Appendix A Requirement and Verification Table

Table A.1 shows the performance requirements and verifications procedures for each block of the project.

Table A.1: System Requirements and Verifications

Performance Requirement	Verification	Verification status (Y or N)
User Interface		
The voltage at a particular position on the rotary switch should differ from its neighbors by 1V+/-20%	The switch will be rotated into each position and we will measure the output voltage with a Multimeter. These are the ranges of values for each position: Position 1 : 4.8 -5.2 volts Position 2 : 3.8 -4.2 volts Position 3 : 2.8 -3.2 volts Position 4 : 1.8 -2.2 volts Position 5 : 0.8 -1.2 volts Position 6 : 0.0 -0.2 volts	Y
Each specific button on the keypad needs to send a distinct voltage level to the Arduino. The voltage levels are TBD, since we need to build the circuit to determine what voltage each key outputs.	The keypad output will be connected to a Multimeter. The keys will be pressed one at a time and the voltage will be measured and recorded. The voltage levels for each key will be distinct.	Y
Push buttons must output 5V+/-20% when pressed, 0V+/-20% when released.	The output of the switch will be connected to a Multimeter and the button will be pressed then released. When pressed, the Multimeter will read between 5V+/-20%. When the button is released, the Multimeter will read 0V+/-20%.	Y
Rotary encoder will output Gray code	Connect the output of the rotary encoder to the	Y

	oscilloscope. Rotate clockwise and then counterclockwise. The waveform will output Gray code which will change based on rotation direction. The waveforms will match the waveforms shown on the datasheet.	
LCD		
LCD display must refresh in less than 250ms and needs to properly display data	Connect the LCD to the Arduino and connect the output of the LCD to an oscilloscope. Set up the Arduino so that it prompts the LCD to cycle through characters. The oscilloscope will show voltage spikes when the characters change. Measure the time between voltage peaks to determine how quickly the LCD refreshed. This will be less than 250ms.	Y
LCD must constantly have 5V+/-10% for power	Power the LCD and measure the voltage at the power pin with a Multimeter. The voltage will be 5V+/-10%.	Y
Microcontroller		
Microcontroller needs to map keypad voltage presses to the corresponding key value	Write code on Arduino that maps voltage levels to specific number or character. The code will output serial data to the PC and display the values using Arduino software. Input the specific voltages directly to the Arduino input pin using the digital power supply. The values displayed on the PC will match the key pressed on the keypad.	Y
Will map switch voltage to proper	Write Arduino code that	Y

switch position	interprets specific voltage level as switch positions. The code will output serial data to the PC and display the values using Arduino software. Connect the specific voltage levels directly to the Arduino input pin using a digital power supply. The values displayed on the PC will correspond to the position that the switch is in.	
Will determine which button is pressed and be able to determine which position it is in (e.g. 0V for off and 5V+/-20% for on)	Write code on the Arduino that interprets if a button is pressed. The code will output serial data to the PC and display the values using Arduino software. Connect 5V and then GND directly to the Arduino input pin. The PC will display a 1 if the Arduino input is 5V and a 0 if the Arduino input is GND.	Y
Direction of rotation for the rotary encoder will be determined by the Gray code input	Write Arduino code that interprets grey code. The code will output serial data to the PC and display the values using Arduino software. Connect the output of the rotary encoder directly to the Arduino input pin. Rotate the knob clockwise and then counter-clockwise. The PC will display “clockwise” for clockwise rotation and “counter-clockwise” for counter-clockwise rotation.	Y
Microcontroller will transmit data via USB to PC	Connect the Arduino to the USB interface and connect the USB interface to the PC. Write Arduino code that outputs serial data. The PC will show the same data.	Y

Microcontroller will transmit data to LCD	Connect the LCD to the Arduino. Write Arduino code that displays “Hello World”. The LCD display will say “Hello World”.	Y
Power supply		
Power supply will provide 12V+/-20%	Plug power supply into the wall. Connect the output of the power supply to a breadboard. Use a Multimeter to measure the output voltage. The Multimeter will read 12V+/-20%.	Y
USB Interface		
USB will transfer data reliably without errors from the microcontroller to the PC	Write code on the Arduino that outputs a basic text file. Display the text file on the computer. The displayed file will be the same data the Arduino output	Y
PC		
The GUI will change values on the based on the values from the microcontroller	Write Arduino code that sets heart rate to 60. The Laerdal software will change the heart rate to 60.	Y

Appendix B User Interface Supplemental Materials

Table B.1: Pin shorts for each key pressed

Key Pressed	Pins shorted
0	1,4
1	2, 3
2	1, 2
3	2, 5
4	3, 7
5	1, 7
6	5, 7
7	3, 6
8	1, 6
9	5, 6
#	4, 5
*	3, 4

Figure B.1: Top view of user interface

Figure B.2: Left side view of user interface

Figure B.3 Left side view of user interface

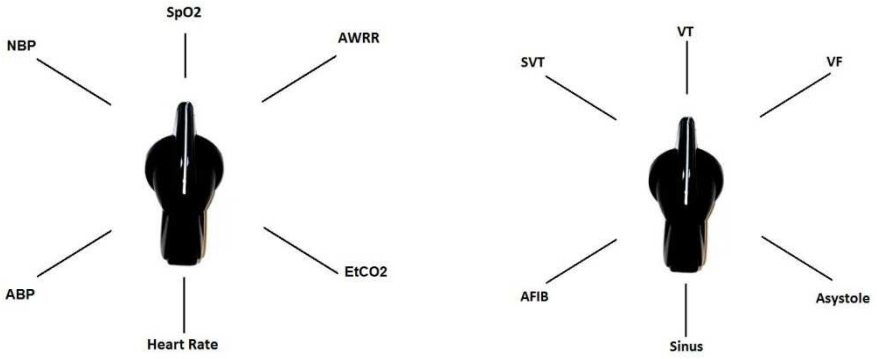

Figure B.4: Switch layouts

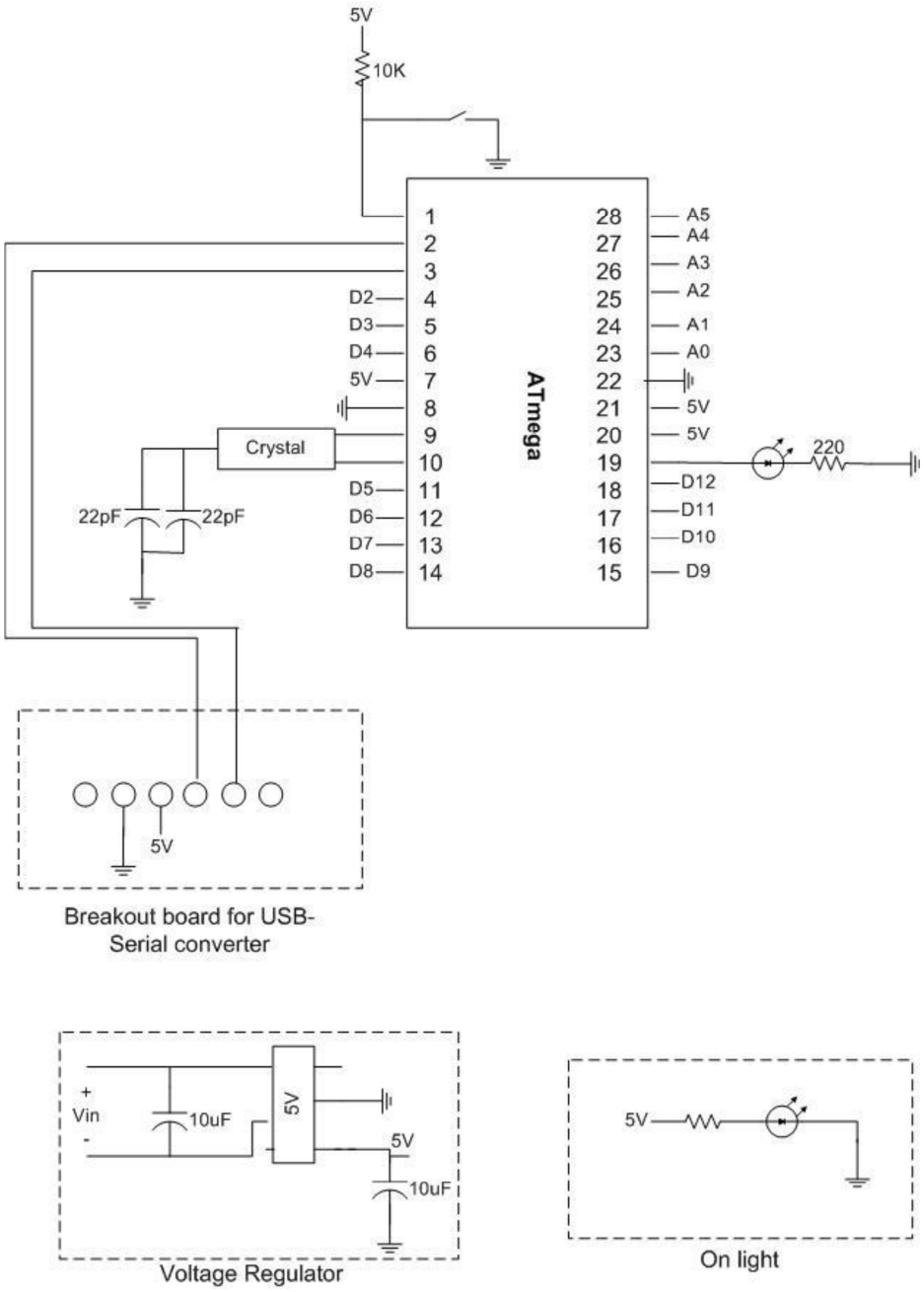

Figure B.5: Arduino schematic

Appendix C Verification data

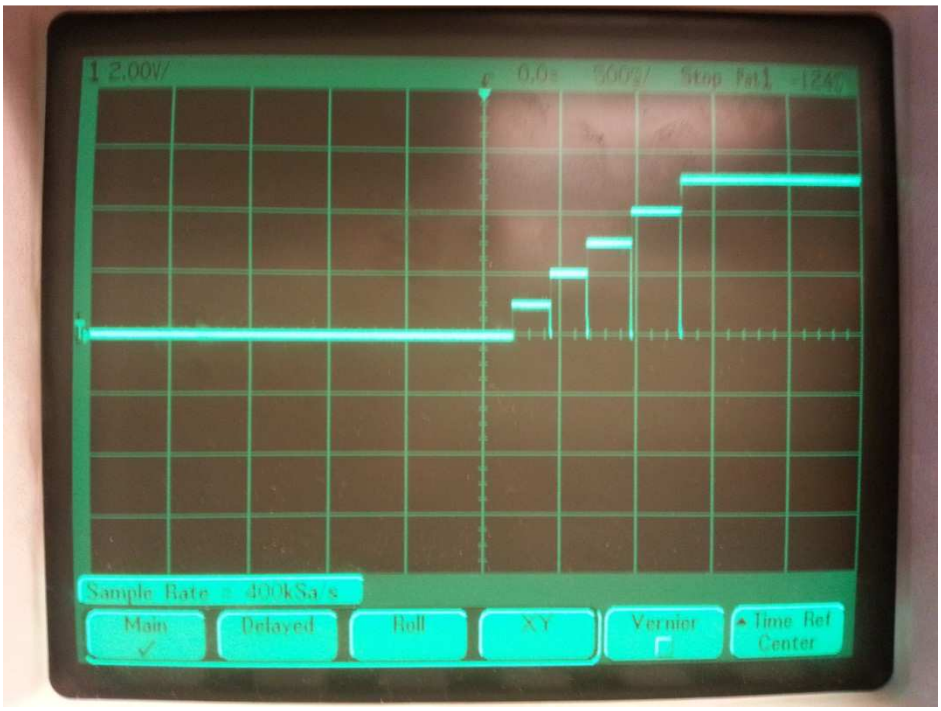

Figure C.1: Rotary switch verification

Table C.1: Keypad verification data

Key Pressed	Key Accuracy (out of 25 key presses)
0	15
1	24
2	24
3	24
4	23

5	23
6	23
7	21
8	21
9	20
#	12
*	10

Table C.2: Keypad verification data after averaging function

Key Pressed	Key Accuracy (out of 25 key presses)
0	25
1	25
2	25
3	25
4	25
5	25
6	25
7	25
8	25
9	25
#	25
*	24

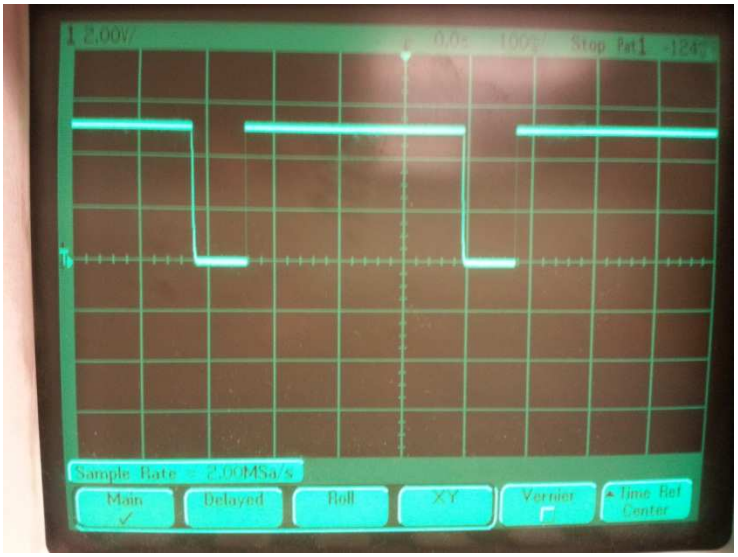

Figure C.2: Button verification

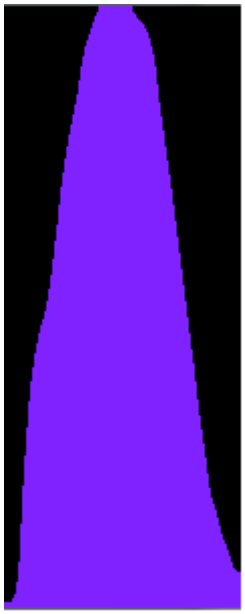

Figure C.3: Rotary encoder verification

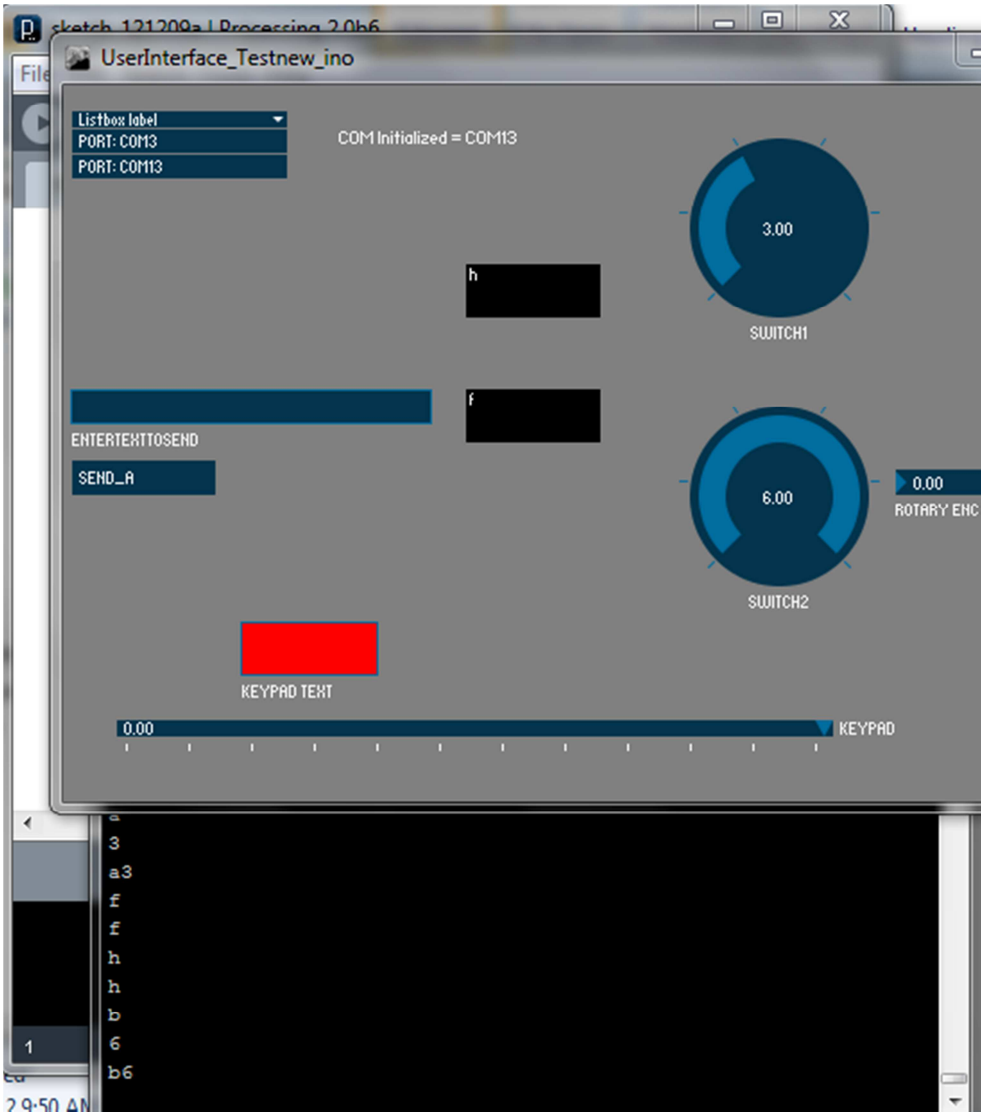

Figure C.4: USB interface verification

Figure C.5: Power supply verification

Figure C.6: Voltage regulator verification

Figure C.7: Microcontroller verification to the LCD

Appendix D PCB layout and schematics

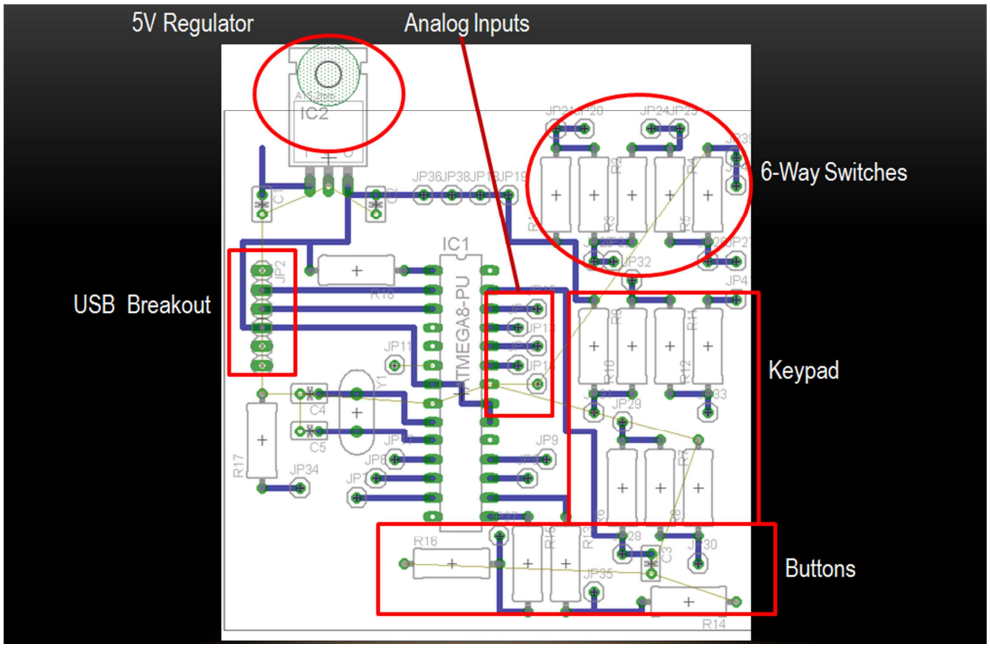

Figure D.1: PCB Layout

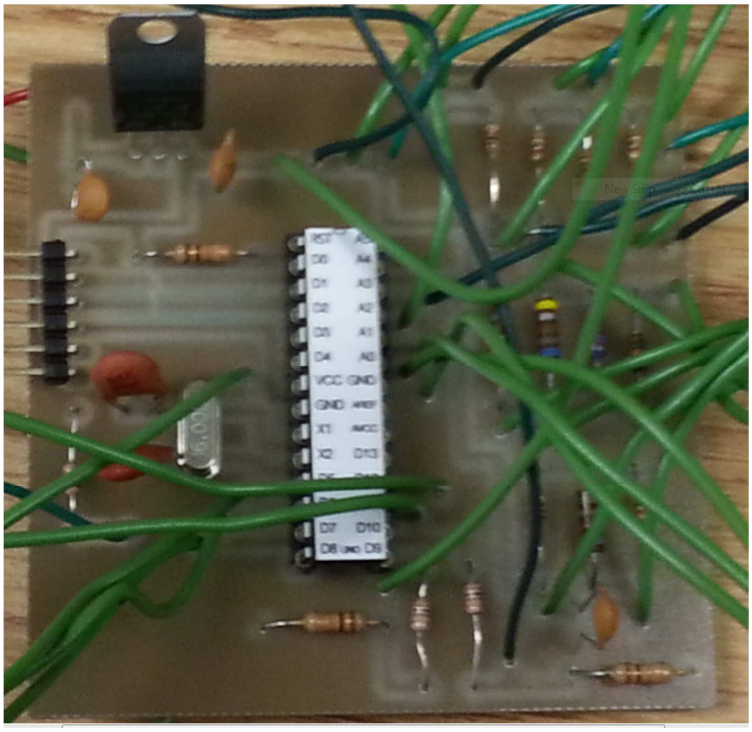

Figure D.2: Actual PCB with all components soldered to it

Appendix E Arduino Code

```
/*
The circuit:
 * LCD RS pin to digital pin 12
 * LCD Enable pin to digital pin 11
 * LCD D4 pin to digital pin 5
 * LCD D5 pin to digital pin 4
 * LCD D6 pin to digital pin 3
 * LCD D7 pin to digital pin 2
 * LCD R/W pin to ground
 * 10K resistor:
 * ends to +5V and ground
 * wiper to LCD VO pin (pin 3)
 */

// include the library code:
#include <LiquidCrystal.h>
#define ENC_A 14
#define ENC_B 15
#define ENC_PORT PINC


// initialize the library with the numbers of the interface pins

LiquidCrystal lcd(12, 11, 4, 5, 6, 7);
int keypressed = 0;
int keyboardPin = 4;    // Analog input pin that the keypad is attached to
int keyboardValue = 0;   // value read from the keyboard
int SixWayPin = 3;
int SixWayPos = 0;
int SixWayPin2 = 2;  // not sure about the pin number  DAVID  !!!
int SixWayPos2 = 0;
int buttonPin = 9;
int buttonState = 0;
// added for second button serial output
int buttonState2 = 0;
int buttonPin2 = 10;


void setup() {
  Serial.begin(115200);
  //Serial.begin(9600);
```

```
  Serial.println("Start");
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  // Print a message to the LCD.
//  lcd.print("KEY  ROT SW    B");
  lcd.print("HRtm Vital Val #");
  pinMode(ENC_A, INPUT);
  digitalWrite(ENC_A, HIGH);
  pinMode(ENC_B, INPUT);
  digitalWrite(ENC_B, HIGH);

  pinMode(buttonPin , INPUT);
  pinMode(buttonPin2, INPUT);

}

void loop() {
 //Button Pos
 // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);
  buttonState2 = digitalRead(buttonPin2);
//   lcd.setCursor(15, 1);
//   lcd.print("0");
  // check if the pushbutton is pressed.
  // if it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    // turn LED on:
//    lcd.setCursor(15, 1);
//    lcd.print(buttonState);
    // serial output for gui
    Serial.println("e");
  }
  else if (buttonState == LOW) {
  // lcd.setCursor(15, 1);
  // lcd.print(buttonState);
    // serial output for gui
    Serial.println("f");
  }
  // second button at input pin 10
  if (buttonState2 == HIGH ) {
   // lcd.setCursor(15, 1);
```

```
   // lcd.print(buttonState2);
    Serial.println("g");
  }
  else if (buttonState2 == LOW){
    Serial.println("h");
//    lcd.setCursor(15, 1);
//   lcd.print(buttonState2);
  }



 //6-way switch (B)
 SixWayPos2 = analogRead(SixWayPin2);
 if((SixWayPos2 <171)){
  lcd.setCursor(5, 1);
  lcd.print("    ");
  lcd.setCursor(5, 1);
  lcd.print("HR  ");
  Serial.println("b1");
 }
 else if (SixWayPos2 < 342 ) {
  lcd.setCursor(5, 1);
  lcd.print("    ");
  lcd.setCursor(5, 1);
  lcd.print("ABP ");
  Serial.println("b2");
 }
 else if (SixWayPos2 < 513) {
  lcd.setCursor(5, 1);
  lcd.print("    ");
  lcd.setCursor(5, 1);
  lcd.print("NBP");
  Serial.println("b3");
 }
 else if (SixWayPos2 < 684) {
  lcd.setCursor(5, 1);
  lcd.print("    ");
  lcd.setCursor(5, 1);
  lcd.print("SpO2");
  Serial.println("b4");
 }
```

```
else if (SixWayPos2 < 855) {
 lcd.setCursor(5, 1);
 lcd.print("    ");
 lcd.setCursor(5, 1);
 lcd.print("AWRR");
   Serial.println("b5");
}
else {
 lcd.setCursor(5, 1);
 lcd.print("    ");
 lcd.setCursor(5, 1);
 lcd.print("EC02");
 Serial.println("b6");
}

//6-way switch (a)
SixWayPos = analogRead(SixWayPin);
if((SixWayPos <171)){
 lcd.setCursor(0, 1);
 lcd.print("    ");
 lcd.setCursor(0, 1);
 lcd.print("Snus");
 Serial.println("a1");
}
else if (SixWayPos < 342 ) {
 lcd.setCursor(0, 1);
 lcd.print("    ");
 lcd.setCursor(0, 1);
 lcd.print("Afib");
 Serial.println("a2");
}
else if (SixWayPos < 513) {
 lcd.setCursor(0, 1);
 lcd.print("    ");
 lcd.setCursor(0, 1);
 lcd.print("SVT ");
 Serial.println("a3");
}
else if (SixWayPos < 684) {
 lcd.setCursor(0, 1);
 lcd.print("    ");
```

```
 lcd.setCursor(0, 1);
 lcd.print("VT  ");
 Serial.println("a4");
}
else if (SixWayPos < 855) {
 lcd.setCursor(0, 1);
 lcd.print("    ");
 lcd.setCursor(0, 1);
 lcd.print("VF  ");
   Serial.println("a5");
}
else {
 lcd.setCursor(0, 1);
 lcd.print("    ");
 lcd.setCursor(0, 1);
 lcd.print("Asys");
 Serial.println("a6");
}

 //Keyboard
 keyboardValue = analogRead(keyboardPin);//read the keyboard value (0 - 1023)

 if (keyboardValue > 25){
 readkeyboard(); //get the value of key being pressed "keypressed" i.e. 0-9
 }

 //Rot
 static uint8_t counter = 0; //this variable will be changed by encoder input
int8_t tmpdata;

 tmpdata = read_encoder();
 if( tmpdata ) {
   Serial.print("r");
   Serial.println(counter, DEC);
   counter += tmpdata;
    //LCD
 // (note: line 1 is the second row, since counting begins with 0):

 lcd.setCursor(11, 1);
 // print the number of seconds since reset:
 lcd.print("   ");
```

```
  lcd.setCursor(11, 1);
  lcd.print(counter);
  //Serial.println("r"+counter);
  }



}
int8_t read_encoder(){

  static int8_t enc_states[] = {0,-1,1,0,1,0,0,-1,-1,0,0,1,0,1,-1,0};
  static uint8_t old_AB = 0;
 /**/
  old_AB <<= 2; //remember previous state
  old_AB |= ( ENC_PORT & 0x03 );  //add current state
  return ( enc_states[( old_AB & 0x0f )]);
}
//read the keyboard routine
void readkeyboard(){
   int long temp_val = 0;

   for(int i = 0;i<1000;i++){
     temp_val += analogRead(keyboardPin);
   }

   keyboardValue = temp_val/1000; // read the value (0-1023)
   // Change to integer values but we could make these values strings
   // that make more sense
   if (keyboardValue <25){keypressed = 11;}//0
   if ((keyboardValue >25) && (keyboardValue < 67)){keypressed = 12;}//1
   if ((keyboardValue >67) && (keyboardValue < 108)){keypressed = 13;}//2
   if ((keyboardValue >108) && (keyboardValue < 162)){keypressed = 14;}//3
   if ((keyboardValue >162) && (keyboardValue < 253)){keypressed = 15;}//4
   if ((keyboardValue >253) && (keyboardValue < 361)){keypressed = 16;}//5
   if ((keyboardValue >361) && (keyboardValue < 479)){keypressed = 17;}//6
   if ((keyboardValue >479) && (keyboardValue < 619)){keypressed = 18;}//7
   if ((keyboardValue >619) && (keyboardValue < 700)){keypressed = 19;}//8
   if ((keyboardValue >700) && (keyboardValue < 800)){keypressed = 20;}//9
   if ((keyboardValue >800) && (keyboardValue < 865)){keypressed = 22;}//*
   if ((keyboardValue >865) && (keyboardValue < 910)){keypressed = 11;}//0
   if (keyboardValue >910){keypressed = 21;}//#
```

```
   //NOTE: the values used above are all halfway between the value obtained
   //with each keypress in previous test sketch
   // Serial.println(keyboardValue);
    while (keyboardValue > 25) {
      delay (100);
      keyboardValue = analogRead(keyboardPin); // read the value (0-1023)
    // Serial.println(keypressed);
    }//wait until key no longer being pressed before continuing

    Serial.println(keypressed);
   // lcd.clear();
       //LCD
   // (note: line 1 is the second row, since counting begins with 0):
   lcd.setCursor(15, 1);
   // print the number of seconds since reset:
   if (keypressed == 22){
     lcd.print("B");
   }
   else if (keypressed == 21){
     lcd.print("E");
   }
    else {
     lcd.print(keypressed- 11);
   }
    delay(250);      // wait 1000 milliseconds before the next loop

 }
 //end of read the keyboard routine


/* returns change in encoder state (-1,0,1) */
```

Appendix F GUI layout and Processing code

Figure E.1: GUI layout

Figure E.2: Processing code

```
size(700,400);
frameRate(30);

controlP5 = new ControlP5(this); // initialize the GUI controls

  println(Serial.list()); // print the comm ports to the debug window for debugging purposes

// make a listbox and populate it with the available comm ports
commListbox = controlP5.addListBox("myList",5,25,120,120); //addListBox(name,x,y,width,height)
commListbox.captionLabel().toUpperCase(false);
commListbox.captionLabel().set("Ports");
for(int i=0;i<Serial.list().length;i++) {
  commListbox.addItem("port: "+Serial.list()[i],i); // addItem(name,value)
}

// text label for which comm port selected
txtlblWhichcom = controlP5.addTextlabel("txtlblWhichcom","No Port Selected", 122,20); // textlabel(name,text,x,y)
txtlblWhichcom.setFont(label);
//More Text labels
Switch1 = controlP5.addTextlabel("Switch1","Heart Rhythm", 305,10); // textlabel(name,text,x,y)
Switch1.setFont(label);
Switch2 = controlP5.addTextlabel("Switch2","Vitals", 530,10); // textlabel(name,text,x,y)
Switch2.setFont(label);
keyPadLbl = controlP5.addTextlabel("keyPadLbl","Keypad", 600,350); // textlabel(name,text,x,y)
keyPadLbl.setFont(label);
rotEnclbl = controlP5.addTextlabel("rotEnclbl","Rotary Encoder", 500,255); // textlabel(name,text,x,y)
rotEnclbl.setFont(label);
// when triggered, write the text over the serial line
controlP5.addTextfield("EnterTextToSend",5,170,200,20); // addTextfield(name,x,y,width,height)

// when new info comes into the serial port, write it to this text area
commTextarea = controlP5.addTextarea( // addTextarea(name,text,x,y,width,height)
"labelltextarea",
"No data yet...",
310,215,80,40);
commTextarea.setColorBackground(0xffff0000);
commTextarea.setFont(commTextFont);

commTextarea2 = controlP5.addTextarea( // addTextarea(name,text,x,y,width,height)
"labelltextarea2",
"No data yet...",
310,270,80,40);
commTextarea2.setColorBackground(0xffff0000);
commTextarea2.setFont(commTextFont);
```

```
89
90      Vitals = controlP5.addTextarea( // addTextarea(name,text,x,y,width,height)
91      "label1textarea3",
92      "No data yet...",
93      310,135,80,40);
94      Vitals.setColorBackground(0xffff0000);
95      Vitals.setFont(commTextFont);
96
97      HRtm = controlP5.addTextarea( // addTextarea(name,text,x,y,width,height)
98      "label1textarea4",
99      "No data yet...",
100     510,135,80,40);
101     HRtm.setColorBackground(0xffff0000);
102     HRtm.setFont(commTextFont);
103
104     Rotenc = controlP5.addTextarea( // addTextarea(name,text,x,y,width,height)
105     "label1textarea5",
106     "No data yet...",
107     510,215,80,40);
108     Rotenc.setColorBackground(0xffff0000);
109     Rotenc.setFont(commTextFont);
110
111     //commTextarea2.setColorBackground(0); Black
112     // a button
113     controlP5.addButton("Send_A",1,5,210,80,19); // buton(name,value,x,y,width,height)
114
115     // 6- way switch
116     //create 6-switches
117     myKnobA = controlP5.addKnob("")
118                    .setRange(1,6)
119                    .setValue(1)
120                    .setPosition(300,30)
121                    .setNumberOfTickMarks(5)
122                    .setTickMarkLength(4)
123                    .snapToTickMarks(true)
124                    .setRadius(50)
125                    .setDragDirection(Knob.HORIZONTAL)
126
127                    ;
128     myKnobB = controlP5.addKnob(" ")
129                    .setRange(1,6)
130                    .setValue(1)
131                    .setPosition(500,30)
132                    .setNumberOfTickMarks(5)
```

```
                .setTickMarkLength(4)
                .snapToTickMarks(true)
                .setRadius(50)
                .setDragDirection(Knob.HORIZONTAL)
                ;
  myKeypad =  controlP5.addSlider("   ")
              .setPosition(400,355)
              .setWidth(200)
              .setRange(11,0)
              .setValue(0)
              .setNumberOfTickMarks(12)
              .setSliderMode(Slider.FLEXIBLE)
              ;

  // Keypad text box
  keypadText = controlP5.addTextfield( // addTextarea(name,text,x,y,width,height)
  "keypad text",
  280,335,100,50);
  keypadText.setColorBackground(0xffff0000);
  keypadText.setFont(p);
  //controlP5.addTextfield("EnterTextToSend",5,170,200,20);


}

// infinite loop
void draw() {
  background(128);
}

// print the name of the control being triggered (for debugging) and see if it was a Listbox event
public void controlEvent(ControlEvent theEvent) {
  // ListBox is if type ControlGroup,
  if (theEvent.isGroup()) {
    // an event from a group
    if (theEvent.name()=="myList") {
      InitSerial(theEvent.group().value()); // initialize the serial port selected
      //println("got myList"+"   value = "+theEvent.group().value()); // for debugging
    }
  }
  else {
    //println(theEvent.controller().name()); // for debugging
  }
}
```

```
177
178   // run this when there is an enter in the textfield
179 public void EnterTextToSend(String theText) {
180     myPorts[0].write(theText+"\n"); // write the text in the field
181     println("sent "+theText); // print it to the debug screen as well
182   }
183
184   // run this when buttonA is triggered, send an a
185 public void Send_A(int theValue) {
186     myPorts[0].write("a\n"); // write an a
187     println("sent a"); // print it to the debug screen as well
188   }
189
190   // initialize the serial port selected in the listBox
191 void InitSerial(float portValue) {
192     println("initializing serial " + int(portValue) + " in serial.list()"); // for debugging
193
194     String portPos = Serial.list()[int(portValue)]; // grab the name of the serial port
195     txtlblWhichcom.setValue("COM Initialized = " + portPos);
196     myPorts[0] = new Serial(this, portPos, 115200); // initialize the port
197
198     // read bytes into a buffer until you get a linefeed (ASCII 10):
199     myPorts[0].bufferUntil('\n');
200     println("done init serial");
201   }
202
203   // serial event, check which port generated the event
204   // just in case there are more than 1 ports open
205 void serialEvent(Serial thisPort) {
206     // variable to hold the number of the port:
207     int portNumber = -1;
208
209     // iterate over the list of ports opened, and match the
210     // one that generated this event:
211     for (int p = 0; p < myPorts.length; p++) {
212       if (thisPort == myPorts[p]) {
213         portNumber = p;
214       }
215     }
216
217     // read the serial buffer until a newline appears
218     String myString = thisPort.readStringUntil('\n');
219     myString = trim(myString); // ditch the newline
220     //println("got: " + myString); // print to debug window
```

```
221  if (myString.equals("e") || myString.equals("f")){
222    if(myString.equals("e")){
223      commTextarea.setText("Next State");
224      commTextarea.setColorBackground(0);
225    }
226    else{
227      commTextarea.setText("1 Unpressed");
228      commTextarea.setColorBackground(0);
229    }
230  }
231
232  if (myString.equals("g") || myString.equals("h")){
233   if(myString.equals("g")){
234      commTextarea2.setText("Waiting Rhythm");
235     commTextarea2.setColorBackground(0);
236    }
237    else{
238      commTextarea2.setText("2 Unpressed");
239     commTextarea2.setColorBackground(0);
240    }
241
242  }
243  char test1 = myString.charAt(0);
244  String test = String.valueOf(test1);
245  println(test);
246  if (test.equals("a") || test.equals("b")){
247  // for changing switch position (switch A)
248  // If we can use string manipulation we can make this shorter
249  int len = myString.length();
250  int i;
251  String charstring = myString.substring(1);
252  Integer temp2;
253  charstring = trim(charstring);
254  println(charstring);
255  temp2 = Integer.valueOf(charstring);
256
257  if (test.equals("a")){
258    println(myString);
259    temp2 = abs(temp2 -6)+1;
260    myKnobA.setValue(temp2);
261    switch(temp2) {
262     case 6 :       Vitals.setText("Snus"); // put it in the text area
263                    Vitals.setColorBackground(0);
264                    break;
```