Connecting the Physical World with Arduino in SECE

Hyunwoo Nam
Department of Electrical Engineering
Columbia University
New York, NY
hn2203@columbia.edu

Jan Janak
Department of Computer Science
Columbia University
New York, NY
janakj@cs.columbia.edu

Henning Schulzrinne
Department of Computer Science
Columbia University
New York, NY
hgs@cs.columbia.edu

***Abstract*—The Internet of Things (IoT) enables the physical world to be connected and controlled over the Internet. This paper presents a smart gateway platform that connects everyday objects such as lights, thermometers, and TVs over the Internet. The proposed hardware architecture is implemented on an Arduino platform with a variety of off-the-shelf home automation technologies such as Zigbee and X10. Using the microcontroller-based platform, the SECE (Sense Everything, Control Everything) system allows users to create various IoT services such as monitoring sensors, controlling actuators, triggering action events, and periodic sensor reporting. We give an overview of the Arduino-based smart gateway architecture and its integration into SECE.**

I. Introduction

We are developing SECE to create services that integrate with location and presence, email, Twitter and Facebook, networked home appliances, and sensors [1], [2]. SECE is an event-driven system which enables users to design on-line services for creating communication flows and controlling physical devices over the Internet. For instance, a user can set up a rule to have SECE extract meeting details from his Google calendar, make a Skype group call from his desk phone, and change his Facebook status to "I'm on a call". Based on the location information from his smartphone, SECE can automatically turn on the home air conditioner just before coming back home from work.

Although recent home gadgets like smart TVs can connect to the Internet using Wi-Fi, we cannot assume that most of the everyday objects (e.g., lights and air conditioners) have such Internet access. To connect the non-networked devices to the Internet, SECE uses the smart gateway based platform. Using a microcontroller-based platform, SECE allows users to operate, monitor, and control sensors and actuators remotely over the Internet.

In this paper, we propose an easy way to build a simple and cost effective home gateway using the Arduino platform [3]. Arduino is an open-source prototyping platform that provides easy-to-use hardware and programming environments. It is relatively inexpensive compared to other microcontroller-based platforms like BeagleBone [4]. In order to prototype a simple home and building automation system, we are developing the Arduino-based platform with various off-the-shelf home automation modules and communication technologies such as Zigbee [5], Bluetooth, and X10 [6]. It is designed for creating simple IoT applications that integrate a variety of sensors and actuators within the SECE system such as monitoring sensors, controlling actuators, triggering action events, and periodic sensor reporting.

The remainder of the paper is organized as follows. The second section of the paper looks at the related work. In Section 3, we describe the implementation. We introduce various IoT applications that control a variety of sensors and actuators over the Internet using the Arduino-based platform in Section 4. Finally, we summarize our experiments in Section 5.

II. Related work

Much work has been done in designing a gateway using a microcontroller-based platform. Zulkifli et al. take advantage of Arduino and XBee [7] to build a wireless heart rate monitoring system for sport training [8]. Rao et al. have implemented a home automation system using the Global System for Mobile communication (GSM) technology to control home appliances from mobile phones [9]. Piyare and Tazil have adopted Bluetooth technology to build a home automation system [10]. Home appliances are enhanced with Arduino to be controlled from mobile phones over the Bluetooth networks. Gill et al. have developed a Zigbee-based home gateway that provides interoperability between Zigbee networks and the Internet for designing a home automation system [11]. Van Der Werff et al. have implemented a mobile-based home automation system [12]. To demonstrate a prototype, they have developed a cellular phone emulator which enables to send SMS messages to home gateways for controlling home appliances.

Most of the proposed smart gateways are not easy to be built, and provide limited capabilities. They use a specific communication technology to monitor sensors or control home appliances. In this paper, we are developing a simple but powerful smart gateway combining an Arduino platform with a variety of off-the-shelf home automation modules and communication protocols such as Zigbee [5], infrared, and X10 [6]. Using the Arduino-based platform, we have implemented IoT applications such as monitoring sensors, controlling actuators, triggering action events, and periodic sensor reporting. Based on the set of applications, SECE will allow users easily to create their own IoT services.

III. Building a smart gateway with Arduino in SECE

In this section, we introduce an Arduino platform for building a smart gateway in SECE. We compare it with other microcontroller-based platforms such as BeagleBone [4] and Raspberry Pi [13], and describe the overall communication between the Arduino and the SECE server.

TABLE I: Microcontroller-based platforms

	Arduino UNO	BeagleBone	Raspberry Pi
Model	R3	Rev A5	Model B
Processor	ATmega328	ARMCortex-A8	ARM11
Clock speed	16 MHz	700 MHz	700 MHz
RAM	2 KB	256 MB	256 MB
Flash	32 KB	4 GB	SD
Min. power	42 mA	170 mA	700 mA
Digital input	14	66	8
Analog input	6	7	N/A
Ethernet	N/A	10/100	10/100
Dev. IDE	Arduino tool	Python,Scratch, Cloud9/Linux	IDLE,Scratch, Squeak/Linux
Cost	$ 29.95	$ 199.95	$ 35.00

A. Arduino platform

Arduino is an open-source single board microcontroller which can be connected to a variety of sensors and actuators for creating interactive objects and environments. The Arduino programming language is based on Wiring [14]. Wiring is an open-source programming framework for microcontrollers. Instead of low-level coding, it has defined a set of abstractions which make it easier to write software. The abstractions are functions and libraries written in C and C++. The Arduino development environment (IDE) is written in Java (based on Processing [15]). The IDE provides a basic debugging environment, and runs the Arduino boot loader to write a program into Flash memory on an Arduino board. The Processing graphics tool and a simplified programming style allow users easily to communicate with an Arduino board from a computer via a serial port. The IDE is available on Linux, Windows, and Mac OS.

B. Other microcontroller-based platforms

In Table I, we briefly compare an Arduino platform with other popular microcontroller-based platforms such as BeagleBone [4] and Raspberry Pi [13]. According to the table above, Arduino and Raspberry Pi are inexpensive, and Raspberry Pi and BeagleBone show powerful performance. When it comes to the cost and the hardware performance, it seems like Raspberry Pi is the best choice at this point. In order to run Raspberry Pi, however, users need to purchase SD cards and external shields for connecting analog sensors to the board. Both Raspberry Pi and BeagleBone run the Linux operating system, which can run multiple applications at the same time, and be programmed in many different languages. On the other hand, Arduino is very simple in design. The Arduino IDEs allow even non-technical users to create sensor applications on the Arduino boards. Compared to BeagleBone and Raspberry Pi, Arduino has the lowest power consumption and provides the simplest way to interface with external devices.

C. Arduino-based platform in SECE

To show the feasibility of building sensor and actuator prototypes in SECE, we select the Arduino UNO. Figure 1 shows the Arduino-based platform in SECE. Many kinds of off-the-shelf sensors, actuators, and communication modules are available for the use of designing the IoT

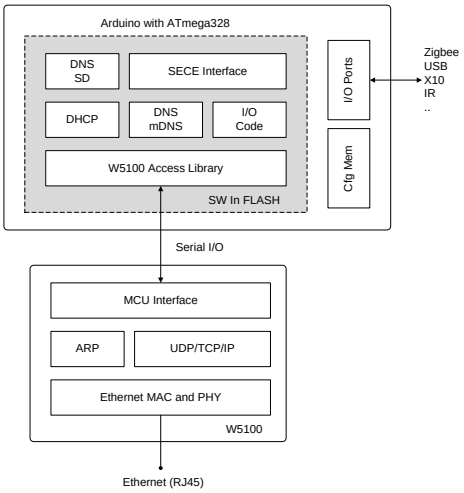

Fig. 1: Arduino-based platform in SECE

services within the SECE system. For example, Phidgets [16] provide a variety of plug and play sensors such as thermometers and motion detectors. X10 [6] modules are used to control electrical devices such as lights, projectors, and TVs over power lines or radio frequencies. We select XBee [7] modules to access Zigbee [5] networks from the Arduino boards. For well-known home automation modules such as Zigbee and X10, the libraries are already available.

In SECE, the Arduino is connected to the SECE server over TCP/IP. We use the Ethernet shield to connect the Arduino board to the Internet with an RJ-45 cable (Figure 2). The standard Internet protocols such as TCP/IP and DHCP are supported in the Arduino IDEs. The Arduino communicates with the SECE server over WebSocket and HTTP. WebSocket establishes bi-directional and full-duplex communication channels over a persistent TCP connection [17]. Once the WebSocket connection is established, a two-way communication channel between the Arduino and the SECE server remains active until the connection has been closed. The property enables Arduinos behind NATs to establish a reliable connection to the SECE server. All communications between the Arduino and the SECE server are done over TCP port number 80, which is of benefit for environments where firewalls do not allow connections to ports other than 80.

HTTP and JavaScript Object Notation (JSON) are used for exchanging messages between the Arduino and the SECE server. JSON is a more light-weight data format than XML [18], and it is easy for human beings to read and write. A simple JSON schema can be used to create a

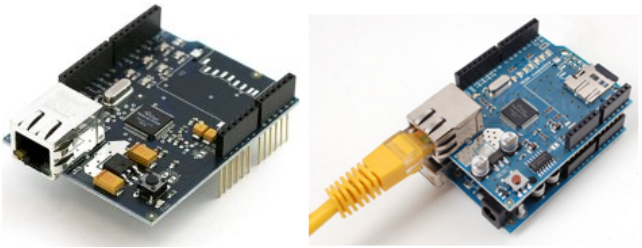

Fig. 2: Arduino Ethernet shield

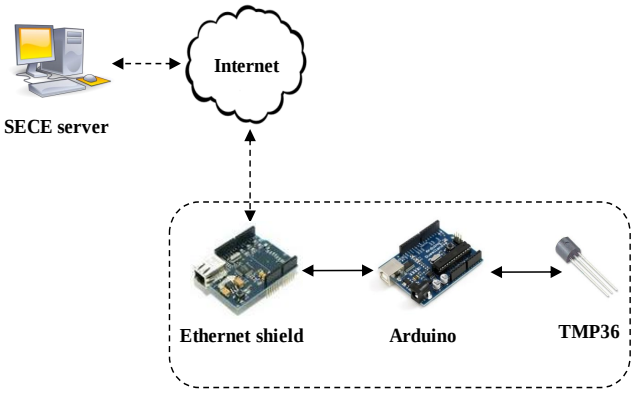

Fig. 3: Monitoring temperature using TMP36

message between the Arduino and the SECE server. As an example, the following JSON message is generated from the Arduino to report the list of the attached sensors and actuators on the Arduino to the SECE server.

```
{ "gateway": "Arduino",
  "id": "7A:62:4B:CD:89:12",
  "devices": [
   { "name": "TV",
     "Pin Number": "D1",
     "type": "digital",
     "communication": "X10",
     "value": "on"},
   { "name": "temperature",
     "Pin Number": "A1",
     "type": "analog",
     "communication": "wired",
     "value": "65"}]}
```

The unique identification of the Arduino consists of the MAC address of the Arduino and the names of the attached devices on the Arduino. It can be used for discovering sensors and actuators in the SECE system.

IV. Connecting the Physical World with Arduino

In this section, we describe a set of experiments to connect a variety of sensors and actuators to the Internet for creating simple IoT services in SECE such as monitoring sensors, controlling actuators, triggering action events, and periodic sensor reporting. For each service, we connect an Arduino board to a different set of sensors and communication technologies such Zigbee, X10, and infrared. Zeroconf enables Arduinos to register services on local networks. To discover the Arduinos, we have implemented a Bonjour application, which also allows users to access the SECE capabilities on the Arduinos over HTTP.

A. Monitoring sensor reading

1) Sensor reading from Arduino: Arduino enables users to monitor various kinds of sensors such as thermometers and motion detectors in real-time. The analog and digital pins on the Arduino board can all serve as general purpose input and output pins (GPIO). The ATmega328 microcontroller embedded on the Arduino board contains the analog-to-digital converter (ADC), which translates the

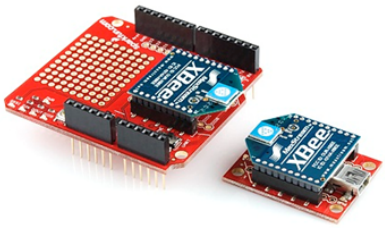

Fig. 4: Arduino XBee shield and dongle

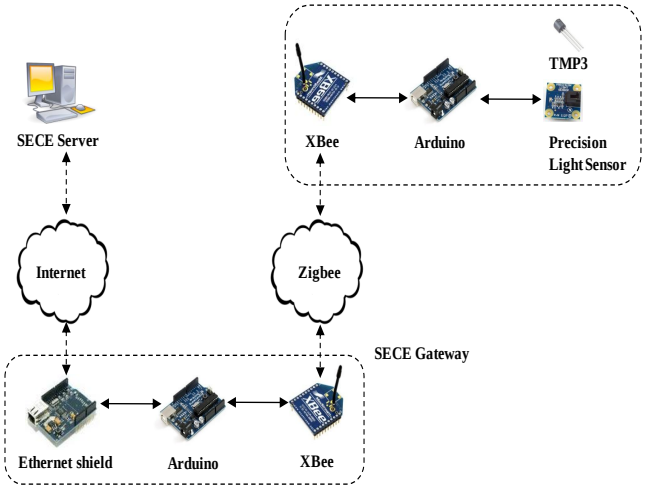

Fig. 5: Monitoring remote sensors over Zigbee

analog input signal to a number between 0 and 1023 [19]. The integer number is proportional to the amount of the voltage being applied to the analog input.

Any sensor operating on 5 volts can be directly connected to the Arduino board. As a prototype for monitoring sensor readings with Arduino, we have implemented a simple setup to connect the analog sensor to the Arduino board, and receive the sensor readings from the SECE server over the Internet (Figure 3). In this experiment, the TMP36 temperature sensor is attached to the Arduino board. The sensor has three pins: ground, signal, and 5 volts. It generates 10 mV per degree centigrade on the signal pin. The math functions in the Arduino IDEs are used to convert the raw values to temperature measures. The SECE sensor monitoring application running on the Arduino converts the temperature data to JSON, and the Ethernet shield enables the Arduino to send the message to the SECE server over the Internet.

2) Remote sensor reading over Zigbee: The XBee radio frequency (RF) modules enable the Arduino boards to access Zigbee networks. The XBee module uses the IEEE 802.15.4 protocol (the basis for Zigbee), and allows very reliable and simple communications between microcontrollers, computers, systems, and really anything with a serial port. We use the Arduino XBee shield (Figure 4) to connect the XBee module to the Arduino board in the SECE system. The XBee antenna on the module can communicate up to 100 feet (indoors) and 300 feet (outdoors) within line-of-sight. XBee libraries are available for the Arduino IDEs [20].

Figure 5 shows a simple setup to monitor remote sensors from the SECE server using the Zigbee protocol. In the

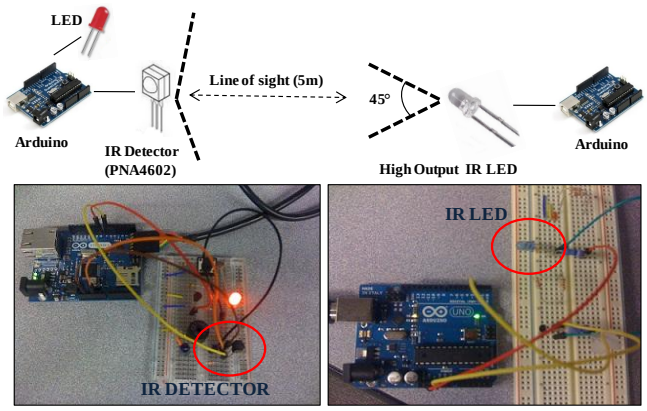

Fig. 6: Connecting an IR detector (PNA4602) and an IR LED to Arduino

experiment, two analog sensors (a TMP36 temperature sensor and a precision light sensor) are wired on the Arduino board.

B. Controlling actuators

1) Infrared signals: The Arduino can decode infrared (IR) signals from remote controls, and generate IR signals for controlling IR-enabled devices such as DVD players and TVs. To build the IR sensor prototype in the SECE system, the PNA4602 IR detector and the high output IR LED are wired on the Arduino board (Figure 6). The IR detector has the peak frequency detection at 38 kHz and the peak LED color at 940 nm. To meet the specification, we have programmed the Arduino to generate an IR signal at 38 kHz using the high output IR LED. By experiments, we determined that the IR detector has a range of 4 to 5 meters.

Most electronic companies comply with the standards of the consumer IR signals (CIR) such as RC-5, RC-6, and NEC [21]. The standard IR signals can be obtained from an on-line IR signal repository [22]. SECE users could provide the model numbers and brand names of their IR-enabled devices on the SECE website. SECE automatically searches and obtains the right set of IR signals from the on-line repository.

2) X10 protocol: The X10 home automation protocol enables the Arduino to control electrical devices over power lines. Almost anything powered by electricity can be directly plugged into X10 modules or, in some cases, wirelessly via 310 MHz in the U.S. and 433 MHz in European systems (Figure 7). Each command consists of four bits, and it represents the X10 module identifications and a set of functions such as on, off, dim, and bright. For instance, the X10 module can cut power to turn the projector off, and adjust the amount of power to dim the light.

Fig. 7: X10 RF (left) and PLC (right) transmitters

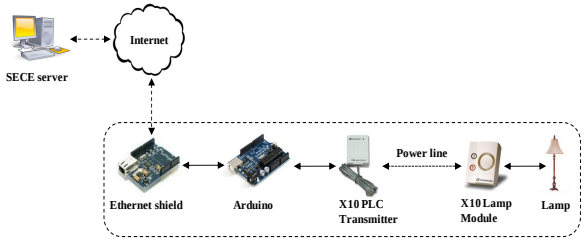

Fig. 8: Controlling devices over power lines using X10

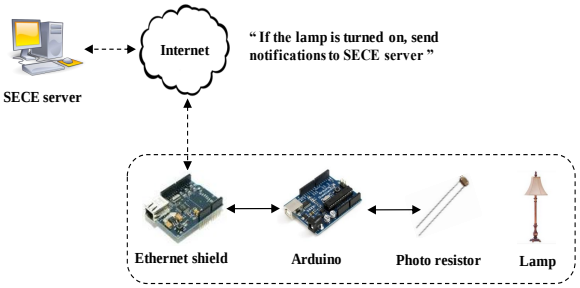

Fig. 9: Event notification system using a photo resistor

As an example of using the X10 modules in the SECE system, we have implemented a simple setup to control a lamp (Figure 8). In the experiment, the X10 power line control (PLC) transmitter is wired on the Arduino board. When the Arduino receives a command from the SECE server over the Internet, it has the X10 PLC transmitter generate the right X10 signals.

C. Triggering action events

Within SECE, the Arduino gateway can trigger actions (e.g., pushing notifications and turning on or off switches) while monitoring sensors in real-time. As shown in Figure 9, we have designed a simple notification system using the Arduino, the Ethernet shield, and the photo resistor. The photo resistor can measure brightness. While reading the sensor data in real-time, the Arduino also compares the value with the predefined threshold. The threshold can be configured through the SECE server over HTTP. If the measurement is above the threshold, it triggers the predefined actions. For instance, it can tweet a message to the SECE users in Twitter saying that "Somebody turned on the light", or turn on the air conditioner in the room.

The finite state machine of the event notification system on the Arduino side is shown in Figure 10. An HTTP PUT message is used to send a notification to the SECE server. An HTTP POST message from the SECE server is used to activate, deactivate, or change parameters of the notification system.

D. Periodic sensor reporting

When polling, the SECE server periodically sends a request to the Arduino to obtain a measurement. The timer function enables the Arduino to deliver sensor readings to the SECE server at regular time intervals. Once the timer is activated on the Arduino, it periodically reports a measurement without receiving any further request from the SECE

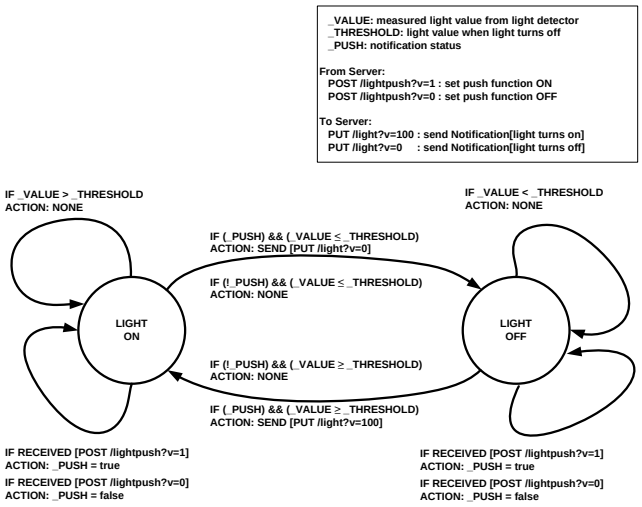

Fig. 10: Finite state machine on the Arduino side for the event notification system

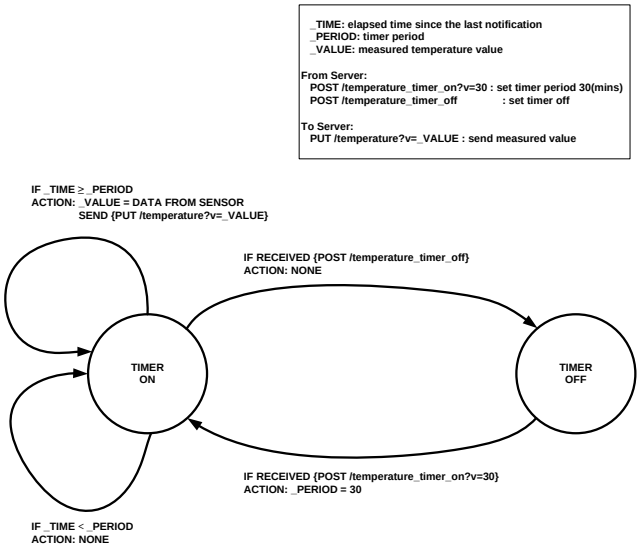

Fig. 11: Finite state machine on the Arduino side for the timer function

server. Unlike triggering action events, the Arduino sends a message to the SECE server even if there has been no change in the connected sensors.

The timer function can be also used to periodically notify the SECE server of the current state of the Arduino such as the network connectivity and the attached sensor information. Figure 11 shows the finite state machine on the Arduino side for implementing the timer function in the SECE system.

E. Discovering Arduinos in local networks using Zeroconf

A small implementation of Zero configuration networking (Zeroconf) can run on IP-enabled Arduinos to register services in local networks [23]. It implements multicast DNS (mDNS) and DNS Service Discovery (DNS-SD). Using the library, we can register a small web-server that provides SECE services running on the Arduino in a local network. The Arduino can be easily discovered by our

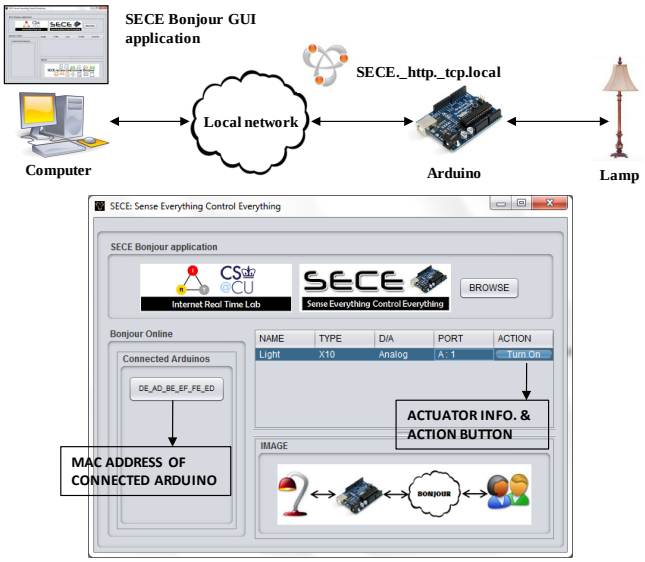

Fig. 12: Connecting to SECE services in local networks using a SECE Bonjour application

SECE Bonjour GUI application running on a computer (Figure 12). It allows users to connect to the SECE service on the local Arduino over HTTP even though the Internet access is not available. The following steps show how the Arduino is registered in a local network, and the application can find and connect to it over HTTP.

a) **Register Arduinos:** The Arduino performs as a web-server. Using the Zeroconf library, it can be registered in the local network with the service (SRV) record: $SECE.\_http.\_tcp.\_local$. The host name contains the MAC address of the Arduino. A text (TXT) record can be used to announce the list of URLs for users to connect to the Arduino over HTTP.

b) **Browse and resolve IP address:** The SECE Bonjour GUI application running on a computer can discover the Arduino using the service record. Then, it sends a DNS query to obtain the host name and port number, and resolve the IP address of the Arduino.

c) **Get service information from Arduinos:** The GUI application uses the obtained IP address and port number to connect to the Arduino. It can get the service information provided by the Arduino by visiting the URL: $GET\ services/list$.

d) **Access to Arduinos:** The GUI application enables users to send a set of commands to the Arduino over HTTP (Figure 12). For example, to turn off the light on the Arduino, a user can push the button on the GUI, which has the application send an HTTP POST message ($POST\ /light?v = 0$) to control the Arduino.

We have also developed a Zeroconf mobile application for Android smartphones to find the local Arduinos that provide SECE services. As shown in Figure 13, the smartphone can discover the Arduinos on the Wi-Fi network, and obtain the list of available SECE services over HTTP.

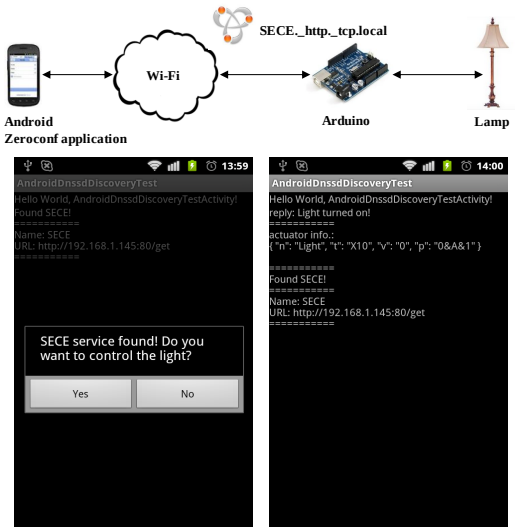

Fig. 13: Connecting to SECE services in local networks from Android smartphones

V. Conclusion

This technical report introduces the Arduino-based platform to control non-IP-capable physical devices over the Internet in SECE. As a selected gateway, an Arduino platform has the following advantages. Compared to other microcontroller-based platforms such as BeagleBone and Raspberry Pi, it is much lighter and simpler weight. For example, the Arduino IDE allows even non-engineers to write various sensing applications with rich library resources. Also, it provides a simple way to interface with a variety of external shields such as Ethernet, Bluetooth, and XBee. The standard Internet protocols such as UDP, TCP/IP, and DHCP are already supported in the Arduino IDEs. Furthermore, it has been shown that many off-the-shelf home automation sensors and communication modules such as X10 and Zigbee are compatible with the Arduino-based platform in SECE.

We have shown that the Arduino-based platform is suitable for designing simple IoT services within the SECE system such as monitoring sensors, controlling actuators, triggering action events, and periodic sensor reporting. Due to the lack of memory resources, it is difficult to run multiple applications at the same time on a single Arduino board. Depending on the service types, we have built multiple single function devices using Arduinos in SECE. We have also developed a SECE Bonjour application which enables users to find Arduinos that provide SECE services in local networks, and access the SECE capabilities over HTTP even though the Internet connection is not available. We expect that even non-technical users can create their own IoT services by following our Arduino-based sensor and actuator prototypes in SECE.

References

[1] O. Boyaci, V. Beltran, and H. Schulzrinne, "Bridging communications and the physical world: Sense Everything, Control Everything," in *Proceedings of the 5th International Conference on Principles, Systems and Applications of IP Telecommunications*, ser. IPTcomm '11. New York, NY, USA: ACM, 2011, pp. 14:1–14:6. [Online]. Available: http://doi.acm.org/10.1145/2124436.2124455

[2] J. Janak, H. Nam, and H. Schulzrinne, "On Access Control in the Internet of Things," in *IAB Workshop on Smart Object Security*, Paris, France, Mar. 2012.

[3] Official Arduino website. [Online]. Available: http://www.arduino.cc

[4] Official BeagleBoard website. [Online]. Available: http://beagleboard.org

[5] Zigbee alliances. [Online]. Available: http://www.zigbee.org

[6] X10 home automation modules. [Online]. Available: http://www.x10.com

[7] Official XBee website. [Online]. Available: http://www.digi.com/xbee

[8] N. S. A. Zulkifli, F. K. C. Harun, and N. S. Azahar, "XBee wireless sensor networks for Heart Rate Monitoring in sport training," in *IEEE International Conference on Biomedical Engineering (ICoBE)*, Penang, Malaysia, Feb. 2012, pp. 441–444. [Online]. Available: http://dx.doi.org/10.1109/icobe.2012.6179054

[9] B. Rao, S. Prasad, and R. Mohan, "A Proto-type for Home Automation using GSM technology," in *IEEE International Conference on Power, Control and Embedded Systems (ICPCES)*, Allahabad, India, Dec. 2010.

[10] R. Piyare and M. Tazil, "Bluetooth based Home Automation System using Cell phone," in *IEEE 15th International Symposium on Consumer Electronics (ISCE)*, Singapore, Singapore, Jun. 2011.

[11] K. Gill, S.-H. Yang, F. Yao, and X. Lu, "A Zigbee-based Home Automation System," *IEEE Trans. on*, vol. 55, no. 2, pp. 422 –430, May 2009.

[12] M. van der Werff, X. Gui, and W. Xu, "A Mobile-based Home Automation System," in *IEEE 2nd International Conference on Mobile Technology, Applications and Systems*, Guangzhou, China, Nov. 2005.

[13] Official Raspberrypi website. [Online]. Available: http://www.raspberrypi.org

[14] Wiring open-source programming framework. [Online]. Available: http://wiring.org.co

[15] Processing open-source visual programming language. [Online]. Available: http://processingjs.org

[16] Phidgets plug and play sensors and actuators. Phidgets Inc. [Online]. Available: http://www.phidgets.com

[17] I. Fette and A. Melnikov, "The WebSocket Protocol," RFC 6455, Dec. 2011.

[18] D. Crockford, "The application/json Media Type for JavaScript Object Notation (JSON)," RFC 4627, Jul. 2006.

[19] Atmega328 - understanding ADC parameters. Atmel Corporation. [Online]. Available: http://www.atmel.com/Images/doc8456.pdf

[20] Arduino library for communicating with XBees in API mode. [Online]. Available: http://code.google.com/p/xbee-arduino/

[21] Data formats for IR remote control. [Online]. Available: http://www.vishay.com/docs/80071/dataform.pdf

[22] Infrared library resource. [Online]. Available: http://lirc.sourceforge.net/remotes

[23] Bonjour/Zeroconf with Arduino. [Online]. Available: http://gkaindl.com/software/arduino-ethernet/bonjour