Jump Trading Medical Simulation Controller

Design Review

ECE 445 Senior Design Project 35

Jian Chen - Tanmay Mishra - Michal Rys

TA: Justine Fortier

Feb 23, 2013

Table of Content

Content

I. INTRODUCTION

1.1 Statement of Interest

The Jump Trading Simulation and Education Center is a high-tech facility used train future medical doctors. The facility's surgery simulators are currently controlled by software running on laptops. In order to make an adjustment to the simulation, the technician must tediously navigate the software's interface with a mouse and keyboard, making on-the-fly adjustments difficult and slow. This goal of this project is to simplify such adjustments by designing a hardware interface into the system. This will be done by building a hardware controller that is easy to navigate even with minimal attention from the technician.

1.2 Benefits and Features

1.2.1 Benefits

- Provides an ergonomic solution to making on-the-fly adjustments of simulation parameters (heart rate, airway respiratory rate, diastolic blood pressure, systolic blood pressure, oxygen saturation, adjustment time period)
- Eliminates the use of software GUI as a form of making adjustments to the aforementioned parameters
- Minimizes the amount of attention the technician focuses on physically making an adjustment to a parameter.
- Maximises the amount of attention the technician focuses on the students and simulation.

1.2.2 Features

- Large wall-mounted backlit LCD unit for easy glancing at parameter values
- Two adjustment knobs for quick changes to a parameter's target value and time period of adjustment
- Precision switches that allow changes to granularity of parameter adjustments
- Ability to interface with existing Laerdal software used to control simulations

II. DESIGN

2.1 Block Diagrams

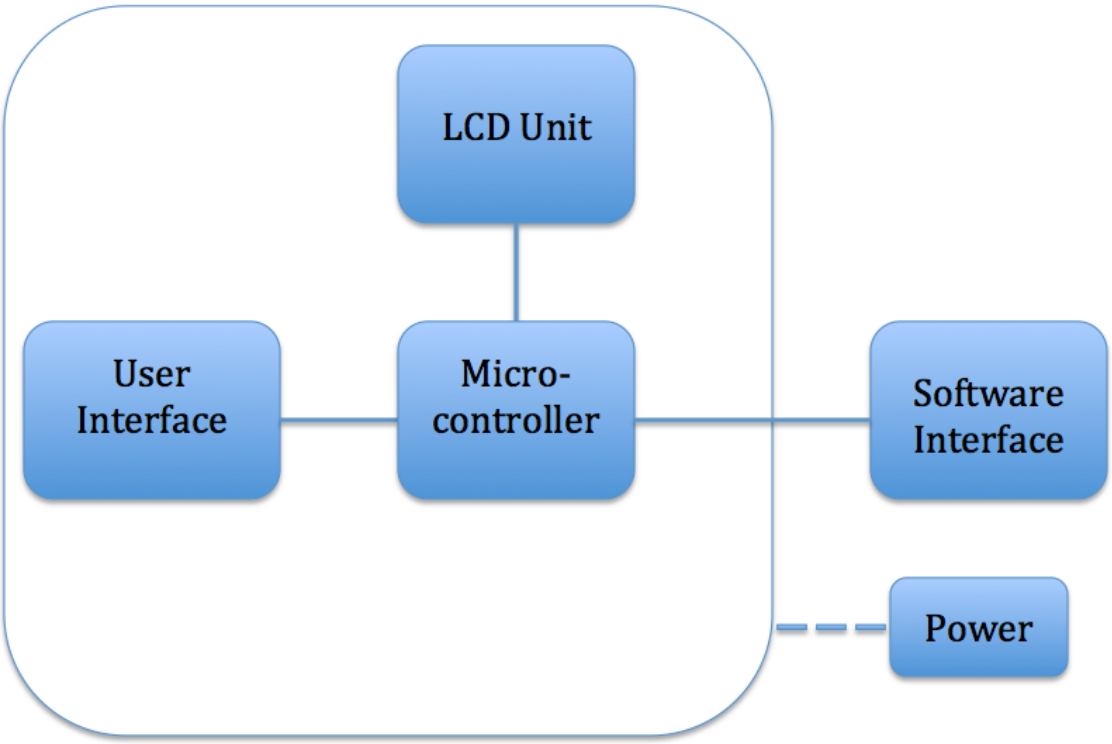

Figure 1 Top Level Block Diagram

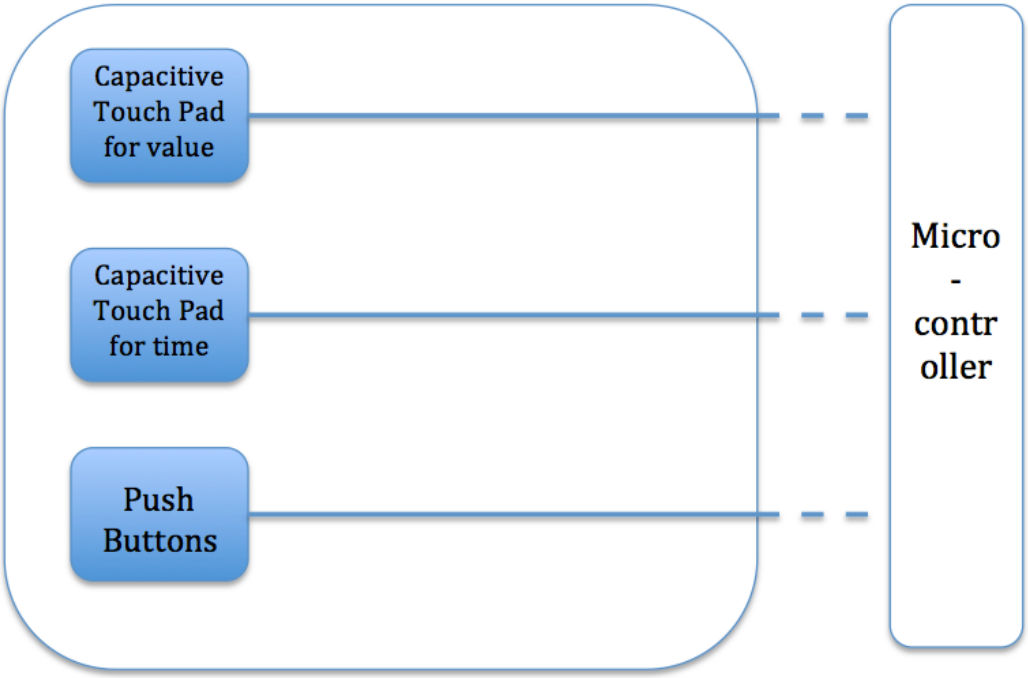

Figure 2 User Interface

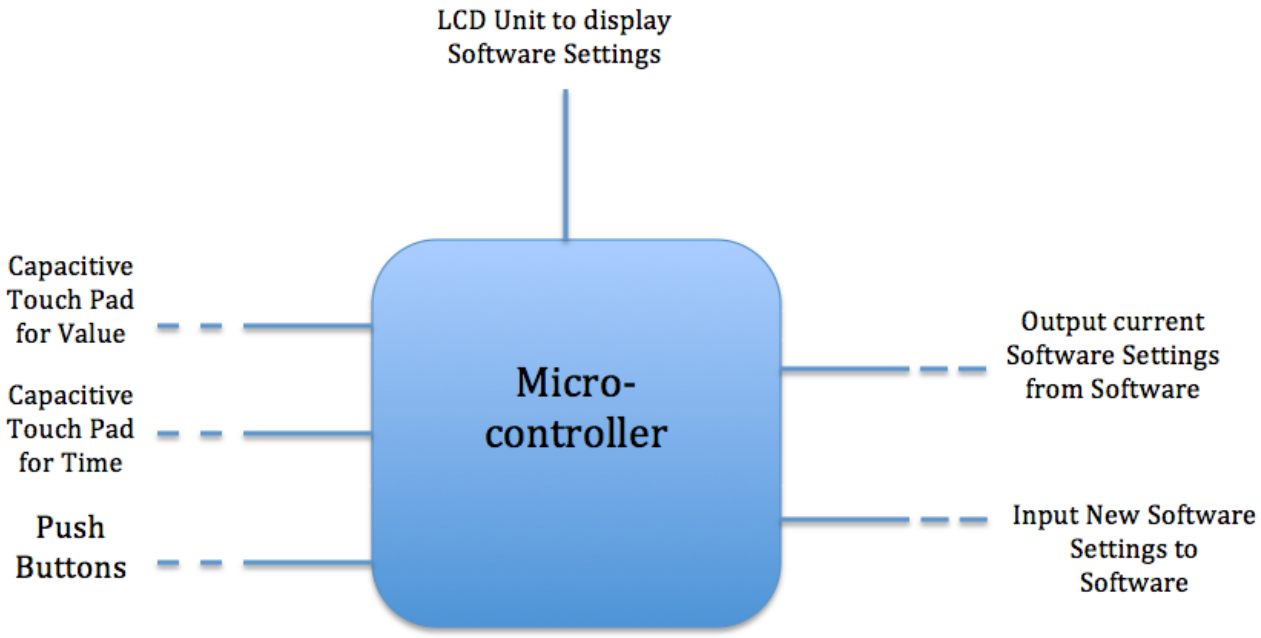

Figure 3 Micro-controller

2.2 BLOCK DESCRIPTIONS

2.2.1 User Interface:

Push buttons will be used for parameter selection. Every adjustable parameter will have a dedicated corresponding button on the device. The device will have 5 such buttons, one each for heart rate, airway respiratory rate (awrr), diastolic blood pressure, systolic blood pressure, and oxygen saturation.

Every parameter change in the Laerdal simulation software requires two values: the target parameter value, and the adjustment time period over which the value will gradually reach the target value. These will be inputted by the user through two capacitive touch surfaces running in parallel, one each for parameter value and time period. Each capacitive touch surface will accept single-finger swipes where movements in the Y axis directions of the sensor will correspond to coarse-grained adjustments, and movements in X axis directions of the sensor will correspond to fine-grained adjustments.

The digital circuitry of our device will run at 3.3 volts (V). The Freescale mpr121 capacitive sensor selected for this project is a 5V sensor. A simple voltage divider will be used on the digital output of each sensor to interface with our system.
Value submission will be controlled by by a single "Submit" button on the device. If the controller is in a value adjustment state, then this button will cause the microcontroller to begin pushing the values to the Laerdal software.
An illuminated latching pushbutton switch will allow the simulation to be paused.
All User Interface components feed directly into digital and analog pins on the microcontroller.

2.2.1 Microcontroller:

The device will use an Arduino Due microcontroller. The Arduino Due runs an Atmel SAM3X8E ARM Cortex-M3 processor and features 54 digital and 12 analog input/output (i/o) pins and an 84 MHz clock speed. All digital circuitry associated with the microcontroller will run at 3.3V. The microcontroller will receive data from two sources: parameter values from the User Interface over digital and analog i/o lines, and Laerdal software status over the Universal Serial Bus (USB) mounted on the board from the software interface. After processing the input, the microcontroller will output values to the LCD Unit over digital i/o pins, and send commands to the Software Interface over the USB.

2.2.3 LCD Unit:

The LCD Unit will consist of 4 separate 4 lines*20 characters LCD's. Two LCD's will be dedicated to displaying the values of all parameters currently set in the Laerdal software. This will allow the technician controlling the simulation to easily view all current values simultaneously. If the device is in a parameter-setting state, then the other two LCD's will display a live value and adjustment time of the parameter being set by the technician through the user interface. The LCD Unit will receive all display data from the Microcontroller.

2.2.4 Software Interface:

The Software Interface will facilitate all communication between the device and the Laerdal software. The Software interface will invoke various tools available through the Laerdal SDK to perform two main tasks: updating

parameters as inputted by the user, and sending update packets to the microcontroller whenever a parameter is changed within the Laerdal software itself. This two-way communication allows the device to display the freshest possible values on the device, and to actually perform the update of a parameter in the system. The communication interface will be implemented as a C# windows application using a USB connection to communicate with the microcontroller.

2.2.5 Power Unit

Overall power is supplied by a 120 V plug into the wall. Then the adapter converts 120-Volt AC into 12-Volt DC and feed into the Arduino microcontroller. On the Arduino, there are a 5-Volt output pin and a 3.3-Volt output pin. 5-Volt output pin is used to power up User Interface (Capacitive Touch Pad, latch buttons, and regular buttons), 3.3-Volt pin is used to power up LCD Unit. The total current can't be more than 4.5-Amp as that is the maximum current supply by the power unit. The total current draw calculation is:

Each 5-volt pin has a maximum draw of current 40mA, and the 3.3-Volt pin draws 50mA current. We are using 2 5-volt pins (2 for Capacitive Touch Wheel), we are also using 14 3.3-volt pins(10 for buttons, and 4 for 4 LCDs). One LED is used on the User Interface to indicate that simulation is in pause. It draws current 0.5mA.

Thus:

I(Arduino 5-volt pins) = 2*40mA = 80mA;

I(3.3-Volt pin) = 14*50mA = 700mA;

I(LED) = 0.5mA;

I(total) = I(Arduino 5-volt pins) + I(3.3-Volt pin) + I(LED) = 780.5mA.

All calculation is based on theoretical maximum current draw, in practical case, the current should be less than 780.5mA.

2.3 Schematics and Calculation

2.3.1 Schematic for overall system

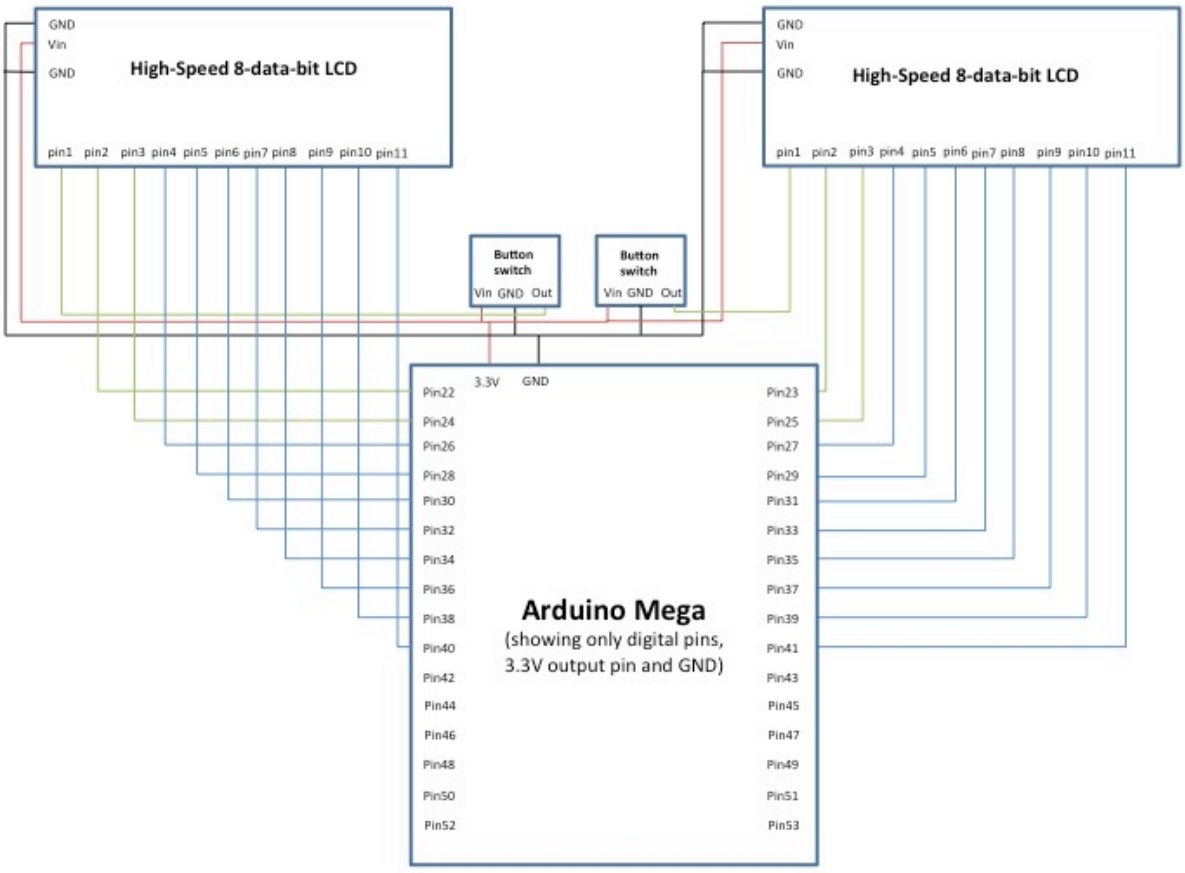

Figure 4: Schematic for high-speed 8-data-bit LCD

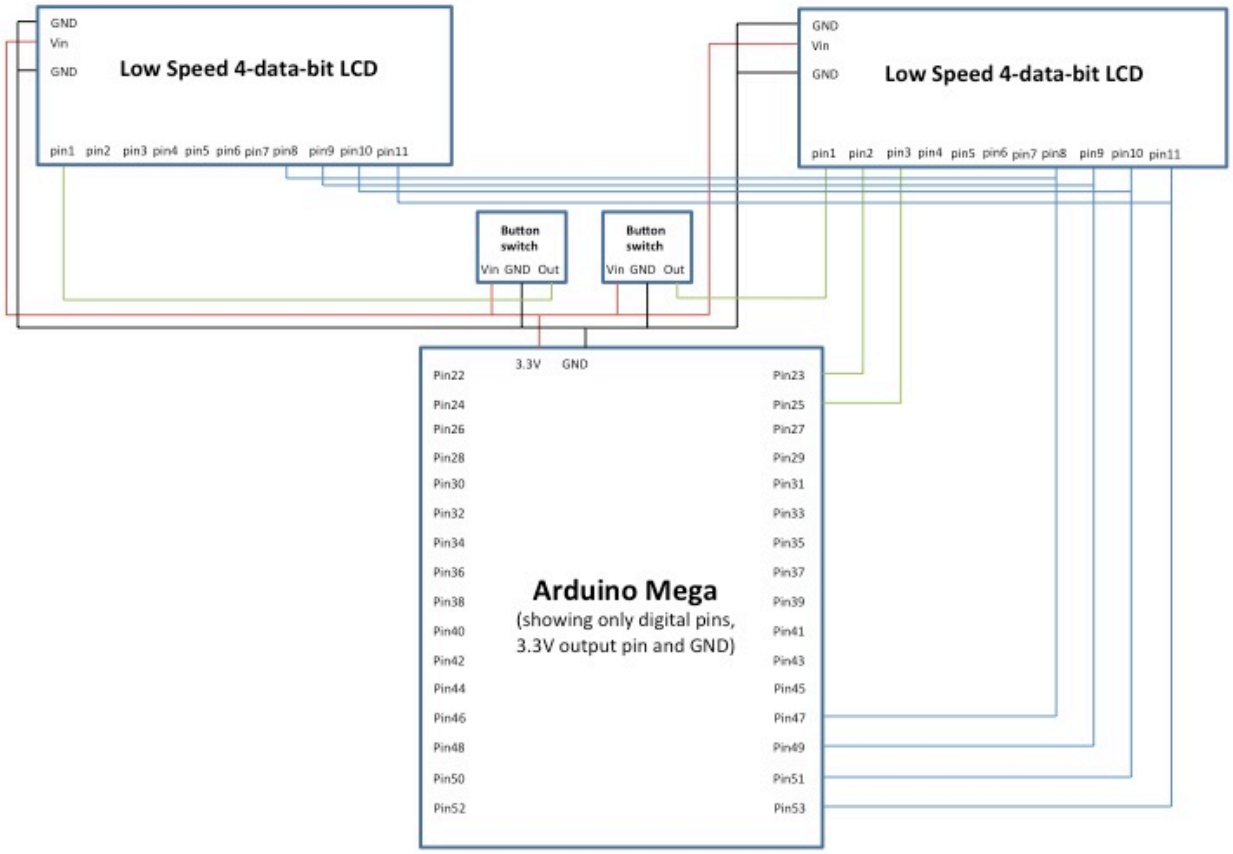

Figure 5: Schematic for low speed 4-data-bid LCD using shared data lines and separate Enable Lines

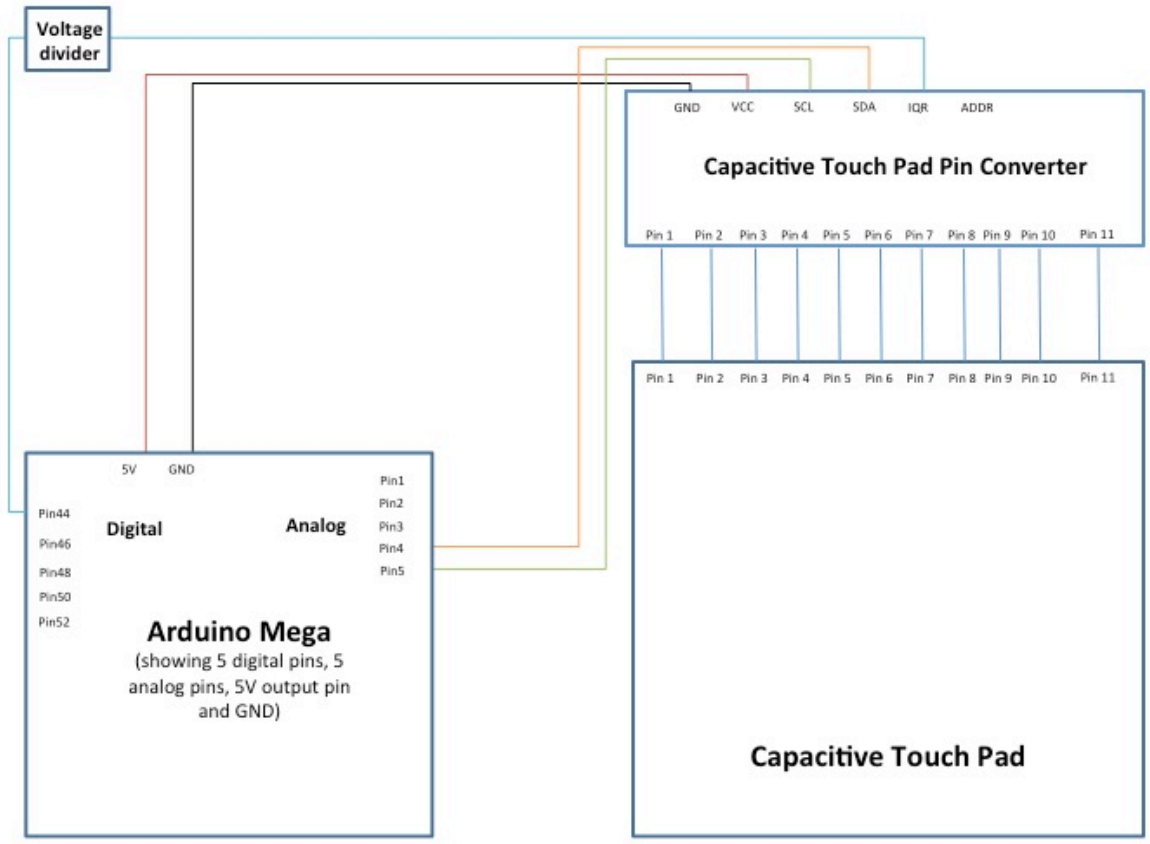

Figure 6: Schematic for Capacitive Touch Pad (2 Capacitive Touch Pads will be connected with similar wiring)

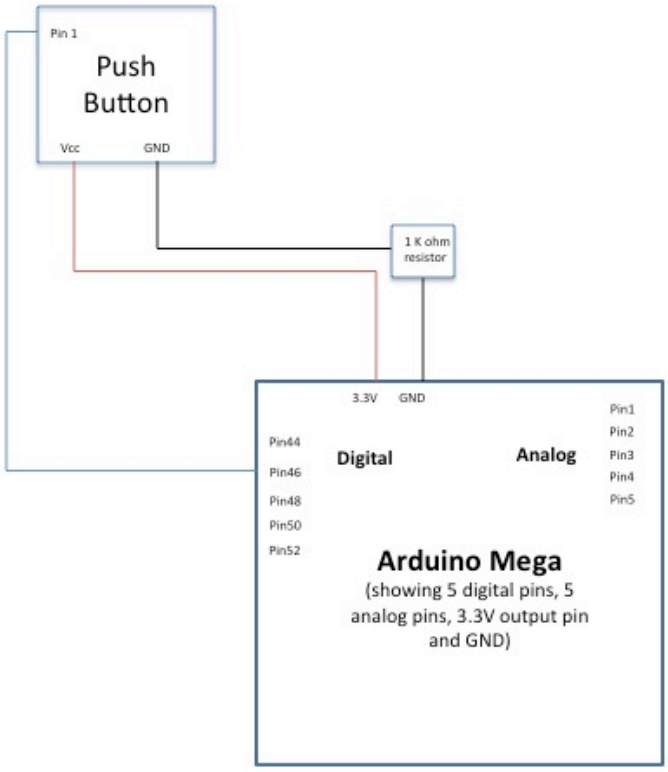

Figure 7: Schematic for pushbutton with pull-down resistor (8 buttons and 1 pushbutton switch will be connected with similar wiring)

2.3.2 Schematic and Calculation for voltage divider

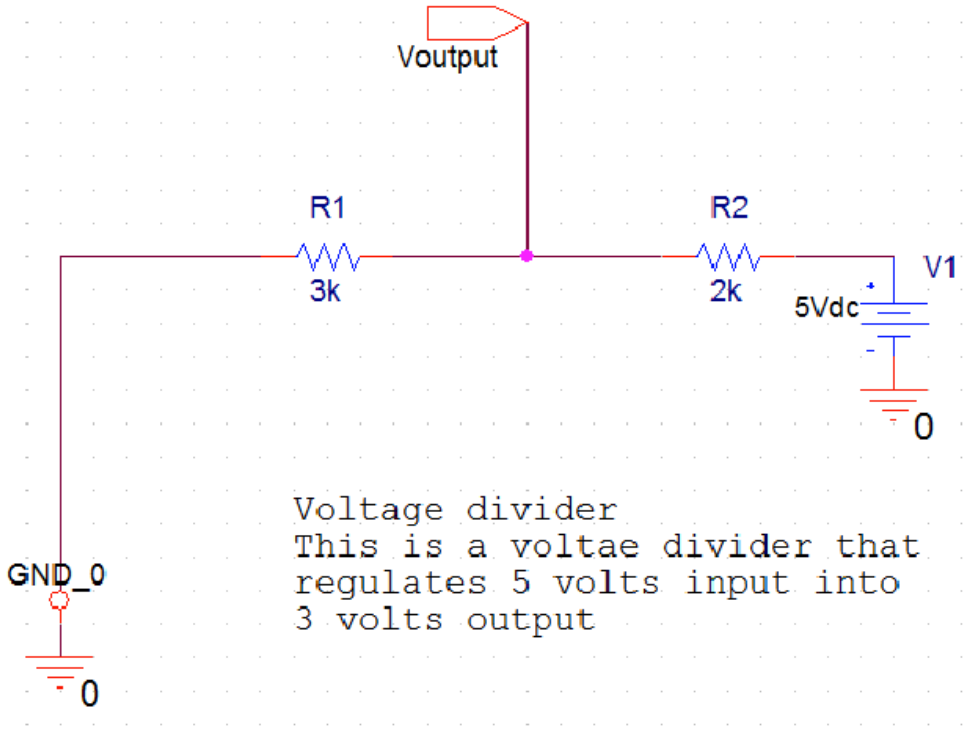

Figure 8

Voltage Divider Calculation:

This circuit is used to regulate the input voltage to the new Arduino micro-controller. It decrease a 5-volt input voltage to 3-volt and feed into the Arduino VCC pin.
The basic calculation is as follows:

Voutput = 5V * 3KOhm / (3+2)KOhm = 3 V

2.3.3 Flow Charts for System Software

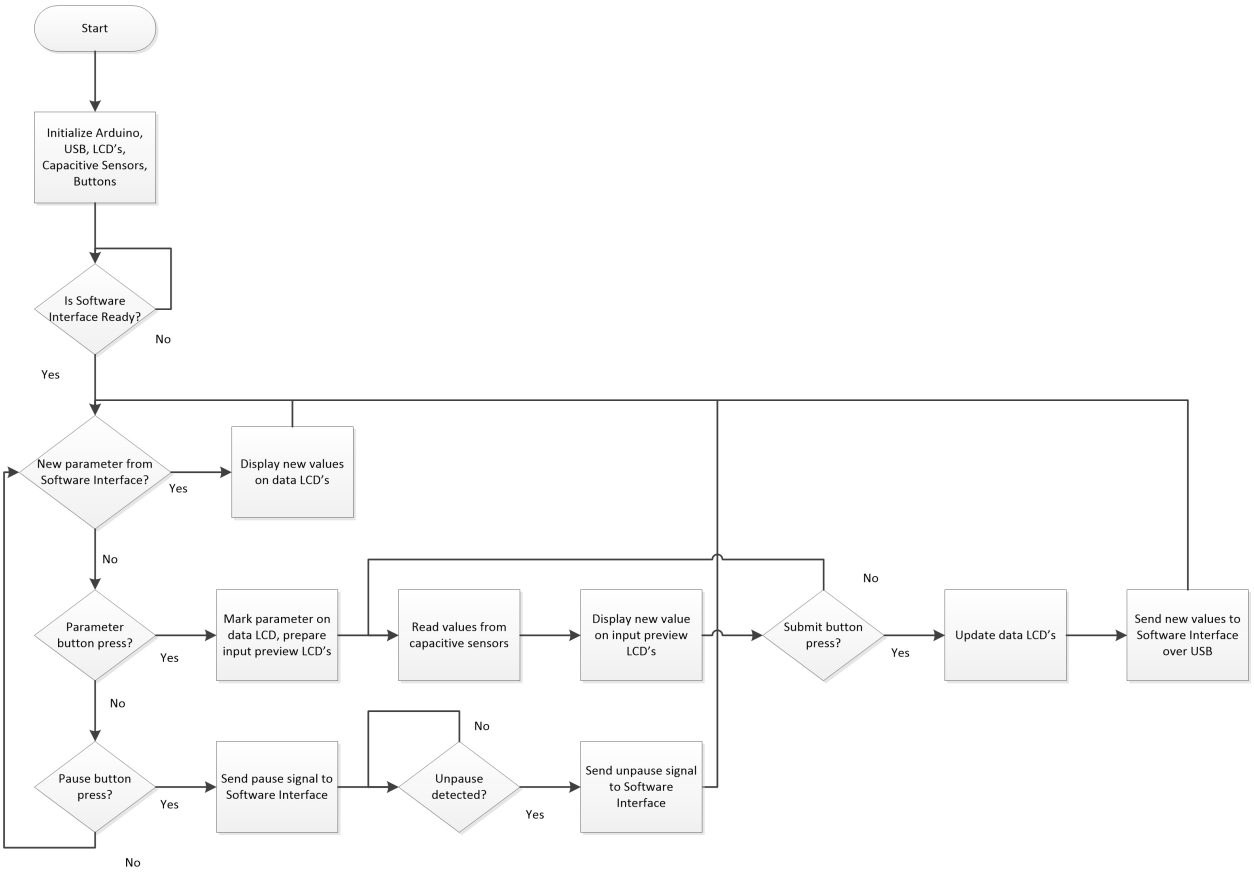

Figure 9: Flow chart for microcontroller software

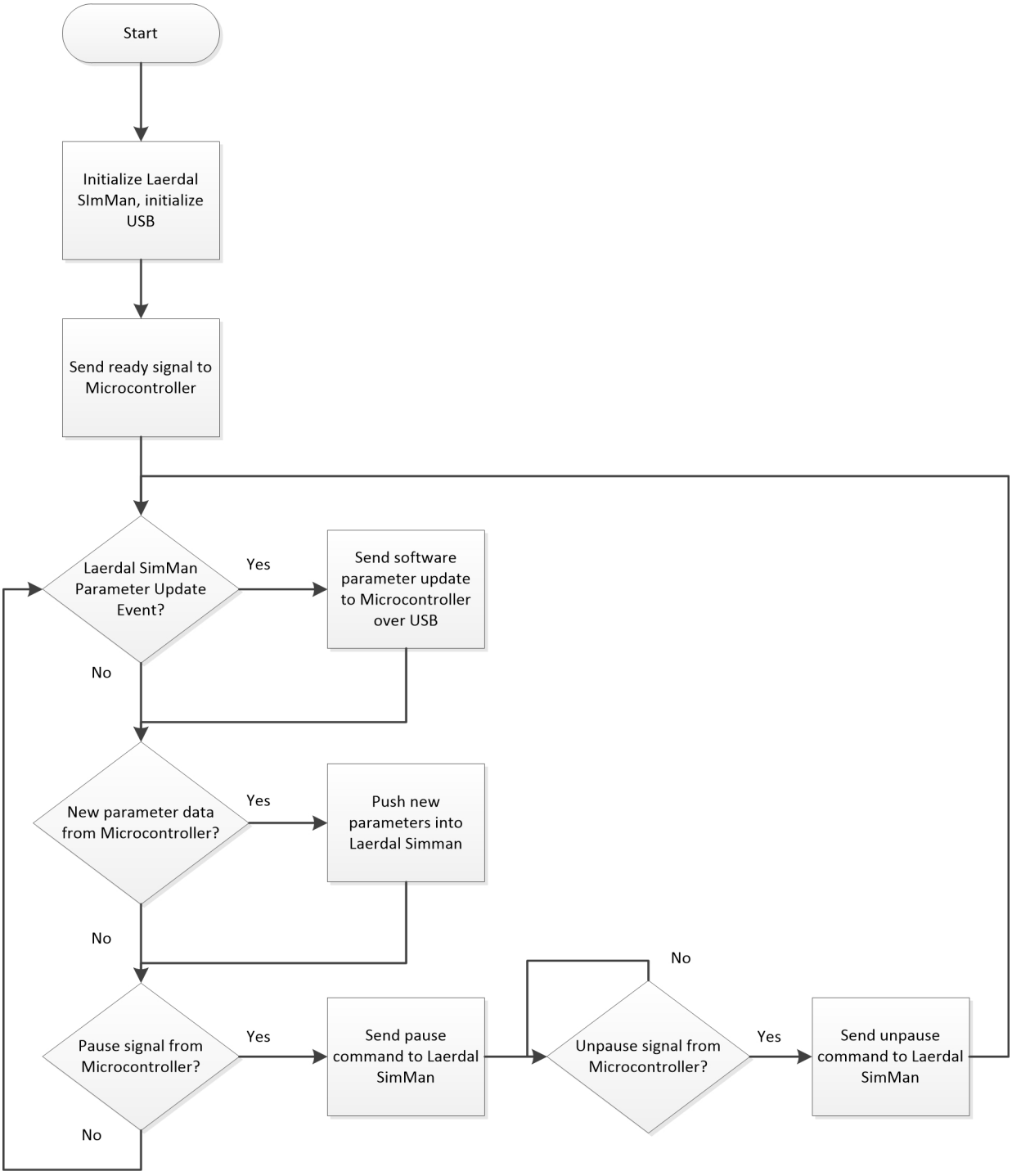

Figure 10: Flow chart for Software Interface

III REQUIREMENTS & VERIFICATION

3.1 VERIFICATION PROCEDURE

Performance Requirements	Testing/ Verification
User Interface 1. Each of the 7 buttons should function properly as on/off buttons. 1.1. 6 of the buttons should be edge-triggered and they are push buttons. 1.2. 1 of the buttons should be edge-triggered and stay high as long as it is pressed. They are latching push button switch. 2. Capacitive Touch Surface for Value should function properly. 3. Capacitive Touch Surface for Time should function properly. 4. LED should be lighted when PAUSE button is pressed, and should be turned off when the PAUSE button is un-pressed. 5. The output voltage of Capacitive Touch Surface for Value's digital pins should be regulated to 3.3 volts. (The new Arduino digital pins only take 3.3 Volts input) 6. The output voltage of Capacitive Touch Surface for Time's digital pins should be regulated to 3.3 volts. (The new Arduino digital pins only take 3.3 Volts input)	1. Using a breadboard, feeding it with 5V voltage, connecting a button in series with a 1.000KOhm±100Ohm resister, using a Multimeter to measure the voltage across the resister. 1.1. While each the 6 of the push buttons are pushed, the Multimeter should read 3.3±0.3 V. When button is released, the Multimeter should read 0+0.1 V. 1.2. When this one of the button is pressed once, the button should stay latched. The Multimeter should read 0.3±0.1 V; when it is pressed second time, the button should be un-latched. The Multimeter should read 0+0.1 V. 2. Connect Capacitive Touch Surface for Value to a 12pin-to-6pin converter, feed the VCC pin with 5 V DC voltage, connect GND pin to the Ground, use a Multimeter to test the voltage of the rest 4 pins when touch different grid on the Touch sensing part of the device. 3. Connect Capacitive Touch Surface for Time to a 12pin-to-6pin converter, feed the VCC pin with 5 V DC voltage, connect GND pin to the Ground, use a Multimeter to test the voltage of the rest 4 pins when touch different grid on the Touch sensing part of the device. 4. Connect the LED button in series

	with button on the breadboard, connect the breadboard with 3.3V DC voltage and ground accordingly. When button is pressed, LED should light up, when button is un-pressed, the light should turn off. 5. Following the same procedures of step2, but extend the output to pass it through a voltage divider, regulate output from 5V to 3.00V±.33V. When the corresponding pin is high, the voltage reading from the Multimeter should be 3.00±.33 Volts. 6. Following the same procedures of step3, but extend the output to pass it through a voltage divider, regulate output from 5V to 3.00±.33V. When the corresponding pin is high, the voltage reading from the Multimeter should be 3 Volts.
LCD Unit 1. When turned on, LCD_names1 must stably runs on 3.3 ± 10% volts. 2. When turned on, LCD_name2 must stably runs on 3.3 ± 10% volts. 3. When turned on, LCD_values must stably runs on 3.3 ± 10% volts. 4. When turned on, LCD_time must stably runs on 3.3 ± 10% volts. 5. LCD 's display delay from the input submition must be less than 0.3 second.	1. Power the LCD, use a Multimeter to measure the power pin of the LCD, the reading should be 3.30 ± 0.33 Volts. 2. Power the LCD, use a Multimeter to measure the power pin of the LCD, the reading should be 3.30 ± 0.33 Volts. 3. Power the LCD, use a Multimeter to measure the power pin of the LCD, the reading should be 3.30 ± 0.33 Volts. 4. Power the LCD, use a Multimeter to measure the

	power pin of the LCD, the reading should be 3.3 ± 0.33 Volts. 5. Connect the LCD to Arduino, connect Arduino to User Interface, use a oscilloscope to measure the voltages of LCD input pin and User Interface output pin, read the time difference of two spikes, this difference should be less then 0.3 seconds.
<u>Micro-controller</u> Inputs: 1. Each of the 7 buttons should map to a specific parameter we are going to change. 2. Capacitive Touch Surface for Value should behave according to the design, with moving up mapping to 1 unit increase value of the chosen parameter; moving down mapping to 1 unit decrease value of the chosen parameter; moving right mapping to 0.1 unit increase value of the chosen parameter; moving left mapping to 0.1 unit decrease of the chosen parameter. 3. Capacitive Touch Surface for Time should behave according to the design, with moving up mapping to 1 unit increase time of the chosen parameter; moving down mapping to 1 unit decrease time of the chosen parameter; moving right	Inputs: 1. Connect the buttons through wires to the pins on the Arduino board. Using a Arduino test code on the computer to read out the value when different buttons were pressed: pin1 (Heart Rate): correspond variable value is assigned to 1; pin2 (Respiratory Rate/ awRR): correspond variable value is assigned to 1; pin3 Oxygen Saturation: correspond variable value is assigned to 1; pin4 Blood Pressure (Diastolic): correspond variable value is assigned to 1; pin5 Blood Pressure (Systolic): correspond variable value is assigned to 1; pin6 Pause: correspond variable value is assigned to 1; pin7 Submit/ Save: correspond variable value is assigned to 1; 2. Connect the Capacitive Surface for Value through wires to the pins on the Arduino board. Using a Arduino test code on the computer to read

mapping to 0.1 unit increase time of the chosen parameter; moving left mapping to 0.1 unit decrease time of the chosen parameter. 4. Arduino should correctly present the input value from SimMan software through USB cable connection. Outputs: 1. Arduino should output a 6-pin output to the LCD_names1 to display the parameter names. 2. Arduino should output a 6-pin output to the LCD_names2 to display the parameter names. 3. Arduino should output a 6-pin output to the LCD_values to display the parameter values. 4. Arduino should output a 6-pin output to the LCD_time to display the parameter evolution time. 5. 3V3 VCC pin outputs 3.00V ± .33V 6. 5 VCC pin outputs 5.0V ± .5V	out the value when different gestures were performed: Sweeping up-to-down by one sensing grid: value decrease by 1 unit. Sweeping down-to-up by one sensing grid: value increase by 1 unit. Sweeping left-to-right by one sensing grid: value increase by 0.1 units. Sweeping right-to-left by one sensing grid: value decrease by 0.1 units. 3. Connect the Capacitive Surface for Time through wires to the pins on the Arduino board. Using a Arduino test code on the computer to read out the value when different gestures were performed: Sweeping up-to-down by one sensing grid: value decrease by 1 unit. Sweeping down-to-up by one sensing grid: value increase by 1 unit. Sweeping left-to-right by one sensing grid: value increase by 0.1 units. Sweeping right-to-left by one sensing grid: value decrease by 0.1 units. 4. Arduino is connected to the computer (running SimMan) through USB cable, a Arduino test code is going to test and display the value received from the SimMan software on the test computer. When change heart rate from 80 to 90, the reading pulling out from the SimMan is also going to update from 80 to 90.

	Outputs: The five pins for the data line of the 6-pin digital lines are shared by the 4 LCDs, while the 6th control line would be used to select between different LCDs to change display. 1. When one of the 'Heart Rate' 'Respiratory Rate/ awRR' 'Blood Pressure (Diastolic)' 'Blood Pressure (Systolic)' is being selected, the LCD_names1 is going to be selected, and the corresponding parameter is going to be marked with a little '*' beside it. 2. When 'Oxygen Saturation' is being selected, the LCD_names2 is going to be selected, and the corresponding parameter is going to be marked with a '*' beside it. (Three more parameters could be implemented for future expansion). 3. When one of the parameters was chosen, the LCD_values should be selected, and start to display the selected value. 4. When one of the parameters was chosen, the LCD_time should be selected, and start to display the selected value's evolution time. 5. When measured with a multimeter, the voltage drop between VCC 3V3 and ground is within the voltage tolerance. The voltage should remain within the tolerance both with no external devices hooked up to the microcontroller, and under full load.

	6. When measured with a multimeter, the voltage drop between VCC 5V and ground is within the voltage tolerance. The voltage should remain within the tolerance both with no external devices hooked up to the microcontroller, and under full load.
Power Supply 1. Power supply should be regulated between 7-12 Volts, 1000±100 mA. 2. Power from wall should be 120 ± 12V AC	1. Plug the power cord into the wall, measure the output voltage use a Multimeter. It should read 9.0±0.9 Volts; measure the current using a Multimeter. It should read 1000±100mA. 2. Measure the voltage with a multimeter and check if it is within the acceptable bounds.
Software Interface 1. Parameter updates from the microcontroller should be received through the usb connection. 2. Parameter updates from the microntroller should be properly parsed and communicated to the Laerdal SimMan software. 3. Pause signals from the microcontroller should be received through the usb connection. 4. Pause signals from the microntroller should be properly parsed and communicated to the Laerdal SimMan software.	1. Test code will be used to display the values received from the microcontroller, without sending them to the Laerdal SimMan software. These values will be checked against the input into the microcontroller to confirm that they are the same. 2. Test code will be used to pass parameters to the Laerdal SimMan software without input from the Microcontroller. These values will be checked both ends to verify that they are the same after the code executes. 3. Test code will be used to display when a pause signal is received from the microcontroller, without sending it to the Laerdal

	SimMan software. One pause signal sent by the microcontroller should correspond to one pause signal received by the software interface. 4. Test code will be used to send pause signals to the Laerdal SimMan software without input from the Microcontroller.

3.2 TOLERANCE ANALYSIS

Using the touch panel as user input runs the risk of accidental adjustments. For example, the technician might unintentionally introduce some upward motion into a horizontal swipe across the screen. We will design our device to maximize the tolerance of unintentional inputs from the user. We will first determine the average error a person tends to have when touching a screen, and keep these in mind during our implementation. After the design is complete, we will test the gesture tolerances by methodically performing them with a stylus.

3.3 Ethical Issues

Our project is to create a front user input device for the medical simulation system used by Jump Trading, thus to help the simulator smoothly transition between the process of observing (when the simulator is looking out horizontally) and parameters-adjusting (when the user is looking down to the device holding in one's hands). Several IEEE Code of Ethics would be addressed as following:

1 To accept responsibility in making decisions consistent with the safety, health, and	The Jump Trading Simulation and Education Center offers medical training programs by simulating

welfare of the public, and to disclose promptly factors that might endanger the public or the environment;	real-time medical scenario thus training medical students who is better prepared for the real-life practice in their future. Our device make it easier for the simulator to change parameters during the process of simulation. Free their eyes from looking at the monitor each time changing the parameters. Thus they could better focus on the on-going simulation. Our project serves for the safety, health, and public welfare.
2. To avoid real or perceived conflicts of interest whenever possible, and to disclose them to affected parties when they do exist;	We are keeping a frequent communication channel with Jump Trading through emails, and phone calls. Through bi-weekly updating and email exchange, our project design is built based on our natural understanding. If a conflict does occur we will disclose them as soon as possible. We are also working on open source components and software, trying to avoid any potential patent infringement by other third parties.
3. To be honest and realistic in stating claims or estimates based on available data;	The verification plan will be set as detailed as possible, and the real verification process will be conducted strictly according to plan with everything recorded truthfully and correctly.
4. To reject bribery in all its forms;	Bribery would be an unlikely event, in the case it happened, we will reject any kind of bribery and report the incident.
5. To improve the understanding of technology; it's appropriate application, and potential consequences;	By searching and researching on different technologies, we are learning to use different technologies to approach problem.

	By comparing the cons and pros, we are learning different technologies and their characteristics. We are reaching out for the newest and most advanced technology to fit in our design.
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 will carefully evaluate every design, choose the one physically feasible to build. If it was not realistic to build, we will report to both parties and amend the design.
7. To seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others;	We will open to any suggestions advice, we are welcome any criticism, and will acknowledge and correct errors. All the citations and acknowledgement would be referenced in the documents.
8. To treat fairly all persons regardless of such factors as race, religion, gender, disability, age, or national origin;	We will do every effort to keep the equality of our working environment. No racial, religious, gender or any kind of bias would be tolerated in the team.
9. To avoid injuring others, their property, reputation, or employment by false or malicious action;	We will strictly follow all the lab safety codes and regulations. We will also keep a high self-discipline, respect each other and conduct our behavior lawfully.
10. To assist colleagues and co-workers in their professional development and to support them in following this code of ethics.	We are committed to every team member in helping with each other's academic and professional development.

IV. COST & SCHEDULE

4.1 Cost Analysis

4.1.1 LABOR:

Member	$/hour	# of weeks	Hours/week	Total hours	Subtotal	(x2.5)
Jian	50	12	15	180	9000	22,500
Michal	50	12	15	180	9000	22,500
Tanmay	50	12	15	180	9000	22,500
					Total: $ 67,500	

4.1.2 PARTS:

Name	**Cost Each**	**Quantity Needed**	**Total Cost**
Arduino Due	48.45	1	48.45
Capacitive Touch Kit For Arduino	18.50	2	37.00
small push buttons	n/a	8	5.95
illuminated latching pushbutton switch	3.95	1	3.95
Large button	3.95	1	3.95
3.3 V backlit lcd (20x4)	21.95	4	87.80

Arduino Power Supply	6.95	1	6.95
2m micro usb cable	8.60	1	8.60
potentiometer (for controlling contrast on lcd)	1.25	4	5.00
Logic Level Converter	1.95	2	3.90
Chassis	50	1	50

- **GRAND TOTAL = LABOR + PARTS = $67761.55**

4.2 SCHEDULE

	Jian	**Tanmay**	**Michal**
2/3/2013	Confirm Jump Trading's requirement for the design	Work on project proposal	Work On project proposal
2/10/2013	Finalize options for Communication Device and Power	Finalize options of user input, order microcontroller and touch screen; start learning microcontroller	Confirm Jump Trading's requirements for the design, establish budget constraints with Jump Trading, acquire software SDK from Jump

		programming	Trading; create repository for project code
2/17/2013	Testing input and power option on breadboard; design user output	Implement microcontroller code that reacts to touch input	Learn how to use Laerdal SDK, design software interface
2/24/2013	Implement and test Communication Device	Implement code that converts gestures into parameter values	Prepare for design review
3/3/2013	Prototype user output; debug and optimize hardware	Debug/Optimize microcontroller code	Implement software interface
3/10/2013	Design PCB	Preliminary design for Chassis	Debug/Optimize software interface
3/17/2013	Spring Break	Spring Break	Spring Break
3/24/2013	Fix issues in prototype device	Assemble PCB	Finalize Design of Chassis
3/31/2013	Assemble Device in Chassis	Ensure Completion	Fix remaining issues
4/7/2013	Fix Remaining Issues	Verification Testing	Tolerance Analysis
4/14/2013	Prepare Demo	Prepare Presentation	Prepare Paper
4/21/2013	Demo	Presentation	Final Paper

4/28/ 2013	Presentation	Final Paper	Check-In Supplies

V. Safety Statement

5.1 Identify Safety Risk

Working with electricity would always bring extra risk to the work environment. As the project is powered by 120 VAC source, transferring the power would be a noticeable safety checkpoint. A 120 VAC power source has the potential to shock human as well as to cause fire damage.
Working in the dry winter, static electricity could have potential of hurting human and damaging circuit.
The project uses LCD to display information. LCD may contain mercury that is a health hazard to human beings.

5.2 Safety Code

Given by the identified safety risk, every team member working at our project is required to:

1. Dry your hands before starting the project, preventing electrical shock or damaging the circuit.
2. Eliminate static electricity on your body, preventing electrical shock as well as to prevent damage the circuit.
3. Check the cable plug-in to the wall, make sure no wire is exposed, and the connection is firm and tight. Also check no any part of the wire is exposed in the water.
4. Check the LCD units, make sure the LCD is in good quality, no package damaging, no leaking, no inner parts exposed outside.

Every team member should commit to obey the safety code. Keep a safe working environment, and follow safe procedure.

VI. References

1. "Arduino - Analog Input." Arduino - AnalogInput. N.p., n.d. Web. 30 Sept. 2012. <http://arduino.cc/en/Tutorial/AnalogInput>.

2. "Arduino - ArduinoBoardUno." Arduino - ArduinoBoardUno. N.p., n.d. Web. 30 Sept. 2012. <http://arduino.cc/en/Main/ArduinoBoardUno>.

3. "Arduino." Build Your Own. N.p., n.d. Web. 30 Sept. 2012. <http://www.instructables.com/id/Build-Your-Own-Arduino/>.

4. "Arduino Character LCD Tutorial." Arduino Character LCD Tutorial. N.p., n.d. Web. 30 Sept. 2012. Sept. 2012. <http://www.hacktronics.com/Tutorials/arduino-character-lcd-tutorial.html>.

5. "Setting up an Arduino on a Breadboard." Physical Computing at ITP. N.p., n.d. Web. 30 <http://itp.nyu.edu/physcomp/Tutorials/ArduinoBreadboard>.