A Visual Design Toolset for Drag-and-drop Smart Space Configuration

Darren Carlson*, Matthias Mögerle†, Max Pagel*, Shivam Verma*, and David S. Rosenblum*
*Felicitous Computing Institute, National University of Singapore
{carlson, pagel, shivam, david}@comp.nus.edu.sg
†Institute for Visualization and Interactive Systems, University of Stuttgart
matthias.moegerle@studi.informatik.uni-stuttgart.de

I. Background

The number of networked smart devices available in everyday Internet of Things (IoT) environments is rapidly increasing; however, many current devices adopt mutually incompatible networks, protocols, and application programming interfaces (APIs). For example, devices like the Sphero Robotic ball and Parrot AR Drone helicopter both provide dedicated controller apps, but remain inherently incompatible. The Sphero supports sensor data streaming of its inertial measurement unit (IMU) over a Bluetooth connection, which can be used to determine the current orientation of the robot using a proprietary API. The AR Drone supports data connectivity over WiFi and accepts flight control commands that can be used to remotely pilot the drone. A grasped Sphero device could theoretically be used as an intuitive flight controller for the drone (by feeding its streaming IMU data into the drone's flight control API); however, such interactions are not inherently supported by the devices.

A. Ambient Dynamix

To address the network, protocol and API heterogeneity challenges introduced above, we developed *Ambient Dynamix* (Dynamix for short) [1], a plug-and-play middleware framework that enables mobile apps and Web apps to sense the user's context (e.g., location, identity, activity) and perform fluid smart device interactions through plug-ins that can be dynamically installed into the user's Android-based mobile device (e.g., smartphone or tablet) on-demand. Dynamix runs as lightweight background service, leveraging the user's mobile device itself as a sensing, processing and communications platform. Dynamix automatically discovers, downloads and installs the plug-ins needed for a given sensing or control task. When the user changes environments, new or updated plug-ins can be deployed to the device at runtime, without the need to restart the framework. Dynamix comes with a growing collection of ready-made plug-ins and provides open software development kits (SDKs) and a scalable repository architecture, which enable 3rd party developers to quickly create and share new plug-in types with the community.

B. The Ambient Control library

To enable dynamic "remixing" of encountered IoT resources, we developed the *Ambient Control* (AC) library for Dynamix [2], which aims to simplify the ad-hoc discovery, selection and integration of smart devices in highly heterogeneous environments. The AC library defines a set of control commands that can be associated with Dynamix plug-ins as a mechanism for unifying their interaction semantics and associated data. It also includes a Smart Wiring feature that optimally matches the inputs and outputs of the plug-ins in a given control graph according to priority values. The library supports 1-to-n connections that involve 1 receiver and any number of controllers. We call such a configuration a control graph, since it consists of nodes (plug-ins) and directed control edges between nodes, over which certain types of controls are exchanged. The library coordinates requested control graphs by managing required plug-in installations via Dynamix, handling the setup handshake process between plug-ins and managing full duplex communication channels between controllers and a receiver. Plug-in control profiles are stored inside a Dynamix web service, which offers a REST interface for configurable, query-based access.

II. Ambient Flow Demonstration

Although manual control graph creation works well in experimental scenarios, configuring real-world environments requires addressing multiple interaction patterns and a larger number of devices, which can quickly become complex when writing configurations by hand [3]. To address this complexity, we devised a smart space configuration approach, called Ambient Flow [4], which enables non-programmers to create control graphs visually, and then load them into a paired Dynamix-based device for realization. This demonstration showcases the Ambient Flow prototype shown in Figure 1.

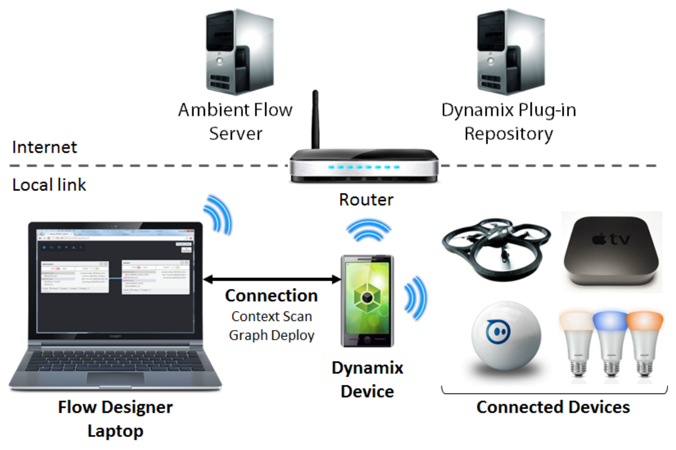

Fig. 1. Ambient Flow prototype

The demonstrator explores techniques for representing IoT scenarios using a flow-based programming model. Flow-based programing [5] is a visualization technique that allows users without coding experience to understand, create and alter programs by manipulating graphical blocks that represent program components or functionalities. These blocks expose available parameters and interaction possibilities, such as inputs and outputs. Completed control graphs can be collapsed into a single subgraph to allow for a more tidy representation. Global inputs and outputs for a complete subgraph can be assigned, which enables interactions between graphs or groups of smart objects. By configuring individual blocks, and connecting the inputs and outputs of various blocks together, a user can create a fully functional program that solves a given task.

During the demonstration, participants use the Web-based ”Flow Designer” (running on a laptop) to visually create smart space configurations using a drag-and-drop visual interface. The Flow Designer was implemented by extending the open-source Meemoo framework [6], which provides high-level visual programming features such as block rendering, wiring support, subgraphs, etc. To enable smart device sensing from the browser, the Flow Designer supports remote pairing with a remote Dynamix instance (running on an Android mobile device) when situated in the same local network. The pairing process operates using an out-of-band credential exchange (a pairing token shared optically via a barcode).

Live information from the remotely paired Dynamix device (e.g., a smartphone) is used to render discovered connected devices in the Flow Designer as block diagrams that can be placed on the canvas. The Flow Designer utilizes a REST interface provided by the Ambient Flow Server (AFS) to obtain high-level input and output data-types for each smart device detected in the environment. Connections between blocks are made by dragging virtual ”wires” between device outputs and inputs. The interface guides users when creating connections by graying out invalid targets and by adding protocol translators as required, as shown in Figure 2.

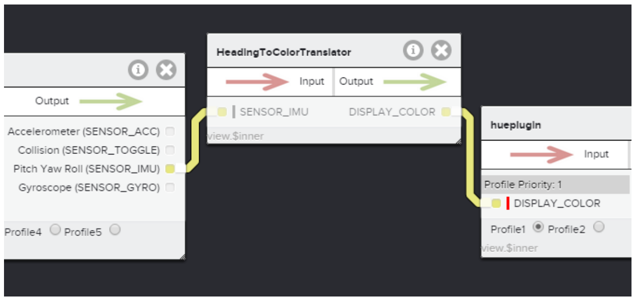

Fig. 2. A completed control graph shown in the Flow Designer

By connecting the inputs and outputs of various blocks together, participants are able to mix and match control surfaces and controllable objects in new, playful and potentially unforeseen ways. For example, the Sphero robot might be connected to a Hue network light in such a away that the orientation of the robot controlled the color of the Hue light. Alternatively, the Sphero might also be used as a flight controller for the Parrot drone or as a media playback controller for an Apple TV. The demonstrator supports a variety of interaction scenarios that can be designed and realized on-the-fly by participants.

Completed control graphs can be sent over the network from the Flow Designer to the paired Dynamix device for testing. The AC library running within the Dynamix device receives incoming control graphs (as XML), parses the configuration, uses the Dynamix instance to install specified plug-ins, and then sets up the plug-in intercommunication channels necessary to render the graph. As completed control graphs contain live data from the environment (e.g., device identifiers), the underlying plug-ins are able to properly connect with specified devices. Figure 3 shows a user experimenting with a deployed control graph by rotating a Sphero device to control the color of a Hue network light.

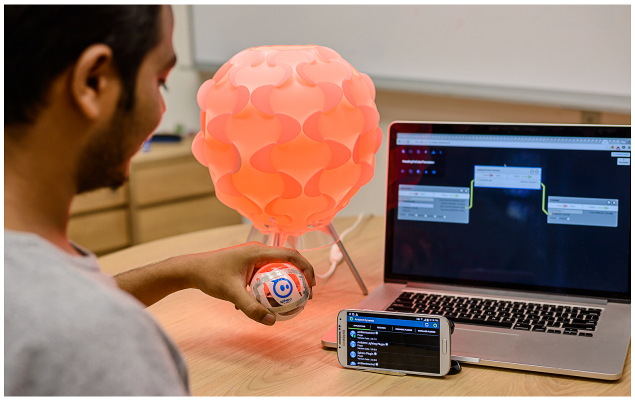

Fig. 3. Connecting a Sphero robot ball to a Hue-based smart lamp.

Finalized smart space designs can be published to the AFS (or a private repository) together with contextual scoping tags (e.g., radio beacon or geo-fence data). During runtime, mobile users are able to discover and try shared designs when situated within the specified context using only a Dynamix-based device.

We envisage this type of tooling as useful for new types of design professionals that will likely emerge as the IoT continues to expand into everyday environments. Such ”ambient designers” will likely prefer to focus on creating innovative user experiences rather than solve low-level IoT interoperability issues. Accordingly, Ambient Flow lowers the complexity of smart space orchestration through efficient user guidance and automated configuration.

References

[1] D. Carlson and A. Schrader, “Dynamix: An open plug-and-play context framework for android,” in *Internet of Things (IOT), 2012 3rd International Conference on the*, Oct 2012, pp. 151–158.

[2] M. Pagel and D. Carlson, “Ambient control: A mobile framework for dynamically remixing the internet of things,” in *IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks*. IEEE, 2015, Conference Proceedings.

[3] M. Blackstock and R. Lea, “Iot mashups with the wotkit,” in *International Conference on the Internet of Things (IoT 2012)*. IEEE, Conference Proceedings.

[4] D. Carlson, M. Mögerle, M. Pagel, S. Verma, and D. S. Rosenblum, “Ambient flow: A visual approach for remixing the internet of things,” in *Proceedings of the 5th International Conference on the Internet of Things (IOT 2015)*. IEEE, 2015, Conference Proceedings.

[5] J. Morrison, *Flow-based Programming: A New Approach to Application Development*. J.P. Morrison Enterprises, 2010.

[6] F. Oliphant *et al.*, “Meemoo: Hackable web app framework,” 2012.