

Arduino para medir la humedad de la tierra de una maceta y mostrar el resultado en un display LCD.

Arduino recibe un voltaje proporcional a la humedad en una maceta. Para ello utiliza dos clavos como electrodos insertados en la tierra. Idealmente muy poco espaciados entre ellos y lo más cerca posible de la raíz. En el caso de macetas pequeñas puede ser prácticamente en cualquier lugar.

El problema con tener electrodos en un medio húmedo es que la electrólisis los corroe fácilmente, así que se recomienda que sean galvanizados (y además *hot-dipped* que desconozco cuál sea el equivalente en español). Otra sugerencia es alambre de acero inoxidable.

El circuito para acondicionar la señal de los clavos es el de www.botanicalls.com/twitter (ver figura 1).

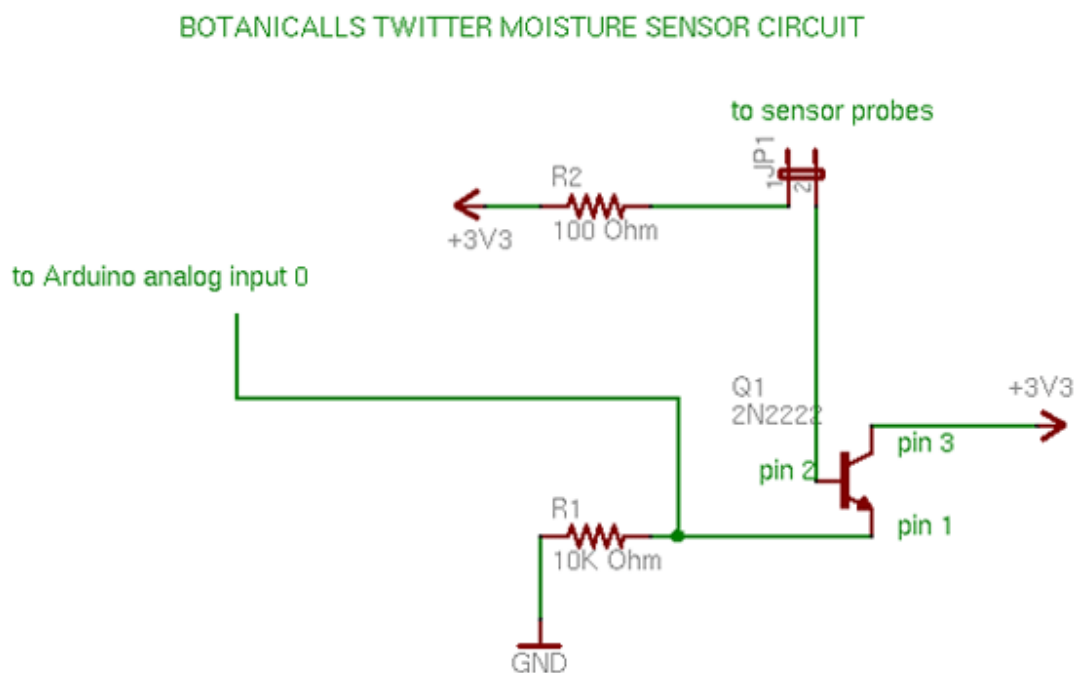


Fig. 1. Circuito para acondicionar la señal de los electrodos.

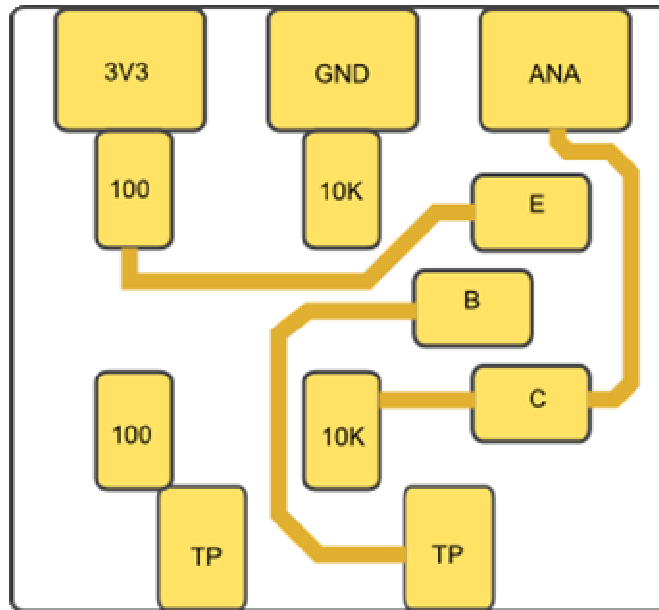


Fig. 2. Un posible diseño para un PCB sencillo de montaje superficial.

Ese voltaje proporcional es leído y comparado contra algunos niveles (definidos en el código) para mostrar un mensaje adecuado en el display LCD.

El display LCD se debe conectar al Arduino como se muestra en la tabla 1. Es preferencia personal usar la librería de 4 bits. También podría implementarse utilizando las 8 líneas de datos pero es un desperdicio.

Tabla 1. Conexiones Display + Arduino

Display	Arduino
1 – GND	
2 – VCC	
3 – Contraste	
4 – RS	12
5 – R/W	GND
6 - Enable	2
7 – NC	
8 – NC	
9 – NC	
10 – NC	
11	7
12	8
13	9
14	10

Para probarlo puede utilizarse el siguiente código:

```

#include <LCD4Bit.h>
//create object to control an LCD.
//number of lines in display=1
LCD4Bit lcd = LCD4Bit(2);

//some messages to display on the LCD
char msgs[6][15] = {"regar", "me ahogo", "tengo sed!",
"fertilizar", "Agua!", "Tierra = GND"};
int NUM_MSGS = 6;

void setup() {
  pinMode(13, OUTPUT); //we'll use the debug LED to output a
heartbeat

  lcd.init();
  //optionally, now set up our application-specific display
settings, overriding whatever the lcd did in lcd.init()
  //lcd.commandWrite(0x0F);//cursor on, display on, blink on.
(nasty!)
}

void loop() {
  digitalWrite(13, HIGH); //light the debug LED

  //pick a random message from the array
  int pick = random(NUM_MSGS);
  char* msg = msgs[pick];

  lcd.clear();
  lcd.printIn(msg);
  delay(1000);
  digitalWrite(13, LOW);

  //print some dots individually
  for (int i=0; i<3; i++){
    lcd.print('.');
    delay(100);
  }
  //print something on the display's second line.
  //uncomment this if your display HAS two lines!

  lcd.cursorTo(2, 0); //line=2, x=0.
  lcd.printIn("Agua: 73%");
  delay(1000);

  //scroll entire display 20 chars to left, delaying 50ms each inc
  lcd.leftScroll(20, 50);
}

```

Una vez funcionando correctamente el display puede hacerse pruebas de lectura del sensor acondicionado.

Para probar el sensor acondicionado.

Se sugiere usar processing para graficar el voltaje proveniente del sensor. El código para graficar ese voltaje es:

```
// Graph
// by David A. Mellis
//
// Demonstrates reading data from the Arduino board by graphing the
// values received.
//
// based on Analog In
// by <a href="http://itp.jtnimoy.com">Josh Nimoy</a>.

import processing.serial.*;

Serial port;
String buff = "";
int NEWLINE = 10;

// Store the last 64 values received so we can graph them.
int[] values = new int[64];

void setup()
{
  size(512, 256);

  println("Available serial ports:");
  println(Serial.list());

  // Uses the first port in this list (number 0). Change this to
  // select the port corresponding to your Arduino board. The last
  // parameter (e.g. 9600) is the speed of the communication. It
  // has to correspond to the value passed to Serial.begin() in
  your
  // Arduino sketch.
  port = new Serial(this, Serial.list()[0], 9600);

  // If you know the name of the port used by the Arduino board,
  you
  // can specify it directly like this.
  //port = new Serial(this, "COM1", 9600);
}

void draw()
{
  background(53);
  stroke(255);

  // Graph the stored values by drawing a lines between them.
  for (int i = 0; i < 63; i++)
    line(i * 8, 255 - values[i], (i + 1) * 8, 255 - values[i + 1]);

  while (port.available() > 0)
```

```

    serialEvent(port.read());
}

void serialEvent(int serial)
{
  if (serial != NEWLINE) {
    // Store all the characters on the line.
    buff += char(serial);
  } else {
    // The end of each line is marked by two characters, a carriage
    // return and a newline. We're here because we've gotten a
    newline,
    // but we still need to strip off the carriage return.
    buff = buff.substring(0, buff.length()-1);

    // Parse the String into an integer. We divide by 4 because
    // analog inputs go from 0 to 1023 while colors in Processing
    // only go from 0 to 255.
    int val = Integer.parseInt(buff)/4;

    // Clear the value of "buff"
    buff = "";

    // Shift over the existing values to make room for the new one.
    for (int i = 0; i < 63; i++)
      values[i] = values[i + 1];

    // Add the received value to the array.
    values[63] = val;
  }
}
}

```

Con ese código podemos observar una gráfica de la salida del sensor como se muestra a continuación:

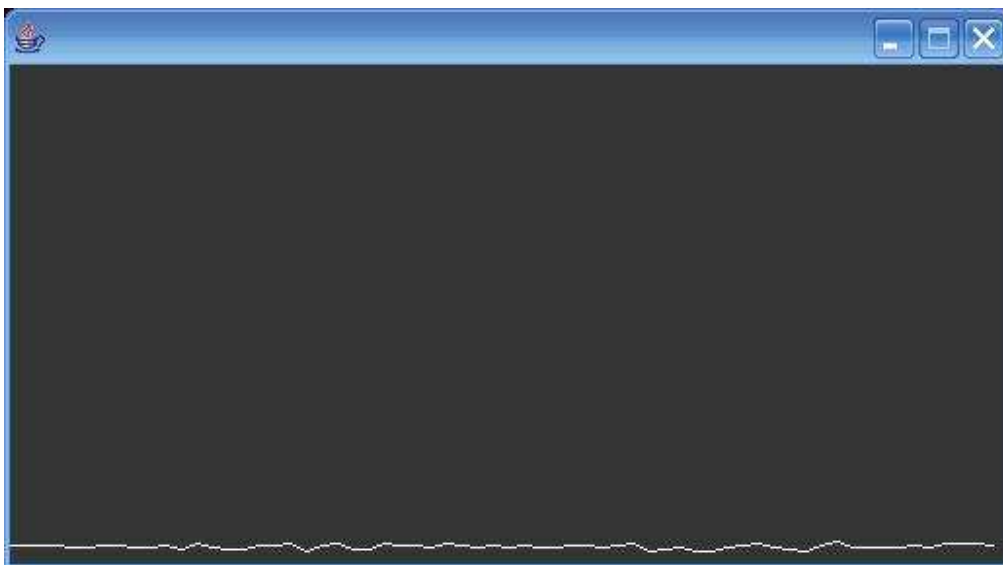


Fig. 3. Grafica de salida de sensor con Processing, en tierra prácticamente seca.