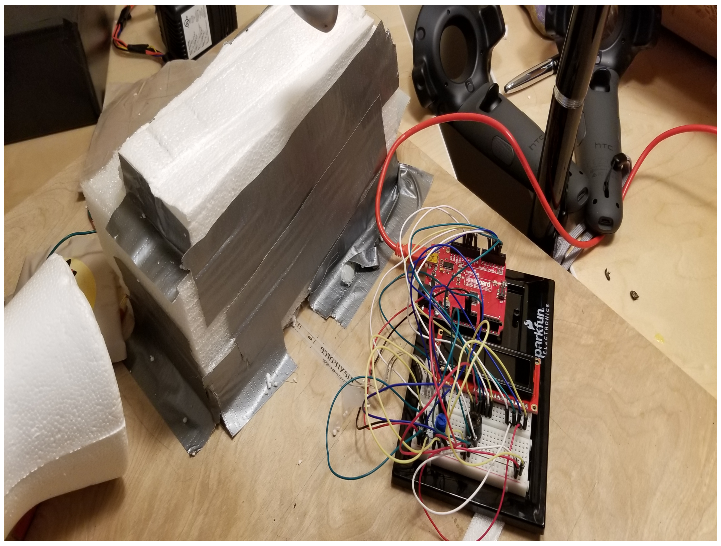
CPR Simulator

CPR Saves

Learn CPR. Save a life.



Do it yourself guide

This document takes you step by step on how to make your own low cost high fidelity CPR simulator

Table of Content

Developed by Mrinali Kesavadas

For more information about CPR visit http://cprsaves.org

Contact: info@cprsaves.org

Edited on January 6, 2019

CPR Saves

Learn CPR. Save a life.

Making the CPR Dummy

INTRODUCTION

Ives CPR simulator, consists of a box with number of thick rubber bands on the top. The rubber bands are attached to a plastic frame. The system has a Styrofoam surface for the pushing and the recoiling of the chest is achieved by rubber bands. The Styrofoam is the only low cost consumable in the system and must be replaced based on the usage.

The electronic heart of the simulator is based on an Arduino UNO microcontroller. The Arduino allows interfacing and control of several sensors and can be programed using its own programing language. To measure the force involved in CPR, a Flexi Force sensor is used. A piezo buzzer is used to provide the rate of compressions at 120 beeps/minute. The force sensor is placed at a depth of 5 cm. The software to control the simulation is in Arduino programing language and can be downloaded to the Arduino microcontroller. Once the download is complete no computer is needed. The portable simulator is powered by 4 1.5 Volt batteries. A solar cell may be used in place of the battery for the use of the simulator in rural areas.

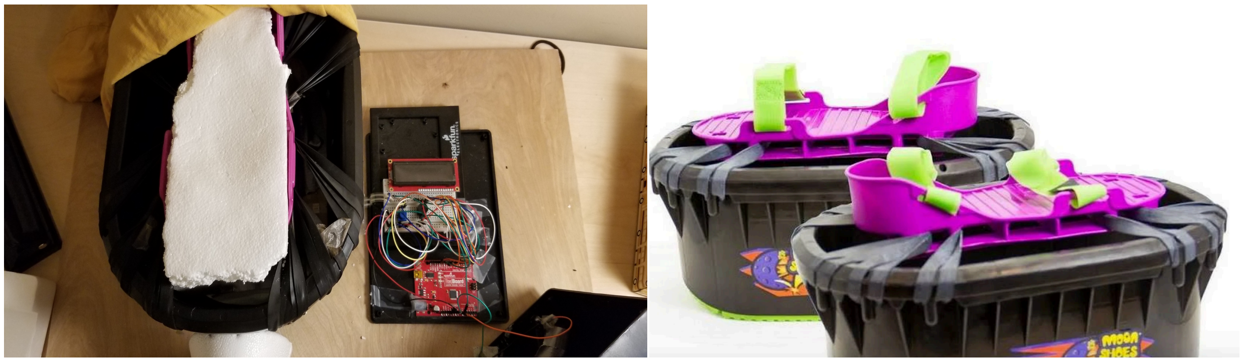


Figure 1 (left) Basic framework of the CPR Simulator (Right) Moon Shoes

Physical simulator using rubber bands and block of foam

When CPR performed, the simulator must essentially help in learning two skills – pushing chest down (compressing) followed by a recoil of the ribs followed (process is repeated). To obtain this simulation, a product called Moon Shoes (Figure 1) is used. Under the Moon shoe, a block of foam is kept (Figure 2). When pressure is applied on the top of the shoe, the rubber bands provide the resistance and recoil like the chest of an adult. Figure 3 shows the CPR compression procedure on the simulator and Figure 4 shows the logic and display.

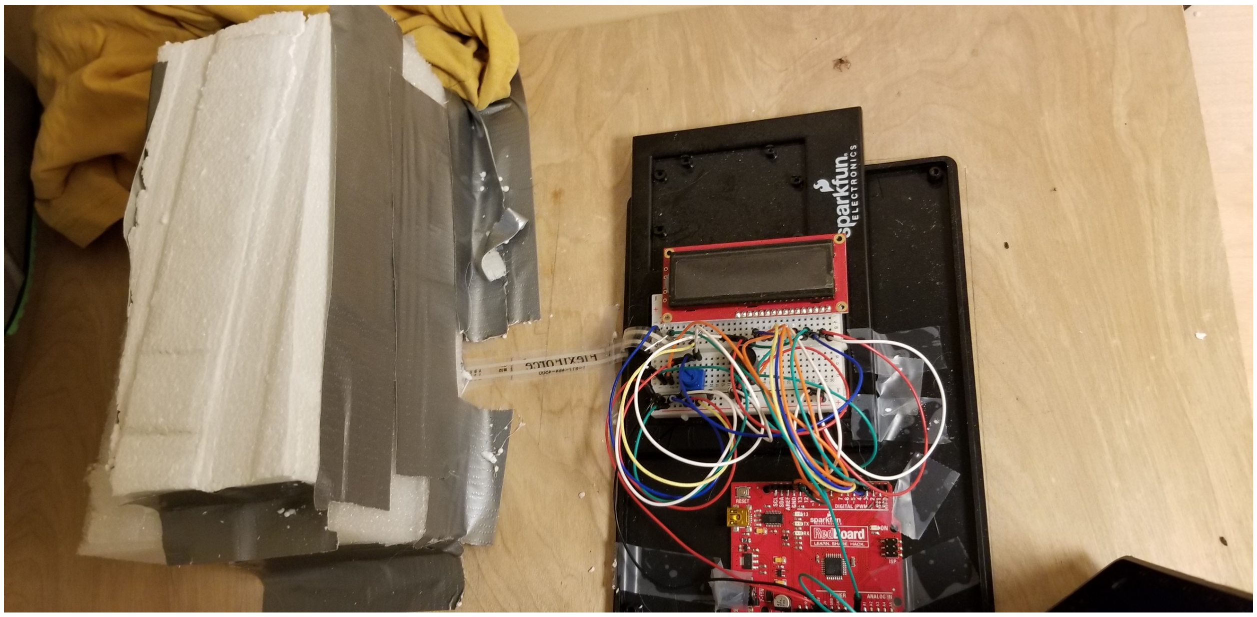


Figure 2 Inside the Moon Shoes – a block of foam 8”x5”x5”. Under the foam is a flexi force sensor which is connected to the circuit

Placement of sensors

To measure the actual pressure applied and to count the number of valid compression cycles, we place a sensor at the bottom of the foam. A Flexiforce sensor (Figure 2) is used to measure these two events. The pressure sensor data is measured using an Arduino Uno as is discussed later in the next section.

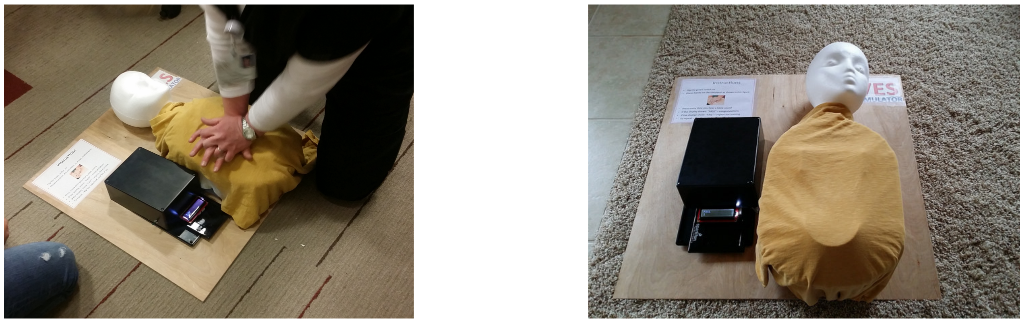
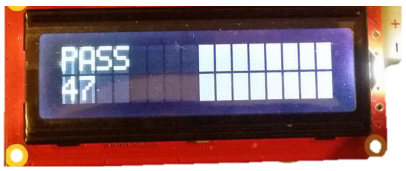


Figure 3 Shows the simulator being used to practice CPR. The left figure shows the proper way of performing CPR and the right the CPR simulator is shown covered with a cloth.



If simulator is pushed 45 times or more successfully in 30 Seconds – **PASS** or else **FAIL**
Push is a success only when **force, depth and rate** meet the requirement (as per AHA guidelines)

Figure 4 LCD display and the logic for successful completion of training

Developing the Electronics

Electronics for the CPR

The electronic and hardware is used to perform four main functions:

1. It provides 120 beeps/minute – matching with the required CPR rate recommended by AHA
2. It measures the force with which the compression is applied during the period when the beeping sound is generated
3. It counts every compressive action if the force applied is above 30lb force
4. At the end of 30 seconds it calculates the number of valid compressions and displays if the CPR was a “Fail” or a “Pass”. At least 45 good compressions must be performed during a 60 second period to get a “Pass” (figure 4)

The circuit is developed using an Arduino Uno or Arduino RedBoard. Table 1 provides the items required to develop the simulator. All these items can be purchased from sparkfun.com

Table 1: Part list for the simulator

ITEMS
Arduino Redboard
Half Breadboard
10 K Ohms Resistors (2)
Piezo Buzzer - Mini Speaker - PC Mount 12mm 2.048kHz
Potentiometer - Trimpot 10K with Knob from SparksFun
LCD display - SparkFun 16x2 SerLCD
FlexiForce Pressure Sensor 100 Lbs
4 AAA 1.6 V Batteries and a battery holder
Moon Shoes
Styrofoam
Foam head
Wires of various colors
Plastic box for the circuit

Circuit Diagram

Circuit diagram

In the table below – the wiring diagram for the entire circuit is shown. It uses a standard half breadboard and the items that were mentioned in Table 1. Figure 5 shows the completed diagram.

Table 2: Circuit connection

Items	Wire	Connection	Connection	Pins
	Red	5V	Positive	
	Black	Ground	Negative	
	Green	a2	Positive	
	Blue	g1	Negative	
	Yello	b1	Negative	
	Red	e6	positive	
	Yellow	f7	Positive	
	Blue	e8	Negative	
	Yellow	f15	Negative	?
	Red	f16	Positive	
	Green	a15	Negative	?
	Red	f29	Positive	
	White	f26	Negative	
	Yellow	f30	Negative	
	White	a3	PIN 9	
	White	a17	PIN 13	
Potentiometer				a6, a7, a8
Piezo Buzzer				c15, c17
	Blue	e7	f28	
	Green	PIN 2	f17	
	Blue	Pin3	f18	
	Whie	Pin4	f19	
	Yellow	Pin 5	f20	
	White	Pin 11	f25	
	Green	Pin 12	f27	
Resistor 10KOhm		g4	g8	
Resistor 10KOhm		g3	h7	
	Green	A0	h3	
LCD Display				j15 to j30

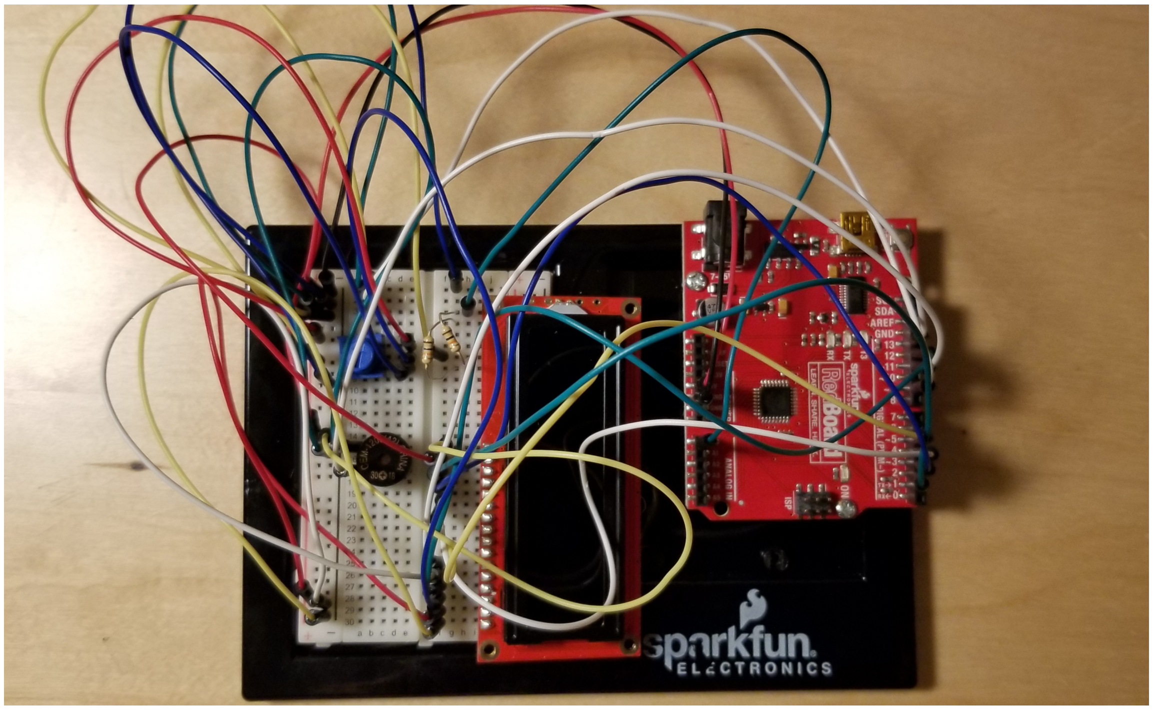
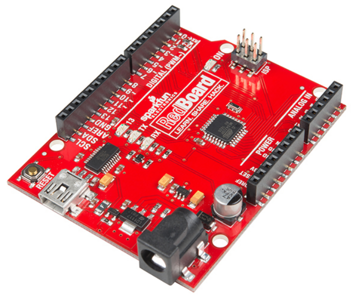
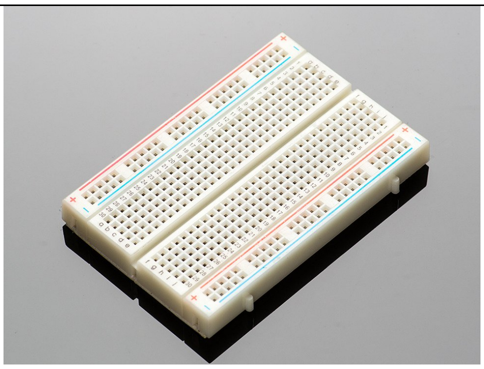
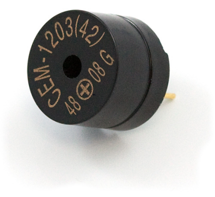
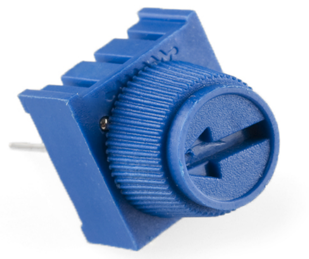


Figure 5: Completed circuit is shown based on the wiring logic shown in Table 2

Part Pictures

 Arduino RedBoard	 Breadboard
LCD Display	 Piezo Buzzer - Mini Speaker - PC Mount 12mm 2.048kHz
Flexi Force Sensor 100 Lbs	 Potentiometer - Trimpot 10K with Knob from SparksFun

Coding

Developing the code

The code for the software is shown in table 3 below. The coding is in Arduino programing language. You can also download the code here: **https://drive.google.com/open?id=1irb89C152FW8eeLi6242nppmXfoL-lL3**

Table 3: Code for the simulator (ver: 1/3/19)

```
/*
CPR Sample Code - Mrinali Kesavadas
For more information - contact info@cprsaves.org
I used this code to develop a low cost CPR system that was placed in top ten for the Discovery 3M
award in 2015/2016
See the full demo of the CPR simulator here:
https://www.youtube.com/watch?v=5w9C-fs-XWE

Essentially the code does the following:

When you switch on the device - it starts beeping at 120 beeps/minute for 30 seconds.
The Flexiforce sensor measures if proper force has been applied
For every proper push a counter is updated
At the end of 30 seconds if there is atleast 45 proper pushes, the LCD display shows PASS
or else FAIL is displayed.
The flexi force sensor has been calibrated to expect at least 30lbs of force to register a proper
push.

Hardware connections:

  Flex sensor:

    The flex sensor is the plastic strip with black stripes.
    It senses bending away from the striped side.

    The flex sensor has two pins, and since it's a resistor,
    the pins are interchangable.

    Connect one of the pins to ANALOG IN pin 0 on the Arduino.
    Connect the same pin, through a 10K Ohm resistor (brown
   black orange) to GND.
   Connect the other pin to 5V.

    http://www.arduino.cc/cgi-bin/yabb2/YaBB.pl?num=1239464763

This sketch is a combination of my code along with many codes from SparkFun Electronics, with
lots of help from the Arduino community.
This code is completely free for any use.
Visit http://learn.sparkfun.com/products/2 for SIK information.
Visit http://www.arduino.cc to learn about the Arduino.

Version 3.0 1/3/19
*/



#include <LiquidCrystal.h>

LiquidCrystal lcd(12,11,5,4,3,2);



// Create Beep Sound for CPR at the rate recommended by American Heart Association

const int buzzerPin = 13;
const int songLength = 100;
char notes[] = "c c c c c c c c c c c c c c c c c c c c c c c c c c c c c c c c c c c c c c c c c c c c c c c c
c c ";
int beats[] =
{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1};
int tempo = 280;
const int flexpin = 0;

int count = 1;

void setup()
{
  // Use the serial monitor window to help debug our sketch:

  Serial.begin(9600);

  lcd.begin(16, 2);

  lcd.clear();

 pinMode(buzzerPin, OUTPUT);
}

void loop()
{

void loop2();
```

```
{
  int i, duration;

  for (i = 0; i < songLength; i++) // step through the song arrays
  {
    duration = beats[i] * tempo;  // length of note/rest in ms

    if (notes[i] == ' ')          // is this a rest?
    {
      delay(duration);            // then pause for a moment
    }
    else                          // otherwise, play the note
    {
      tone(buzzerPin,400, duration);
      delay(duration);            // wait for tone to finish
    }
    delay(tempo/10);
    void loop1();
 {

  int flexposition;    // Input value from the analog pin.

  // Read the position of the flex sensor (0 to 1023):

  flexposition = analogRead(flexpin);

int analogValue = analogRead(flexpin);

  // if the analog value is high enough, turn on the LED:


  if (flexposition < 1000) {

    count = count+1;
  }

Serial.println(count);

  lcd.setCursor(0,1);

  lcd.print(count);

// if (count > 100) {
 // lcd.setCursor(0,0);
//  lcd.print("PASS");}
// Send message to BioGears here showing that CPR was successful
if (0 < count )
{ lcd.setCursor(0,0);
  lcd.print("PUSH");}
//Send message to BioGears here is CPR is not done properly
if ((millis()>5000) && (count > 44))
{ lcd.setCursor(0,0);
  lcd.print("PASS");}

  else if ((millis()>30000) && (count < 45))

 {lcd.setCursor(0,0);
  lcd.print("FAIL");}

  //lcd.print("FAIL");}
  //Serial.print("sensor: ");
  //Serial.print(flexposition);

}// brief pause between notes
  }

  // We only want to play the CPR tones once, so we'll pause forever:
while(true){}
  // If you'd like your beeps to play over and over,
  // remove the above statement

}
  // Note that all of the above lines are "print" except for the
  // last line which is "println". This puts everything on the
  // same line, then sends a final carriage return to move to
  // the next line.

  // After you upload the sketch, turn on the serial monitor
  // (the magnifying-glass icon to the right of the icon bar).
  // You'll be able to see the sensor values. Bend the flex sensor
  // and note its minimum and maximum values. If you replace the
  // 600 and 900 in the map() function above, you'll exactly match
  // the flex sensor's range with the servo's range.


}
```

Watch a video of the simulator being used

https://www.youtube.com/watch?v=5w9C-fs-XWE

Thank you!

CPRsaves.org