Supplementary Information

A step forward for smart clothes—fabric-based microfluidic sensors for wearable health monitoring

Tao Zhang[1], Adam Michael Ratajczak[1], Hui Chen[2], John A. Terrell[1], and Chengpeng Chen[1,*]

1. Department of Chemistry and Biochemistry,
   University of Maryland Baltimore County,
   MD, USA, 21250
2. Department of Chemical, Biochemical and Environmental Engineering,
   University of Maryland Baltimore County,
   MD, USA, 21250

*Corresponding to: Dr. Chengpeng Chen
1000 Hilltop Circle, Baltimore, MD, USA
Chemistry and Biochemistry
University of Maryland Baltimore County
+1-4104553053
cpchen@umbc.edu

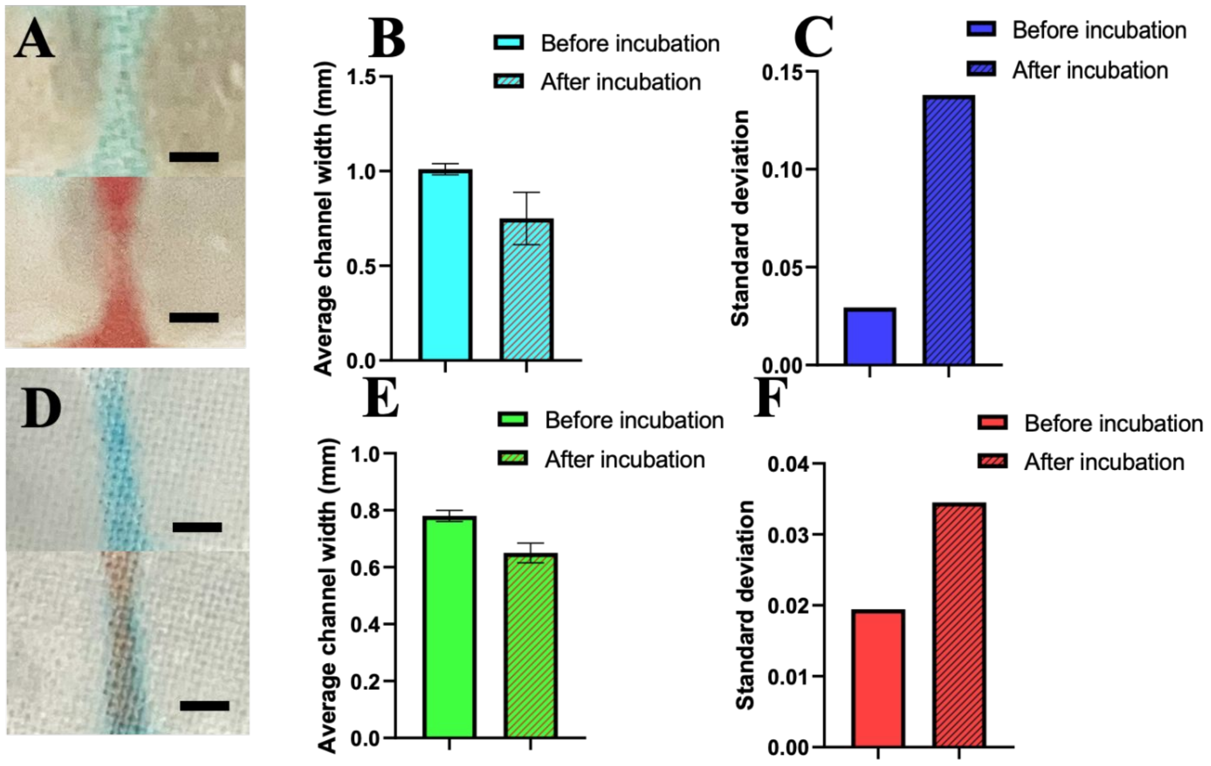


Figure S1. Validation of the unsuitability of wax-printing microfluidics for wearable sensing. (Scale bar = 1 mm). The commonly used paraffin wax for paper microfluidic fabrication was tested, with another common wax of soy wax. (A) The paraffin-printed microchannel before incubation (37 ˚C, 1 hour) was stained with a blue dye (top). The paraffin-based microchannel after incubation was marked with a red dye (bottom). (B) The average paraffin-based channel width before and after the incubation, measured at five random locations across the channel. (N=5, error = standard deviation). (C) The standard deviation of widths of the same paraffin-based channel before and after incubation. (D) The soy wax-printed microchannel before incubation was stained with a blue dye (top). The soy wax-based microchannel after incubation was marked with a red dye (bottom). (E) The average soy wax-based channel width before and after the incubation, measured at five random locations. (N=5, error = standard deviation). (F) The standard deviation of widths of the same soy wax-based channel before and after incubation.

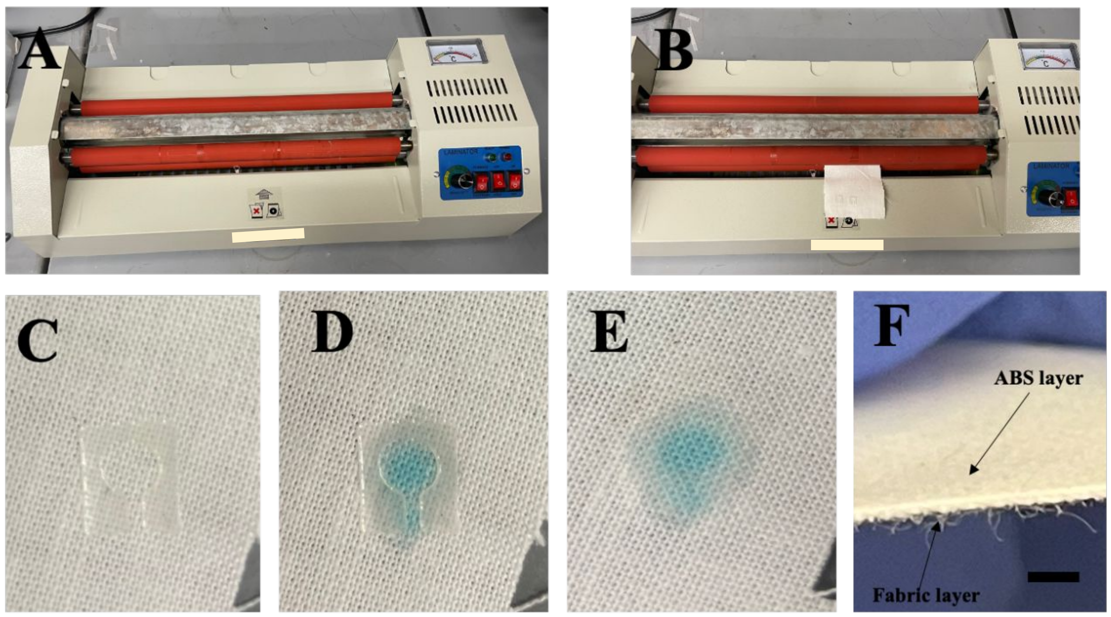


Figure S2. Attempts of using lamination to fuse ABS layers into cotton fabrics. (A, B) The laminator and experimental setup with ABS patterns on top of a piece of fabric fed into the laminator at 200 ^{o}C (glass transition temperature of ABS is 105 ^{o}C[1]). We did not further increase the temperature because cotton fabrics start to combust at 210 ^{o}C.[2] (C) The laminated ABS onto the fabric. The ABS layer was a distinct layer on the fabric. (D) Leakage test on the front side of the laminated device. The blue dye solution added to the reservoir leaked out of the microfluidic areas. (E) The leakage test from the back side (the opposite side where ABS was laminated onto). Leakage was detected. (F) Side view of the laminated layers showing the ABS layer (pale yellow, smooth) stayed on top of the fabric layer (white, fibrous). These data strongly suggest that lamination is not sufficient to form ABS hydrophobic areas through the fabric. (Scale bar = 3 mm)

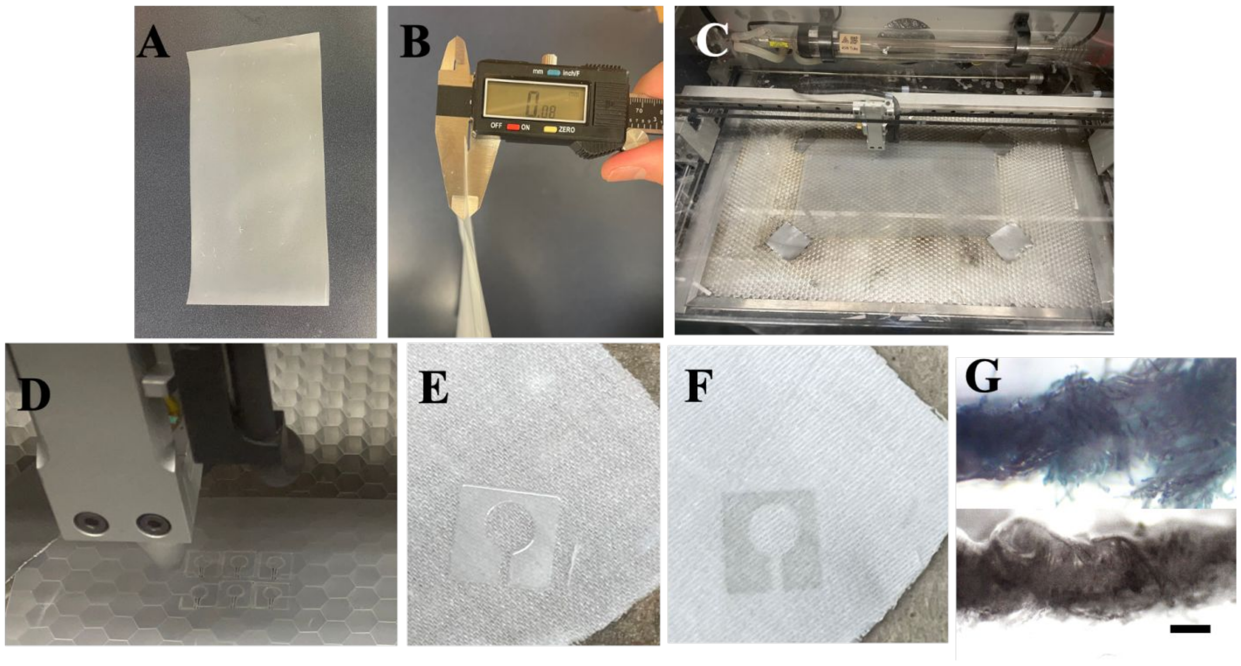


Figure S3. The protocol to fabricate fabric-based microfluidics. (A) View of the ABS film. (B) The thickness of the ABS film = 0.08 mm. (C) The setup of the laser cutter to cut microfluidic patterns from the ABS film. (D) The cut microfluidic patterns from the ABS film. (E) The ABS-fabric stack was placed on top of a layer of paper towel saturated with acetone. (F) A microfluidic device was fabricated with hydrophobic barriers (darker) through the cotton fabric after 10 s. (G) Cross views of fabrics before and after the ABS film infiltration under an optical microscope. A water-soluble hydrophilic blue dye was used to stain the specimen. The top image was just fabric showing blue stained by the dye. The bottom image was fabric with the ABS infiltration showing high hydrophobicity without picking up the hydrophilic blue dye. (Scale bar = 200 μm)

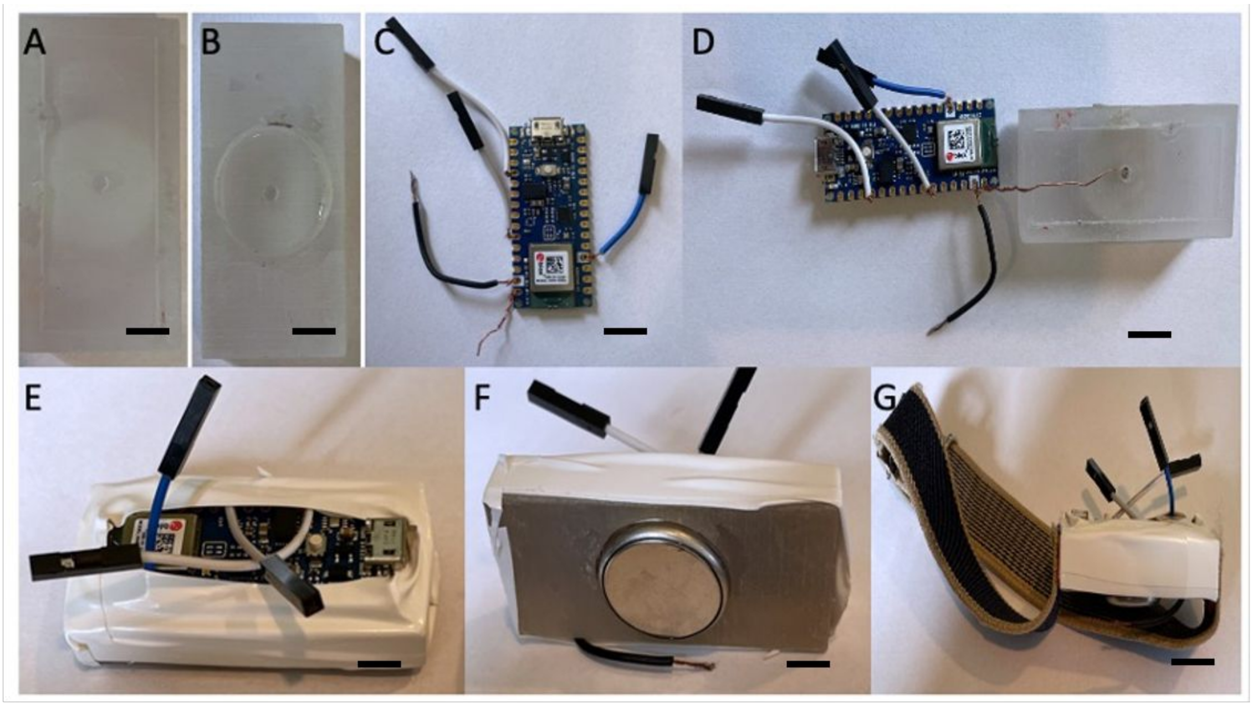
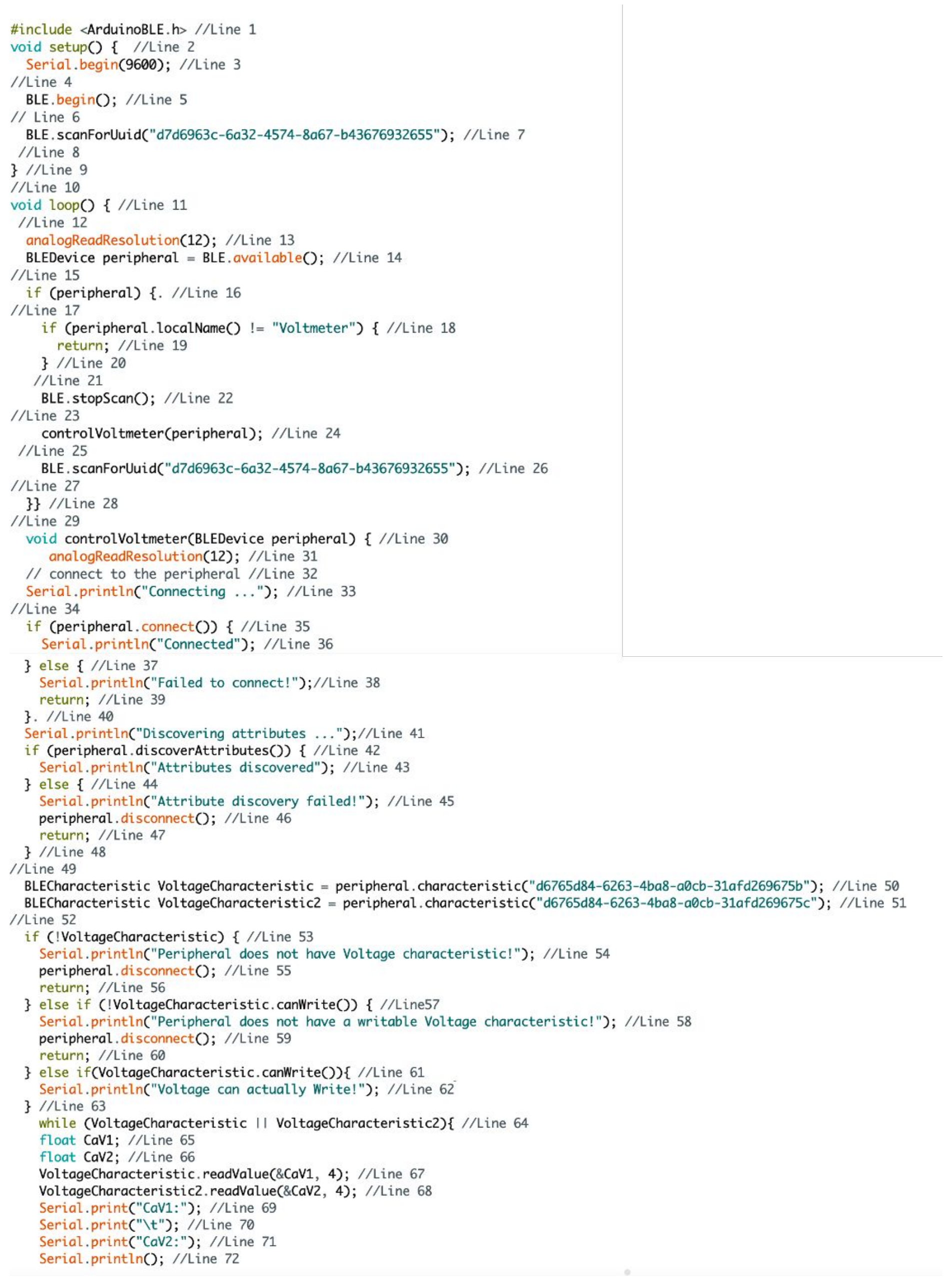


Figure S4. The wearable setup. (A) A top-down view of the 3D-printed Arduino board holder. Scale bar = 5 mm. (B) A bottom-side view of the Arduino board holder – two 3.3V lithium coin batteries were inserted in the circular pit to power the system. Scale bar = 5 mm. (C) A fully wired Arduino board. Note the white wires to the A0 and A6 pins, which received data from the working electrodes. The blue GND wire connects to the reference electrode, whereas the black Ground connected to the batteries under the board to complete the circuit. The bare copper wire is connected to the Vin pin, which was ultimately soldered to the top of the batteries, thereby powering the board. Scale bar = 7 mm. (D) The completed connection. Scale bar = 5 mm. (E) The fully connected Arduino board taped together. Note: the board can also be screwed/bolted in for a tighter fit. Scale bar = 5 mm. (F) An underside view of the fully assembled Arduino. Scale bar = 5 mm. (G) An armband was added to the device to complete the wearable setup. Scale bar = 5 mm.



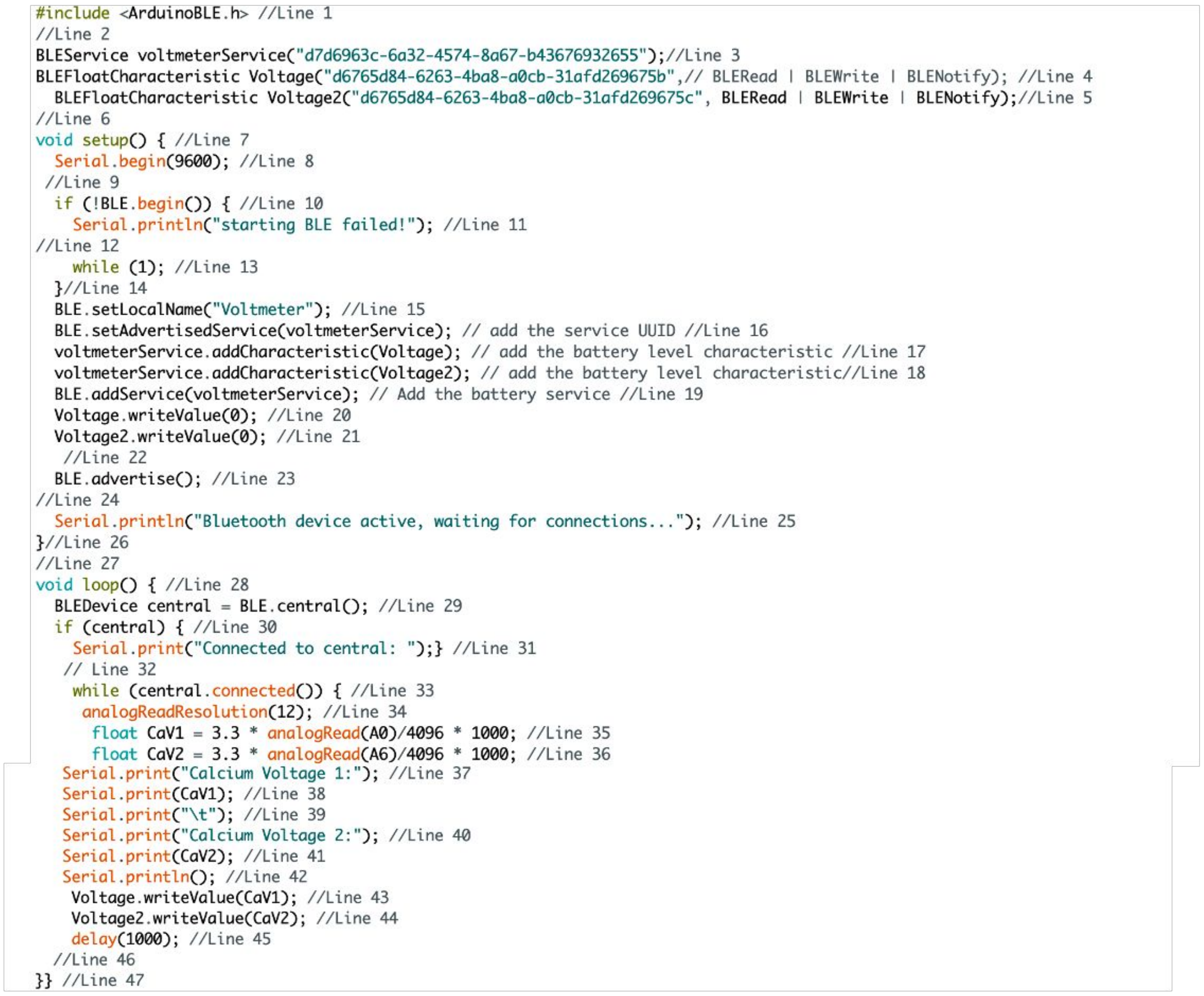
```
#include <ArduinoBLE.h> //Line 1
void setup() {  //Line 2
  Serial.begin(9600); //Line 3
//Line 4
  BLE.begin(); //Line 5
// Line 6
  BLE.scanForUuid("d7d6963c-6a32-4574-8a67-b43676932655"); //Line 7
 //Line 8
} //Line 9
//Line 10
void loop() { //Line 11
 //Line 12
  analogReadResolution(12); //Line 13
  BLEDevice peripheral = BLE.available(); //Line 14
//Line 15
  if (peripheral) {. //Line 16
//Line 17
    if (peripheral.localName() != "Voltmeter") { //Line 18
      return; //Line 19
    } //Line 20
   //Line 21
    BLE.stopScan(); //Line 22
//Line 23
    controlVoltmeter(peripheral); //Line 24
 //Line 25
    BLE.scanForUuid("d7d6963c-6a32-4574-8a67-b43676932655"); //Line 26
//Line 27
  }} //Line 28
//Line 29
  void controlVoltmeter(BLEDevice peripheral) { //Line 30
     analogReadResolution(12); //Line 31
  // connect to the peripheral //Line 32
  Serial.println("Connecting ..."); //Line 33
//Line 34
  if (peripheral.connect()) { //Line 35
    Serial.println("Connected"); //Line 36
  } else { //Line 37
    Serial.println("Failed to connect!");//Line 38
    return; //Line 39
  }. //Line 40
  Serial.println("Discovering attributes ...");//Line 41
  if (peripheral.discoverAttributes()) { //Line 42
    Serial.println("Attributes discovered"); //Line 43
  } else { //Line 44
    Serial.println("Attribute discovery failed!"); //Line 45
    peripheral.disconnect(); //Line 46
    return; //Line 47
  } //Line 48
//Line 49
  BLECharacteristic VoltageCharacteristic = peripheral.characteristic("d6765d84-6263-4ba8-a0cb-31afd269675b"); //Line 50
  BLECharacteristic VoltageCharacteristic2 = peripheral.characteristic("d6765d84-6263-4ba8-a0cb-31afd269675c"); //Line 51
//Line 52
  if (!VoltageCharacteristic) { //Line 53
    Serial.println("Peripheral does not have Voltage characteristic!"); //Line 54
    peripheral.disconnect(); //Line 55
    return; //Line 56
  } else if (!VoltageCharacteristic.canWrite()) { //Line57
    Serial.println("Peripheral does not have a writable Voltage characteristic!"); //Line 58
    peripheral.disconnect(); //Line 59
    return; //Line 60
  } else if(VoltageCharacteristic.canWrite()){ //Line 61
    Serial.println("Voltage can actually Write!"); //Line 62
  } //Line 63
    while (VoltageCharacteristic || VoltageCharacteristic2){ //Line 64
    float CaV1; //Line 65
    float CaV2; //Line 66
    VoltageCharacteristic.readValue(&CaV1, 4); //Line 67
    VoltageCharacteristic2.readValue(&CaV2, 4); //Line 68
    Serial.print("CaV1:"); //Line 69
    Serial.print("\t"); //Line 70
    Serial.print("CaV2:"); //Line 71
    Serial.println(); //Line 72
```

Figure S5. The Arduino code to assess the analytical merits of the fabric-based microfluidic sensor with a computer

LINE	**PURPOSE**
1	Incorporates the Arduino BLE library into the code, which is necessary for much of the syntax used throughout.
2	Void setup defines the basics of how the code is to run for the following program. Anything inside the brackets tells the Arduino how to begin.
3	This tells the Arduino to adhere to a baud rate of 9600 (this is a standard rate that is often used for similar applications)

4	Blank
5	Tells Arduino to initialize the BLE device
6	blank
7	Tells the Arduino to scan for this unique identifier (uuid is defined in the peripheral device code)
8	blank
9	End bracket of void setup – Arduino is prepped to run the experiment
10	blank
11	Void Loop defines the following code as running repeatedly – the following code will be used for every measurement
12	blank
13	Sets the resolution of the measurements to 12 bits – this enhances resolution between all data points acquired
14	Tells Arduino that if another device is detected via BLE, then this is the peripheral device.
15	blank
16	Beginning of if-statement for the presence of a peripheral device. If a peripheral BLE device is detected, the following code (within brackets) will be run.
17	blank
18	If the name of the detected device is not Voltmeter, the code within the following brackets will be run
19	The device is ignored
20	End bracket
21	blank
22	Stops scanning for BLE devices
23	Blank
24	When the voltmeter is not found, continue searching for the device
25	blank
26	Continues scanning for the desired device via the uuid defined for its service (view the wearable code)
27	blank
28	End brackets of void setup code and if-statement
29	blank
30	The following code is defined for once the peripheral BLE device is connected
31	Once again sets the resolution of the measurements to 12 bits
32	Comment stating what the following code is intended to do
33	Prints “Connecting…” to the serial monitor as the BLE central attempts to establish a strong connection to the peripheral
34	blank
35	If the peripheral is connected – run the following code (within the brackets)
36	Print “Connected” to the serial monitor to confirm the connection has occurred
37	If the peripheral is not connected, run the following code (within the brackets)

38	Print “Failed to connect” to the serial monitor to confirm connection failure
39	Break from the code – this will restart the search for a device
40	End bracket
41	Prints line “Discovering attributes” to the serial monitor – this finds whether the peripheral is able to share data
42	If attributes are discovered run the following code (between the brackets)
43	Print “Attributes discovered” to the serial monitor
44	If attributes are not discovered, run the following code
45	Print “Attribute discovery failed!” to the serial monitor
46	Disconnect from the peripheral device
47	Break and start over searching for a peripheral device
48	End bracket
49	Blank
50	Defines a unique identifier for data transmission from one calcium-selective electrode in the voltmeter
51	Defines a unique identifier for data transmission from the other calcium-selective electrode in the voltmeter
52	Blank
53	If the voltage characteristic is not connected, run the following code
54	Print the line: “Peripheral does not have the Voltage characteristic!”
55	Disconnect the peripheral
56	Break
57	If Voltage characteristic is present, and it cannot present data, run the following code.
58	Print “Peripheral does not have a writable Voltage characteristic!” to the monitor
59	Disconnect the peripheral
60	Break
61	If the Voltage characteristic is present, and it can send data, then run the following code.
62	Print “Voltage can write!” to the serial monitor
63	End bracket
64	While both voltage characteristics are sending data, run the following code
65	Define CaV1 as a float variable
66	Define CaV2 as a float variable
67	Read values from VoltageCharacteristic and assign these as the output of CaV1 to 4 digits
68	Read values from VoltageCharacteristic2 and assign these as the output of CaV2 to 4 digits
69	Print the value of CaV1 to the serial monitor
70	Print a tab (helps to organize the data)
71	Print the value of CaV1 to the serial monitor
72	Produce a new line (basically adds a new row to the data between each data point)



```
#include <ArduinoBLE.h> //Line 1
//Line 2
BLEService voltmeterService("d7d6963c-6a32-4574-8a67-b43676932655");//Line 3
BLEFloatCharacteristic Voltage("d6765d84-6263-4ba8-a0cb-31afd269675b",// BLERead | BLEWrite | BLENotify); //Line 4
  BLEFloatCharacteristic Voltage2("d6765d84-6263-4ba8-a0cb-31afd269675c", BLERead | BLEWrite | BLENotify);//Line 5
//Line 6
void setup() { //Line 7
  Serial.begin(9600); //Line 8
 //Line 9
  if (!BLE.begin()) { //Line 10
    Serial.println("starting BLE failed!"); //Line 11
//Line 12
    while (1); //Line 13
  }//Line 14
  BLE.setLocalName("Voltmeter"); //Line 15
  BLE.setAdvertisedService(voltmeterService); // add the service UUID //Line 16
  voltmeterService.addCharacteristic(Voltage); // add the battery level characteristic //Line 17
  voltmeterService.addCharacteristic(Voltage2); // add the battery level characteristic//Line 18
  BLE.addService(voltmeterService); // Add the battery service //Line 19
  Voltage.writeValue(0); //Line 20
  Voltage2.writeValue(0); //Line 21
   //Line 22
  BLE.advertise(); //Line 23
//Line 24
  Serial.println("Bluetooth device active, waiting for connections..."); //Line 25
}//Line 26
//Line 27
void loop() { //Line 28
  BLEDevice central = BLE.central(); //Line 29
  if (central) { //Line 30
    Serial.print("Connected to central: ");} //Line 31
   // Line 32
    while (central.connected()) { //Line 33
     analogReadResolution(12); //Line 34
      float CaV1 = 3.3 * analogRead(A0)/4096 * 1000; //Line 35
      float CaV2 = 3.3 * analogRead(A6)/4096 * 1000; //Line 36
   Serial.print("Calcium Voltage 1:"); //Line 37
   Serial.print(CaV1); //Line 38
   Serial.print("\t"); //Line 39
   Serial.print("Calcium Voltage 2:"); //Line 40
   Serial.print(CaV2); //Line 41
   Serial.println(); //Line 42
    Voltage.writeValue(CaV1); //Line 43
    Voltage2.writeValue(CaV2); //Line 44
    delay(1000); //Line 45
  //Line 46
}} //Line 47
```

Figure S6. The Arduino code for wearable applications

LINE	PURPOSE
1	Incorporates the Arduino BLE library into the code, which is necessary for much of the syntax used throughout.
2	Blank
3	Defines the service offered by the device as "Voltmeter" and gives it a unique identifier
4	Defines the Voltage characteristic as a float variable and calls for it to be available to be read, written, and for notifications of changes to be made
5	Defines the Voltage2 characteristic as a float variable and calls for it to be available to be read, written, and for notifications of changes to be made
6	Blank
7	Initial setup code: to following is used to start the Arduino and tell it how to prepare for the coming experiment. This code is only run once.
8	Baud rate is again set to 9600

9	Blank
10	If the BLE does not start to transmit, run the following code
11	Print “Starting BLE failed!” to the serial monitor
12	blank
13	Continue to loop through this code until BLE detected
14	End bracket
15	Set the name of the BLE device as Voltmeter
16	Make the service defined earlier the service which the device will advertise
17	Add the Voltage characteristic to this service
18	Add the Voltage2 characteristic to this service
19	Add the Voltmeter service
20	Write the initial value for the Voltmeter characteristic as 0
21	Write the initial value for the Voltmeter2 characteristic as 0
22	Blank
23	Advertise the BLE device (the peripheral)
24	Blank
25	Print what is within quotes to the serial monitor
26	End bracket
27	Blank
28	The void loop defines code that will be run repeatedly throughout the experiment
29	The BLE device connecting to the peripheral is set as the central
30	If the central is present, the following code is run
31	Print “Connected to central” to the serial monitor
32	Blank
33	While the central is connected to the peripheral, run the following code
34	Set the resolution of the data transmission to 12 bits
35	Define the variable CaV1 as being calculated by the following equation.
36	Define the variable CaV2 as being calculated by the following equation.
37	Print “Calcium Voltage 1:” to the serial monitor
38	Print the values of CaV1 to the serial monitor
39	Print a tab to help with data organization
40	Print “Calcium Voltage 2:” to the serial monitor
41	Print the values of CaV2 to the serial monitor
42	Print a blank line to add a row to the data
43	Tells the Voltage characteristic to write the value of CaV1
44	Tells the Voltage2 characteristic to write the value of CaV2
45	Take one data point every 1 second
46	Blank
47	End brackets

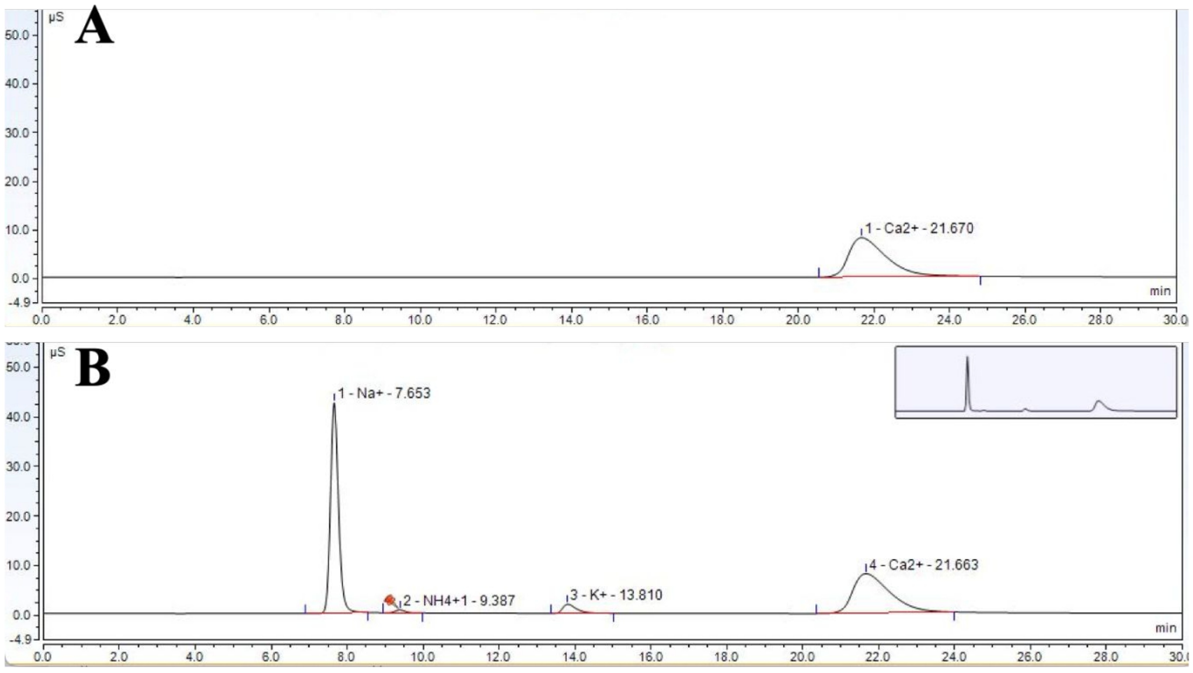


Figure S7. Example of ion chromatography to detect [Ca^{2+}] in sweat for result validation. (A) Standard Ca^{2+} solution (0.5 mM). (B) Sweat samples (10 µL) spiked in the standard [Ca^{2+}] (5 mL).

Reference # in manuscript	The detection range	Ca^{2+} LOD
53	0.25 mM – 2 mM	N/A
56	1 µm - 10 mM	1 µM
57	0.01 mM - 100 mM	~8 µM
58	1 mM - 10 mM	N/A
59	0.1 mM - 100 mM	N/A

Table S1. Summarization of detection range and LOD of recent Ca^{2+} ion selective electrode for wearable application

Reference:

1. Billah, K. M. M.; Lorenzana, F. A.; Martinez, N. L.; Chacon, S.; Wicker, R. B.; Espalin, D. In *Thermal analysis of thermoplastic materials filled with chopped fiber for large area 3D printing*, 2019 International Solid Freeform Fabrication Symposium, University of Texas at Austin: 2019.

2. Chang, S.; Condon, B.; Smith, J.; Nam, S., Flame Resistant Cotton Fabric Containing Casein and Inorganic Materials Using an Environmentally-Friendly Microwave Assisted Technique. *Fibers and Polymers* **2020,** *21* (10), 2246-2252.