Doyle *Bulletin of the National Research Centre* (2022) 46:258
https://doi.org/10.1186/s42269-022-00946-2

Bulletin of the National Research Centre

RESEARCH

Open Access

Design and implementation of a simple and inexpensive respiratory synchronization control platform

John Doyle[1,2*]

Abstract

Background: In a number of clinical and research settings, it is desirable to have an individual breathe in a particular fixed pattern (respiratory synchronized breathing). The purpose of this brief technical report is to show how a control system for this purpose can be easily and inexpensively developed using an Arduino UNO microcontroller platform.

Results: We programmed an Arduino UNO microcontroller to develop a respiratory timing system with selectable respiratory rate and inspiratory to expiratory ratio. Test subjects are instructed to breathe in when the light-emitting diode (LED) is illuminated and breathe out when the LED is dark. Both the duration of inspiration and that of expiration can be easily adjusted by the user to meet various requirements. The system was tested and found to function satisfactorily.

Conclusions: An Arduino UNO microcontroller was used to develop a respiratory timing system. This platform is likely to be of value to clinicians and investigators looking for a simple and inexpensive system for respiratory synchronized breathing.

Keywords: Arduino microcontroller, Interbreath interval control, Respiratory rate control, Respiratory synchronization, Respiratory timing system

Background

Typically, most adults breathe between 12 and 20 cycles per minute (0.2–0.3 Hz). In some clinical and research settings, it is desirable to have the test subject or patient breathe in a particular fixed pattern (respiratory synchronized breathing, paced breathing). For example, paced breathing has been advocated as a nonpharmacological means of reducing stress and anxiety (Clark and Hirschman 1990; Blumenstein et al. 1995; Laborde et al. 2017), as a behavioral adjunct in the treatment of hypertension (Hateren et al. 2015; Schein et al. 2009; Elliott and Izzo 2006; Cernes and Zimlichman 2017; Brenner et al. 2020; Adler et al. 2019; Viskoper et al. 2003), as a treatment for overactive bladders (Huang et al. 2019) as well as a treatment for menopausal hot flushes (Huang et al. 2015), as a means to improve sleep (Tsai et al. 2015), and as a means to manage food craving (Meule and Kübler 2017). Mechanistically, it has been hypothesized that synchronized breathing techniques producing slow breathing patterns at a frequency near 0.1 Hz (6 breaths per minute) "promotes behavioral relaxation and baroreflex resonance effects that maximize heart rate variability" and serve to "elicit resonant and coherent features in neuro-mechanical interactions that optimize physiological function" (Noble and Hochman 2019).

Another important application of paced breathing is as a research tool to better understand the effect of breathing on heart rhythm patterns (Wilhelm et al. 2004; Sin et al. 2010). Paced breathing studies can be particularly valuable in this instance because respiration changes the heart position due to movement of the diaphragm,

*Correspondence: djdoyle@hotmail.com
[1] Department of General Anesthesiology, Cleveland Clinic, Cleveland, USA
Full list of author information is available at the end of the article

affecting obtained cardiac waveforms such as the electrocardiogram, the blood pressure wave and the photoplethysmograph waveform. Additionally, changes in conductivity in thoracic tissues and changes in thoracic blood volume caused by lung inflation will similarly influence cardiac waveforms.

Two approaches have been proposed to deal with this issue. The earliest proposed method to control respiratory-induced variations in cardiac events was to require that the tested individual remains apneic (hold his or her breath) during the data recording interval. Unfortunately, the limitations of this approach are strikingly obvious.

A more practical technique is to have the tested individual breathe in a predetermined pattern under the control of an electronic timing apparatus with a light (e.g., light-emitting diode (LED) indicator (or sound source) signaling when to breathe in and when to breathe out. A typical arrangement here might be to have the test subject breathe at 15 breaths per minute (interbreath interval of 4000 ms) with an inspiratory to expiratory (I:E) ratio of 1:2 (i.e., inspiration for 1333 ms and expiration for 2666 ms). In this setting, the start of inspiration and the start of expiration are recorded electronically along with the cardiac signals of interest, possibly with a view to later segment the recorded cardiac signals into inspiratory and expiratory phases.

For slow breathing relaxation protocols requiring breathing at 6 breaths per minute, one option would be an inspiration phase of 3333 ms with an expiration phase of 6666 ms, maintaining the I:E ratio at 1:2.

In this report, we show how such a platform can be easily and inexpensively developed using an Arduino UNO microcontroller platform.

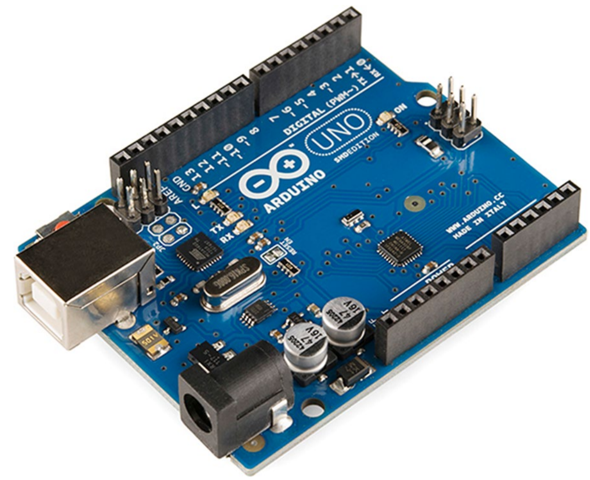

Fig. 1 The Arduino UNO is a popular and inexpensive open-source microcontroller board based on the ATmega328P microcontroller. The board features 14 digital pins and 6 analog input pins (featuring 10-bit ADC resolution). Also, 6 of the 14 digital pins can be configured as pseudo-analog output pins, with the analog outputs being simulated using PWM (pulse width modulation). The Arduino UNO can be powered using a USB cable or externally via a 7–20-V source, such as an ordinary 9-V battery. It also features a clock speed of 16 MHz, support for external interrupts, and a built-in LED driven by digital pin 13. Programming the Arduino UNO (and other family members) is usually done via the Arduino integrated development environment (IDE), available as a free download for the Windows, macOS, and Linux operating systems. *Image credit*: https://commons.wikimedia.org/wiki/File:Arduino_Uno_-_R3.jpg. Image used under the Creative Commons Attribution 2.0 Generic license

Methods

The Arduino UNO microcontroller (Fig. 1) served as the centerpiece of this project. It is the best-known Arduino

Table 1 Synopsis of pinout and other technical characteristics of the Arduino UNO microcontroller

Input and output: Each of the 14 digital pins on the Arduino UNO can be used as an input or output, using the pinMode(), digitalWrite(), and digitalRead() functions. These pins operate at 5 V and each can supply or receive a maximum of 40 mA. Each pin has an internal pull-up resistor, disconnected by default. In addition, some pins have specialized functions, as discussed below
Serial: Pins 0 (RX) and 1 (TX). These are used to receive (RX) and transmit (TX) TTL level serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip
External interrupts: Pins 2 and 3. These pins can be configured to trigger an interrupt on a low value, on a rising or falling edge, or on a change in value. See the attachInterrupt() function for details
Pulse width modulation (PWM): Digital pins 3, 5, 6, 9, 10, and 11 can provide an 8-bit PWM output with the analogWrite() function. They are identified with a tilde (~) symbol
SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support the SPI serial communication protocol using the SPI library
LED: 13. There is a built-in LED connected to digital pin 13. When the pin is in the HIGH state, the LED is on, and when the pin is LOW, the LED is off
Analog inputs: The UNO has 6 analog input lines, labeled A0 through A5, each providing 10-bit resolution (i.e., 1024 different values). By default, they measure from ground to 5 V, though it is possible to change the upper end of their range using the AREF pin and the analogReference() function. Additionally, some pins have specialized functionality
TWI: A4 or SDA pin and A5 or SCL pin. Support TWI communication using the wire library
AREF: Reference voltage for the analog inputs. Used with analogReference()
Reset: Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board

Modified from https://www.electroschematics.com/7958/arduino-uno-pinout/

```
/* RESPIRATORY SYNCHRONIZATION PROGRAM FOR ARDUINOS

D. John Doyle MD PhD. Revision 1.3. Tested December 11, 2021

This program makes an Arduino into a respiratory synchronizer operating
at a rate of 15 breaths per minute with an I:E ratio of 1:2. (These
parameters reflect the natural breathing pattern of normal adults).
In its simplest form a respiratory synchronizer, used primarily in
cardiorespiratory research applications, illuminates an LED when the
test subject should breathe in, and turns off the LED when the test
subject should breathe out. (A more complex option would be to use
two LEDS, one illuminated when the test subject is to breathe in and
a second LED illuminated when the test subject is to breathe out.)
Note that below that IT=4000/3 while ET=2 x IT (I:E being 1:2). The
4000 term used here is the interbreath interval in milliseconds
(15 breaths per minute). Note also that I:E ratio stands for the
inspiratory to expiratory breathing time ratio.*/

// Define LED driving pin and timing information
unsigned int LED=13;
// Inspiratory time in milliseconds
unsigned int InspiratoryTime =1333;
// Expiratory time in milliseconds
unsigned int ExpiratoryTime =2666;

void setup() {
// open the serial port at 9600 bps
Serial.begin(9600);
// Set LED driving pin to OUTPUT mode
pinMode(LED, OUTPUT);
}

void loop() {
// Set LED driving pins to HIGH state (on)
digitalWrite(LED, HIGH);
// wait InspiratoryTime milliseconds
delay(InspiratoryTime);
// Set LED driving pins to LOW state (off)
digitalWrite(LED, LOW);
// wait ExpiratoryTime milliseconds
delay(ExpiratoryTime);
}
```

Fig. 2 This program makes an Arduino into a respiratory synchronizer / breathing metronome operating at a rate of 15 breaths per minute with an I:E ratio of 1:2

microcontroller as it is both popular and very inexpensive[1]and has been used many biomedical applications (Veldscholte et al. 2021; Wandling et al. 2021; Holovatyy et al. 2020; Hoang et al. 2021; Zuckerberg et al. 2020; Cobo et al. 2020; Vallejo et al. 2020; Chen and Li 2017; Das et al. 2017).

[1] Chinese clones of the Arduino UNO are available on eBay for under $10.00, including world-wide shipping.

This microcontroller is an open-source product based on the ATmega328P chip containing 14 digital input/output pins and 6 analog input pins (featuring 10-bit analog-to-digital (ADC) resolution). The Arduino UNO can be powered using a USB cable or externally via a 7–20-V source, such as an ordinary 9-V battery. It also features a clock speed of 16 MHz, support for external interrupts, and a built-in light-emitting diode (LED) driven by digital pin 13.

Programming the Arduino UNO (and other family members) is usually done via the Arduino integrated development environment (IDE), and available as a free download for the Windows, macOS, and Linux operating systems at www.arduino.cc. Table 1 provides more technical details concerning the Arduino UNO.

In this technical report, we offer the reader two project designs to consider (Figs. 2 and 3), one design (Fig. 2) being especially simple and the other far more flexible but also more complex (Fig. 3). The first design requires no hardware other than the Arduino UNO microcontroller itself and the host computer, while the more advanced design uses a 16 character by 2-line LCD display and a 4×1 push button array to allow the user to select the desired respiratory rate and I:E ratio.

In either the simple design or the advanced design, the subject is asked to breathe in for a specific time span (known as the inspiratory time) and then to breathe out for another specified time (known as the expiratory time).

In the simple design version, an LED connected to Arduino digital pin 13 is used as the breathe in/breathe out control signal. In this instance, one could also connect a second external LED in parallel if desired or use an external LED connected to another digital output pin. In this program version, the respiratory rate and I:E ratio are "hard wired" into the program code itself; changing these parameters requires changing the program itself (not a difficult task, however) and running the program again.

In the more advanced respiratory metronome design featured in Figs. 4 and 5, a multi-color LED is used. Here, GREEN corresponds to a command to breathe in and RED corresponds to a command to breathe out. In addition, a BLUE light is used to indicate that the system has entered a special "setup" mode where the user can select the desired respiratory rate and I:E ratio using assigned control keys.

As shown in Fig. 2, the Arduino computer code for the basic application is not complicated. First, we define the variable "InspiratoryTime" as the inspiratory time in milliseconds and the variable "ExpiratoryTime" as the expiratory time in milliseconds. The program is then a loop whereby an LED is repeatedly illuminated for "InspiratoryTime" milliseconds and then turned off for "ExpiratoryTime" milliseconds, with the test subject simply instructed to breathe in when the LED is illuminated and breathe out when the LED becomes dark again.

In the case of the more advanced program illustrated in Fig. 4, the code has been enhanced to provide a 2-line by 16-character LCD display for displaying the respiratory rate and I:E ratio (see Fig. 5) as well as a 4×1 array of pushbuttons (K1, K2, K3 and K4). These pushbuttons are used to enter (K1) and exit (K4) setup mode as well as to cycle through available selections for respiratory rate (K3) and I:E ratio (K4) when in setup mode.

Results

The bottom portion of Fig. 3 provides some sample results acquired using the Dataq DI-1100 multichannel data acquisition system, which was configured to record the inspiration/expiration control signals produced by the Arduino microcontroller. These two signals have a 5 V positive voltage corresponding to LED illumination during commands for inspiration (signal 1) and for expiration (signal 2).

Discussion

Several variations in the basic arrangement described can be envisioned. One variation could be to use two LEDs, one for requesting inspiration and the other for requesting expiration. Another variation would be to add an "inspiratory hold" LED signifying that the test subject should hold his or her breath (neither breathe in or breathe out) following inspiration, and to do so until the expiration LED is illuminated. In this last instance, a third variable (let's call it "InspiratoryHoldTime") would need to be introduced. In such a case, one might use an inexpensive three-color "GYR" LED assembly with green signifying "breathe in," yellow signifying "hold your breath" and red signifying "breathe out."

(See figure on next page.)

Fig. 3 TOP: Enhancement of the program shown in Fig. 2 recorded to additionally produce two electrical digital status outputs, first a 5-V level on Pin 7 during inspiration (and zero volts during expiration) and also a 5-V level on Pin 8 during expiration (with zero volts during inspiration). BOTTOM: Strip chart recording of Arduino UNO Pins 7 and 8 obtained using the DATAQ DI-1100 data acquisition system. The top graph is a plot against time of the inspiratory control signal, while the bottom graph is for the expiratory control signal. The software package supporting the strip chart system is by DATAQ. Known as WinDaq, it is available for free download to users of their data acquisition systems, with some USB-based data acquisition systems priced well under 100 dollars US (e.g., Model DI-1100)

```
// Define LED driving pin and signal pins for I and E
unsigned int LED=13;
unsigned int ISignalPin=7; / 5 volt level when in inspiration phase
unsigned int ESignalPin=8; / 5 volt level when in expiration phase

// Inspiratory time in milliseconds
unsigned int InspiratoryTime=1333;
// Expiratory time in milliseconds
unsigned int ExpiratoryTime=2666;

void setup() {
// open the serial port at 9600 bps (for debugging)
Serial.begin(9600);
// Set LED driving pin to OUTPUT mode
pinMode(LED, OUTPUT);
pinMode(ISignalPin, OUTPUT);
pinMode(ESignalPin, OUTPUT);
}

void loop() {
// Set LED driving pins to HIGH state (on)
digitalWrite(LED, HIGH);
digitalWrite(ISignalPin, HIGH);
digitalWrite(ESignalPin, LOW);
// wait InspiratoryTime milliseconds
delay(InspiratoryTime);
// Set LED driving pins to LOW state (off)
digitalWrite(LED, LOW);
digitalWrite(ISignalPin, LOW);
digitalWrite(ESignalPin, HIGH);
// wait ExpiratoryTime milliseconds
delay(ExpiratoryTime);
}
```

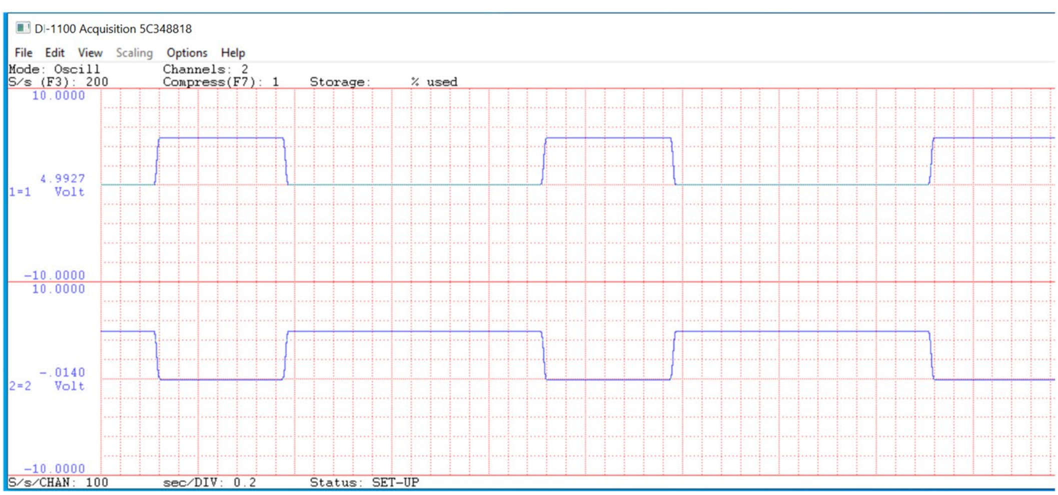

Fig. 3 (See legend on previous page.)

```
// Respiratory Synchronization Metronome Revision 16
//
// December 12, 2021
//
// ADVANCED VERSION
// (Simple version posted elsewhere uses "hardwired" but easily changed parameters.)
//
//
// D. John Doyle MD PhD
// djdoyle@hotmail.com)
//
// THIS IS TESTED CODE
//
//
// This program is an LED based breathing metronome, turning on a green LED when it is time to breathe in (inspiration)
// and turning on a red LED when it is time to breathe out again (expiration). Respiratory rates of 6,8,10,12,15,20,25,30,
// and 35 bpm are supported, along with I:E ratio options of I:E = 1.0,1.5,2.0,2.5,3.0, and 3.5
//
// Data is displayed on a 16 character by 2-line LCD display connected to the Arduino by an I2C interface.
// A 4 x 1 pushbutton array labeled K1 to K4 are used to enter programming mode and then used to select rate and I:E ratio
// The default respiratory pattern is 15 breaths per minute with an I:E ratio of 1:2.0.
//
// The use of arrays for RR, IBI, E and EF help provide an easy-to-understand data storage arrangement
//
//  SUBROUTINES
//   void setup() - you know this one
//   void loop() - you know this one as well
//   void DisplayParameters() - display rate and I:E ratio on LCD display
//   void ParameterUpdate() - update the rate and I:E ratio choices
//
//  LIBRARY USE
//   LiquidCrystal_I2C lcd(0x27,16,2)
//
//  The 4 x 1 pushbutton array labeled K1 to K4 is used as follows
//   pinMode(10, INPUT_PULLUP); // K4 Input - Request to Update Parameters
//   pinMode(11, INPUT_PULLUP); // K3 Input - Cycle through parameter A (e.g., I:E Ratio)
//   pinMode(12, INPUT_PULLUP); // K2 Input - Cycle through parameter B (e.g., Rate)
//   pinMode(13, INPUT_PULLUP); // K1 Input - Return to Standard Operation
//
//
//  PRINTING TO THE SERIAL MONITOR:
//   Serial.println (RRindex); // respiratory rate choice - array counter
//   Serial.println (IBI[RRindex]); // interbreath interval for current selection
//   Serial.println (IF); // Inspiratory fraction
//   Serial.println (InspiratoryTime);
//   Serial.println (ExpiratoryTime);
//   Serial.println (" ");
//
//
//
//

#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27,16,2);  // set the LCD address to 0x27 for a 16 chars and 2 line display

  int RR[9]   ={6,8,10,12,15,20,25,30,35};
  int IBI[9]  ={10000,7500,6000,5000,4000,3000,2400,2000,1714}; // InterBreath Interval(IBI)in milliseconds
  float E[6]  ={1.0,1.5,2.0,2.5,3.0,3.5}; // E part of I:E ratio (I part is always 1.0)
  float EF[6] ={0.5,0.6,0.667,0.714,0.75,0.777}; // E part expressed as a fraction of total respiratory cycle
  int RRindex=4;      // default to 15 bpm respiratory rate
  int Eindex=2;       // default to I:E ratio of 1:2.0
  int Blue_LED=5;     // LED Pin=5 for BLUE LED
  int Green_LED=6;    // LED Pin=6 for GREEN LED
  int Red_LED=7;      // LED Pin=7 for RED LED

  void setup() {
  Serial.begin(9600); // For debugging
  lcd.init();      // initialize the lcd;
  lcd.backlight(); // backlight on
  lcd.clear();
  lcd.setCursor(0,0);   // lcd.setCursor(col, row) }
  pinMode(10, INPUT_PULLUP); //K4
  pinMode(11, INPUT_PULLUP); //K3
  pinMode(12, INPUT_PULLUP); //K2
```

Fig. 4 (Three panels). Enhancement of the original program shown in Fig. 2 to provide both a 2-line by 16-character LCD display (for displaying the respiratory rate and I:E ratio) as well as a 4 × 1 array of pushbuttons (K1, K2, K3 and K4). These pushbuttons are used to enter (K1) and exit (K4) setup mode as well as to cycle through available selections for respiratory rate (K3) and I:E ratio (K4) when in setup mode

```
pinMode(13, INPUT_PULLUP); //K1
pinMode(Blue_LED, OUTPUT);
pinMode(Green_LED, OUTPUT);
pinMode(Red_LED, OUTPUT);
digitalWrite(Blue_LED, LOW);    // turn the LED off
digitalWrite(Green_LED, LOW);   // turn the LED off
digitalWrite(Red_LED, LOW);     // turn the LED off
}

void loop(){
int InspiratoryTime;
int ExpiratoryTime;
float IF; // Inspiratory fraction
IF=(1-EF[Eindex]);
ExpiratoryTime=  IBI[RRindex]*EF[Eindex];
InspiratoryTime= IBI[RRindex] - ExpiratoryTime;
Serial.println (RRindex);
Serial.println (IBI[RRindex]);
Serial.println (IF);
Serial.println (InspiratoryTime);
Serial.println (ExpiratoryTime);
Serial.println (" ");
digitalWrite(Green_LED,HIGH); //Inspiration LED
digitalWrite(Red_LED,LOW);    //Expiration LED
delay(InspiratoryTime);
digitalWrite(Green_LED,LOW); //Inspiration LED
digitalWrite(Red_LED,HIGH);  //Expiration LED
delay(ExpiratoryTime);

// check for parameter changes
// if pushbutton K1 is depressed, call change parameter subroutine
if (digitalRead(13)==LOW) ParameterUpdate();
DisplayParameters();
}

void DisplayParameters(){
// display parameters on LCD screen
lcd.clear();
lcd.setCursor(0,0);   // lcd.setCursor(col, row)
lcd.print("Resp Rate: ");
lcd.print(RR[RRindex]);
lcd.setCursor(0,1);
lcd.print("I:E Ratio: 1:" );
lcd.print(E[Eindex]);
}

void ParameterUpdate(){
digitalWrite(Blue_LED, HIGH);    // turn BLUE LED on
lcd.clear();
alpha: lcd.setCursor(0,0);   // lcd.setCursor(col, row)
// if pushbutton K2 is depressed, change RRindex
if (digitalRead(12)==LOW) {RRindex++; if (RRindex==9) RRindex=0;}
// if pushbutton K3 is depressed, change Eindex
if (digitalRead(11)==LOW) {Eindex++; if (Eindex==6) Eindex=0;}
DisplayParameters();
// if pushbutton K4 is depressed, return to main program
delay(1000);
if (digitalRead(10)==HIGH) goto alpha;
digitalWrite(Blue_LED, LOW);    // turn BLUE LED off
}

// K1 = pin 13 = enter reprogram subroutine
// K2 = pin 12 = cycle respiratory rate: 6,8,10,12,15,20,25,30,35,6,8,10,12,15,20,25,30,35,6,8,10,12,15,20,25,30,35, etc.
// K3 = pin 11 = cycle I:E ratio: 1:1.0,1:1.5,1:2.0,1:2.5,1:3.0,1:3.5,1:1.0,1:1.5,1:2.0,1:2.5,1:3.0,1:3.5, etc.
// K4 = pin 10 = exit reprogram subroutine
```

Fig. 4 continued

The use of two p-Channel JFET transistors (e.g., 2SJ177) operating as switches can be used to divide the obtained acoustic signal into separate inspiratory and expiratory channels, i.e., one analog channel containing only inspiratory sounds and another analog channel containing only expiratory sounds (Fig. 6). In this arrangement, each JFET transistor is configured such that the acoustic signal can pass through from source (*S*) to drain (*D*) only when the gate (*G*) is presented with a zero-voltage control voltage from the microcontroller. In the inspiratory channel,

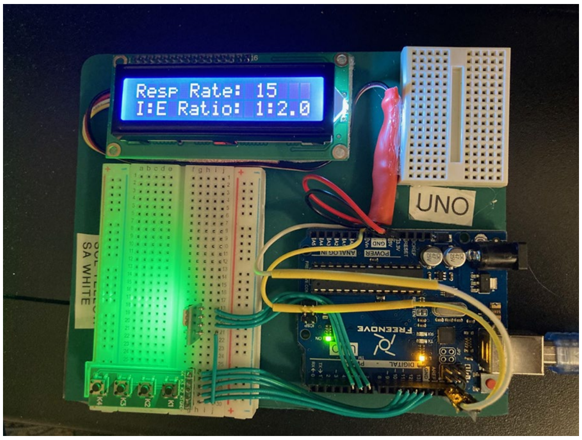

Fig. 5 Photograph of the prototype for the advanced system corresponding to the code in Fig. 4. Note the presence of a 2-line by16-character LCD display (for displaying the respiratory rate and I:E ratio) as well as a 4 × 1 array of pushbuttons (K1, K2, K3 and K4) used to setup (initialize) the operation of the unit. Notice also the illuminated green LED

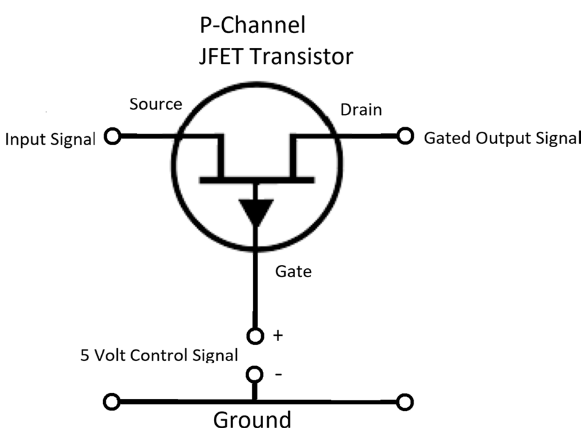

Fig. 6 If needed, one can use P-Channel JFET transistors (e.g., 2SJ177) operating as switches to divide the obtained acoustic signal into separate inspiratory and expiratory channels. In this arrangement, each of two P-Channel JFET transistors is configured such that the acoustic signal passing through from source (*S*) to drain (*D*) is blocked when the gate (*G*) is presented with a positive control voltage from the microcontroller. In the inspiratory channel, the microcontroller outputs a 5-V (blocking) control signal during the time that the LED is off (i.e., during expiration), while in case of the expiratory channel, the microcontroller outputs a 5-V (blocking) control signal during the time that the LED is turned on (i.e., during inspiration). Note that, the above circuit must be duplicated, once for the inspiratory channel and once again for the expiratory channel

the microcontroller outputs a zero-voltage control signal during the time that the LED is illuminated (and a 5-V control signal otherwise), while in the expiratory channel, the microcontroller outputs a zero-voltage control signal during the time that the LED is turned off (and a 5-V control signal otherwise).

Note that, it is likely that in any research setting, all respiratory timing variables would remain the same across all test subjects, making it more practical to simply define the relevant timing variables as fixed values at the time of program initialization rather than to ask the operator to enter them via a keypad or by other means each time the program is run.

Finally, in many research settings, the state of all control signals would be recorded in real time on a real or virtual strip chart recorded, along with physiological signals such as the electrocardiogram (ECG), phonocardiogram (PCG) or photoplethysmogram (PPG). Digitally recording both respiratory events and cardiac signals together offers the possibility of further exploring the effect of respiration on cardiovascular events. Example exploratory questions one might ask concern the phase of respiration that provides for (Clark and Hirschman 1990) the loudest first heart sound (Blumenstein et al. 1995), the largest photoplethysmographic signal amplitude or (Laborde et al. 2017) the smallest QRS amplitude in the electrocardiogram.

Conclusions

We describe two versions of an inexpensive and easily constructed platform for synchronized breathing applications. Both designs are based on light-emitting diode (LEDs) under the control of an Arduino UNO microcontroller. In the advanced version, both the respiratory rate and I:E ratio can be easily adjusted by the user, while in the simple version, changing the respiratory rate and I:E ratio parameters requires editing the program.

Abbreviations

ADC: Analog-to-digital converter; JFET: Junction-gate field-effect transistor; LCD: Liquid crystal display; LED: Light-emitting diode; MHz: Megahertz; PWM: Pulse width modulation; USB: Universal serial bus.

Acknowledgements

Not applicable

Author contributions

The author contributed to all aspects of this project, including design, implementation, testing and documentation. The author read and approved the final manuscript.

Funding

This activity was entirely self-funded.

Availability of data and materials

Not applicable.

Code availability

The software code for this project is available from two sources. The first source is from GitHub using the following URL:
https://github.com/djdoylemd/Respiratory-Synchronizer-Project
The second source is from the following box.com download link:
https://app.box.com/s/eiijmjbdw94426av6vzz6r3p59nvl1ul

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The author declares that he has no conflicts of interest, financial or otherwise.

Author details
[1]Department of General Anesthesiology, Cleveland Clinic, Cleveland, USA.
[2]Cleveland Clinic Lerner College of Medicine, Cleveland, OH, USA.

Received: 14 May 2022 Accepted: 6 October 2022
Published online: 15 October 2022

References
Adler TE, Coovadia Y, Cirone D, Khemakhem ML, Usselman CW (2019) Device-guided slow breathing reduces blood pressure and sympathetic activity in young normotensive individuals of both sexes. J Appl Physiol 127(4):1042–1049
Blumenstein B, Breslav I, Bar-Eli M, Tenenbaum G, Weinstein Y (1995) Regulation of mental states and biofeedback techniques: effects on breathing pattern. Biofeedback Self Regul 20(2):169–183
Brenner J, LeBlang S, Lizotte-Waniewski M, Schmidt B, Espinosa PS, DeMets DL et al (2020) Mindfulness with paced breathing reduces blood pressure. Med Hypotheses 142:109780
Cernes R, Zimlichman R (2017) Role of paced breathing for treatment of hypertension. Curr Hypertens Rep 19(6):45
Chen X, Li H (2017) ArControl: an arduino-based comprehensive behavioral platform with real-time performance. Front Behav Neurosci 11:244
Clark ME, Hirschman R (1990) Effects of paced respiration on anxiety reduction in a clinical population. Biofeedback Self Regul 15(3):273–284
Cobo A, Villalba-Mora E, Hayn D, Ferre X, Pérez-Rodríguez R, Sánchez-Sánchez A et al (2020) Portable ultrasound-based device for detecting older adults' sit-to-stand transitions in unsupervised 30-second chair-stand tests. Sensors (basel). 20(7):1975
Das S, Pal S, Mitra M (2017) Arduino-based noise robust online heart-rate detection. J Med Eng Technol 41(3):170–178
Elliott WJ, Izzo JL (2006) Device-guided breathing to lower blood pressure: case report and clinical overview. MedGenMed 8(3):23
Hoang LQ, Chi HBL, Khanh DNN, Vy NTT, Hanh PX, Vu TN et al (2021) Development of a low-cost colorimeter and its application for determination of environmental pollutants. Spectrochim Acta A Mol Biomol Spectrosc 249:119212
Holovatyy A, Teslyuk V, Kryvinska N, Kazarian A (2020) Development of microcontroller-based system for background radiation monitoring. Sensors (basel) 20(24):7322
Huang AJ, Phillips S, Schembri M, Vittinghoff E, Grady D (2015) Device-guided slow-paced respiration for menopausal hot flushes: a randomized controlled trial. Obstet Gynecol 125(5):1130–1138
Huang AJ, Grady D, Mendes WB, Hernandez C, Schembri M, Subak LL (2019) A randomized controlled trial of device guided, slow-paced respiration in women with overactive bladder syndrome. J Urol 202(4):787–794
Laborde S, Allen MS, Göhring N, Dosseville F (2017) The effect of slow-paced breathing on stress management in adolescents with intellectual disability. J Intellect Disabil Res 61(6):560–567
Meule A, Kübler A (2017) A pilot study on the effects of slow paced breathing on current food craving. Appl Psychophysiol Biofeedback 42(1):59–68
Noble DJ, Hochman S (2019) Hypothesis: pulmonary afferent activity patterns during slow, deep breathing contribute to the neural induction of physiological relaxation. Front Physiol 10:1176
Schein MH, Gavish B, Baevsky T, Kaufman M, Levine S, Nessing A et al (2009) Treating hypertension in type II diabetic patients with device-guided breathing: a randomized controlled trial. J Hum Hypertens 23(5):325–331
Sin PYW, Galletly DC, Tzeng YC (2010) Influence of breathing frequency on the pattern of respiratory sinus arrhythmia and blood pressure: old questions revisited. Am J Physiol Heart Circ Physiol 298(5):H1588–H1599
Tsai HJ, Kuo TBJ, Lee G-S, Yang CCH (2015) Efficacy of paced breathing for insomnia: enhances vagal activity and improves sleep quality. Psychophysiology 52(3):388–396
Vallejo W, Diaz-Uribe C, Fajardo C (2020) Do-it-yourself methodology for calorimeter construction based in Arduino data acquisition device for introductory chemical laboratories. Heliyon 6(3):e03591
van Hateren KJJ, Landman GWD, Kleefstra N (2015) Re: "RESPeRATE: the role of paced breathing in hypertension treatment." J Am Soc Hypertens 9(8):656–657
Veldscholte LB, Horst RJ, de Beer S (2021) Design, construction, and testing of an accurate low-cost humidistat for laboratory-scale applications. Eur Phys J E Soft Matter 44(4):48
Viskoper R, Shapira I, Priluck R, Mindlin R, Chornia L, Laszt A et al (2003) Nonpharmacologic treatment of resistant hypertensives by device-guided slow breathing exercises. Am J Hypertens 16(6):484–487
Wandling GD, Lee JI, Talukder MAH, Govindappa PK, Elfar JC (2021) Novel real-time digital pressure sensor reveals wide variations in current nerve crush injury models. Mil Med 186(Suppl 1):473–478
Wilhelm FH, Grossman P, Coyle MA (2004) Improving estimation of cardiac vagal tone during spontaneous breathing using a paced breathing calibration. Biomed Sci Instrum 40:317–324
Zuckerberg J, Shaik M, Widmeier K, Kilbaugh T, Nelin TD (2020) A lung for all: novel mechanical ventilator for emergency and low-resource settings. Life Sci 257:118113

Publisher's Note