BME 484/485 Biomedical Engineering Capstone Design II
Department of Electrical, Computer and Biomedical Engineering
University of Rhode Island, Kingston, RI 02881

Design Progress Report

Designing Android Applications with Voice Activated Bluetooth Control of Household Environments

Fall 2012/Spring 2013

Principal Investigator:
Dr. Ying Sun

Student Assistant:
Aleksey Gladkov

Team Members:

Manager: Nicholas Mulhern <nmulhern@my.uri.edu>
Hardware Engineer: Neil McCaffrey <mccaffrey.n379@gmail.com>
Software Engineer: Nicholas Beretta <nberetta@my.uri.edu >

Date: October 31, 2012

Abstract – **Utilizing assistive technology to aid persons with neuromuscular disabilities, this study improves upon previously created offline voice control of household devices through an input-output peripheral interface control processor driven using an application on the Galaxy S Android smart phone. In this exercise the most effective type of voice recognition software for offline communication was Pocketsphinx. This offline capability is used and tested for accuracy of several commands on a direct television remote though the exchange of Bluetooth signals between the phone and transfer station. The application was designed to recognize keyword commands and send signals to corresponding pins on the input-output board. The execution of a command was enabled through the interface of the smart phone with a television remote. In laboratory experiments were tested for accuracy taking into account distance between person and device as well as level of ambient noise and the vocalization of the subject.**

I. Introduction

The aim of this project is to use assistive technology (AT) to help persons with disabilities. Persons with quadriplegia are affected by limitations in physical function and independence [1]. A field of biomedical engineering, AT is dedicated to increasing the independence and mobility of these persons [2]. In this project, voice recognition is explored as a template upon which the independence of persons with neuromuscular disorders can be expanded. Previous studies have suggested that this provides a viable means for control of devices by persons with disabilities [3, 4]. For this reason, Android applications were developed on a smart phone to operate a television remote via Bluetooth exchange and PIC processing. Offline voice-controlled Android applications were created for accessibility to those with and without wireless fidelity and/or a mobile data plan. The Internet independent application used Carnegie Mellon's PocketSphinx as the application for offline Bluetooth communication [6]. Electronics, in addition to the Bluetooth chip integration; we have harnessed a peripheral interface controller to establish control of the remote using Android application [7].

Project Management

It is important that the concepts, approaches, and designs that each team member makes are well documented and easy to understand. This was particularly important because each team member has individual time commitments making regular meeting detrimental. If for any reason a team member is unable to meet and work on the project as a group, managerial responsibility is to inform the absent team member of any and all information pertaining specifically to the project and its progress. Every meeting or briefing thus far has been recorded and all progress made on the project is documented. Current documentation specifically includes Internet websites found that provided source code, tutorials, material that was purchased, and hardware design schematics. In addition, any debugging that needs to be done and/or challenges faced that may cause a delay in the progress of the project will/have also be well documented. By this, the team should be able to avoid negative progress by running into identical problems in the future. The final objective for this capstone project is to have and present an optimized fully operational prototype by January 31, 2013. In order to reach this goal, it is crucial that the team continues followed the set time schedule (Fig.1) made at the beginning of the fall semester.

Figure 1

Project Timeline

Weeks:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Aim 1	■	■	■																	
Aim 2			■	■	■	■	■													
Aim 3			■	■	■	■	■	■	■	■	■	■	■	■						
Aim 4							■	■	■	■	■	■	■	■	■	■				
Aim 5												■	■	■						
Aim 6														■	■	■	■	■	■	■
Aim 7																		■	■	■

Aim 1: Create objective/gather necessary software and hardware
Aim 2: Test voice recognition program, work with Bluetooth capabilities
Aim 3: Hardware: Detection and Correction of external device flaws
Aim 4: Software modification for optimal use and functionality
Aim 5: Assistive Technology Conference Preparation/ Presentation
Aim 6: Project finalization and presentation NEBEC
Aim 7: Patient interaction

Both the software and hardware engineer are working individually, taking on their independent challenges throughout the project. The software engineer is currently working on understanding and integrating the various source codes that were used in order for the application to read and understand voice command to allow for optimization of the application. The hardware engineer is currently working on establishing a connection between the phone and the remote control via a PIC processor and Bluetooth exchange. It is vital that as project manager, I understand the design approach both engineers are taking to avoid any miscommunication between the two engineers when it comes time to interface the Smartphone to the remote control in the final product.

II. Methods

Software Progress:

Present - Currently, the software that was inherited is being debugged. This involves installing the eclipse IDE to the computer (See Eclipse software) and copying over the code to the new computer. Much of the code is tied to paths from the previous code. Currently the software engineer is finding which paths are not connected properly and fixing these paths.

Future - The goal for the software is to be able to compile the given code. Once the code is running; errors in the code will be able to be fixed. Currently, the major problem with the project is memory storage. The project stores all voiced command without deleting any of them from the internal cache. This causes the internal memory to build up until there is no available space left in turn forcing a crash or program malfunction. To fix this, automatic memory deletion will be integrated into the coding to prevent memory cache overloading.

There are currently three separate approaches the group plans on experimenting with to establish which is most effective. The first method would be to delete the command instantaneously as the most recent command is spoken. The benefit of this would be that there is that no long term memory storage taking place. However, this may affect how the program runs and could force the program to unexpectedly shut down. When the user says, “begin”, the program starts and waits for the next command. If the begin command was instantly deleted once the program accepts it, then the program may shut off since begin is no longer stored in memory. If this theory is accurate in real life testing then this approach would prove ineffective.

The second approach would be to store a certain portion of the received command data (i.e. 3 commands) and then delete them once the limit was met. This should allow for long term memory use without crashing the program and would potentially correct the issue determined in the first method. Ideally the application would receive three commands before deleting the internal memory cache. This would allow the application to use of two separately independent functions to delete the stored commands. The first function could delete all of the commands at the same time. This brings up the same issue as the previous method however, just in a delayed sense of time. For example the phone will store three commands as follows: “begin”, “power”, and “begin”. When the next command comes in the previous “begin” command has already been deleted. If this causes the program to turn off then this will once again prove ineffective in deleting the memory appropriately. Instead the program could be altered to delete the oldest stored command from memory and then slide the remaining stored commands down the line. This should provide space for the new command in memory without deleting any potentially important commands.

The intended goal of this project is to optimize the function of the application and accompanying remote device. Currently the project allows the user to communicate with their TV. It has the ability to control power, volume up and down, and channel up and down. The hopes are to take this project to the next level in user interaction. In terms of the TV, potentially add more voice commanded button functions by extending the library of voice commands. In terms of the user environment there are endless possibilities in using the Bluetooth capabilities to connect with other devices such as lights and fans/AC units.

Eclipse (software):

It quickly became a vital part of software development to install and understand the eclipse tool onto a personal computer. Eclipse is a free android software development environment that works with mainly java base codding. The following steps were taken to initially begin our Android research and development using the available Eclipse (software).

Development Tools plugin and Android Starter Development kit:

1. Prepare the computer you wish to use for development by confirming that it meets System Requirements: they can be found at
   http://developer.android.com/sdk/requirements.html

   For example, our team used Windows for development of the Online application and found that Windows 7 and Eclipse 3.6.2 (or higher versions) with the Java Software Engineering Development Tool (JDT) plugin were appropriate for development.

   The team used Linux for the development of the Offline application. The operating system, Ubuntu Linux, version 8.04 or later was required along with the same Eclipse requirements.

   If eclipse is not downloaded, it can be downloaded from:
   http://www.eclipse.org/downloads/

   This site will automatically give you options for your particular operating system, and options will be represented symbolically with green arrows pointing downward onto a horizontal green bar (symbol for download). i.e. if you have windows, there will be options for Windows 32 bit and Windows 64 bit next to green download arrows. Go ahead and download Eclipse Classic in 64 bit (if your computer is 64 bit), making sure to choose a version that is 3.6.2 or higher.
2. Download the Software Development Kit (SDK) Starter Package
   http://developer.android.com/sdk/index.html
   The correct package will be listed for your particular operating system. If there is more than one option, choose the package that is "recommended."
   If downloading in Windows, run the Windows installer (.exe file). This will check whether the proper Java SE Development Kit (JDK) is installed, then install the SDK tools into a default location.
   Note, if downloading in Linux, you will download a .zip or .tgz package, unpack it to a safe place (the default is android-sdk-<machine-platform>).
3. Install the Android Development Tools (ADT) Plugin for Eclipse. This is just a custom plugin for the Eclipse IDE. The Update Manager feature of your Eclipse installation can be used to install the latest revision of ADT on your development computer.

a. Start Eclipse, select Help > Install New Software
b. Click Add (top right corner)
c. In the 'Add Repository' dialog, enter 'ADT Plugin' for the name with
   The URL 'https://dl-ssl.google.com/android/eclipse/
d. Click OK
e. In the Available Software dialog, select the check box next to Developer Tools, click Next
f. You should see a list of tools to be downloaded, click Next
g. Read and accept the license agreements, then click Finish
h. Restart Eclipse

4. Configure the ADT plugin (this will modify the ADT preferences in Eclipse to point to the Android SDK directory)

a. Select Windows > Preferences
b. Select Android from the left panel
   (if Google asks to send your usage statistics, make your choice and click proceed before continuing)

c. For the SDK location in the main panel, click Browse and locate your downloaded SDK directory

(note: if you download the SDK in windows 64 bit, the automatic location for program files will have a space in the directory. Eclipse has a hard time recognizing locations with spaces, so you will need to copy and paste the SDK information to another location on your computer without a space. Then, when you go to configure the ADT plugin, browse to this new location, without a space.)

d. Click Apply, then OK

5. The last step in setting up the SDK is to use the Android SDK and AVD manager (included in the SDK startup package) to download essential SDK components for Android App Development.

a. If using windows, the Windows Installer will launch the Android SDK and AVD manager with a default set of platforms and other components automatically selected. This selection is good enough for now, go ahead and click Install. You are done with this step. The Android SDK and AVD manager can be executed by double-clicking the SDKManager.exe file in the Android SDK directory

b. If using Linux, open a terminal, navigate to the tools/ directory of the Android SDK and execute 'android.'

c. You can choose which components to download by checking them in the graphical user interface (GUI) that opens. Especially make sure that the samples are open.

6. Now, make sure that everything has installed correctly and your development environment is properly set up by running a few sample programs.

a. Install a Platform

a.i. In the SDK and AVD manager, choose Avaliable Packages in the Left panel

a.ii. In the right panel expand the Android Repository list to display the components available for installation

a.iii. Select at least one platform to install (may be helpful to choose one that matches the phone you plan to develop on).

b. Create an Android Virtual Device (AVD). This basically is a computer simulation of your android phone.

b.i. In Eclipse, select Window > Android SDK and AVD Manager

b.ii. Select Virtual Devices in the Left Panel

b.iii. Click New and the "Create New AVD" dialog will appear

b.iv. Type the desired name for your AVD (for example, "John_Smith")

b.v. Choose a target (i.e. platform, i.e. the version of Android you want to run on your AVD)

b.vi. Click Create AVD

c. Open up a sample program

c.i. To download samples, launch the Android SDK and AVD manager tool

c.ii. Select desired sample components from the Avaliable Packages panel

c.iii. For this project, make sure to select "Samples for SDK API 7" or higher. We will later be combining sample applications to produce the offline and online voice recognition.

c.iv. Select Install Selected, verify and accept the download,

c.v. Select Install Accepted

c.vi. The sample applications can be found under <sdk>/samples/android-<level>/

c.vii. Click New > Project for the API Demos

c.viii. Select "Create Project from existing source"

c.ix. Browse until you find a sample application c.x. Click the green arrow right facing arrow in Eclipse to run the program

c.xi. A virtual device should show up on your screen, you can slide the cursor to unlock the phone and the new app you have created should show up on the phone. Click it to run the app.

(Note: If you want to install the actual application on your development phone, connect the phone to the computer via a micro-usb cable. You will be prompted about your intended target device, just check off your phone and from now on all new compilations will occur on your phone, not on the AVD.)

(See reference for more details on installing and configuring these programs [8].)

Hardware Progress:

Bill Of Materials

Manufacturer #	Description	Ordered From	Ro HS
PIC18f4525-I/P	PIC Microcontroller	Microchip Technology Inc	Yes
WRL-10269	Bluetooth Modem	Sparkfun Electronics	Yes
L78M05CV	Voltage Regulator (5 Volt)	Mouser Electronics	Yes
MC14066BCP	Quad Bilateral Switches	Mouser Electronics	No
CF1/4W105JRC	1 M Ohm resistor	Jameco Electronics	Yes
CF1/4W103JRC	10 K Ohm resistor	Jameco Electronics	Yes
FCR4.0MC5	4 MHz Ceramic Resonator	TDK Electronics	Yes
DP J-021A-R	Power Connector Male	Jameco Electronics	Yes
DBU060020H4510	AC/DC Power Supply	Jameco Electronics	No
28RC16-10VP	Ribbon Cable	Jameco Electronics	Yes

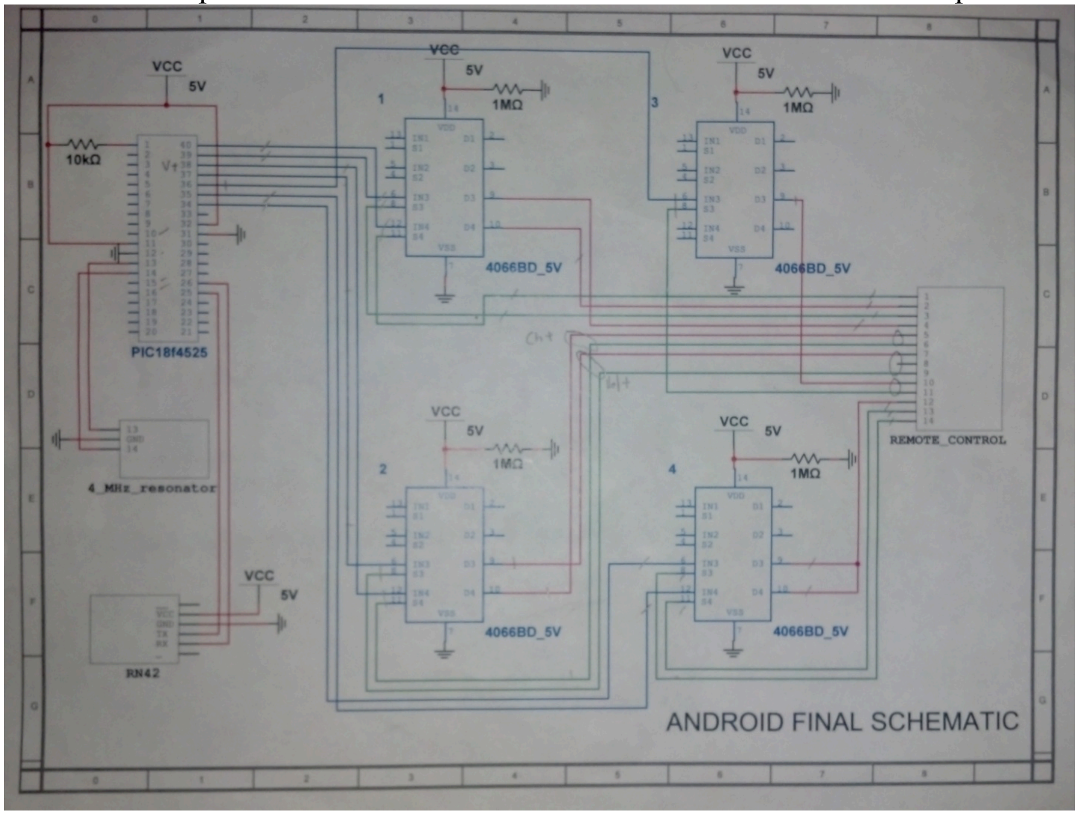
Present –An AC/DC power supply delivers 6 volts DC 200mA via a male power connector to a voltage regulator which supplies the entire circuit with 5 volts. The degree of shock protection is Class I according to the standards of section 6.2 of IEC 60601-1. All the connections for 5 volts and ground as well as the connections between all the components are included in the appendix for reference.

The hardware implements a Bluetooth modem, BlueSMiRF sliver, which receives the commands from the android application software that was developed for the smart phone. The BlueSMiRF modem communicates with the PIC ® Microcontroller via the EUSART transmit and receive pins on the microcontroller, pins 25 and 26, respectively. Each function in the android application recognizes keywords which then send a specific signal via Bluetooth connection to the microcontroller. A simple code in the microcontroller, which is included in the appendix, interprets the signals received from the BlueSMiRF and sends a corresponding signal out of one of the seven outputs of port B to the quad bilateral switches.

In order to trigger specific buttons on the Direct TV remote control, reverse engineering was used by soldering wires from a ribbon cable to the specific button locations inside the remote. Each remote button has two corresponding wires, which when connected, complete the circuit and trigger the button on the remote control. The connections were completed using quad bilateral switches that are triggered by the port B output of the microcontroller. In order to prevent the switches from floating, a 1MΩ resistor was added to ground the 5 volt input.

In order for the device to work as expected, there is some calibration that is necessary. First, the Direct TV remote must be programmed for the patients TV, which is can be done by looking up the code on the direct TV website. When the correct code is programmed, each command in the android application needs to be tested for functionality. Every television has a different sensitivity to the length of the signal received from the remote. If the system is not calibrated correctly, the length of the signal could be too long which would result in changing ten channels instead of only changing one. The length of the signal transmitted from the remote is determined by the code in the microcontroller and by changing this duration the system will be calibrated to work on the patient's television.

Future – The goal of the hardware is to possibly add a few new connections in order to add more accessibility to the TV. If the patient does have Direct TV, then it would be worth connecting the circuit to the "guide" button on the controller in order to access the TV guide. If this is an option, then connections to the up and down arrows as well as the select button are also required. If however the patient does not have direct TV then this would not be an option.



```
/*********************************************************************************/
/* Bluetooth Send and Receive Example                                            */
/* Instructors: Prof. Ying Sun                                                   */
/* Contributors: Kyle Rafferty, Vanessa Landes, Melissa Andrews, Christina Liese */
/* Last update: June 6, 2012                                                     */
```

```
/********************************************************************************/

/********************* Specify the chip that we are using *************************/
#pragma chip PIC18f4525

/************************* Define prototype functions ****************************/
void delay_ms(uns16 x);

#pragma origin 0x8

void delay_ms(uns16 x) /* generate a delay for x ms, assuming 4 MHz clock **/
{
unsigned char y;
        for(;x > 0; x--) for(y=0; y< 165;y++);
}

// for bluetooth
void Setup_USART_Receive () {
        TRISC = 0b.1100.0000;
        SPBRG = 25;
        BRG16 = 0;
        BRGH = 1;
        SYNC = 0;
        SPEN = 1;
        CREN = 1;
}

// for bluetooth
void Setup_USART_Transmit () {
        TRISC = 0b.1100.0000;
        SPBRG = 8;
        BRG16 = 1;
        BRGH = 1;
        TXEN = 1;
        SYNC = 0;
        SPEN = 1;
        CREN = 1;
}

void main() /************************ main program ******************************/
{
        TRISB = 0b.0000.0000;           // Set all port D pins as outputs (connected to D/A)
        T0CON = 0b.1000.1000;           // Setup the timer control register
        // Setup timer interrupt support
        // bit 7 = GIE - Global Interrupt Enable
        // bit 5 = TMROIE - Timer 0 Overflow Interrupt Enable
        // bit 2 = TMR0IF - Timer 0 Interrupt Flag

        //for bluetooth (setup)
        //TRISC = 0b.1100.0000; // enable RX and TX for USART
        //RCSTA = 0b.1011.0000; // enable bit SPEN (serial port bit enable) for USART, also SREN and CREN
        Setup_USART_Receive();
        Setup_USART_Transmit();

        INTCON = 0b.1010.0000;
        PORTB = 0b.0000.0000;

//      bit pflag, oflag, wflag, eflag, rflag;
//      pflag = oflag = wflag = eflag = rflag = 0;

        while(1) {                      // forever loop
                PORTB = 0b.0000.0000;

        //      delay_ms(10);
                if (RCIF == 1) {
                        if (TXIF == 1){
                                TXREG = RCREG;
                                if(TXREG == 0x41){ // A
                                        PORTB.7 = 1;
                                        delay_ms(2000);
```

```
                                                }
                                                if(TXREG == 0x42){ // B
                                                        PORTB.6 = 1;
                                                        delay_ms(2000);
                                                }
                                                if(TXREG == 0x43){ // C
                                                        PORTB.5 = 1;
                                                        delay_ms(2000);
                                                }
                                                if(TXREG == 0x44){ // D
                                                        PORTB.4 = 1;
                                                        delay_ms(2000);
                                                }
                                                if(TXREG == 0x45){ // E
                                                        PORTB.3 = 1;
                                                        delay_ms(2000);
                                                }
                                                if(TXREG == 0x46){ // F
                                                        PORTB.2 = 1;
                                                        delay_ms(2000);
                                                }
                                                if(TXREG == 0x47){ // G
                                                        PORTB.1 = 1;
                                                        delay_ms(2000);
                                                }


                                        }
                                }
                        }
                }
```

III. Design

The voice activated switch design project was originally presented by Dr. Sun as a first hand application of assistive technology to his students. The hopes have been that the group could perfect the project and hopefully present this to his friend Tom by the end of the capstone design program. Tom is a longtime friend of Dr. Suns who has muscular dystrophy. Tom is slowly losing his ability to use his hands to complete simple tasks while at his home. To accommodate to this real world application has altered the design process significantly to allow for the voice activated switch to serve a purpose in helping tom during his everyday life. The participant has provided the team with many real world challenges that the team has had to overcome to allow practical use.

Fortunately the original project was passed on to us by the previous capstone team with a lot of the requested alterations which were well documented providing a solid base to work from. The project was originally created to work with Google though a wirelessly connected internet. Due to the participants current environmental situation we have had to recreate and correct the offline Bluetooth system to allow short wave-length transmissions from the phone to the remote/Bluetooth hub. This remote/ Bluetooth transfer hub has also been modified to run off of AC power to provide users with the most practical environmental control and superior sustainably.

The social aspect of the project has also played a huge part in how the team thinks about design development. Socially choosing to use an android smartphone to help in the world of assistive technology has been a very progressive approach. Smart phones are rapidly sneaking into the everyday lives of many people around the world. Though the project is currently being created with a specific individual in mind, the potential to market this technology to other smartphone users is endless. With the ability to simply publish this application to the android marketplace creates the ability to spread the technology to users very quickly and economically. Its factors like these which are helping to mold our design process into creating something with proficient manufacturability in real world application. Keeping in mind that the current design is functional without needing wireless internet or a mobile data plan.

Other key factors which play a role in the design of this voice activate switch are obviously the health and safety factors. The team has determined that the device meets many if not all of the critical engineering standard, and the current degree of shock protection is Class I according to the standards of section 6.2 of IEC 60601-1documentation which pertains specifically to electrical equipment. As a team it was decided to research any health issues which may create issues during actual use within the laboratory and with the participant in his home environment. The way with which the Bluetooth Hub has been designed contains all essential hardware and wiring within a closed box though it does receive AC power from a standard 110V outlet. This box has been sealed with four external facets to prevent any potential electrical tampering during use. It has been determined that based on the extremely low possibility of death or serious injury resulting from an American household shock that the device should be safe for laboratory work. More extensive research will be pursued in an effort to ensure the safest design possible before presenting the final device to the participant.

The current design of the voice activated switch begins with an Android Galaxy S smartphone which is complete with fully functional Bluetooth capabilities. Once the voice application is installed on the device we must locate the AC powered Bluetooth Input-output transfer Hub in the discoverable device options. The internal hardware in the Hub is composed of a Bluetooth chip to receive the phones Bluetooth exchange. It also has a programmed PIC processor which helps to complete specific commands on the hardwired remote control.

The design of the software has been quite the challenge due mainly to the participant's ability to vocalize. The participant is slowly losing the ability to use his vocal muscles due to his degenerative disease. This issue keeps his voice down to a very soft tone and limits the duration of his sentences. The commands chosen by the team were applied so that they were short and simple enough to be said without fatigue. One factor of his environment which is working to our benefit during the design process is the fact that there is a lot more background noise present in the laboratory then at his home. It has been assumed by the team that as long as the device and software design has been tested within an environment with more variables then it should work flawlessly during actual application with the participant.

References

[1] K.E. Chad and P.J. Manns, "Components of quality of life for persons with a quadriplegic and paraplegic spinal cord injury," *Qualitative Health Research. Sage Publications*, vol. 11 no. 6, pp. 795-811, 1 Nov. 2001.
[2] G.J. Gelderblom and L.P De Witte. "The Assessment of Assistive Technology Outcomes, Effects and Costs." *IOS Press. Technology and Disability* 4, 91-94, 2002.

[3] J.M. Noyes and C.R. Frankish, “Speech recognition technology for individuals with disabilities,” *ISAAC.* vol. 8, December 1992.
[4] J.M. Noyes, R. Haigh, and A.F. Starr, “Automatic speech recognition for disabled people,” *ScienceDirect,* vol. 20, pp. 293-298, December 1989.
[5] IOIO for Android, *Sparkfun Electronics.* http://www.sparkfun.com/ products/10748. Jan. 17, 2012.
[6] Huggins-Daines, D.; Kumar, M.; Chan, A.; Black, A.W.; Ravishankar, M.; Rudnicky, A.I.; , "Pocketsphinx: A Free, Real-Time Continuous Speech Recognition System for Hand-Held Devices," *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on* , vol.1, no., pp.I, 14-19 May 2006.
[7] ^ *a* *b* "1977 Data Catalog", Micro Electronics from General Instrument Corporation http://www.rhoent.com/pic16xx.pdf
[8] Installing the SDK, *Android Developers.* http://developer.android.com/ sdk/installing.html. Jan. 15, 2012.
[9] Android Application Development Tutorial- 186 – Voice Recognition Result, The New Boston, *YouTube.* http://www.youtube.com/watch?v=8_XW_5JDxXI. Oct. 2011.