Flow-O-Matic; Portable Arduino Spirometer

Agoston Walter   Cody Morris and Alex Shang

The spirometer device built by our group can conduct a range of pulmonary function tests, measuring inspiratory capacity, expiratory capacity, tidal volume, and airflow. One can calibrate the device with a syringe of a known volume, and can select between two pneumotachometers, whose stored calibration constants are saved onto the device even in the case of a loss of power. The device does all the calculations using (1) in an Arduino processor. The spirometer finds the measurements with a pneumotachometer and converts them into a full 0-5 V resolution accessible by an Arduino, using two amplifiers and a low pass filter. The performance of the device is reliable. The measured volumes deviate from the true volume by a rate of 3-6%, the resolution of the airflow is 30 L/min by conservative estimates, and 8 L/min by liberal ones, and the device is linear for airflows typical for human subjects.

I. INTRODUCTION

Spirometers come in many different types, such as plethysmographs, pneumotachometers, and windmill type spirometers. For this lab report, our team used a pneumotachometer, which measures airflow by detecting pressure differences across a fine mesh, using a pressure sensor. Pneumotachometers use an equation to relate voltage to airflow, shown in (1), where V is voltage, Δt is difference in time between measurements, and K is the calibration constant.

To find the calibration constant of the spirometer, K, a known volume of air is pressed through the pneumotachometer. K is equal to the known volume divided by the sum of the resulting voltages multiplied by the difference of time between measurements, an application of the trapezoidal rule. With the calibration constants set, volumes and current airflow can be estimated by using (1) with or without the Σ and Δt terms, respectively.

$$Known\ Volume = K \sum (V \Delta t) \tag{1}$$

Clinically, spirometers are used for diagnosing respiratory conditions, such as asthma, bronchitis, and COPD (Chronic Obstructive Pulmonary Disease). To understand if a patient's lung function is normal or not, data points are collected by the physician which include Inspiratory Capacity, Expiratory Capacity, and Tidal Volume, shown in Fig. 1 as IC, not pictured, and VT, respectively. The deviation in the patient's values from the normalized values for the patient's demographic features, including age, height, weight, and origin, inform the physician if there is still something wrong with the patient's lung function.

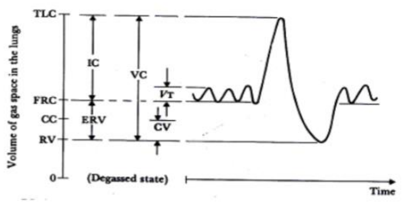


Fig. 1: Spirometry Measurements

II. DESIGN SPECIFICATIONS

Under the specifications, the Arduino is the power source for the signal processing circuit and the device. The Arduino can output voltages from 0 to 5 V, making the pneumotachometer's desired range be ±2.5V. The resolution of the Arduino is 10-bit, making the resolution of system 4.88 mV. Peak expiratory flow for humans ranges up to 800 L/min, so the target resolution for airflow is 1.56 L/min, found by dividing the 800 L/min by 512.

In summary, the flowrates over which the device is expected to function is ±800 L/min, with a resolution of 1.56 L/min. The spirometer is designed to be used with a power supply that provides a supply voltage to the Arduino. The device is able to estimate volume, with a bias of 0.086 L for inspiration and -0.177 L for expiration, with a standard deviation of 0.199 and 0.105, respectively. For airflow, the device operates under linearity, as required.

III. HARDWARE DESIGN

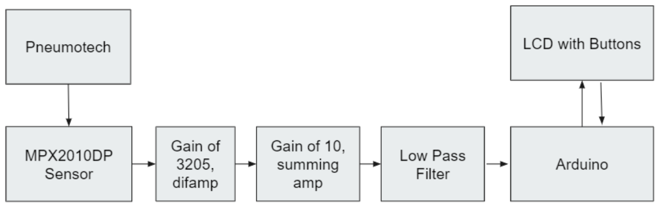


Fig. 2: Spirometer Block Diagram

The MPX2010DP sensor outputs 2.5 mV/kPa. To cover the full resolution of 2.5 V requires a high gain. The max gain possible from the instrumentation amplifer was 3205, with a gain resistor of 25Ω. The instrumentation amplifier was used because it has several advantages over a typical three op-amp differential amplifier. Like a three op-amp differential amplifier, the instrumentation amplifier has a very high input impedance, and balanced inputs. The advantages are the simplicity of the design and improved performance over a three op-amp differential amplifier, at the drawback of an increased price. Another drawback of the instrumentation amplifier is that it rails at voltages of 0.5 V and 4.5 V, necessitating for the incorporation of a second amplifier into the circuit for full resolution from 0 to 5 V. In addition, a voltage divider was placed, with a resistor of value 2.7 kΩ and a potentiometer (indicated by the resistor with an arrow through it) in series, with the intention of delivering an offset voltage to the Voltage reference of the instrumentation amplifier. With the voltage reference set to 2.5 V, the outputted values of the instrumentation amplifier would be centered at 2.5 V with a maximum of 4.5 V and a minimum of 0.5 V, as opposed to the positive and negative input voltage values coming into the amplifier.

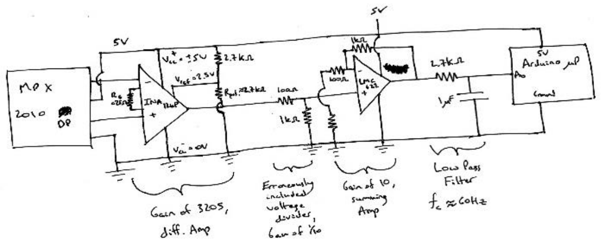


Figure 3: Spirometer Circuit Diagram

Erroneously, a voltage divider that reduced the gain by 1/10 was soldered into the circuit, which remained unnoticed until the demo, when the keen eyes of the instructor found the error in the diagram. The voltage divider was placed into the circuit because an op-amp sequence found while researching circuit diagrams had a voltage divider embedded into the circuit, and our group was searching for solutions, so we incorporated it into our circuit and never thought to remove it. The voltage divider is malicious because it reduces the gain by 10, neutralizing the summing amplifier's added gain of 10 to the circuit. The presence of the voltage divider explains why our group was having trouble understanding why the summing amplifier was not having a noticeable effect on the gain, when we were comparing the output voltages of the instrumentation amplifier and the summing amplifier.

After the voltage divider, a summing amplifier was used with the intention of expanding the resolution to the full possible range. The LMC 662 amplifier was used because it has no railing, and thereby had the potential to increase the resolution from 4 Volts to the full range of 5 Volts. A gain of 10 was thought to be sufficient, but this did not work in testing due to the mistakenly incorporated voltage divider, and this error was not discovered until the demonstration. To adjust the gain, a potentiometer was used in series with a 100 Ω and 1k Ω resistor, connecting the inverting input to the output, in an inverting amplifier configuration.

Finally, the signal passes through a passive Low Pass Filter (LPF) with a cutoff frequency of 60 Hz. Ideally, the Low Pass Filter removes any higher frequency signals because the frequency of human breathing ranges from 0.1 Hz to 5 Hz, and higher frequency signals are not the target signal.

IV. SOFTWARE DESIGN

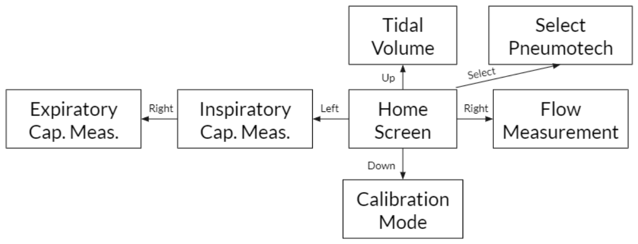


Figure 4: Software Design Diagram

Methods (Appendix Sections)

Global Variables (Global Variables Used and Imports):

Several global variables are used. The difference in time, Δt, is initialized as dt with a value of 1 ms, the sampling rate used in our code. The float sum is used as a global summation variable, and reset after use. The global variable time is used for the LCD screen. The calibration constants are initialized as floats, and their addresses for saving are initialized as well. The voltage offset is initialized to 2.5 V and the hysteresis constant is initialized as well, both will be updated in the Setup. The Boolean selection value for pneumotachometers 1 and 2 is also initialized, and set to true, indicating that pneumotach 1 has been selected.

Home Screen Code Description (Section A):

Upon initialization, the calibration constants for both pneumotachometers is read in from the values stored in memory. The voltage offset is calculated from the one second average of voltage values read in from the pin A0, and the hysteresis constant is set to be 0.05 times the voltage offset. It is important to refresh the voltage offset each time the spirometer is turned on because the voltage offset can fluctuate, so it cannot be hard-coded and is important to refresh with each session. After the voltage offset and hysteresis constant are calculated, the home screen displays the voltage offset and goes blank. This is the home screen, and from here, the spirometer's methods can be accessed.

Select Pneumotach Code Description (Section B):

If one presses the select button, the select pneumotach method is run. In this method, the Boolean for the selected pneumotach is flipped, making the other pneumotach's values the ones used for volume and airflow estimation. The display also prints out the current values being used, so that the user knows which pneumotach is selected and the calibration constants. Afterwards, the spirometer returns to the home screen.

Calibration Mode Code Description (Section C):

If one presses the down button, the spirometer's calibration method is run. In this method, the user is prompted to breathe in by commands displayed on the LCD screen. Afterwards, three seconds of measurements are taken with a sampling rate of 1 ms. From these measurements, the calibration constant for inhaling is calculated using (1), where the known value is 3 L. After the measurement is over, the display tells the user the calculated inspiratory calibration constant, and then instructs the user to breathe out. Three seconds of measurement are taken again with a sampling rate of 1 ms, and again, the calibration constant for exhaling is measured using (1), with a known value of 3 L, the volume of the calibration syringe. The LCD screen displays the calculated expiratory capacity constant, and then returns to the home screen.

Tidal Volume Code Description (Section D):

The first if statement sets the calibration constants to the correct ones, using the state of the global Boolean used for pneumotach selection. After displaying that the mode selected by the user is the tidal volume mode, the spirometer instructs the user to breathe in, and takes 3 seconds of measurement, and then instructs the user to breathe out, taking three seconds of measurement. The volumes are estimated using (1) and they are then added together to estimate tidal volume, which is then displayed before returning to the home screen.

Flow Measurement Code Description (Section E):

Flow measurement mode is selected by pressing the right button. Upon selection, the screen displays “Flow Mode Selected”. Then, 250 measurements are taken in intervals of 10 ms each, and if the voltage difference is not above the hysteresis constant, the flow rate is not displayed. The flow rate is calculated using (1), neglecting the dt term to find an instantaneous airflow rate using only the volume measured and the selected calibration constant, and displayed. Afterwards, the method exits to the home screen.

Inspiratory Mode Code Part 1 Description (Section F):

To enter inspiratory or expiratory capacity calculation mode, the user must press the left button. The initial part of the inspiratory capacity mode code was built to allow the user to enter either the inspiratory or expiratory mode. If the user wishes to enter the inspiratory mode, they should do nothing. If the user wishes to enter the expiratory mode, they are instructed to press the right button and are given a one second interval to do so. If they press the right button, the code jumps to the expiratory mode code using the goto method.

Inspiratory Mode Code Part 2 Description (Section G):

If the user chooses not to enter the expiratory capacity calculation mode, they proceed to the second part of the inspiratory capacity calculation mode. In the second part, the user is instructed to breathe in, and three seconds of measurement are then taken to calculate the estimated inspiratory capacity using the correct pneumotachometer calibration constant and (1). Afterwards, the spirometer returns to the home screen.

Expiratory Mode Code Description (Section H):

If the user chooses to press the right button, they are taken to the expiratory capacity mode. The user is instructed to breathe out, and three seconds of measurement are then taken to calculate the estimated expiratory capacity using the correct pneumotachometer calibration constant and (1). Afterwards, the spirometer returns to the home screen.

V. INSTRUCTIONS FOR USE

Upon initialization, the spirometer makes takes the average of one second of value readings and stores that as the voltage offset, and stores 1/20 of the voltage offset as the hysteresis constant. After displaying the voltage offset, the LCD screen goes blank, this indicates it has entered the neutral state. From the neutral state, all of the methods can be accessed using the LCD screen’s buttons. Code can be found in Appendix Section A.

To select which pneumotachometer’s calibration constants to use for either measurements or calibration, the user should press the select button, upon which the selected pneumotachometer’s expiration or inspiration constants will be displayed along with its number. Code can be found in Appendix Section B.

To calibrate the spirometer, the down button must be pressed on the LCD screen. The calibration is set for a volume of 3 Liters in the software, this can be changed depending on the syringe pump being used by updating the value in the calibration method. Upon the instruction of, “Please Breath In Now”, the user has three seconds to fill the syringe pump (fitted to the pneumotachometer), with air. After the conclusion of the three seconds, the LCD screen prints “Measurement Over”, and displays the calculated calibration constant for inspiration. This is repeated for the calculation of the expiratory capacity constant with differences made in the displayed instructions on the LCD screen. Code can be found in Appendix Section C.

To calculate the Tidal Volume, the up button on the spirometer’s LCD screen must be pressed. Upon instruction of “Quietly Breathe in” and “Quietly Breathe out”, the user has three seconds to comply with the instructions. After the measurements are taken, the estimated tidal volume is given after the following statement, “Total Tidal Volume: ”. Code can be found in Appendix Section D.

To calculate the Flow Measurement, the right button on the spirometer’s LCD screen must be pressed. If the voltage difference detected is not greater than the hysteresis constant calculated upon initialization, the flow measured will be displayed as 0 L/min. 250 measurements will be displayed as “Flow is: ”, each in intervals of 10 ms, and then the method will exit to the neutral state, indicated by a blank LCD screen. Code can be found in Appendix Section E.

To calculate either the Expiratory Capacity or Inspiratory Capacity, first the left button on the LCd screen must be pressed. The screen displays, “Insp. Cap. Mode Sel.”, and then gives the user an option to switch to the Expiratory Mode with the instruction “For Exp. Cap. Press Right Btn”, and giving a time delay of around one second for the user to respond. Code can be found in Appendix Section F.

If the user does nothing, the spirometer enters Inspiratory Capacity Mode, asking the user to “Strongly Breathe In Now”, and then displaying the total inspiratory volume with the statement “Tot. Insp. Vol. is: ”, after three seconds of measurement. Code can be found in Appendix Section G.

If the user presses the right button, the spirometer enters Expiratory Capacity Mode, asking the user to “Strongly Breathe

Out Now", and then displaying the total expiratory volume with the statement "Tot. Exp. Vol. is: ", after three seconds of measurement. Code can be found in Appendix Section H.

After any of the methods are finished executing, the spirometer returns to the neutral state, with the LCD screen displaying nothing. If the user wishes to power down the spirometer, the power cord can be pulled out from the Arduino or the power supply source. In this case, the pneumotachometer calibration constants are saved and upon the next initialization, ready for use. All of the code used is available in the Appendix.

VI. Evaluation of performance

This section should include evaluation of the repeatability of measurements made with this device as well as an assessment of the accuracy, precision and bias of the measurements. You may wish to include figures to display this data.

To evaluate the performance of the spirometer, the estimated volumes of the device for the calibration syringe were compared against the true values of the syringe.

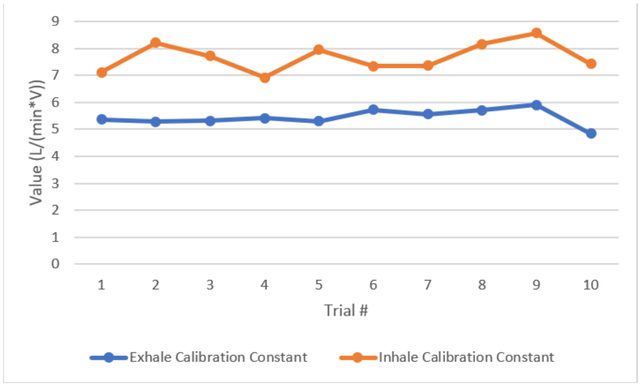


Fig. 4: Calculation of Calibration Constant Average

To gain accurate exhale and inhale calibration constants, ten trials were done measuring the calibration constants using the calibration syringe, with the averages taken for both inspiration and expiration. The average expiration calibration constant was found to be 5.44 L/(V*s), and the average inspiration calibration constant was found to be 7.673 L/(V*s). These were then used to estimate the calibration syringe volume, and the trials with the Estimated Volume are shown in Fig. 5.

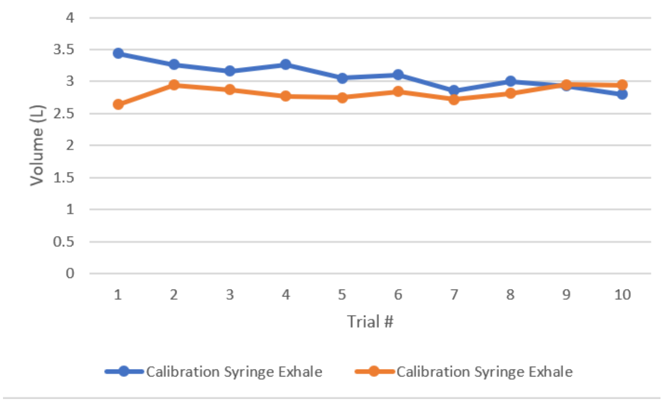


Fig. 5: Estimation of Calibration Syringe Volume

The average estimated volumes for inspiration and expiration are in the table below, along with the bias of the measurements against the true syringe volume of 3L. The bias is around 0.1-0.2, giving the estimated volume an error percentage of between 3 and 6 %. The standard deviation for the measurements also fell between 0.1 – 0.2 L. Considering the number of trials being 10, the result is satisfactory, with the error being below 10%. There are several areas for potential improvement, including resolution, shown in Fig. 7, that will be discussed in Section VIII.

	Syringe Volume Estimate Inhale	Syringe Volume Estimate Exhale
Average Estimated Volume	3.086	2.823
Bias	0.086	-0.177
Standard Deviation	0.199	0.105
Constants	5.44	7.67

Fig. 6: Accuracy Measurements for 10 samples

The device performed linearly for airflows up to 800 L/min as shown in Fig. 7, making it possible for use in human studies. The resolution was shown to be around 1.2 V, far from the ideal full resolution of 5 V, and this is a definite area for improvement. In addition, the resolution for airflow was found to be around 30 L/min, based on Fig. 7.

However, this is a conservative estimate, given that our dynamic range is 1.2 V, about one-fourth of the target dynamic range. With this information, our group estimates our true resolution for airflow to be around four to five times worse than our target airflow resolution, with the assumption that under the full target resolution, full airflow resolution of 1.56 L/min would be achieved. With this knowledge, our group estimates the true airflow resolution of the device to be between 6 to 8 L/min.

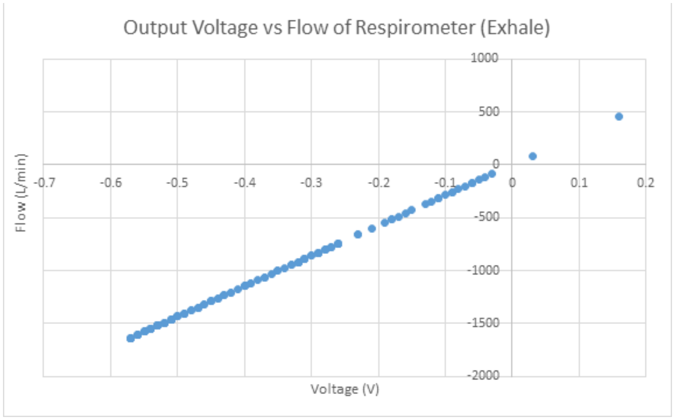


Fig. 7: Linearity of Device for Resolution of ±0.6V

VII. Lessions Learned

Challenges we encountered during the project included the limited number of buttons available on the LCD screen for picking methods, a mistakenly added voltage divider, and editing mistakes on the soldered PCB.

We had six methods for our spirometer but only five buttons available on the LCD screen, making it necessary for some of our methods to be accessible by a combination of buttons. Our group decided on making the Inhale/Exhale functions accessible by a combination of the Left and Left/Right buttons, respectively. To make this more accessible and user friendly, we included prompts explaining this in the operation of the device as well as the placement of text on the holder of the device, as shown in Fig. 8.

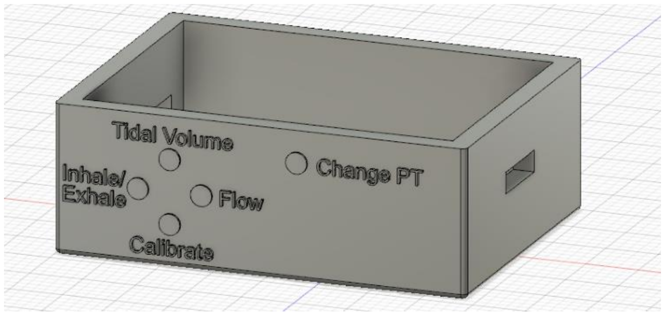


Fig. 8: 3D printed holder for device

The biggest challenge our group faced was the addition of a voltage divider between the instrumentation and summing amplifiers on the circuit, as seen in the circuit diagram in Fig. 3. The voltage divider had a gain of 1/10 and neutralized the gain of 10 the summing amplifier provided, leaving our group scratching our heads when the output continued to rail at 0.5 V and 4.5 V, and we suffered from a poor gain, as seen in our poor resolution of 1.2 V in Fig. 7. This was only figured out when the instructor pointed out the mistakenly included block in our circuit diagram.

While debugging our circuit, we believed it was not working due to faulty header pins, so some of the connections were cut in an effort to debug. It turned out the error was due to a short circuit of hanging wires on the solder underneath, which was easily fixed with a wire cutter. However, the pin headers needed to be removed from the PCB because the LCD screen required the pins whose headers were snipped. Removing the header was a difficult process because there were multiple contacts that were soldered to the board. It was also difficult to then place a new header into the place of the old header, because upon removal, the solder plugged the holes, making it necessary to melt each of the 8 holes before plugging the header back in.

References

Include relevant references using formatting similar to that below.

[1] Duke BME354 Lab 8 Protocol: Arduino Spirometry, Spring, 2019

[2] Duke BME354 Final Project Protocol: Design and Build a Spirometer, Spring 2019

[3] Freescale Semiconductor, "10 kPa On-Chip Temperature Compensated and Calibrated Silicon Pressure Sensors," MPX2010 Series datasheet, Revised October 2008.

[4] *Clement Clarke International (2004).* "Predictive Normal Values (Nomogram, EU scale)". *Clement Clarke International*. Retrieved 2006-06-06. - Downloadable PDF charts for adults and children using EU scale

[5] Texas Instruments, "INAx126 MicroPower Instrumentation Amplier Single and Dual Versions," SBOS062B datasheet, Sep. 2000 [Revised Dec. 2015].

[6] Texas Instruments, "LMC662 CMOS Dual Operational Amplifier," SNOSC51C datasheet, April 1998 [Revised March 2013].

CODE APPENDIX

GLOBAL VARIABLES USED AND IMPORTS

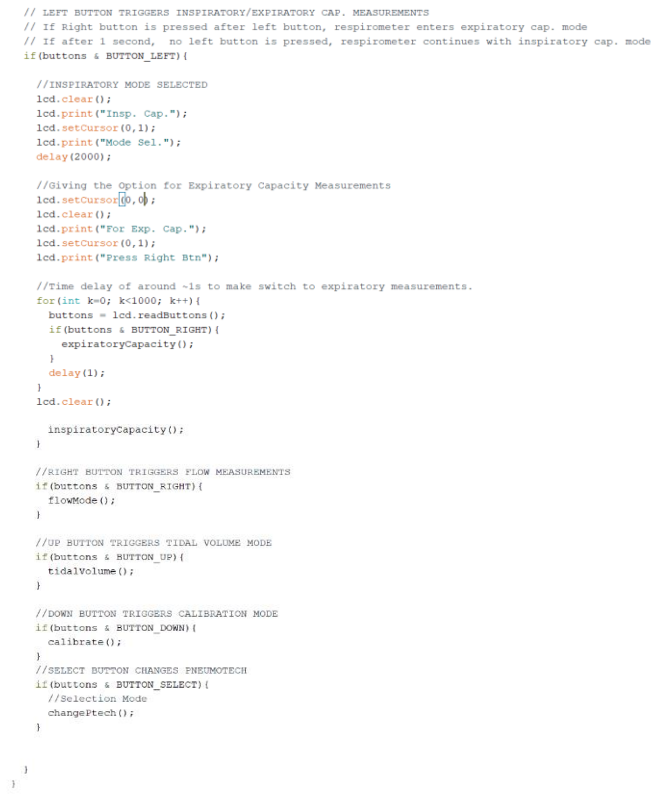
```
//Code Written by Agoston Walter and Alex Shang, BME354 April 2019

//***********************GLOBAL VARIABLES USED*************
//initialization of time difference to 1 ms and sum to 0
float dt = 0.001;
float sum = 0;
unsigned long time;
//initialization calibration constants and addresses used for EEPROMPT stores
float K_inhale;
float K_exhale;
//default calibration co nstants
float K_inhale1=100;
float K_exhale1=100;
float K_inhale2=100;
float K_exhale2=100;
int inhale1_address=0;
int exhale1_address=10;
int inhale2_address=30;
int exhale2_address=40;

//initialization of the voltage offset to 2.5 V
float V_offset=2.5;
//initialization of global hsyteresis constant
float hysteresis_const;
//intialization of boolean selection value for pneumotach
bool k1_selected = true;
//*******************************************************************
```

SECTION A

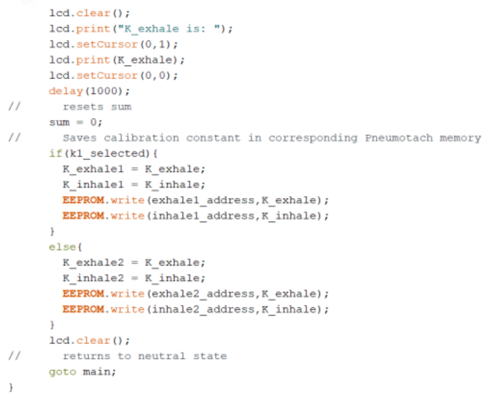
```
void setup() {
  //  initialization
  //  Extract pneumotachometer values from memory
  K_inhale1=EEPROM.read(inhale1_address);
  K_exhale1=EEPROM.read(exhale1_address);
  K_inhale2=EEPROM.read(inhale2_address);
  K_exhale2=EEPROM.read(exhale2_address);

  //  Sets up pins and Serial
  Serial.begin(9600);
  pinMode(A0, INPUT);
  int time = millis();

  //  Sets up LCD screen
  lcd.setCursor(0,0);
  lcd.begin(16,2);

  //  variable sensorValue is voltage of pin A0
  int sensorValue = analogRead(A0);
  //  Reads voltages from A0 for 1 second
    for(int i=0; i<1000; i++){
      float voltage= sensorValue * (5.0 / 1023.0);
      sum=sum+voltage;
      delay(1);
     }
  //  Calculates voltage offset and hysteresis constant based on the average of the read values
  V_offset = sum/1000;
  hysteresis_const = V_offset*0.05;
  //  Print statements
  lcd.print("V Offset: ");
  lcd.print(V_offset);
  lcd.print(" V");
  delay(2000);
  lcd.clear();
  //  resets sum
  sum=0;
}
```

SECTION B

```
void changePtech(){

//      flips boolean, making currently selected pneumotach the other one
      k1_selected = !k1_selected;
//      prints out currently selected pneumotach
      if(k1_selected){
        lcd.clear();
        lcd.print("Pneumotech 1");
        lcd.setCursor(0,1);
        lcd.print("selected");
        delay(1000);
        lcd.setCursor(0,0);
      }
      else{
        lcd.clear();
        lcd.print("Pneumotech 2");
        lcd.setCursor(0,1);
        lcd.print("selected");
        delay(1000);
        lcd.setCursor(0,0);
      }
//      sets currently used calibration constants to selected pneumotach's
      if(k1_selected){
        K_exhale = K_exhale1;
        K_inhale = K_inhale1;
      }
      else{
```

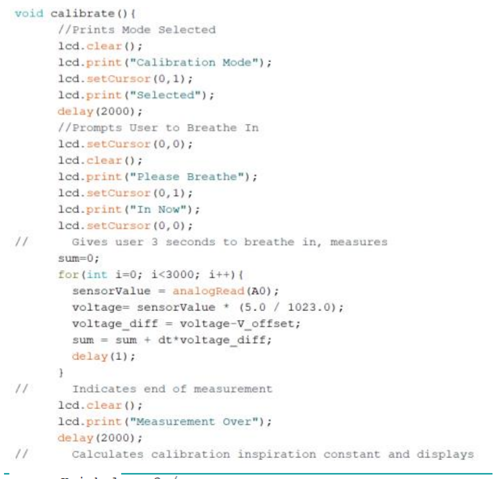


```
// LEFT BUTTON TRIGGERS INSPIRATORY/EXPIRATORY CAP. MEASUREMENTS
// If Right button is pressed after left button, respirometer enters expiratory cap. mode
// If after 1 second,  no left button is pressed, respirometer continues with inspiratory cap. mode
if(buttons & BUTTON_LEFT){

  //INSPIRATORY MODE SELECTED
  lcd.clear();
  lcd.print("Insp. Cap.");
  lcd.setCursor(0,1);
  lcd.print("Mode Sel.");
  delay(2000);

  //Giving the Option for Expiratory Capacity Measurements
  lcd.setCursor(0,0);
  lcd.clear();
  lcd.print("For Exp. Cap.");
  lcd.setCursor(0,1);
  lcd.print("Press Right Btn");

  //Time delay of around ~1s to make switch to expiratory measurements.
  for(int k=0; k<1000; k++){
    buttons = lcd.readButtons();
    if(buttons & BUTTON_RIGHT){
      expiratoryCapacity();
    }
    delay(1);
  }
  lcd.clear();

    inspiratoryCapacity();
  }

  //RIGHT BUTTON TRIGGERS FLOW MEASUREMENTS
  if(buttons & BUTTON_RIGHT){
    flowMode();
  }

  //UP BUTTON TRIGGERS TIDAL VOLUME MODE
  if(buttons & BUTTON_UP){
    tidalVolume();
  }

  //DOWN BUTTON TRIGGERS CALIBRATION MODE
  if(buttons & BUTTON_DOWN){
    calibrate();
  }
  //SELECT BUTTON CHANGES PNEUMOTECH
  if(buttons & BUTTON_SELECT){
    //Selection Mode
    changePtech();
  }


 }
}
```

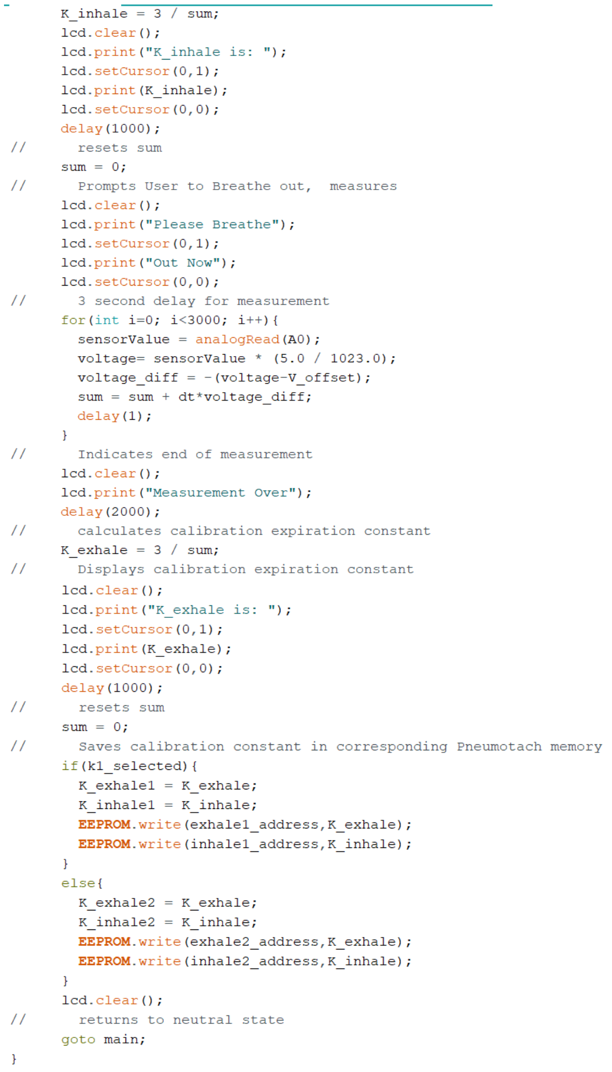


```
        lcd.clear();
        lcd.print("K_exhale is: ");
        lcd.setCursor(0,1);
        lcd.print(K_exhale);
        lcd.setCursor(0,0);
        delay(1000);
//        resets sum
        sum = 0;
//        Saves calibration constant in corresponding Pneumotach memory
        if(k1_selected){
          K_exhale1 = K_exhale;
          K_inhale1 = K_inhale;
          EEPROM.write(exhale1_address,K_exhale);
          EEPROM.write(inhale1_address,K_inhale);
        }
        else{
          K_exhale2 = K_exhale;
          K_inhale2 = K_inhale;
          EEPROM.write(exhale2_address,K_exhale);
          EEPROM.write(inhale2_address,K_inhale);
        }
        lcd.clear();
//        returns to neutral state
        goto main;
}
```

SECTION C



```
void calibrate(){
        //Prints Mode Selected
        lcd.clear();
        lcd.print("Calibration Mode");
        lcd.setCursor(0,1);
        lcd.print("Selected");
        delay(2000);
        //Prompts User to Breathe In
        lcd.setCursor(0,0);
        lcd.clear();
        lcd.print("Please Breathe");
        lcd.setCursor(0,1);
        lcd.print("In Now");
        lcd.setCursor(0,0);
//        Gives user 3 seconds to breathe in, measures
        sum=0;
        for(int i=0; i<3000; i++){
          sensorValue = analogRead(A0);
          voltage= sensorValue * (5.0 / 1023.0);
          voltage_diff = voltage-V_offset;
          sum = sum + dt*voltage_diff;
          delay(1);
        }
//        Indicates end of measurement
        lcd.clear();
        lcd.print("Measurement Over");
        delay(2000);
//        Calculates calibration inspiration constant and displays
```



```
        K_inhale = 3 / sum;
        lcd.clear();
        lcd.print("K_inhale is: ");
        lcd.setCursor(0,1);
        lcd.print(K_inhale);
        lcd.setCursor(0,0);
        delay(1000);
//        resets sum
        sum = 0;
//        Prompts User to Breathe out,  measures
        lcd.clear();
        lcd.print("Please Breathe");
        lcd.setCursor(0,1);
        lcd.print("Out Now");
        lcd.setCursor(0,0);
//        3 second delay for measurement
        for(int i=0; i<3000; i++){
          sensorValue = analogRead(A0);
          voltage= sensorValue * (5.0 / 1023.0);
          voltage_diff = -(voltage-V_offset);
          sum = sum + dt*voltage_diff;
          delay(1);
        }
//        Indicates end of measurement
        lcd.clear();
        lcd.print("Measurement Over");
        delay(2000);
//        calculates calibration expiration constant
        K_exhale = 3 / sum;
//        Displays calibration expiration constant
        lcd.clear();
        lcd.print("K_exhale is: ");
        lcd.setCursor(0,1);
        lcd.print(K_exhale);
        lcd.setCursor(0,0);
        delay(1000);
//        resets sum
        sum = 0;
//        Saves calibration constant in corresponding Pneumotach memory
        if(k1_selected){
          K_exhale1 = K_exhale;
          K_inhale1 = K_inhale;
          EEPROM.write(exhale1_address,K_exhale);
          EEPROM.write(inhale1_address,K_inhale);
        }
        else{
          K_exhale2 = K_exhale;
          K_inhale2 = K_inhale;
          EEPROM.write(exhale2_address,K_exhale);
          EEPROM.write(inhale2_address,K_inhale);
        }
        lcd.clear();
//        returns to neutral state
        goto main;
}
```

SECTION D

```
void tidalVolume(){
//      Selects Pneumotach Calibration Constant
      if(k1_selected){
        K_exhale = K_exhale1;
        K_inhale = K_inhale1;
      }
      else{
        K_exhale = K_exhale2;
        K_inhale = K_inhale2;
      }
//      Prints Mode Selected
      lcd.clear();
      lcd.print("Tidal Volume");
      lcd.setCursor(0,1);
      lcd.print("Mode");
      delay(2000);
      lcd.setCursor(0,0);
      lcd.clear();
//      Prompts User to Breathe in
      lcd.print("Quietly Breathe");
      lcd.setCursor(0,1);
      lcd.print("In");
      lcd.setCursor(0,0);
//      Gives User 3 seconds to breathe in, measures
      for(int i=0; i<3000; i++){
        sensorValue = analogRead(A0);
        voltage= sensorValue * (5.0 / 1023.0);
        voltage_diff = voltage-V_offset;
        sum = sum + K_inhale*dt*voltage_diff;
        delay(1);
      }
//      creates variable to store volume inspired, resets sum
      float TV_in = sum;
      sum = 0;
//      prompts user to breathe out
      lcd.clear();
      lcd.print("Quietly Breathe");
      lcd.setCursor(0,1);
      lcd.print("Out");
      lcd.setCursor(0,0);
//      gives three seconds for user to breathe out, measures
      for(int i=0; i<3000; i++){
        sensorValue = analogRead(A0);
        voltage= sensorValue * (5.0 / 1023.0);
        voltage_diff = -(voltage-V_offset);
        sum = sum + K_exhale*dt*voltage_diff;
        delay(1);
      }
//      creates variables to store total volume and expired volume, resets sum
      float TV_out = sum;
      float TV_total = TV_in + TV_out;
      sum = 0;
//      prints out total tidal volume
      lcd.print("Total Tidal");
      lcd.setCursor(0,1);
      lcd.print("Volume: ");
      lcd.print(TV_total);
      lcd.print(" L");
      delay(2000);
      lcd.setCursor(0,0);
      lcd.clear();
//      returns to neutral state
      goto main;
}
```

SECTION E

```
void flowMode(){
//Selection of Pneumotach Calibration Constant
      if(k1_selected){
        K_exhale = K_exhale1;
        K_inhale = K_inhale1;
      }
      else{
        K_exhale = K_exhale2;
        K_inhale = K_inhale2;
      }
//      Print Mode Selected
      lcd.clear();
      lcd.print("Flow Mode");
      lcd.setCursor(0,1);
      lcd.print("Selected");
      delay(2000);
      lcd.setCursor(0,0);
      lcd.clear();
//      250 measurements taken
      for(int i=0; i<250; i++){
        sensorValue = analogRead(A0);
        voltage= sensorValue * (5.0 / 1023.0);
        voltage_diff = voltage-V_offset;
        buttons = lcd.readButtons();
//        Hysteresis Constant used as threshold for outputting flow measurements
        if(abs(voltage_diff)>hysteresis_const){
//          Distinguishes between which constant to use, inhale selected
          if(voltage_diff<0){

              lcd.clear();
              lcd.print("Flow is: ");
              lcd.setCursor(0,1);
//              Prints out instantaneous flow rate
              lcd.print(voltage_diff*K_inhale*60);
              lcd.print(" L/min");
              lcd.setCursor(0,0);
              delay(10);
            }
//            Distinguishes between which constant to use, exhale selected
            if(voltage_diff>0){
              lcd.clear();
              lcd.print("Flow is: ");
              lcd.setCursor(0,1);
//              Prints out instantaneous flow rate
              lcd.print(voltage_diff*K_exhale*60);
              lcd.print(" L/min");
              lcd.setCursor(0,0);
              delay(10);
            }
          }
//          Voltage difference below hysteresis constant, 0 flow printed
          else{
            lcd.clear();
            lcd.print("Flow is: ");
            lcd.setCursor(0,1);
            lcd.print("0.0 L/min");
            lcd.setCursor(0,0);
            delay(10);
          }
//          sums counter, moves to next measurement
          i++;
        }
        lcd.clear();
        sum=0;
//        returns to neutral state
        goto main;
}
```

SECTION F

```
  //INSPIRATORY MODE SELECTED
  lcd.clear();
  lcd.print("Insp. Cap.");
  lcd.setCursor(0,1);
  lcd.print("Mode Sel.");
  delay(2000);

  //Giving the Option for Expiratory Capacity Measurements
  lcd.setCursor(0,0);
  lcd.clear();
  lcd.print("For Exp. Cap.");
  lcd.setCursor(0,1);
  lcd.print("Press Right Btn");

  //Time delay of around ~1s to make switch to expiratory measurements.
  for(int k=0; k<1000; k++){
    buttons = lcd.readButtons();
    if(buttons & BUTTON_RIGHT){
      expiratoryCapacity();
    }
    delay(1);
  }
  lcd.clear();

  inspiratoryCapacity();
}
```

SECTION G

```
void inspiratoryCapacity(){
//Printing Insp. Mode Selected
      lcd.print("Insp. Cap.");
      lcd.setCursor(0,1);
      lcd.print("Mode Sel.");
      delay(2000);
      lcd.setCursor(0,0);
      lcd.clear();
    //Prompt for User to Breathe
      lcd.print("Strongly Breathe");
      lcd.setCursor(0,1);
      lcd.print("In Now");
      lcd.setCursor(0,0);
    //Selection of Pneumotach Calibration Constant
      if(k1_selected){
        K_inhale = K_inhale1;
      }
      else{
        K_inhale = K_inhale2;
      }
    //Three seconds of measurement
      for(int i=0; i<3000; i++){
        sensorValue = analogRead(A0);
        voltage= sensorValue * (5.0 / 1023.0);
        voltage_diff = voltage-V_offset;
        //Running calculation of volume
        sum = sum + K_inhale*dt*voltage_diff;
        delay(1);
      }
    //Print out the total Inspired Volume
      lcd.clear();
      lcd.print("Tot. Insp.");
      lcd.setCursor(0,1);
      lcd.print("Vol. is: ");
      lcd.print(sum);
      lcd.print("L");
      delay(2000);
      sum=0;
      lcd.clear();
      goto main;
}
```

SECTION H

```
void expiratoryCapacity(){
    //Expiratory Capacity Mode selected
    //      Prints out Mode Selected
      lcd.clear();
      lcd.print("Exp. Cap.");
      lcd.setCursor(0,1);
      lcd.print("Mode Sel.");
      delay(2000);
      lcd.setCursor(0,0);
      lcd.clear();
    //      Prompts User to Breathe In
      lcd.print("Strongly Breathe");
      lcd.setCursor(0,1);
      lcd.print("OUT");
      lcd.setCursor(0,0);
    //    Selects for correct pneumotach value
      if(k1_selected){
        K_exhale = K_exhale1;
      }
      else{
        K_exhale = K_exhale2;
      }
    //      delays for 3 seconds and measures
      for(int i=0; i<3000; i++){
        sensorValue = analogRead(A0);
        voltage= sensorValue * (5.0 / 1023.0);
        voltage_diff = -(voltage-V_offset);
        sum = sum + K exhale*dt*voltage diff;
        delay(1);
      }
    //      Prints out total expired volume
      lcd.print("Tot. Exp.");
      lcd.setCursor(0,1);
      lcd.print("Vol. is: ");
      lcd.print(sum);
      delay(2000);
    //      resets sum
      sum=0;
      lcd.clear();
    //      loops back to neutral state
}
```