Supplementary Material (ESI) for Lab on a Chip
This journal is © The Royal Society of Chemistry 2019

Open Source Acoustofluidics

Hunter Bachman,[a] Hai Fu,[a,b] Po-Hsun Huang,[a] Zhenhua Tian,[a] Jonah Embry-Seckler,[a] Joseph Rufo,[a] Zhemiao Xie,[a] Jessica H. Hartman,[c] Shuaiguo Zhao,[a] Shujie Yang,[a] Joel N. Meyer,[c] and Tony Jun Huang[a]

[a] *Department of Mechanical Engineering and Material Science, Duke University, Durham NC, 27708 USA. Tel: 919-684-5728; E-mail: tony.huang@duke.edu*

[b] *Department of Fluid Control and Automation, Harbin Institute of Technology, Harbin, Heilongjiang 150001, China*

[c] *Nicholas School of the Environment, Duke University, Durham, NC 27708, USA*

1. Arduino Control System Code

This code functions with the Arduino Uno system. The code provides control for an LCD screen, two buttons to vary frequency (Pin 4 and 5), and a button to turn the signal ON/OFF (Pin 3). An LED is connected to pin 2, and the control pin connected to the motor driver is pin 6.

```
// include the library code:
#include <LiquidCrystal.h>
// initialize the library by associating any needed LCD interface pin with the Arduino pin number it is connected to

const int rs = 12, en = 11, d4 = 10, d5 = 9, d6 = 8, d7 = 7;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
int powerButton = 3;
int LED = 2;

// initialize all variables associated with the function generator control and output
int powerButtonState = 0;
int powerButtonCount = 0;
int freqUpButton = 5;
int freqUpButtonState = 0;
int freqDownButton = 4;
int freqDownButtonState = 0;
int frequency = 5000;
int deltFreq = 100;
int powerPin = 6;
String string2 = " Hz";
String string1 = "";
```

```
// setup loop for one-time actions
void setup() {
  // set up the LCD's number of columns and rows:
 pinMode(LED,OUTPUT);
 pinMode(powerButton,INPUT);
 pinMode(freqUpButton,INPUT);
 pinMode(freqDownButton,INPUT);
 pinMode(powerPin,OUTPUT);
  lcd.begin(16, 2);
  // Print a message to the LCD.
  lcd.print("Frequency: ");
}

// running loop with repeated functions
void loop() {
  // set the cursor to column 0, line 1
  // (note: line 1 is the second row, since counting begins with 0):
  lcd.setCursor(0, 1);
  // print the number of seconds since reset:
  powerButtonState = digitalRead(powerButton);
  if (powerButtonState == 1){
    delay(50);
     powerButtonState = digitalRead(powerButton);
     if (powerButtonState == 0){
      powerButtonCount = powerButtonCount + 1;
      }
    }
   freqUpButtonState = digitalRead(freqUpButton);
   if (freqUpButtonState == 1){
    delay(50);
     freqUpButtonState = digitalRead(freqUpButton);
     if (freqUpButtonState == 0){
      frequency = frequency + deltFreq;
      }
    }
   freqDownButtonState = digitalRead(freqDownButton);
   if (freqDownButtonState == 1){
    delay(50);
     freqDownButtonState = digitalRead(freqDownButton);
     if (freqDownButtonState == 0){
      frequency = frequency - deltFreq;
      }
    }
    if ((powerButtonCount % 2 != 0)){
```

```
    digitalWrite(LED,HIGH);
    tone(powerPin,frequency);
   }
    else{
    digitalWrite(LED,LOW);
    noTone(powerPin);
    }
   string1 = frequency + string2;
   lcd.print(string1);
}
```

2. Supplementary Figures

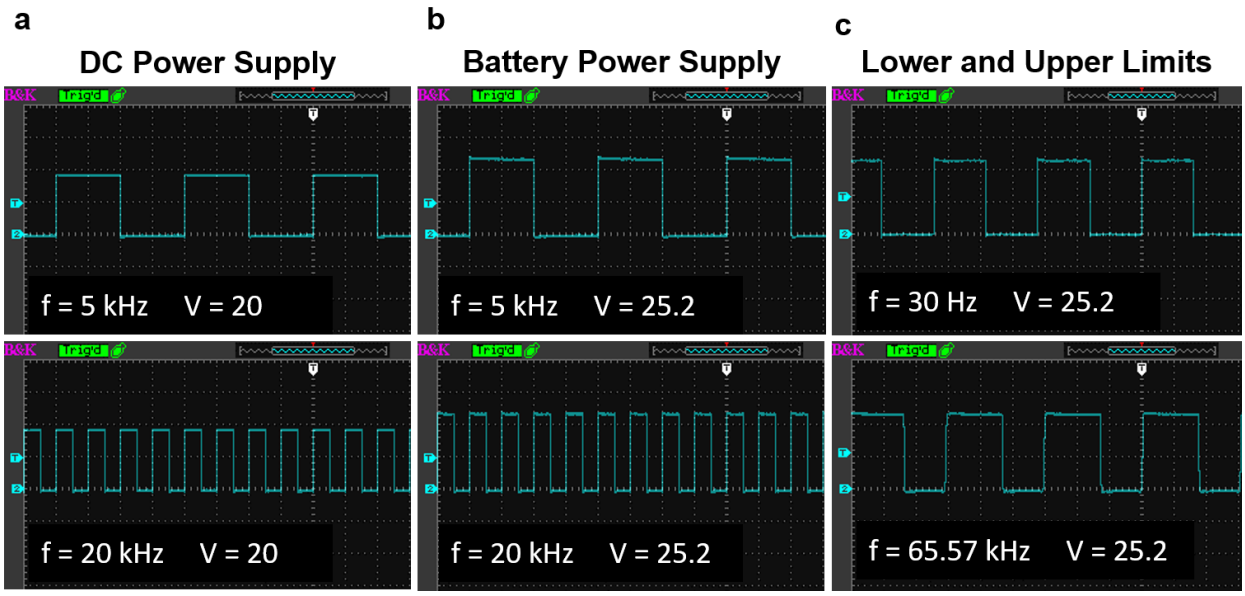

Figure S1 Oscilloscope images of the signal produces by the Arduino system supplied with either a high voltage outlet connected DC power supply (a), or a high voltage battery system (b). c) Oscilloscope images of the signal produced by the Arduino system at the lower (30 Hz) and upper (65.57 kHz) limits. The images for the frequency generation limits were obtained when using a battery power supply, and beyond these limits the signal was inconsistent.

The oscilloscope images provided in Fig. S1 serve to demonstrate the signal generation capabilities of the Arduino based system. It can be seen that whether the system is supplied with a high voltage signal from an outlet based DC power supply or a battery pack, the signals remain very consistent and uniform. The images provided in Fig. S1c demonstrate the performance at the limits of the Arduino code. Referencing the code provided in Section 1 above, we can see that the signal generation comes from the built in "tone" function, which inherently has these limits. If one desired, they could write custom code to circumvent these limits and achieve even lower or higher frequencies. However, the limits of functionality when considering the whole system, including the motor driver, will have its own limits. The published maximum logic speed for the motor driver chip is 40 kHz, so this signal is already pushing past that limit. In fact, we can see that the 65.57 kHz signal produced by the system is not as consistent as the other frequencies; the signal seems to have a duty cycle that is slightly more than 0.5, indicating that there is most likely some problem with the switching in the motor driver.

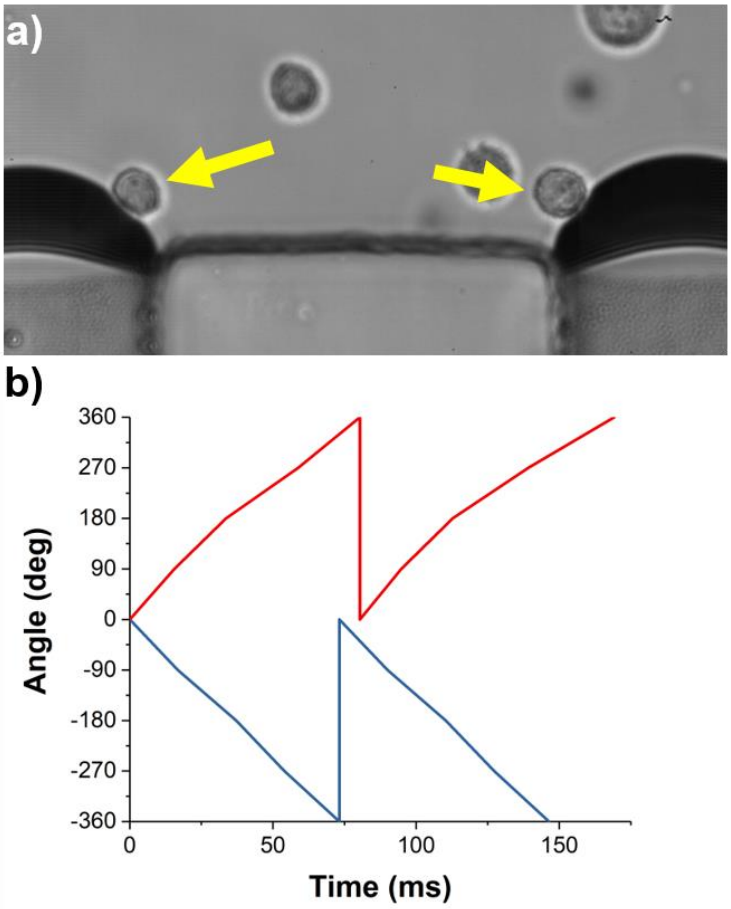

Figure S2 a) photo of two cells being rotated on adjacent bubbles. Additional cells in the photo have adhered to the surface of the device. b) Plot showing the rotation angle versus time for the two cells. The rotation angle is plotted positive for CW (left cell) and negative for CCW (right cell). There is a shift of about 22 ms between cell rotation angles after the second rotation.

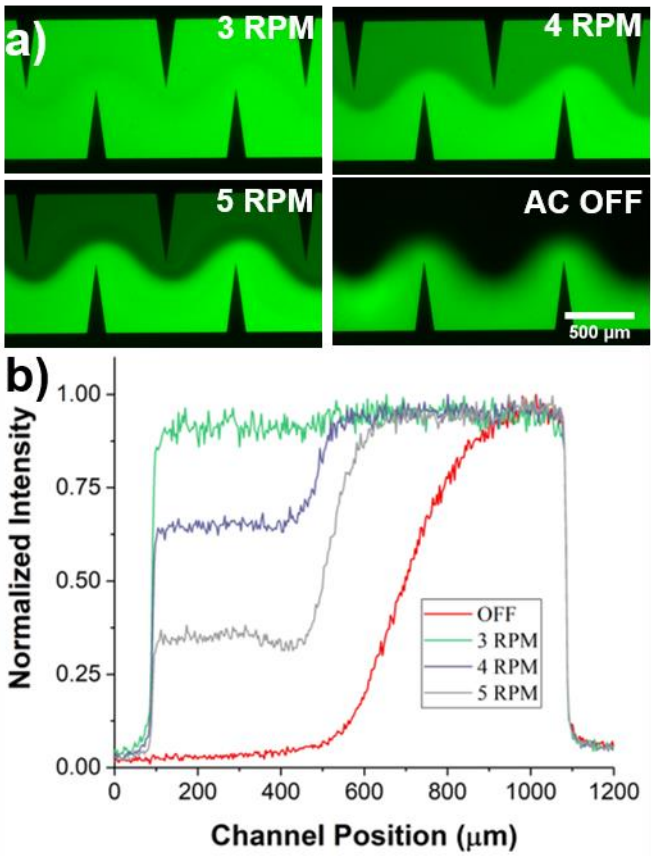

Figure S3 a) Images of mixing performance at different flow rates in the sputum liquefaction chip: 3, 4, and 5 RPM correspond to total flow rates of 79, 109, and 138 µL/min, respectively. The 4th photo in c) shows the flow with acoustics OFF to demonstrate a completely unmixed flow. b) Plot of the intensity profiles across the width of the channel for the images shown above.

3. Video Captions

Video 1: Rotation of *C. elegans*

In this video, a *C. elegans* is rotated within the bubble-based acostofluidic device. A 20V, 19.1 kHz signal was applied to the transducer to achieve rotation. Oocytes can be seen clearly in the proximal distal gonad as they rotate into and out of the focal plane of the camera.

Video 2: Particle Trapping and Release for Size Based Sorting.

The video begins with the syringe pump activated and the acoustic transducer OFF. Both 2.5 μm and 25 μm particles flow through the channel freely. Then, the transducer is turned ON (24.5 V and 4.8 kHz), and the flow rate increases do to the acoustic streaming created by the sharp-edges. However, the vibration of the streaming also attracts the larger particles to its surface where they are trapped. The 2.5 μm particles continue to flow through the channel. After collecting several larger particles, the transducer is turned OFF, and the particles are released back into the bulk flow.