```
/*

References for the flex sensor and accelerometer:

https://learn.sparkfun.com/tutorials/flex-sensor-hookup-guide

https://www.robotshop.com/blog/en/arduino-5-minute-tutorials-lesson-7-
accelerometers-gyros-imus-3634//

https://learn.adafruit.com/adafruit-analog-accelerometer-
breakouts/calibration-and-programming

https://chionophilous.wordpress.com/2011/08/26/accelerometer-calibration-ii-
simple-methods/

*/

// HCDE 439 Final Project Code
// By Melody Xu and Chue Yang
// June 3rd, 2018

#include <SoftwareSerial.h>
SoftwareSerial xBee(2, 3);

// Accelerometer Pins
const int xPin = A0;
const int yPin = A1;
const int zPin = A2;

const int FLEX_PIN2 = A4;   // Flex sensor pin
const int LED_PIN1 = 9;     // Green LED1
const int LED_PIN2 = 10;    // Green LED2
const int LED_PIN3 = 11;    // Red LED
const int BUTTON_PIN1 = 6;  // Pause button
const int BUTTON_PIN2 = 5;  // On/off button

const float VCC = 3.3; // voltage of feather
const float R_DIV = 45000.0; // Measured resistance of 47k ohm resistor

const float STRAIGHT_RESISTANCE = 11600; // resistance when straight
const float BEND_RESISTANCE = 24000.0; // resistance at 90 deg

int timeStep = 20;                       // The interval between samples
measured in millisconds
int averagingTime = 1000;                // Length of time over which
acceleration is averaged
int timeSlices = averagingTime / timeStep; // Number of sampling points
during the averaging time
boolean onState = false;
```

```
boolean pause = false;
String bubMode = "";          // Storing code sent to other xbees
String gestMode = "";         // Storing code sent to other xbees

void setup() {
  // initialize serial communications at 9600 bps:
  Serial.begin(9600);
  xBee.begin(9600);
  pinMode(BUTTON_PIN1, INPUT_PULLUP); // Pause button
  pinMode(BUTTON_PIN2, INPUT_PULLUP); // Power button for LEDs
  pinMode(FLEX_PIN2, INPUT);          // Detects bending
  pinMode(LED_PIN1, OUTPUT);          // Flex sensor indicator1
  pinMode(LED_PIN2, OUTPUT);          // Flex sensor indicator2
  pinMode(LED_PIN3, OUTPUT);          // Power/pause indicator
}

void loop() {
  // Detects if button1 is being pressed, flicks red LED to indicate change
  // If yes, sends code to stop bubble machine movement
  if (digitalRead(BUTTON_PIN1) == LOW) {
    Serial.println("Pause");
    if (pause == false) {
      xBee.println("id:bc|gestMode:0");
      pause = true;
    } else {
      pause = false;
    }
    digitalWrite(LED_PIN3, 1);
    delay(100);
    digitalWrite(LED_PIN3, 0);
  }

  // Detects if button2 is being pressed
  // If turns on/off LEDs
  if (digitalRead(BUTTON_PIN2) == LOW) {
    if (onState == false ) {
      digitalWrite(LED_PIN1, 1);
      digitalWrite(LED_PIN2, 1);
      digitalWrite(LED_PIN3, 1);
      onState = true;
      Serial.println("ON");
    } else if (onState == true ) {
      digitalWrite(LED_PIN1, 0);
      digitalWrite(LED_PIN2, 0);
      digitalWrite(LED_PIN3, 0);
      Serial.println("OFF");
      onState = false;
    }
  }
  getFlexSensorValue();
  detectGesture();
}

// Flex sensor data is converted into degrees
// Bend degrees of the flex sensor are converted into 3 states: 0 (no
bending), 1 (slightly bending), and 2 (bending)
// States are sent to control amount of bubbles produced
```

```
void getFlexSensorValue() {
  float flexR = calculateResistance(FLEX_PIN2);
  // Use the calculated resistance to estimate the sensor's bend angle:
  float angle = map(flexR, STRAIGHT_RESISTANCE, BEND_RESISTANCE, 0, 90.0);
  Serial.println("Bend2: " + String(angle) + " degrees");
  Serial.println();
  String tempBubble = "";

  if (angle < 40) {           // Bend degree = Level 0
    digitalWrite(LED_PIN2, 0);
    Serial.println("LED 0");
    digitalWrite(LED_PIN1, 0);
    tempBubble = "bubMode:0";
    if (tempBubble !=  bubMode) {           // prevents sending redundant
data, only sends to xBee if change is detected
      xBee.println("id:bc|" + tempBubble); // coded message to help other
xBees know what kind of data it is
    }
    bubMode = tempBubble;
  } else if (angle < 85) {    // Bend degree = Level 1
    analogWrite(LED_PIN1, 100);
    analogWrite(LED_PIN2, 100);
    Serial.println("LED 1");
    tempBubble = "bubMode:1";
    if (tempBubble !=  bubMode) {
      xBee.println("id:bc|" + tempBubble);
    }
    bubMode = tempBubble;
  } else {                    // Bend degree = Level 2
    analogWrite(LED_PIN1, 255);
    analogWrite(LED_PIN2, 255);
    Serial.println("LED 2");
    Serial.println("id:bc|bubMode:2");
    tempBubble = "bubMode:2";
    if (tempBubble !=  bubMode) {
      xBee.println("id:bc|" + tempBubble);
    }
    bubMode = tempBubble;
  }
}
// Read the flex sensor pin, and calculate voltage and resistance from it
float calculateResistance(int pin) {
  int flexADC = analogRead(pin);
  float flexV = flexADC * VCC / 1023.0;
  float flexR = R_DIV * (VCC / flexV - 1.0);
  Serial.println("Resistance: " + String(flexR) + " ohms");
  return flexR;
}

// Recieve data about acceleration in 3 axes from accelerometer
// Detects gesture and sends signal to other xBees
void detectGesture() {
  float xTotalAcceleration = 0;   // Variable describes total acceleration
  float xAverageAcceleration = 0;   // Variable describes total acceleration

  float yTotalAcceleration = 0;   // Variable describes total acceleration
```

```
  float yAverageAcceleration = 0;   // Variable describes total y
acceleration

  float zTotalAcceleration = 0;   // Variable describes total acceleration
  float zAverageAcceleration = 0;   // Variable describes total z
acceleration

  float magnitudeAverageAcceleration = 0;

  for (int i = 0; i < timeSlices; i = i + 1) {
    xTotalAcceleration = xTotalAcceleration + abs((analogRead(xPin) - 507));
//update the total acceleration so far
    yTotalAcceleration = yTotalAcceleration + abs((analogRead(yPin) - 512));
//update the total acceleration so far
    zTotalAcceleration = zTotalAcceleration + abs((analogRead(zPin) - 615));
//update the total acceleration so far
    delay(timeStep);                              //delay by timestep
  }

  xAverageAcceleration =  xTotalAcceleration / timeSlices;
  yAverageAcceleration =  yTotalAcceleration / timeSlices;
  zAverageAcceleration =  zTotalAcceleration / timeSlices;

  Serial.print("xA, yA, zA: ");
  Serial.print(xAverageAcceleration);
  Serial.print(", ");
  Serial.print(yAverageAcceleration);
  Serial.print(", ");
  Serial.print(zAverageAcceleration);
  Serial.println(" .");

  magnitudeAverageAcceleration = sqrt(pow(xAverageAcceleration, 2) +
pow(yAverageAcceleration, 2) + pow(zAverageAcceleration, 2));
  String tempMode = "";
  // detect motion in Z axis (hand moving up and down), flicks led to signal
change if detected
  if ((zAverageAcceleration / xAverageAcceleration  > 2) &&
(zAverageAcceleration / yAverageAcceleration  > 2) &&
(magnitudeAverageAcceleration > 15)) {
    Serial.print("zAverageAcceleration: ");
    Serial.println(zAverageAcceleration);
    Serial.println("UP AND DOWN");
    gestMode = "gestMode:1";
    xBee.println("id:bc|" + gestMode);
    analogWrite(LED_PIN3, 255);
    delay(30);
    analogWrite(LED_PIN3, 0);
  } else {
    gestMode = "gestMode:0";
  }
  // Additional gestures that didn't get used.
   //  if ((xAverageAcceleration / yAverageAcceleration  > 2) &&
(xAverageAcceleration / zAverageAcceleration  > 2) &&
(magnitudeAverageAcceleration > 15)) {
  //    // if x acceleration is more than twice as big as both y and z
accelerations, indicating moving hand back and forth motion
  //    Serial.print("xAverageAcceleration: ");
```

```
  //    Serial.println(xAverageAcceleration);
  //    Serial.println("BACK AND FORTH");
  //  } else if ((yAverageAcceleration / xAverageAcceleration  > 2) &&
(yAverageAcceleration / zAverageAcceleration  > 2) &&
(magnitudeAverageAcceleration > 15)) {
  //    // if y acceleration is more than twice as big as both x and z
accelerations, indicating moving hand left and right motion
  //    Serial.print("yAverageAcceleration: ");
  //    Serial.println(yAverageAcceleration);
  //    Serial.println("LEFT AND RIGHT");

}
```