<https://www.instructables.com/id/Low-Cost-Spirometer/>

[Adafruit MPRLS Ported Pressure Sensor Breakout](https://cdn-learn.adafruit.com/downloads/pdf/adafruit-mprls-ported-pressure-sensor-breakout.pdf?timestamp=1587075092)

<https://adafruit.github.io/Adafruit_MPRLS/html/class_adafruit___m_p_r_l_s.html>

**Pseudocode:**

Include Libraries:

~~MPRLS.h~~

TFT Display (depends which one we use, and if we decide to run it in this code)

Wire.h

define Reset pin to -1

define EOC pin to -1

Adafruit\_MPRLS mpr1 = Adafruit\_MPRLS(reset, eoc)

Adafruit\_MPRLS mpr2 = Adafruit\_MPRLS(reset, eoc) //Two objects for two sensors

Float pressure1, pressure2, massFlow, volumetricFlow, volume, area, rho, dt

buzzer pin = a

button pin = b

ledpin1 = c

ledpin2 = d

Setup:

Serial.begin

pinMode(buzzer, OUTPUT)

pinMode(ledPin1, OUTPUT)

pinMode(ledPin1, OUTPUT)

pinMode(button1, INPUT) // Alarm cancel, override activated

pinMode(button2, INPUT) // Alarm Override reset

Loop:

pressure1 = mpr1.readPressure() //will be in hPa

pressure2 = mpr2.readPressure()

Pressure\_hpa = pressure2 - pressure1

massFlow = 1000\*sqrt((abs(pressure\_hpa)\*2\*rho)/((1/(pow(area\_2,2)))-(1/(pow(area\_1,2)))));

volumetricFlow = massFlow / rho

volume = volumetricFlow\*dt + volume //Tidal Volume

delay(X)

//////////Alarm System for Spirometer///////////////

// While you see volume limits here, these should be extreme ones. You don't want the alarm to go off all the time for slight increases/decreases. The negative feedback valve code should take care of this.

buttonState2 = digitalRead(button2)

If buttonState2 == 1:

override == 0

// If the override reset button is triggered, alarm system is reactivated (override needs to be 0 to activate alarm system

if volume > X && override == 0: //X being the upper limit, override not trigger

tone(buzzer, 1000) //1000 Hz sound signal, can be changed

digitalWrite(ledPin1, HIGH) // Runs buzzer for 1s on/off while volume too high

delay(1000)

noTone(buzzer)

buttonState1 = digitalread(button1)

If buttonState1 == 1: //If you press the button to cancel alarm, override

noTone(buzzer)

digitalWrite(ledPin1, LOW)

override = !override

// So override is 1, doctor will manually adjust valve, alarm will stop ringing

If volume < Y && override == 0: // Lower limit

tone(buzzer, 2000) // Diff frequency

digitalWrite(ledPin2, HIGH) // Diff light

delay(1000)

noTone(buzzer)

buttonState1 = digitalread(button1)

If buttonState1 ==1:

noTone(buzzer)

digitalWrite(ledPin2, LOW)

override = !override //Override is 1 now, as it was 0 to trigger alarm

**General Steps:**

IF button is pressed to take a spirometer reading:

READ analog pin (will be read in as digital)

CALCULATE mass flow from the pressure reading

massFlow = 1000\*sqrt((abs(pressure\_pa)\*2\*rho)/((1/(pow(area\_2,2)))-(1/(pow(area\_1,2)))));

CONVERT mass flow → volumetric flow

volFlow = massFlow/rho; //rho is the density

CONVERT volumetric flow → volume

Volume = volumeFlow \* dt + volume

delay(figure out frequency here)

Not sure if the sensor we are using needs a conversion to pressure ( I think it already outputs pressure readings with certain libraries)