Supporting Information for

3D printed and Microcontrolled: The one hundred dollars Scanning Electrochemical Microscope.

Gabriel N Meloni*

Instituto de Química

Universidade de São Paulo

São Paulo, SP, Brazil

Av. Profesor Lineu Prestes, 748

05508-000

*e-mail: gabriel.meloni@usp.br

*To whom correspondence should be addressed

Content

SI-1. 3D models

3D models files (.stl) for the linear stages, electrode holder and angle bracket (for attaching two axis perpendiculars to each other) are available free of charge via the Internet at http://pubs.acs.org.

SI-2. EC-SPM electronics construction

Although the electronic construction of the proposed EC-SPM setup could be made entirely using breadboards and jumper wires,[1] the device used to perform the experiments reported on the main manuscript was built using custom etched printed circuit boards (PCB). The PCB design was based on the Arduino Uno board and works as an "Arduino shield" making it easy to connect to an Arduino Uno board without the concern of connecting the pins wrong. Despite being more expensive than the breadboard construction, the PCB construction is sturdier and more reliable, increasing the quality of the experimental data acquired.

This type of construction allowed for the current range of the equipment to be quickly and easily changed by swapping a small PCB that is attached to the top of the potentiostat board. Those small PCBs, named "current shields" contain resistors R_5 and R_6 (figure 1, main manuscript) of the potentiostat circuit along with two other resistors (forming a voltage divider) responsible for signaling to the microcontroller board which current shield is set atop of the potentiostat. The relationship between R_5 and R_6 sets the current range of the equipment (Equation 1) while the other two resistors on the voltage divider set a specific voltage output that the microcontroller can read and identify the current shield used and change the current range on the software.

$$\boldsymbol{V_{CVC} = (i_{in} + \frac{V_{b2}}{R_5}) \times R_6} \qquad (1)$$

To calculate the resistors value one should note that the Arduino Uno does not reads negative potentials, hence a bias potential (V_{b2}) should be applied Through R_5 in

order to offset the current reading so the maximum values of current, positive and negative, lay between 0V and 5V. Figure S1 shows the equipment built using the custom etched PCB and the current shields used to set the current range.

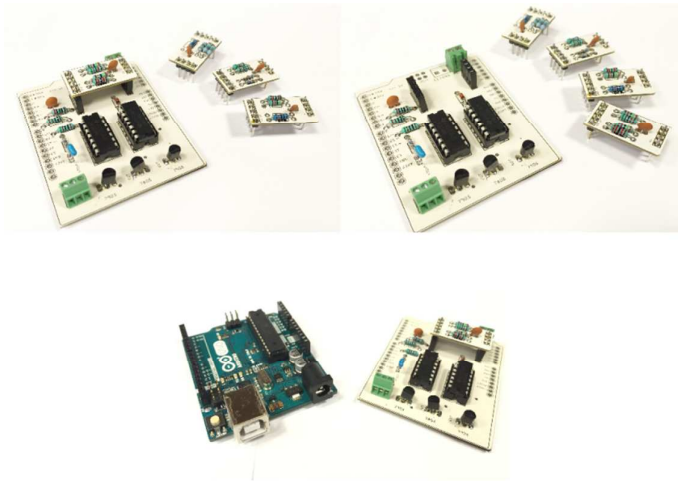

Figure S1. *Multiple pictures of the proposed potentiostat design fabricated using a custom etched PCB showing the current shields (small PCBs) that allow for fast and easy change of the current limits.*

The Arduino custom written sketch (Section SI-5) recognizes the current shield being used and sets the appropriate current range so the correct current values can be recorded. The recognition of the shield comes from reading a voltage, produced by a voltage divider in the current shield, which should be unique for each current limit. This voltage divider is formed by two resistors, labeled *5V* and *GND* (for ground) on the current shield (figure S2), and for the sketch presented on section SI-5, the values of those resistor together with resistor R_5 and R_6, (named *Bias* and *FB* respectively on the current shield, Figure S2) and the current limits can be seen on Table S1.

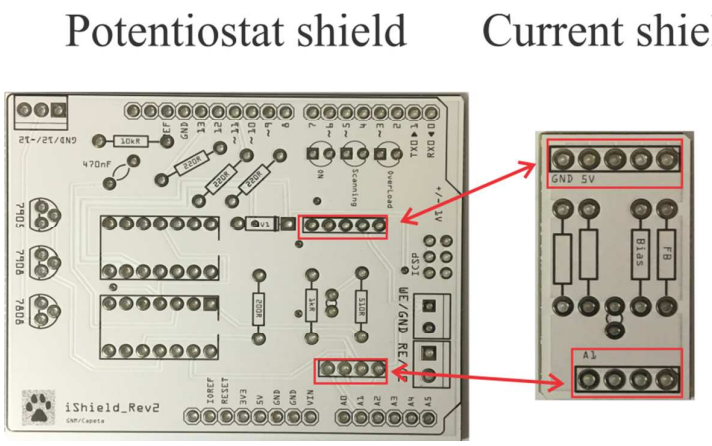

Figure S2. Photos of the potentiostat shield and current shield PCBs with component labels without any component solder. Red rectangles correspond to the connection site between the two shields.

Table S1. *Resistance values for R_5, R_6, and voltage divider resistors and the resulting voltage divider voltage for different current limits. Values are the expected on the Arduino sketch on Section SI-5.*

Current Limit (µA)	FB resistance/ R_6 (Ω)	Bias resistance/ R_5 (Ω)	R to GND (Ω)	R to 5V (Ω)	Voltage (V)	10bit ADC
200	*12k*	*24k*	*10k*	*0k*	*0*	*0*
100	*24k*	*50k*	*24k*	*12k*	*1,6667*	*341*
50	*50k*	*100k*	*10k*	*10k*	*2,5*	*512*
0,5	*5M*	*10M*	*0*	*10k*	*5*	*1023*

PCB layout files. Layout files (.fzz) for producing custom PCB potentiostat and current shield (Figure S2) are available free of charge via the Internet at http://pubs.acs.org. The layout files can be used for production or open modified on a PCB design software (Fritzing is a good free alternative). The layout and routing of the tracks were made in a "functional" manner and are not optimum, use of the layouts should be made at the readers own discretion.

SI-3 Potentiostat specifications

The potentiostat used to perform all experiments presented on the main manuscript was custom design and built according to experimental needs. Limits of operation (current limits, voltage compliance and maximum scan rate) and component values follow. More detailed information can be found elsewhere. [1]

Table S2. *Current, voltage and scan rate limits for the potentiostat used during the experiments performed in the main manuscript.*

	Min.	**Max.**
Potential (V)	-1	1
Current (µA)	-0.5	0.5
Voltage compliance (V)	-5	5
Scan rate (Vs^{-1})		0.780

Table S3*. Component values used during the electrochemical experiments on the main manuscript. Components are labeled as on Figure 1.*

Component label on Figure S1	***Component Value***
R1	*10k*Ω
R2	*510*Ω
R3	*1k*Ω
R4	*200*Ω
R5	*10M*Ω
R6	*5M*Ω
C1	*470nF*
C2	*100nF*

SI-4. EC-SPM mechanical construction

The mechanical part of the proposed EC-SPM is comprised of the linear stages used in each one of the axis of movement which are mostly 3D printed. Figure S3 shows different views of the fully assembled linear stages used during the experiments reported on the main manuscript. 3D files (.stl) for the linear stages and electrode holder can be found on section SI-1.

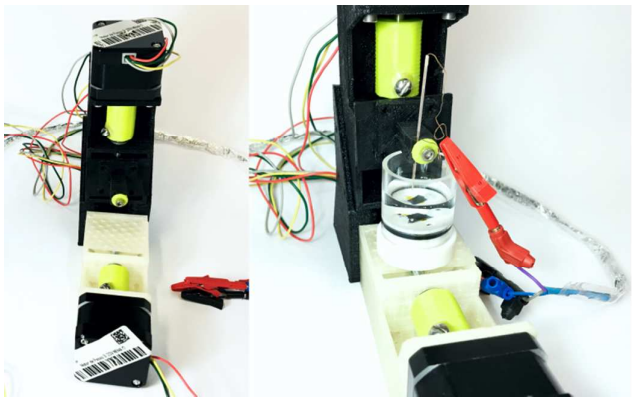

Figure S3. *Pictures of two 3D printed linear stages fully assembled showing the placement of the electrochemical cell and the 3D printed UME holder.*

SI-5. Arduino Sketch

Basic Arduino script used to run electrochemical experiments. The latest Arduino IDE can be downloaded from: https://www.arduino.cc/en/Main/Software.

The script presented below was used to perform the line scans reported on the main manuscript during the SECM experiments and is intended to be an example of the capabilities of the proposed design. It can be easily changed to perform multiple techniques and should be adapted according to the users need. The script relies on the Arduino serial communication port to receive and transmit data to a computer running the latest Arduino IDE. Using the serial port all parameters needed to perform a line scan can be defined, such as: Axis of movement, Direction of movement, Total displacement of the tip, Step width, Scan rate and SECM tip potential. The current shield used (Section SI-2) is automatic detected by the program and the necessary

parameters are automatic set accordingly. Data outputs as tab separated value on the serial port.

```
//Script starts

int LEDon = 12; // LED connected to pin 12 to indicate the equipment is on
int LEDscan = 11; // LED connected to pin 11 to indicate when a scan is being performed
int cRangePin = A1; // Pin to read the voltage divider on the Current Shield
int cRange = 0; //
char* range[]={}; // Current shield designation
float A = 0; // slop current conversion (Current shields)
float B = 0; // intersect current conversion (Current shields)
float C = 0; // decimal places, current conversion (Current shields)
int comando = 0; // Indicator for starting the scan or returning on the serial setup menu
const int startScan = {1};// type 1 on the serial monitor to start the scan
const int Return ={0}; // type zero on serial monitor to go back to the setup
int dac = 10; // pin connected to the DAC
int val = 0; // variable for counting microsteps performed
float ct = A0; // current-to-voltage converter/ADC pin
float c = 0;  // Variable to store de current value
int n = 0; // Stores the count of microsteps performed
float Potstep = 0.007812; // V/PWMstep
int intervalos = 0; // Variable to store the Interval between microsteps
int AA = 0; // Indicator for the serial setup menu
int BB = 0; // Indicator for the serial setup menu
int CC = 0; // Indicator for stopping the scan

/*Variables to be set using the serial port*/

float Eapp = 0; //Tip potential
float dD = 0; // Total electrode displacement of the electrode
float Step = 0; // Electrode step during the scan
float Speed = 0; // Electrode displacement speed
int Direction = 0; // Variable to store the travel direction up=1/donw=2
int MicroperStep = 0; // Variable to store the numbers of microsteps per step (point) of the line scan
int MicroperPac = 0; // Variable to store the numbers of microsteps on the total displacement
int stopButton = 13; // Push switch connected to pin 13 to stop scan (connected pull-down resistor)
int axis = 0; // Variable to store the axis of movement. Axis should be assign to numbers (eg. 1 = X axis, 2 = Y axis, 3 = Z axis)

/*STEPPER AND STAGE SETUP*/
float pitch = 500; // Thread pitch of the stage in micrometers
float microStep = 8; // number of microsteps on the stepper
float stepRev = 200; // steps per revolution on the stepper
int dirpinz = 2; // pin that gives the direction of movement of Z to the EasyDrive controller
int steppinz = 3; // pin that toggles to signal a microstep on Z to the EasyDrive controller
int dirpinx = 4; //pin that gives the direction of movement of X to the EasyDrive controller
int steppinx = 5; // pin that toggles to signal a microstep on X to the EasyDrive controller
int dirpiny = 6; //pin that gives the direction of movement of Y to the EasyDrive controller
int steppiny = 7; // pin that toggles to signal a microstep on Y to the EasyDrive controller

void setup() {

 TCCR1B = TCCR1B & B11111000 | B00000001;
 Serial.begin(9600);
 pinMode(dac,OUTPUT);
```

```
  pinMode(ct,INPUT);
  pinMode(cRangePin, INPUT);
  pinMode(LEDon,OUTPUT);
  pinMode(LEDscan,OUTPUT);
  pinMode(dirpinx, OUTPUT);
  pinMode(steppinx,OUTPUT);
  pinMode(dirpiny, OUTPUT);
  pinMode(steppiny,OUTPUT);
  pinMode(dirpinz, OUTPUT);
  pinMode(steppinz,OUTPUT);
  delay(100);
  digitalWrite(LEDon,HIGH);


}

void loop() {
   digitalWrite(LEDscan,LOW);

/*Serial port setup menu*/
    AA = 0;
    while(AA==0){
      BB = 0;
      while(BB==0){

    Serial.println("");
    Serial.println("Axis (x,y,z) X=1, Y=2, Z=3");
    while(!Serial.available());
    axis=Serial.parseInt();
    Serial.print("Axis");
    Serial.print(axis); Serial.println("(X=1, Y=2, Z=3)");

    Serial.println("");
    Serial.println("Potencial?(mV)");
    while(!Serial.available());
    Eapp=Serial.parseInt();
    Serial.print("E=");
    Serial.print(Eapp/1000,3); Serial.println("V");

    Serial.println("Direction? (UP=1/DOWN=2)");
    while(!Serial.available());
    Direction=Serial.parseInt();
    Serial.print("Dir=");
    Serial.print(Direction); Serial.println("(UP=1/DOWN=2)");

    Serial.println("Total displacement?(um)");
    while(!Serial.available());
    dD=Serial.parseInt();
    Serial.print("d=");
    Serial.print(dD,3); Serial.println("um");

    Serial.println("Step?");
    while(!Serial.available());
    Step=Serial.parseInt();
    Serial.print("Step=");
    Serial.print(Step); Serial.println("um");
```

```
   Serial.println("Speed?(um/s)");
   while(!Serial.available());
   Speed=Serial.parseInt();
   Serial.print("Speed=");
   Serial.print(Speed); Serial.println("um/s");

   Serial.println("Start? (Y=1/N=0)");
   while(!Serial.available());
   comando=Serial.parseInt();
   delay(100);

   if(comando==startScan){
   AA = 1;
   BB = 1;
  }
  if(comando==Return){
    AA = 0;
    BB = 1;
  }
 }
}

      MicroperStep = (Step/(pitch/(microStep*stepRev))); // Number of microsteps before a data point
is acquired.
      MicroperPac = ((dD/Step));
      intervalos =  1000*((pitch/(microStep*stepRev))/Speed); // interval between each microstep


      int Ei = ((-127.5*(-Eapp/1000))+127.5); // tip potential converted to a DAC value (between 0-255)


       //Check for different current shields and set the current scale
       cRange = analogRead(cRangePin);
        if(cRange < 100) { //200uA Range
          A = 0.391; //slop current conversion
          B = -200; // intersect current conversion
          C = 1; // decimal places, current resolution
          range[0]="Curent range:200uA";
        }
        if(cRange >= 300 && cRange <= 400){ //100uA Range
          A = 0.1955;
          B = -100;
          C = 1;
          range[0]="Curent range:100uA";
        }
        if(cRange >= 450 && cRange <= 560){ //50uA Range
          A = 0.0978;
          B = -50;
          C = 2;
          range[0]="Curent range:50uA";
        }
        if(cRange > 900){//500nA Range
          A = 0.000978;
          B = -0.5;
          C = 3;
          range[0]="Curent range:500nA";
        }
```

```
/*Print the parameters set on the serial setup menu*/
        Serial.println("");
        Serial.print("E="); Serial.print((Eapp/1000),3); Serial.println("V");
        Serial.print("T.Disp="); Serial.print((dD),3); Serial.println("um");
        Serial.print("Step="); Serial.print(Step,3); Serial.println("um");
        Serial.print("Speed="); Serial.print(Speed,3);Serial.println("um/s");
        Serial.print("Axis="); Serial.print(axis); Serial.println(" (X=1, Y=2, Z=3)");
       Serial.println(range[0]);
       digitalWrite(LEDscan,HIGH);
       delay(100);

 /*Line scan starts here*/
 if(axis == 1){
 if(Direction == 1){
  Serial.print("Direction=");Serial.println("Up");
  Serial.println("d/um  I/uA ");
  digitalWrite(dirpinx,LOW);
  analogWrite(dac,Ei);

  n = 0;
  CC = 0;

   while(n != MicroperPac){
    if(CC == 1){
      break;
    }

 for(val = 0; val < MicroperStep; val++){
   digitalWrite(steppinx,LOW);
   delayMicroseconds(100);
   digitalWrite(steppinx,HIGH);
   delay(intervalos);
   if (digitalRead(stopButton)==HIGH){
     CC = 1;
     break;
   }
 }
 delay(50);
  c = ((A*analogRead(ct))+B); // outputs in uA!!!
   Serial.println(n);
   Serial.print("  ");
   Serial.print(c,C);
   n=n+1;
 }
}

if(Direction == 2){
   Serial.print("Direction=");Serial.println("Down");
   Serial.println("d/um  I/uA  n  V/s");
   digitalWrite(dirpinx,HIGH);
   analogWrite(dac,Ei);

  n = 0;
  CC = 0;

   while(n != MicroperPac){
    if(CC == 1){
```

```
      break;
    }

 for(val = 0; val < MicroperStep; val++){
   digitalWrite(steppinx,LOW);
   delayMicroseconds(100);
   digitalWrite(steppinx,HIGH);
   delay(intervalos);
   if (digitalRead(stopButton)==HIGH){
     CC = 1;
     break;
   }
 }
 delay(50);
 c = ((A*analogRead(ct))+B); // outputs in uA!!!
   Serial.println(n);
   Serial.print("  ");
   Serial.print(c,C);
   n=n+1;
 }
}
if(CC==1){
  AA = 0;}
 }


 if(axis == 2){
  if(Direction == 1){
  Serial.print("Direction=");Serial.println("Up");
   Serial.println("d/um  I/uA  n  V/s");
   digitalWrite(dirpiny,LOW);
   analogWrite(dac,Ei);

  n = 0;
  CC = 0;

   while(n != MicroperPac){
    if(CC == 1){
      break;
    }

 for(val = 0; val < MicroperStep; val++){
   digitalWrite(steppiny,LOW);
   delayMicroseconds(100);
   digitalWrite(steppiny,HIGH);
   delay(intervalos);
   if (digitalRead(stopButton)==HIGH){
     CC = 1;
     break;
   }
 }
  delay(50);
  c = ((A*analogRead(ct))+B); // outputs in uA!!!
    Serial.println(n);
   Serial.print("  ");
   Serial.print(c,C);
    n=n+1;
```

```
 }
}

if(Direction == 2){
  Serial.print("Direction=");Serial.println("Down");
  Serial.println("d/um  I/uA  n  V/s");
  digitalWrite(dirpiny,HIGH);
  analogWrite(dac,Ei);

 n = 0;
 CC = 0;

  while(n != MicroperPac){
   if(CC == 1){
     break;
   }

 for(val = 0; val < MicroperStep; val++){
   Serial.print(val);
   digitalWrite(steppiny,LOW);
   delayMicroseconds(100);
   digitalWrite(steppiny,HIGH);
   delay(intervalos);
   if (digitalRead(stopButton)==HIGH){
    CC = 1;
    break;
   }
 }
 delay(50);
 c = ((A*analogRead(ct))+B); // outputs in uA!!!
   Serial.println(n);
   Serial.print("  ");
   Serial.print(c,C);
   n=n+1;
 }
}
if(CC==1){
 AA = 0;}
 }

 if(axis == 3){
 if(Direction == 1){
  Serial.print("Direction=");Serial.println("Up");
  Serial.println("d/um  I/uA  n  V/s");
  digitalWrite(dirpinz,LOW);
  analogWrite(dac,Ei);

 n = 0;
 CC = 0;

  while(n != MicroperPac){
   if(CC == 1){
     break;
   }

 for(val = 0; val < MicroperStep; val++){
   digitalWrite(steppinz,LOW);
```

```
   delayMicroseconds(100);
   digitalWrite(steppinz,HIGH);
   delay(intervalos);
   if (digitalRead(stopButton)==HIGH){
     CC = 1;
     break;
   }
 }
  delay(50);
  c = ((A*analogRead(ct))+B); // outputs in uA!!!
   Serial.println(n);
   Serial.print("  ");
   Serial.print(c,C);
   n=n+1;
 }
}

if(Direction == 2){
   Serial.print("Direction=");Serial.println("Down");
   Serial.println("d/um  I/uA  n  V/s");
   digitalWrite(dirpinz,HIGH);
   analogWrite(dac,Ei);

  n = 0;
  CC = 0;

   while(n != MicroperPac){
     if(CC == 1){
       break;
     }

 for(val = 0; val < MicroperStep; val++){
   digitalWrite(steppinz,LOW);
   delayMicroseconds(100);
   digitalWrite(steppinz,HIGH);
   delay(intervalos);
   if (digitalRead(stopButton)==HIGH){
     CC = 1;
     break;
   }
 }
 delay(50);
 c = ((A*analogRead(ct))+B); // outputs in uA!!!
   Serial.println(n);
   Serial.print("  ");
   Serial.print(c,C);
   n=n+1;
 }
}
if(CC==1){
  AA = 0;}
 }
}
//Script ends
```

SI-6. Bill of materials and cost

Although pricing the materials can be complicated as prices vary vastly between country, supplier and purchase amount, with most electronic compounds only sold in bulk, the total cost of the materials found on the bill of materials should fall under $100.00 USD for the equipment presented on the main manuscript (2 axes). This including the more expensive items such as the microcontroller board, stepper motors and controller board. The prices presented on Table S4 are for the unit price of each component and were priced at DigiKey (www.digikey.com). The price for the PLA printed parts was calculated by dividing the price of a 1 kg spool of filament by the weight of the one 3D printed stage, which, for the 3D printer infill settings used, was 50 g. The bill of materials and the pricing do not include a microcomputer or any chemicals, substrate or electrode, it only regard the parts and price for the proposed SEPM equipment.

Table S4. *Material list for the fabrication of 1 unit of the proposed EC-SPM equipment. Items marked with '*' represent items to be used only if not using an adjustable power supply.*

Component	***Quantity (Unities)***	***Unit price / USD*****	***DigiKey part number***
*LM79L05(Voltage regulator)**	*1*	*0.43*	*LM79L05ACZFS-ND*
*LM79L08(Voltage regulator)**	*1*	*0.41*	*497-7299-1-ND*
*LM78L08(Voltage regulator)**	*1*	*0.41*	*497-10099-1-ND*
5V1 Zener diode	*1*	*0.12*	*1N5231BFSCT-ND*
200Ω resistor	*1*	*0.1*	*S200CACT-ND*
510Ω resistor	*1*	*0.1*	*S510CACT-ND*
1kΩ resistor	*1*	*0.1*	*RNF14FTD1K00CT-ND*

10kΩ resistor	*1*	*0.1*	*RNF14FTD10K0CT-ND*
10MΩ resistor	*1*	*0.19*	*RNF14FTD10M0CT-ND*
5MΩ resistor	*1*	*0.9*	*CMF5.00MHJ-ND*
100nF ceramic capacitor	*1*	*0.12*	*1276-6990-1-ND*
470nF ceramic capacitor	*1*	*0.49*	*399-4374-ND*
3 way screw terminal	*1*	*1.03*	*ED10562-ND*
2 way screw terminal	*2*	*1.08*	*A98333-ND*
LM324 (quad Op-Amp)	*2*	*0.39*	*296-9542-5-ND*
Arduino Uno board	*1*	*22.00*	*1050-1041-ND*
EasyDriver stepper driver	*1 per axis*	*14.95*	*1568-1108-ND*
Nema 17 stepper motor	*1 per axis*	*14.95*	*1568-1105-ND*
M5 threaded rod	*1 per axis*	*1.3/ 30 cm*	*280-385****
M5 nut	*3 per axis*	*0.04*	*280-385****
3D printed parts ***(printed out of PLA)***		*28.55 / kg*	*1738-1173-ND*
Motor coupler	*1 per axis (50g of PLA)*	*1.43*	
Stage moving platform			
Stage body			
Total (2 axes equipment)		***93.64***	

*** Prices from July, 2017.*

****RS (http://www.rs-components.com) catalog number.*

References

1. G. N. Meloni, *J. Chem. Educ.*, 2016, **93**, 1320–1322.