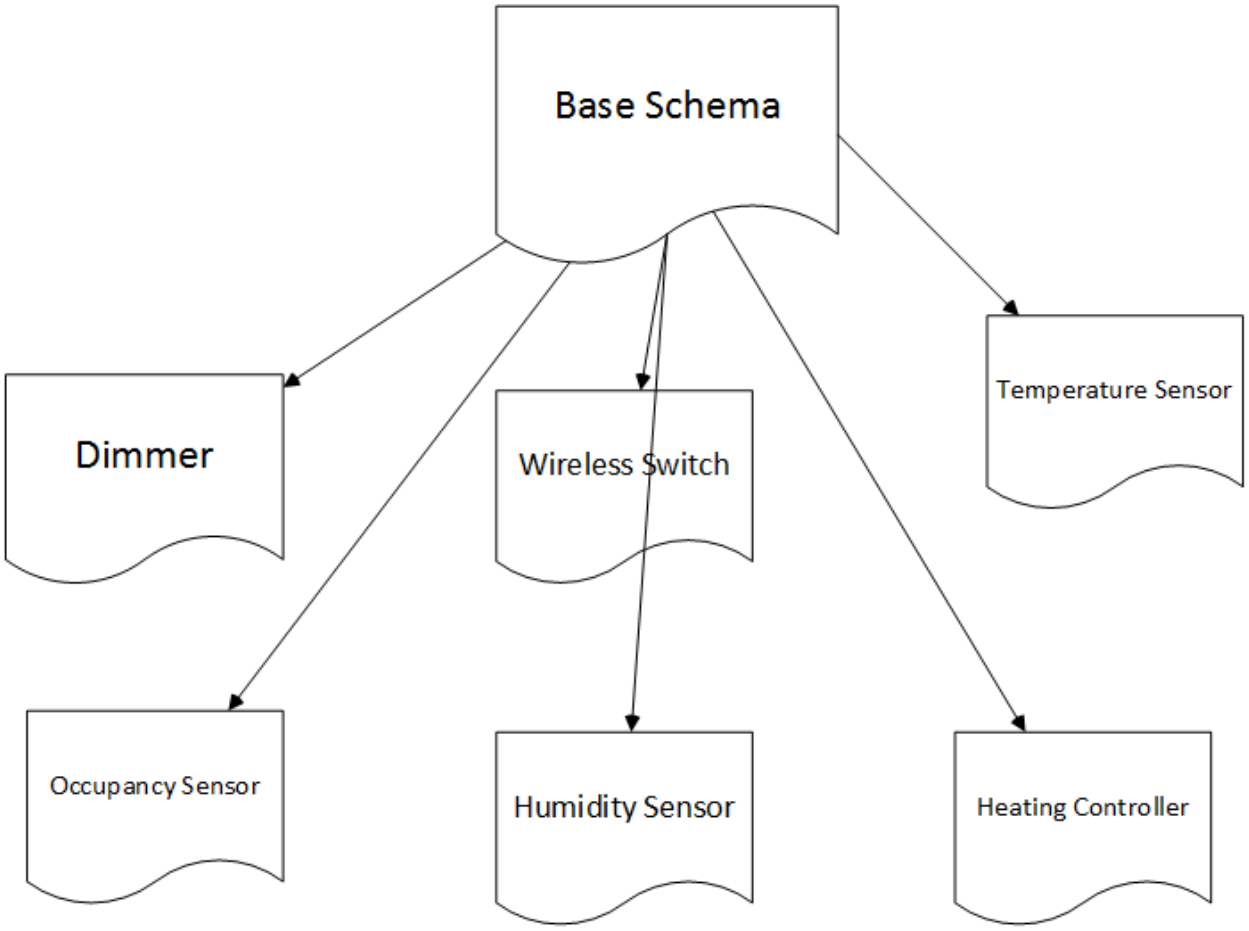

Figure 12: Schema for Device Descriptions

In defining a schema for device descriptions, there are options to choose from. Prominent among the options for schema representations is XML[11] and JSON[12]. XML is the basis for representation in DDL, DeviceKit as well as SensorML. XML as a schema representation language is advantageous in heterogeneous internet environments (Maeda 2012). Given the pervasive support for XML parsing in several modern programming languages, it is a viable option for representing the schema for the device description being proposed.

The other option for schema representation is JSON. While JSON has taken off within the web technology platforms as a lightweight data interchange format, it isn't a popular choice for schema definitions. This doesn't mean that it's not possible to use JSON as a schema representation format.

[11] eXtensible Markup Language - is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.

[12] JavaScript Object Notation - is a lightweight data-interchange format. It is language independent, "self-describing" and easy to understand.

In fact, JSON is a capable format for schema representation as reflected in the documentation of the JSON-schema project (Space Telescope 2016). JSON Schema is a tool for validating the structure of JSON data. It is intended to describe the data format, provide a clear human- and machine-readable documentation as well a comprehensive structural validation which can be used to automated testing or validating client-submitted data (JSON Schema 2016).

JSON was chosen as the schema representation format for the device description because of its lightweight, textual and language-independent format (Juárez et al. 2014). Furthermore, JSON has been designed to be more human readable and easier to parse for computers than XML and has been demonstrated to be faster and use fewer resources than equivalent XML (Nurseitov et al. 2009).

The base schema is shown in figure 13 below. Its purpose is to assert/validate the high level and generic structure of device descriptions. It details the important aspects of devices, namely attributes and actions. The attributes section is defined as an array of objects each with two properties – *attr_name* and *attr_type* which define the name of the property and its type respectively. The *actions* section is defined as an array of objects with properties – *action_name* for the name of the supported action, *params* which is an array of parameter object definitions having a *name* and *type* parameters, as well as the *return_values* which is an array of definitions of return values each with a *name* and *type* as well.

```
{
    "$schema": "http://json-schema.org/schema#",
    "id": "http://hadds.s3-website.eu-central-1.amazonaws.com/device-spec.json",
    "title": "Device",
    "description": "A base schema for device descriptions targeting the FHEM Home Automa-
tion Server",
    "device_type": {"type" : "string"},
    "attributes": {
        "type": "array",
        "items": {
            "type": "object",
            "properties": {
                "attr_name": {"type": "string"},
                "attr_type": { "type": "string", "enum": ["string", "number", "boolean"]}
            }
        },
        "uniqueItems": true
    },
    "actions": {
        "type": "array",
        "items": {
            "type" : "object",
            "properties" : {
                "action_name": {"type": "string"},
                "params": {
                    "type" : "array",
                    "items": {
                        "type" : "object",
                        "properties" : {
                            "param_name": {"type" : "string"},
                            "param_type" : {"type": "string", "enum": ["string", "num-
ber", "boolean"]}
                        }
                    }
                },
                "return_values": {
                    "type": "array",
                    "items": {
                        "type" : "object",
                      "properties": {
                        "value_name": { "type": "string" },
                        "value_type": {
                          "type": "string",
                          "enum": [ "string", "number", "boolean" ]
                        }
                      }
                    }
                }
            }
        }
    }

}
```

Figure 13: Base Schema for Devices

3.4 System Concept & Specific Device Schema

This thesis proposes a system concept to put the description mechanism to use and address the identified problem/gap within FHEM. The system concept has several aspects which together extend FHEM and enable the realization of the objectives. More-so, the system demonstrates a sustainable and scalable method for supporting new/unknown devices in FHEM. Following is a description of the role of each layer. The architecture of the system concept is shown in figure 14.

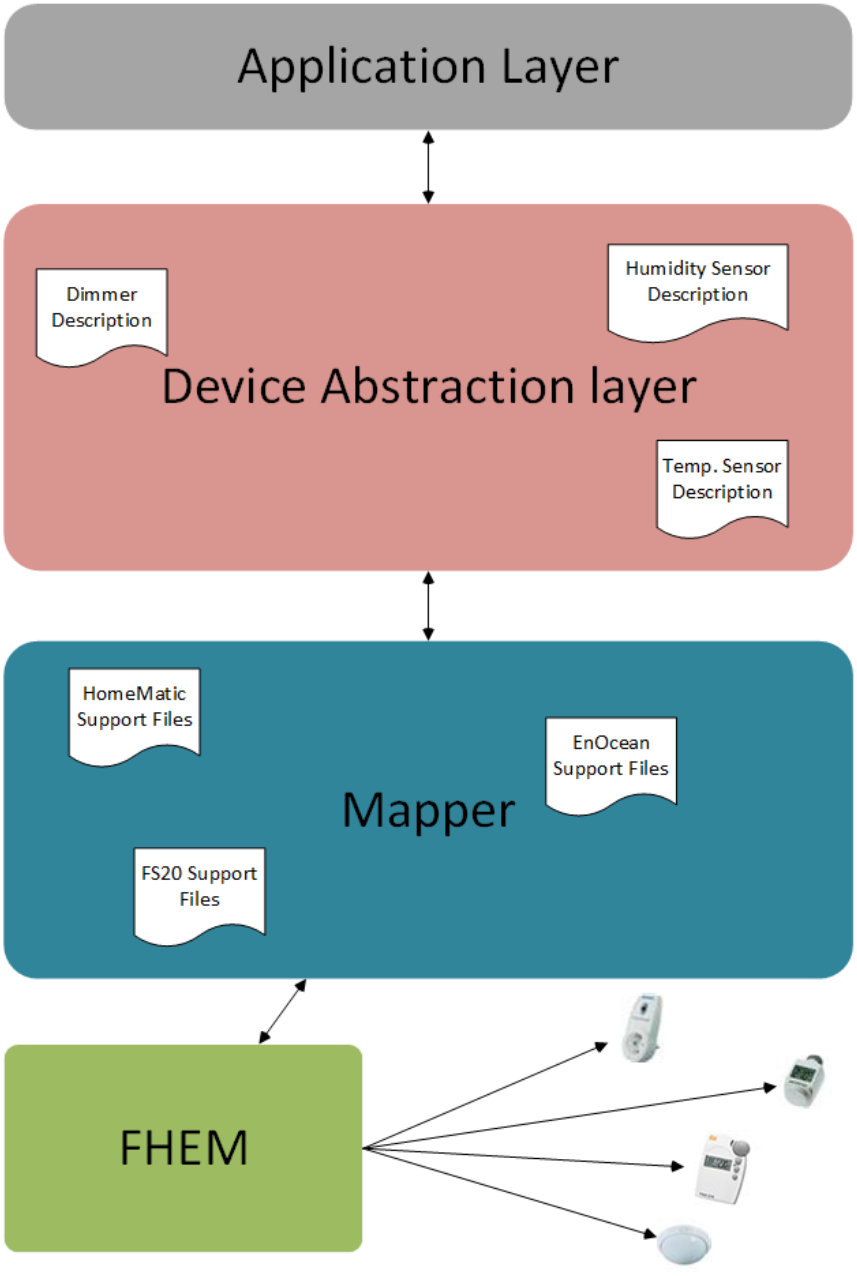

Figure 14: System Concept

3.4.1 Application Layer

This is the layer at which application developers will make use of the description mechanism to uniformly develop HA applications which rely on FHEM as the home automation server. With this layer in place, developers only need to have knowledge of the interface exposed by the abstraction layer and they can program against that interface in a transparent manner. The applications can be developed in any language or platform of choice as determined by the developer.

3.4.2 Device Abstraction Layer

This layer will expose an object-oriented interface by parsing device description files for each device category. The device description files are JSON files that adhere to a common schema – the base schema which is processed by this layer. To add support for new devices, a JSON file which represents the abstraction for that device category is created. With that, the layer is able to generate and expose an Object-Oriented interface for a developer to program against.

3.4.3 Mapper

The schema-based representation is only one part of the solution. An even more critical aspect of the solution is how to map the representation into FHEM. Such a mapping layer is required for at least two reasons.

a. A need to integrate the proposed description mechanism into FHEM in a sustainable, repeatable and scalable manner.
b. In order to realize the objective of providing application developers with streamlined process of developing apps transparently, it is necessary to define a layer which wraps over FHEM. Application developers can then program using that layer.

The proposed mapper is the closest layer to FHEM and it understands how to interact with devices at a technology-specific level. For example, the mapper can setup FS20, HomeMatic and EnOcean devices. The abstraction layer depends on the mapper to “speak” the commands specific to each technology. The mapper will make use of what is proposed to be technology-support files that

contain mappings from the generic device abstractions to the specific technology commands with which FHEM controls and accesses devices. For example, calling a constructor for an FS20 dimmer in the Object-Oriented interface in the abstraction layer above will map to a define command at the mapper layer.

3.4.4 Device Description Files

Conforming to the generic base device schema, the description file for a Dimmer was proposed incorporating the commonalities between dimmers within FHEM. This abstract dimmer description can be parsed by the application support layer to create and object-oriented interface for a dimmer device which can be programmed against from the application point of view and which maps to the specific communication technology as determined by the mapper layer. A snapshot from it is shown in figure 15. The full JSON description is included in the appendix.

3.4.5 Technology Support Files

The device description files constitute a generic representation of devices of a particular type. For them to be usable, it is required that technology specific interactions be defined in order to work with devices using that particular technology. For example, to be able to operate a HomeMatic heating controller, the device description specifies a generic structure of heating controllers in terms of how it is defined, the commands it supports as well as the typical attributes. That is sufficient at the device abstraction layer to work with the device, but in order for FHEM to setup and control the device, the technology specific commands need to be known. This is the role of the technology support files. They contain the commands which are specific to communication technologies. These files are used by the mapper to determine what commands to run for a given method in a device abstraction layer. If there are interactions which are not supported by a given technology, the technology support files would have null commands for that particular interaction.

```
{
  "device_type": "Dimmer",
  "attributes": [
    {
      "attr_name": "device_name",
      "attr_type": "string"
    },
    {
      "attr_name": "comm_tech",
      "attr_type": "string"
    }
  ],
  "actions": [
    {
      "action_name": "DimUp",
      "params": [ ],
      "return_values": [
        {
          "value_name": "action_succeeded",
          "value_type": "boolean"
        }
      ]
    },
    {
      "action_name": "OnTill",
      "params": [
        {
          "param_name": "targetTime",
          "param_type": "number"
        }
      ],
      "return_values": [
        {
          "value_name": "action_succeeded",
          "value_type": "boolean"
        }
      ]
    }
  ]
}
```


Figure 15: Dimmer Device Description

The last two steps in the process are discussed in chapter 4 – Implementation.

4. IMPLEMENTATION

This chapter discusses the implementation of a mobile application as a proof of concept of applying the proposed description mechanism. The mobile application is intended to incorporate the proposed device description as well as the system concept for extending FHEM. The app demonstrates how the application layer makes use of the device abstractions to transparently manipulate and interact with devices in FHEM.

4.1 Architecture

Figure 15 shows the architecture for the proof of concept. The layers in the proposed system concept discussed in chapter 3 have corresponding elements within the architecture of the proof of concept implementation.

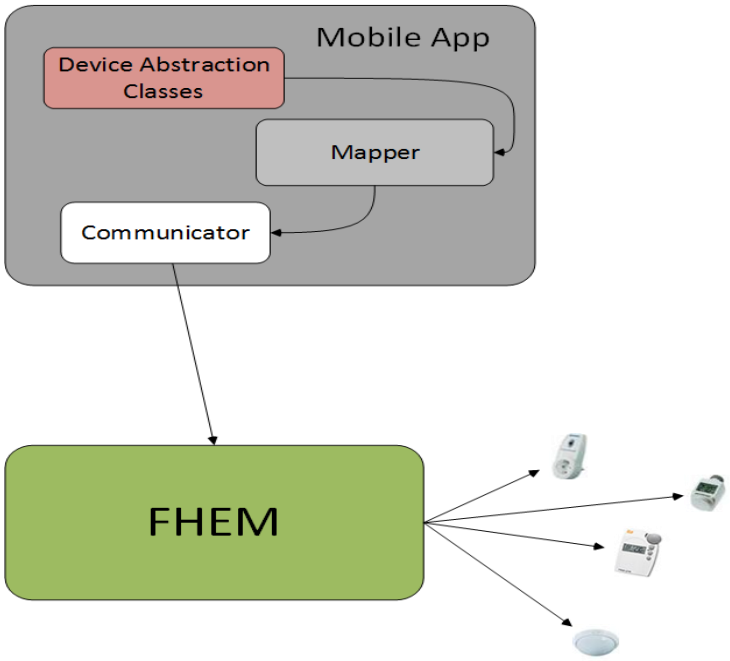

Figure 16: Proof of Concept Architecture

For example, the application layer corresponds to the mobile app, the device abstraction layer is implemented as object oriented device classes while the mapping layer is implemented as a command generator.

4.2 Mobile Application

The sole purpose of the mobile application is to demonstrate a common way of accessing heterogeneous devices using the proposed description mechanism. Its requirements are hence streamlined to:

a. *Add/Create devices in FHEM* - this activity is to create a new device in FHEM, specifying its name, manufacturer, device type and communication technology

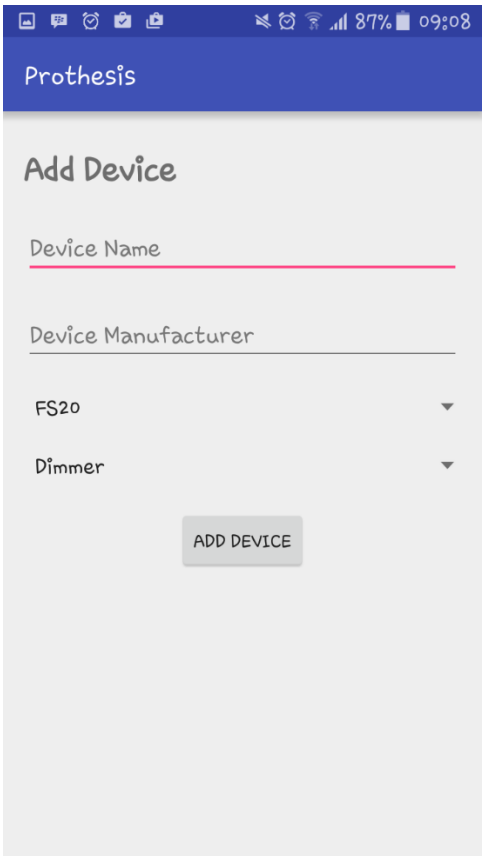

Figure 17: Add Device Activity

b. *List/Retrieve devices in FHEM* – this activity lists the devices that have been created it the Create devices activity. It is updated as devices as added.

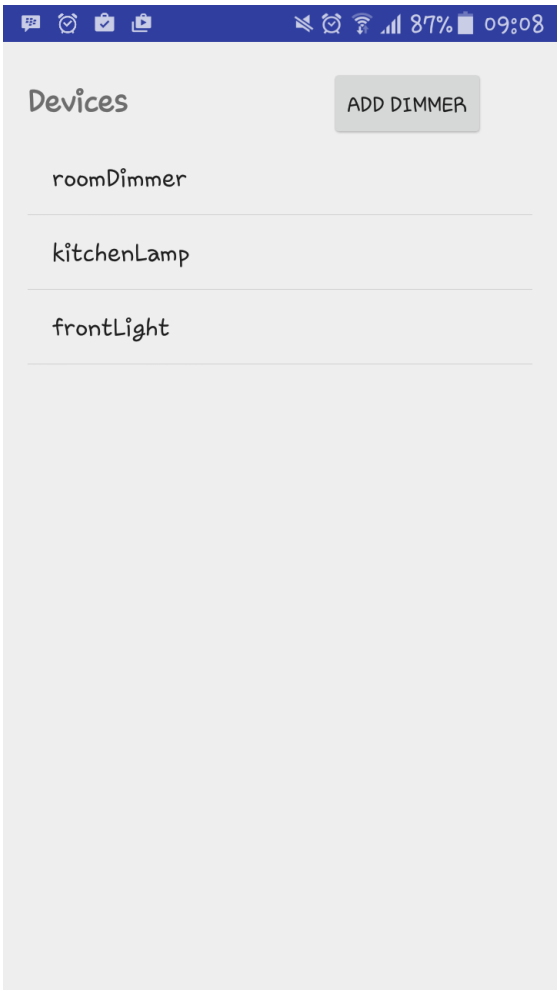

Figure 18: Device List Activity

c. *Operate devices in FHEM* – this activity is to operate or control the devices that have been defined. In the demonstration, the operation of a Dimmer is shown. The typical actions of a dimmer are visible – dimming up/down, on till, dim up by, dim down by etc.

Figure 19: Operate Device Activity

4.2.1 Platform

Android was chosen as the mobile platform to develop the application on. It was chosen because Android has the highest market share of the mobile operating systems market. More importantly, Android is an open source operating system developed by Google, based on the Linux kernel and intended for touchscreen devices like phones, tablets. There are also versions of Android run on Televisions (Android TV), Cars (Android Auto) and Wristwatches (Android Wear). The latest major release is Android 6.0 “Marshmallow”.

4.2.2 Development Environment

The application was developed in Android Studio – the official IDE[13] for Android applications development, released and maintained by Google. It features code editing, debugging, performance tooling, flexible build system and instant build/deploy system which enables a developer to efficiently develop android apps. The application was built to target the 5.0 release of Android – "Lollipop". A mobile phone running the Lollipop version of Android was used as the development device instead of an emulator.

4.2.3 Modules

In order to fulfill the requirements of the app as listed in section 4.2, a few modules were developed while also mirroring elements from the proposed system concept.

a. *Device Description* – this module was developed to mirror the device abstraction layer in the proposed system concept. It contains classes which define methods and attributes to represent devices. The structure of the classes were based on the description files presented in chapter 3. In the proof of concept mobile application, a Dimmer device description was implemented. Besides containing attributes that define the device e.g. name, manufacturer and communication technology, it also contains methods common to dimmers across various vendors or communication technologies. For instance, dimUp, dimDown, dimUpBy, dimDownBy and toggle among others. The device description class thus abstract the most typical elements of devices of a particular category. The class contents are included in the appendix.
b. *Mapper* – this module corresponds to the mapper layer in the proposed system concept. It determines which technology specific commands are required for a given action from the device description module. It is able to do this by means of Technology support files, which define the specific commands requires to interact with devices of a particular technology type – HomeMatic, FS20, EnOcean etc. The mapper class exposes a simple method that

[13] IDE - Integrated Development Environment is a software application that provides comprehensive facilities to computer programmers for software development.

takes in the action to perform and the target device. It then uses the appropriate technology support file to determine the command to apply.

```
static ITechnologySupport techSupportClass;
//this method is able to accept an intended command for a given device and use to call the appropriate tech support class
//
public static boolean Do(String what, Dimmer targetDimmer){

    switch(targetDimmer.getTech()){
        case "FS20":
            techSupportClass = new FS20();
            break;
        case "HomeMatic":
            techSupportClass = new HomeMatic();
            break;
        case "EnOcean":
            techSupportClass = new EnOcean();
            break;
        default:
            break;
    }
    //check the tech of the device, then instantiate the appropriate technology class
    String cmdText  = techSupportClass.ParseCommand(targetDimmer.getDeviceName(), what);
```

Figure 20: Mapper Class Implementation

c. *ITechnologySuppport* – this is a Java interface that abstracts the concept of a component which holds the specific commands for interacting with devices of a particular technology type. Its single method takes in the command to perform as well as the target device and is then able to generate the technology specific command to that effect.

```
public String ParseCommand(String deviceName, String commandKey );
```

Figure 21: ITechnologySupport Definition

4.3 Implementation Setup

In order to test the proof of concept application, a setup of an FHEM system was done. It consisted of a Tux radio, a LAN box, a running instance of FHEM Perl server as well as a few home automation devices across different communication technologies. The TUX radio serves as an RF transceiver and it comes with a minimal Debian operating system which can be operated via a console. It is connected to a router so it can be part of a LAN or a WiFi network. The IP address of the FHEM server running on the Tux radio is used by the communication module within the app to access FHEM.

4.4 Results

Testing the mobile application to demonstrate the practicality of the proposed description mechanism in providing a uniform interface for interacting with heterogeneous device technologies. The results are shown with screen shots of the compiler output from the IDE.

a. Device Definition: it was possible to define a device (dimmer) transparently regardless of its communication technology.

```
08-11 09:59:42.520 11517-11517/pritibits.com.prothesis D/DEFINE: define roomDimmer CUL_HM
08-11 10:00:54.270 11517-11517/pritibits.com.prothesis D/DEFINE: define kitchenLamp FS20
08-11 10:01:54.410 11517-11517/pritibits.com.prothesis D/DEFINE: define frontLight EnOcean
```

Figure 22: Device Definitions

b. Device Operation: it was also possible to transparently operate a device in the same manner no matter the underlying technology. The screen shots from the compiler shown in 23 to 25 below demonstrate this.

```
08-11 10:05:21.440 11517-11517/pritibits.com.prothesis D/DEFINE: set roomDimmer on
08-11 10:05:27.120 11517-11517/pritibits.com.prothesis D/ViewRootImpl: ViewPostImeInputStage ACTION_DOWN
08-11 10:05:27.190 11517-11517/pritibits.com.prothesis D/DEFINE: set roomDimmer up
08-11 10:05:27.970 11517-11517/pritibits.com.prothesis D/ViewRootImpl: ViewPostImeInputStage ACTION_DOWN
08-11 10:05:28.050 11517-11517/pritibits.com.prothesis D/DEFINE: set roomDimmer down
08-11 10:05:29.470 11517-11517/pritibits.com.prothesis D/ViewRootImpl: ViewPostImeInputStage ACTION_DOWN
08-11 10:05:29.560 11517-11517/pritibits.com.prothesis D/DEFINE: set roomDimmer off
```

Figure 23: Operating a HomeMatic dimmer

```
08-11 10:07:46.220 11517-11517/pritibits.com.prothesis D/DEFINE: set kitchenLamp dimdown
08-11 10:07:48.260 11517-11517/pritibits.com.prothesis D/ViewRootImpl: ViewPostImeInputStage ACTION_DOWN
08-11 10:07:48.310 11517-11517/pritibits.com.prothesis D/DEFINE: set kitchenLamp dimup
08-11 10:07:49.770 11517-11517/pritibits.com.prothesis D/ViewRootImpl: ViewPostImeInputStage ACTION_DOWN
08-11 10:07:49.840 11517-11517/pritibits.com.prothesis D/DEFINE: set kitchenLamp off
08-11 10:07:50.880 11517-11517/pritibits.com.prothesis D/ViewRootImpl: ViewPostImeInputStage ACTION_DOWN
08-11 10:07:50.950 11517-11517/pritibits.com.prothesis D/DEFINE: set kitchenLamp on
```

Figure 24: Operating an FS20 dimmer

```
08-11 10:24:46.920 3014-3014/pritibits.com.prothesis D/DEFINE: set frontLight dimdown 10%
08-11 10:24:53.530 3014-3014/pritibits.com.prothesis D/ViewRootImpl: ViewPostImeInputStage ACTION_DOWN
08-11 10:24:53.550 3014-3014/pritibits.com.prothesis D/DEFINE: set frontLight dimup 10%
08-11 10:24:54.900 3014-3014/pritibits.com.prothesis D/ViewRootImpl: ViewPostImeInputStage ACTION_DOWN
08-11 10:24:54.940 3014-3014/pritibits.com.prothesis D/DEFINE: set frontLight off
08-11 10:24:56.240 3014-3014/pritibits.com.prothesis D/ViewRootImpl: ViewPostImeInputStage ACTION_DOWN
08-11 10:24:56.300 3014-3014/pritibits.com.prothesis D/DEFINE: set frontLight on
```

Figure 25: Operating an EnOcean dimmer

As these test results show, the proposed device description as implemented in the proof concept demonstrates the practicality of operating devices in a uniform manner regardless of the technology it relies on. Even though the underlying commands for the actions of a dimmer are different in each technology – for example, Dimming up in FS20 is done by the `set <devicename> dimup`, in EnOcean , the command requires a parameter for the level to which the device should be dimmed up, hence `set <devicename> dimup 10%`

4.5 Reflections

Reflecting on the proposed description mechanism and system concept points out the situation in FHEM with the description mechanism in place and without it.

4.5.1 FHEM without the Description Mechanism

Without the description mechanism, FHEM suffers from two main shortcomings.

a. **No API to create applications that can build on top of FHEM.**

   Without the specified description mechanism, FHEM only exposes a command oriented interface for interacting with the devices it supports. This means that users and developers must deal with the peculiarities of each technology type. Currently, this creates a barrier to adoption for prospective FHEM users since they first have to learn to work with the different technologies. Moreover, as there's no Application Programing Interface (API) with which apps can be developed independent of the underlying communication technology, the range of solutions is limited.

b. Interactions are grouped by technology not device category

There is currently no device-based separation of the interactions within FHEM. The commands are grouped by the technology, making it hard to find out what's required to operate devices installed in the system. For example, FS20 commands are all lumped together and it's difficult to find out how a dimmer's commands is different from that of a lamp. It's the same for HomeMatic and EnOcean devices as well.

4.5.2 FHEM with the Description Mechanism

With the proposed description mechanism and system concept for its application the situation in FHEM is different.

a. **An API to create applications on top of FHEM.**

With the proposed description mechanism in place, there's a consistent, uniform interface for developing applications based on the FHEM server. Since the API exposes methods on a device by device basis, it will be more straightforward to develop end user applications which rely on the abstraction provided by the API.

b. **Interactions based on device category.**

As the description mechanism is proposed based on device categories, the interactions are now well defined. For the user, interaction with a switch is the same, regardless of whether it's HomeMatic or FS20 based. This is possible by means of the device schemas which were defined for each of the categories that were considered.

5. CONCLUSION

In this work, a common description mechanism has been proposed to address an identified problem within a particular HA system, namely FHEM. The description mechanism comprised of a base schema which specifies at a high level, the structure of a device description. It is then used to derive descriptions that are specific to device categories.

While developing the device description mechanism, the research questions set out in the introduction were answered.

a) **How can a sufficient level of abstraction be reached to bridge between the specifics of smart devices from different vendors and in different countries?**

   By comparing the current representations of devices within FHEM and identifying the common elements, it is possible to create a generic device abstraction which specifies at a high level the most common aspects of a device. Such a generic abstraction can then be the basis for abstractions per device category. In this work for example, the generic abstraction was used as a basis for defining a dimmer description.

b) **What device description mechanisms exist in literature?**

   i. DeviceKit: it represents the functional aspects of a device by means of a 3-layered model.
   ii. Device Description Language: describes a device as a set of attributes covering information about the manufacturer, capabilities as well as an interface between the environment and internals of the device.
   iii. SensorML: presents a standard model and XML encoding for representing processes and sensor data.
   iv. IEEE 1451: it creates a common, network independent interface to enable interoperability between devices and networks.
   v. ECHONET: it specifies the device's properties and methods so that vendors can conform to a uniform interface.

c) **Can such existing description mechanisms be adapted to this problem? If yes, how?**

   i. This thesis has taken a cue from the DDL approach to specify device description files which will be parsed in order to create a logical model of the device in order

to facilitate a common interaction between devices in the same category. The proposed description was formalized as a JSON schema which will be parsed to create an in-memory representation of the device for interacting with it.

ii. Also, this thesis applied the idea from the ECHONET standard by specifying common descriptions for specific device categories. The descriptions were based on categories. For example, a dimmer device description which is based on the generic device schema.

d) **What parameters are common to the devices which can be abstracted to describe them?**

This thesis identified the common parameters which together can be used to create an abstract device description as elaborated in section 3.3. They were:

device_category, name/id,

state_attrs (with *attr* and *data_type*)

measure_attrs (with *attr* and *data_type*)

communication_tech

actions/commands (with *action_name*, *parameters* and *return_values/output*)

5.1 Research Contributions

This work has proposed as a contribution, a common description mechanism as well as a system concept for its integration into an open source HA system – FHEM. It also demonstrated the use of JSON as a schema representation format. The practicality of the proposed mechanism and system concept are demonstrated by means of a mobile application.

5.1.1 Environmental Contribution

Considering the potential of home automation systems to reduce the carbon footprint of domestic energy consumption, the thesis presents a streamlined means of interacting with devices in HA systems. This can be expected to lower the barrier to adoption of such open source systems like FHEM. The contribution is an indirect impact on the environment since it enables the adoption of home automation systems which could then influence the consumption patterns of users and consequently a cumulative reduction in the CO_2 footprint of homes.

5.1.2 Ethical Contribution

The streamlined description mechanism presented in this thesis enables a better experience for end users of the HA system. It enables a change from the previous situation in which users had to keep track of the differences in technology of devices in their HA installation. The proposed mechanism can enhance the users' experience with such a system as FHEM. Such an enhanced experience in using HA systems removes the need for expert knowledge barrier to adoption which is otherwise the case. As users can easily include and operate devices from different vendors and technologies, they can expect to have a positive experience from their use of HA systems.

5.1.3 Economic Contribution

The increase adoption of HA systems that can be expected from applying the streamlined description mechanism proposed in this thesis will yield economic benefits to end users. Savings in cost of energy consumption will be realized and cheaper alternatives to vendor-supplied HA systems can also be procured.

Further direction of work in this thesis are highlighted below.

a. Investigating the hypothesis that the proposed streamlined description mechanism will positively impact user adoption.

b. Exploring the possibility of implementing the layers of the proposed system concept within FHEM.
c. Extending the communication technologies beyond those considered in this work (i.e. EnOcean, FS20 and HomeMatic).

In order to promote the use/adoption of the proposed description mechanism and system concept especially within the open source FHEM community, the output of this thesis (device description files, proof of concept application) is being made available under an open source license and is accessible at this URL - https://github.com/adekola/prothesis

APPENDIX A

Below is the full device description for a Dimmer device in JSON, compliant with the JSON-Schema specification

A1: Specific Device Schema - Dimmer

```
{
    "device_type": "Dimmer",
    "attributes": [
        {
            "attr_name": "manufacturer_name",
            "attr_type": "string"
        },
        {
            "attr_name": "device_id",
            "attr_type" : "number"
        },
        {
             "attr_name" : "device_name",
             "attr_type" : "string"
        },
        {
            "attr_name": "comm_tech",
            "attr_type" : "string"
        }
    ],
    "actions": [
        {
            "action_name": "DimUp",
            "params": [],
            "return_values": [
                {
                    "value_name": "action_succeeded",
                    "value_type": "boolean"
                }
            ]
        },
        {
            "action_name": "DimDown",
            "params": [],
            "return_values": [
                {
                    "value_name": "action_succeeded",
                    "value_type": "boolean"
                }
            ]
        },
        {
            "action_name": "Toggle",
            "params": [],
            "return_values": [
                {
                    "value_name": "action_succeeded",
                    "value_type": "boolean"
                }
            ]
        },
```

```
{
    "action_name": "On",
    "params": [],
    "return_values": [
        {
            "value_name": "action_succeeded",
            "value_type": "boolean"
        }
    ]
},
{
    "action_name": "Off",
    "params": [],
    "return_values": [
        {
            "value_name": "action_succeeded",
            "value_type": "boolean"
        }
    ]
},
{
    "action_name": "OnFor",
    "params": [{
        "param_name" : "timeSpan",
        "param_type" : "number"
    }],
    "return_values": [
        {
            "value_name": "action_succeeded",
            "value_type": "boolean"
        }
    ]
},
{
    "action_name": "OnOffFor",
    "params": [{
        "param_name" : "timeSpan",
        "param_type" : "number"
    }],
    "return_values": [
        {
            "value_name": "action_succeeded",
            "value_type": "boolean"
        }
    ]
},
{
    "action_name": "OnTill",
    "params": [{
        "param_name" : "targetTime",
        "param_type" : "number"
    }],
    "return_values": [
        {
            "value_name": "action_succeeded",
            "value_type": "boolean"
        }
    ]
},
{
```

```
            "action_name": "OffTill",
            "params": [{
                "param_name" : "targetTime",
                "param_type" : "number"
            }],
            "return_values": [
                {
                    "value_name": "action_succeeded",
                    "value_type": "boolean"
                }
            ]
        },
        {
            "action_name": "DimUpBy",
            "params": [{
                "param_name" : "amount",
                "param_type" : "number"
            }],
            "return_values": [
                {
                    "value_name": "action_succeeded",
                    "value_type": "boolean"
                }
            ]
        },
        {
            "action_name": "DimDownBy",
            "params": [{
                "param_name" : "amount",
                "param_type" : "number"
            }],
            "return_values": [
                {
                    "value_name": "action_succeeded",
                    "value_type": "boolean"
                }
            ]
        }

    ]
}
```

APPENDIX B

Following is the prototype implementation of the Dimmer device description. It's in Java in the context of an Android mobile application

B1: Device Description Implementation – Dimmer

```
package modules;

import android.os.Parcel;
import android.os.Parcelable;

import java.util.Date;

import javax.xml.datatype.Duration;

/**
 * Created by kola on 6/11/2016.
 */
public class Dimmer implements Parcelable {


    public String _deviceName;
    public String _deviceManufacturer;
    public String _tech;


    protected Dimmer(Parcel in) {
        _deviceName = in.readString();
        _deviceManufacturer = in.readString();
        _tech = in.readString();
    }

    public static final Creator<Dimmer> CREATOR = new Creator<Dimmer>() {
        @Override
        public Dimmer createFromParcel(Parcel in) {
            return new Dimmer(in);
        }

        @Override
        public Dimmer[] newArray(int size) {
            return new Dimmer[size];
        }
    };


    //doubles as the unique device ID
    public String getDeviceName() {
        return _deviceName;
    }

    public void setTech(String _tech) {
        this._tech = _tech;
    }

    public String getTech() {
        return _tech;
    }

    public Dimmer(String _deviceName, String _deviceManufacturer, String _tech) {
        this._deviceName = _deviceName;
        this._deviceManufacturer = _deviceManufacturer;
        this._tech = _tech;
    }
```

```
    @Override
    public String toString() {
        return getDeviceName();
    }

    @Override
    public int describeContents() {
        return 0;
    }

    @Override
    public void writeToParcel(Parcel parcel, int i) {
            parcel.writeString(_deviceName);
            parcel.writeString(_deviceManufacturer);
            parcel.writeString(_tech);
    }


    public boolean dimUp(){
        return Mapper.Do("dimUp", this);
    }

    public boolean dimDown(){
        return Mapper.Do("dimDown", this);
    }

    public boolean toggle(){
        return false;
    }

    public boolean on(){
        return Mapper.Do("on", this);
    }

    public boolean off(){
        return Mapper.Do("off", this);
    }

    public boolean onFor(Duration duration){
        return false;
    }

    public boolean ofFor(Duration duration){
        return false;
    }

    public boolean offTill(Date targetDate){
        return false;
    }

    public boolean onTill(Date targetDate){
        return false;
    }

    public boolean dimDownBy(double amount){
        return false;
    }

    public boolean dimUpBy(double amount){
        return false;
    }
}
```

BIBLIOGRAPHY

Aberer, K., Hauswirth, M. & Salehi, A., 2006. Middleware support for the "Internet of Things." *5th GI/ITG KuVS Fachgespräch "Drahtlose Sensornetze"*, (5005), p.5.

Amaral, L.A. et al., 2015. Cooperative middleware platform as a service for internet of things applications. *Proceedings of the 30th Annual ACM Symposium on Applied Computing - SAC '15*, pp.488–493. Available at: http://dl.acm.org/citation.cfm?doid=2695664.2695799.

Boman, J., Taylor, J. & Ngu, A., 2014. Flexible IoT Middleware for Integration of Things and Applications. *Proceedings of the 10th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing*, (CollaborateCom), pp.481–488. Available at: http://eudl.eu/doi/10.4108/icst.collaboratecom.2014.257533.

Bonino, D., Castellina, E. & Corno, F., 2008. The DOG gateway: Enabling ontology-based intelligent domotic environments. *IEEE Transactions on Consumer Electronics*, 54(4), pp.1656–1664.

Chattopadhyay, D., 2012a. A Novel Comprehensive Sensor Model for Cyber Physical System Interoperability for heterogeneous sensor. , pp.179–183.

Chattopadhyay, D., 2012b. A Survey of Available Sensor Data Modeling Techniques.

Chattopadhyay, D., Dasgupta, R. & Pal, A., 2013. Sensor data modeling for smart meters - A methodology to compare different systems. *2013 International Conference on Computing, Networking and Communications, ICNC 2013*, pp.215–221.

European Commission, 2014. EU 2030 Energy Strategy. Available at: http://ec.europa.eu/energy/en/topics/energy-strategy/2030-energy-strategy [Accessed April 20, 2016].

European Commission, 2010. EU Energy Efficiency of Buildings. Available at: http://ec.europa.eu/energy/en/topics/energy-efficiency/buildings [Accessed April 20, 2016].

Gauger, M., Minder, D. & Lachenmann, A., 2008. Prototyping Sensor-Actuator Networks for Home Automation. , pp.56–60.

Gonnot, T. & Saniie, J., 2014. User defined interactions between devices on a 6LoWPAN

network for home automation. *2014 IEEE International Technology Management Conference, ITMC 2014*.

JSON Schema, 2016. The Home of JSON Schema. Available at: http://json-schema.org/ [Accessed May 26, 2016].

Juárez, J., Rodríguez-Mondéjar, J.A. & García-Castro, R., 2014. An ontology-driven communication architecture for spontaneous interoperability in Home Automation systems. *19th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2014*.

Kim, J.E. et al., 2012. Seamless Integration of Heterogeneous Devices and Access Control in Smart Homes. *2012 Eighth International Conference on Intelligent Environments*, pp.206–213. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6258524.

LinkSmart Open Source Team, 2014. Hydra. Available at: http://www.hydramiddleware.eu/news.php [Accessed May 21, 2016].

Maeda, K., 2012. Performance Evaluation of Object Serialization Libraries in XML , JSON and Binary Formats. , pp.177–182.

Ml, I., 2009. On Inference of XML Schema with the Knowledge of an Obsolete One. , 92(Adc).

Nachabe, L., Girod-genet, M. & Hassan, B. El, 2015. Unified Data Model for Wireless Sensor Network. , 15(7), pp.3657–3667.

Nasrin, S. & Radcliffe, P.J., 2015. Novel protocol enables DIY home automation. *2014 Australasian Telecommunication Networks and Applications Conference, ATNAC 2014*, pp.212–216.

NEST Labs, 2016. NEST Learning Thermostat. Available at: https://nest.com/thermostat/meet-nest-thermostat/ [Accessed May 20, 2016].

Nurseitov, N. et al., 2009. Comparison of JSON and XML Data Interchange Formats : A Case Study.

Nye, P., 2014. Device description language: Factorizing the control of arbitrary networked devices. *Proceedings - 2014 International Conference on Future Internet of Things and*

Cloud, FiCloud 2014, pp.346–350.

Park, S.O., Kim, J.S. & Kim, S.J., 2013. An object-based middleware supporting efficient interoperability on a smart home network. *Multimedia Tools and Applications*, 63(1), pp.227–246.

Paz-Lopez, A. et al., 2012. DAAF: A device abstraction and aggregation framework for smart environments. *Proceedings - 6th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, IMIS 2012*, pp.739–744.

Pires, P.F. et al., 2014. A Platform for Integrating Physical Devices in the Internet of Things. *Proceedings - 2014 International Conference on Embedded and Ubiquitous Computing, EUC 2014*, pp.234–241.

Ryan, J.L., 1989. Home automation. *Electronics & Communication Engineering Journal*, (August), pp.185–192.

Samsung, 2016. Samsung Smarthings. Available at: https://www.smartthings.com/ [Accessed May 20, 2016].

Sangogboye, F.C., 2016. HomeAutomation - Using OpenSource to fulfill EU-Directives -.

Sangogboye, F.C., Droegehorn, O. & Porras, J., 2015. Analyzing the payback time of investments in Building Automation.

Song, J. et al., 2014. Parameter-Based Mechanism for Unifying User Interaction, Applications and Communication Protocols. *2014 2nd International Conference on Artificial Intelligence, Modelling and Simulation*, pp.131–136. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7102448.

Space Telescope, 2016. JSON Schema. Available at: http://spacetelescope.github.io/understanding-json-schema/index.html [Accessed May 25, 2016].

Statista, 2016a. Home Automation Globally. *Digital Market Outlook*. Available at: https://www.statista.com/outlook/280/100/home-automation/worldwide# [Accessed May 28, 2016].

Statista, 2016b. Home Automation in Germany. *Digital Market Outlook*. Available at:

https://www.statista.com/outlook/280/137/home-automation/germany# [Accessed May 28, 2016].

Torbensen, R., Hansen, K.M. & Hjorth, T.S., 2011. My home is my bazaar - A taxonomy and classification of current wireless home network protocols. *Proceedings - 2011 2nd Eastern European Regional Conference on the Engineering of Computer Based Systems, ECBS-EERC 2011*, pp.35–43.

Villalonga, C. et al., 2010. Modeling of Sensor Data and Context for the Real World Internet. , pp.1–6.

Wang, M., 2010. An XML Schema-Based Data Integration Chong-Shan Ran. , pp.100–102.

Withanage, C. et al., 2014. A comparison of the popular home automation technologies. *2014 IEEE Innovative Smart Grid Technologies - Asia, ISGT ASIA 2014*, pp.600–605.