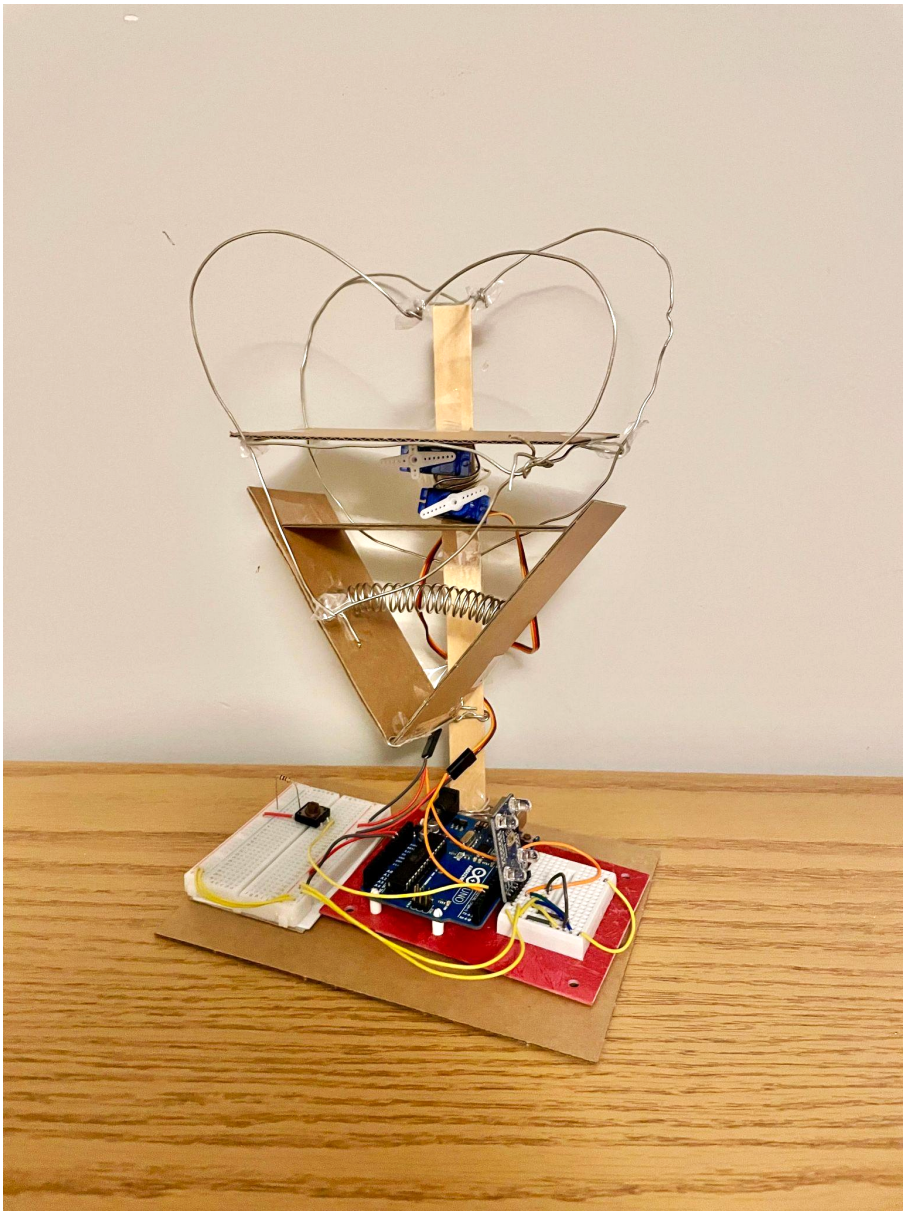
Michelle Zhou, mwz7, MAE 3780 Individual Project Technical Report, Fall 2021
The Breathuino

1. Overview

The Breathuino is a physical visualization of one's heartbeat that expands and contracts at the user's heart rate. The idea is that seeing your heart rate physically manifested in front of you will help you relax and meditate. The user begins the kinetic art piece's movement by interacting with a TCS3200 color sensor and a button. The two positional micro servos are mounted in such a way that they interact with the mechanical structure to turn rotational motion into expansion and linear motions, at a rate dictated by the user input during the calibration process. The device is powered by the Arduino and sits nicely on one's desktop.



2. Design Considerations

My advice to someone attempting to replicate this project, including changes I would make with the same budget and *with a higher budget/more time.*

To replicate this project, one would need to determine the "threshold" period value that corresponds to the heartbeat phase that causes blood to block light through a finger. I would recommend having a dedicated workspace while working on this project in which you can control the lighting, i.e. not next to a window nor outside. The color sensor is very sensitive to lighting conditions, so even the orientation that the Arduino board and sensor are in next to a stationary lamp matters.

If I were to do this project again with the same $20 budget, I would choose a different vendor that sells the complete sensor at a reasonable price of $5.68. I purchased a heartbeat sensor that turned out to have an empty printed circuit board with no components, so with the full sensor, I would still have to do the same smoothing/averaging calculations as with the color sensor, but the data would be more robust and less susceptible to changes in lighting.

If I had more time, I would add the feature to continuously update the angle written to the positional servos such that the user could visualize themselves calming down even more -- since the expansion/contraction of the structure would slow as their heartbeat slows. This may be implemented with a tasker library or interrupts. In terms of mechanical structure, I would make the back post more sturdy with a stronger attachment to the base.

With a larger budget, I would primarily focus on obtaining a more robust sensor, such as the PulseSensor that is more natural to hold one's finger on. The developers of the PulseSensor have also created an Arduino library that directly outputs a beats per minute reading, which would make the code much more concise.

3. Assembly Instructions

Here are my written instructions for how to assemble the mechanical structure of the project, broken up into sections: the bottom chamber, the upper chamber, the base, and the backing. The figures referenced are located in Appendix C: Drawings and Assembly Diagrams.

The Bottom Chamber (see Figure 4)

1. **Create a spring** (see Figure 2) by wrapping the tinned copper wire around a pen 30 times, leaving extra unbent wire at the beginning and end for mounting, as shown in Figure 1.
2. **Cut 2 pieces of 3.5" x 6" x 1/16" cardboard** from 2 pieces of 5" x 7" x 1/16" cardboard.
3. Tape the bottoms of the 2 cardboard pieces together **to form the bottom of the heart.**
4. **Tape the 2 ends of the spring** to the V, 1 on each piece of cardboard (see Figure 3).
5. **Cut 1 piece of 3" x 5" x 1/16" cardboard** from another piece of stock cardboard.

6. **Tape this smaller piece to *one* of the side panel**s of the V configuration about 1” from the top, measured along the diagonal panel.

The Upper Chamber (see Figure 4)

1. Using the bottom V shape assembly as reference for size, **bend 1 continuous piece of wire into a heart shape,** using pliers to make the sharp corner at the crest.
2. **Repeat** to end up with 2 planar wire hearts.
3. **Fold a piece of 8” wire in half** and loop the 2 planar hearts together, using pliers to press the 2 halves of the wire firmly together.
4. **Twist the ends of the wire** together to secure the connection of the 2 planar hearts
5. Using pliers again, **fold a rectangle** that fits around the top third of the heart shape.
6. **Secure as needed** with tape.

The Base

1. Tape the bottom of the breadboard and Arduino to a piece of cardboard.

The Backing

1. Tape 2 tongue depressors together with 1” overlap in the middle **to create the post.**
2. Cut off the 2 ends of the post with scissors **such that the ends are flat.**
3. **Tape the post to the bottom piece of cardboard,** placed behind the Arduino and breadboard, using a few loops of wire wrapped around the sticks for more surface area for the tape to hold onto if necessary.
4. **To secure the cardboard configuration to the post, bend a wire in half** around the tip/crevice of the V-shaped bottom chamber, with room at the wire ends to then wrap around the post. Tape the wire to the cardboard along the outside of the V-shaped chamber.
5. Also **tape the wire that’s looped around the back post,** adding tape as needed between the post and the cardboard V itself.
6. Testing how close the servo needs to be to the top flap of the bottom chamber, **tape the bottom servo to the post.**
7. Wrap a piece of wire around the post to act as a **spacer between the two servos.**
8. **Tape the top servo** to the post.
9. Balance/position the heart skeleton on top of the top servo, **taping down the V-shape bottom of the *wire* heart skeleton to the post.**

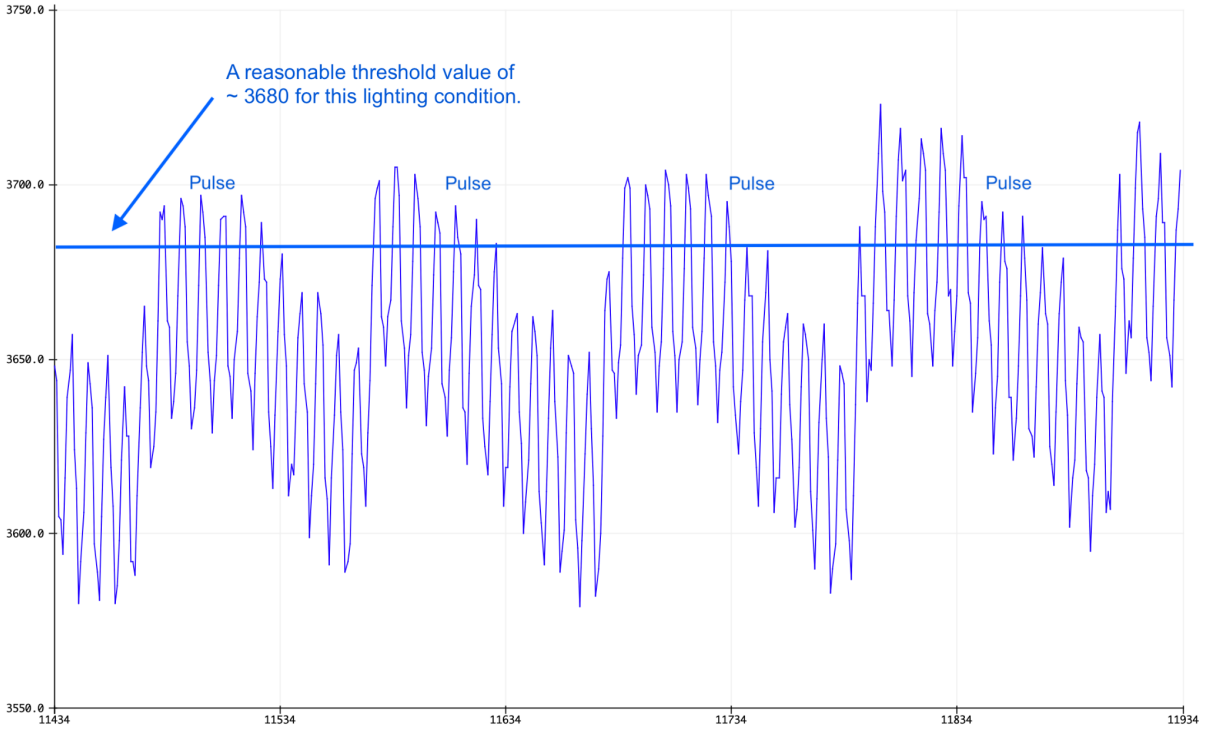
4. Operation Instructions

Upon being fully assembled, the Breathuino will require varying levels of calibration depending on if it is the first time you are using it in a certain environment.

1. Be in a well-lit, controlled environment.
2. Plug the Arudino into a computer for power via the USB cable.
3. If this is the first time in a new environment, upload the “color_sensor.ino” code to identify the threshold value for a heartbeat pulse via Serial Plotter. Change this value in

the “// sensor-related variables” section of global variable declarations in line 24 of “Michelle_Zhou_Mechatronics_Individual_Project.ino”

a. For example, the figure below shows how to choose a good threshold value for a given lighting condition. The waveform shown is the output from the color sensor using “pulseIn(colorPin, HIGH)” whose values are periods in milliseconds that correspond to detected color frequencies. There is a noise factor in the code to account for multiple crossings of the threshold due to noise.



4. Otherwise/afterward, upload “Michelle_Zhou_Mechatronics_Individual_Project.ino” to the Arduino and open the Serial Monitor.
5. Wait for the Serial Monitor to print “Begin calibration mode by pressing the button and holding your finger to the sensor for 10 seconds!”
6. Press the button once you are ready to enjoy the project and hold your finger still to the color sensor for 10 seconds.
7. Now the project is in READY MODE, and pressing the button will enable the structure to expand and contract at your heart rate.
8. To recalibrate with a new heart rate, unplug the Arduino and begin again.

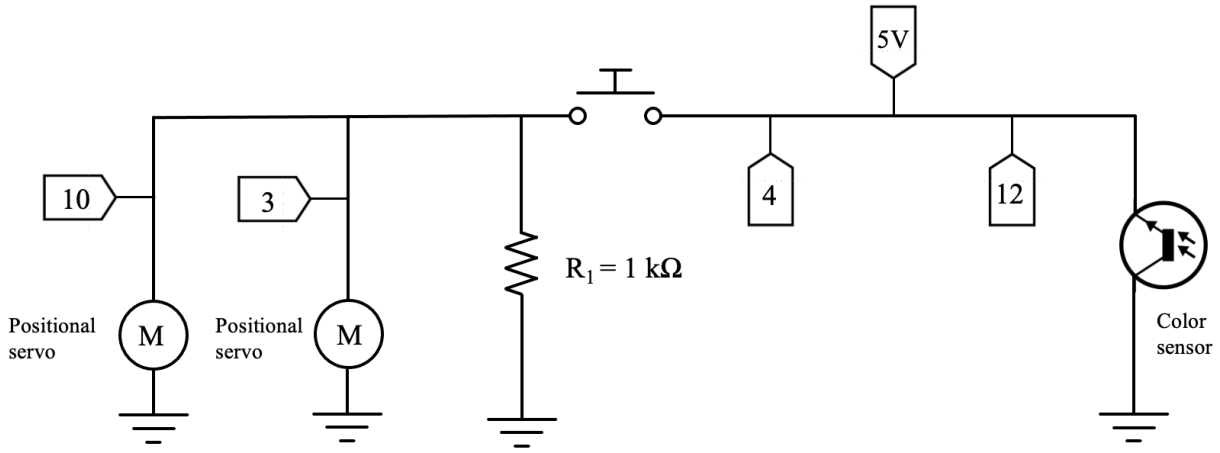
5. Appendix A: Bill of Materials

This is a table listing all parts and materials used in this project, including purchased, scavenged, and kit items. For each, the vendor/source, part name, quantity, and unit/subtotal prices are listed (some may include part numbers). There are 2 totals listed: 1 for building the project from scratch, and 1 for the total cost of purchased/scavenged parts.

Purchased Items	Part Number / ASIN	From where	Unit Price	Quantity	Units	Total Cost
5" x 7" x 1/16" Cardboard		In-Stock Craft and Circuit Supplies	$0.11	3	count	$0.33
Tongue Depressor		In-Stock Craft and Circuit Supplies	$0.03	2	count	$0.06
Belden wire and cable, 18 awg, 8013, 100ft, bus bar wire, solid tin cu		Dave Hartino/Digikey	$0.50	5	ft	$2.50
Clear Tape		Home	$2.64	0.25	count	$0.66
					TOTAL $20 BUDGET COST:	$3.55
Kit Items	**Part Number / ASIN**	**From where**	**Unit Price**	**Quantity**	**Units**	**Total Cost**
Bread Board	79X3922	Newark	$2.71	1	count	$2.71
Mini Breadboard	98AC7296	Newark	$1.05	1	count	$1.05
Tactile Switch Push Button	155380	Jameco	$0.35	1	count	$0.35
Wire Kit	ASIN : B07PQKNQ22	Amazon: Austor	$2.17	1	count	$2.17
Resistor 1k Ω	1.0kQBK-ND	Digi-Key	$0.01	1	count	$0.01
Arduino Board	1050-1024-ND	Digi-Key	$20.90	1	count	$20.90
Color Sensor	RB-Wav-10	RobotShop.com	$6.27	1	count	$6.27
Micro Servo positional	SER0006	Dfrobot	$3.30	1	count	$3.30
					TOTAL PROJECT COST:	$40.31

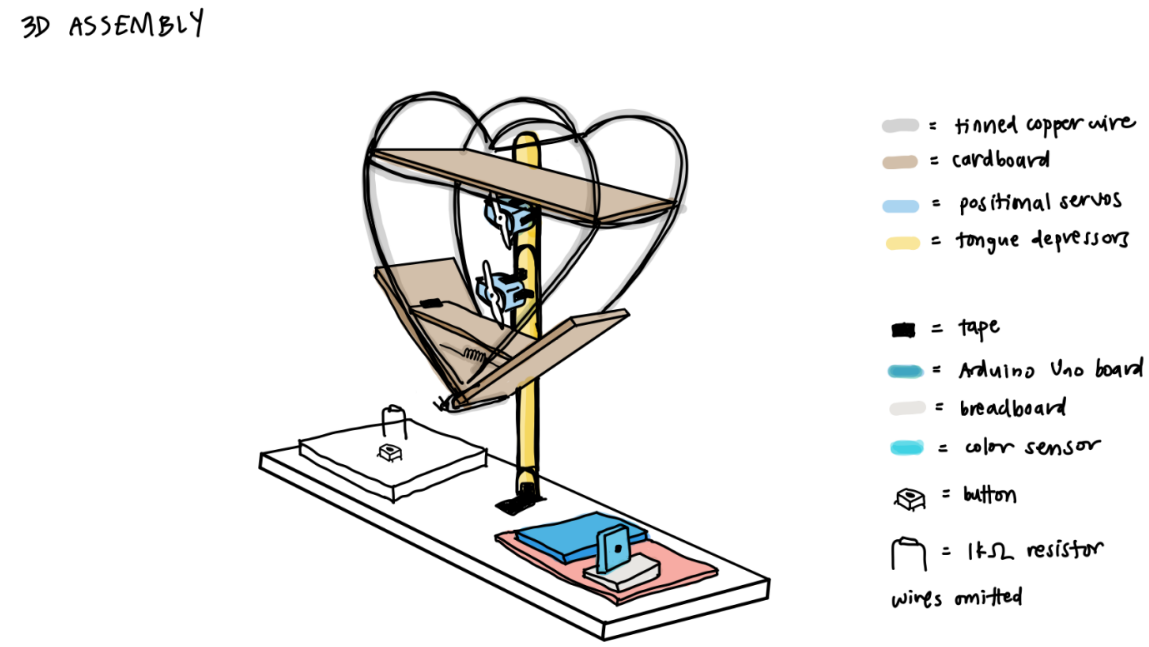
6. Appendix B: Circuit Diagram

Here is the circuit diagram for the project.



7. Appendix C: Drawings and Assembly Diagrams

The following are visual aids to help understand the mechanical structure and assembly process of the project. For full written instructions, refer to the Assembly Instructions section.



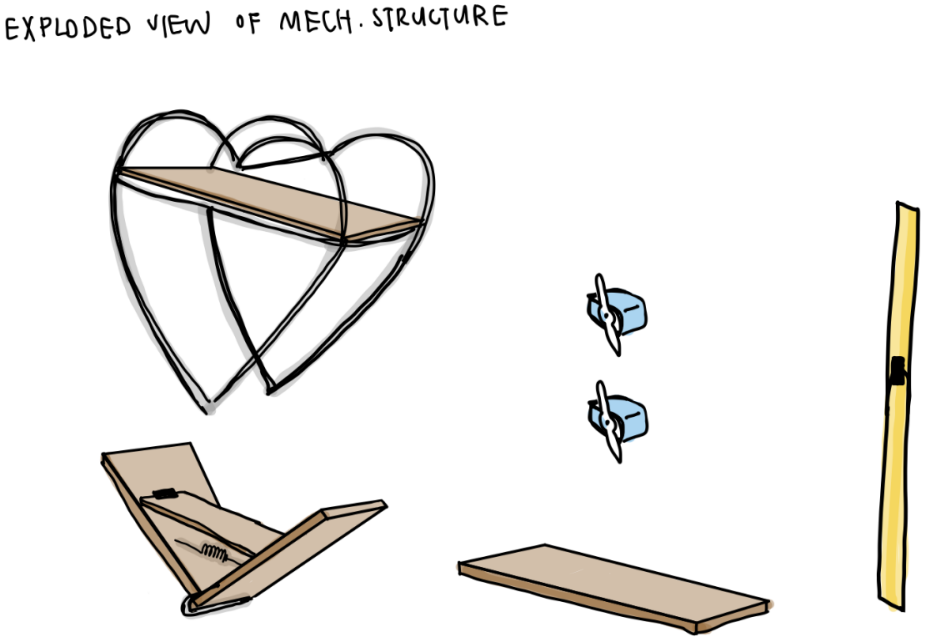


Figure 0: A complete drawing and exploded view drawing of the mechanical assembly.

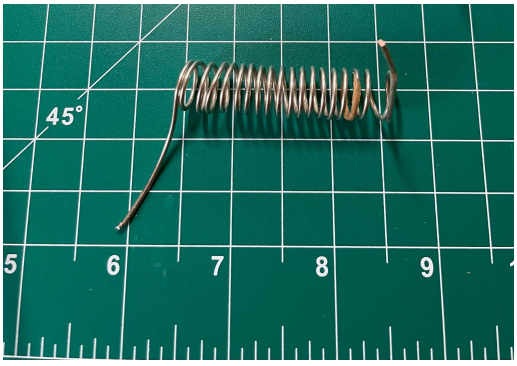


Figure 1 (left): A demonstration of how to coil wire around a cylindrical object to make a spring.
Figure 2 (right): The spring I made before I stretched it out to better fit the V shape better.

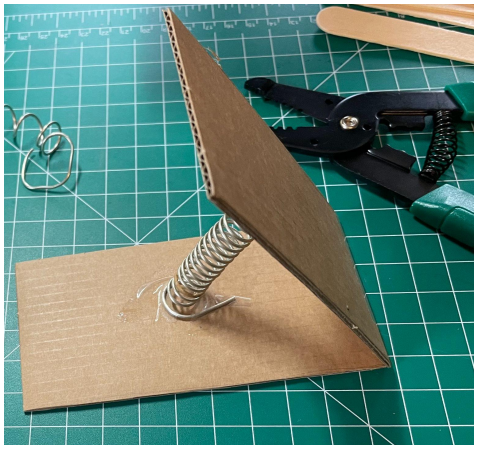


Figure 3: A close-up shot of how to mount the spring between the side panels.

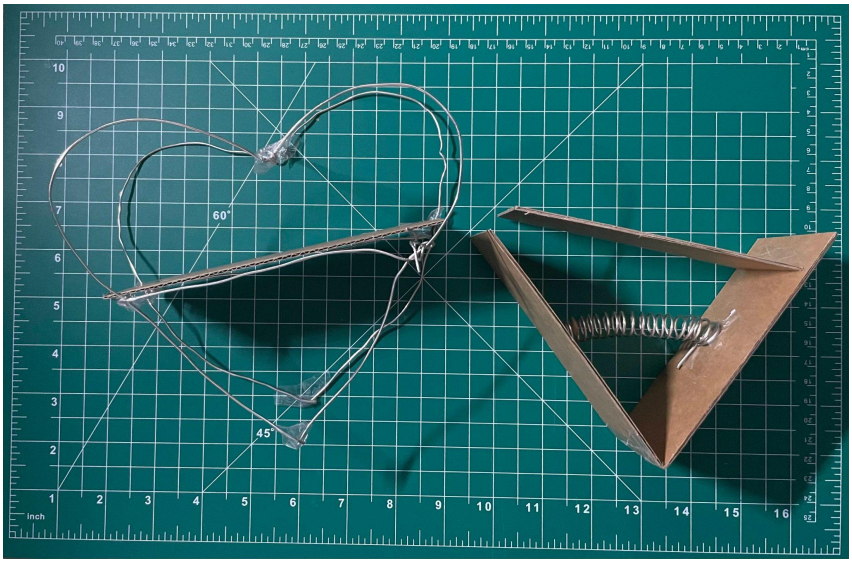


Figure 4: The top chamber (left) and bottom chamber (right) unassembled with measurements in inches.

8. Appendix D: Commented Arduino code

Here is the commented code for the project:
"Michelle_Zhou_Mechatronics_Individual_Project.ino".

```
/**
MAE 3780 Individual Project, Fall 2021
Michelle Zhou mwz7, Mechatronics Friday Lab

This code controls 2 input signals: a button and a color sensor, and
2 output signals: 2 positional servos. While in calibration mode, the code reads
in period values in ms that correspond to color wave frequencies from the color
sensor. It then calculates an appropriate angle range for the positional servos and
writes them to the servo.

Note: the color sensor + bright ambient light is meant to mimic a heartbeat.
**/

#include <Servo.h>    // load servo library

Servo myservo1;       // create positional servo object
Servo myservo2;       // create positional servo object 2

// servo-related variables
int angle = 0;           // set default beginning angle
float angleRange = 0.0;   // declare variable for positional servo angle range
const int colorPin = 12;  // pin number for color sensor input

// sensor-related variables
float threshold = 3520.0; // "red" threshold determined from Serial Plotter experiments in a
consistent lighting condition
float sensorVal;         // the color frequency's corresponding period read in by pulseIn()
float pulse = 0.0;       // the number of pulses in the sample interval
float bpm;               // beats per minute = 6*(# of pulses in a 10 second interval)/(noise
calibration factor for other signals that go over the threshold)

bool readyMode = false;      // boolean to answer the question: is this the initialization stage?

// this code is performed once
void setup() {

myservo1.attach(3);       // tell the Arduino pin 3 drives the positional servo 1
myservo2.attach(10);      // tell the Arduino pin 10 drives the positional servo 2
```

```
Serial.begin (9600);        // serial output initialization
pinMode(colorPin, INPUT);    // set color sensor pin to INPUT
pinMode(3, OUTPUT);         // set positional servo 1 pin to OUTPUT
pinMode(10, OUTPUT);         // set positional servo 2 pin to OUTPUT
}

// this code runs continuously
void loop() {

 // INITIALIZATION PERIOD
if (readyMode == false){
        delay(5000);  // wait for 5 seconds to allow user to get oriented

        // tell user that they may begin calibration mode
        Serial.println("Begin calibration mode by pressing the button and holding your finger to
        the sensor for 10 seconds!");

         // CALIBRATION MODE
        while (digitalRead(4) == HIGH){         // when button is pressed

                sensorVal = pulseIn(colorPin, HIGH);    // store sensor input value, which is the
                period that corresponds to a certain color's wave frequency

                Serial.println("Reading~");           // let user know that the sensor is working and
                reading their pulse

                 if (sensorVal > threshold)           // compare sensor value to "red" threshold
                {
                        pulse++;         // count the # of pulses
                        Serial.println("Pulse"); // for testing purposes
                }

                readyMode = true;   // tell the program that the calibration mode is over and will
                never be begin again until device is unplugged
        }
}

// READY MODE
else{
        bpm = pulse*6.0/30.0;
        // ^ extrapolating bpm from 10 s of data
         //   divide by 30.0 to account for the noise from the sensor
         //   multiply by 6.0 to get beats per 60 s
```

```
        //   decimals because we want float math, not int math

        angleRange = 4000.0/bpm;
        // ^ translating bpm to a sensible corresponding angle range st when bpm increases,
        angle range decreases,
        // i.e. servos move back and forth more often

        angleRange = (int) angleRange;
        // ^ cast angleRange to an int for comparison without turning angle into a float since
        .write() needs to be an int

        Serial.println(bpm); // to see bpm in the Serial Monitor
        Serial.println(angleRange); // to see angleRange in the Serial Monitor

        // while button is pressed and angle can still increase,
        // move the servos in one direction
        while(angle < angleRange && digitalRead(4) == HIGH){

                angle++;                    // increment angle
                myservo1.write(angle);      // move positional servo to new angle
                myservo2.write(angle);      // move positional servo to new angle
                delay(10);                  // give Arduino 0.010 s to process
        }

        angleRange = 0.8*angleRange;     // to mimic the sound of a real life heartbeat

        // while button is pressed and angle has reached max angle range,
        // move the servos in the other direction

        while(angle > 0 && digitalRead(4) == HIGH){
                angle--;                    // decrement angle
                myservo1.write(angle);      // move positional servo to new angle
                myservo2.write(angle);      // move positional servo to new angle
                delay(10);                  // give Arduino 0.010 s to process
        }
}

}
```

Here is a copy of supplemental code, from TA Ethan Liu's code on Canvas for the color sensor, useful for calibrating the color sensor for heartbeat detection in a new environment.

```
// Color Sensor Example Code
// This code prints half the period of the signal sensed (pin7)
//
// MAE 3780 - Mechatronics
// Ethan Liu
// Last edited: 9/29/2021
// pin number of color sensor

const int colorPin = 12;
// declare variable to keep track of duration of wave
long duration;

void setup() {

 Serial.begin(9600); // initialize serial communication
 pinMode(colorPin, INPUT); // set color sensor pin to input

}

void loop() {

 duration = pulseIn(colorPin, HIGH);
 // for pin starting at HIGH to go to LOW

 Serial.println(duration); // print duration to serial
 // monitor every 100ms
}
```