Thomas Kubitza (2015):
Towards a Toolkit for the Rapid Creation and Programming of Smart Environments.
In International Reports on Socio-Informatics (IRSI),
Proceedings of the CHI 2015 - Workshop on End User Development in the Internet of Things Era
(Vol. 12, Iss. 2, pp. 61-66)

Towards a Toolkit for the Rapid Creation and Programming of Smart Environments

Thomas Kubitza

University of Stuttgart
thomas.kubitza@vis.uni-stuttgart.de

Abstract. “Smart” environments rely on the interconnection of various devices that are equipped with sensors and actuators and are statically or dynamically deployed in rooms or buildings or worn by users. Although it has become much easier to build and program single device installations with platforms such as Arduino, it still remains a challenging task to build whole environments with heterogeneous interconnected devices. A lot of thiseffort is due to the implementation of similar functionality in different programming languages for different device platforms and the bridging of the different communication technologies, protocols and formats between devices. The author believes that the right toolkit can widely cover this technical complexity so that designers and users of a smart environment can focus on the interaction design and the programming of intelligent and useful behavior. Such a toolkit is currently being developed within the meSch EU project and is presented and demonstrated by the author.

1 Introduction

Smart and interactive spaces are based on a common principle; different kinds of devices with sensors and actuators attached are statically installed in rooms, levels, whole buildings or are even worn by users. All these heterogeneous devices need to talk to each other or to an entity that constantly combines system state and generates system reactions. Multiple reasons make the setup of such environments a complex task. Firstly, similar functionality has to be implemented

on various platforms ranging from microcontrollers to High-End computers. This requires expert knowledge in very specific programming languages and platforms as well as the management of various development environments (IDEs) and compilation tool chains. Secondly, different communications technologies, protocols and formats have to be bridged so that devices can actually exchange data. Thirdly, devices have to be deployed in their target environment, supplied with electricity and wired or wireless communication infrastructure (e.g. WiFi access points). Especially the first two reasons put up high boundary for non-experts in electronics and programming. This limits its usage to small and mostly only professional user groups. We believe that the right tools can open up the creation of smart environments to a much larger audience in the same way as physical prototyping platforms such as Arduino made the access to microcontrollers much easier and in the same way as Apps made potentially everyone the programmer of his own cell phone (and the phones of millions of others). By empowering groups such as user experience designers, scientist, designers, artists, makers and hobbyists we envision the creation of a large set of truly useful applications evaluated in realistic environments and addressing a broad range of problems. In this workshop-paper a toolkit is presented that drastically reduces the technical complexity for creating and programming smart environments. Our approach consists of two main pillars: (1) A client software for each type of end device is provided which allows to remotely access and control all its abilities and to abstract from its specific platform. (2) A server software on a central computer node is provided to bridge between various communication-technologies and to provide unified access to all sensors and actuators of configured devices through a web based JavaScript development environment. This approach allows to quickly implement or change the behaviour of a system without the need to reprogram or physically access any of the associated or deployed devices. JavaScript, one of the most widespread and growing programming languages, is used to define the system behaviour in a singlespot.

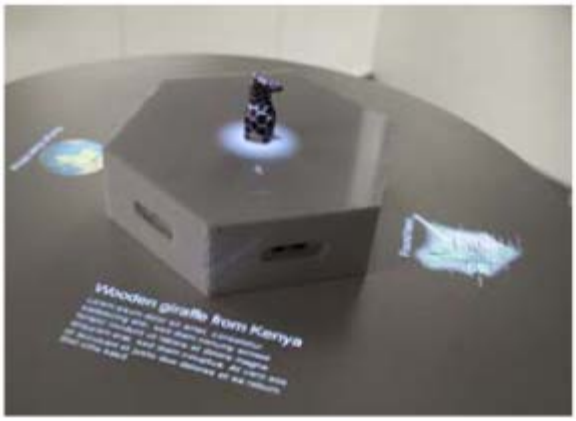

Figure 1: Interactive museum setup – a WiFi projector lamp and a distance sensor plinth with an exhibit on top

Figure 2: Various interactive content can be projected around the exhibit placed on the plinth based on visitors detected in proximity

2 System Concept

Conceptually our toolkit strictly follows a master-slave architecture; the client side software allows to treat each client device as a source of sensor events, a sink for actuation commands, or both. It allows abstracting from the underlying operation system and hardware as well as the communication technology and protocol. For each end device platform a specific client firmware has to be implemented. However, this implementation effort is done just once by experts for this specific platform; after that, users of our toolkit just need to install the

firmware once and from that point on they can access and control all its abilities from the central server node.

Figure 3: a) Graphical device configuration, b) Rules overview and control, c) Code auto-completion in rules editor

The central server node provides a web based user interface for the configuration of devices and the creation of behaviour rules. It includes a rule engine that triggers events and runs rules as soon as sensor data is received. Rules may then

again trigger actuator commands which are instantly sent to the appropriate devices.

3 User Interface

The toolkit user interface runs in any browser and is structured in three sections: Devices, Events and Rules. New devices automatically appear in the device overview after the installation of their specific client software. In its initial state a device has no modules configured; this means no sensor or actuators are activated yet. In the device configuration view, modules can be selected from a list of supported sensors and actuator modules (figure 3a). By double-clicking on a module, it is automatically assigned to a compatible and free port; this way users don't need to care about the right attachment of sensors beforehand, the device configuration steps already gives visual hints about the right ports to connect to. After saving a configuration, it is instantly pushed to the device where the selected modules are activated. Client software for a wide range of physical prototyping and mobile platforms is already available.

In the the rules-view an overview of available rules is given. These are structured in groups and consist of a name, a description and an execution priority. Groups allow to tie together rules which are logically associated or only apply to a certain space in the environment (e.g. "Office Floor 1"). Groups and rules can be enabled and disabled individually which allows easy instant switching of behaviour for single spaces or whole environments (figure 3b). Single rules can be edited any time and the changes are applied immediately. Syntax highlighting and auto-completion features help novice users to shimmy along available devices, modules and their individual properties without any previous knowledge (figure 3c); advanced users benefit from the coding speedup and correct referencing of objects. Multiple application setups in the cultural heritage domain (see figures 1 and 2) as well as in office automatisatin have been already realized using our toolkit – more details on these will be given during the workshop.

4 Demonstration during Workshop

The author will bring all necessary components to the workshop so that multiple groups can independently get hands-on experience with the toolkit. The components will include two smartphones, two tablets, two WiFi projectors, two .NET Gadgeteer kits, two Arduino kits, multiple RPis and Intel Edisons, various sensors and actuators as well as multiple Bluetooth Low Energy beacons for proximity experiments. Participants will be able to program all these devices through a web based IDE which they can run in the browser of their own PC or

tablet. Further, devices can be integrated that are brought by participants (e.g. Android smartphones, fitness trackers, etc.). Quick and simple mashups can be demonstrated as well as more advanced setups that integrate a larger set of heterogeneous devices and more complex logic.

5 Conclusion and Future Work

The toolkit introduced in this workshop-paper aims to drastically reduce the technical effort for setting up a smart environment and to provide a single interface in which its behavior can be programmed using a popular web programming language. We envision this toolkit to be aplatform for more advanced research on the programming of smart environments and interactive spaces: We are planning to extend the toolkit with various additional layers that lie on top of the Javascript interpreter and target users with different expertise, e.g. configurable behavior scripts, visual programming, programming by natural language. Further, we are exploring new ways of programming environments apart from writing code on a single desktop PC, e.g. automatic code creation by demonstration as well as proximity based code generation and assistance. We further want to use the power of the community and are therefore developing means that allow easy sharing and integrating single behavior rules or full setups with a single click. The author believes that the presentation and hands-on experience with the toolkit can lead to an active discussion and extremely valuable feedback from the EUD community. This will be an important input into the further development of the toolkit and related projects.