ASPIRE – ometer

"Aspiring to bring you a better breathing experience"

Aimee McVey

TA: Kay Palopoli

Lab Partner: Adrienne Hawkes

***Abstract*—The purpose of the ASPIRE-ometer is to assess lung health and capability through the measurement of several parameters, including lung volumes and flowrates during normal, full capacity, and forceful breaths. The device works by measuring the pressure differential across a thin wire screen placed in the airflow path. This pressure differential is directly proportional to flowrate and can be manipulated to give volume. The ASPIRE-ometer is user-friendly, guiding the user through calibration and testing step-by-step and allowing repetition and correction for mistakes. After selecting component parameters to maximize resolution and minimize noise, the device was tested on two subjects with two separate pneumotachs. All measurements were found to be consistent across the pneumotachs, and all values but peak flow were similar to those available in literature, with percent errors of 18%, 7.08%, and 1.94% for tidal volume, inspiratory capacity, and expiratory capacity, respectively.**

I. INTRODUCTION

A. The Spirometer [1]

The spirometer is a device used to measure respiratory parameters, especially in the context of assessing lung health. It can differentiate between inhalation and exhalation values and measure both flowrates and volumes. The tool is often used to test for lung diseases, including asthma, cystic fibrosis, and Chronic Obstructive Pulmonary Disease (COPD), or to perform clinical Pulmonary Function Tests (PFTs) to quantify lung fitness. The measurements can then be used to prescribe a ventilator or medication or to assess lung health during treatment.

Typical measurements can be visualized via Figure 1 and contain the following: Tidal Volume (V_T) measures the amount of air ventilated in normal quiet breath. Vital Capacity (VC) is the maximum amount of air that can be ventilated with maximum effort. Inspiratory and Expiratory Capacity (IC/EC) are the maximum amounts of air that can be inspired or expired, respectively. Inspiratory and Expiratory Reserve Volumes (IRV/ERV) involve the amount of air that can be inspired or expired, respectively, beyond a normal tidal breath. Finally, Reserve Volume (RV) is the amount of air that remains in the lungs after maximum expiratory effort, and Minute Respiratory Volume (MRV) is the amount of air ventilated per minute during normal quiet respiration. In the ASPIRE-ometer, Tidal Volume, Inspiratory and Expiratory Capacity, and peak inhalation and exhalation flows are calculated.

B. Methods of Measurement

To measure these parameters, the spirometer contains a transducer known as a pneumotachometer, which linearly

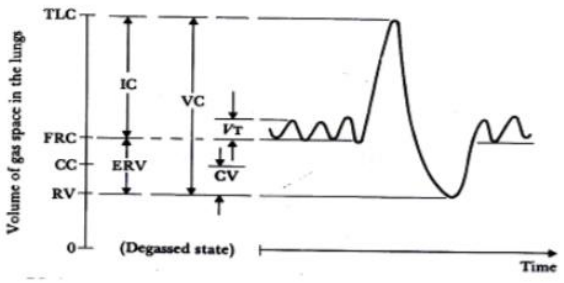

Figure 1: Depiction of typical spirometry measurements

transduces airflow into differential pressure in accordance with Equation 1 [2], where ΔP is difference in pressure across the screen, Q is flowrate, and R is resistance of the screen. To do this, pressure is sampled on both sides of a fine wire screen inserted into the airflow. This screen disrupts the flow to create the pressure differential. In this spirometry device, the transducer used was the MPX2010DP, which uses two input ports, one on each side of the screen, a strain gage, and a Wheatstone bridge to convert the differential pressure into a proportional differential voltage signal. After signal processing, a microprocessor can then read the user-friendly output of flowrates, which can be converted to volumes via Equation 2 [3].

$$\Delta P = QxR \tag{1}$$

$$Vol = \int Q\, dt \tag{2}$$

The system must show zero flow when airflow is indeed zero. As such, the DC offset must be removed from every measurement. In addition, the device must be calibrated with a known volume in order to report accurate volumes and flows. This is done via Equation 3, where K is the calibration constant, V is volume, and t is time, as the known volume is divided by the measured volume over time. For maximum accuracy, two calibration constants are calculated due to potential differences in flowrate: K^+ for inhalation and K^- for exhalation.

$$Known\ Volume = K \sum (V \Delta t) \tag{3}$$

C. Goals

The overall goal of the ASPIRE-ometer was to measure and display several typical spirometer measurements through a user-friendly interface. The user is carefully guided through each step of testing and prompted to error-check values along the way. If needed, restarting the device, retesting, or redisplaying results only requires the click of a button. The

ASPIRE-ometer provides a quick and easy experience in a mobile package – perfect for any user in any place.

II. Design Specifications [2]

The device will perform linearly for airflow ranging from 0 to 200 L/min, covering as much of the full available bit range of 0 to 1023 bits and voltage range of 0 to 5 V for the Arduino microprocessor as possible with the available components. It will display zero flow when the device is not being used and display flow rate in L/min at rest during testing. The device will contain calibration constants for two pre-calibrated and stored pneuomotachs and allow a third to be calibrated and stored. If desired, any of the three pneuomotachs will be able to be calibrated by the user with a 3 L syringe. The device will perform precise, accurate, and repeatable testing for tidal volume, inspiratory capacity, expiratory capacity, inhale peak flow, and exhale peak flow with a goal percent error of 10%. If values used to calculate measurements are not in normal ranges due to user error, the user will be informed and directed to take further action.

III. Hardware Design

A. Hardware Block Diagram

As seen in the hardware block diagram for the ASPIRE-ometer in Figure 2, the airflow information ultimately travels from the pneuomatch (left) to the user display on the LCD screen (right). The user breathes into the pneumotach, which contains a mesh screen. This mesh screen creates a pressure differential read and transduced into a proportional voltage signal by the pressure sensor (MPX2010DP). Temperature effects are minimized for this sensor, and it performs linearly for pressures from 0 to 10 kPa, well encompassing typical pressures resulting from breathing [3].

This voltage signal is amplified through an instrumentation amplifier and passed through a 4.8 Hz low pass filter to filter out noise (circuit further explained below). The instrumentation amplifier was selected to maximize the gain of the difference between the two signals from each side of the mesh screen; the gain of this amplifier is easily adjusted via a single resistor, as further explained via Equation 3. Maximizing gain thus maximizes the range of values the signal can cover, maximizing resolution and increasing accuracy of airflow measurements. Instrumentation amplifiers also provide high input impedance and common mode noise rejection. The cutoff frequency of 4.8 Hz was chosen to be as low as possible using available components to avoid complications with high frequency noise. The typical maximum human breathing pattern is 20 breaths per minute, a much lower frequency than 4.8 Hz.

The microprocessor (Arduino) then converts the voltage signal into flowrate and volume for spirometry testing. A microprocessor allows the device to remain small and portable. The LCD display provides instant direction and feedback for the user. The Arduino and the LCD display are powered by the computer but could be externally sourced if necessary.

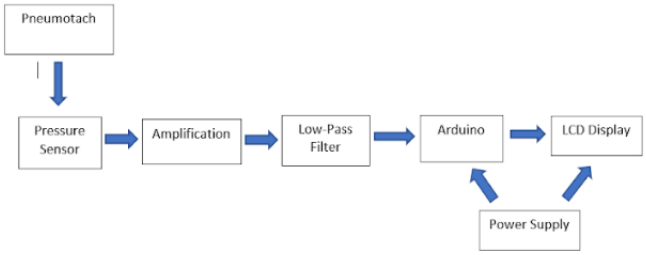

Figure 2: Hardware block diagram

B. Circuit Diagram

The circuit components of the ASPIRE-ometer are shown in Figure 3, from the pressure sensor (left) to the analog input pin of the Arduino (A0). The MPX2010DP is connected to ground and the Arduino power supply and outputs the differential voltage transduced from the differential pressure across the mesh screen as $+V_O$ and $-V_O$. The instrumentation amplifier (INA126) takes these two outputs as inputs and amplifies the difference between them. The resistor value R_G was chosen as 10 Ω, the minimum resistance available, to maximize the gain at 8,005 V/V, as calculated via Equation 4 [4].

$$G_{INA126} = 5 + \frac{80\ k\Omega}{R_G} \tag{4}$$

$$G = 5 + \frac{80\ k\Omega}{10\ \Omega} = 8{,}005\ V/V$$

The reference value was set at 2.5 V via a 10 kΩ potentiometer set at approximately 5 kΩ to split the 5V Arduino power source in half. The potentiometer can be adjusted to readjust the baseline if needed. A reference voltage of 2.5 V increases the baseline voltage from 0 to 2.5 V, changing the input voltage range from -2.5 to +2.5 V to 0 to 5 V. This step was necessary because the Arduino cannot intake negative voltages. After amplification, the signal passes through a passive low pass filter comprised of a 330 kΩ resistor and a 0.1 µF capacitor for a cutoff frequency of 4.8 Hz (Equation 5).

$$f_C = \frac{1}{2\pi RC} \tag{5}$$

$$f_C = \frac{1}{2\pi(330\ k\Omega)(0.1\ \mu F)} = 4.8\ Hz$$

After filtering, the Arduino reads the voltages at the analog pin A0.

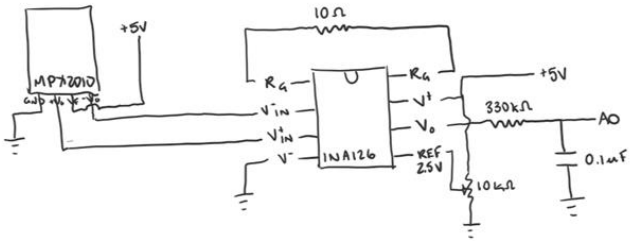

Figure 3: Circuit diagram with component values

IV. SOFTWARE DESIGN

A. Offset, Calibration, and Pneumotach Selection

The software flow diagram for the spirometer can be followed in Figure 4, while the software itself can be found in the Appendix. The ASPIRE-ometer experience begins with a simple question: "Calibrate?" From here, the user is guided step by step through spirometry testing, with questions, value checking, and commentary along the way. If the pneumotach in use has been previously calibrated, the user doesn't need to waste time re-calibrating. If the user is incorporating a new pneunomatach into testing, calibration is simple and straightforward.

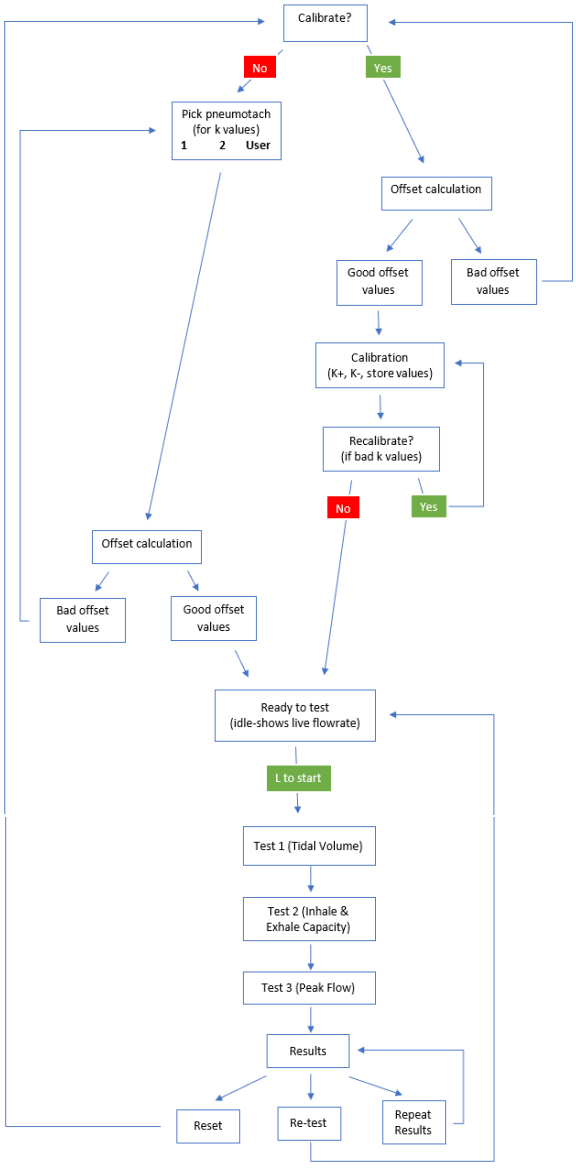

Figure 4: Software flow diagram

If the user decides to calibrate, the offset calculation is performed: a simple average of the first twenty voltage values read. The offset is necessary due to setup of the Arduino itself, which reads voltages in the 0 to 5 V range in terms of a 0- to 1023-bit range and will be subtracted from every ensuing measurement. Because the reference voltage was set to 2.5 V for the instrumentation amplifier, an offset of approximately 512 bits is appropriate. This ensures that inhalation will return positive flow values, and exhalation will return negative flow values. The spirometer automatically checks if the offset is in the 490- to 530-bit range; if it is not, the device will automatically restart. This check was implemented due to occasional random errors in calculation and has subsequently decreased error margin in airflow measurements substantially.

If the offset is calculated in range, the user can calibrate and store K+ and K- values with a 3 L syringe. The device will then ask if the user would like to recalibrate in case values are out of range or a second pass is necessary. If so, the calibration steps will repeat. If not, the spirometer will be ready to test. If the user decides not to calibrate, they can choose to use calibration values from two pre-calibrated pneumotachs or from the user-calibrated pneumotach. After selection, the offset calculation will be performed in the same manner as stated above. Then, the device will be ready to test.

B. Spirometry Testing

While waiting for the user to start testing, live flowrate is displayed onscreen. Here, the user can ensure no breathing gives a flowrate near 0 L/min and can practice breathing into the pneumotach. Once the user decides to proceed, three tests are performed in sequence. First, the user is instructed to take a normal breath in five seconds for calculation of tidal volume. Then, the user is directed to take a full capacity breath to determine inhale and exhale capacity. Finally, the user is ordered to take a forceful breath for calculation of inhale and exhale peak flow. The three tests are administered in sequence to maximize testing ease and ensure full results.

Pressure differentials from inhalation and exhalation are converted into voltage values as the user breathes, which the software then converts into flowrate in L/min via Equation 6, where K is the calibration constant in L/(bit*s) and V is the voltage read from the circuit in bits.

$$flowrate = K * V * 60 \quad (6)$$

To obtain tidal volume, positive voltages corresponding to inhalation are converted to flowrate and summed for one normal breath. This flowrate sum is then converted to volume in L via Equation 7, where the 0.01 refers to the time delay of 10 ms between measurements.

$$V = flowrate * 0.01 / 60 \quad (7)$$

Inspiratory and expiratory capacity are calculated in the same way, though for a full capacity breath and divided into inspiratory and expiratory capacity via positive and negative voltage values, respectively. Peak flow is calculated via conversion to flowrate (Equation 6) for a forceful breath. The maximum and minimum flowrates yield peak flow for inhalation and exhalation, respectively.

C. Results and Repetition

When testing is finished, results for all three tests are shown in succession: tidal volume, inhale capacity, exhale capacity,

inhale peak flow, and exhale peak flow. After scrolling is complete, the screen stalls on three options: reset, re-test, or repeat results. From here, the user can reset the device to the beginning, re-test for confirmation of values, or repeat the results shown previously. These options are available so that the user can use a different pneumotach if desired, confirm testing results via a second trial, or simply take more time to view results.

V. Instructions for Use

The ASPIRE-ometer provides a user-driven experience. From the beginning, the user can choose their testing path and proceed at their own pace – through calibration, pneuomotach selection, and options to restart, repeat testing, or display results again. To ensure accuracy and precision, no knowledge is presumed of the user. Values are constantly checked automatically or via user interaction, and the user can repeat steps if measurements appear incorrect.

A. Manual Calibration

To begin, choose whether to calibrate manually (R: yes) or to use pre-calibrated calibration constants (L: no). To calibrate manually, follow the onscreen instructions using the 3 L calibration syringe. First connect the syringe to the pneumotach in the orientation denoted on the pneumotach (the side marked). To obtain K^+, press the up button and then pull the syringe handle, fully extending the rod over three seconds. The pull should be complete when the screen displays "stop pull." K^+ will then be displayed. To obtain K^-, follow a similar procedure. Press the down button when ready and push the syringe handle, fully compressing the rod over three seconds. The push should be complete when the screen displays "stop push." K^- will then be displayed. As the screen directs, if the magnitude of the displayed calibration constants is not between 10 and 20, the device should be recalibrated (R: yes). If the calibration constants are in range, the device is ready for testing (L: test).

B. Automatic Calibration

If manual calibration is chosen, the pneumotach in use must be selected to ensure correct calibration constants. If these constants do not match the pneumotach in use, measurements will be inaccurate. Two pneumotachs are pre-programmed (L: 1, R: 2) The third selection contains the latest stored constants by the user (Up: User). Once this selection is made, the device is ready for testing (L: test).

C. Offset Failure

If manual calibration is selected, offset calculation occurs before calibration. If automatic calibration is selected, offset calculation occurs after pneumotach selection. If offset calculation fails, the device will display "Offset failure. Restarting," and the device will restart to the "Calibrate?" screen. The user can then proceed from the beginning. If this occurs more than twice, the device should be restarted.

D. Testing

Before beginning testing, live flowrate is displayed onscreen in L/min. If not breathing, the flowrate should be close to zero. If it is not, straighten the pneumotach tubing. When ready, press the left button to begin testing. The first test is for tidal volume. Take one normal breath when directed. The user has five seconds. The second test is for inspiratory and expiratory capacity. Take one full capacity breath, filling up the lungs and expelling as much air as possible when directed. The user again has five seconds. The third and final test is for peak flow. Take one forceful breath when directed, inspiring and expelling air as fast as possible. The allotted time remains five seconds.

E. Results

Results will be automatically displayed in order of testing: tidal volume, inspiratory capacity, expiratory capacity, peak flow while inhaling, and peak flow while exhaling. After scrolling is complete, a menu will appear asking if the user would like to restart the device (Up: Reset), repeat the results (L: Results), or repeat testing (R: Test). Restarting will return the spirometer to the initial "Calibrate?" screen. Results will display the results in order again. Test will repeat the three tests in sequence, followed by results.

VI. Evaluation of Performance

A. Calibration Constants

Calibration constants K^+ and K^- were determined through correlation between flowrate received from the flow meter and analogread values given by the Arduino in bits ranging from 0 to 1023. The correlation constant was determined as the slope of the best fit of the points, as seen in Figures 5-8. Each of the fits displayed linear behavior, with a lowest R^2 value of 0.994. The pneumotachs thus displayed linear behavior between 0.5 and 4 L/s, or between 30 and 240 L/min, meeting the specification of linearity between 0 and 200 L/min. For visibility, the calibration constant values displayed on the LCD screen are multiplied by 1000.

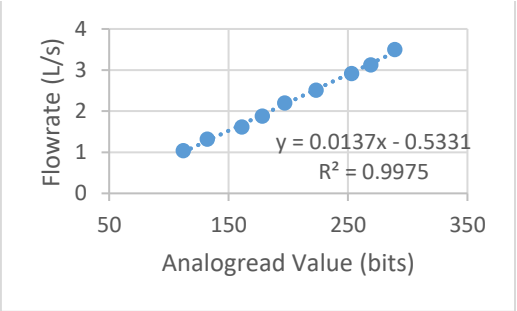

Figure 5: Determination of K^+ for P1: 0.01375

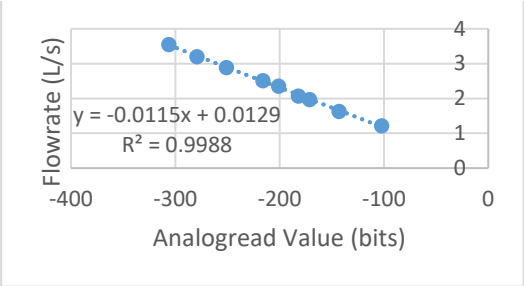

Figure 6: Determination of K^- for P1: -0.01151

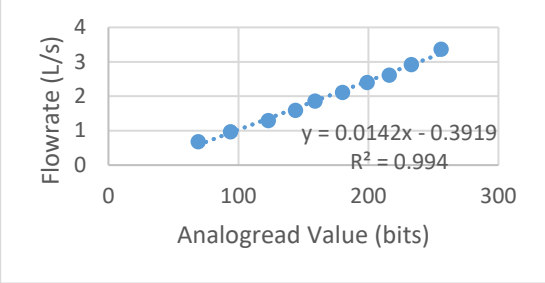

Figure 7: Determination of K^- for P2: -0.01428

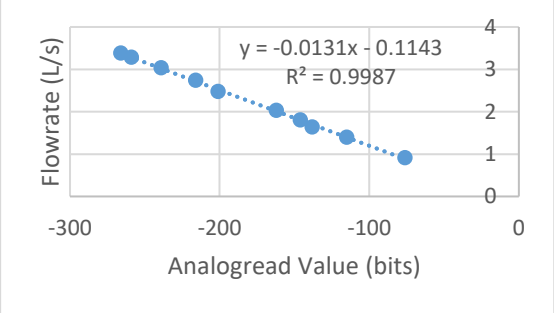

Figure 8: Determination of K^- for P2: -0.01313

As seen in Table 1, the correlation constants between pneumotachs and for inhalation (K^+) and exhalation (K^-) differ slightly. The difference between pneumotachs is most likely due to construction, as each was 3D printed, resulting in slight individuality for each one from the printing process. The differences between K^+ and K^- are most likely due to the printing process as well; each half of each pneumotach was printed separately and the mesh screen inserted. In addition, the screen could have directionality such that the pressure drop in one direction isn't necessarily the same as in the other.

Table 1: Correlation constants for two pneumotachs (P1, P2)

	$1000*K^+$	$1000*K^-$
P1	13.75	-11.51
P2	14.28	-13.13

Following constant determination, flowrates determined and displayed by the ASPIRE-ometer were compared with those of the flow meter. The flowmeter and device flow rates agreed for flowrates between 0 and 250 L/min.

B. Measurement Testing

Testing of the ASPIRE-ometer was conducted to determine the accuracy and precision of its measurements. The results of testing the two pre-calibrated pneumotachs on two subjects can be seen in Table 2. Both subjects were 21-year-old females. Average values across pneumotachs and subjects correspond closely to literature values, with the exception of expiratory peak flow [5, 6]. Average tidal volume for a female is 0.50 L, compared to the value of 0.59±0.16 measured here, for 18% error. Average inspiratory and expiratory capacities are 2.4 and -3.1 L, respectively, compared to averages of 2.23±0.23 and -3.04±0.65 L measured here, for errors of 7.08 and 1.94%.

The average expiratory peak flow for a 20-year-old female of 63" (approximate height of the two subjects) is -420 L/min [6], compared to a measured value of -251.19±23.8 L/m, for an error of 40.2%. Inspiratory peak flow was not readily available in literature. While this measured value was significantly different from literature values, some of this difference could be attributed to the stance of the subjects (sitting) compared to the typical testing mode of standing. In addition, peak flows are tested last in the ASPIRE-ometer sequence after two preceding breaths, potentially decreasing measured values. Finally, peak flows are dependent on forceful breaths creating turbulent flow in the spirometer. These high pressures probably reached the non-linear range of the pneumotach's capabilities.

Spirometry measurements will vary with height, living situation (oxygen content of air), and fitness levels. The values measured here were fairly accurate with the exception of peak flow. However, all values were precise, with low standard deviations.

Table 2: Test results using two subjects (S1, S2) and two pneumotachs (P1, P2)

	V_T (L)	IC (L)	EC (L)	I Peak Flow (L/min)	E Peak Flow (L/min)
S1P1	0.41	2.19	-1.68	240.15	-251.54
	0.43	2.73	-3.38	204.09	-259.95
S1P2	0.54	2.21	-3.59	217.74	-296.94
	0.66	2.08	-3.78	213.18	-216.68
S2P1	0.57	1.93	-2.76	182.61	-227.61
	0.54	2.12	-2.88	196.42	-249.60
S2P2	0.91	2.29	-3.00	144.15	-248.60
	0.62	2.27	-3.23	153.25	-258.58
Avg	**0.59**	**2.23**	**-3.04**	**193.95**	**-251.19**
St Dev	**0.16**	**0.23**	**0.65**	**32.6**	**23.8**
%Error	**18.0**	**7.08**	**1.94**	**N/A**	**40.2**
Female	0.50[5]	2.4[5]	-3.1[5]	N/A	-420[6]

C. *Overall Evaluation*

The ASPIRE-ometer is easy to use. It ensures the user is ready for the next step before and during pre-testing and testing through clear prompts and questions. In addition, displays are carefully timed for reading and understanding, and the device offers options to repeat tests, results, or restart if mistakes occur. For further ease, automatic calibration is available with built-in constants. If manual calibration is selected, step-by-step instruction is available.

Included failure cases improve accuracy. The ASPIRE-ometer automatically checks and recalculates the offset if outside of the preferred range. The device encourages recalculation of manual calibration constants if outside of the preferred range. Furthermore, retesting is offered to check or confirm measurements.

Overall, the spirometer functions well with reasonable test results. While the 3D printed pneumotachs and tubing can be sensitive, resulting in an inconsistent offset measurement, this problem was solved via a failure case. The offset does still occasionally give a value of 900. However, this problem can be solved via a hard restart. Regardless, the device gives volume values with 2-18% error, proven via 3 L syringe comparisons and subject testing, and produces consistent measurements.

VII. LESSIONS LEARNED

The ASPIRE-ometer is a user-friendly device capable of measuring five clinically important values with accuracy and precision. Components chosen and design choices made resulted in a sleek physical product with maximized resolution and minimized noise – but only following multiple design iterations. On the hardware side, differing resistor values had to be tested to adjust the gain and the reference voltage of the op-amp. On the software side, calibration and testing steps were continually written and checked until the final process resulted. Troubleshooting was relied on for designing the circuit, calibrating voltages to flowrates, and creating an easy spirometry testing experience. Overall, the design experience was challenging, but rewarding, as each piece of the ASPIRE-ometer came together to create a presentable final product.

REFERENCES

[1] Duke BME 354 Final Project Protocol: Design and Build a Spirometer, March 23, 2019.

[2] NXP Products, "Freescale Semiconductor – MPX2010 Series," NXP Products, 2008. [Online]. Available: https://www.nxp.com/docs/en/data-sheet/MPX2010.pdf. [Accessed: Apr. 27, 2019).

[3] Duke BME Lab 8 Protocol: Arduino Spirometry, January 15, 2019.

[4] Burr-Brown Products from Texas Instruments, "MicroPower Instrumentation Amplifier – Single and Dual Versions," Burr-Brown Products from Texas Instruments, 2005. [Online]. Available: http://www.ti.com/product/INA2126. [Accessed: Apr. 27, 2019].

[5] Ganong, William. "Fig 35-7". *Review of Medical Physiology* (21st ed).

[6] Nunn, A. J., and I. Gregg. 1989. New regression equations for predicting peak expiratory flow in adults. Br. Med. J. 298: 1068-1070.

VIII. APPENDIX: ARDUINO CODE

```
//Final Project: Spirometer
//Aimee McVey and Adrienne Hawkes

#include <Wire.h>
#include <Adafruit_RGBLCDShield.h>
#include <utility/Adafruit_MCP23017.h>
Adafruit_RGBLCDShield lcd = Adafruit_RGBLCDShield();
#define WHITE 0x7
#include <EEPROM.h>

//initialize variables
int pin = A0;
float flowrate = 0;
int Vo;
int var1[300];
int var2[300];
float Kp = 0;
float Km = 0;
long offset = 0;
int state;
int buttonstate;
int substate = 0;
float IC = 0;
float EC = 0;
float FRmax = 0;
float FRmin = 0;
float Vt = 0;
int eeAddress = 16;
float Kp1;
float Km1;
float Kp2;
float Km2;
float Kp3;
float Km3;

void setup() {
  lcd.begin(16, 2); //begin LCD
  lcd.setBacklight(WHITE);
  lcd.clear();
  pinMode(pin, INPUT); //take in A0 - analog input
  Serial.begin(9600);
  state = 0;
  buttonstate = 0;
}

void loop() {

  if (state == 0) {     //select calibration
    manualcal();
  }

  if (state == 1) {
    offsetcalc();           //offset calculation
    if (substate == 1)   {  //allows for offset checking
      calibrate();          //manual calibration
```

```
    }
  }

  if (state == 2) {
    offsetcalc();   //offset calculation if auto calibration
  }

  if (state == 3) {
    disp();         //continually displays flow until started
  }

  if (state == 4) {
    test1();        //normal breath - Vt
  }

  if (state == 5) {
    test2();        //full capacity breath - IC, EC
  }

  if (state == 6) {
    test3();        //forceful breath - peak flow
  }

  if (state == 7) {
    results();      //display results
  }

  if (state == 8) {
    repeat();       //select restart, retest, results
  }
}

void manualcal() {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Calibrate??");
  lcd.setCursor(0, 1);
  lcd.print("L: no  R: yes");
  uint8_t buttons = lcd.readButtons();
  uint8_t i = 0;
  buttons = lcd.readButtons();
  int buttonstate2 = 0;
  delay(200);

  if (buttons) {
    if (buttons & BUTTON_RIGHT) { //yes - manual calibration
      state = 1;
    }
    if (buttons & BUTTON_LEFT) { //no - auto calibration
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Pick pneum");   //select pneumotach used
      lcd.setCursor(0, 1);
      lcd.print("L:1  R:2  U:User");
      delay(200);
      while (buttonstate2 == 0) {
        buttons = lcd.readButtons();
```

```
      if (buttons) {
        if (buttons & BUTTON_LEFT) { //pneum 1
          delay(100);
          //retrieve stored values for E1
          EEPROM.get(0, Kp1);
          Kp = Kp1 * 4;
          EEPROM.get(4, Km1);
          Km = Km1 * 4;
          buttonstate2 = 1;
        }

        if (buttons & BUTTON_RIGHT) { //pneum 2
          delay(100);
          //retrieve stored values for E2
          EEPROM.get(8, Kp2);
          Kp = Kp2 * 4;
          EEPROM.get(12, Km2);
          Km = Km2 * 4;
          buttonstate2 = 1;
        }

        if (buttons & BUTTON_UP) { //user created vals for any pneum
          delay(100);
          //retrieve stored values for E3
          EEPROM.get(16, Kp3);
          Kp = Kp3 * 4;
          EEPROM.get(20, Km3);
          Km = Km3 * 4;
          Serial.println("Kp");
          Serial.println(Kp * 1000);
          Serial.println("Km");
          Serial.println(Km * 1000);
          buttonstate2 = 1;
        }
       }
       state = 2;
      }
    }
  }
}

void offsetcalc() {
  substate = 0;
  //take average of 20 readings for offset
  for (int n = 0; n <= 19; n++) {
    offset += analogRead(pin); //read voltage out from circuit
    delay(20);
  }
  offset = offset / 20;
  Serial.println("offset");
  Serial.println(offset);
  if (offset > 530 || offset < 490) { //if offset bad, redo
    lcd.clear();
    lcd.print("offset failure");
    state = 0;
    lcd.setCursor(0, 1);
    lcd.print("restarting");
```

```
    delay(2000);
    return state;
  }

  if (state == 1) { //if manual cal, proceed to cal
    substate = 1;
  }
  if (state == 2) { //if auto cal, proceed to test
    state = 3;
  }
  delay(1000);
}

void calibrate() {
  uint8_t buttons = lcd.readButtons();
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Push up for K+");
  lcd.setCursor(0, 1);
  lcd.print("when ready");
  delay(1000);
  long s1 = 0;
  long s2 = 0;
  uint8_t i = 0; //initialize buttons

  while (buttonstate == 0) {
    buttons = lcd.readButtons();
    if (buttons) {
      if (buttons & BUTTON_UP) {
        delay(100);
        buttonstate = 1;
      }
    }
  }

  if (buttonstate == 1) {
    s1 = 0;
    lcd.clear();
    lcd.print("K+: pull 3s now");
    for (int i = 0; i <= 299; i++) {
      var1[i] = analogRead(pin); //read and store in array
      var1[i] = var1[i] - offset;
      s1 += var1[i];
      delay(10);
    }
    lcd.clear();
    lcd.print("stop pull"); // tell user to stop calibrating
    delay(1000);
    Kp = 3 / (s1 * 0.01); //calc calibration constant
    lcd.clear();
    lcd.print("K+ = ");
    lcd.print(Kp * 1000); //mult by 1000 to make it more visible
    delay(2000);
    buttonstate = 0;
  }

  lcd.clear();
```

```
lcd.setCursor(0, 0);
lcd.print("Push down for K-");
lcd.setCursor(0, 1);
lcd.print("when ready");
delay(1000);

while (buttonstate == 0) {
  buttons = lcd.readButtons();
  if (buttons) {
    if (buttons & BUTTON_DOWN) {
      delay(100);
      buttonstate = 2;
    }
  }
}

if (buttonstate == 2) {
  s2 = 0;
  lcd.clear();
  lcd.print("K-: push 3s now");
  for (int i = 0; i <= 299; i++) {
    var2[i] = analogRead(pin); //read and store in array
    var2[i] = var2[i] - offset;
    s2 += var2[i];
    delay(10);
  }
  lcd.clear();
  lcd.print("stop push"); // tell user to stop calibrating
  delay(1000);
  Km = 3 / (s2 * 0.01); //L/(bit*s)
  lcd.clear();
  lcd.print("K- = ");
  lcd.print(Km * 1000);
  delay(1000);
  buttonstate = 0;
}

//Store user-calibrated vals in EEPROM
Kp3 = Kp / 4;
EEPROM.put(eeAddress, Kp3);
Serial.println(Kp3);
Serial.println(eeAddress);
eeAddress += sizeof(Kp3);
Km3 = Km / 4;
EEPROM.put(eeAddress, Km3);
Serial.println(Km3);
Serial.println(eeAddress);
eeAddress += sizeof(Km3);

int buttonstate2 = 0;
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("If |vals| not");
lcd.setCursor(0, 1);
lcd.print("btw 10-20, redo"); //instruct user to retest if bad vals
delay(2000);
lcd.clear();
```

```
  lcd.setCursor(0, 0);
  lcd.print("Recalibrate?");
  lcd.setCursor(0, 1);
  lcd.print("L: test  R: yes");
  delay(1000);

  while (buttonstate2 == 0) {
    buttons = lcd.readButtons();
    if (buttons) {
      if (buttons & BUTTON_RIGHT) { //restart calibration
        delay(100);
        state = 1;
        buttonstate2 = 1;
      }
      if (buttons & BUTTON_LEFT) { //proceed to testing
        delay(100);
        state = 3;
        buttonstate2 = 1;
      }
    }
  }
  return state;
}

void disp() {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Ready to test");
  delay(1000);
  uint8_t buttons = lcd.readButtons();
  uint8_t i = 0;
  while (state == 3) {
    buttons = lcd.readButtons();
    flow();
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Rate:");
    lcd.print(flowrate);  //continuously display airflow on LCD (L/min)
    lcd.print(" L/min");
    lcd.setCursor(0, 1);
    lcd.print("L: Start Testing");
    delay(200);
    buttons = lcd.readButtons();
    if (buttons) {
      if (buttons & BUTTON_LEFT) {
        delay(100);
        state = 4; //starts testing
      }
    }
  }
}

void test1() {
  //Tidal Volume (Vt): amt of air ventilated in a normal quiet breath
  lcd.clear();
  lcd.print("Begin testing");
  delay(2000);
```

```
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Take 1 normal");
  lcd.setCursor(0, 1);
  lcd.print("breath in 5s");
  delay(250);
  float vol = 0;
  float flowratesum = 0;
  for (int i = 0; i < 499; i++) {
    delay(10);
    Vo = analogRead(pin) - offset; //read voltage in bits
    if (Vo > 0) {
      flowrate = Kp * Vo * 60; //L/(bit*s)*bits*60 s/min
      flowratesum += flowrate;
    }
  }
  Vt = flowratesum * 0.01 / 60; //convert flow (L/min) to vol (L)
  state = 5;
}

void test2() {
  //Inspiratory Capacity (IC): max amount of air inspired after
  //normal tidal expiration
  //Expiratory Capacity (EC): max amount of air expired after
  lcd.clear();
  lcd.print("2nd test");
  delay(2000);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("1 full capacity");
  lcd.setCursor(0, 1);
  lcd.print("breath in 5s");
  delay(250);
  float vol = 0;
  float volmaxinhale = 0;
  float volmaxexhale = 0;
  float flowratesuminhale = 0;
  float flowratesumexhale = 0;
  for (int i = 0; i < 499; i++) {
    delay(10);
    Vo = analogRead(pin) - offset; //read voltage in bits
    if (Vo > 0) { //inhale
      flowrate = Kp * Vo * 60;
      flowratesuminhale += flowrate;
    }
    else if (Vo < 0) { //exhale
      flowrate = -Km * Vo * 60;
      flowratesumexhale += flowrate;
    }
  }
  IC = flowratesuminhale * 0.01 / 60; //convert flow (L/min) to vol (L)
  EC = flowratesumexhale * 0.01 / 60;
  state = 6;
}

void test3() {
  //Peak flow
```

```
  lcd.clear();
  lcd.print("3rd test");
  delay(2000);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("1 forceful");
  lcd.setCursor(0, 1);
  lcd.print("breath in 5s");
  delay(250);
  float maxflowrate = 0;
  float minflowrate = 0;
  float flowrateP = 0;
  float flowrateM = 0;
  for (int i = 0; i < 499; i++) {
    delay(10);
    Vo = analogRead(pin) - offset; //read voltage in bits
    //differentiate btw inhale & exhale
    if (Vo > 0) { //inhale
      flowrateP = Kp * Vo * 60;
    }
    else if (Vo < 0) { //exhale
      flowrateM = -Km * Vo * 60;
    }
    if (flowrateP > maxflowrate && Vo > 0) { //inhale
      maxflowrate = flowrateP;
    }
    else if (flowrateM < minflowrate && Vo < 0) { //exhale
      minflowrate = flowrateM;
    }
  }
  FRmax = maxflowrate;
  FRmin = minflowrate;
  state = 7;
}

void flow() {
  //Factor in offset and calibration constant to give airflow (L/min)
  Vo = analogRead(pin) - offset; //read voltage in bits
  if (Vo > 0) {    //if Vo > 0, inhale, use K+
    flowrate = Kp * Vo * 60; //L/(bit*s)*bits*60 s/min
  }
  else if (Vo < 0) {  //if Vo < 0, use K-
    flowrate = -Km * Vo * 60;
  }
}

void results() {
  lcd.clear();
  lcd.print("Results");
  delay(2000);
  lcd.clear();
  lcd.print("Tidal Volume:");
  lcd.setCursor(0, 1);
  lcd.print(Vt);
  lcd.print(" L");
  delay(2000);
  lcd.clear();
```

```
  lcd.print("Inhale Cap:");
  lcd.setCursor(0, 1);
  lcd.print(IC);
  lcd.print(" L");
  delay(2000);
  lcd.clear();
  lcd.print("Exhale Cap:");
  lcd.setCursor(0, 1);
  lcd.print(EC);
  lcd.print(" L");
  delay(2000);
  lcd.clear();
  lcd.print("Inh. Peak Flow:");
  lcd.setCursor(0, 1);
  lcd.print(FRmax);
  lcd.print(" L/min");
  delay(2000);
  lcd.clear();
  lcd.print("Ex. Peak Flow:");
  lcd.setCursor(0, 1);
  lcd.print(FRmin);
  lcd.print(" L/min");
  delay(2000);
  state = 8;
}

void repeat() {
  uint8_t buttons = lcd.readButtons();
  lcd.clear();
  lcd.print("Repeat? ");
  delay(1000);
  lcd.clear();
  lcd.print("Up:Reset");
  lcd.setCursor(0, 1);
  lcd.print("L:Results R:Test");
  delay(1000);
  uint8_t i = 0; //initialize buttons

  while (state == 8) {
    buttons = lcd.readButtons();
    if (buttons) {
      if (buttons & BUTTON_UP) {
        delay(100);
        state = 0; //repeat calibration
      }
      if (buttons & BUTTON_RIGHT) {
        delay(100);
        state = 3; //repeat testing
      }
      if (buttons & BUTTON_LEFT) {
        delay(100);
        state = 7; //repeat results
      }
    }
  }
}
```