Institute of Visualization and Interactive Systems

University of Stuttgart
Pfaffenwaldring 5a
D–70569 Stuttgart

Bachelorarbeit Nr. 181

A Visual Approach for Orchestrating Smart Devices in the Internet of Things

Matthias Mögerle

Course of Study:	Informatik
Examiner:	Prof. Dr. Albrecht Schmidt
Supervisor:	M.Sc. Thomas Kubitza, Dr.-Ing. Darren Carlson, M.Sc. Max Pagel
Commenced:	August 11, 2014
Completed:	February 10, 2015
CR-Classification:	H.5.2

Abstract

We are currently witnessing the invention of more and more smart devices, which are having an increasing impact on our daily life. Smart devices perform small tasks in our environment. Most smart devices can only be controlled by direct user interactions e.g. mobile applications, which are often limited to only one user and one smart device. This restriction is hiding potential use-cases of direct smart device intercommunication, which is often impossible due to the variety of network technologies used by smart devices. Current mobile devices, such as mobile phones and tablets are providing a wide range of network technologies and are carried by the user most of the time. This makes mobile devices a great mediator to interconnect smart devices. There are already existing frameworks which provide this functionality but they are often very complex to use for users without programming experience. Therefore, the contribution of this thesis is a visual user-interface, which allows users without programming experience to connect smart devices with each other. Visual programming is used to represent devices as visible device nodes, which can be connected by edges to visualize the connection between smart devices. To collect reasonable feedback, a user study was conducted, where users had to use the visual user-interface prototype. This proved the usability of the implemented prototype and improved our belief in the necessity of this approach.

Kurzfassung

Zur Zeit sehen wir eine starke Entwicklung immer neuer intelligenter Geräte (smart devices), welche einen immer größeren Einfluss auf unser tägliches Leben haben. Intelligente Geräte führen in unserer Umgebung kleine Aufgaben aus und können nur über einen direkten Zugriff gesteuert werden, wie z.B. durch mobile Anwendungen. Diese Zugriffe sind meist auf einen Nutzer und ein Gerät beschränkt. Dadurch bleibt großes Potential der Nutzung mit einer direkten Kommunikation zwischen Geräten verwehrt. Dies ist auf die große Anzahl verschiedener Netzwerk Technologien zurück zu führen, die eine direkte Kommunikation schwierig machen. Heutige Smartphones oder Tablets haben eine Vielzahl an Netzwerk Technologien und werden die meiste Zeit vom Nutzer mitgenommen. Darum bieten sich mobile Geräte als Verbindungsglied zwischen intelligenten Geräten an. Es gibt bereits Software Systeme, welche den Nutzen des mobilen Gerätes verwenden um damit intelligente Geräte zu verbinden. Die Bedienung davon benötigt jedoch Programmiererfahrung. Darum trägt diese Arbeit dazu bei, eine grafische Schnittstelle zu erstellen, welche von Nutzern ohne Programmierkenntnisse verwendet werden kann um intelligenten Geräten miteinander zu verbinden. Zur Realisierung wurde der Ansatz des grafischen Programmierens verwendet, so dass intelligenter Geräte als Knoten visuell dargestellt werden. Verbindungen zwischen Geräten werden als Kanten zwischen Knoten dargestellt. Um die Meinung der Benutzer zur Verwendung der visuellen Nutzeroberfläche zu bewerten, wurde eine Nutzerstudie durchgeführt. Diese belegte die Nutzbarkeit der entwickelten Oberfläche und bestärkt unseren Glauben an die Wichtigkeit des grafischen Ansatzes.

Contents

List of Figures

1 Introduction

More and more smart devices are entering our daily life. They are platforms, designed to perform small tasks in our environment. This can be to perform an action or sense data like movements or gestures. For example, the light color and intensity of smart light bulbs can be set by using an app on a mobile devices. Another functional smart device is called smart switch which has the same functions as a regular switch (on/off). In addition, the user can control the switch remotely with an user-interface. This allows further funcitonalities such as an integrated time-switch to perform automated actions.

These smart devices are accessible through network connections which allows to address them individually. A network of smart devices is called Internet of Things (IoT) and enables a large amount of possible interactions. For example, smart devices used for home automation can be controlled remotely to turn of all the lights at home. Mobile phones can be connected with smart devices like speakers to play music. And in factories, smart devices are used to track productivity parameters or machine states.

1.1 Statement of the Problem

It can be imagined to implement more interactions with smart devices. An example is to add a cinema feature at home, lights could be dimmed when starting to watch a movie. This requires interactions between the home automation and smart media devices. Both actions can be done independently, via a user's mobile device (e.g. smart phone or tablet). However performing them automatically, as in one action triggers the other, is currently only achievable with a lot of domain knowledge and almost always requires programming and IT administration skills.

Generally, most smart devices are connected to a mediator which is called smart gateway to translate different protocols. This is especially required by smart devices using connections without IP-reference such as Bluetooth and IEEE 802.15.4. This allows to integrate smart devices into the Internet of Things (IoT) which describes all smart devices which are accessible through the internet. Aside of the functionality to bridge networks, smart gateways manage access permissions which provide additional security features. Many wireless connection technologies and protocols are supported by mobile devices (e.g. smart phones or tables). In addition, they are carried by the user most of the time. This combination suggest mobile devices as personal smart gateways.

Various frameworks were developed which are providing user-interfaces to access individual smart devices. However, little work is done to seamlessly let these devices interact with each other, this completely neglects the potential of connecting smart devices with each other. Direct messages can be send to each other and create truly automated and interactive environments based on the available devices in one's surroundings. Even though some frameworks have the capability to orchestrate and even connect smart devices in an environment with each other, they are often not usable by regular users because of their complexity.

Therefore, the contribution of this thesis is to introduce a visual user-interface which allows users without programming experience to connect smart devices with each other. These connections can grow to a graph orchestration of smart devices. This enables user to be creative and create new use-cases which suit their requirements best.

1.2 Structure of the Thesis

Outline

This paragraph describes the structure of this thesis. After the introduction we start with related work (starting on page 9). The section of related work covers approaches to connect smart devices by defining rules, automatically computed and visually orchestrated which leads to the introduction of visual programming. Foundations (starting at pate 15) are introduced right after related work and describe projects to interact with smart devices. The system concept (starting at page 21) is introducing the project structure and the concept of the visual web user-interface. Hereafter details about the implementation (starting at page 31) are declared. The user study with the prototype is reported in the evaluation chapter (starting at page 37). At the end of this thesis, conclusion and future work are formulated (starting at page 49).

2 Related Work

There is a large body of work in the area of smart device interactions. In this section, we focus our discussion of related work on approaches which are rule based, automatically computed and visually orchestrated.

All of these approaches have their specific benefits to orchestrate smart devices in behalf of the user. Each introduced approaches is explicitly considered in focus of end-user interactions.

2.1 Rule Based

A rule based agent is introduced by [GHHA10] García-Herranz et al.. They invented an end-user independent rule language, which can be used to employ rules for smart device interactions in various applications. Their research focus on studies and tests to achieve the maximal understanding of their rules for various user groups, such as a separation between software developers and non software developers. The overall goal is to allow end-users to control and program their own smart environments. A rule could be for example, "When the professor is entering the lecture theater then...". As part of their studies, they compared applicable rules for working environments with rules for personal homes. It became clear, that working environments are more structured and therefore simpler to describe by rules.

However, the user is always required to learn at least some rules of the rule-set, before it is possible for him to define rule of his environment. This is a burden for first-time users of this interface and could avoid creativity.

2.2 Artificial Computing

In [VBCMV+12] Vega-Barbas et al. introduces the use of artificial computing to control smart devices. All smart devices are integrated in and adaptive network, which is creating a local smart space. The specific architecture is called S^3OiA and is learning through intents of the user as shown in figure 2.1. It can be pre-configured and adapts its settings while the user is interacting within the smart space. For example, a user makes coffee for breakfast every morning after entering the kitchen. After a while the system recognizes this pattern and is making coffee for him.

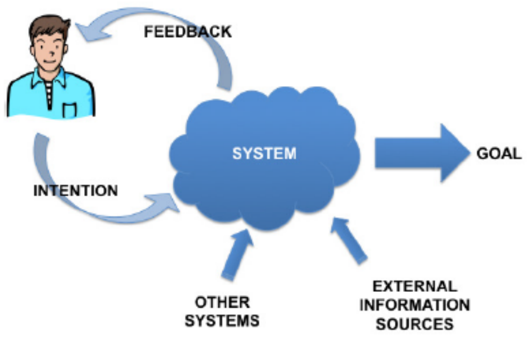


Figure 2.1: S^3OiA is recognizing user intents and uses them to optimize the system, figure from [VBCMV$^+$12].

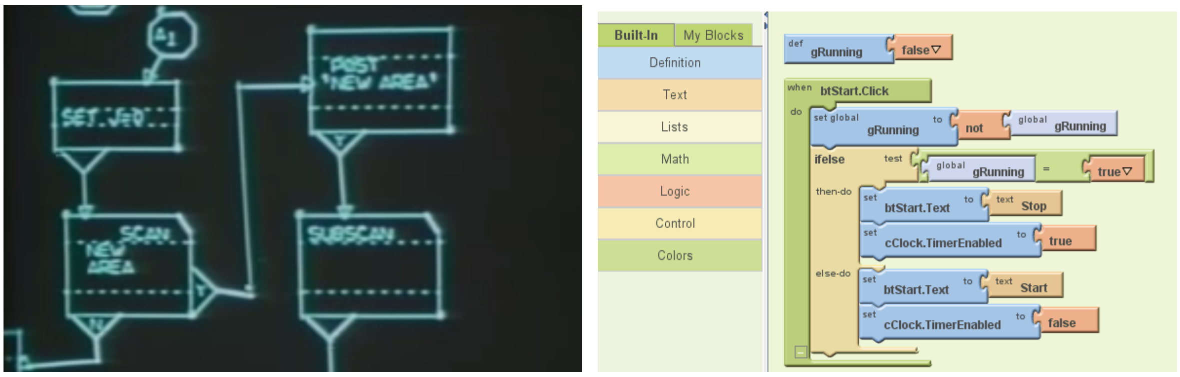
However, this approach works well for highly equipped smart environments containing a lot of sensors and actuators. Usually, an environment does not contain as many sensors as required to make improvements through learning. Another issue is, that we focus on a fast changing environment, where a user may want to connect two smart devices instantly. When this functionality is replaced by the leaning process, the user is not able to introduce his own ideas and creativity.

2.3 Visual Programming

Several projects and approaches are connecting smart devices through a visual user-interface before it is deployed to smart devices. The most common type of graphical user-interfaces is called program visualization and is introduced in the next paragraphs. Hereafter, various related work projects are shown, using the introduced program visualization.

To be able to write a program requires the ability to read and write code. It takes time to learn syntax details and be able to create a complex program. To solve this, graphical programming was first introduced at MIT called GRAIL (GRAphical Input Language [EHS69]) in 1968. Each functionality is represented by a node, which can be connected with other nodes. A program is implemented through connecting visual nodes which are representing a sequence of functionalities.

The human brain is trained to analyze visual input. This attribute is addressed through graphical elements of visual programming user-interfaces. These are using more-dimensional elements to represent program functionalities, while the letters of a usual code editor can be seen as one-dimensional. Therefore, visual programming user-interface do not contain code


SET. VED
POST "NEW AREA"
SCAN "NEW AREA"
SUBSCAN
Built-In
My Blocks
Definition
Text
Lists
Math
Logic
Control
Colors
def gRunning false
when btStart.Click
do set global gRunning to not global gRunning
ifelse test global gRunning = true
then-do set btStart.Text to text Stop
set cClock.TimerEnabled to true
else-do set btStart.Text to text Start
set cClock.TimerEnabled to false

(a) (b)

Figure 2.2: (a) The first visual programming interface, GRAIL [EHS69]. (b) Instead of edges, nodes are connected as puzzle pieces, `http://appinventor.mit.edu/`

as known from textual program editors. Instead they focus on visual elements to generate a program.

The graphical representation allows most users without programming experience to understand visual programs way faster then they could understand pure program code. It must be considered that the simplification absorbs some of the flexibility and functionality pure code provides. But still, the implementation of the visual programming user-interface can be done well, so that the user can adjust all parameters required for this particular use-case. A remarkable benefit is, that programs can be build straight forward without major knowledge. Therefore, visual programming is often used to fill the gap of programming a simple prototype without getting into deep details, introduced by [Mye90] Myers.

While the usual visualization approach is to connect nodes through edges, Scratch [RMMH$^+$09] introduced another visualization. Scratch is using puzzle pieces as nodes having the feature that only fitting puzzle pieces can be connected. Various elements, such as loops and other statements of a usual program are visualized and can be puzzled. This approach is used by Appinventor [1] to develop Android applications via puzzle pieces. After the puzzle orchestration is completed, it used to automatically generate the real source files and pack them to an Android mobile application. This allows a user without programming experience to use his own creativity and generate a real application.

All of the following approaches are connecting smart devices through visual programming user-interfaces.

[1]`http://appinventor.mit.edu/`

2.3.1 SAM Labs

SAM[2] is an IoT development kit, based on a set of smart devices. All of them are connected per Bluetooth LE and are physically standalone smart devices, since a battery provides energy for each smart device to run autonomously. Each smart device provides a simple functionality which can be buttons, LEDs and speakers for example. The user can arrange these smart devices by the use of a visual programming interface, running on the computer (figure 2.3). The program is using nodes and edges to connect smart devices and orchestrate their functions per drag and drop. Aside of hardware integration, software functionalities such as social media access are additionally integrated.

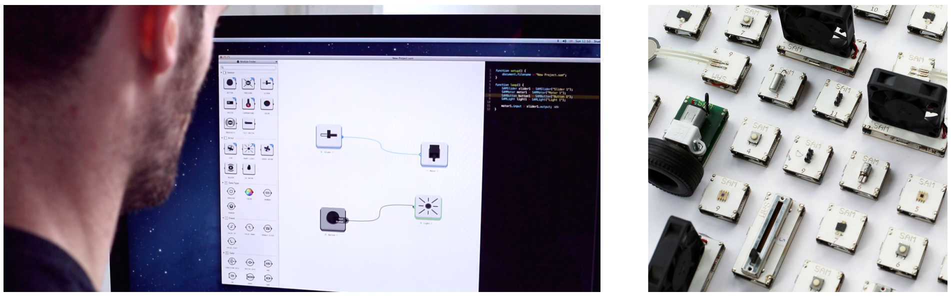


Figure 2.3: SAM contains many compact smart devices which can be connected through flow-based-programming. (Photo courtesy of: http://samlabs.me/)

However, SAM does provide a great platform to teach the orchestration of hardware elements. The focus of SAM is the local development environment, while the personal smart space we are focusing is a changing environment which requires a wider and more dynamic approach. In addition, the smart devices of SAM are only limited to some basic functionalities, but it is not possible to integrate off-the-shelf smart devices containing more functionalities.

[2] http://samlabs.me/

2.3.2 Node Red

A flow based approach to support various functionalities and smart devices is given by Node Red [3]. Node Red is a browser-based, data-flow editing framework and enables to visually orchestrate programs. Functional blocks, so-called nodes can represent databases, functions or smart devices. These nodes can be wired together and build a functional graph. This graph can already be deployed during the time of its orchestration. The nodes are encapsulating some functionalities, while it is possible for the developer to implement functions within code, added to a node. This is a great way to rapidly develop a first prototype using existing parts, see figure 2.4.

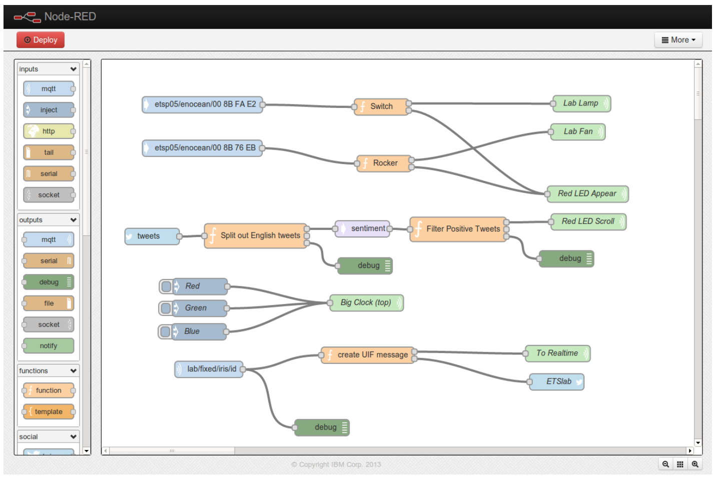


Figure 2.4: Node Red is enabling visual programming by the use of functional nodes which can be wired together. (Photo courtesy of: Node Red)

However, Node Red is connecting smart devices such as lights and switches, the user-interface is primary defined for developers. This makes it very hard for users without programming experience to implement interactions between smart devices which they have not done before. Another point against Node Red is that is must be installed as a mediator on a host, such as Raspberry Pi or Arduino, this is not designed for a moving personal smart space.

[3]http://nodered.org/

3 Foundations

A complete solution to connect smart devices with each other consists of several components. This section is introducing frameworks and concepts to connect smart devices, based on which the proposed user-interface is implemented. To solve the aforementioned problem of heterogeneous device landscapes, Ambient Dynamix [CS12] was developed to interact with smart devices and is now explained in more detail.

3.1 Ambient Dynamix - Control IoT Devices

A mobile device like a smart phone or tablet contains various communication technologies to connect with smart devices (e.g. Bluetooth or WI-FI). This functionality is often required by smart devices which can be controlled through native mobile applications. These applications can be separated into an user-interface and a connection layer which interacts with the smart device. Whenever a developer wants to implement a smart device into a new native application, the complete connection code must be duplicated or developed from scratch. This takes time or can cause unexpected errors.

Dynamix can be installed on android mobile devices and provides a solution for this problem. During runtime, it works as a background service and makes use of the phone's various network interfaces. This allows Dynamix to access smart devices (shown on the left side of 3.1), which are connected to the mobile device. Examples of these smart devices can be bluetooth speakers or network integrated smart light bulbs.

The Dynamix service works as an IoT gateway, which provides access to connected smart devices for third party applications. To do that, Dynamix provides simple application programming interfaces (APIs, shown on the right side of figure 3.1). These APIs enable developers of native android or web applications to use standardized interfaces instead of having to learn specific code for each smart device. In every application, Dynamix support can simply be added by a few lines of code.

To provide access for a smart device, Dynamix must integrate the individual interface which enables the interaction with the smart device and is called plug-in. An open plug-in development SDK is available, which allows independent developers to develop their own plug-in. This enables Dynamix to communicate with a connected smart device. These device plug-ins are publicly available through a plug-in repository. Third-party applications can request support for

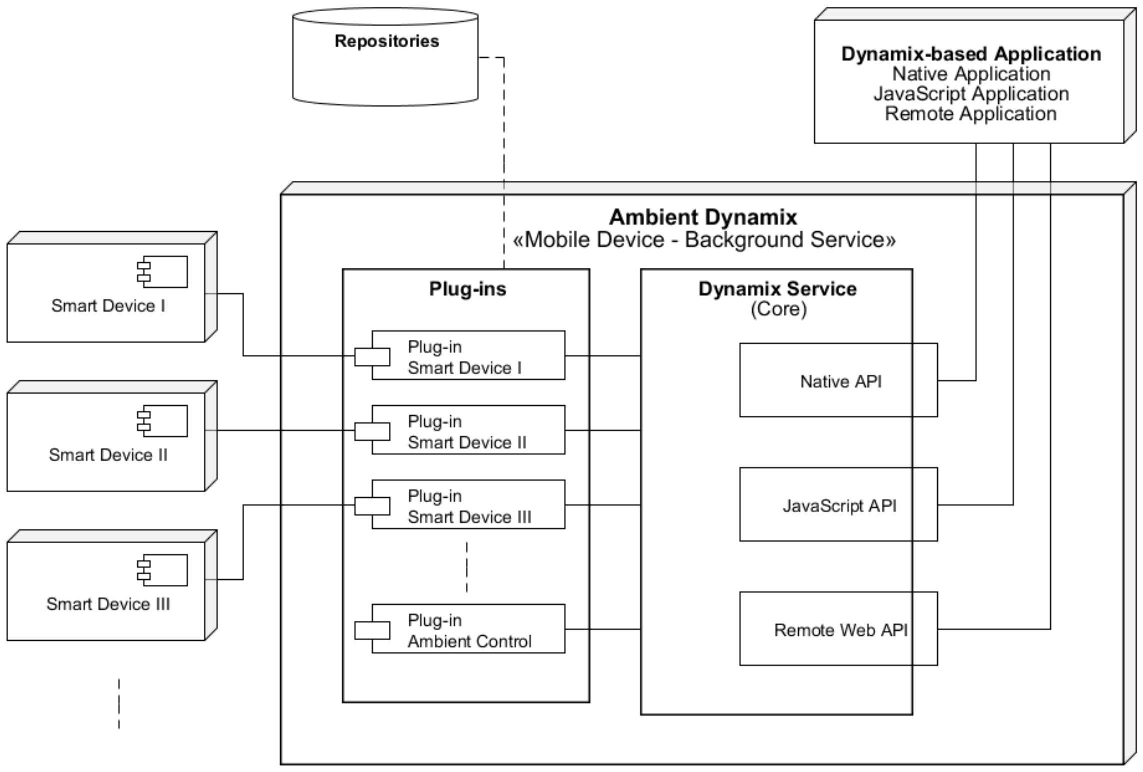


Figure 3.1: Overview of Dynamix. Smart devices are reached through plug-ins from a repository. APIs allow third party applications to interact with smart devices.

a specific device they want to interact with. Dynamix is then downloading the corresponding plug-in from the repository and installs it. After this, the third party application can use Dynamix to communicate with the integrated smart device.

Usually, the user is continuously moving around the mobile device, because a smart phone or tablet can be used everywhere. Since the environment changes often, the set of available smart devices is not static. It depends on the position of the mobile device and the position of the smart devices to register if smart devices are reachable. Therefore, the smart environment of a mobile device can change rapidly, smart devices can appear or are not accessible anymore. To suit this fast changing environment, plug-ins can be installed within Dynamix at runtime. An additional feature is that while IoT requirements are changing, Dynamix is able to stay up to date through integrating the latest plug-ins.

Plug-ins must not necessarily be device plug-ins. It is also possible to load functional plug-ins within Dynamix. An example is the Ambient Control framework, which is implemented as a library plug-in. The integration of Ambient Control within Dynamix is shown in figure 3.1.

Ambient Control extends Dynamix by enabling the sending of messages between smart devices, while it works as a mediator in between them.

3.2 Ambient Control - Inter Connecting Smart Devices

Today, most smart devices can be accessed through native applications running on mobile devices. These are supporting connections between a mobile device and a smart device, such as to control a smart light bulb by the use of a mobile application user-interface. Meanwhile, our environment transforms towards a smart space, where smart devices are not only used by direct user interactions. They are integrated in complex use-cases, where a combination of actions is performed. These involve more than one smart device, e.g. to lock the door of a building, which is instantly closing all the windows and turn off the lights in each room. Very often, direct interactions between smart devices are not supported due to their use of heterogeneous protocols.

Ambient Control can be added to Ambient Dynamix and focuses on orchestration and connection of heterogeneous interactions between smart devices in a flexible and dynamic way. The mobile device is used to mediate messages between smart devices in a personal smart space.

Each smart device is able to contain inputs as well as outputs. A smart light bulb has an input for the light color and another smart device has a movement sensor sharing movement data through its output. Two smart devices are connected after at least one output of the first smart device is able to send messages to an input port of a second smart device. That's the minimum relation to connecting two smart devices.

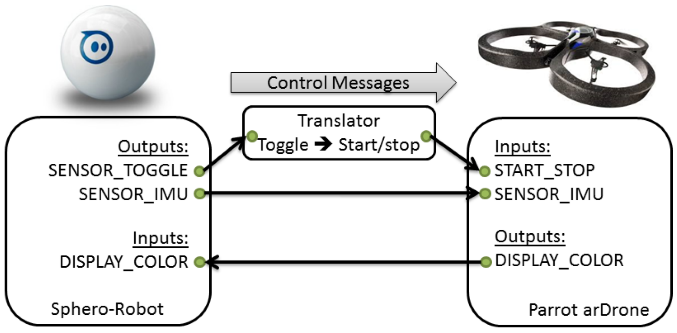
Whenever relations between entities are given, the model of a graph can be applied. A graph contains nodes to represent each entity and edges to connect these nodes. Introducing this in the world of smart devices, each smart device is represented by a graph node, where edges between nodes are representations for connections. These are used to send messages between an output and an input.

To facilitate this, each smart device supported by Ambient Control has a standardized plug-in description. This description contains information about input requirements and output possibilities of a smart device, which can be seen in figure 3.2a. Inputs and outputs have command types which are required to match in order to connect input and output. For example, command types can be movement commands like *up, down, left, right* which depend on the smart device. These command types describe the expected format of the message sent between connected smart devices.

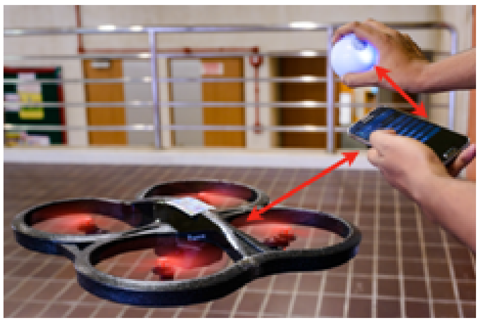
To inter-connect smart devices, the graph configuration must be deployed on the mobile device, which then mediates between smart devices. After evaluating the graph configuration, Ambient Control requests Dynamix to download and install necessary device plug-ins as introduced in

3.1. All integrated smart device plug-ins can be accessed through Ambient Control which uses the graph configuration to forward messages between smart devices.

In case both command types are not equal, a translator can be employed to mediate between different command types. Such a connection between two unequal command types using a translator in between can be seen in 3.2a. The Sphero robotic ball does not match all the inputs of the Ar Drone Parrot helicopter. Translators are added within the repository and have inputs and outputs like plug-in descriptions. The function of a translator is to use the input values from one command type and calculates the output values of another type. Explicit details about descriptions, translators and connections are given in 3.3.



(a)



(b)

Figure 3.2: Ambient Control is connecting a Sphero robotic ball and the Ar Drone Parrot helicopter. (a) Sketch of smart device inputs and outputs, figure from [PC14]. (b) Drone controlled by Sphero. (Photo courtesy of: Max Pagel)

3.3 Ambient Control - Communication

The implemented prototype of this thesis uses Ambient Control to deploy the graph orchestrations of the visual user-interface to smart devices. Therefore, a specific graph representation format must be used to be supported by Ambient Control. Details of this representation are introduced in the following paragraphs.

To interact with smart devices, Ambient Control is using plug-ins of smart devices which can be dynamically installed in Dynamix. Each smart device plug-in used by Ambient Control, follows the same description. It contains command types of inputs and outputs as introduced in the previous section. This plug-in description is named plug-in control description (PCD). Plug-in capabilities are described by:

Mandatory Input Profile

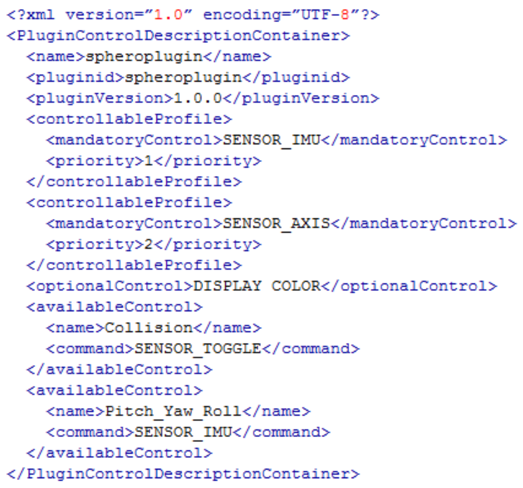
A profile describes a set of inputs, which are described by command types. A plug-in description can contain more than one mandatory input profile. At least one profile must be fulfilled in order to connect the inputs of this plug-in.

Optional Input

Inputs which are not essential are optional controls. For example speakers can contain a status LED. This LED does not influence the main function to produce sound and can therefore be controlled optionally.

Output

Outputs describe which commands are provided by a specific device. These can be sensor data such as the orientation of a device which can be provided via *SENSOR_IMU* outputs.



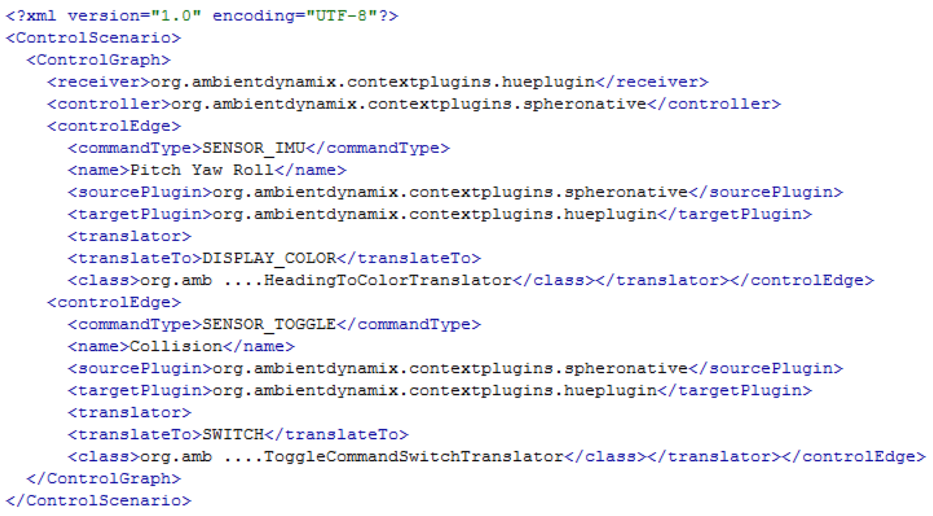
```xml
<?xml version="1.0" encoding="UTF-8"?>
<PluginControlDescriptionContainer>
  <name>spheroplugin</name>
  <pluginid>spheroplugin</pluginid>
  <pluginVersion>1.0.0</pluginVersion>
  <controllableProfile>
    <mandatoryControl>SENSOR_IMU</mandatoryControl>
    <priority>1</priority>
  </controllableProfile>
  <controllableProfile>
    <mandatoryControl>SENSOR_AXIS</mandatoryControl>
    <priority>2</priority>
  </controllableProfile>
  <optionalControl>DISPLAY COLOR</optionalControl>
  <availableControl>
    <name>Collision</name>
    <command>SENSOR_TOGGLE</command>
  </availableControl>
  <availableControl>
    <name>Pitch_Yaw_Roll</name>
    <command>SENSOR_IMU</command>
  </availableControl>
</PluginControlDescriptionContainer>
```

Figure 3.3: This XML snippet shows a PCD of a Sphero containing mandatory inputs, optional inputs and outputs

Each PCD is a XML snippet, such as the PCD of the Sphero robot ball in figure 3.3. The description specifies two mandatory inputs called control profile, the preferred profile contains *SENSOR_IMU*, which has the higher priority and controls the Sphero via data from an orientation sensor (IMU). The second control profile is using *SENSOR_AXIS* which is the command type of data provided by a joystick. Priorities are used to rank profiles with graceful degradation of control. Through the optional control of *DISPLAY_COLOR*, the color of the Sphero can be set which is not necessary and therefore optional. Provided outputs are *SENSOR_TOGGLE* which notifies about collisions and *SENSOR_IMU* which provides orientation data of the Sphero.

In case output the command type and the addressed input command type are not equal, transferred messages from the output are undefined for the input. To solve this, translators can be used to mediate between unequal outputs and inputs. They have descriptions as smart device-plug-ins which contain inputs and outputs.

The configuration to connect smart devices is defined through the so-called control graph which can be seen in figure 3.4. It is defined by at least one controller and and exactly one receiver and describes the connection between outputs and inputs per control edges. Control edges are defined by command type and name as well as source plug-in and target plug-in. In case a translator is required, it is added within the description of the edge.



```xml
<?xml version="1.0" encoding="UTF-8"?>
<ControlScenario>
  <ControlGraph>
    <receiver>org.ambientdynamix.contextplugins.hueplugin</receiver>
    <controller>org.ambientdynamix.contextplugins.spheronative</controller>
    <controlEdge>
      <commandType>SENSOR_IMU</commandType>
      <name>Pitch Yaw Roll</name>
      <sourcePlugin>org.ambientdynamix.contextplugins.spheronative</sourcePlugin>
      <targetPlugin>org.ambientdynamix.contextplugins.hueplugin</targetPlugin>
      <translator>
      <translateTo>DISPLAY_COLOR</translateTo>
      <class>org.amb ....HeadingToColorTranslator</class></translator></controlEdge>
    <controlEdge>
      <commandType>SENSOR_TOGGLE</commandType>
      <name>Collision</name>
      <sourcePlugin>org.ambientdynamix.contextplugins.spheronative</sourcePlugin>
      <targetPlugin>org.ambientdynamix.contextplugins.hueplugin</targetPlugin>
      <translator>
      <translateTo>SWITCH</translateTo>
      <class>org.amb ....ToggleCommandSwitchTranslator</class></translator></controlEdge>
  </ControlGraph>
</ControlScenario>
```

Figure 3.4: This XML snippet shows a scenario, which contains control graphs and represents smart device connections.

It should be possible to connect more then one receiver within a smart space. Therefore, graphs can be grouped by using a scenario, which contains various graphs. Ambient Control requires a scenario as input, in order to configure the surrounding smart devices. The visual user-interface must exactly provide the requested layout to describe the connection of smart devices.

The previous part of this thesis introduced various approaches to support end-users connecting smart devices. Especially visual programming is used for large amount of projects, which mostly focus on static orchestrations. To benefit from a dynamic orchestration of devices, Ambient Dynamix and Ambient Control are introduced to handle the technical part of connecting smart devices with each other. The next chapter introduces the system concept and is focusing on the part of the user-interface.

4 System Concept

The rise of smart devices requires new approaches to integrate them in our environments. While a lot of research was already done, they mostly focused on approaches for static environments. In contrast, this thesis contributes to mobile and personalized smart environments. One purpose of this thesis is that users must be able to set up their own connections between smart devices. For users without programming experience it is difficult initiate direct interactions between smart devices. Therefore, it is important to offer a user-interface which assists during the process of connecting smart devices. Necessary user-interface attributes are:

Uniform smart device representation
: All device representations must follow the same pattern. Each type of smart device must have its own representation. Each smart device must be addressable.

Structured connections
: Users must exactly know what a specific connection implies and which devices can be connected.

Reusable
: To save time and reduce the possibility of errors, device orchestrations must be reusable. It must be possible to personalize predefined configurations.

Support creativity
: Users must feel confident following their own ideas and use the user-interface as a creative tool.

Visual programming defines simple graphical elements, which are connected in order to build a program. Some approaches of visual programming are graph representations, where each node embodies a specific function. Usually, each node has the same structure while small layout differences depict variations in the specific node functionality. As introduced by Pagel et al. [PC14], smart devices can be viewed as nodes that have inputs and outputs which can be used to differentiate between them. The visual user-interface allows to add device nodes into the graph representation which fulfills the first requirement of **uniform smart device representations**.

Structured connections between smart devices are the drawing of edges between inputs and outputs. To know which smart device is connected, each edge can contain ids of the sending smart device and the receiving smart device. In addition, message types show the user which messages are implied by a direct connection. Connected smart devices are arranged in a graph representation

Each orchestrated graph is stored in a database which is used to deploy the graph. Additional, graphs can be edited through the web user-interface and can be added in an existing graph. This allows the user to **reuse existing graphs**, add and apply existing smart device orchestrations stored in the database.

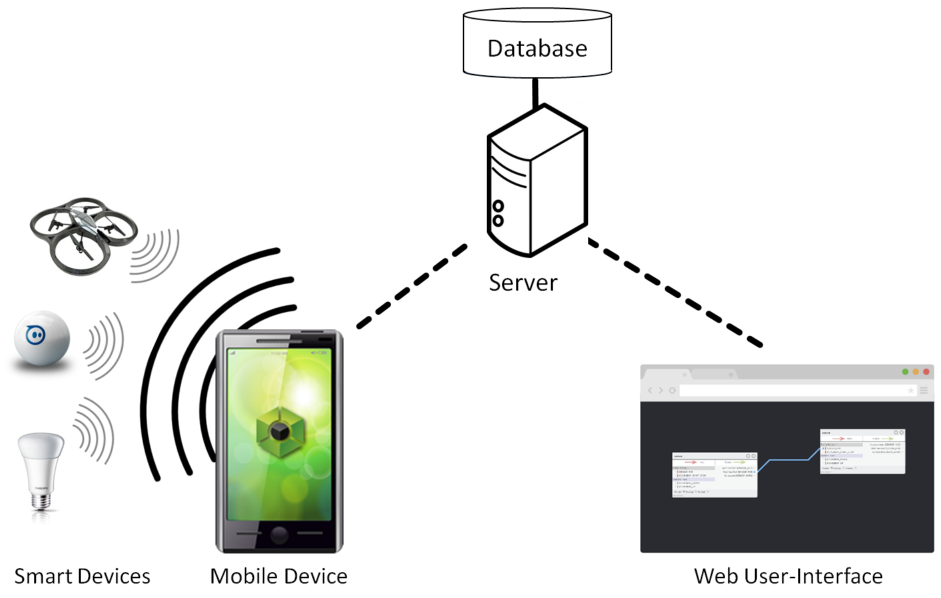


Figure 4.1: System Concept of web user-interface to orchestrated smart devices in a personal smart space.

In figure 4.1 the complete system concept to connect smart devices is displayed. A visual programming based user-interface allows to visually connect smart devices. The database allows to store the graph of connected smart device representations. All smart devices must be connected to the mobile device in order to be usable. The mobile device is using the graph structure to define which smart devices is connected to another. All messages between smart devices are mediated through the mobile device, as defined through the initialized graph representation.

In addition to the web user-interface, an administrative web page should enable to change database entries such as device and translator descriptions manually. This should avoid that the database needs to be directly accessed for small changes by administrative users.

4.1 Visual Web User-Interface

This section focus on the user-interface design, which should be used by users without programming experience. Limited amounts of possibilities are important, to keep the interface simple and user-friendly. The graph representation of visual programming interface should provide simplicity.

In a project called Meemoo [O+12], Oliphant invented a visual programming environment, to provide the possibility for end-users to orchestrate their own programs. While Meemoo focus on orchestrations of functionalities such as graphical transformation. The front-end visualization can be used for smart devices as well. Therefore, the Meemoo dataflow visualization was used as framework for the visualization of this prototype. It contains the visualization and basic functionalities to orchestrate nodes and edges. The orchestrated graph visualization can be extracted into a JSON-snippet, which contains information about inserted nodes and their cross-connections. The complete Meemoo project is licensed under the MIT and APGL licenses.

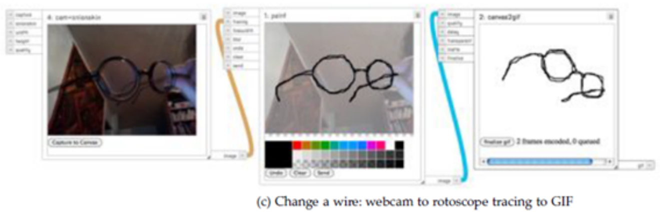


Figure 4.2: Meemoo front-end visual interface, figure from [O+12].

The prototype of this thesis was invented based on the latest version of the Meemoo visual programming tool which contained a reworked user-interface. Based on that, the complete web page user-interface of the prototype can be seen in figure 4.3.

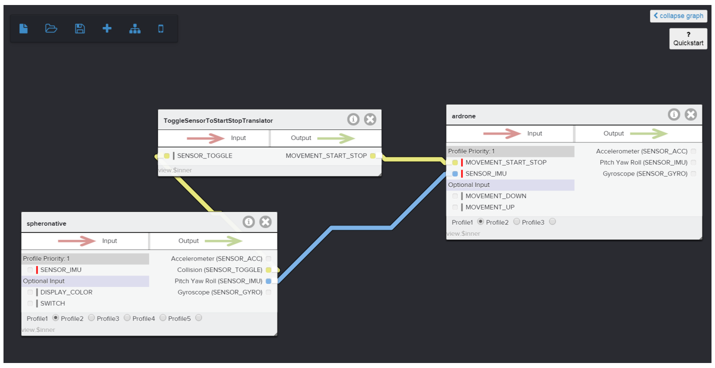


Figure 4.3: Final web user-interface to connect smart devices.

To allow easy actions, we integrated a main menu containing 6 buttons, to execute basic functionalities as shown in figure 4.4. A new graph can be created, previous graphs can be loaded or the current graph saved. *Add device* adds a PCDs node, while *add graph* inserts a complete graph into the existing graph which is called subgraph. *Remote connect* is connecting the web interface with the Dynamix instance, running on a mobile device.

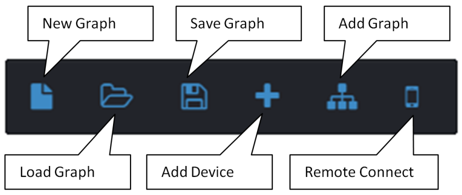


Figure 4.4: Main menu of web user-interface, contains six references to basic functions.

Plug-in Control Description Node

After adding a device, a device node (shown in figure 4.5) is added to the user-interface which contains most information, stored in PCD, introduced in chapter 3.3. These are name (1), outputs (2), optional inputs (3), a mandatory input profile (4). Only one mandatory input profile can be used at once but this can be changed through radio-buttons at the bottom of the node (5). Nodes can be removed by pressing the delete key or the displayed *X* (7).

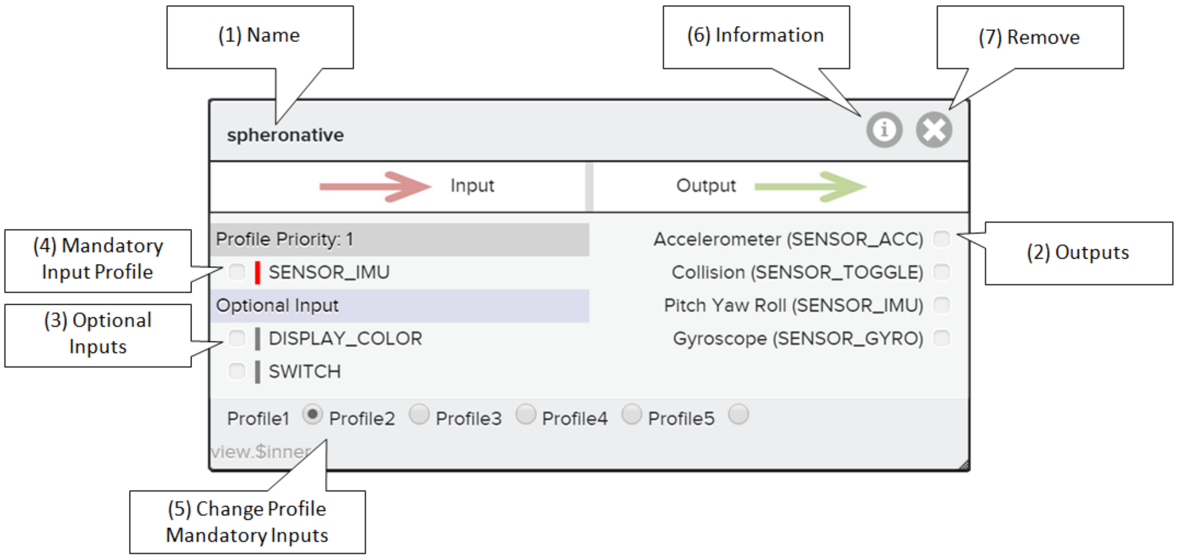
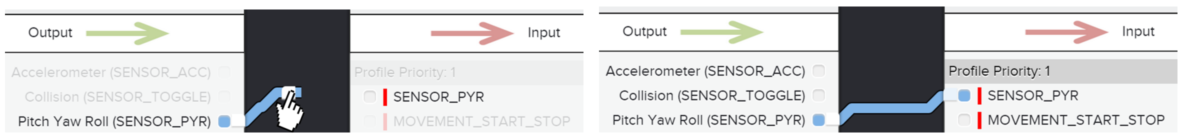


Figure 4.5: This node is generated through a PCD containing inputs and outputs.

Connections - Wire & Translator

It is highly important to provide a functionality to connect device nodes. The connection of nodes works through a drag and drop interaction. The user can simply drag aside the requested output and drag the appearing edge to the input he wants to connect. This works the other way round as well by dragging the wire from an input to an output.

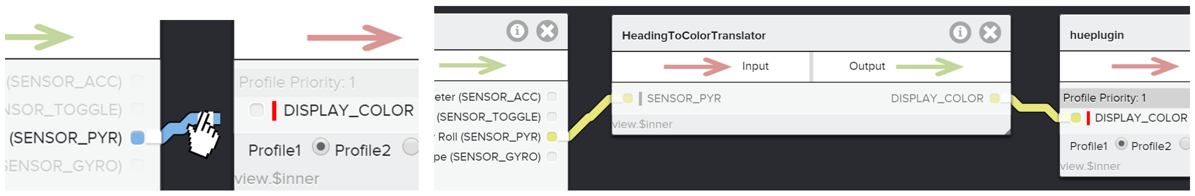
Theoretically, the user could connect any input with any output. Due to different command types this is not working in every case. Only equal input and output command types match, while unequal command types can be connected by adding a translator. To help the user, connections which are not possible are shadowed while dragging the wire. Therefore, the user gets visual hints of possible connections, as shown in figure 4.6.


Output
Input
Accelerometer (SENSOR_ACC)
Collision (SENSOR_TOGGLE)
Pitch Yaw Roll (SENSOR_PYR)
Profile Priority: 1
SENSOR_PYR
MOVEMENT_START_STOP

(a) (b)

Figure 4.6: (a) The user is dragging the edge to connect two device nodes. All not possible inputs are shadowed. (b) Connected output and input, all other inputs are active again.

An unequal type of input or output is not shadowed in case a translator is able to mediate the different command types. In figure 4.7a, two unequal input and output command types are shown. After the user drops the edge on top of the input, the translator representation is added automatically. In case more than one translator is available to mediate the command types, the user is asked which translator should be used. In figure 4.8, a translator is shown which has one input and several outputs. This is required for translators which split their input signal into several outputs for example an orientation signal which is one command and contains 3-dimensional information and can be spit into x-, y- and z-axis, which are three separate command types and would require three outputs.



(a) (b)

Figure 4.7: (a) The user is dragging the wire to connect two unequal command types. (b) Connected translator to mediate between command types.

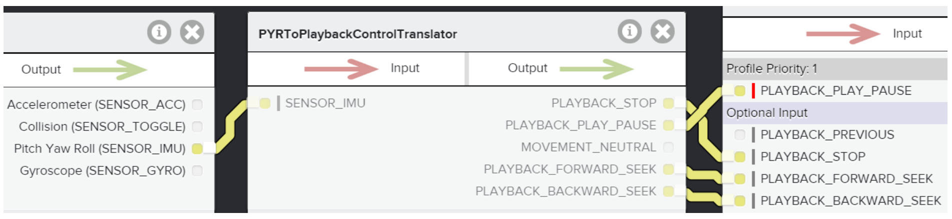


Figure 4.8: Multi-output translator.

A complete orchestration of two devices is presented in figure 4.9. The sphero (spheronative) robotic ball is controlling the Ar Drone Parrot helicopter which requires orientation commands (*SENSOR_IMU*) and the start/stop signal (*MOVEMENT_START_STOP*). To fulfill the complete mandatory input profile of ardrone, both command types are required. Spheronative is providing *SENSOR_IMU* directly, while *SENSOR_TOGGLE* is translated to *MOVEMENT_START_STOP*.

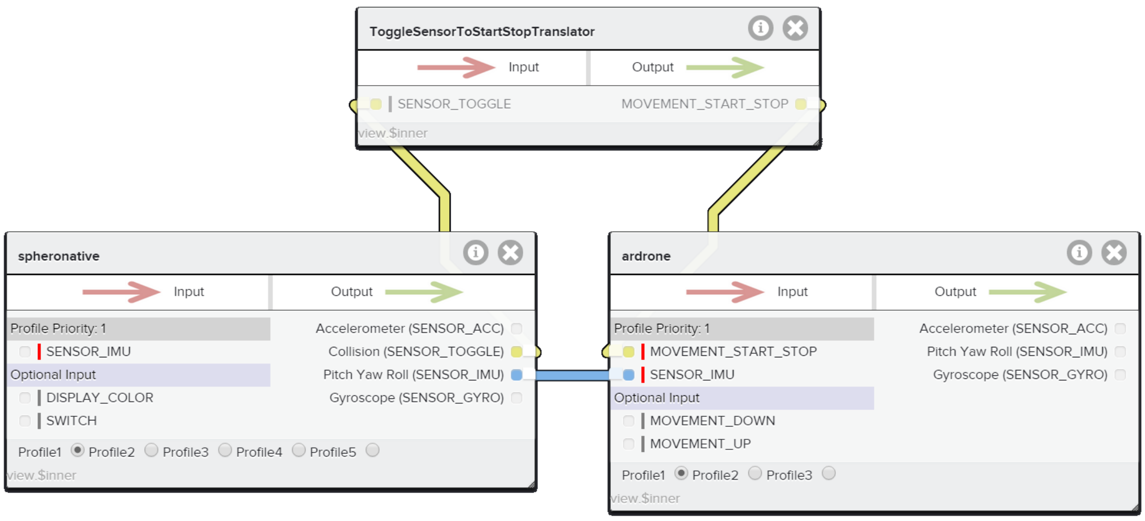
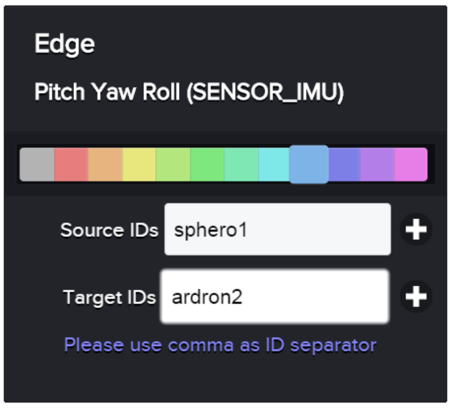


Figure 4.9: Orchestration of Sphero and Ar Drone connected through direct connection and translator.

Each device node is a representation of the device type and is not related to a unique smart device. To address a single smart device, this must be notated at the connecting edge between two device nodes. By clicking on an edge, a menu (shown in 4.10) appears to specify the source and target smart devices. A blank Id fields intent broadcast messages from all/to all devices of the specific device type while they can be filled with unique Ids. The Ids are given by the individual smart device manufacturer and can often be changed through their native applications.



Edge

Pitch Yaw Roll (SENSOR_IMU)

Source IDs: sphero1

Target IDs: ardron2

Please use comma as ID separator

Figure 4.10: In the shown edge menu, the exact input and output device can be notated. Blank fields intend broadcasting.

The *plus* aside the text field can be used, after a Dynamix instance on a mobile device is connected to this web application. It request Dynamix to send all device ids of a specific device type. In case smart devices are found, they appear in a pop-up and can be selected without typing them manually into the text field. In addition, the colors within this menu let the user change the color of the edge between device nodes.

Node-Menu and Auto Matching

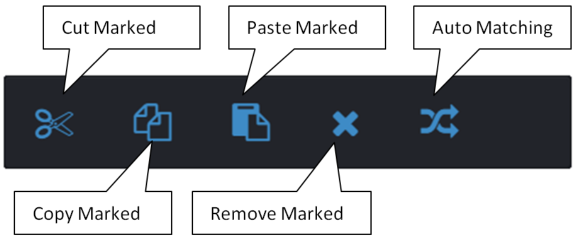


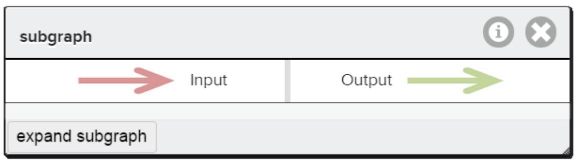
Figure 4.11: Menu to edit marked graph elements.

A specific menu appears after marking nodes or wires. Common functionalities are provided to cut, to copy or to paste the marked part of the graph. In addition, the market part can be removed which is also possible by pressing the key *del* on the keyboard. These are common commands which are known from text-editors for example. An additional feature is called auto matching. This allows to mark two or more device nodes and wire them automatically. The user has to define the receiver, while the other smart devices are arranged as message sending controllers.

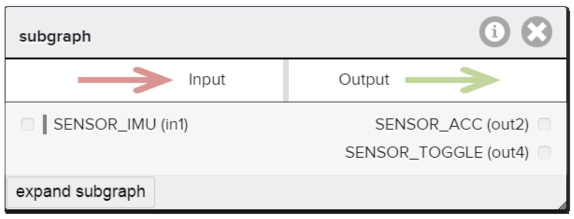
4.1.1 Graph and Subgraph

The whole set of connected devices is called graph. The graph can be stored into the database, which allows to reload or reuse the graph. The reuse of a graph within an existing graph is defined as subgraph and can be added as a device node through the basic menu.

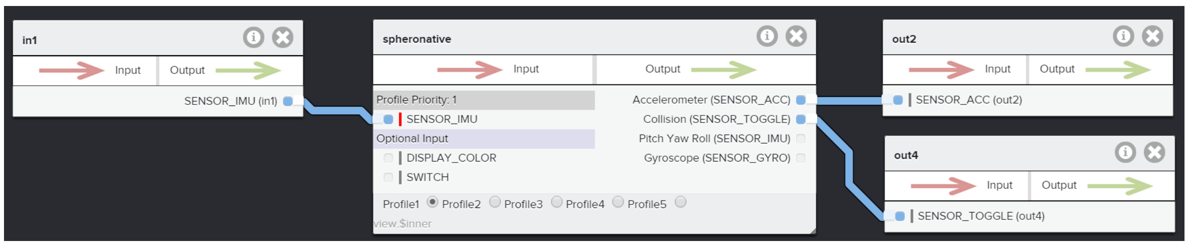
While starting the web application, it is possible to select a prior orchestrated graph by using a drop down menu or by adding the graph name as an URL parameter. Graphs can have various layers of subgraphs, which can be added to the current graph and expanded. By expanding, the subgraph is the newly visible graph. Special input and output nodes can be added to send messages between the higher graph and the subgraph (figure 4.12c). This example shows one input from the higher graph and two outputs back to the higher graph. After collapsing the subgraph, those inputs and outpus are also accessible at the node of the subgraph in the higher graph (figure 4.12a and 4.12b). The command types of the subgraph inputs and outputs are generally neutral until they are linked to a device node which specifies them as long as they are connected.



(a)



(b)



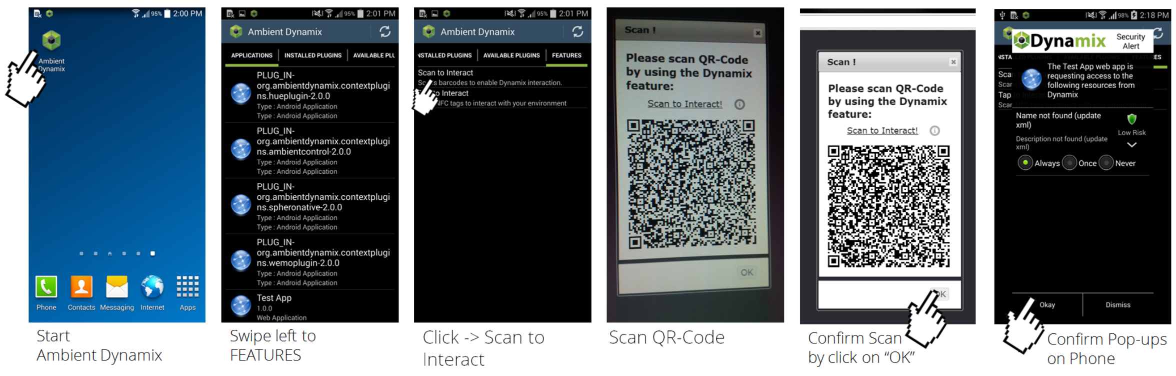
(c)

Figure 4.12: (a) Node of a subgraph without external inputs or outputs. (b) Node of a subgraph with external input and outputs. (c) Expanded subgraph with input and outputs.

Subgraphs can also work without external connections. Then they provide only internal connections of devices while the subgraph node does not have any inputs or outputs as visible in figure 4.12a. Inputs and outputs can be added any time and are visualized at the subgraph node dynamically.

4.1.2 Deploy on phone

To use the graph orchestration on real smart devices, the graph must be deployed to Dynamix and Ambient Control running on the mobile devices. Dynamix must also install a feature called scan-to-interact. Features are extending Dynamix by functionalities, such as scan-to-interact, which is a QR-code based authentication process. When the user clicks on *connect dynamix* in the base menu, the process works as follows. First the graph must be stored in the database, second a token is created which is displayed in a pop-up to get scanned by the scan-to-interact feature. This authenticates the mobile device at the server and receives the graph representation from the database. The Dynamix instance is paired to the web browser, which allows to reload a new graph automatically. The instructions to use scan-to-interact are shown in figure 4.13.


Scan !

Please scan QR-Code by using the Dynamix feature:

Scan to Interact!

The Test App web app is requesting access to the following resources from Dynamix

Name not found (update xml)

Description not found (update xml)

Always Once Never

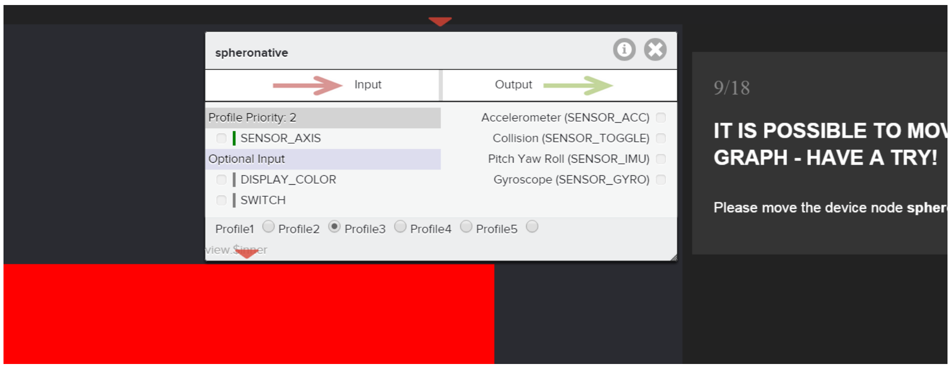
Figure 4.13: Quick-Start to use the feature scan-to-interact to connect Dynamix and the web user-interface.

4.1.3 User-Interface Tutorial

The first time a user visits a new user-interface, it is important to be familiar with the main functions quickly. This motivates the user to try out the system and even discover more functionalities.

To introduce the first possible actions, an interactive tutorial is provided which is based on sideshow[1], including some minor adaptions. This tutorial generates a complete control graph to connect the Myo[2], which is a gesture sensing smart device and a media device like the Apple TV[3]. The user is instructed to add device nodes, to connect them and to save the graph. During

[1]https://github.com/fortesinformatica/Sideshow
[2]https://www.thalmic.com/en/myo/
[3]https://www.apple.com/appletv/



IT IS POSSIBLE TO MOV
GRAPH - HAVE A TRY!

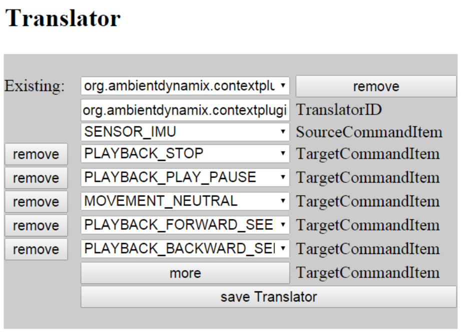
Please move the device node spher

Figure 4.14: Interactive tutorial instruction to drag the node to another position on the screen.

this process inputs and outputs, as well as translators are introduced. Figure 4.14 shows the interactive instructions to drag the device node to a different position.

4.1.4 Database Maintenance

In addition to the web based user-front-end, a administrative web based back-end is integrated. This is called *DatabaseMaintenance* and enables to change descriptions of device plug-ins and translators, stored in the database by using a visual web user-interface. Command items can only be added at a specific point, not at any single point when a new input or output is added to avoid typos. These command types can then be chosen for inputs and outputs of PCDs and translator descriptions. It is even possible to show JSON snippets of graph representations and XML scenarios. it is possible to directly show a graph in the front-end user-interface. Figure 4.15 shows the interface to configure translators. It is used to add, change and remove database entries through an administrator. Every time a new plug-in description is released, it must additional be integrated into this database



Translator

Existing:	org.ambientdynamix.contextplu	remove
	org.ambientdynamix.contextplugi	TranslatorID
	SENSOR_IMU	SourceCommandItem
remove	PLAYBACK_STOP	TargetCommandItem
remove	PLAYBACK_PLAY_PAUSE	TargetCommandItem
remove	MOVEMENT_NEUTRAL	TargetCommandItem
remove	PLAYBACK_FORWARD_SEE	TargetCommandItem
remove	PLAYBACK_BACKWARD_SEI	TargetCommandItem
	more	TargetCommandItem
	save Translator	

Figure 4.15: Back-end user-interface to add and edit translator descriptions.

5 Implementation

The visual user-interface is implemented to evaluate the usability through studying real users and to extend the Ambient Control framework to inspire future projects. This prototype is a web user-interface, connected to a database as suggested in 4.1. The next sections introduce the prototype implementation.

5.1 Changing Environment

The implemented software of this project must be able to be extended and reused by future projects. Therefore, constraints of reusability as well as flexibility are employed and implemented. This project extends Ambient Control by adding a visual user-interface.

Required technical details to orchestrate smart devices are already provided by Ambient Control and Ambient Dynamix. Smart devices are developing and are added, updated or extended quite often, which makes it difficult to develop a static system. Right before this thesis started, Ambient Dynamix upgraded to 2.0 which caused major architectural changes of Ambient Dynamix. During the implementation of this prototype, new ideas, possibilities and features of Ambient Dynamix, Ambient Control and this user-interface motivated a continuous and creative growth of of all three projects.

5.2 System Implementation

The implementation itself is separated in different projects, as shown in figure 5.1. The web user-interface is implemented in the project, called Web Graph. Database requests via RESTful calls are reaching the REST-endpoint in ControlProfileServer which is accessing the database. PCDs are used on the web user-interface and within Ambient Control. These PCDs are stored within the database and can be extracted as XML snippets. Ambient Control requires java objects instead of XML snippets. This requires to transform the XML representation of the PCD to a java object which is called marshalling.

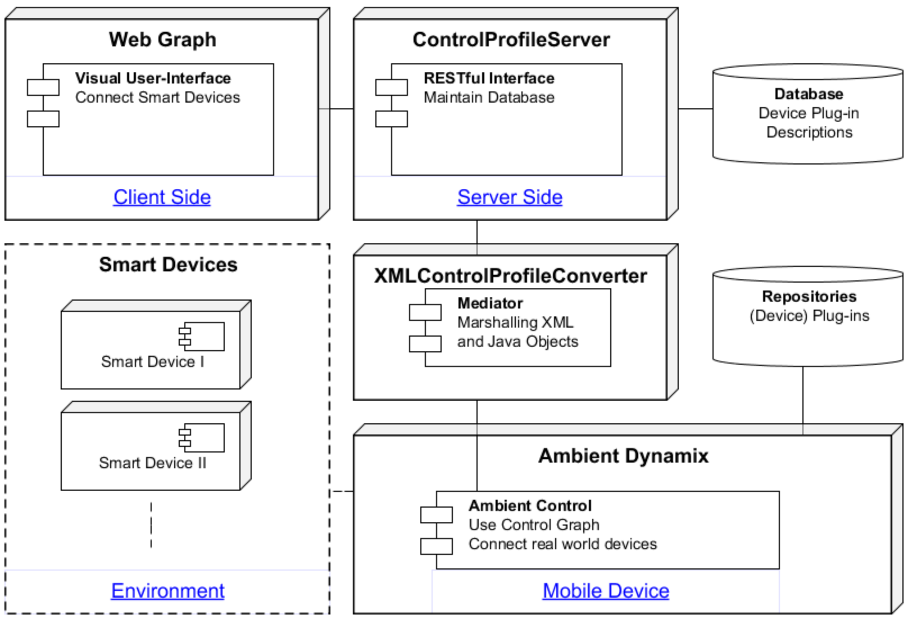


Figure 5.1: Separated projects to enable database support for device (descriptions) and graphical user-interface.

5.2.1 Database

The database must be able to store PCDs and the orchestrated control graphs. PCDs are simple to represent by joined SQL-tables containing atomic entries while graphs are more difficult to represent within a regular SQL database structure. Therefore, it is possible to use so-called NoSQL (Not only SQL) databases, which are invented to store hierarchical structures and focus on graph calculations. After evaluating NoSQL and SQL the decision was made to use SQL for PCDs as well as for generated graphs for the following reasons.

SQL is widely spread, simple to use and can be scaled in various sizes and the large amount of users using SQL, fixed security leaks and made it a very stable and secure database application. It is to notice, that aside of the positive features, SQL is not performing well on graph calculations compared to NoSQL. Since the used graphs are only stored in the database without any calculations, NoSQL is not required and graph snippets in XML and JSON are stored in plain string format. On the local test environment SQL Workbench 6.1 CE is used. The public database is running phpMyAdmin version 3.3.7deb7. Based on the format of PCD XML snippets, I orchestrated a suiting database architecture which can be seen in figure 5.2.

The table named *plugincontroldescription* contains all context information required to set up the plug-in environment like the platform version. The *ARTIFACT_ID* is a unique plug-in id, such as

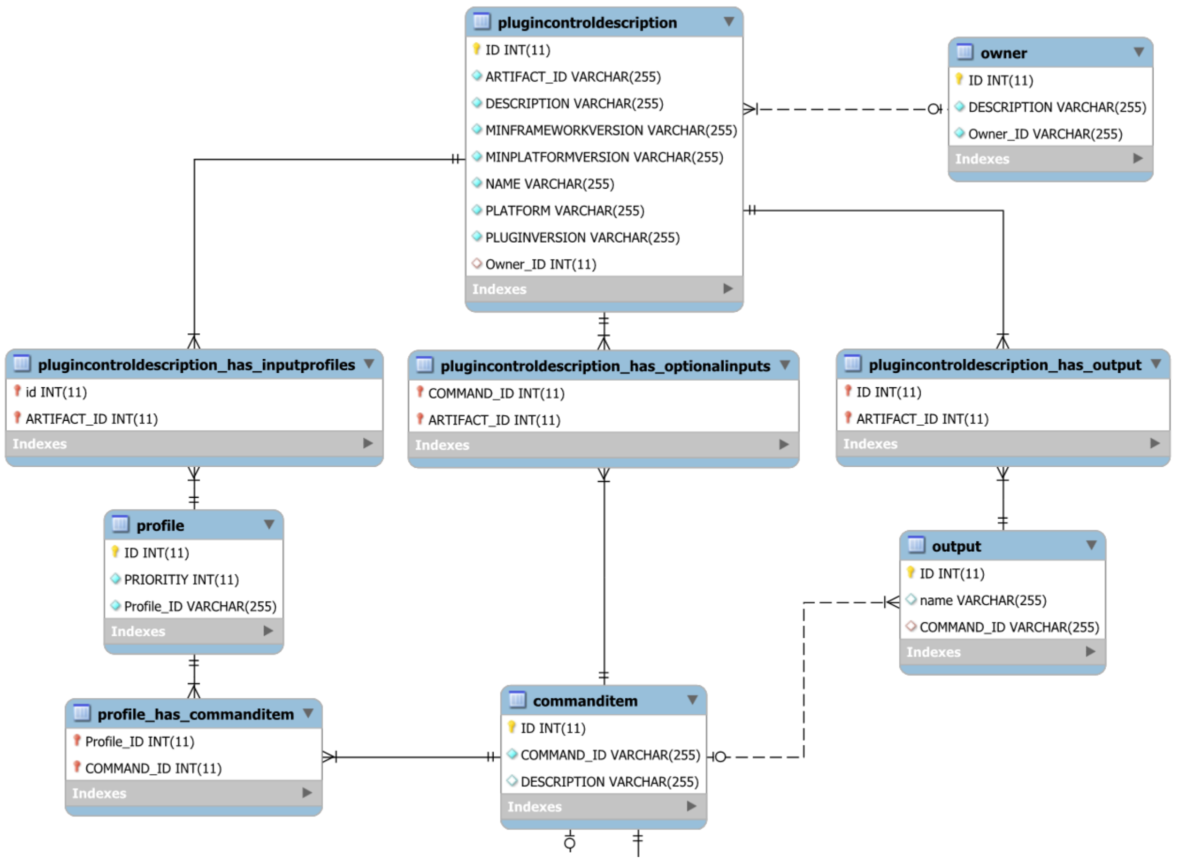


Figure 5.2: Each device plug-in description is stored within this database design.

org.ambientdynamix.contextplugins.spheronative. All PCDs contain input or output commands in order to get connected. The table called *commanditem* is containing these command types and is referenced by inputs as well as outputs. In order to connect two plug-in descriptions without a translator, the *COMMAND_ID* (which is also called command type) must be equal. Therefore, command items are very important to connect PCDs.

While output and input are referencing *commanditem* directly, each plug-in description can contain various input profiles. These profiles describe the required input set which contain a set of command items. Each profile contains a priority to rank possible input sets.

Like device plug-in descriptions, descriptions of translators are stored in the database as well see figure 5.3. Each translator description has its unique id equivalent to an *Artifact_ID*, called *Translator_ID* e.g. *org.ambientdynamix.contextplugins.ambientcontrol.DisplayToSwitchTranslator*. The input and outputs of a translator are referencing the same *commanditem* table as inputs and outputs of plug-in descriptions. The translator has only one input and an arbitrary number of outputs, which requires an additional table to match output commands.

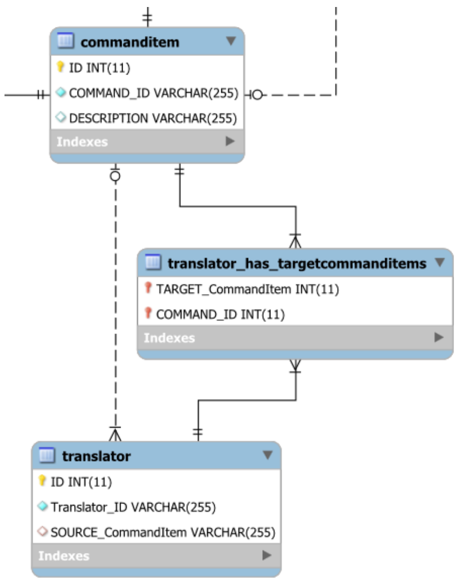


Figure 5.3: Database design for translator descriptions. A translator description contains the same input and output commanditem as PCDs.

The table to store control graphs only contains the unique graph-id, the text-type column for JSON snippets and a text-type column for XML snippets. The JSON snippet is used to initialize the web user-interface while the XML snippet is used to marshal the required object for Ambient Control.

5.2.2 Server

Java servlets are used on the server side which support RESTful requests. All database access is done via this interface, which includes the checking of the XML input of PCDs. To encapsulate the server structure, Data Access Objects (DAO) are used. The database access is realized through the Java Persistence API. Incoming HTTP-requests address the domain logic (RESTful Interface) and are processed through DAOs which generate the HTTP-response. Implemented functionalities are to add a new entry, clear an entry and change an entry, which immediately manipulates the database. As local server, Apache Tomcat version 6.0.41 is used, while version 6.0.35 is running on the public server.

The detailed REST-calls can be found in appendix, while a short overview is given in table 5.1. The Java Persistence API is structuring the classes quite equal to the database model introduced earlier in figure 5.2.

HTTP-Request	Description	Example URL
GET	All description ids	/PluginControlDescprition/ids
GET/PUT/ POST/DELETE	Change entry of description	/PluginControlDescprition/{id}?format={xml|json}
GET	Query descriptions	/PluginControlDescprition?Input={SENSOR_PYR,} &MOVEMENT_UP}&Output={MOVEMENT_DOWN} &Constraint={ALL|ONE}
GET	Query translators	/Translator?Input={SENSOR_PYR} &Output={MOVEMENT_DOWN}

Table 5.1: RESTful Interface URLs

For GET-requests, the parameter $format$ is defining the return type which can be XML or JSON formatted. JSON is required to setup the user-interface while XML is used for Ambient Control. The XML formatted plugin-description or control profile is marshalled to XMLControlProfile by using JIBX [1]. This java object is then used by Ambient Control.

5.2.3 Web user-interface

The front-end-code is completely written in HTML, JavaScript and CSS. The web user-interface was developed and tested by using Google Chrome version 40.0.2214.94.

5.2.4 Connect to Phone

On the mobile side, Ambient Dynamix 2.1.10 is installed on the android mobile phone and the Ambient Control plug-in library within Dynamix. To connect the web page and the mobile device, the feature scan-to-interact is installed which enables to scan QR-codes in order to connect with a remote application. The used mobile device is a Samsung S4 running Android 4.4.

[1] http://jibx.org/

6 Evaluation

6.1 User Study

To evaluate the usability of the implemented prototype in quantitative and qualitative aspects, this user study was accomplished. It is a composition of interview questions and practical tasks. A close contact with each participant was necessary to get their honest feedback and to see where issues or confusion about the visual user-interface are occurring. The details about the methodology, participants, setup, results, and their discussion are provided in the following sections.

6.1.1 Introduction

This thesis proposes an approach to connect smart devices visually, expected to not require any programming experience. This user study should use the prototype and get solid feedback about the user experience and their creativity to connect smart devices. In order to be creative the user must be excited by connecting smart devices. The expectations are that the visual user-interface motivates and excites users without programming experience to connect smart devices. Existing work such as stationary gateways have not covered this specific use-case to connect surrounding smart devices by using the mobile device as a mediator in between. The use of a simple graph configuration, hopefully enables many users to connect their smart devices. Aside of quantitative results, user emotions, behaviors and comments on this prototype are highly interesting. Therefore, each participant was listened and watched very carefully during this user study.

The aim of this study is twofold. Firstly, it should provide honest user feedback and usability results based on the prototype. Secondly, it should give results about the user creativity to connect various smart devices which are available in their surroundings.

The result of this thesis should prove the theory, that the graphical interface of the prototype is usable by users without programming experience. In addition, the user study asks the user which features could be added or improved to increase the usability. In the first step, smart devices were connected on the visual user-interface. Thereafter, this orchestration was deployed on the phone. This user experience of seeing the complete prototype working engaged the user to participate very interested and engaged during the interview parts, thinking about future use-cases.

6.1.2 Method

To gather direct results of users' opinion and data, the method of a user study was chosen. Through this method it was possible to understand usability pros and cons, as well as to receive detailed user feedback. After completing the user study with each participant, all data were analyzed and evaluated.

6.1.3 Participants

The user study was completed by 10 participants which were separated in 8 participants owning a mobile phone and 2 participants not owning a mobile phone. The ratio of male and female separated in 6 male to 4 female participants. Since this study introduced a new field of interaction to each of them, they got asked about their interest in trying new technologic trends. The majority tended to be interested. The user study was held in German, it was processed in Germany and every participant was a native German speaker. Only the online-tutorial instructions and the web user-interface were English. Therefore, the German answers were translated into English, which I have done to the best of my abilities. The majority of all participants were between 16 and 23 years old while only two participants have been 50 years old. All results have been generated trough voluntarily participation of the participants without any payments.

6.1.4 Setup

The environment to process this user study required a computer, a mobile phone and two smart devices called Sphero and Lifx. The computer was running a server hosting the web page as well as the database, which contained graph, translator, smart device plug-in descriptions. Another table within the database was used by a server script to authenticate mobile phone and web user-interface to interact with each other.

The mobile phone was in the same network as the computer and ran Dynamix [1]. Within Dynamix the feature scan-to-interact and Ambient Control were installed. The scan-to-interact feature enabled the user to scan the QR-code on the web user-interface and deploy the generated graph to the mobile phone by the use of Ambient Control. The mobile phone itself was a Samsung Galaxy S2 running Android 4.4. To connect with smart devices, WI-FI and Bluetooth had to be enabled.

At the end of this user study, each participant should have connected two smart devices within the web user-interface to deploy this graph on a the mobile phone. The used smart

[1]Dynamix Version: 2.1.10

devices for demonstration purpose were a Sphero and a Lifix, which are introduced in the next paragraphs.

At first the Lifx[2] is introduced, which is a light bulb and can be connected to a local network via WI-FI. This light bulb has impressively bright LEDs, which can generate a wide variation of 16 million colors by using red, green and blue LEDs. All regular light bulbs have standardized sockets which is also provided by Lifix and can therefore replace any retro light bulb while consuming less energy. The Lifx has the same features as an equal LED-Light from Philips which is called Hue[3]. Since the plugin within Dynamix is called Hueplugin, in the future writing, Hue is used instead of Lifx (both light bulbs can be seen as devices providing equal functionalities).

The second used smart device was the Sphero, which is a robotic ball, having a huge amount of sensors and actuators. Sphero can be controlled through a native app, which employs the mobile phone display as a remote control. Many actuators of the Sphero can be controlled. For example the LED-color, the speed, and the orientation. Additional, it is possible to sense collisions and the orientation of Sphero.

6.1.5 Procedure

To process the same procedure every time, a detailed script was defined and used for each participant, it is added in German language within the appendix. To get an insight of the user thoughts and behaviors, the study was divided into three separate parts. The first part has been an introduction of the study and an interview regarding currently used smart devices and ideas to control new smart devices. This was followed by the second part, letting the participant process and evaluate several tasks, which started with a tutorial. Afterwards, the participant used the gained knowledge to fulfill the task of connecting devices on his own, while the last task was to deploy the graph configuration on a phone and see the result on real smart devices. Finally, each user answered System Usability Scale (SUS) questions and got interviewed about future developments he could imagine within IoT.

The SUS test and the time a user took to complete a task are quantitative results, while all answers and thoughts given during the interview were noted immediately. Each user study took between 30 and 45 minutes, depending on the participation of each participant during the interview sessions and the time each participant took to do the tutorial more or less carefully.

[2] http://lifx.co/
[3] http://www.usa.philips.com/e/hue/hue.html

6.1.6 Results Part I: Guided Interview - Introduction

To introduce this field of research to participants, an introduction about smart devices and IoT was given. After the participant got a rough idea about what is meant by smart devices, he was asked for smart devices, which he already connects to his mobile phone. The second question was to name devices in the daily environment which the participant would like to control by using his mobile phone. These questions were designed to get an idea of what each participant could imagine or what he would like to use, before being introduced to the smart devices used during the practical tasks, which otherwise might have manipulated the answers.

All participants did not have detailed knowledge about the IoT. Therefore it was very interesting which devices they are already using and which they would like to use. The results can be found in figure 6.1. On the left hand side of this intersection diagram, the smart devices which are already used can be found. These are connections between mobile phones and between mobile phone and computer to share images, music and other data. Second, participants are connecting their mobile phones to cameras like the GoPro or other toys like for example a helicopter, controlled through native mobile applications. Third, the automotive section was very common answer where participants are connecting their smart phones already with hands-free speakers.

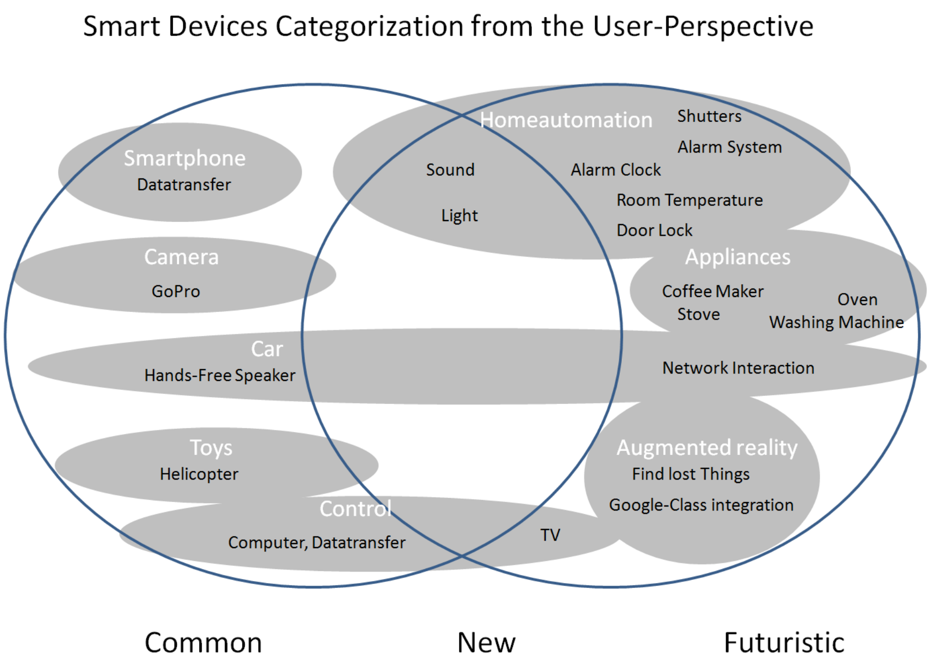


Figure 6.1: Participants replied those smart devices and use-cases, separated in what use-cases are common, which are new and which are futuristic, from the perspective of the participant, which is not equal to the state of available products.

Quite often, smart phones are wirelessly connected to speakers in order to play music. A second use case is to connect the smart phone with the locally installed lights. They can be controlled individually or grouped. Some participants are already using smart lighting, while others wanted to use them in the future.

Generally, all participants plan to control their homes by using their mobile phone in the future. A variety of sensors and actuators, summarized by home automation, including shutters, alarm system, door lock and room temperature were named by participants. Furthermore, especially females named household appliances which they would like to control by their mobile phone. They would appreciate to control stove, oven, washing machine and coffee maker wirelessly. Male participants focused on the car and asked for network interaction within the car and in between cars. Finally, to find lost things was a future request, as well as an usable integration of Google-Class.

6.1.7 Results Part II: Practical Part - Tasks

Firstly, the participant completed a tutorial, which guided a connection to use a gesture sensing wrist band to control a media device. Such media device could be an Apple TV while the used Ambient Control plug-in to control an Apple TV is called ambientmedia. For gesture recognition the wrist band called Myo[4] was used, which can sense gestures like arm up, arm down, swipe left, swipe right. A typical use-case would be to stop the movie by a gesture and continue the movie by another gesture, which are both detected by the Myo wrist band.

Every participant got introduced to Myo by showing them the Myo add[5], which contains a wide range of use-cases. The graph configuration to control the ambientmedia smart device through Myo was successfully completed by each participant. It was important, that the tutorial was processed by an automatic tutorial, which gave every participant equal conditions while the time to complete the tutorial was measured secretly.

The next task for each participant was to configure a new graph configuration and let the Sphero control the color, displayed by the Hue. While completing this task, the time was measured secretly, again. In order to get valuable usability results, all participants were asked standardized questions trough the SUS multiple choice test. Therefore all results are comparable and give more insight about usability, in addition to notes which were made during the participant processed the given task.

Until this point, each participant only used the web user-interface but has not interacted with the available smart devices Hue and Sphero directly. Therefore the next task was to deploy the generated graph configuration on the mobile phone, where the user got help by quick start instructions.The user had to scan a QR-code in order to connect web page and phone. I have

[4]https://www.thalmic.com/en/myo/
[5]https://www.youtube.com/watch?v=oWu9TFJjHaM

not implemented the QR-authentication, nor the handling on the mobile device. But because this feature is part of the deployment process, I wanted to get feedback on that as well. This helps to improve future development on usability and extensions of that feature and additional shows each participant the complete cycle of creating a graph configuration and to use this configuration on real smart devices afterwards.

To complete the tutorial, was the first interaction of each participant with the web user-interface. The average time each participant took to do the tutorial was 277.2 seconds. In comparison, each participant took in average 121.5 seconds to proceed the task to connect the Sphero and the Hue in their own graph configuration.

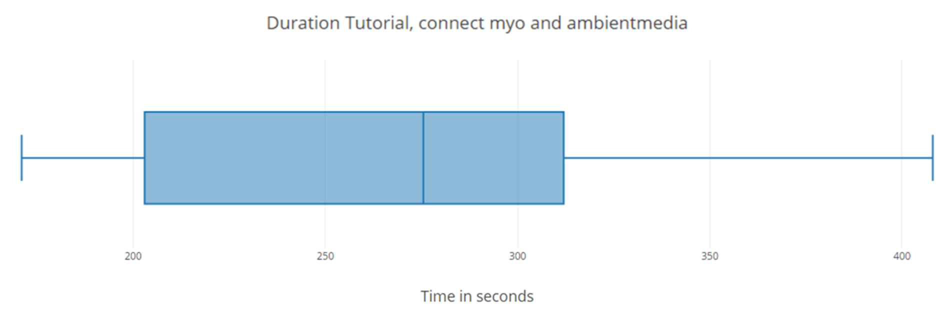


Figure 6.2: The boxplot shows the timings which participants needed to complete the tutorial.

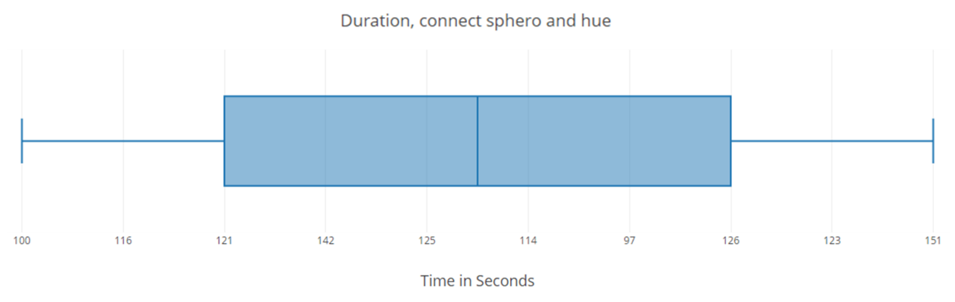


Figure 6.3: The boxplot shows the timings which participants needed to complete the task to connect Sphero and Hue.

To get valuable usability results the System Usability Scale contains 10 standardized questions:

Col 1. I think that I would like to use this system frequently.

Col 2. I found the system unnecessarily complex.

Col 3. I thought the system was easy to use.

Col 4. I think that I would need the support of a technical person to be able to use this system.

Col 5. I found the various functions in this system were well integrated.

Col 6. I thought there was too much inconsistency in this system.

Col 7. I would imagine that most people would learn to use this system very quickly.

Col 8. I found the system very cumbersome to use.

Col 9. I felt very confident using the system.

Col 10. I needed to learn a lot of things before I could get going with this system.

A box plot of each answer set was calculated, which can be seen in figure 6.4 and 6.5.

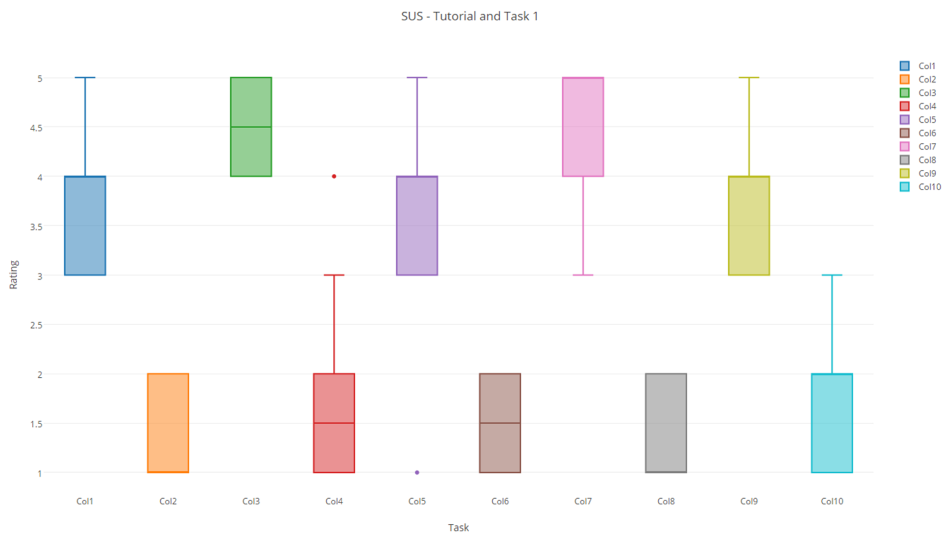


Figure 6.4: SUS - test results after tutorial and connecting two devices visually.

The results of each boxplot which is representing a question, are mostly very compact, which shows that most of the participants had the same experience during the tutorial and the followed task. After completing the tutorial and their fist own connection, the calculated result of the SUS is 81, which states a positive result regarding the usability of the requested tasks. In comparison the results of the task to deploy the configuration of Sperho and Hue on the phone gave a wider variety of answers which can be seen in figure6.5.

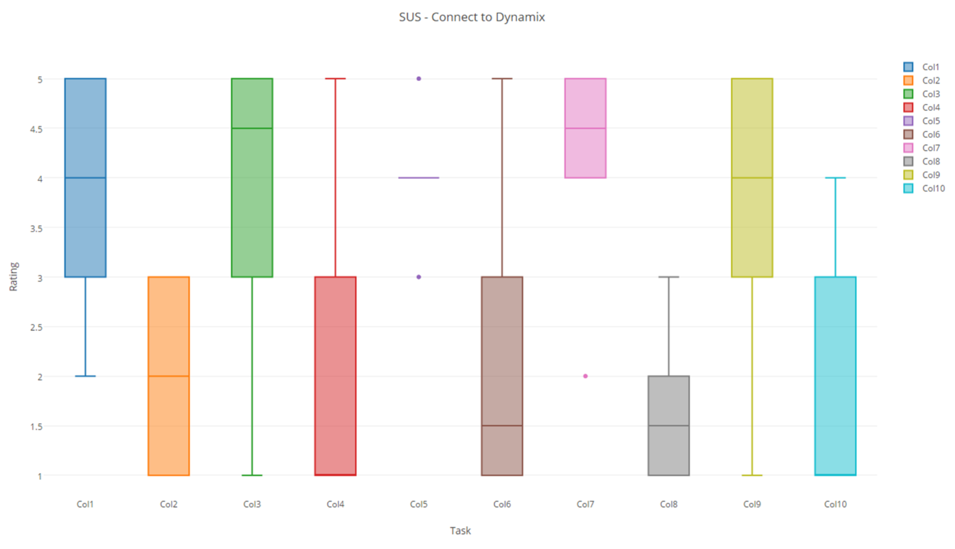


Figure 6.5: SUS - test results after connecting the web user-interface and Dynamix.

The SUS index of the task to connect mobile phone and web user-interface was 74.5, which is lower than the result of the first tasks and states a lower usability.

6.1.8 Results Part III: Guided Interview - Future Questions

After the user processed the third task, the user study was ended by an interview focusing on future questions of this project. According to the participants, they would like to access a database where configurations are stored, download them and use orchestrations of other users. To use the download feature most efficient, it is important to sort the database for the users needs. Another remark was, that users would favor uploads deployed by manufacturers. Most are not interested to deploy their own configuration online.

In discussion about which devices the user would like to connect in the future, home automation and controlling was the most received answer, which contains heating, lights and to lock doors, including the use of alarm systems. Right after home automation, participants asked to connect household appliances with each other. The washing machine and dryer or the fridge and an app which can be used during shopping and let the user check the missing items in the fridge while walking through the supermarket. The participants also thought about health wrist bands to control a healthy living, combined with speakers to use during sports or just monitoring activities and recommend to do new activities. Most participants wanted to have more possibilities to interact with their cars, which mean to save the settings of the car seat and temperature in a specific user profile. In addition the fuel status should be controllable per mobile phone to check if it is necessary to add several minutes while using the car next time, because a fuel station visit must be added. Additionally, the wish to control the auxiliary heating by mobile phone and be able to see if the windows are covered with ice to start heating in the morning before using the car.

Finally, hobbyists suggested to connect the phone and the soaring plane to visualize flow monitoring and more details about the surrounding. This should be provided by a huge database, which give the pilot as much information as required to make the best decision.

A special tablet user-interface, where it is possible to use finger gestures to connect device nodes was not interesting to most of the participants. One given answer stated that it is enough to have only a computer version of the web page, since the user only configures the device setup once in a while, where it is possible to use a computer. The overall opinion was to focus on the usability of the current version and to add more devices as well as device descriptions within the web user-interface.

Each participant was asked to point out things which he did not enjoy during the first interaction with the web user-interface and the deployment on the mobile phone. Most people complained about the user feedback which makes it hard to connect web user-interface and mobile phone to deploy and test the device connections. Some participants wished to have a more colorful user interface instead of the dark theme. Within Dynamix, participants requested a simple landing page, while others asked for a German user-interface.

The most liked features was the graphical interaction, which made it simple to understand the functionalities and was perceived as structured, ordered, simple and clear. The participants enjoyed the compact menu is containing known icons which allowed fast interactions with the new web user-interface.

The hands on tutorial also gained a lot of positive feedback and enabled a fast introduction to web user-interface. The users were very interested in the new possibilities to just connect various devices. This allows users without programming experience to use their own imagination and creativity to create new device orchestrations without having any knowledge about the background implementation. The statement of a participant was:

> "Drawing connections is fun."

In future versions the users would like to see an improvement in the usability of the connection to the phone. Most user wished to get more feedback during the connecting process and to receive a visible feedback on the web user-interface after the pairing is successful. According to participants, more explanations of plugins, images and names of current graphs while working in it improves the usability. The view window where parts of the graph are visible can be dragged, wherever the user wants to shift it. Sometimes it is hard to find the center position, where the device nodes are located. Therefore, a center button is required to bring the user back to the center of the graph. A very interesting idea was to let users vote for needed translators. Everyone could be sure that the most required translator is implemented first. The most futuristic approach was to use voice recognition to configure the device graph where a web graph wouldn't be necessary during the first round and could later be used to configure the voice recorded setup.

6.1.9 Interpretation

More smart devices are emerging our lives every day in various fields. After evaluating this user study, it seems like not many users are using the full set of available possibilities given by smart devices. Most people are using a mobile phone today, which nobody would have expected in 2007, when Steve Jobs released the first iPhone[6]. This leads to the clue that after a period of adaption the use of smart devices can be adapted quite broad and quite fast.

As the interviews during the user study showed, most people have a rough idea of smart devices, but often don't call them smart devices. Most of the participants are using smart devices connected to their phone more or less every day, which includes home automation and hands-free speakers in the car, as well as cameras and toys. The usability of some of them is really worse, but others perform well and let the user enjoy the provided benefit. The reason why this user study was processed, was to get valuable results of the usability of the visual interface to connect smart devices. Additionally, insights are given, about how users could imagine to connect smart devices in order to have benefits based on their orchestration.

Web Graph Usability

At first I would like to discuss about the usability results of the web user interface. The in-page tutorial was build to enable users without programming experience a soft start with this new web page. It was great to see users interact with the website guided through tutorial. Its user oriented and interactive design helped a lot. While each participant did the tutorial, different user behaviors were noticed. Some users have read every single line and checked anything in detail, while others just followed the signs on the screen, but did not read the description which introduced each step. This was the reason why the timings of doing the tutorial vary between 171 and 408 seconds. The fact that some user did not read the descriptions and were very fast shows, that the steps and hints of the tutorial are very intuitive. The time for 18 tutorial steps were just 408 seconds in maximum, which is only twice the time of the fastest participant. This shows that even the careful participants finished the task quite efficiently.

It is remarkable that every participant was able to do the task of connecting the Sphero and Hue without any help, except the prior done tutorial. During this task, it turned out that the timings are switched the other way round, since participants who finished the tutorial fast, did

[6] `https://www.apple.com/pr/library/2007/01/09Apple-Reinvents-the-Phone-with-iPhone.html`

not remember every single step and had to try how to progress at some points while solving the given task. Participants who had read the tutorial instructions carefully were more efficient and fast while completing this task. It is obvious, that users had to learn how to interact for the first time. Since a lot of common software user-interface elements were used, the interaction was easy to learn and apply on first tasks. Users felt confident by interacting with the web user-interface as confirmed trough the SUS evaluation.

The connection between Dynamix which is running on the mobile phone and the web user-interface was not very smooth to connect since the WI-FI connection was not very reliable. This asked for some reloads and retries. Those rough conditions were an additional feature to test the usability, because it gave an insight into feedback and error handling. A lot of users complained about that, because they could not find any details about what is causing the issue of not connecting web user-interface and mobile phone. This caused a lot of new ideas to improve further user feedback. The scan-to-interact feature can be used in a wide field of future applications, connecting the mobile phone and the web browser must therefore be improved. Because of those issues, it was not possible to measure reliable timings of the pure pairing process.

Aside of feedback issues, the participants really enjoyed to connect the web user-interface with the mobile phone and use its networkability to instantly connect smart devices. This might be a reason why the SUS score is not worse, even though they didn't had a great usability experience all the time.

We thought about developing a version where the user can drag connections by using his finger on a tablet. The results were not as expected, since most of the participants don't belief this improves the usability of the web application. Since the application usually won't be used every day, it is possible to use the computer for the (initial) setup and deploy the configuration on the mobile phone. Since everything is going to be mobile nowadays, the expectations were the other way round. This is a great chance to focus on the computer and continue to improve the web user-interface. User critics mostly remarked the low amount of feedback provided during the pairing process after scanning the QR-code. This can simply be fixed in a new version, where the user gets more feedback about errors and current states during the connecting process. Participants particularly asked for a German translation, since it might be easier to understand in the foreign language.

Overall, participants really enjoyed to connect off-the-shelf devices. This time, participants had to fulfill the requested tasks, but already a lot of them felt inspired and would like to continue connecting smart devices and create new use-cases based on their own creativity. This could generate unexpected results, which can be completely new fields of usages which nobody thought of, until somebody draws the connection and enabled a great new user experience. The focus during the implementation was to make the web user-interface as useful as possible, by avoid user inputs, which are not possible so far. This policy got a positive remark by users, which noticed that whatever they do within the web user-interface, they can't create a graph which is not plausible or incompatible.

Discussion to optimize the existing approach

The user study helped a lot to prioritize future developments. As declared, graph configurations can be large which would benefit a lot when previous work could be reused. Therefore we would like to support this kind of reuse by adding a database where configurations can be shared. The idea to set up a database and share graphs, was supported well, while potential users tent to be more eager to download configurations than to upload their own configurations. This could therefore be a typical model like an App-Store, where users just download what they need, but don't have to care about anything else. The idea of a store and community could also be very interesting to link developers and user interests. To let the community vote for translators and devices,could increase the functionality a lot and should therefore be added to the framework.

A benefit of this thesis could be to connect smart devices for handicapped people. Every day devices like the TV, web browsers as well as gadgets could be connected to input devices which enables them user-interactions, even if they can't use the default control because of their handicap. They could have the possibility to use another control, which is more addressing their capabilities.

My hypothesis was that regular users without programming background are able to configure the graph configuration of smart devices through the visual user interface. This hypothesis can be confirmed, since every new user was able to finish the tasks for the first time, even though not every participant is using a computer every day.

The participants feedback showed, that it engaged them to think about new connections and orchestrations. This proved my second hypotheses, that users are engaged and have ideas to connect existing smart devices to interact with each other. To support users, all devices must clearly specify, on which inputs and outputs which actions are performed. By using defined command types, it is simple for users to understand the possibilities of each device, after using the web user-interface for the first time. The prototype worked well and gave each participant the chance enjoy their own hands-on-experience followed by their honest feedback. Without deploying the graph on the mobile phone, the users were not as motivated to share their feedback and thoughts of future orchestrations, as they were. I have to thank every participant taking time to improve the usability of connecting smart devices.

7 Discussion, Conclusion and Future Work

7.1 Discussion

In this section, results about the user-interface are discussed. The evaluation of the user study shows the usability of the web user-interface to orchestrate smart devices. After a short tutorial, each participant was able to solve the given task, to connect two devices in the web user-interface. The participants were motivated to connect new smart devices and define their own connections.

During the user study, the process to deploy the orchestrated graph to the mobile device was not working well due to connection issues. This situation was still positive, because it showed that not enough feedback is provided during the paring process between mobile device and web browser. To avoid this, the visual user-interface is guiding the user to avoid situations where the user does not feel confident. The guiding is done through restricting the possibilities such as to hide not possible connections. In comparison to the paring process, it shows that the pro-active approach works well since nobody complained about restrictions and one participant noticed that only user-actions are possible which make sense.

To improve the user-interface, we thought about to add a tablet optimized user-interface which is controllable per finger moves. Contrary to our expectation, this is not required by users, which rather use the computer to orchestrate devices, because they don't expect to rearrange their orchestrations every day.

The high amount of new smart devices let imagine unlimited use-cases. While the current amount of plug-ins supported by the prototype is very limited, the possibility to invent new connections is restricted. To solve this, more smart devices must be integrated by Dynamix plug-ins to enable users to orchestrate smart devices connections, which can enrich their daily life.

Fortunately, the participants of the user study were very interested to be creative and try new connections. It is not sure if this apply for a wider field of users. It might be the case that users generally just want to use orchestrations instead of invent them. This could lead to a market system, where smart device orchestrations can be downloaded. Some advanced users are generating the graph orchestrations, while a majority is just applying them to their smart space.

7.2 Conclusion

We are currently witnessing the invention of more and more smart devices, which have an increasing impact on our daily life. Most of these devices can only be controlled by a native application on a mobile device or through other closed systems. Direct interaction between smart devices could introduce a huge amount of new use-cases. For most smart devices such direct interaction is not possible because of hardware and software restrictions. Since most smart devices are already able to connect to a mobile device e.g. smart phone. Ambient Dynamix is using the mobile device as a platform to interact with smart devices. It is installed as a background service and is able to integrate and interact with smart devices through downloadable plug-ins. A library called Ambient Control runs as a plug-in within Dynamix and provides a standardized means of describing each smart device with inputs and outputs. These provides a way to orchestrate a setup of inter-device communications while using Dynamix to communicate with smart devices. Ambient Dynamix and Ambient Control are complex, even for experienced programmers. To support orchestrations of smart devices for users without programming experience a simplified user-interface was required.

This thesis introduced an approach to connect smart devices through visual programming. The resulting visual interface was a representation of smart devices, which were visualized in a graph representation. Smart devices were represented as nodes which were connected through edges.

Each device node had a specific layout, which contained inputs and outputs of the represented smart device. Each input and output was related to a command type and showed user directly, which type of data are used, such as color or movement data. Connections between device nodes were dragged from an output to an input and the other way round. While dragging, the connection starting at an output was only suiting inputs on other nodes which were shown active. Inputs which were not possible to match have been hidden and won't be connected. This feature was used to guide the interactions of the user because it was not possible to make an incorrect connection.

For the user it was beneficial do a quick interactive tutorial which guided him to orchestrate the first connection. After completing the tutorial, all participants of the user study were able to connect a first smart device connection on their own.

The user-interface only supported actions, which were reasonable and helped the user to orchestrate smart devices. On the one hand, this made the user feel confident and improved the motivation to be creative and connect smart devices. While on the other hand it is not possible to do small adaptions of parameters which reduces the flexibility.

The possibility to deploy the graph directly on the mobile device to set up the smart environment showed the user instant the result of connected smart devices. To improve the user experience, more smart devices must be integrated and further extensions can be made as mentioned in the section of future work.

Overall, by introducing graphs, nodes, translators and connections, we invented a user-interface which can be used by users without coding experience to orchestrate smart devices. In addition, the evaluation of the user study showed, that users are very interested to orchestrate various smart devices.

7.3 Future Work

In the future we want to add several extensions to increase the usability to orchestrate smart devices. One feature could be, to detect smart devices in the personal smart space. This can be done by scanning a QR-code or NFC-tag, which are used to identify the smart device. All scanned smart devices could then appear in the web user-interface, ready to connect with other smart devices. This decreases the difficulty to find the right plug-in for a specific smart device. A user can just walk through the room, scan devices like smart lights, smart switches and smart speakers. Afterwards the orchestration to connect these devices can be done through the web user-interface.

It is planned to let more users work with this system to get feedback of users which are regular using this web user-interface to connect smart devices. To avoid frustration, the pairing process must be improved first by giving more valuable user-feedback.

To deploy this system, we first want to focus on a particular area, where it is possible to only integrate devices which are of interest for this area. After users are used using the smart device orchestration in a specific domain, it can be extended to other fields. Such a first domain could be physically handicapped people, because they have a clear need for adaptation and interaction assistance.

To enrich use-cases, we plan to add nodes which represent functionalities instead of smart devices. These functions could be provided by pure functional plug-ins instead of device plug ins. Functional and logical statements can then be used to calculate the output, based on the provided input.

Related to functional nodes, we think about actions which can be initiated automatically. These automatic triggers can be used to send specific message as a trigger event. These can also be implemented in a reoccurring behavior.

Most of the user study participants requested a database with contains smart device orchestrations. Therefore, we are planning to integrate a store of orchestrations. The user can query the store for an orchestration which suits his connected devices or a orchestration, which suits a specific use-case. This can be one of the extensions we want to realize in the future.

Bibliography

[CS12] D. Carlson, A. Schrader. Dynamix: An open plug-and-play context framework for android. In *Internet of Things (IOT), 2012 3rd International Conference on the*, pp. 151–158. IEEE, 2012. (Cited on page 15)

[EHS69] T. O. Ellis, J. F. Heafner, W. Sibley. The GRAIL language and operations. Technical report, DTIC Document, 1969. (Cited on pages 10 and 11)

[GHHA10] M. García-Herranz, P. A. Haya, X. Alamán. Towards a Ubiquitous End-User Programming System for Smart Spaces. *J. UCS*, 16(12):1633–1649, 2010. (Cited on page 9)

[Mye90] B. A. Myers. Taxonomies of visual programming and program visualization. *Journal of Visual Languages & Computing*, 1(1):97–123, 1990. (Cited on page 11)

[O$^+$12] F. Oliphant, et al. Meemoo: Hackable Web App Framework. 2012. (Cited on page 23)

[PC14] M. Pagel, D. Carlson. Ambient Control: A Mobile Framework for Dynamically Remixing the Internet of Things. pp. 1–9. 2014. (Cited on pages 18 and 21)

[RMMH$^+$09] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, Y. Kafai. Scratch: Programming for All. *Commun. ACM*, 52(11):60–67, 2009. doi:10.1145/1592761.1592779. URL http://doi.acm.org/10.1145/1592761.1592779. (Cited on page 11)

[VBCMV$^+$12] M. Vega-Barbas, D. Casado-Mansilla, M. A. Valero, D. López-de Ipina, J. Bravo, F. Flórez. Smart spaces and smart objects interoperability architecture (S3OiA). In *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on*, pp. 725–730. IEEE, 2012. (Cited on pages 9 and 10)

All links were last followed on February 08, 2015.

Appendix

RESTful - Interface ControlProfileServer

URL of the Server-Application:

http://localhost:8080/ControlProfileServer

GET PluginControlDescription

ALL available PluginControlDescriptions (Artifact_IDs)

Return complete PluginControlDescription List as JSON String format.
/PluginControlDescprition/ids?format={format}
Result: JSON | XML List with all IDs
Default format without parameter format: JSON

Get ONE specific PluginControlDescription

Different formats, parameter: format = {XML|JSON}
/PluginControlDescprition/ids/{id}?format={format}
{id} e.g. org.ambientdynamix.contextplugins.ambientmedia
{format} = XML or JSON (complete PluginControlDescription)
Default format without parameter format: XML

Debug easy access refer by name:

Same as above. Because name is not unique please use not in finished code.
/PluginControlDescprition/names
/PluginControlDescprition/names/{name}?format={format}
{name} e.g. ambientmedia
{format} = XML or JSON (complete PluginControlDescription)

PluginControlDescription

PUT new/ POST update existing PluginControlDescription

Write to server from Client, requested format = (default) XML
/PluginControlDescprition/ids/{id}?format={format]

DELETE existing PluginControlDescription

DELETE existing PluginControlDescription from server.
/PluginControlDescprition/ids/{id}

QUERIES

All PluginControlDescriptions with requested input

ControlProfiles that can be controlled by/work on those inputs/outputs
/PluginControlDescprition?Input={SENSOR_PYR, MOVEMENT_UP}&Output={MOVEMENT_DOWN}&Constraint={ALL|ONE}
Return JSON List of all PluginControLDescriptionIDs
Important: Command could be used more than one time in a query.

RESTful - Interface ControlProfileServer

Query translators

All translators with requested inputs/outputs

Return IDs of all translators.
/Translator?Input={SENSOR_PYR}&Output={MOVEMENT_DOWN}

GET PluginControlDescription
/PluginControlDescprition/names
/PluginControlDescprition/names/{name}
/PluginControlDescprition/names/{name}?format={format}

/PluginControlDescprition/ids
/PluginControlDescprition/ids/{id}
/PluginControlDescprition/ids/{id}?format={format}

PUT/POST PluginControlDescription
/PluginControlDescription/ids/{id}?format={format}

DELETE PluginControlDescription
/PluginControlDescription/ids/{id}

Query PluginControlDescriptions
/PluginControlDescprition?Input={MOVEMENT_DOWN}
/PluginControlDescprition?Output={MOVEMENT_DOWN}
/PluginControlDescprition? Input={SENSOR_PYR, MOVEMENT_UP}&Output={MOVEMENT_DOWN}

Query Translations
/Translator?Input={MOVEMENT_DOWN}
/Translator?Output={MOVEMENT_DOWN}
/Translator? Input={SENSOR_PYR, MOVEMENT_UP}&Output={MOVEMENT_DOWN}

Query ControlGraphs
/ControlGraph/{id}?format={format}

User Study

Bachelorarbeit - Matthias Moegerle

AmbientControl - WebGraph

Universität Stuttgart

NUS National University of Singapore

Geführtes Interview - Hinführung:

____
Datum und Uhrzeit

Info: Erklärung der Durchführung: Interview, Aufgaben, Interview

Info: Vorstellung Internet of Things, simplifiziert: Interaktion der umgebenden Geräte. User-Interaktion nur teilweise nötig. Geräte können z.T. über eine Nutzeroberfläche wie z.B. das Smartphone gesteuert werden.

1. Frage: Was für Geräte hast Du bereits mit dem Smartphone verbunden?

2. Frage: Angenommen, Du kannst alle Geräte in Deiner Umgebung steuern, welche würdest Du gerne mit dem Smartphone steuern?

Info: Bisher können bereits viele Geräte mit dem Smartphone verbunden werden und sogar miteinander interagieren (AmbientControl).
Use case Ambient-TV: Nutzer schaut TV mit dem AppleTV und möchte auf die Toilette. Darum macht der Nutzer eine Geste mit seinem Arm, wodurch der Film pausiert. Als der Nutzer dann wieder zurück kommt macht er eine weiter Geste und der Film wird fortgesetzt.

Durchführung - WebGraph:

Info: Beim folgenden Tutorial werden Geräte so verbunden, dass sich über ein Armband (genannt Myo) der AppleTV (genannt ambientmedia) gesteuert werden kann. Das Myo ist ein Armband, welches die Armbewegungen an das Smartphone weiterleitet. Der Apple-TV kann durch das Smartphone als Fernbedienung gesteuert werden. Somit werden auf dem Smartphone Arm-Befehle an den Apple-TV weitergeleitet. Die Verbindung der Geräte wird auf der Webseite (der praktische Teil meiner Bachelorarbeit) zusammengestellt. (Video zu Myo und iTV)

Nutzer: Online-Tutorial

Info: Erklärung des Sphero und des Hue(Lifx)-Beleuchtung (Video zu Sphero und Lifx)

Nutzer: Steuere die Lichtfarbe der Hue mit dem Sphero

Nutzer: Lade die Konfiguration über die Weboberfläche auf das Smartphone.

Geführtes Interview - Zukunftsfragen (Offene Fragen!)

3. Frage: Würdest Du gerne Geräte-Verknüpfungen herunterladen und mit Freunden teilen?

4. Frage: Es gibt in Zukunft mehr tragbare Geräte. Welche Geräte möchtest Du gern miteinander Verbinden?

5. Frage: Wie sehr steigt die Nutzbarkeit, wenn die Webseite auf einem Tablet/iPad mit den Fingern zu bedienen ist?

6. Frage: Was mochtest Du besonders?

7. Frage: Was hat Dir weniger gefallen?

8. Frage: Welche Zusatzfunktionen würdest Du Dir in einer zukünftigen Version wünschen?

User Study

Bachelorarbeit - Matthias Moegerle

AmbientControl - WebGraph

Universität Stuttgart

NUS National University of Singapore

____

Datum und Uhrzeit

Angaben zur Person:

Alter:

Geschlecht:

Tätigkeit:

Ich besitze ein Smartphone: ☐ ja ☐ nein

	STIMME GAR NICHT ZU				STIMME VOLL ZU
Ich probiere gerne neue Technik-Trends	☐	☐	☐	☐	☐
Ich kann ohne mein Smartphone überleben	☐	☐	☐	☐	☐

Interview - Einleitung

1. Aufgabe: Durchführung Online-Tutorial (Myo steuert Ambient Media)

2. Aufgabe: Erstelle einen neuen Graph und erstelle den Graph um die Lichtfarbe der Hue mit dem Sphero zu steuern.

	STIMME GAR NICHT ZU				STIMME VOLL ZU
Ich kann mir sehr gut vorstellen, das System regelmäßig zu nutzen.	☐	☐	☐	☐	☐
Ich empfinde das System als unnötig komplex.	☐	☐	☐	☐	☐
Ich empfinde das System als einfach zu nutzen.	☐	☐	☐	☐	☐
Ich denke, dass ich technischen Support brauchen würde, um das System zu nutzen.	☐	☐	☐	☐	☐
Ich finde, dass die verschiedenen Funktionen des Systems gut integriert sind.	☐	☐	☐	☐	☐
Ich finde, dass es im System zu viele Inkonsistenzen gibt.	☐	☐	☐	☐	☐
Ich kann mir vorstellen, dass die meisten Leute das System schnell zu beherrschen lernen.	☐	☐	☐	☐	☐
Ich empfinde die Bedienung als sehr umständlich.	☐	☐	☐	☐	☐
Ich habe mich bei der Nutzung des Systems sehr sicher gefühlt.	☐	☐	☐	☐	☐
Ich musste eine Menge Dinge lernen, bevor ich mit dem System arbeiten konnte.	☐	☐	☐	☐	☐

3. Aufgabe: Lade die Konfiguration über die Weboberfläche auf das Smartphone.
Dazu im Hauptmenü bitte folgenden Menüeintrag wählen: 📱

	STIMME GAR NICHT ZU				STIMME VOLL ZU
Ich kann mir sehr gut vorstellen, das System regelmäßig zu nutzen.	☐	☐	☐	☐	☐
Ich empfinde das System als unnötig komplex.	☐	☐	☐	☐	☐
Ich empfinde das System als einfach zu nutzen.	☐	☐	☐	☐	☐
Ich denke, dass ich technischen Support brauchen würde, um das System zu nutzen.	☐	☐	☐	☐	☐
Ich finde, dass die verschiedenen Funktionen des Systems gut integriert sind.	☐	☐	☐	☐	☐
Ich finde, dass es im System zu viele Inkonsistenzen gibt.	☐	☐	☐	☐	☐
Ich kann mir vorstellen, dass die meisten Leute das System schnell zu beherrschen lernen.	☐	☐	☐	☐	☐
Ich empfinde die Bedienung als sehr umständlich.	☐	☐	☐	☐	☐
Ich habe mich bei der Nutzung des Systems sehr sicher gefühlt.	☐	☐	☐	☐	☐
Ich musste eine Menge Dinge lernen, bevor ich mit dem System arbeiten konnte.	☐	☐	☐	☐	☐

Interview - Zukunftsfragen

Vielen Dank für die Teilnahme!

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature