Demo Abstract: Augmenting Homes with Custom Devices Using .NET Gadgeteer and HomeOS

James Scott
Microsoft Research,UK
jws@microsoft.com

A.J. Bernheim Brush
Microsoft Research, USA
ajbrush@microsoft.com

Ratul Mahajan
Microsoft Research, USA
ratul@microsoft.com

Abstract

We describe how two of our recent research technologies, .NET Gadgeteer and HomeOS, complement each other in enabling research using custom devices in home deployments. .NET Gadgeteer enables rapid prototyping of custom devices based on solderless assembly from a wide range of publicly available hardware modules, and managed C# software. HomeOS provides abstraction layers so that "apps" can be written which leverage devices in the home (e.g. wireless sensors, webcams) using standard APIs. In combination, HomeOS and .NET Gadgeteer make it easy to construct custom devices which interact with HomeOS apps. We showcase an example custom device and application a plumbing leak sensor and leak notification application.

1 Introduction

There is a plentiful and growing body of research into introducing new technology in the home. Such studies often require deployment of devices (sensors, actuators, physical user interfaces) as well as application software. Over the past few years at Microsoft Research, we have developed two complementary publicly-available systems relevant to home deployments: HomeOS and .NET Gadgeteer. We have integrated these systems to help enable home-deployment research. We briefly describe both systems, and an example .NET Gadgeteer device and associated HomeOS application which can serve as a template for future research projects in this area.

2 HomeOS

HomeOS is a platform which provides standard APIs for software running on a hub device (e.g. a PC) to communicate with and control devices in the home such as thermostats, wireless cameras, etc. [1]. HomeOS "drivers" communicate to devices using protocols that are specific to the device but expose to applications high-level APIs that depend on the type of the device and its functionality. For instance, the API for a light includes "on" and "off" commands. Therefore, HomeOS applications can be independent of device protocols and vendors.

HomeOS currently runs in 12 homes in the Pacific Northwest region of the USA. It has support for many types of devices such as light switches, dimmers, door/window sensors, and cameras. We have also written eighteen applications that use these devices in various ways. We have made the HomeOS prototype available to academic institutions. Over 50 students across twenty research groups have developed applications and drivers for HomeOS (see [2]). Applications include energy profiling, remote monitoring, and end user programming.

3 .NET Gadgeteer

.NET Gadgeteer [3] is an open source platform for rapidly prototyping custom devices, providing solderless assembly of hardware from a wide range of modules, programming using managed code (C#) rather than low-level C code, and support for physical prototyping. .NET Gadgeteer can be used to build a very broad range of devices, thanks to a range of hardware modules [4] including sensors (gyro, compass, etc.), networking (WiFi, Bluetooth, etc.), user interface elements (LCDs, joysticks, etc.), actuation (motor/servos, relays, etc.) This hardware is sold by multiple manufacturers (e.g. through amazon.com or mouser.com) and the open source .NET Gadgeteer hardware specifications and core platform software allow for interoperability across manufacturers.

4 Custom Devices for the Home

HomeOS and .NET Gadgeteer are highly complementary when it comes to home deployments. To illustrate why, we discuss what happens when each is used in isolation.

HomeOS can be used without .NET Gadgeteer, if the necessary sensors, actuators and I/O units can be found "off the shelf", e.g. existing devices are available which use a supported interface such as z-wave or Insteon. However, in our experience [5], which echoes that of others [6], such commercial off-the-shelf (COTS) devices are often a problem for research, for a number of reasons.

First, COTS devices are finite and inflexible. If one exists that is precisely what is required, then all is well. However, research deployments often involve new types of devices and sensors, which are not available. Furthermore,

Buildsys'12, November 6, 2012, Toronto, ON, Canada.
ACM 978-1-4503-1170-0

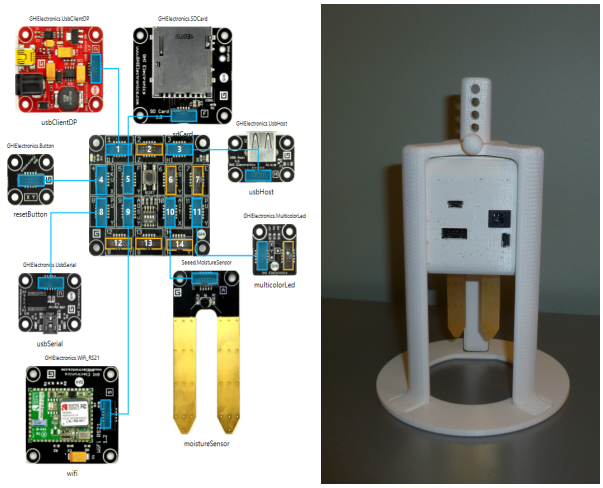

Figure 1. .NET Gadgeteer modules (left) and encased prototype (right) for moisture sensor

.

even if a COTS device is available, even small changes to the devices operation may be impossible, e.g. modifying the polling period of a sensor, or providing a custom user interface. Second, a deployment may often require a range of COTS devices using a number of different standards for communication and control. This leads to an overall system quickly becoming a hodge-podge prone to failures and hard to maintain. Third, COTS devices may not be designed for failsafe operation and deep logging that is necessary for a successful research experiment. For example, in a previous study on predictive heating [5], we looked at COTS Passive Infra-Red (PIR) motion sensors and none were quite right. One such sensor only sent a message when motion started or ended, so when a message was lost, there would be an indefinitely long period of misbehavior due to the incorrect motion state, and lack of access to raw motion events inhibited our ability to verify ground truth occupancy.

On the other hand, .NET Gadgeteer can be used to build custom devices for the home, without HomeOS. However, we often find with home deployments that multiple custom devices are required, and that there may need to be a "backend" software component that communicates with them all (e.g. running on a PC). HomeOS provides a way to easily integrate custom devices and the software backend, as well as manage the configuration of unique devices in each house. It also enables COTS devices to be easily used as part of a deployment, thereby avoiding the overhead of building custom devices when possible.

5 Example HomeOS/.NET Gadgeteer Device

We have developed an example device with Gadgeteer and an associated HomeOS application: a moisture sensor to detect plumbing leaks before they cause significant damage. While the sensor and application are simple, they provide a template for rapid prototyping of custom home devices and software.

Figure 1 illustrates the Gadgeteer-based prototype of a moisture sensor which uses WiFi and a RESTFUL API to communicate with HomeOS. One challenge with custom devices in homes is associating them with the home network. Showcasing the flexibility of Gadgeteer and HomeOS, we have implemented numerous association techniques including the ability to put a USB memory stick or SD card with a credentials.txt file to provide credentials (using the USB Host and SD card modules), and a USB serial interface so that the device can be plugged (temporarily) straight into a PC running HomeOS, which will send it WiFi credentials automatically (using the USB Serial module). We use a multicolor LED module to provide color-code feedback and a button module to provide a reset button.

Once the credentials have been transferred, HomeOS and the Gadgeteer device communicate using standard TCP/IP networking to perform device discovery, functionality discovery and provide a sensor data feed that can be used by HomeOS apps. We wrote a simple leak detection app which alerts the user if a leak is detected.

6 References

[1] C. Dixon, R. Mahajaran, S. Agarwal, A. J. Brush, B. Lee, S. Saroiu, and P. Bahl. An Operating System for the Home. In *Proc. of the NSDI'12: 9th USENIX Symp. on Networked Systems Design and Implementation*, pages 337–352, April 2012.

[2] HomeOS. http://research.microsoft.com/homeOS.

[3] N. Villar, J. Scott, S. Hodges, K. Hammil, and C. Miller. .NET Gadgeteer: A Platform for Custom Devices. In *Proc. of Pervasive 2012*, pages 216–233, June 2012.

[4] Gadgeteer. http://netmf.com/gadgeteer.

[5] J. Scott, A. J. Bernheim Brush, J. Krumm, B. Meyers, M. Hazas, S. Hodges, and N. Villar. PreHeat: Controlling Home Heating Using Occupancy Prediction. In *Proc. of the 13th Int'l Conf. on Ubiquitous Computing*, pages 281–290, 2011.

[6] T. W. Hnat, V. Srinivasan, J. Lu, T. I. Sookoor, R. Dawson, J. Stankovic, and K. Whitehouse. The hitchhiker's guide to Successful Residential Sensing Deployments. In *Proc. of the 9th Int'l Conf. on Embedded Networked Sensor Systems*, pages 232–245, 2011.