ASEE's 123rd Annual • Conference & Exposition • New Orleans, LA • June 26-29, 2016

Paper ID #15670

Android Device as a "Programmable" Sensor Module

Dr. Pong P. Chu, Cleveland State University

Android Device as a "Programmable" Sensor Module

Abstract

An Android device is a miniature computer with a touch screen and an array of sensors. There are many used and inexpensive entry-level Android phones and tablets available. In this paper, we describe an innovative method to transform an Android device into a versatile "programmable" I/O sensor module. It uses a serial Bluetooth adaptor and a custom app to obtain the desired sensor measurements and user inputs. The Bluetooth adaptor creates a wireless serial port and the Android app retrieves the desired sensor information, formats the data into a byte stream, and transmits it via the Bluetooth link. The app development is done in the *App Inventor* platform, which is based on a visual programming environment and can be quickly learned. Students can easily develop basic, functional apps and create a customized I/O module that can be incorporated into microcontroller, digital systems, or embedded systems projects.

1. Introduction

1.1 Motivation

The input devices of a computer system are peripherals to take user command, such as switches and keypad, and sensors to measure environmental conditions, such a barometer and an accelerometer. In the computer engineering curriculum, the devices are used in the experiments and projects of many courses, such as digital systems, microcontroller, embedded systems, computer organization, etc. There are wide varieties of input devices. It is difficult to maintain a complete inventory for the lab. In addition, because these input devices are customized for a small and specialized market, they are relatively expensive. For example, a GPS module or a touch sensor module costs more than many processor boards.

An Android device (a phone or a tablet) is a miniature computer with a touch screen and an array of sensors. There is an opportunity to use its sensors as the I/O peripherals. A low-end entry-level device is just "commodity" and cheaper than special I/O modules. Furthermore, since many phones are replaced in a regular basis, there are a larger number of "retired" Android devices (it is reported that more than one billion Android phones were sold in 2014[1]).

1.2 Objective

The objective of this project is to introduce a simple method to use an Android device (including the old phones) as *programmable sensors* and incorporate it into microcontroller, digital systems, or embedded systems projects. Although Android devices are used in many academic settings[2,3,4,5], they are used as general-purpose computers. Our work looks at an Android device from a different perspective. We treat it as *a collection of physical and emulated input sensors* with an embedded processor. The main task of the processor is to retrieve the sensor reading, format the data, and transmit the data via a serial link. Thus, the Android device acts as an I/O module that can be programmed and configured to perform specific I/O

functionalities. It can replace multiple I/O modules, including switches, a force sensor, a keypad, a touchpad, an accelerometer, a GPS module, a real-time clock, an NFC sensor, and a proximity sensor, as demonstrated in Figure 1.

Figure 1. Android device as sensors and input devices

The remaining article is organized as follows: Section 2 discusses a virtual wireless serial port with a Bluetooth adaptor; Section 3 provides an overview of the App Inventor platform; Section 4 discusses the sensors and input methods available from an Android device; Section 5 discusses the usage in a lab setting and provides an example; and last Section summarizes the work.

2. Bluetooth Serial Link

2.1 Overview of I/O interface

A sensor detects events or measures physical conditions in its environment, such as temperature, and provides a corresponding response, usually in a form of an analog electrical signal. A simple sensor module outputs the analog signal directly. We need to condition the signal and then feed it to an ADC (analog-to-digital converter) to digitize the signal, which then can be accessed by a digital circuit or a processor. A more sophisticated module incorporates the conditioning circuit and an ADC within the device and digitizes the signal internally. To reduce the pin count, it also performs the parallel-to-serial conversion and outputs the measurement as a serial data stream. There are several serial interface standards and protocols. The SPI[6] and I2C[7] standards are the most commonly used ones. A UART (universal asynchronous receiver and transmitter) is also used in some peripherals, such as the GPS module and the XBee (for ZigBee protocol) module[8].

The SPI, I2C, and UART ports are standard I/O interfaces in today's microcontrollers and embedded processors. From the processor's perspective, accessing sensor readings corresponds to transfer data from a serial interface.

2.2 Bluetooth serial port

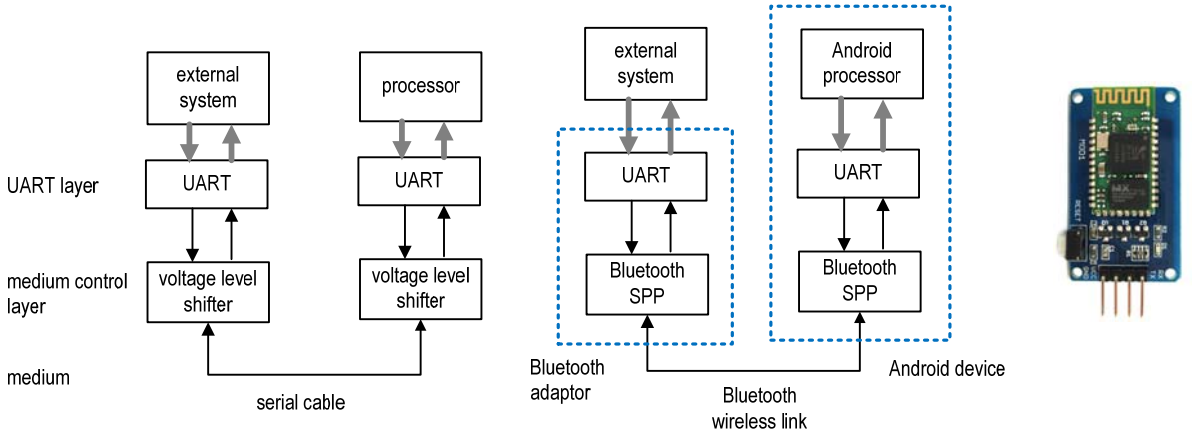
To make an Android device behave like an I/O module, we need to "wrap" it with a compatible serial interface. Since an Android tablet or phone is intended to be a portable and handheld device, its design minimizes the number and type of physical I/O ports. In addition to the phone jack, the only available physical connection is an USB port. Using USB as a general-purpose I/O port is very difficult. First, Android's USB port is normally configured as a salve, appearing as a flash drive when it is connected a PC. In order to exchange information via a USB port, the Android device must function as a *USB host* and the port must support *USB OTG (on the go) mode*. Not all Android devices have this feature. Second, developing software to control the USB OTG is not trivial. The user needs to have in-depth knowledge about the platform and the software skill to access the port. A better alternative for the data access is to establish a *virtual serial port* via a Bluetooth link.

The UART is sometimes referred to as a serial port. Originally, the serial port is associated with the RS-232 standard[9], which formally defines the electrical characteristics and the physical dimension of the connectors. The valid signal voltage level is either in the range of +3 to +15 volts or -3 to -15 volts and the connection requires a thick cable with a large 9-pin D-sub connector. Because of its bulkiness, very few systems support the original RS-232 port now. However, due to the simple and robust implementation, the serial port is still widely used in embedded systems as a communication link and a variety of adaptors are developed to construct an emulated or virtual serial port.

Bluetooth is a wireless communication standard for two devices to exchange data over a short distance[8]. The Bluetooth standard defines a collection of *profiles*, which can be thought as the application layer protocols. The *SPP (serial port profile)* protocol is designed to set up virtual serial ports to connect two Bluetooth enabled devices. The virtual serial port can be best explained by a layered model, as shown in Figure 2. The layers of a "serial-like" port are

- *UART layer*: convert a byte stream to a bit stream.
- *Medium control layer*: control the bit transfer over the physical medium.
- *Physical medium*.

In the original RS-232 based serial port shown in Figure 2(a), the medium control layer is a voltage level shifting device, such as MAX232, which converts the signals to and from the designated voltage levels defined in physical medium, and the physical medium is the RS-232 connector and cable. In the Bluetooth model shown in Figure 2(b), the medium control layer converts the bit stream via the Bluetooth protocol stack and transmits them thought the RF signal. Thus, the Bluetooth SPP protocol can be treated as a *wireless serial port*.



(a). RS-232 serial port

(b). Virtual Bluetooth serial port

(c). Bluetooth adaptor

Figure 2. Original and Bluetooth serial ports

Most Android devices have a built-in Bluetooth module and the Android platform supports the SPP protocol. It can be paired with another Bluetooth device. There are many inexpensive Bluetooth serial adaptors available, as shown Figure 2(c). After pairing with an adaptor, a virtual wireless serial port is established. From the perspective of an external system, the link appears as a normal serial UART port and the data can be transferred and processed following the normal serial protocol. Any digital device with a UART port can access the Android device.

3. App Inventor Android Development Platform

After creating a virtual serial port, we need to create an application program (i.e., an "app") in an Android device to access the sensor measurement and to transmit the data via the Bluetooth link. Android apps are usually developed in Java programming language in conjunction with the *Android SDK* (software development kit)[10]. It requires a substantial amount of programming skills and this topic is typically covered in an upper-level technical elective course. This prerequisite sets a high bar and severely limits the applicability.

A good alternative is the *App Inventor* framework, which is a simple graphic platform to build an Android app[5,11,12]. It was originally provided by Google and is now maintained by MIT. It targets a non-technical audience. The computer science or engineering students can learn the platform quickly and create an app to access the required sensor data.

The development follows the object-oriented principle and contains two basic steps. The first step is to construct one or more cell phone screen layouts in the *Designer* view and to assemble the required ingredients, as demonstrated in Figure 3. App Inventor defines several groups of *components*, which can be thought as Java classes. A class can be a visual subject (such as a button) and placed on screen, or can be a hidden subject, such as the system clock. Each component class is associated with a set of properties, events, and methods. In the Designer view, component instances are instantiated and placed on the screen if visible. Their

properties can also be specified. In Figure 3, the *Palette* column shows the available component classes, the *Viewer* column shows the screen layout, the *Components* column lists the instantiated component instances, and the *Properties* column shows the property of a selected instance. This step creates all the instances needed for the app.

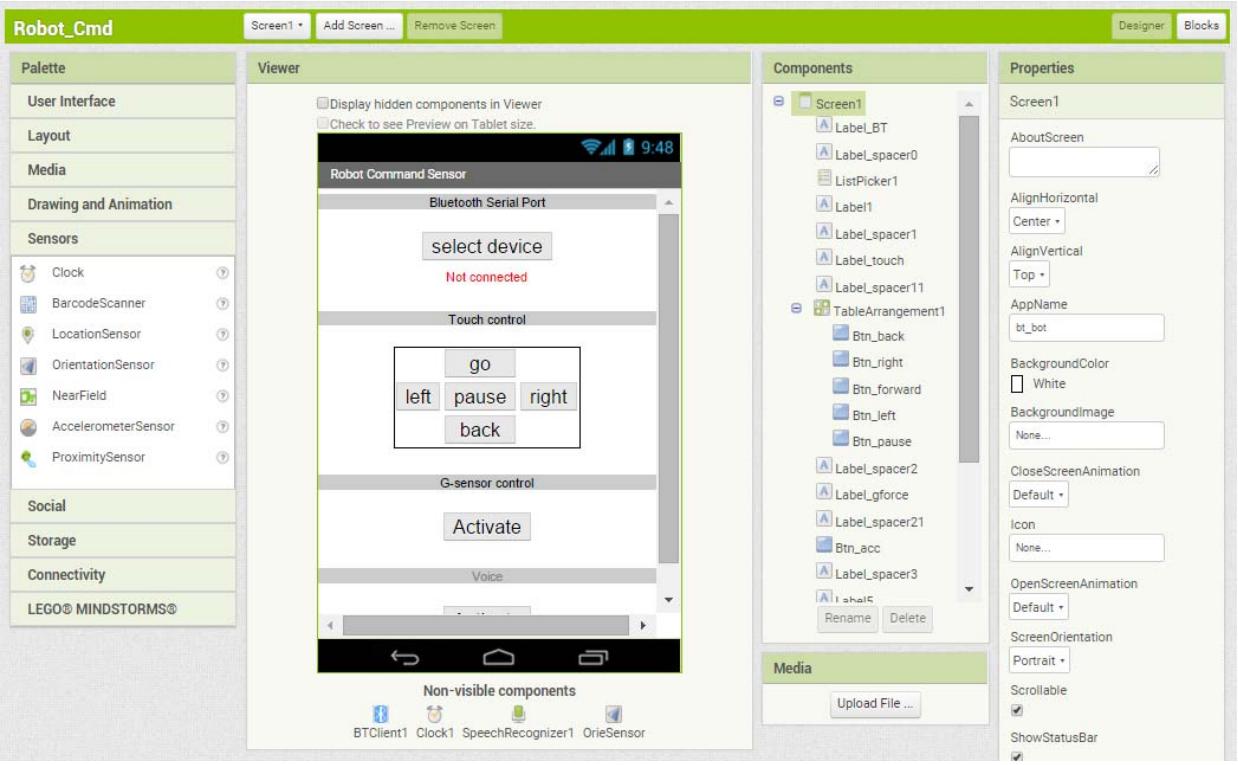


Figure 3. App Inventor "Designer View"

The second step is to develop the app program in the *Block* view, as demonstrated in Figure 4. The app basically defines actions and responses associated with the instantiated components. In App inventor, the language constructs, such as operators and control structures, as well as the properties, methods, and events associated with component classes, are shown as "graphic blocks," which look like small jigsaw pieces. "Deriving app code" becomes the process of dragging and assembling the jigsaw pieces and the finished app resembles a completed jigsaw puzzle.

Despite of its simplicity, App Inventor contains a Bluetooth component class that supports the SPP protocol and various component classes to access the physical sensors and to emulate some commonly used input devices. Since the main purpose of the app is to retrieve the sensor information and transmit the data through the Bluetooth link, the App Inventor platform is adequate for the development.

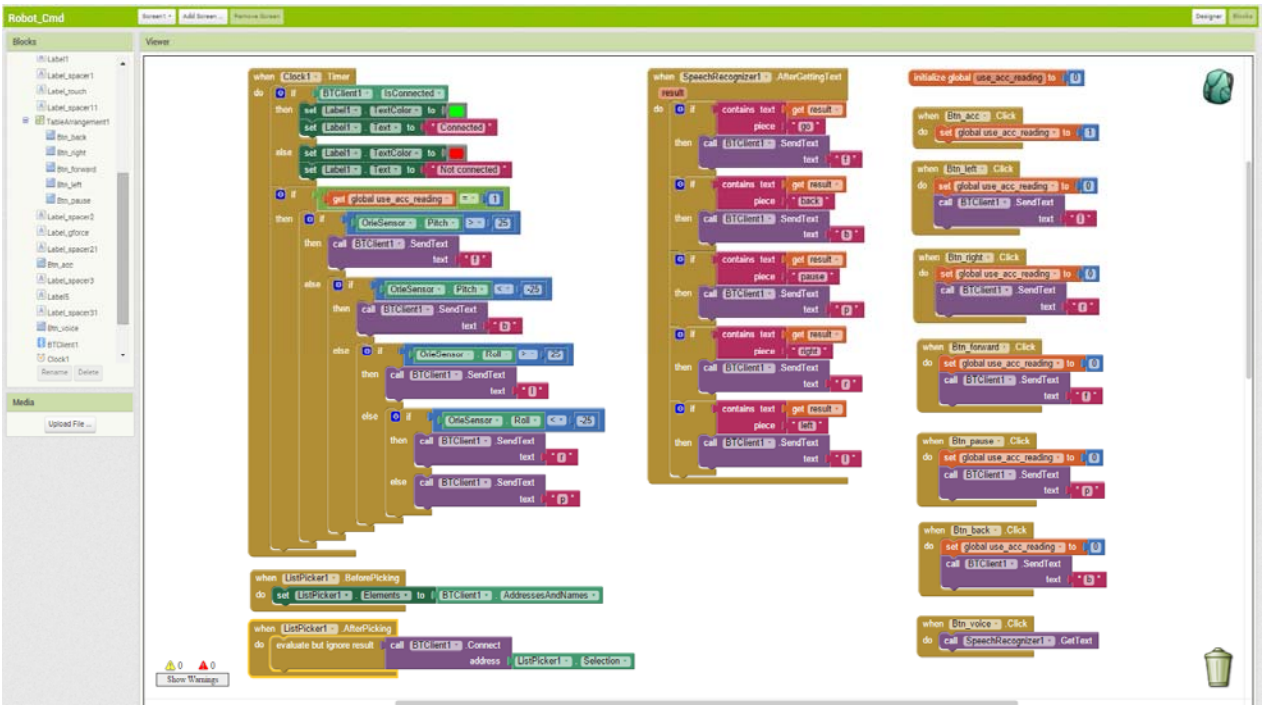


Figure 4. App Inventor "Blocks View"

4. Android Sensors

An Android device can provide a variety of measurements through its built-in sensors or emulation. Since App Inventor is used for the development, our discussion is limited to the measurements that can be accessed or implemented within this platform. We divide the sensor inputs into three categories:

- Obtained by physical sensors.
- Obtained by GUI (graphic user interface) emulation.
- Obtained by advanced Android built-in functionalities.

The details are discussed in the following subsections.

4.1 Physical sensors

An Android device contains an array of physical sensors and their readings and measurements can be accessed and transmitted to an external system. Following are the sensors that can be accessed by the component classes:

- Accelerometer: measure the linear acceleration in three dimensions.
- Real-time clock: keep an internal system clock synchronized to the internet.
- GPS: provide location information, including longitude, latitude, and altitude.
- Proximity sensor: measure the proximity of an object relative to the view screen of a device (usually within a few centimeters).
- Barcode scanner: read the bar code.

- Orientation sensor: determine the spatial orientation (in terms of roll, pitch, and azimuth).
- NFC (near field communication) reader: read NFC text tag.

Note that the orientation sensor is a "software sensor," which uses the measurements from accelerometer, magnetometer, and optional gyroscope to calculate the spatial information. Also, some older and entry-level Android phones do not have the NFC capability.

4.2 **Emulated devices**

The App Inventor platform provides several basic component classes of visual objects, such as a button and a list. They are aimed to create GUI. However, several components can be configured to emulate commonly used input devices:

- *Button* component: emulate a push-button switch.
- *Slider* component: emulate a digitized analog input from a slide potentiometer.
- *Canvas* component: emulate a touch pad.

The button component, as its name shows, is a visual subject that mimics the function of a push button and can detect click activity. It can be used to emulate a physical pushbutton switch and to generate a one-bit digital input. Multiple button components can be instantiated for a multiple-bit input.

The slider component is a progress bar with a draggable thumb, as shown in Figure 5(c). The thumb can be dragged to left or right to a desired position. An event reports the value of the thumb position. It can be used to emulate a digitized potentiometer. A common form of the analog input control is to use a potentiometer to adjust the voltage level and then digitize the analog signal with an ADC, as shown in Figure 5(a). This setup is commonly used to control the volume, set the speed, etc. There are many types of potentiometers and some of them, such as the slide potentiometer and "SoftPot" membrane potentiometer shown in Figure 5(b), change the resistance value based on the linear position. The slider component resembles these linear potentiometers in conjunction with an ADC.

(a). Potentiometer with ADC (b). Slide pot and SoftPot (c). Slider component

Figure 5. Physical and emulated potentiometers

The canvas component is a two-dimensional touch-sensitive rectangular panel on which an object can be drawn and be dragged around. An event reports the x-axis and y-axis coordinate of the object when it is touched or moved, as shown in Figure 6(b). It can be used to emulate a physical touchpad shown in Figure 6(a).

(a). Touch pad (b). Canvas and ball components

Figure 6. Physical and emulated touch pad

4.3 **Advanced input methods**

An Android device provides many features and functionalities and some of them can be adopted for innovative input methods. Following are examples to obtain inputs from the "non-traditional" methods:

- Voice input.
- SMS (short message service, i.e., "text") input.
- Web input.

The App Inventor platform includes Android device's built-in *speech-to-text* functionality. Although it cannot perform natural language processing, the platform provides a method to check the existence of a specific keyword. For example, to use the voice command to control a light, an app can detect whether the keywords "on" and "off" exist in the speech, and activate or deactivate a one-bit output signal accordingly. Thus, when we say "computer, turn off the light," the system will deactivate the signal since the keyword "off" is detected.

An Android phone has a built-in function to handle "text" from another phone. The App Inventor platform has a component class to retrieve the text message. As with the voice input, we can use the predefined keywords to infer specific output values. This implies that we have an "I/O module" that can receive a simple command or input from any phone via texting.

The App Inventor platform provides a web component to access the web service via the standard Hypertext Transfer Protocol (HTTP). That protocol provides Get, Put, and Post methods to bring information into an app. The component is fairly low level and using it requires the web programming skill to parse and extract the needed information[13]. A simple alternative is to set up a *TinyWebDB* compliant web service[14]. It is a primitive web based data base, in which the data is stored in a simple tag-value format. App Inventor has a component class to retrieve the desired data value via a tag. Any system with internet access can update the data value via the designated web service. This implies that we have an "I/O module" that can receive a simple command or input from the internet.

5. Class Usage and Example

5.1 Development

Developing a custom Android I/O module consists of following steps:

- Determine the types of inputs and define the format of the output byte stream.
- Select proper App Inventor components classes and create a GUI screen.
- Extract the sensor or input data and convert them into a byte stream.
- Transmit the byte stream via the Bluetooth serial port.

For an external system, the Android device appears as an input module with a wireless serial port that sends its reading through a byte stream.

5.2 Class Usage

This work is not intended to create a new course or revise the main lab content. Its purpose is to introduce a new innovative peripheral into a microcontroller or embedded system project. It is best suited for student proposed multi-week term projects. Since the App Inventor platform is very simple, the concepts can be covered in one or two recitation sessions. Students can then decide whether to use it in their projects. They may need one or two additional weeks to learn the basic functionalities and derive the needed app. Students can use their own Android devices or the lab can stock a few entry-level Android phones, which can be obtained between $50 and $100. Except for the SMS feature mentioned in Section 4.3, no active phone plan is needed.

This idea has been introduced in an advanced digital systems and senior design classes and several student teams chose to incorporate the device into their projects. Student's responses are very positive and they feel excited to use their smart phones in lab projects.

5.3 Example

One design example is a robot control module that "senses" the input command. The GUI and the "code" are shown in Figures 3 and 4, respectively. The module specifies the movement of a robotic platform and sends five commands, which are 'f' (for moving forward), 'b' (for moving backward), 'l' (for turning left), 'r' (for turning right), and 'p' (for pause). It includes three types of sensing activities:

- Use five push buttons for the five desired actions.
- Use the orientation sensor to obtain the pitch and roll of the device and then translate the reading into the desired action (i.e., controlling the movement by rotating the device horizontally or vertically).
- Use the speech-to-text functionality and detect the keywords of "go," "back," "left," "right," and "pause" for the five desired actions (i.e., controlling the movement by the voice command).

Note that the code shown in Figure 4 is the complete "program listing."

6. Summary

With a Bluetooth module and a custom app, an Android device can be converted into a programmable I/O module with a serial port. The module can be configured (i.e., "programmed") in App Inventor, a graphic development platform. Once the app is running, the Android device

appears as a regular I/O peripheral that transmits the reading via a standard serial port. It does not impose any special hardware requirement on the external system.

This setting can be used in experiments and projects in hardware related courses, such as digital systems, embedded systems, microcontroller, capstone design, etc. It complements the normal I/O devices and introduces the concept of IoT (internet of things).

7. Acknowledgments

This material is based upon work partially supported by the Cleveland State University Undergraduate Summer Research Award Program.

Bibliography

[1] CNET website. http://www.cnet.com/news/android-shipments-exceed-1-billion-for-first-time-in-2014.

[2] H. Abelson and M. Friedman, "App Inventor – A view into learning about computers through building mobile applications," *Proceedings of the 2010 SIGCSE Symposium*, March 2010.

[3] E. Spertus, M. L. Chang, P. Gestwicki, and D. Wolber, "Novel approaches to CS 0 with app inventor for Android," *Proceedings of the 41st ACM technical symposium on Computer science education (SIGCSE '10),* 2010.

[4] J. Gray, H. Abelson, D. Wolber, and M. Friend, "Teaching CS principles with app inventor," *Proceedings of the 50th Annual Southeast Regional Conference (ACM-SE '12),* 2012.

[5] D. Wolber, H. Abelson, M. Friedman, "Democratizing Computing with App Inventor," *ACM GetMobile: Mobile Computing and Communications,* October 2014.

[6] Wikipedia website. https://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus

[7] Wikipedia website. https://en.wikipedia.org/wiki/I2C

[8] R. Faludi,. *Building Wireless Sensor Networks: With Zigbee, Xbee, Arduino, and Processing*, O'Reilly Media, Inc., 2010.

[9] Wikipedia website. https://en.wikipedia.org/wiki/RS-232

[10] B. Phillips, C. Stewart, B. Hardy, and K. Marsicano, *Android Programming: The Big Nerd Ranch Guide, 2nd edition*, 2015.

[11] D. Wolber at. al., *App Inventor 2, 2nd edition*, O'Reilly Media, Inc, 2014.

[12] MIT App Inventor website. http://explore.appinventor.mit.edu

[13] J. N. Robbins, *Learning Web Design: A Beginner's Guide to HTML, CSS, Graphics, and Beyond*. O'Reilly Media, Inc, 2012.

[14] S. R. Madden et. al., "TinyDB: an Acquisitional Query Processing System for Sensor Networks," *ACM Trans. Database System,* March 2005.