Supporting Information

Early detection and monitoring of chronic wounds using low-cost, omniphobic paper-based smart bandages

Aniket Pal \textsuperscript{a}, Debkalpa Goswami \textsuperscript{a}, Hugo E. Cuellar \textsuperscript{a}, Beatriz Castro \textsuperscript{b}, Shihuan Kuang \textsuperscript{b}, and Ramses V. Martinez \textsuperscript{a,c,}* 

\textsuperscript{a} School of Industrial Engineering, Purdue University, 315 N. Grant Street, West Lafayette, IN 47907, USA

\textsuperscript{b} Department of Animal Sciences, Purdue University, 270 S. Russell St, West Lafayette, IN 47907, USA

\textsuperscript{c} Weldon School of Biomedical Engineering, Purdue University, 206 S. Martin Jischke Drive, West Lafayette, IN 47907, USA

* Corresponding author.
E-mail address: rmartinez@purdue.edu

Keywords: smart bandages, chronic wound monitoring, pressure ulcers, wearable sensor, pH sensor, uric acid sensor.
1. **Stencil printing of flexible conductive electrodes on omniphobic paper**

We used AutoCAD 2016 (Autodesk Inc.) to design the electrodes embedded in the omniphobic paper-based smart bandages (OPSBs). We fabricated the stencil masks used to print the electrodes by cutting the designs into transparency sheets (VWO100C-BE, Apollo Inc.) using a commercial CO₂ laser cutter (MT3050D, Morntech Inc.). After cutting, the transparency sheets served as the stencil masks used to print the electrodes on omniphobic Whatman #1 paper with conductive inks (AGCL-675 and C-200, Applied Ink Solutions). Fig. S1 shows the dimensions of the different electrodes embedded in the OPSBs. To pattern the electrodes used to measure uric acid (UA), the working and counter electrodes (WE and CE) were first printed using carbon ink, followed by the reference electrode (RE) and contact pads using Ag/AgCl ink. We overlapped the electrodes and contact pads to ensure good electrical conductivity.
Fig. S1. Schematics and pictures of the stencil-printed electrodes embedded in the OPSBs. (a) Carbon and Ag/AgCl electrodes used for the electrochemical quantification of UA at the wound site. (b) Electrodes used to monitor pH levels at the wound site. (c) Electrode array used for the early detection and monitoring of pressure ulcers on an in-vivo mouse model.
2. **Reproducibility of stencil printed electrodes**

We measured the resistance of 25 sets of electrodes used to measure UA (Fig. S1a). The measured resistances for each of the electrodes followed a normal distribution with low variability (coefficients of variation for WE, RE and CE are 16%, 11%, and 12% respectively, Fig. S2). The WE and CE, printed with carbon ink, have a higher resistance (mean of 120 $\Omega$ and 225 $\Omega$, respectively). The RE, which was printed with Ag/AgCl ink, has a much lower resistance (mean of 1.1 $\Omega$).

![Graphs showing distribution of measured resistances for WE, RE, and CE electrodes.](image)

**Fig. S2.** Reproducibility of stencil printed electrodes. The distribution of the measured resistances of (a) working electrode, (b) reference electrode, and (c) counter electrode used to measure UA with OPSBs.

3. **Gas Permeability of omniphobic paper-based sensors**

We fabricated our sensors on Whatman #1 paper, which was rendered omniphobic by spraying a 2% solution of R$_3$SiCl$_3$ in IPA. This solution makes the paper omniphobic by chemically altering the surface of cellulose fibers without clogging the pores of the paper. The porosity of the omniphobic paper ensures that the OPSBs remains breathable and do not impede the oxygenation of the wound. We demonstrate the breathability of omniphobic paper by covering a vial of ammonium hydroxide (NH$_4$OH, 35.50 g/mol, Consolidated Chemical and Solvents LLC.) with a piece of the omniphobic paper and depositing a drop of pH indicator (pH = 5, pH indicator A20, Water Company Inc.) on it. Fig. S3 and Movie S1 shows that the color of the of the indicator changes due to the NH$_3$ gas flow of through the omniphobic paper.
**Fig. S3.** Snapshots from Movie S1 demonstrates the gas permeability of omniphobic paper. (a) We placed a drop of a liquid pH indicator on top of omniphobic paper, and separated it from an open scintillation vial containing NH$_4$OH by a plastic film (gas barrier). (b) When the plastic film is removed, NH$_3$ in the gas phase crosses the porous structure of the omniphobic paper and reacts with the pH indicator at time $t = 0$ s. (c, d) The color of the pH indicator gradually changes from yellow (pH = 5) to green (pH = 8) in 20 s as the NH$_3$ gas permeates through the omniphobic paper.

4. **Design of the housing for the wearable potentiostat**

The components of the wearable potentiostat were enclosed in a foam-based housing to keep it light in weight (~8 g, Fig. S4). A “smiley face” sticker was placed on top to increase patient compliance and adherence.
5. Diffusion controlled electrochemical reactions governed by the Cottrell equation

During chronamperometry measurements, we applied a constant voltage (300 mV) and measured the resulting current due to redox reactions between the electrochemical species. Diffusion controlled reactions follow the Cottrell Equation (Eq. S1) (Hamedi et al., 2016), where the current \( i \) is proportional to the initial concentration \( C_0 \) of the analyte:

\[
i = \frac{nFAC_0\sqrt{D}}{\sqrt{\pi t}} \tag{S1}
\]

Here \( n \) is the number of electrons, \( F \) is the Faraday constant (96,485 C/mol), \( A \) is the area of the electrode, \( t \) is time, and \( D \) is the diffusion coefficient. Using the Cottrell equation, we can calibrate the concentration of the analyte by measuring the current of the diffusion controlled electrochemical reaction. Integrating the measured current with respect to time, we obtain the net charge exchanged \( (Q) \) during the redox reaction, which also is proportional to the initial concentration. (Eq. S2).

\[
Q = \int_0^t idt; \quad Q = -\frac{nFAC_0\sqrt{D}t}{2\sqrt{\pi}} \tag{S2}
\]

We used \( Q \) to calculate the concentration of UA using chronocoulometry.
6. Oxidation of uric acid in OPSBs

We used uricase in the OPSBs, so that it oxidises UA and produces allantoin (Eq. S3). Potassium ferricyanide is reduced to potassium ferrocyanide to complete the redox couple (Pal et al., 2017).

\[
\text{uric acid} + K_3[Fe^{III}(CN)_6]^{uricase} \rightarrow \text{allantoin} + K_4[Fe^{II}(CN)_6]
\]

7. pH sensing mechanism of OPSBs

We fabricated omniphobic paper-based pH sensors using a pH sensitive polymer (polyaniline, PANi). We mixed 150 mg of PANi and 250 mg of silver microflakes in 5 mL of IPA and pipetted 10 µL of the solution between the electrodes to fabricate the pH sensors. Polyaniline changes between two forms—the blue colored emeraldine base (PANi-EB), and the green colored emeraldine salt (PANi-ES). Fig. S5 shows how, in the presence of protonic acids, PANi-EB transforms to PANi-ES. The ES phase of PANi is more conductive (~10 S/m) than the EB phase (~10^{-8} S/m) (Sukitpaneenit et al., 2007). This difference in conductivity allows us to calibrate the impedance of the sensor to the pH of the wound exudate model.

**Fig. S5.** Chemical structures of PANi-EB and PANi-ES and their reversible transformation by protonic acids.
8. Wound exudate model used to measure pH using OPSBs

Table S1 shows the volumes in which we mixed 0.2 M disodium phosphate and 0.1 M citric acid to obtain 20 mL McIlvaine buffers of different pH values (Pearse, 1968). We used solutions with 5.5–8.5 pH values—covering the clinical range of pH values in open wounds—to calibrate the pH sensors embedded in the OPSBs.

Table S1. McIlvaine buffer compositions used to generate the 20-mL-solutions with pH values ranging 5.5 to 8.5.

<table>
<thead>
<tr>
<th>pH</th>
<th>0.2M disodium phosphate (mL)</th>
<th>0.1M citric acid (mL)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.5</td>
<td>11.5</td>
<td>8.5</td>
</tr>
<tr>
<td>6.0</td>
<td>12.6</td>
<td>7.4</td>
</tr>
<tr>
<td>6.5</td>
<td>14.5</td>
<td>5.5</td>
</tr>
<tr>
<td>7.0</td>
<td>16.5</td>
<td>3.5</td>
</tr>
<tr>
<td>7.5</td>
<td>18.0</td>
<td>2.0</td>
</tr>
<tr>
<td>8.0</td>
<td>19.5</td>
<td>0.5</td>
</tr>
<tr>
<td>8.5</td>
<td>20.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>
9. Scanning Electron Microscopy

We used silver microflakes to fabricate a Ag/PANi-ES composite for the pH sensors. We took scanning electron microscope (SEM) images of the microflakes (Fig. S6a, b) and the Ag/PANi-ES composite (Fig. S6c, d) using an FEI Nova NanoSEM. The composites were prepared on Whatman #1 paper and sputter coated with a ~10-nm-thick layer of Pt using a 20 mA filament current for 60 s. The unprocessed silver microflakes were imaged without requiring any metallization (Fig. S6a). We used a 5.0 kV electron accelerating potential for all images. A 4.8 mm working distance with an Everhart-Thornley Detector (ETD) was used to image all the Ag/PANi composite samples. For the silver microflakes, we used a 5.7 mm working distance with a Through the Lens Detector (TLD).

Fig. S6. Scanning electron microscopy images of Ag/PANi layers. (a) Ag microflakes used to increase the conductivity of PANi-EB. (b) High resolution SEM image of a representative microflake. (c) Image of Ag/PANi-ES layer printed on Whatman #1 paper. (d) High resolution SEM image of the Ag/PANi-ES composite.
10. *In-vivo* detection of pressure ulcers

Laboratory mice (C57B6J, 8 to 15 weeks old, male) were used in our experiments (Fig. S7a). Mice were of mixed backgrounds and housed in the animal facility with free access to water and standard rodent chow food. The mice were anaesthetized using ketamine-xylazine cocktail (0.1 g/kg of body weight). We shaved the back of the mice and then used Nair depilatory cream to remove all the hair (Fig. S7b) and cleaned the area with Dawn detergent. Fig. S7c shows how the shaved skin was gently tented up and placed between two disc magnets (10 mm diameter, 2 mm thickness). The magnets did not interfere with the normal activity of the mice once the anesthesia had worn off. The magnets were kept in place for 1 h or 3 h to create different degrees of tissue damage. Each mouse received only one pressure induced wound. The mice were always anesthetized prior to performing impedance spectroscopy. All procedures involving mice were performed in accordance with Purdue University’s Animal Care and Use Committee.

![Mice preparation for the *in-vivo* early detection of pressure ulcers.](image)

**Fig. S7.** Mice preparation for the *in-vivo* early detection of pressure ulcers. (a) Mouse model used in the experiments. (b) The hair covering the back of the mouse is shaved using a trimmer and depilatory cream. (c) The shaved skin is pressed between two disc magnets to create an *in-vivo* pressure ulcers model.

11. Impedance spectroscopy using an AD5933 IC

The wearable potentiostat uses an integrated circuit chip, AD5933 (Analog Devices Inc.) to perform impedance spectroscopy. The AD5933 chip outputs an AC excitation signal, which passes through the tissue under the OPSB and is received by the chip. The impedance is calculated by taking the voltage to
current ratio. The excitation signal is created at a certain frequency by a 27-bit phase accumulator direct digital synthesizer (DDS) core in the AD5933 chip. During the impedance spectroscopy, the frequency was swept from 10 to 100 kHz with an AC voltage of 0.97 V with a DC bias of 0.76 V. The response signal from the tissue passes through a gain amplifier and a low pass filter to increase the signal-to-noise ratio. The signal is then sampled by the on-board analog-to-digital converter (ADC). Then a discrete Fourier transform (DFT) is processed by an on-board digital signal processing (DSP) engine. Finally, the DSP engine returns the real and imaginary components of the signal.

12. Fabrication costs

We estimate the cost of each OPSB to be less than $0.10 without considering labor costs or capital expenses (Table S2, S3).

Table S2. Itemized cost of materials to fabricate an OPSB for monitoring open wounds.

<table>
<thead>
<tr>
<th></th>
<th>Cost of OPSB for open wounds</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>UA sensor</td>
</tr>
<tr>
<td>Ag/AgCl ink</td>
<td>$0.00500</td>
</tr>
<tr>
<td>C ink</td>
<td>$0.00100</td>
</tr>
<tr>
<td>Whatman #1 paper</td>
<td>$0.00100</td>
</tr>
<tr>
<td>Omniphobic solution</td>
<td>$0.00008</td>
</tr>
<tr>
<td>Reagents</td>
<td>$0.00500</td>
</tr>
<tr>
<td>Commercial bandage</td>
<td></td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
</tr>
</tbody>
</table>

Table S3. Itemized cost of materials to fabricate an OPSB for monitoring pressure ulcers.

<table>
<thead>
<tr>
<th></th>
<th>Cost of OPSB for pressure ulcers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ag/AgCl ink</td>
<td>$0.03500</td>
</tr>
<tr>
<td>Whatman #1 paper</td>
<td>$0.00350</td>
</tr>
<tr>
<td>Omniphobic solution</td>
<td>$0.00056</td>
</tr>
<tr>
<td>Conductive hydrogel</td>
<td>$0.00004</td>
</tr>
<tr>
<td>Commercial bandage</td>
<td>$0.05000</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>$0.08910</strong></td>
</tr>
</tbody>
</table>
The wearable potentiostat to work with the OPSB costs ~$18, the individual component costs are shown in Table S4.

**Table S4.** Cost of components to fabricate the wearable potentiostat.

<table>
<thead>
<tr>
<th>Cost of components of the wearable potentiostat</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Electrochemical sensor (LMP91000)</td>
<td>$3.05</td>
</tr>
<tr>
<td>3.6V coin battery (LIR2032)</td>
<td>$2.75</td>
</tr>
<tr>
<td>Arduino Nano microcontroller</td>
<td>$4.00</td>
</tr>
<tr>
<td>Impedance analyser (AD5933)</td>
<td>$7.47</td>
</tr>
<tr>
<td>RF Transceiver (nRF24l01)</td>
<td>$1.57</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>$18.84</td>
</tr>
</tbody>
</table>

13. **Disposability of the OPSBs**

OPSBs can be easily disposed by incineration, generating minimal amounts of solid byproducts (Fig. S8).

**Fig. S8.** Demonstration of the disposal of an OPSB by incineration. All scale bars are 2 cm.
14. Arduino code to operate the wearable potentiostat

14.1. Arduino code to perform chronoamperometry using the LMP91000 chip

```c
#include <Wire.h>
#include "LMP91000.h"

int NumberOfScans = 1;
int Amp350Time = 10; // in minutes
int Amp120Time = 4; // in minutes
int Amp35Time = 6; // in minutes
int InitialDelayTime = 10; // in milliseconds
int ScanNumber = 0;
LMP91000 lmp91000;
int NoOf350Samples = Amp350Time * 600; // 10 readings per second
int NoOf120Samples = Amp120Time * 600; // 10 readings per second
int NoOf35Samples = Amp35Time * 600; // 10 readings per second

void setup(void) {
    Serial.begin(115200);
    Serial.println("LMP91000 Test");
    Serial.println(NoOf120Samples);
    Serial.println(NoOf350Samples);
    Wire.begin();
}

void loop(void) {
    lmp91000.configure(
        LMP91000_TIA_GAIN_350K   | LMP91000_RLOAD_10OHM,
        LMP91000_REF_SOURCE_INT  | LMP91000_INT_Z_20PCT
        | LMP91000_BIAS_SIGN_POS  | LMP91000_BIAS_0PCT,
        LMP91000_FET_SHORT_DISABLED | LMP91000_OP_MODE_AMPEROMETRIC
    );
    if (millis() > InitialDelayTime) {
        if (ScanNumber < NumberOfScans) {
            lmp91000.configure(
                LMP91000_TIA_GAIN_350K   | LMP91000_RLOAD_10OHM,
                LMP91000_REF_SOURCE_INT  | LMP91000_INT_Z_20PCT
                | LMP91000_BIAS_SIGN_POS  | LMP91000_BIAS_6PCT,
                LMP91000_FET_SHORT_DISABLED | LMP91000_OP_MODE_AMPEROMETRIC
            );
            Serial.print("Config Result: ");
            Serial.println(res);
            for (int i = 0; i < NoOf350Samples; i++) {
                delay(100);
                Serial.println(analogRead(A1));
            }
            lmp91000.configure(
                LMP91000_TIA_GAIN_120K   | LMP91000_RLOAD_10OHM,
                LMP91000_REF_SOURCE_INT  | LMP91000_INT_Z_20PCT
                | LMP91000_BIAS_SIGN_POS  | LMP91000_BIAS_6PCT,
            );
        }
    }
}```
14.2. **Arduino code to perform impedance spectroscopy using the AD5933 chip**

```c
#include "Wire.h"
define button 2
#define SLAVE_ADDR 0x0D
#define ADDR_PTR 0xB0
#define START_FREQ_R1 0x82
#define START_FREQ_R2 0x83
#define START_FREQ_R3 0x84
#define FREQ_INCRE_R1 0x85
#define FREQ_INCRE_R2 0x86
#define FREQ_INCRE_R3 0x87
#define NUM_INCRE_R1 0x88
#define NUM_INCRE_R2 0x89
#define NUM_SCYCLES_R1 0x8A
#define NUM_SCYCLES_R2 0x8B
#define RE_DATA_R1 0x94
#define RE_DATA_R2 0x95
#define IMG_DATA_R1 0x96
#define IMG_DATA_R2 0x97
#define TEMP_R1 0x92
#define TEMP_R2 0x93
#define CTRL_REG 0x80
#define CTRL_REG2 0x81
#define STATUS_REG 0x8F
```
const float MCLK = 16.776 * pow(10, 6); // AD5933 Internal Clock Speed
16.776 MHz
const float start_freq = 60 * pow(10, 3); // Set start freq, < 100Khz
const float incre_freq = 1 * pow(10, 3); // Set freq increment
const int incre_num = 40; // Set number of increments; < 511

char state;

void setup() {
    Wire.begin();
    Serial.begin(115200);
    pinMode(button, INPUT);
    writeData(CTRL_REG, 0x0);  //clear ctrl-reg
    writeData(CTRL_REG2, 0x10);  //reset ctrl register
    programReg();
}

void loop() {
    if (Serial.available() > 0) {
        state = Serial.read();
        switch (state) {
            case 'A':  //Program Registers
                programReg();
                break;
            case 'B':  //Measure Temperature
                measureTemperature();
                break;
            case 'C':
                runSweep();
                delay(1000);
                break;
        }
        Serial.flush();
    }
}

void programReg() {
    writeData(CTRL_REG, 0x01);  // Set Range 1, PGA gain 1
    writeData(NUM_SCYCLES_R1, 0x07);  // Set settling cycles
    writeData(NUM_SCYCLES_R2, 0xFF);
    // Start frequency of 1kHz
    writeData(START_FREQ_R1, getFrequency(start_freq, 1));
    writeData(START_FREQ_R2, getFrequency(start_freq, 2));
    writeData(START_FREQ_R3, getFrequency(start_freq, 3));
    // Increment by 1 kHz
    writeData(FREG_INCRE_R1, getFrequency(incre_freq, 1));
    writeData(FREG_INCRE_R2, getFrequency(incre_freq, 2));
    writeData(FREG_INCRE_R3, getFrequency(incre_freq, 3));
    // Points in frequency sweep (100), max 511
    writeData(NUM_INCRE_R1, (incre_num & 0x001F00) >> 0x08 );
    writeData(NUM_INCRE_R2, (incre_num & 0x0000FF));
}

void runSweep() {
short re;  
short img;  
double freq;  
double mag;  
double phase;  
double gain;  
double Impedance;  
double GF;  
double FFW;  
double wt;  
double BF;  
double totmag = 0;  
double count = 0;  
double avgmag;  
double totimp = 0;  
double avgimp;  
int i = 0;  
double Resistance;  
double Reactance;  

programReg();  
// 1. Standby '10110000' Mask D8-10 of avoid tampering with gains  
writeData(CTRL_REG, (readData(CTRL_REG) & 0x07) | 0xB0);  
// 2. Initialize sweep  
writeData(CTRL_REG, (readData(CTRL_REG) & 0x07) | 0x10);  
// 3. Start sweep  
writeData(CTRL_REG, (readData(CTRL_REG) & 0x07) | 0x20);  

while ((readData(STATUS_REG) & 0x07) < 4) {  
delay(100); // delay between measurements  
int flag = readData(STATUS_REG) & 2;  
if (flag == 2) {  
byte R1 = readData(RE_DATA_R1);  
byte R2 = readData(RE_DATA_R2);  
re = (R1 << 8) | R2;  
R1 = readData(IMG_DATA_R1);  
R2 = readData(IMG_DATA_R2);  
img = (R1 << 8) | R2;  
freq = start_freq + i * incre_freq;  
mag = sqrt(pow(double(re), 2) + pow(double(img), 2));  
totmag = totmag + mag;  
//Gain factor calibration  
GF = 103.496 * (pow(10, -11)) + 811.151 * (pow(10, -18)) * (freq -  
start_freq); //100kohm 40-100kHz  
Impedance = 1 / (GF * mag);  
totimp = totimp + Impedance;  
count = count + 1;  
Resistance = 1 / (GF * re);  
Reactance = 1 / (GF * img);  
phase = atan(double(img) / double(re));  
phase = (180.0 / 3.1415926) * phase; //convert phase angle to  
degrees  
//Phase Calibration

System phase = 118;
phase = phase - 118;

Serial.print("Frequency: ");
Serial.print(freq / 1000);
Serial.print("kHz");
Serial.print(" Magnitude: ");
Serial.print(mag);
Serial.print(" Impedance: ");
Serial.print(Impeance / 1000);
Serial.println("kHz");

//Increment frequency
if ((readData(STATUS_REG) & 0x07) < 4) {
    writeData(CTRL_REG, (readData(CTRL_REG) & 0x07) | 0x30);
    i++;
    avgmag = totmag / count;
    avgimp = totimp / count;
}
Serial.print(" Avg Mag: ");
Serial.print(avgmag);
Serial.print(" ");
Serial.print(" Avg Impedance: ");
Serial.print(avgimp / 1000);
Serial.print("kOhm");
writeData(CTRL_REG, (readData(CTRL_REG) & 0x07) | 0xA0); //Power down; writeData(CTRL_REG,0xA0);

void writeData(int addr, int data) {
    Wire.beginTransmission(SLAVE_ADDR);
    Wire.write(addr);
    Wire.write(data);
    Wire.endTransmission();
    delay(1);
}

int readData(int addr) {
    int data;
    Wire.beginTransmission(SLAVE_ADDR);
    Wire.write(ADDR_PTR);
    Wire.write(addr);
    Wire.endTransmission();
    delay(1);
    Wire.requestFrom(SLAVE_ADDR, 1);
    if (Wire.available() >= 1) {
        data = Wire.read();
    } else {
        data = -1;
    }
    delay(1);
    return data;
}
boolean measureTemperature() {
    writeData(CTRL_REG, 0x90);
    delay(10);
    int flag = readData(STATUS_REG) & 1;    //Check status reg for temp measurement available
    if (flag == 1) {    // Temperature is available
        int temperatureData = readData(TEMP_R1) << 8;
        temperatureData |= readData(TEMP_R2);
        temperatureData &= 0x3FFF; // remove first two bits
        if (temperatureData & 0x2000 == 1) { // negative temperature
            temperatureData -= 0x4000;
        }
        double val = double(temperatureData) / 32;
        temperatureData /= 32;
        Serial.print("Temperature: ");
        Serial.print(val);
        Serial.println("C.");
        writeData(CTRL_REG, 0xA0);    // Power Down '10100000'
        return true;
    } else {
        return false;
    }
}

byte getFrequency(float freq, int n) {
    long val = long((freq / (MCLK / 4)) * pow(2, 27));
    byte code;
    switch (n) {
    case 1:
        code = (val & 0xFF0000) >> 0x10;
        break;
    case 2:
        code = (val & 0x00FF00) >> 0x08;
        break;
    case 3:
        code = (val & 0x0000FF);
        break;
    default:
        code = 0;
    }
    return code;
}
References


