# Purpose and Importance of a Breathing Sensor

Traditional measurement methods for monitoring a person’s breathing are often manual, these approaches are inefficient, prone to error, and not continuously tracked, yet irregular breathing or apnea (i.e. temporary cessation of breathing) can be life-threatening if not noticed.

Thus, this design includes a wearable breathing sensor, in which it alerts the caregiver to any abnormal respiration, thereby enabling prompt intervention.

# Stretchable Conductive Sensor Working Principle

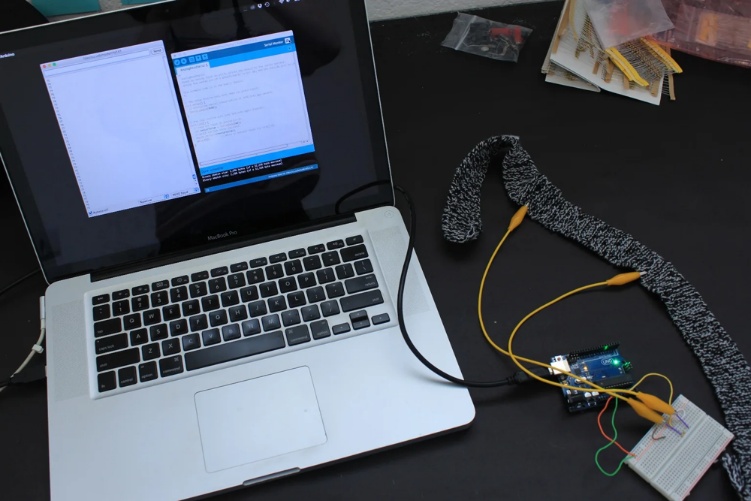


Figure 1: Example of a stretchable knitted breathing sensor band connected to an Arduino for testing.

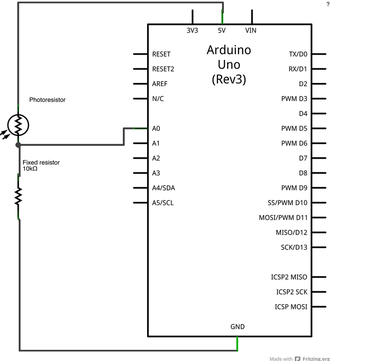
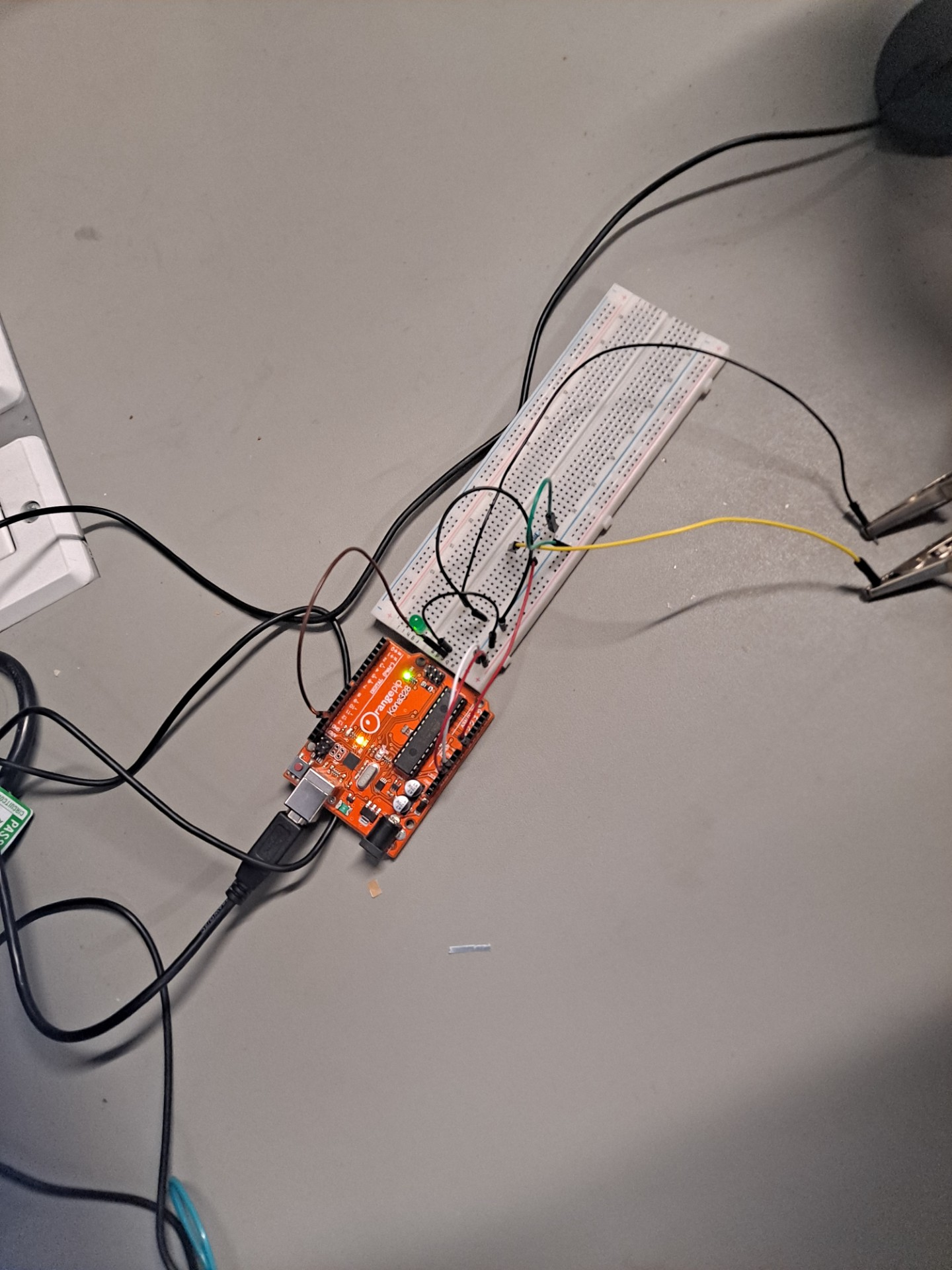
This design introduces a soft, stretchable conductive band placed around the chest. As the user inhales, the chest expands, so does the band; exhaling lets it relax. The band itself is a stretchable sensor, and it is made of conductive material (e.g. carbon, silver, conductive rubber[1] and so forth) whose electrical resistance changes with length. When stretched, the conductive particles of the band spread apart, and therefore electrons take a less direct pass. This increases the sensor’s resistance. When the band contracts, the resistance decreases. This design therefore uses these changes of electrical resistance to model the chest motion (i.e. the breathing motion).

However, Arduino cannot measure resistance directly, it can only measure voltage. So, there are two ways to covert resistance into voltage:

* First, using the Ohm’s Law (), and let (), then . This approach is very accurate but it is not easy to achieve, especially for a simple microcontroller-based design – it requires additional power regulation components to achieve a stable current source.
* Second, using a voltage divider, it is a simple circuit in which the breathing sensor is connected in series with a fixed resistor, the output volage (i.e. the voltage measured by the Arduino) is such that: , as the user inhales, increases, so does . When the user exhales, decreases, reducing . This creates a continuous breathing waveform, and can be analysed to determine breathing rate (a.k.a. BPM, breaths per minute).

This design adopts the second approach, namely, the voltage divider, since it is easy to achieve, cheaper, no extra power regulation needed (unlike the first approach), and allows direct integration with microcontrollers that read analog voltage data.

# Circuit Design



# Source Code

1. /\*

2.   AnalogReadSerial

3.   Reads an analog input on pin 0, prints the result to the Serial Monitor.

4.   Turns an LED on/off based on sensor input.

5.   Connect the anode (long leg) of the LED to pin 13 on the Arduino.

6.   Connect the cathode (short leg) of the LED to GND through a 220Ω resistor.

7. \*/

8.

9. // Define the LED pin

10. int ledPin = 13;

11.

12. // Define the sensor pin

13. int sensorPin = A0;

14.

15. void setup() {

16.   // Initialize serial communication at 9600 bits per second:

17.   Serial.begin(9600);

18.

19.   // Initialize the LED pin as an output:

20.   pinMode(ledPin, OUTPUT);

21. }

22.

23. void loop() {

24.   // Read the input from the sensor on analog pin 0:

25.   int sensorValue = analogRead(sensorPin);

26.

27.   // Print out the value to the Serial Monitor:

28.   Serial.println(sensorValue);

29.

30.   // Define a threshold for when to turn the LED on/off

31.   if (sensorValue > 180) {  // Adjust this value based on your sensor's range

32.     digitalWrite(ledPin, HIGH);  // Turn the LED ON

33.   } else {

34.     digitalWrite(ledPin, LOW);   // Turn the LED OFF

35.   }

36.

37.     delay(100);// Delay for stability

38. }

Line 25, analogRead(sensorPin) reads the voltage output from the sensor. If R\_sensor increases (inhale), so does V\_out, which leads to higher analogRead(sensorPin) value.

Line 28 prints the raw voltage readings, which reflect breathing motion.

From Line 31 to Line 35, it is a threshold-based LED control in which the LED is “blinking” in sync with breathing. The dealy(100) function in line 37 introduces a short pause (100ms) between readings. Without this function, the Arduino would print too many values per unit time, making it hard to analyse.

# References

[1]<https://www.adafruit.com/product/519#:~:text=In%20a%20%27relaxed%27%20state%2C%20the,a%20way%20to%20measure%20stretching>

[2]

# Image Source

[1]<https://content.instructables.com/FDX/KBXS/IAI1PVHY/FDXKBXSIAI1PVHY.jpg?auto=webp&frame=1&width=600&height=1024&fit=bounds&md=MjAxNS0wNi0wNCAwODo0MDo1OS4w>