Emotional Intelligence Device

By

Jonathan Fouk

Matt Palmer

Vivian Tseng

Final Report for ECE 445, Senior Design, Spring 2015

TA: Jacob Brian

6 May 2015

Project No. 41

Abstract

Emotion detection was intended to be done with a combination of speech, heart rate, and skin temperature sensors. The device was envisioned to be wearable and provide programmable text output to the user via LCD screen as well as LED indicators for 4 emotional categories. Successes of the project include meeting the design requirements for microphone, lithium-ion battery charging circuit, LCD screen, LED lights, and speech accuracy. Design requirements were not met for SPI communication between microcontroller and DSP chip, heart rate detection, implementation of speech algorithm/system classifier on chips, and overall device size.

Contents

1. Introduction 1

1.1 Statement of purpose 1

2. Design 1

2.1 Block Diagrams 1

2.2 Data Collection 2

2.2.1 Microphone 2

2.2.2 Temperature 6

2.2.3 Heart Rate Monitor 7

2.3 User Interface 7

2.3.1 LCD Display 7

2.3.2 LED light indicator 7

2.3.3 Switch 7

2.4 Power Systems 7

2.5 Microcontroller and Processing 9

2.6 Emotion Classification 10

2.6.1 Data Set 11

2.6.2 Feature Extraction 11

2.6.3 Support Vector Machines 13

2.6.4 Classification 15

3. Design Verification 16

3.1 Data Collection 16

3.1.1 Microphone 16

3.1.2 Temperature 16

3.1.3 Heart Rate monitor 16

3.2 User Interface 16

3.2.1 LCD Display 16

3.2.2 LED Light Indicator 17

3.2.3 Switch 17

3.3 Power Systems 17

3.4 Microcontroller and Processing 17

3.5 Emotion Classification 17

4. Costs 18
4.1 Parts 18
4.2 Labor 19
4.3 Grand Total 19
5. Conclusion 19
5.1 Accomplishments 19
5.2 Uncertainties 19
5.3 Ethical considerations 19
5.4 Future work 20
5.4.1 Emotion Classification 20
5.4.2 Interfacing between modules 20
References 21
Appendix A Requirement and Verification Table 23
Appendix B Detailed Parts Cost Table 25
Appendix C Circuit Schematics 26
Appendix D PCB Layout 30
Appendix E Codes 30

1. Introduction

1.1 Statement of purpose

We live in a digital world where people can become lost in their devices, decreasing the daily amount of personable social interaction. As a result, their communication skills deteriorate; specifically, their ability to identify and express their own emotions in a constructive manner becomes more difficult. There are also people with medical disorders that are unable to identify emotions in their peers, such as people with autism.

To address this problem, we designed a wearable device that will help people identify their emotions, give the option to display their emotions, and provide constructive suggestions to the users to help alleviate stressful social situations. Our device aims to help someone explore the root cause of their feelings as well as offer the ability to display his/hers broad emotional state to help others identify with him/her. This concept would allow people to effectively communicate their emotional state, leading to less conflict on a daily basis and a better quality of life for all involved.

2. Design

2.1 Block Diagrams

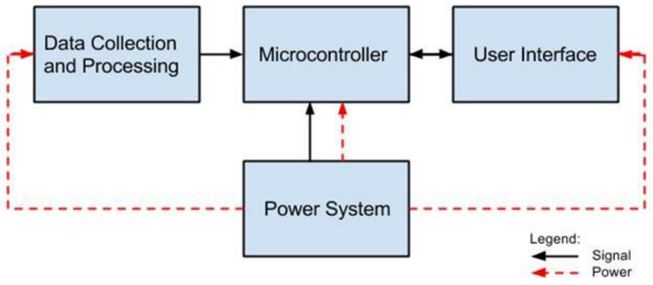


Figure 1. High Level Block Diagram for System

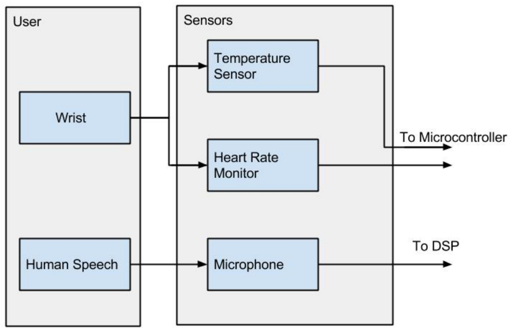


Figure 2. Block Diagram Data Collection

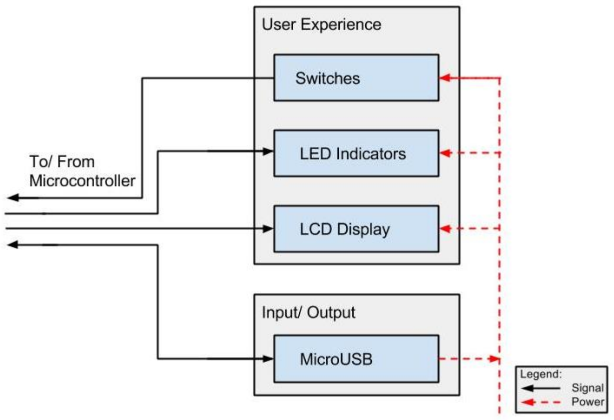


Figure 3. Block Diagram User Interface

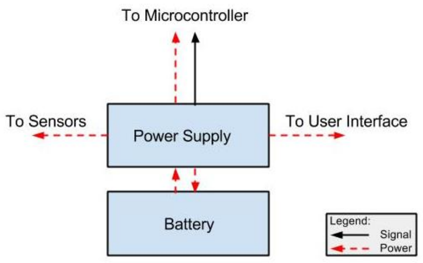


Figure 4. Block Diagram Power Supply

2.2 Data Collection

2.2.1 Microphone

The purpose of the microphone circuit is to record the speech, amplify the speech, and then pass it through a filter. Table 1 lists the defining output characteristics of our microphone, the ICS-40300 from its datasheet in [1]. From Table 1, the 94 dB SPL corresponds to the Vrms of speech spoken directly into the microphone. Converting to V_{pp}, that is approximately 20 mV_{pp}. Given that the DSPs have a 12 bit ADC converter with 17 channels, the equation for finding the bit resolution is listed in Equation 1.

$$\Delta V_{res} = \frac{V_{pp}}{2^n - 1} = \frac{2}{2^{12} - 1} \approx 0.5mV \tag{1}$$

where ΔV_{res} is the voltage step a bit respresents, V_{pp} is the largest voltage peak-to-peak the ADC is set to analyze, and n is the number of bits dedicated to the ADC. Utilizing this resolution and the maximum V_{pp} amplitude of a person speaking into the microphone, we only use approximately 5 bits out of the 12 to resolve the voice. An amplifier is required to increase the number of bits we can use to represent the speech signal. From the datasheet in [2] a typical amplifier is suggested with an OP-AMP shown in

Table 1. Output Information for ICS-40300 Microphone

Output Impedance	200 Ω
Output DC Offset	0.8 V
1 kHz 94 dB SPL Maximum	-43 dBV

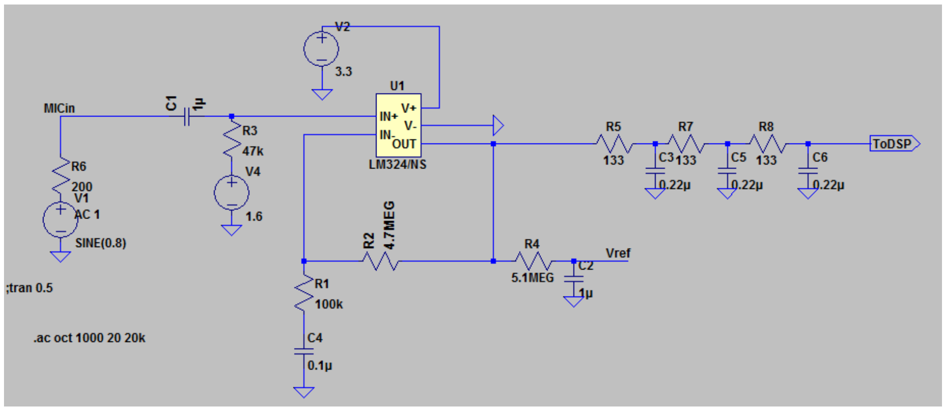


Figure 5. . LTSpice Mic + OP-AMP Simulator and Circuit Diagram

Figure 5, but it requires a decoupling capacitor grounded to replace the V_{ref} on IN_- stage. The gain can be approximated as shown in Equation 2. The chosen OP-AMP for this application was the LM321 which is a single OP-AMP version of the LM324/NS, which has small phase noise and operates at 3.3 V. It's datasheet is listed in [2].

$$Gain = \frac{R_1 + R_2}{R_1} = \frac{100 \pm 5 + 4700 \pm 235}{100 \pm 5} k\Omega = 48 \pm 5V/V \tag{2}$$

where R1 and R2 correspond to the resistors R1 and R2 shown in Figure 5. The goal was to have an amplification of approximately 50 due to the maximum speech signal being 20 mV_{pp}. The output at 60 Hz is shown in Figure 6, where Vref is the voltage reference and MIC_{IN} is the simulated microphone signal at 60 Hz.

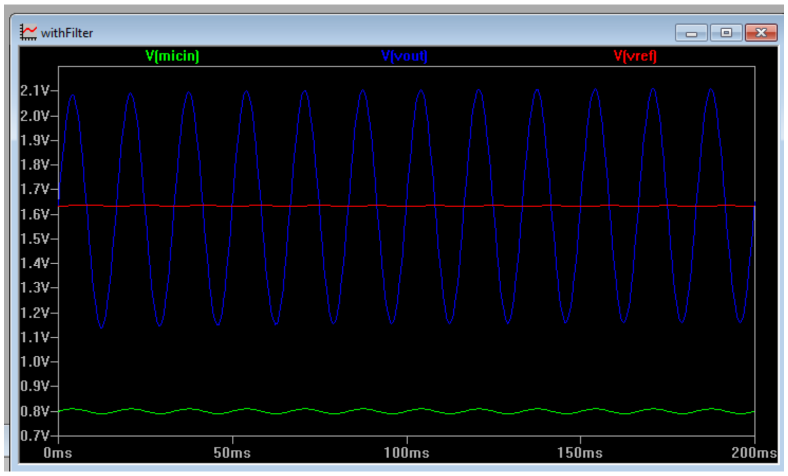


Figure 6. LTSpice simulation of Figure 5 LTSpice Schematic with Green as the micin signal, blue as Vout, and red as Vref.

It is important to get the DC offset from the microphone circuit such that it is approximately equal the reference voltage for the DSP. The reference voltage can be set using an input pin on the DSP chips, which are depicted in the datasheet in [3].

In order to reduce the amount of computational responsibility of the DSPs, an analog filter was implemented to reduce the DSP computational requirement. An analog implementation of MFCC filters was investigated by [4], but required eight 4th order bi-quad filters or a custom IC chip to implement. The bi-quad filters required too much area which resulted in the LPF being the only analog option to decrease computational responsibility of the DSP, which is a third order LPF from OUT port to V_{out} shown in Figure 5.

The first aspect of the filter that was considered is the impedance seen by the ADC12. According to microchip's ADC guide in [5] for the 12 bit converter a maximum value of 2.5 kΩ should be used as the input impedance into the DSP but to meet our samples per second requirement and provide some flexibility in samples per second, 400 Ω is desired as the total filter impedance to the input. Each IIR filter and bi-quad filter was analyzed, but each had a significant draw back in either area or phase noise, which is supported by [6].

The standard RC filter was investigated with up to 4 cascades, 4th order, in MATLab due to its robust phase noise through the passband. The frequency response requirements for the filter are an attenuation of 20 dB at 10 kHz and a maximum of 0.5 ms delay. Due to the cascade and requiring an output impedance of 400 Ω, the resistance value at each stage was found using

$$R_{casc} = \frac{R_{out}}{n} \tag{3}$$

where R_{casc} is the impedance at each stage, R_{out} is the total impedance of the filter, and n is the order of the filter. R_{casc} was used to find the stage capacitance with

$$C = \sqrt{\frac{3}{\omega^2 (\frac{R_{out}}{n})^2}} \tag{4}$$

where ω is the frequency in rad/s, R_{out} is the total filter impedance, C is the capacitance of each stage, and n is the order of the filter. A 3rd order filter was found and its values shown in Figure 5 from Equation 3 and Equation 4. The order of the filter is determined by the number of cascaded RC equivalents, which have a magnitude calculated by

$$|H(\omega)| = \frac{1}{\sqrt{1 + \omega^2 R^2 C^2}} \tag{5}$$

where ω is the frequency in rad/s, R is the resistance of each stage, and C is the capacitance of each stage. A -6 dB cutoff with respect to the magnitude was desired at 8 kHz as well as a 0.5 ms time delay. The time delay was calculated in MATLab with

$$t_d = \frac{\phi(f)}{f(2\pi)} \tag{6}$$

where t_d is the time delay in seconds, ϕ(f) is the phaseshift at frequency f in radians, and f is the frequency. Equation 5 and Equation 6 were used to calculate the magnitude frequency response and time delay for first to fourth order filters shown in Figure 7 and Figure 8, respectively.

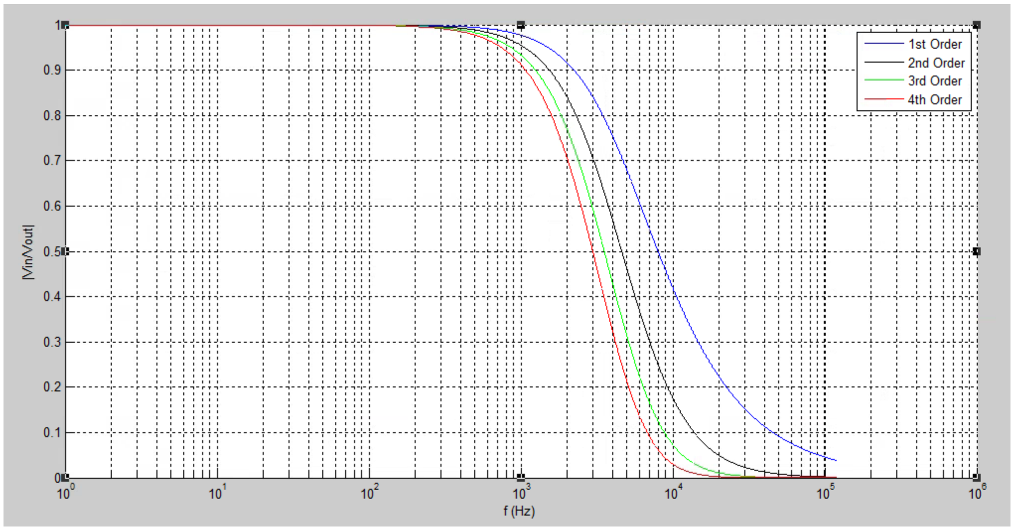


Figure 7. Magnitude Frequency Response of orders 1-4 Cascaded RC Filters

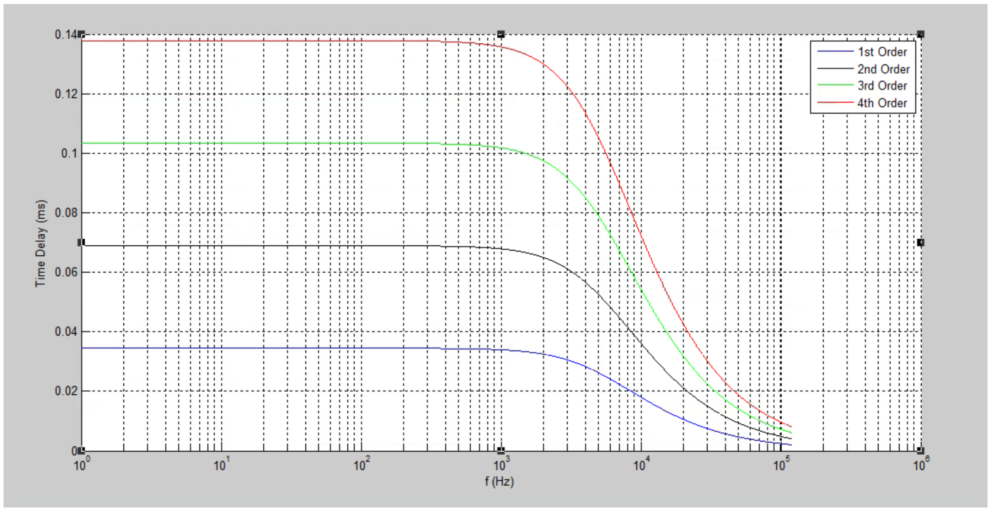


Figure 8. Time Delay (ms) vs Frequency of each Order Filter

From Figure 7 and Figure 8, it is clear that the 3rd order filter is a good compromise between all design requirements previously mentioned for the filter. It uses a little more area and has a little more time delay, but it is on approximately the same order as the 1st and 2nd time delays as well as having similar attenuation as the 4th order filter. The speech frame of 25 ms and with a maximum phase delay of approximately 0.1 ms for the 3rd order filter, the delay is less than 0.5% of the speech window. Theoretically, we can assume approximately no time delay throughout the speech frequencies of interest (< 8 kHz) with respect to the window. The frequency response and phase delay of the microphone amplifier system from LTspice is shown in Figure 9.

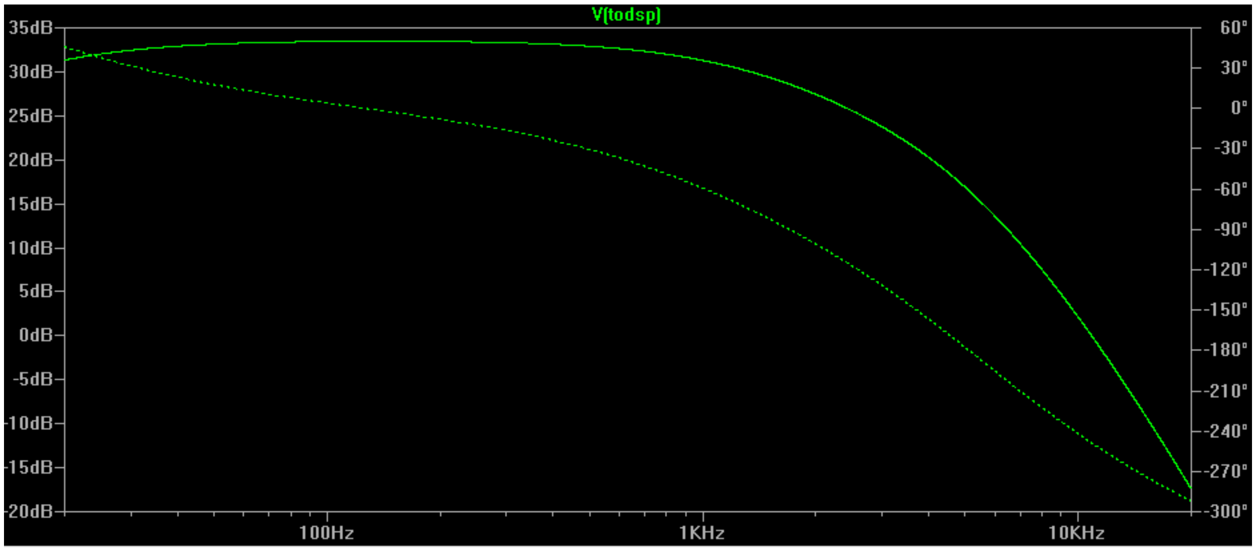


Figure 9. Frequency Sweep Simulation of amplified and filtered output to the DSP. Dashed line and solid line are the phase delay and magnitude response, respectively.

From Figure 9, it can be seen that we meet the requirements for time delay and amplification for most of the frequency range. There is approximately 15 dB lost at 8 kHz, but this is acceptable because the speech signal is mostly less than 1 kHz. Section 3.1.1 contains the test and verification procedure as well as the results for the microphone circuit.

2.2.2 Temperature

The temperature sensor part make use of two thermistors. One thermistor will be measuring the ambient temperature of the whole device, and the other will be in close contact with the user's skin. The thermistors are connected to the Analog Digital Converter (ADC) I/O port of the microcontroller. The microcontroller will then calculate temperature with the input data at an interval of 1 seconds.

The thermistor used in the project is US sensor PS103J2. This senor was chosen because of its higher sensitivity compared to most other thermistors, it has an accuracy of ±0.1C. After reading in the data through the ADC port of the microcontroller, a mathematical formula is used to calculate the surrounding temperature. Equation 7 and Equation 8 were used to find the resistance of the thermistor at the moment, and then Equation 9 is used to find the temperature at the given moment.

$$V_{out} = V_{in}\frac{R_2}{R_1 + R_2} \tag{7}$$

where R2 is the resistance of the resistor, R1 is the resistance of the thermistor, and Vin is the voltage supplied, which will be 3.3V in this case. Vout is the voltage sent to ADC.

$$R_{thermistor} = \frac{3.3 * 10k}{3.3 - V_{out}} - 10k \tag{8}$$

where $R_{thermistor}$ is the resistance of the thermistor, and Vout is the voltage to the ADC pin.

$$Temperature = \frac{B * T_n}{B + ln(\frac{R_T}{R_N}) * T_n} \tag{9}$$

where B is the thermistor parameter constant, T_n is the standard temperature (298.15K), R_T is the measured resistance of the thermistor, and R_N is the standard resistance of the thermistor, 10kΩ in this case.

To increase the accuracy in this process, the value of the resistor in this circuit is measured with a multimeter. The value of the resistor will be used for the final calculation of temperature in the microcontroller. It is also found that there is a delay in reading the ADC value, this might due to the heat retained inside and on the surface of the thermistor. This factor is noted in the final algorithm implementation.

2.2.3 Heart Rate Monitor

The heart rate sensor uses an analog front end chip, AFE4400, to perform measurement and computation of the heart rate of the user. This component communicates with the microcontroller using Serial Peripheral Interface (SPI). The Master Out Slave In (MOSI), Master In Slave Out (MISO), Slave Select (SS) and clock signal (SCLK) will be used for this interface. To calculate the heart rate, the three beat average calculation as shown in Equation 10 is used and implemented in the microcontroller:

$$Heart\ rate\ per\ minute = \frac{sampling\ frequency * 60 * 3}{no.\ of\ samples\ in\ 3\ beats} \tag{10}$$

where sampling frequency is determined in the code, and number of samples in 3 beats can be counted by the microcontroller.

2.3 User Interface

2.3.1 LCD Display

The LCD screen uses a standard 16x2 LCD Display. There will be two rolls of text and 16 characters per row. The display module is designed to display the calculated emotion state of the user. This LCD display is designed to be placed at the bottom of the wrist of the user.

2.3.2 LED light indicator

Other than the LCD screen for display, the device is also designed to have 3 LED lights for display. They will be used for a non-discrete display of emotion. The connection of these 3 LEDs are connected to the microcontroller through General Purpose Input/ Output (GPIO) pins. An npn transistor will be used to control the current to protect the microcontroller from burning out.

2.3.3 Switch

Slide Switches are used for turning things on and off in the design. A total of three switches are used in our design. One is used for the LCD Display, one for the LED lights, and last one for the whole system. Switches are connected to 3.3 V voltage supply and ground, with the middle pin connected to the corresponding component.

2.4 Power Systems

The power system supplies the power to all of the modules in the device. Lithium-Ion Polymer (LiPo) battery was chosen due to the extremely light weight as well as the energy capacity per unit volume. There are some safety concerns, which are outlined in greater detail in Section 5.3. But, in order to help mitigate any issues during operation, the LTC3553 buck convert and single cell LiPo charger was selected because of its ability to monitor the temperature, voltage, and charge/discharge rate of the battery.

Table 2 lists the power requirements for every component on the device. The mA-hr is calculated for each device and the efficiency of the buck converter on the LTC3553 charger is used to approximate the energy capacitance required for the battery. Ideally, the design requires a minimum life of approximately 5 hours.

Table 2. Power Requirements

Component	Operating Voltage Range	Maximum Current
LEDs	3.3 V	30 mA
AFE Rx	3-5.25 V	25 mA
AFE Tx	2-3.6 V	0.67 mA
Microphone	1.5-3.63 V	0.22 mA
Temperature Sensor	3.3 V	0.65 mA
OLED Display	2.4 or 3.3 V	5 mA with backlight, 0.14 mA st
IC Charger	3.3 V	0.12 mA
MSP430F6635 MCU	1.8 to 3.6 V	1.84 mA @ 8 mHz
dsPIC33EP512	3 to 3.6 V	42 mA @ Maximum operation
	Total mA @ 3.3 V:	105.5

From Table 2, it is observed that a LiPo battery with approximately 650 mA-hr at 3.7 V would power the device for 5 hours. This assumes a 90% down conversion loss for 5 hours of operation without any power conservation techniques. The Schematic for the IC charger is shown in Figure 10. The values for the capacitors and NTC thermister were specified in the datasheet, shown in [7]. Ideally, the thermister will rest on the LiPo battery while the device is in operation.

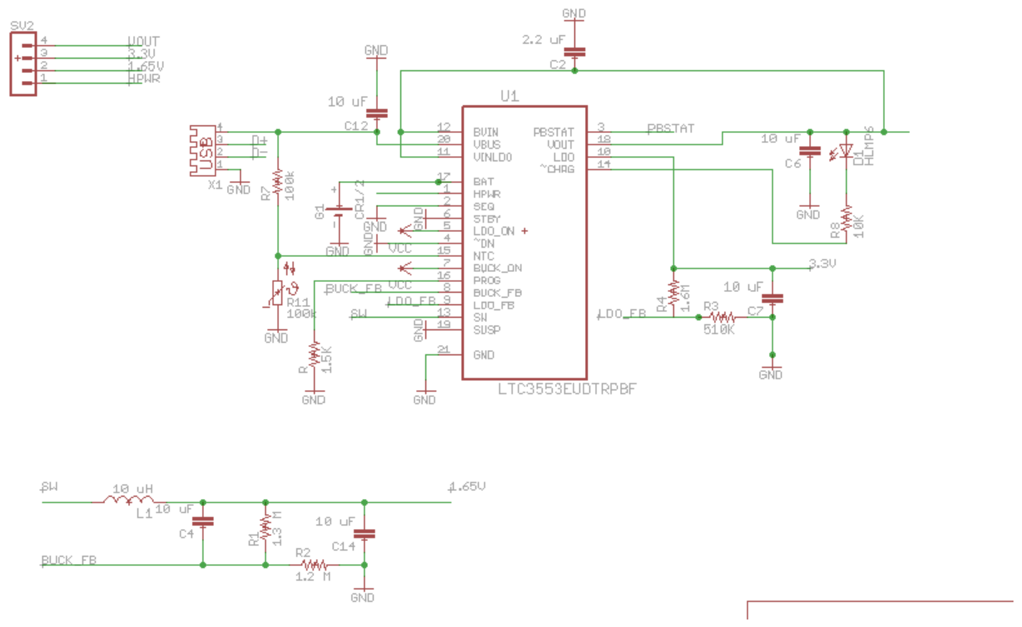


Figure 10. IC Charger Circuit for LiPo Battery

This IC charger a buck converter and LDO implemented with it. The LDO convert will be used to drive the 3.3 V line while the sw/buck_fb converter will drive the OP-AMP VCC/2 terminals at 1.7 V. The maximum current from the combined LDO and buck terminals is 500 mA. The voltage for each terminal, while referencing Figure 10, is set by

$$V_{buck} = (0.8V)(\frac{R1}{R2} + 1) \tag{11}$$

where V_{buck} is the desired output voltage for LDO or buck converter, R1 is the resistor in parallel to the capacitor(s) and R2 is the resistor in series from buck_FB to GND or from LDO_FB to GND terminals. The calculated resistor values using Equation 11 are shown in Figure 10 for the buck and LDO converters. The charge current is set by the resistor at the Prog terminal, which is given by

$$I_c = \frac{750V}{R_{prog}} \tag{12}$$

where I_c is the charge current in A, and R_{prog} is the impedance into the P_{rog} [7]. Using Equation 12, and 500 mA as the desired charge current, R_{prog} was found to be 1.5 kΩ. The charge current is also regulated by HPWR port. When it is set to 3.3V, the charge current is I_C and when it is set to 0 V, the charge current becomes 100 mA. This signal is controlled by the microcontroller. The results for this circuit and testing procedure are summarized in Section 3.3.

2.5 Microcontroller and Processing

The MSP430F6xx and F5x series has built in USB port capability. There is no need for an external JTAG or OTG chip to interface USB to SPI or UART with the MCU. The MSP430F6635 was chosen as our MCU due to the number of SPI connections available as well as extended memory capabilities. The design of the resistors and capacitors is explained in detail in [8] and [9]. A snip from the schematic is shown in Figure 11 from the USB interface circuit.

The crystal is necessary to control the USB 2.0 transfer rate. The faster the crystal, the faster the USB 2.0 can transfer data. Data rate is not really a concern with our application, therefore a 4 MHz crystal was chosen to reduce power consumption. The USB power line will come directly from the Power System section. The diode is there to ensure that current is not driven back to the power system/USB per USB 2.0 specifications, [8]. The capacitor values were selected from the suggested values listed in the datasheet in [9]. Electrostatic discharge protection is provided by the TPD2E001 chip shown connected to the data lines of the USB. The switch is used to indicate to the MCU that it should expect to be programmed from USB.

SPI was used to connect the MCU with the heart rate sensor and DSP with independent SPI ports on the MCU with the MCU being the master in both instances, the temperature sensors were connected to the MCU ADC10 channels, and the GPIO ports were used for the lines to the IC charger and User Interface.

Originally, the design did not have a DSP component. After researching the complexities of the speech algorithms, it was decided that a separate component was necessary. The requirements were low footprint, 3.3 V power and moderate current draw. The dsPIC33EP512GM304 was selected as the DSP because it met the requirements with only 44 pins. The amplified microphone out was connected to the ADC12 input and the voltage reference was inputted into the default port. The schematics for the MCU and DSP are depicted in Appendix A.

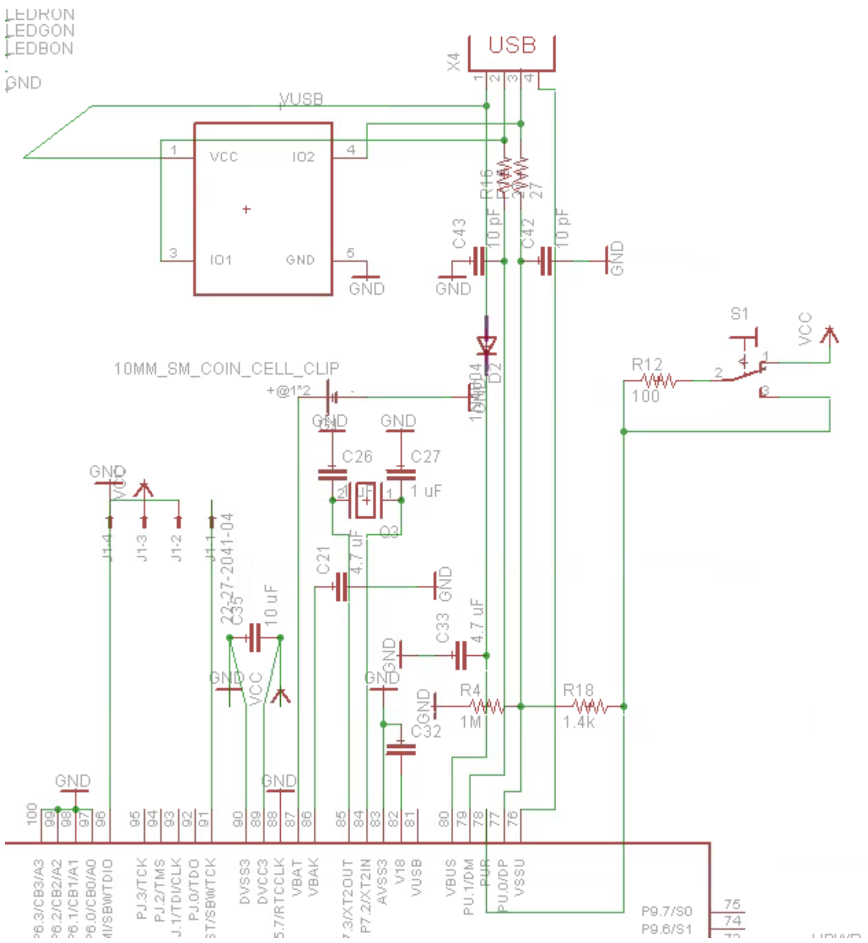


Figure 11. USB to MCU Connection

the ADC12 input and the voltage reference was inputted into the default port. The schematics for the MCU and DSP are depicted in Appendix A.

2.6 Emotion Classification

The emotion detection algorithm is the way our machine will be able to determine what emotion the user is feeling. It contains a few steps throughout our device; first, in the sensor block, the data from each specific sensor is processed and the features we want are extracted in our digital signal processor. Next, these features are sent to our microcontroller, from where we classify what emotions the user is feeling from the data we just received. After an emotion is determined, that data is then sent to our user interface block where the emotion is displayed onto our LCD screen.

2.6.1 Data Set

After extensive research, it was decided that the data points we would use were speech, heart rate, and skin temperature. The main factors behind this decision were that these were the data points that had the most impact on emotion detection while being the most portable and easy to fit into a wrist band. The heart rate and skin temperature sensors would fit right into the band and press against the skin, while the microphone would face the user and pick up any speech data from the user.

For the emotion detection algorithm to work, we need a training set with which we can compare the user data with to determine the current emotion. *Mordkovich et al*, a group which achieved ~85% accuracy with their emotion detection from speech algorithm, used the Emotional Prosody Speech and Transcripts obtained from the Linguistic Data Consortium. With such promising results, we decided to develop our algorithm with the same training set [10]. However, we were not able to find the complete data set online, so we are using a couple samples from the data set at the moment. We were also unable to find a suitable emotional biosensor dataset. We believe that finding a full dataset with more samples will greatly increase the accuracy of our algorithm.

2.6.2 Feature Extraction

When our microphone receives a speech signal, we want to keep only the features of the signal that best describe the emotion the user is feeling. According to multiple papers on the topic, the mel-frequency cepstral coefficients (MFCCs for short), the pitch, and the deltas of those features most accurately represent the frequencies of a speech signal. These features proved to be sufficient for our purposes, though more features such as pulsing and jitter of the speech signal would have improved our accuracy.

2.6.2.1 Mel-Frequency Cepstral Coefficients

The mel-frequency cepstral coefficients (MFCCs) are a feature set widely used in speech processing. It involves calculating the power spectrum of the speech signal, and converting that to the mel scale using Equation 13, then taking the logarithm of those energies, and finally the discrete cosine transform of the log energies. What this serves to do is create a representation of the speech signal on a scale that is similar to the signals our ears receive when we hear somebody talk. The mel-scale helps relate the pitch of a signal to its actual measured frequency, and since humans are better at discerning changes in pitch at low frequencies, this scale makes our signal better match what a human ear would hear [12] **.** The formula for converting to the mel-scale is

$$M(f) = 1125ln(1 + f/700) \tag{13}$$

where *f* is the frequency we want to convert to the mel-scale [12].

2.6.2.2 Pitch

The pitch of a speech signal commonly refers to the fundamental frequency of the speech signal. This correlates to how high or low a person's voice sounds, and contains speaker specific information [13]. However, if we analyze the pitch of the speaker in specific intervals, we are able to see the changes in pitch corresponding to their emotion. For example, an excited person might raise their pitch when speaking, while a bored person's pitch will drop slightly below their normal pitch. A graph depicting the pitch contour is shown in Figure 12, taken from a random segment of speech from our data set.

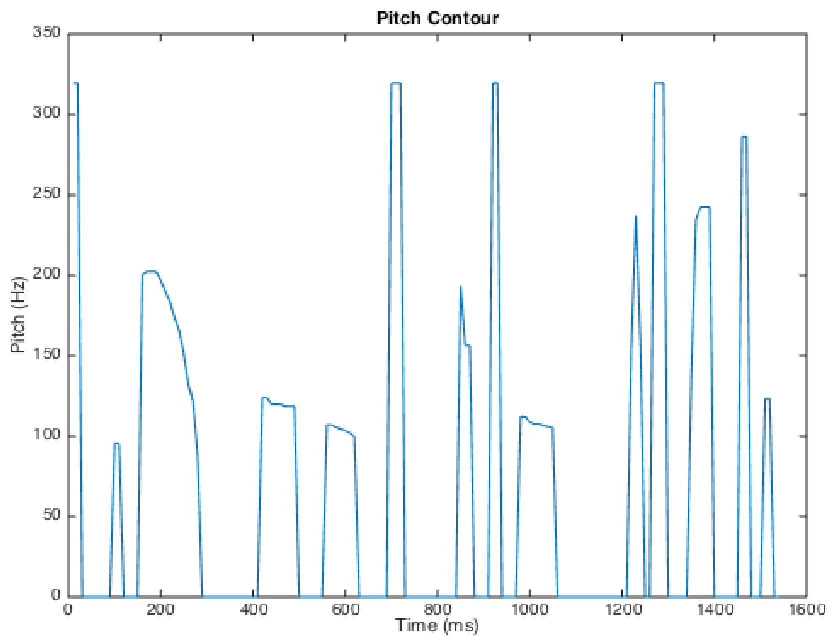


Figure 12. Pitch Contour Plot

In order to calculate the pitch, we use the autocorrelation method to find the fundamental frequency of the speech signal. The autocorrelation of the speech signal is found by

$$r(\tau) = \int_{-\infty}^{\infty} x(t)x(t+\tau)dt \tag{14}$$

where τ is the time delay and **x(t)** is the speech signal [14].

The autocorrelation function takes a signal, and multiplies it by a time delayed version of the same signal. The time delayed signal is then delayed again, until the whole signal has been shifted through once. The resulting graph shows how much the signal correlated with itself at each time delay. The first peak of the autocorrelation sequence corresponds to the zero lag signal, so it will always have the highest amplitude. Since speech can be treated as a periodic signal in short segments [13], the signal will roughly repeat itself at a certain time delay, which is represented in the autocorrelation sequence as the second largest peak. This is the fundamental frequency or pitch of the speech signal. A graph depicting a voice signal and the auto correlation sequence is shown in Figure 13.

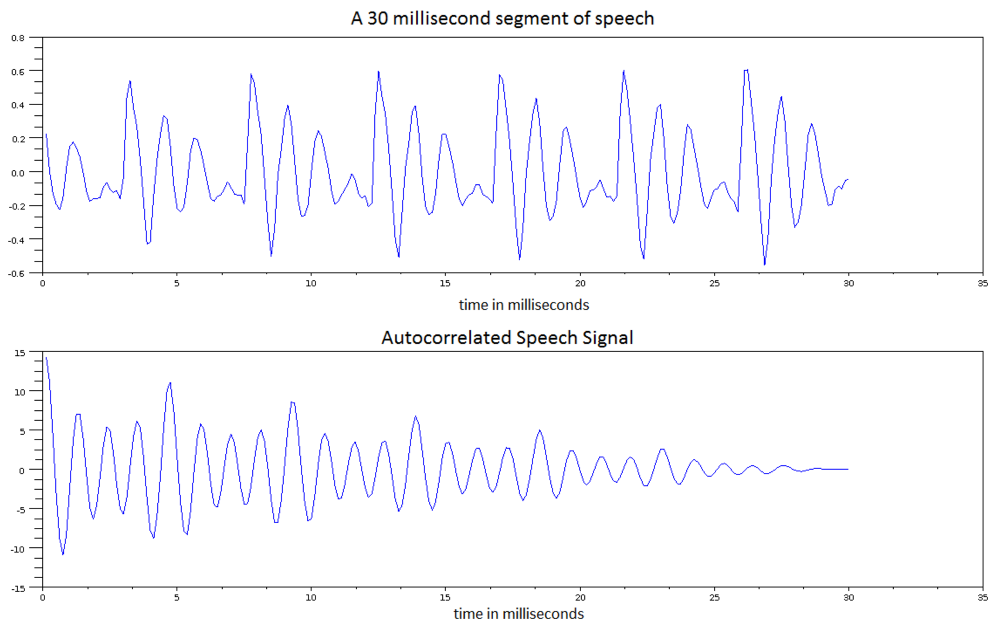


Figure 13. The top graph shows a speech signal, and the bottom graph shows the autocorrelation sequence. At time delay 0, we can see the highest peak of the signal, since we are correlating a signal with itself. At approximately 5 ms, we can see the second larges

2.6.2.3 Deltas

The deltas of the above features represent the change in those features over time. This shows us the speed of the speech and the rate at which the pitch is changing. This is useful because the rate of speech and pitch can tell us useful information about the user's emotion. For example, if the user is excited, the rate of speech will go up, while if the user is bored, the rate of speech will drop back down to normal levels. The formula we used to calculate the deltas is

$$d_t = \frac{\sum_{n=1}^{N} n(c_{t+n} - c_{t-n})}{2\sum_{n=1}^{N} n^2} \qquad (15)$$

where *N* is the window size, and *t* is the frame count [12].

2.6.2.4 Biosensor Data

For the heart rate and skin temperature data, we take the mean of the raw signals and the standard deviation of the raw signals. This data can help characterize an emotion based on how fast a heart is beating or how warm their skin feels. For example, if the user is excited, the heart rate will increase, or if the user is frightened, their skin temperature will usually decrease. We were not able to find a good data set with enough samples to train a sufficient model for biosensor readings based on emotion, so we were not able to test how well our biosensor features categorized emotion.

2.6.3 Support Vector Machines

For our classification method, we chose support vector machines (SVM). SVM is a machine learning algorithm that tries to find the maximum distance plane between 2 or more classes. Once this plane (can be linear or non-linear) is found, any new data that we want to test is compared with the plane to see which side the new test data falls under. For multi-class problems, we use the one vs one strategy, where our test data is compared with every pair of classes to see which class the test data is closest to. Another common method is the one vs rest strategy, where each class is trained against all the rest of the classes combined. The winning class is then picked by which training plane is closest to the test point. The maximum distance plane is found by optimizing the following equation

$$\begin{aligned} \min_{\mathbf{w},b,\xi} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^T\phi(\mathbf{x}_i)+b) \geq 1-\xi_i, \\ & \xi_i \geq 0. \end{aligned} \tag{16}$$

where **x** is the label vector, **y** is the feature vector, and **w** is the normal vector to the hyperplane [15]. The kernel parameters will be optimized in a following step. To better illustrate the maximum distance plane separating two classes, Figure 14 shows a binary SVM. The hyperplanes H1 and H2 represent the boundary of class 1 and 2 respectively. We then find the maximum distance between those two lines to find our hyper plane.

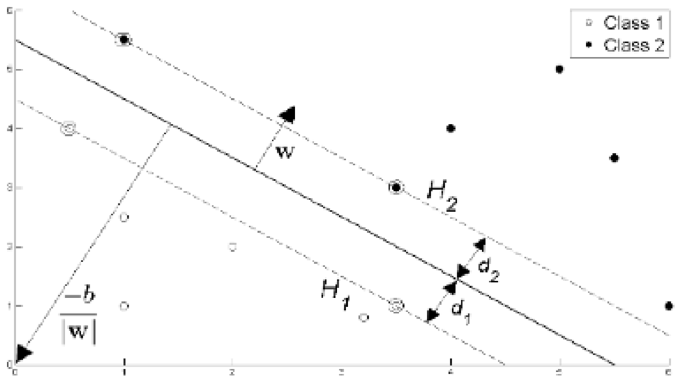


Figure 14. Maximum distance boundary plane between two classes [16]

2.6.3.1 Procedure

In order to speed up our testing process, we used an open source library LIBSVM [17]. This library provides functions that help us train our classifiers, and also predict what class a test speech segment falls under. Before training our feature sets, we first scaled the features so that the whole feature set was within [-1, 1]. This prevents features in a larger numeric scale from dominating features in a smaller numeric scale. Scaling was done with

$$f(x) = C + (D-C)\frac{x-A}{B-A} \tag{17}$$

where [**A,B**] is the range we are converting from, and [**C,D**] is the range we are scaling to. In our case, [**C,D**] is equal to [-1,1]. It is important to remember to scale both the training set features and the testing set with the same constant for the most accurate result.

Next, we pick a kernel, which maps our feature vectors into a higher dimensional space. We picked the Radial Basis Function kernel (RBF) because it non-linearly maps our feature vectors into a higher dimension. That way, our kernel will still be able to map a plane if the maximum distance between two classes is nonlinear [15]. The equation for the RBF kernel is

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2),\ \gamma > 0. \tag{18}$$

where **x** are the training vectors and γ is a kernel parameter [15].

Once we have our kernel picked, LIBSVM has a function *svmtrain* that trains the scaled feature data. We then pass in our test feature vectors for a classification result. As of now we have not optimized the kernel parameters for the training function, but that is something we can do in the future to improve our accuracy.

2.6.4 Classification

The data set we used provided us with speech samples for 15 emotions: hot anger, panic, anxiety, disgust, sadness, contempt, cold anger, shame, despair, boredom, neutral, elation, pride, interest, and happy. Our initial tests classified each individual emotion with a result of 27.53% accuracy. While this result was better than guessing randomly between the 15 emotions, we wanted a better result from our algorithm.

With the same data set, *Mordkovich et al* tested the accuracy of the data set using grouped emotions for better results [10]. Since emotion can be classified in two dimensions by valence (positivity) and arousal (excitability), we are able to split the emotion into a two dimensional space as shown in Table 3. We tried the same emotion grouping, and labeled each emotion group in Table 3 as its own emotion category. Using these new emotion groups as the classes, the accuracy of our algorithm increased to 42.53%. This is testing between 6 different classes, so our results were slightly better than random.

Table 3. Emotion Classification by Arousal/Valence [10]

	Low Valence	Neutral Valence	High Valence
High Arousal	hot anger, panic, anxiety		elation
Neutral Arousal	disgust, sadness, contempt, cold anger, shame		pride, interest, happy
Low Arousal	despair	boredom	

In order to further improve the accuracy of our algorithm, we treat the emotion classification as a two dimensional binary problem. Instead of classifying individual groups of emotions in Table 3, we first test whether an emotion is classified as high valence or low valence. Then we test whether the emotion is classified as high arousal or low arousal. In this way, we can predict the valence and arousal values of an emotion and be able to guess which emotion group it belongs in. Using these new arousal and valence classes, we achieved approximately 66.9% accuracy along the valence axis and approximately 71.8% accuracy along the arousal axis.

3. Design Verification

3.1 Data Collection

3.1.1 Microphone

The microphone functionality was verified with an electret microphone because the MEMS microphone had some complications with soldering to the board. The gain was adjusted to 8 V/V due to the electret output amplitude. The DC offset parameter was verified with DC coupling on the oscilloscope and the gain parameter was verified by comparing the amplitude of the input signal with the amplitude of the output signal with AC coupling.

3.1.2 Temperature

The temperature sensing module functionality was verified by using an external thermometer. The calculated value from the microcontroller is printed out and the values were cross verified. The results showed that the thermistors are working. The circuit is also tested by placing hand over the thermistor, and seeing if the reading changes as expected.

3.1.3 Heart Rate monitor

The functionality of the Heart Rate monitor was not verified because of problem concerning the SPI connection with the microcontroller. During debugging process, the first microcontroller was burnt, and the SPI communication stopped on the second microcontroller. It is concluded that the second microcontroller fails on SPI connection because no clock signal was seen on the oscilloscope, therefore, signals were not able to be shifted in and out of the microcontroller. To verify the soldering of the AFE chip on the PCB, multimeter was used to detect for short circuits, and none were detected. The AFE also has sign of life after powered on, which was verified using suggestions from the Texas Instruments website. [11]

3.2 User Interface

3.2.1 LCD Display

The LCD Display uses a 16x2 LCD. The functionality of this LCD is verified by writing strings of word in the microcontroller and see if the display outputs the design words. This function worked as expected. One of the pins on the LCD Display adjusts the brightness of the backlight based on the voltage input, instead of using a potentiometer for this pin, a 1kΩ resistor is used between power supply and this pin to provide a suitable brightness.

This component is tested again after the switch is implemented, the words display is reset and does not distort after power supply turned off and on.

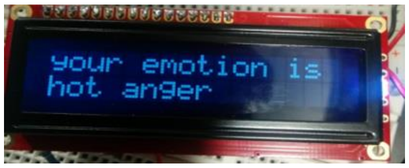


Figure 15. Words display on LCD Screen

3.2.2 LED Light Indicator

The LED lights were first tested its individually for the functionality. After seeing that the LED lights light up and does not overheat over a long period of time then it is tested to see if it lights up in the right order as corresponds to the LCD. This component is tested again after the switch is implemented, the words display is reset and does not distort after power supply turned off and on.

3.2.3 Switch

The switches were verified using a power supply and multimeter. The multimeter set to voltage reading and is connected to the output of the switch. The result is as expected, output is 3.2V ±0.1V when the switch is at high, and 0V when low. It is also then connected to an oscilloscope to observe any debouncing behavior and it is not noticeable on the oscilloscope and did not affect functionality of circuit.

3.3 Power Systems

The voltage ripple was verified by connecting a two watt 33 Ω resistor as the load across the 3.3V output line, to simulate 100 mA current draw. The signal across the resistor was measured with AC coupling on the oscilloscope to measure the peak to peak voltage, which was approximately 100 mVpp. The DC signal was observed with a multimeter under the 100 mA load. The charge current was tested for both 100 and 500 mA, utilizing two Watt 40 Ω and 8 Ω resistors, respectfully, in series with a 100 Ω 15 turn potentiometer. The potentiometer was adjusted such that the voltage across the series capacitors went over 4.1 V and charging was observed to stop.

3.4 Microcontroller and Processing

The MCU was capable of being programmed through the USB and was able to turn on LEDs and accept switch input signals. The DSP was programmable with pickit3 and was able to accept data from the microphone as well as turn on LEDs. The SPI communication failed to work between the MCU and DSP, because the master clock was not being generated by either chip. Failure of the SPI made algorithm integration not possible.

3.5 Emotion Classification

In order to test our emotion detection by speech algorithm, we used a v-fold cross validation method. We first divide our data set into *v* subsets of equal size. Then for each subset, we train the remaining *v-1* subsets and test the subset we left out against this classifier. We do this *v* times, until each subset has been tested once. The accuracy of this is then averaged over all *v* cases to find the overall accuracy of our algorithm.

Using 5-fold cross validation, we achieved approximately 66.9% accuracy along the valence axis and approximately 71.8% accuracy along the arousal axis. We created confusion matrices shown in Table 4 to further analyze the performance of our algorithm. Each column contains our predicted classes, which is the class our algorithm picked, while each row contains the actual class the emotion fell under. In this way we can observe any bias in our algorithm and see if any class is favored over the other. We also calculated sensitivity and specificity values for high valence and high arousal in Table 5 to show the performance of our algorithm.

Table 4. Confusion Matrixes for Valence and Arousal

	Predicted High Valence	Predicted Low Valence
Actual High Valence	32	48
Actual Low Valence	5	75
	Predicted High Arousal	Predicted Low Arousal
Actual High Arousal	41	23
Actual Low Arousal	13	51

From Table 4, we observe that our algorithm tends to pick the low valence class over the high valence class. The algorithm will only classify correctly 40% of the time when the emotion is high valence, but will classify correctly 93.75% of the time when the emotion is low valence. This could be due to our small number of training samples in our data set, or the features we choose to extract. In the future, we can include more features to extract from speech to try and improve the accuracy of our algorithm.

We can also see that our algorithm slightly favors the low arousal class over the high arousal class. The second confusion matrix in Table 4 shows that our algorithm will classify correctly when the emotion is high arousal 64.1% of the time, and will classify correctly when the emotion is low arousal 79.7% of the time. While these values are not ideal, they form a good basis from which to improve in the future.

Table 5. Statistical Measures of Performance for Valence/ Arousal

	Sensitivity	Specificity	Precision	Negative Predictive Value
High Valence	40.00%	93.75%	86.50%	61.00%
High Arousal	64.10%	79.70%	75.90%	68.90%

4. Costs

4.1 Parts

Table 9 is the parts cost table and is listed in Appendix B due to its size. It does not include breakout board parts or other prototyping costs.

4.2 Labor

Table 6. Labor Cost Approximation

Name	**Hourly Rate**	**Total Hours(hr)**	**Total ($) = 2.5*(Hourly Rate)*(Hours)**
Jonathan Fouk	30	150	11,250
Matthew Palmer	30	150	11,250
Vivian Tseng	30	150	11,250
Total			**33,750**

4.3 Grand Total

Table 7. Total Cost

Section	Total ($)
Parts	83.20
Labor	33,750.00
Grand Total	33,833.20

5. Conclusion

5.1 Accomplishments

We have successfully designed and tested most of the components in the project. The display module, biosensors, microphone and amplifying circuit, power supply all worked as designed. We were able to design an algorithm that can extract emotion information from speech, which is a great proof of concept to our project.

5.2 Uncertainties

Because of the malfunction with the SPI communication at the microcontroller, the digital signal processor's functionality could not be fully tested. The algorithm worked well in MATLAB, but since it could not be tested on the microcontroller, it is uncertain how accurate the functionality of the whole system is.

5.3 Ethical considerations

All members in the team are aware of the IEEE Code of Ethics. Our device is designed to help people with social disability express their emotions.

1. to accept responsibility in making decisions consistent with the safety, health, and welfare of the public, and to disclose promptly factors that might endanger the public or the environment;

- We will ensure that the power supply from the lithium polymer battery is connected safely and in addition we will monitor the temperature and shut down the device if the temperature of the battery is over 50 C.

3. to be honest and realistic in stating claims or estimates based on available data;

- We will calculate the battery life of the device and report the battery life in the specifications as measured
- We will test the device on different users and determine the percentage of accuracy our device is. We will not exaggerate the accuracy of the system in our final report.

5. *to improve the understanding of technology; its appropriate application, and potential consequences*;
   - This product helps us understand how our body react and changes with our changing emotion. We see a lot of potential future development in this product and growth by adding different combinations biosensors and having more advance audio processing technology.

6. *to maintain and improve our technical competence and to undertake technological tasks for others only if qualified by training or experience, or after full disclosure of pertinent limitations;*
   - We are continuously seeking more in depth knowledge in the areas of engineering covered in this project. Research notes were taken throughout the process.

7. *to seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others;*
   - Through design review and TA meeting sessions, we take advice from our peer and mentors to improve the design of our system, and we will credit their contributions

9. *to avoid injuring others, their property, reputation, or employment by false or malicious action*;
   - We will design a good algorithm for determining the user emotion, and if the detected result has a high percentage of uncertainty, it will inform the user. We will also state the limitations of the product for potential users.

5.4 Future work

5.4.1 Emotion Classification

We want to further test our emotion classifier and improve the accuracy. This can be done by extracting and testing extra features from the speech and biosensor data. We also want to conduct more testing on the biosensor data and either find a data set that correlates emotion and biosensor data, or create our own data set. Then we can train a new model that includes more biosensor data. Finding a better speech data set will also help to improve the accuracy of our algorithm.

5.4.2 Interfacing between modules

The SPI between the microcontroller, heart rate module and the digital signal processor needs to be fixed. This will be done step by step, first testing the functionality of SPI with the microcontroller being the master and reading the signal output using an oscilloscope. This would help debugging and finding out what is wrong before going too far.

References

[1] *High SPL Analog Microphone with Extended Low Frequency Response,* datasheet, InvenSense, Inc., 2000. Available at: http://www.invensense.com/mems/microphone/documents/ICS-40300_DS.pdf

[2] *LM321 Low Power Single Operational Amplifier,* datasheet, Texas Instruments, Inc., 2014, Available at: http://www.ti.com/lit/ds/symlink/lm321.pdf

[3] *16-Bit Digital Signal Controllers with High-Speed PWM, Op Amps, and Advanced Analog Features*, datasheet, Microchip, Inc., 2014, Available at: http://ww1.microchip.com/downloads/en/DeviceDoc/70000689d.pdf

[4] Y. Deng, S. Chakrabartty, and G. Cauwenberghs, "Analog auditory perception model for robust speech recognition", *2004 IEEE International Joint Conference Proceedings*, vol. 3, no. 1, pp 1705-1709, July 2004.

[5] *Analog-to-Digital Converters*, Microchip, Inc., Chandler, AZ, 2013, Available at: http://ww1.microchip.com/downloads/en/DeviceDoc/adc.pdf

[6] A.B. Williams and F.J. Tayler, "Low-Pass Filter Design," in *Electronic Filter Design Handbook*, 4th ed., New York, NY: McGraw-Hill, 2006, pp. 89-128.

[7] *Micropower USB Power Manager with Li-Ion Charger, LDO, and Buck Regulator*, datasheet, Linear Technology, Inc., 2010, Available at: http://cds.linear.com/docs/en/datasheet/3553fb.pdf

[8] *Starting a USB Design Using MSP430 MCUs*, Application Report, Texas Instruments, 2014, Available at: http://www.ti.com/lit/an/slaa457a/slaa457a.pdf

[9] *Mixed Signal Microcontroller for MSP430F663X Series*, datasheet, Texas Instruments, Inc., 2013, Available at: http://www.ti.com/lit/ds/symlink/msp430f6633.pdf

[10] Mordkovich Mordkovich, Alex, Kelly Veit, and Daniel Zilber, "Detecting Emotion in Human Speech," December 2011.

[11] AFE4400/ AFE4490/ AFE4403 FAQS, Texas Instruments, 2014, Avaliable at: https://e2e.ti.com/support/applications/high_reliability/f/30/t/369445. Accessed April 2015.

[12] Lyons, James, "Mel Frequency Cepstral Coefficient (MFCC) Tutorial," *Practical Cryptography*. Available at: http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/. Accessed April, 2015.

[13] "Estimation Of Pitch From Speech Signals," *Sakshat Virtual Labs*, Web page. Available at: http://iitg.vlab.co.in/?sub=59&brch=164&sim=1012&cnt=1. Accessed April, 2015

[14] "Pitch Detection Methods," Web page. Available at: http://sound.eti.pg.gda.pl/student/eim/synteza/leszczyna/index_ang.htm#a3. Accessed April, 2015

[15] Hsu, Chih-Wei, Chih-Chung Chang, and Chih-Jen Lin, "A Practical Guide to Support Vector Classification,"   National Taiwan University, Taipei 106, Taiwan April 2010.

[16] Fletcher, Tristan. "Support Vector Machines Explained." March 2009.

[17] "LIBSVM -- A Library for Support Vector Machines." *LIBSVM -- A Library for Support Vector Machines*. Web page. Available at: http://www.csie.ntu.edu.tw/~cjlin/libsvm/#nuandone. Accessed April, 2015.

Appendix A Requirement and Verification Table

Table 8. System Requirements and Verifications

	Requirements	Verification	Verification status (Y or N)
Power Supply	a. Charge battery to 4.2 V ± 1% with maximum charge current of 500 mA b. Supply 3.3V with ripple of +/- 0.20V with 150 mA load. c. Supply 1.8V +/- 0.1 V to Vcc/2	a. Use multimeter to measure voltage difference across battery while charging. Measure charge current with separate multimeter. When fully charged, record voltage across battery. b. Use multimeter oscilloscope to measure the voltage across the buck converter output.	Y Y
Microcontroller	a. Print Text onto LCD display b. Correctly choose suggestion words to display onto LCD depending on the detected emotion c. Maximum microcontroller temperature will not exceed 60°C when operating at 8 MHz	a. Input strings of characters to microcontroller and test it's ability to correctly display character on LCD b. Program test cases into the Microcontroller with every possible emotion, then analyze the suggested words and determine relevancy c. Use an external temperature sensor to measure the temperature of the microcontroller when in use, compare that to the internal temperature	Y Y Y
DSP	a. ADC correctly reads and converts values b. DSP can communicate with the MCU through SPI c. Extracts mel-frequency coefficients, pitch, and the deltas of the speech signal	a. Connect photoresistor to ADC and check that bright light and low light are accurately converted to values between -1 and 1 b. Send a specified value across SPI to the MCU and display that value on the PC c. Send test speech signal into DSP to extract features, then compare feature vectors with the results in MATLAB	Y N N
Temperature Sensor	a. Determine temperature of skin with an accuracy error < ±1.0C	a. Using an external temperature sensor (thermometer), compare temperature readings and the difference is less than ±1.0 C	Y
Heart Rate Sensor	a. Determine the heart rate (Beats per minute) of	a. Compare heart rate from sensor with an external heart rate monitor and	N

	user with an accuracy of 95% i. accurate SPI communication between chip and MCU	calculate percentage error of our HRM	
Microphone/ Audio	a. Able to record sound from 1 cm distance and filter out noise b. Output Waveform DC offset is 2.0V ± 0.2 V from amplifier stage. i. Amplitude gain for amplifier must be 10±2 V/V for a frequency sweep from 20 Hz to 10kHz	a. Hold microphone 1 cm from mouth and speak in a regular tone. b. Use signal generator to create a 25 mV_{rms} sinewave w/ 0.8 V offset and frequency sweep from 0 Hz to 10 kHz. Input that into the amplifier stage and record the output. Check the output offset. i. Record the microphone output for a 80 dB noise with frequency between 20 Hz and 22 kHz. ii. From b check the gain as the signal generator's frequency is swept to 10 kHz from 20 Hz.	Y
Display	a. LCD display words that are preloaded onto the microcontroller b. LED light display will be on or off depending on a signal from the Microcontroller	a. Program the display to show characters on display b. Send on and off commands to the LED to check if it turns on and off	Y
Switches	a. When ON state, the buck and LDO converters output 3.3V and 1.8V approximately.	a. Connect multimeter to 3.3V line and 1.8V line, read output when switch is set to off and switch is set to on .	Y
Emotion Classification Algorithm	a. Identifies emotion with an accuracy of 60% along the valence and arousal axes	a. Perform cross validation with the data set for binary valence and count number of correct classifications b. Perform cross validation with the data set for binary arousal and count number of correct classifications	Y Y

Appendix B Detailed Parts Cost Table

Table 9. Parts Costs

Item	Part number	Quantity	Bulk Purchase Cost ($)	Actual Cost ($)
Resistors (Surface Mount)	-	30	0.20	6.00
Capacitors (Surface mount)	-	50	0.20	10.00
Slide Switch	NKK switches MS13ANW03	3	3.84	11.52
Thermistor	PS103J2	2	3.19	6.38
Heart Rate Monitor	AFE4400	1	8.42	8.42
Green LED	APL3015MGC	1	0.58	0.58
photodiode	QSB34CGR	1	1.05	1.05
Display LEDs	-	3	0.25	0.75
NPN Transistor	DSC2A01T0L	1	0.06	0.06
Microphone	InvenSense ICS-40300	1	3.50	3.50
LCD Display	Basic 16x2 Character LCD	1	14.95	14.95
MCU	MSP430F6635	1	5.39	10.78
DSP	dsPIC33EP512G304	1	3.15	6.84
ESD	EPD	1	0.07	0.45
CoinCell Battery	3 V 10mm Coincell P031-ND	1	0.57	1.10
Op-Amp	LM321	1	0.10	0.61
CoinCell Holder	Keystone Coil Cell Battery Holder	1	0.02	0.10
USB Connector	Conn USB MICRO B SMT	1	0.23	0.46
LiPo Battery	650 mA 3.7 V LiPo Battery	1	17.46	17.46
IC Charger	LTC3553	1	2.73	3.22
		Total	**53.15**	**84.20**

Appendix C Circuit Schematics

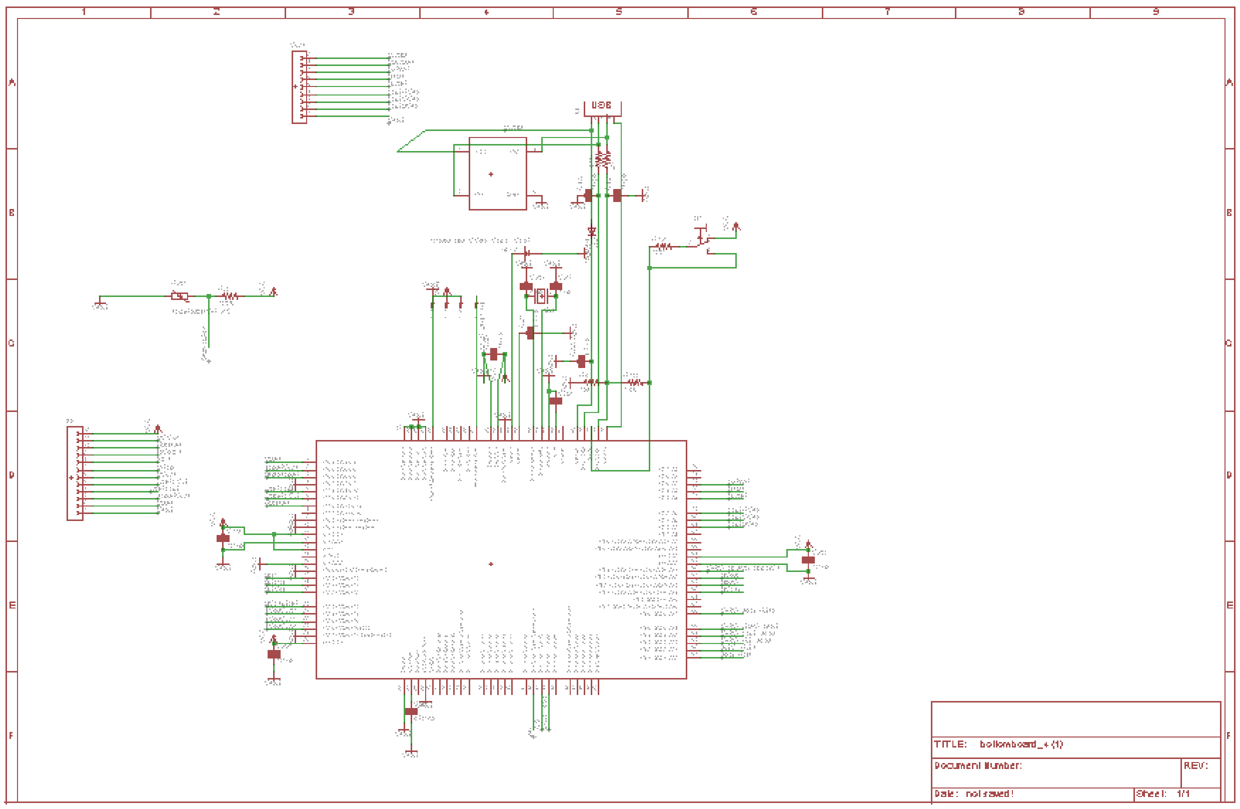


Figure 16. Microcontroller Schematic

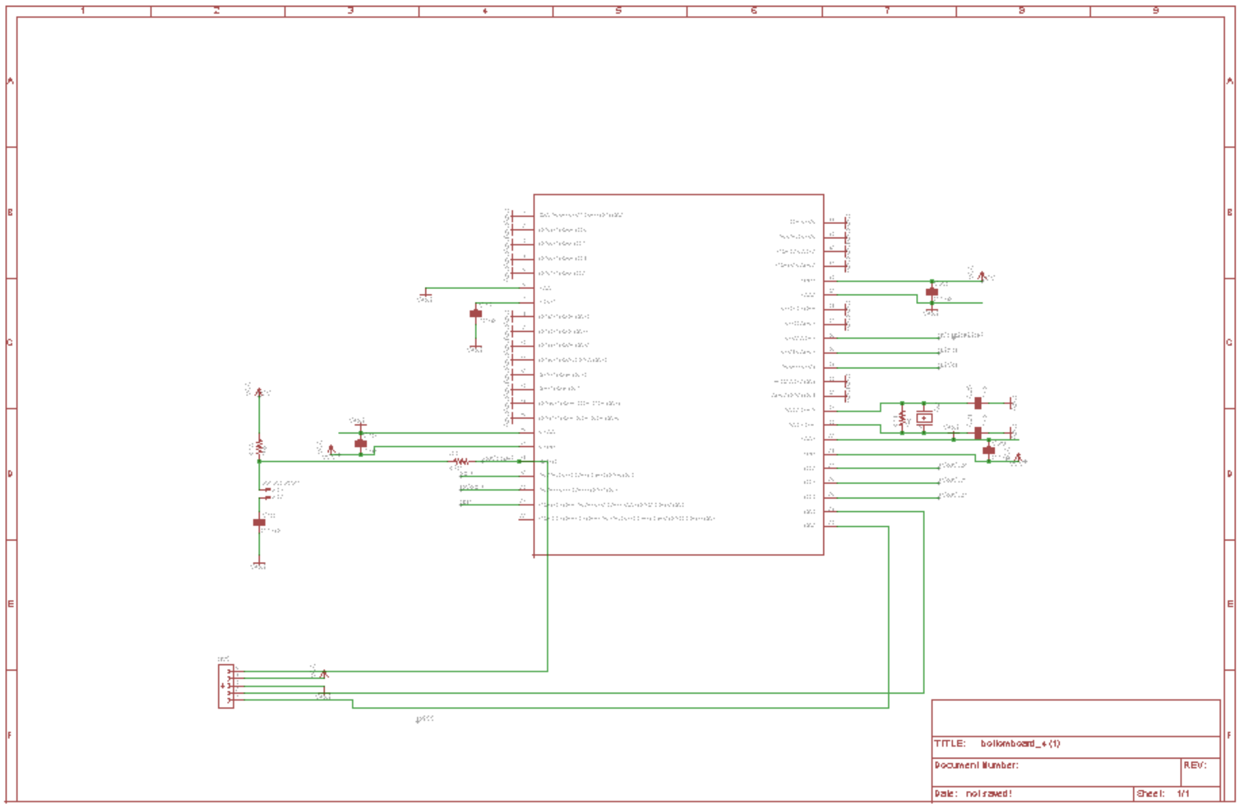


Figure 17. Digital Signal Processor Schematic

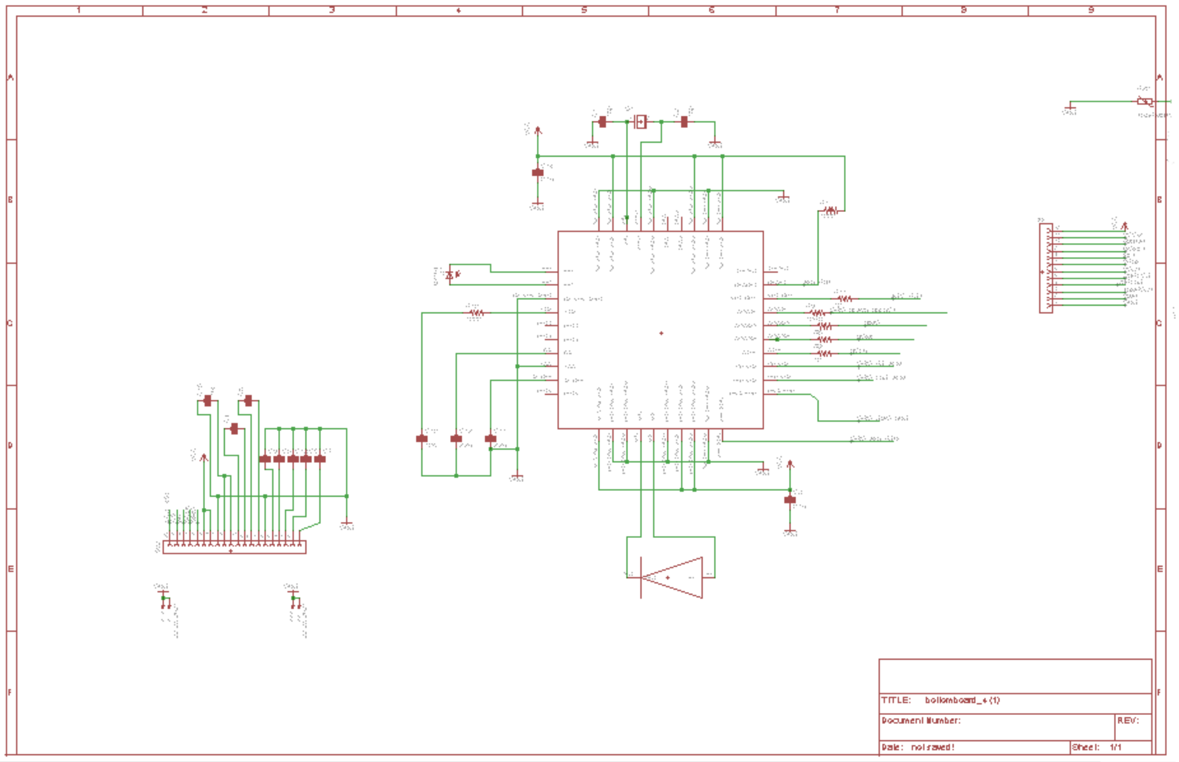


Figure 18. AFE4400 Chip and LCD display circuit schematic

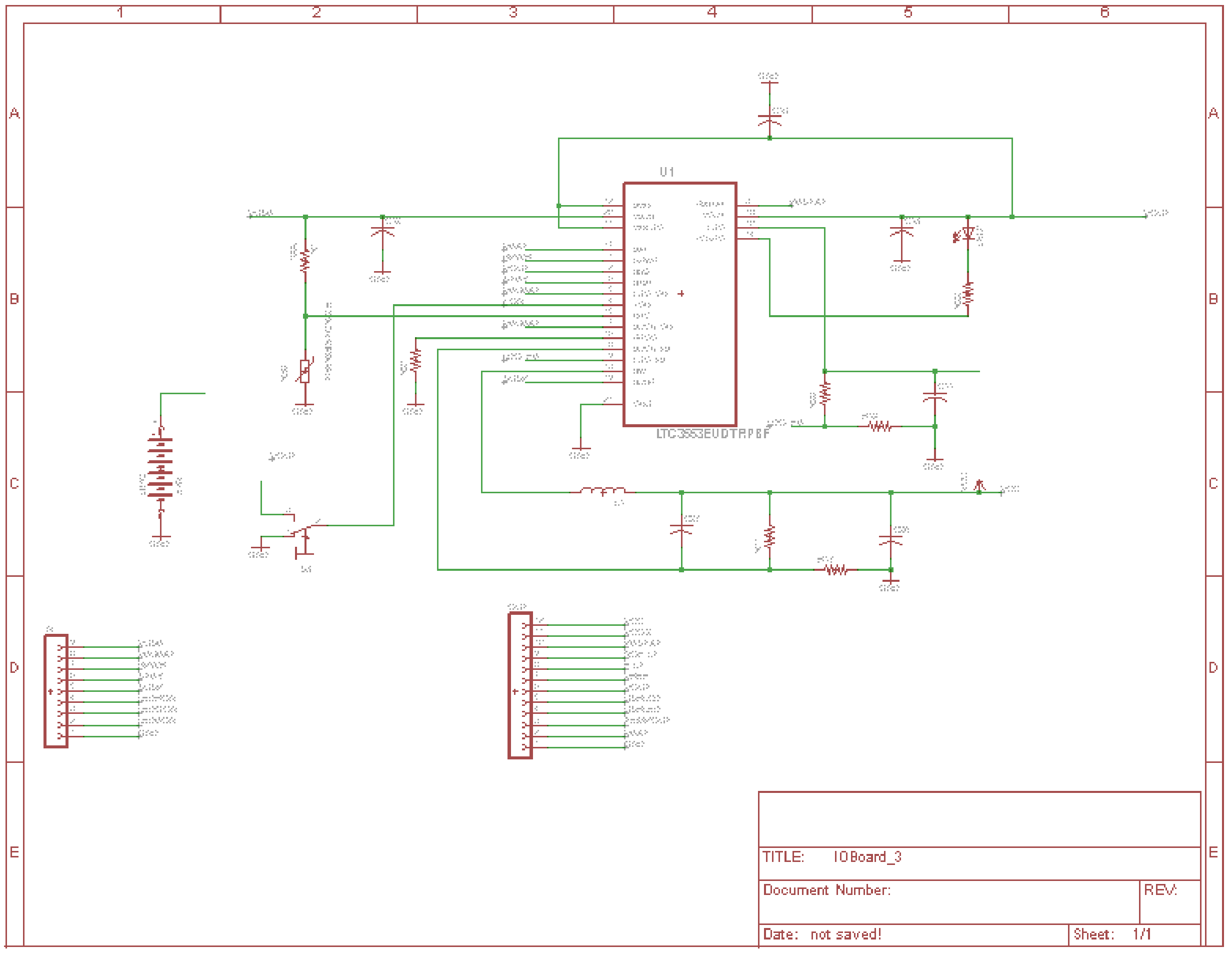


Figure 19. LiPo Charger, part of IOBoard

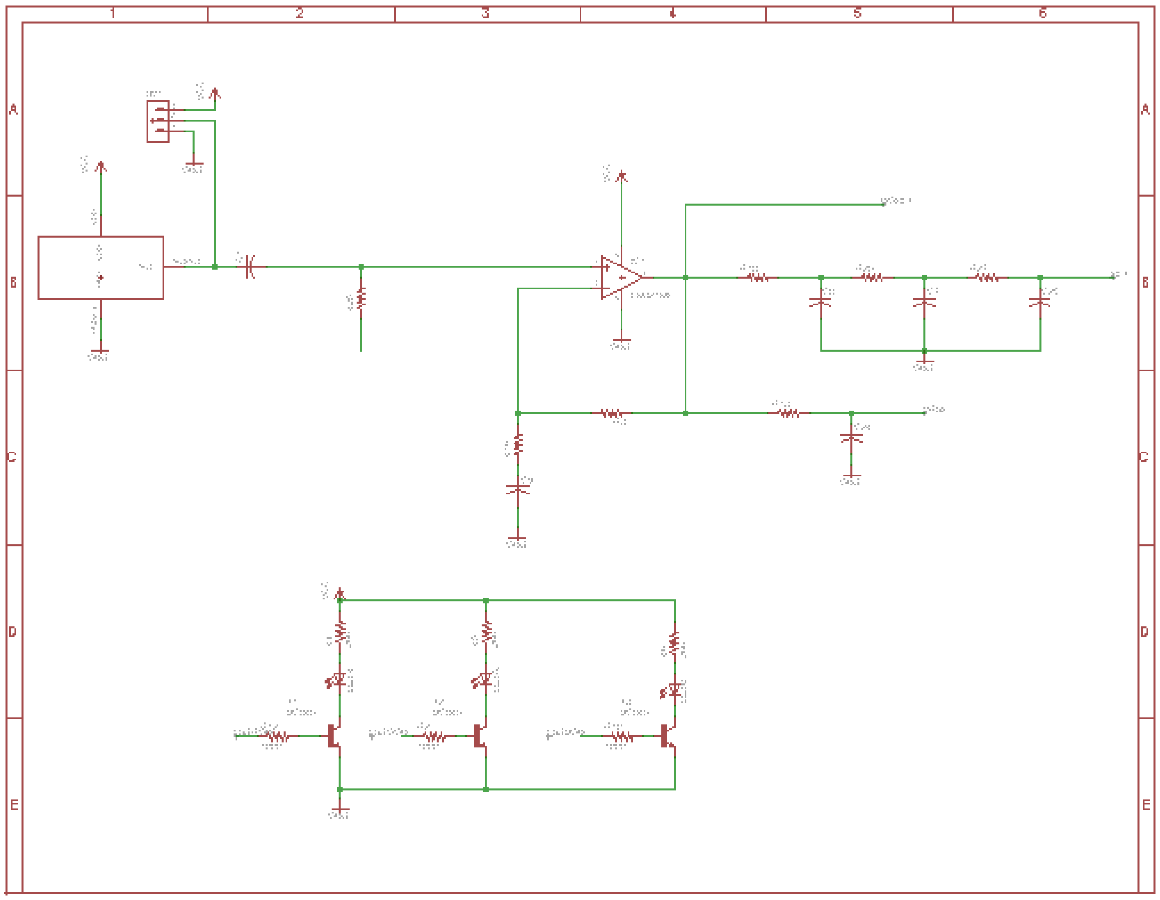
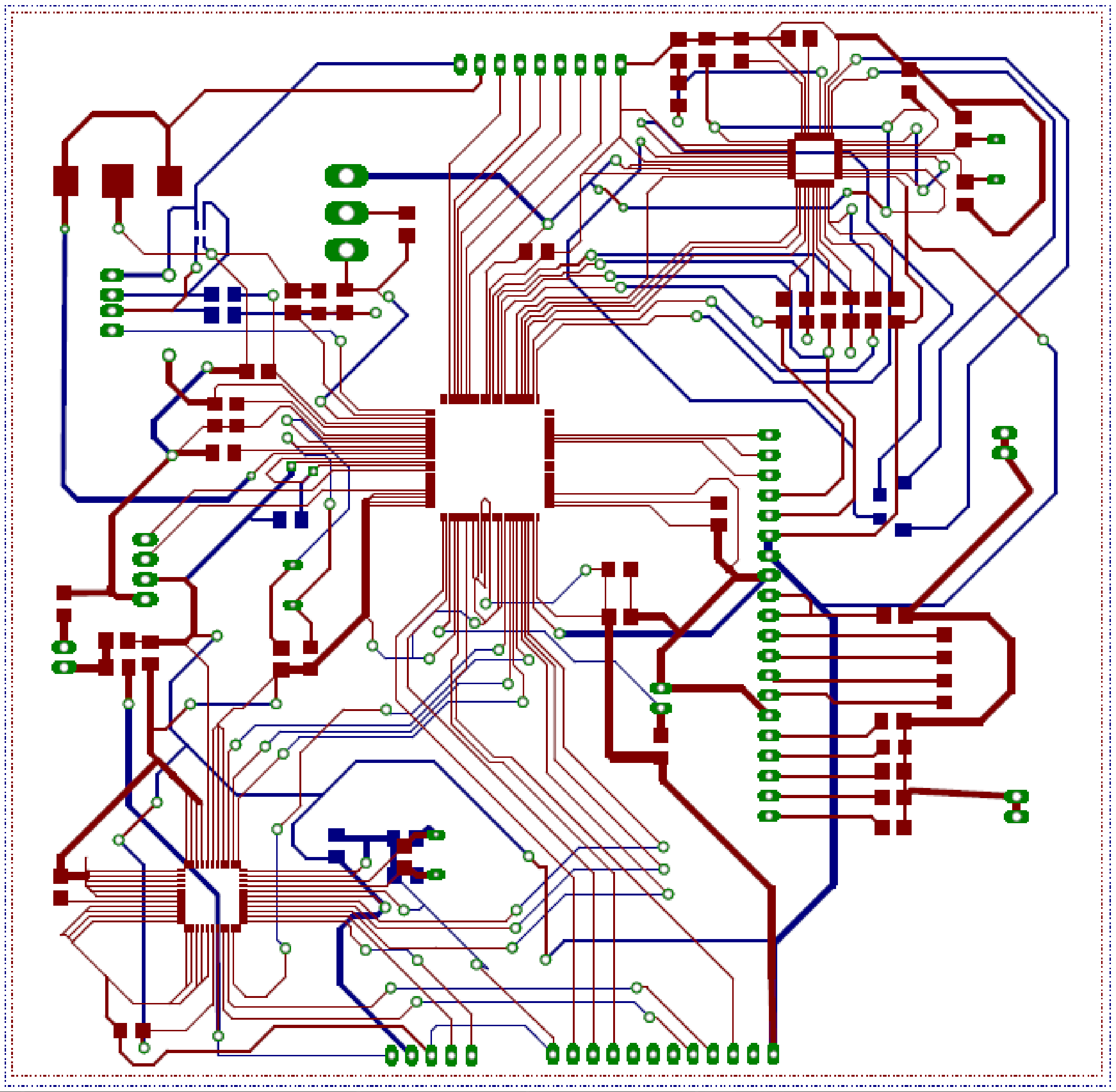


Figure 20. Microphone and OP Amp with LED circuits

Appendix D PCB Layout



FIGURE_F5. Bottom Board layout

Appendix E Codes

```
%% Emotion Detection
%written by Jonathan Fouk
%init libraries
%suppress nonIntegerIndex warnings
warning_id = 'MATLAB:colon:nonIntegerIndex';
warning('off',warning_id);
```

```
%% extract dataset
display('Extracting speech segments...');
[speech_set, Fs] = read_audio_data('Data');
display('Dataset extracted!');

%% feature extraction
display('Extracting features...');
speech_set = extract_features(speech_set,Fs);
display('Features extracted!');


%% convert to libsvm format
[all_labels, all_attributes, speech_set] = export_as_libsvm(speech_set,
'data_set.txt');
```

```
function [audioVector,Fs ] = read_audio_data( file_directory )
%read_audio_data
%   opens up file_directory and extracts all data into 2 cells
%   audioVector: contains audio data
%   Fs: contains frequency data

%% Extract Audio Features
orig = pwd;
cd(file_directory);

[audio, Fs] = audioread('LDC2002S28.wav'); %read the audio data
```

```
fileID = fopen('LDC2002S28.txt');

% process text
%data contains start time in data{1,1}, end time in data{1,2}, and category
%in data{1,3}
data = textscan(fileID, '%f %f A: %s %*[^\n]', 'HeaderLines', 1);
copy = cell(size(data));
%check string to clean up unnecessary lines
numSamp = size(data{1,1},1);
iter = 1;
for i = 1:numSamp
    string = data{1,3}{i};
    if not((string(1) == '[') || (string(1) == '('))
        temp = strsplit(string,',');
        copy{1,1}{iter,1} = data{1,1}(i);
        copy{1,2}{iter,1} = data{1,2}(i);
        copy{1,3}{iter,1} = temp(1);
        iter = iter + 1;
    end
end

numSamp = size(copy{1,1},1);
%extract speech from text
%ex of how to pull out a slice of audio
%test_sample = audio(42.26*Fs:43.28*Fs);
audioVector = cell(numSamp,2);
for i = 1:numSamp
    start_time = (copy{1,1}{i});
    end_time = (copy{1,2}{i});
    speech = audio(copy{1,1}{i}*Fs:copy{1,2}{i}*Fs);
    audioVector{i,1} = copy{1,3}{i};
    audioVector{i,2} = speech;
end


cd(orig);
```

```
function [ set] = extract_features( set,Fs )
%extract_features
%   extracts mfcc, pitch, deltas, mean and covariance

% calculate features

for i = 1:size(set,1)
    %disp(['length of set = ', num2str(length(set))])
    speech = set{i,2};
    %calculate mfcc
    mfcc = melfcc(speech,Fs, 'wintime',0.025, 'hoptime', 0.01);
    mm = mfcc(2:13,:);          %omit first coeff
    set{i,3} = mm;

    %calculate pitch
    [t, f0, avgF0] = pitch(speech,Fs);
    set{i,4} = f0;

    total = [mm;f0];


    %calculate delta
    d = deltas(total,5); %window size of 5
    total = [total;d];
    set{i,5} = total;
   % disp(['i = ',num2str(i)])

    %flatten the matrix
    flattened = reshape(total,[],1);
    set{i,6} = flattened;

    %calculate mean and variance
    total_matrix = [];
    temp_vector = [];
    for j = 1:5:size(total,2)
        endex = j + 10 -1;
        if endex > size(total,2)
            endex = size(total,2);
        end
        means = mean(total(:,j:endex),2);
        vars = var(total(:,j:endex),0,2);
        temp_vector = [means;vars];
        total_matrix = [total_matrix, temp_vector];
    end
    set{i,9} = total_matrix;


end

end





%Arousal Test
%% Cross-validation
results = {};
answer = [];
expected = [];
accuracies = [];
answer_confidence = [];
confusion_matrix_arousal = zeros(2,2);

correct = 0;
```

```
tests = 0;
%read in scaled data
%[label_vector, instance_matrix] = libsvmread('training3.scale');

%randomize order
new_order = randperm(length(speech_set));

arousal_set = speech_set(new_order,:);

%equalize training sets
%equalize training sets
second_set = {};
one_set = {};
rm_idx = [];
count = 0;
idx = 1;
idx2 = 1;
for i = 1:length(arousal_set)
    if unique(arousal_set{i,11} == 2)
        if count < 139
            second_set(idx,:) = arousal_set(i,:);
            rm_idx = [rm_idx i];
            count = count + 1;
            idx = idx + 1;
        end
    elseif unique(arousal_set{i,11} == 1)
        one_set(idx2,:) = arousal_set(i,:);
        idx2 = idx2 + 1;
    end
end

%remove 0 class
for i = 1:length(arousal_set)
    if i > length(arousal_set)
        break;
    end
    if unique(arousal_set{i,11} == 0)
        rm_idx = [rm_idx i];
    end
end
arousal_set(rm_idx,:) = [];
%remix
new_order = randperm(length(arousal_set));
arousal_set = arousal_set(new_order,:);

zero = 0;
one = 0;
two = 0;
for i = 1:length(arousal_set)
    if arousal_set{i,11}(1) == 0
        zero = zero + 1;
    elseif arousal_set{i,11}(1) == 1
        one = one + 1;
    else
        two = two + 1;
    end
end
```

```
display([num2str(zero),' classified as 0']);
display([num2str(one),' classified as 1']);
display([num2str(two),' classified as 2']);

%split into groups
N = 5;
start_index = 1;

i = 1;

%check to see if at end of features
for i = 1:N:length(arousal_set)
%for i = 1
    end_index = i+N-1;
    if end_index > length(arousal_set)
        end_index = length(arousal_set);
    end
    test_set = arousal_set(i:end_index,:);
    training_set = arousal_set;
    training_set(i:end_index,:) = [];


    display('Exporting data_set..');
    test_labels = vertcat(test_set{:,11});
    test_instances = vertcat(test_set{:,8});

    training_labels = vertcat(training_set{:,11});
    training_instances = vertcat(training_set{:,8});

    libsvmwrite('arousal_test_set.txt',test_labels,test_instances);

libsvmwrite('arousal_training_set.txt',training_labels,training_instances);

    %scale
    display('Scaling...');
    [status, cmdout] = unix('../libsvm-3.20/svm-scale -s arousal_scaling
arousal_training_set.txt > arousal_training_set.scale');
    if(status)
        display('scaling failed!');
        break;
    end
    [status, cmdout] = unix('../libsvm-3.20/svm-scale -r arousal_scaling
arousal_test_set.txt > arousal_test_set.scale');
    if(status)
        display('scaling failed!');
        break;
    end
    display('Scaling complete!');


    %training
    display('Training...')
    [training_labels, training_instances] =
libsvmread('arousal_training_set.scale');
    model = svmtrain(training_labels,training_instances);
    display('Trained!');
    %testing
    display('Testing...');
    [testing_labels, testing_instances] =
libsvmread('arousal_test_set.scale');
    display('Predicted!');
    [predicted_label, accuracy, decision_values] = svmpredict(testing_labels,
testing_instances, model);
    accuracies = [accuracies;accuracy(1)];
    %% Accuracy
    startdex = 1;

    for j = 1:size(test_set,1)
        enddex = startdex + size(test_set{j,9},2)-1; %sparse matrices
        group = predicted_label(startdex:enddex);
        total = 0;
        for k = 0:14
            results{j,k+1} = length(find(group == k));
            total = total + results{j,k+1};
        end
        answer = [answer; mode(group)];
        answer_confidence = [answer_confidence;
results{j,mode(group)+1}/total];
        expected = [expected; testing_labels(startdex)];

        if testing_labels(startdex) == mode(group)
            correct = correct + 1;
        end
        confusion_matrix_arousal(testing_labels(startdex),mode(group)) =
confusion_matrix_arousal(testing_labels(startdex),mode(group)) + 1;
        tests = tests + 1;
        startdex = enddex + 1;

    end
    display(['Correct: ', num2str(correct), '\n Total: ', num2str(tests)]);
end








% Valence Test
%% Cross-validation
results = {};
answer = [];
expected = [];
accuracies = [];
answer_confidence = [];
correct = 0;
tests = 0;
confusion_matrix_valence = zeros(2,2);
```

```
%read in scaled data
%[label_vector, instance_matrix] = libsvmread('training3.scale');

%randomize order
new_order = randperm(length(speech_set));

ordered_set = speech_set(new_order,:);

%equalize training sets
second_set = {};
one_set = {};
rm_idx = [];
count = 0;
idx = 1;
idx2 = 1;
for i = 1:length(ordered_set)
    if unique(ordered_set{i,7} == 2)
        if count < 90
            second_set(idx,:) = ordered_set(i,:);
            rm_idx = [rm_idx i];
            count = count + 1;
            idx = idx + 1;
        end
    elseif unique(ordered_set{i,7} == 1)
        one_set(idx2,:) = ordered_set(i,:);
        idx2 = idx2 + 1;
    end
end

%remove 0 class
for i = 1:length(ordered_set)
    if i > length(ordered_set)
        break;
    end
    if unique(ordered_set{i,7} == 0)
        rm_idx = [rm_idx i];
    end
end
ordered_set(rm_idx,:) = [];
%remix
new_order = randperm(length(ordered_set));
ordered_set = ordered_set(new_order,:);

zero = 0;
one = 0;
two = 0;
for i = 1:length(ordered_set)
    if ordered_set{i,7}(1) == 0
        zero = zero + 1;
    elseif ordered_set{i,7}(1) == 1
        one = one + 1;
    else
        two = two + 1;
    end
end
display([num2str(zero),' classified as 0']);
display([num2str(one),' classified as 1']);
display([num2str(two),' classified as 2']);

%split into groups
N = 5;
start_index = 1;

i = 1;

%check to see if at end of features
for i = 1:N:length(ordered_set)
%for i = 1
    end_index = i+N-1;
    if end_index > length(ordered_set)
        end_index = length(ordered_set);
    end
    test_set = ordered_set(i:end_index,:);
    training_set = ordered_set;
    training_set(i:end_index,:) = [];



    test_labels = vertcat(test_set{:,7});
    test_instances = vertcat(test_set{:,8});

    training_labels = vertcat(training_set{:,7});
    training_instances = vertcat(training_set{:,8});

    libsvmwrite('test_set.txt',test_labels,test_instances);
    libsvmwrite('training_set.txt',training_labels,training_instances);

    %scale
    display('Scaling...');
    [status, cmdout] = unix('../libsvm-3.20/svm-scale -s scaling
training_set.txt > training_set.scale');
    if(status)
        display('scaling failed!');
        quit;
    end
    [status, cmdout] = unix('../libsvm-3.20/svm-scale -r scaling test_set.txt
> test_set.scale');
    if(status)
        display('scaling failed!');
        quit;
    end
    display('Scaling complete!');


    %training
    display('Training...')
    [training_labels, training_instances] = libsvmread('training_set.scale');
    model = svmtrain(training_labels,training_instances);
    display('Trained!');

    %testing
    display('Testing...');
    [testing_labels, testing_instances] = libsvmread('test_set.scale');
    display('Predicted!');
    [predicted_label, accuracy, decision_values] = svmpredict(testing_labels,
testing_instances, model);
    accuracies = [accuracies;accuracy(1)];
    %% Accuracy
    startdex = 1;

    for j = 1:size(test_set,1)
        enddex = startdex + size(test_set{j,9},2)-1; %sparse matrices
        group = predicted_label(startdex:enddex);
        total = 0;
        for k = 0:14
            results{j,k+1} = length(find(group == k));
            total = total + results{j,k+1};
        end
        answer = [answer; mode(group)];
        answer_confidence = [answer_confidence;
results{j,mode(group)+1}/total];
        expected = [expected; testing_labels(startdex)];

        if testing_labels(startdex) == mode(group)
            correct = correct + 1;
        end
        confusion_matrix_valence(testing_labels(startdex),mode(group)) =
confusion_matrix_valence(testing_labels(startdex),mode(group)) + 1;
        tests = tests + 1;
        startdex = enddex + 1;

    end
    display(['Correct: ', num2str(correct), '\n Total: ', num2str(tests)]);
end
```

```
%% Voice test
recObj = audiorecorder(Fs,8,1);
disp('Start speaking.')
recordblocking(recObj, 2);
disp('End of Recording.');
y = getaudiodata(recObj);
plot(y);
voice = cell(1,2);
voice{1,1} = {'happy'};
voice{1,2} = y;
```

```
%extract features
display('Extracting features...');
voice = extract_features(voice,Fs);
display('Features extracted!');

[voice_label, voice_instance, voice] = export_as_libsvm(voice,'voice.txt');

[status, cmdout] = unix('../libsvm-3.20/svm-scale -r scaling voice.txt >
voice.scale');
if(status)
    display('scaling failed!');
    break;
end
display('Scaling complete!');
[status, cmdout] = unix('../libsvm-3.20/svm-scale -r arousal_scaling
voice.txt > arousal_voice.scale');
if(status)
    display('scaling failed!');
    break;
end
display('Scaling complete!');

%testing
display('Testing...');
[testing_labels, testing_instances] = libsvmread('voice.scale');
[arousal_testing_labels, arousal_testing_instances] =
libsvmread('arousal_voice.scale');
display('Predicting...');
[v_predicted_label, v_accuracy, v_decision_values] =
svmpredict(testing_labels, testing_instances, valence_model);
[a_predicted_label, a_accuracy, a_decision_values] =
svmpredict(arousal_testing_labels, arousal_testing_instances, arousal_model);

va_accuracy = length(find(v_predicted_label ==
mode(v_predicted_label)))/length(v_predicted_label)*100;
ar_accuracy = length(find(a_predicted_label ==
mode(a_predicted_label)))/length(a_predicted_label)*100;
display(['Valence Voted: ',num2str(mode(v_predicted_label)), ', Confidence:
',num2str(va_accuracy),'%']);
display(['Arousal Voted: ',num2str(mode(a_predicted_label)), ', Confidence:
',num2str(ar_accuracy),'%']);
```