Instituto Tecnológico de Costa Rica

Escuela de Ingeniería en Computación

Programa de Bachillerato en Ingeniería en Computación

Curso: Fundamentos de Organización de Computadoras
Grupo 20

Proyecto 2

Estudiantes:
Valery Mishel Carvajal Oreamuno 2022314299
Marco Rodríguez Vargas 2022149445

Profesor:
Ing. Juan Manuel Sánchez Corrales

Fecha: Junio, 2022

Introducción

El avance continuo y acelerado de la tecnología y, por ende, de los dispositivos electrónicos ha llevado a la aparición de muchos nuevos software en nuestra vida cotidiana que buscan ayudarnos y facilitarnos diversos trabajos y así mejorar nuestras experiencias o, mejor dicho, vida diaria.

En este proyecto se busca crear un sistema, utilizando distintos hardware y un lenguaje de programación similar a C++ pero de menor nivel, en el que el usuario pueda adquirir dos variables ambientales distintas de manera sencilla y clara. Esto se piensa lograr mediante la programación de dos dispositivos Arduino que lea las variables mencionadas y las presente en una pantalla LCD.

Marco Teórico

Los circuitos, caminos cerrados y completos en los cuales puede circular una corriente eléctrica, estos caminos pueden incluir conductores y componentes eléctricos. Cada vez que se acciona un interruptor (se inicia o pone a funcionar) el circuito se completa, dejando así pasar a las corrientes eléctricas.

Por otro lado, un circuito eléctrico es un camino a través del cual fluye una corriente eléctrica. La trayectoria puede estar cerrada (unido en ambos extremos), lo que lo convierte en un bucle y hace posible el flujo de corriente eléctrica; o puede ser un circuito abierto donde el flujo de electrones se corta debido a que la trayectoria se rompe y esto no permite que la corriente eléctrica fluya. De manera sencilla, el circuito eléctrico es la conexión de dos o más componentes eléctricos o electrónicos que permiten generar, transportar y utilizar energía eléctrica y transformarla en otro tipo de energía como: calorífica, luminosa o mecánica. Todos los circuitos eléctricos tienen tres factores asociados con ellos: Corriente, Voltaje y Resistencia.

Esta clase de circuitos los vemos presentes en dispositivos Arduino. La principal característica de estos últimos es que son de código abierto por lo que es totalmente libre y fácil de usar, sin mencionar que es muy flexible ya que se pueden lograr muchas clases de circuitos. Además, incluye un software que utiliza un lenguaje similar a C++ que permite programarlo con gran facilidad.

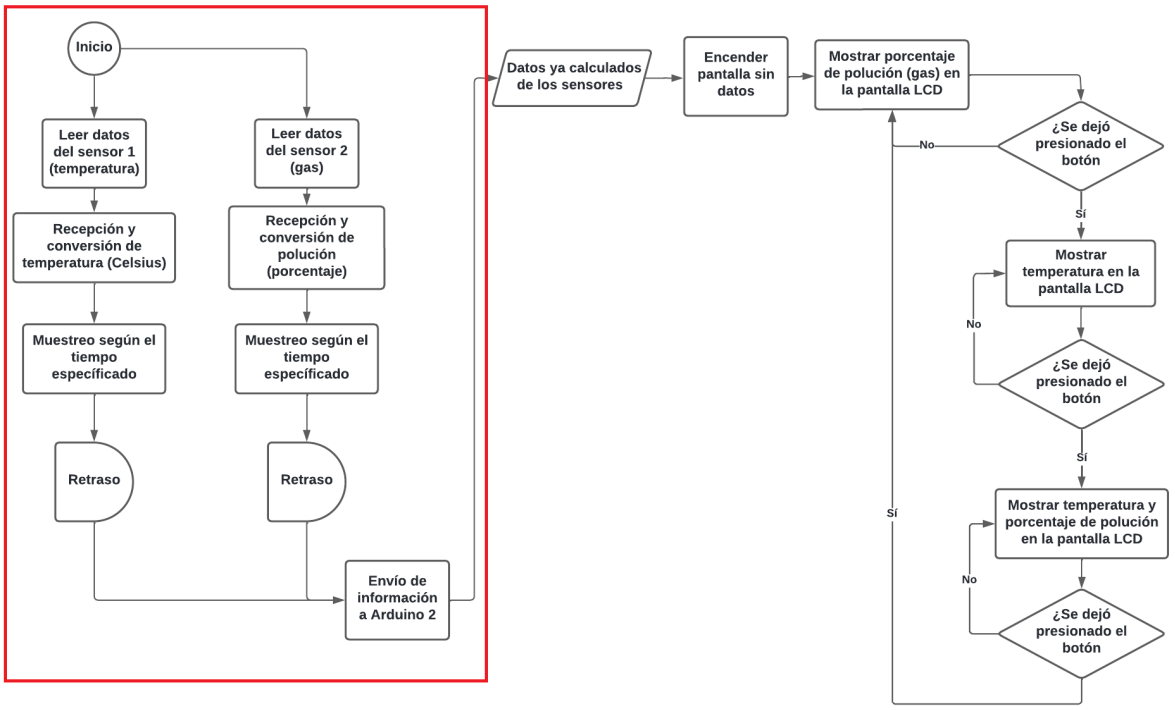
El mencionado (C++) es un lenguaje de programación de alto nivel orientado a objetos basado en C. Fue desarrollado en la década de los 80s e inicialmente llamado “C con clases”. Tiene todas las propiedades del lenguaje C y otras adicionales muy útiles que lo convierten en uno de los preferidos para crear aplicaciones con interfaz gráfica. A pesar de ser el lenguaje predilecto para la creación de juegos y aplicaciones; su mercado principal está en el desarrollo de sistemas operativos, máquinas virtuales, compiladores, navegadores y cualquier aplicación donde sea prioridad un alto rendimiento.

Como se mencionó anteriormente, para programar un Arduino, aunque es posible hacerlo otros, el lenguaje estándar es C++. Sin embargo, no es un C++ puro, sino una adaptación; haciéndolo así de un nivel más sencillo y permitiendo su uso a usuarios no tan conocedores o especializados en el lenguaje.

Conociendo lo que es un Arduino y cómo se trabaja en él, es importante saber que también se pueden utilizar distintos dispositivos adicionales como los sensores; existen variados tipos de sensores, entre estos los de temperatura, gas, humedad, entre otros.

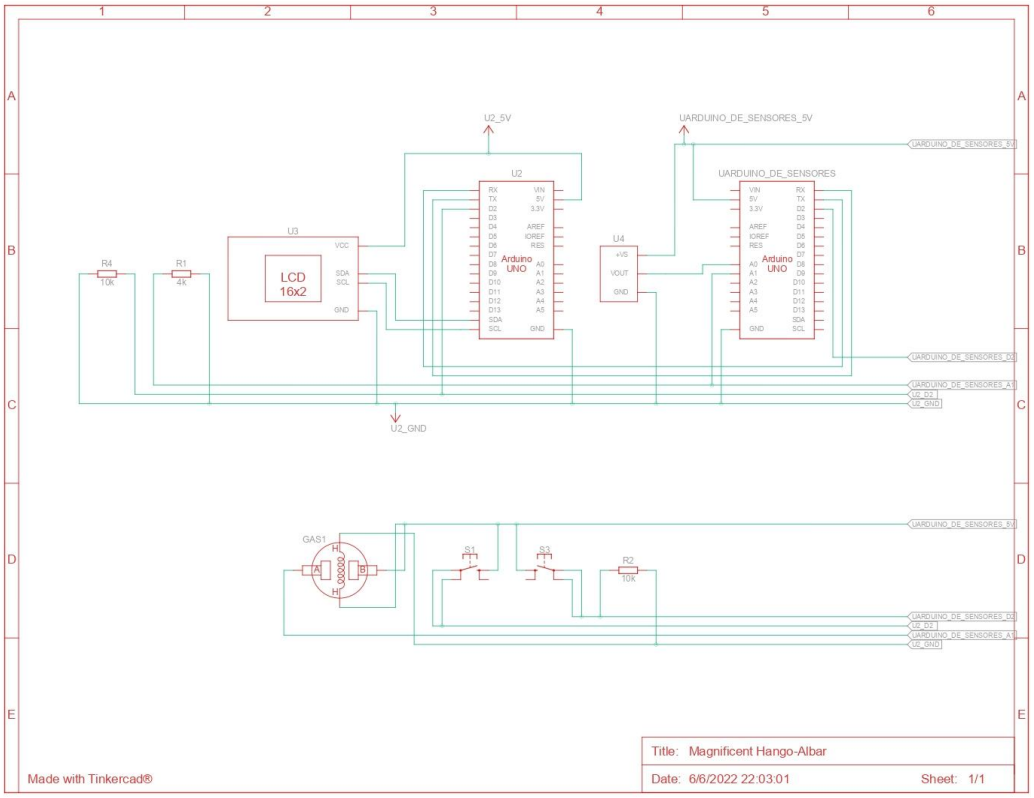
Por último, es beneficioso conocer dónde se pueden crear simulaciones de estos circuitos en caso de no tenerlo físicamente. Una opción es Tinkercad, que fue utilizado en este proyecto. Esta es una aplicación en línea de diseño e impresión 3D creada por la empresa Autodesk; tiene muchas ventajas, por ejemplo: es sencillo de comprender y su aspecto es atractivo. Como desventaja podríamos señalar que sólo posee una versión online, por lo que es necesario una conexión a internet.

Diseño de software (Diagrama de flujo) 1

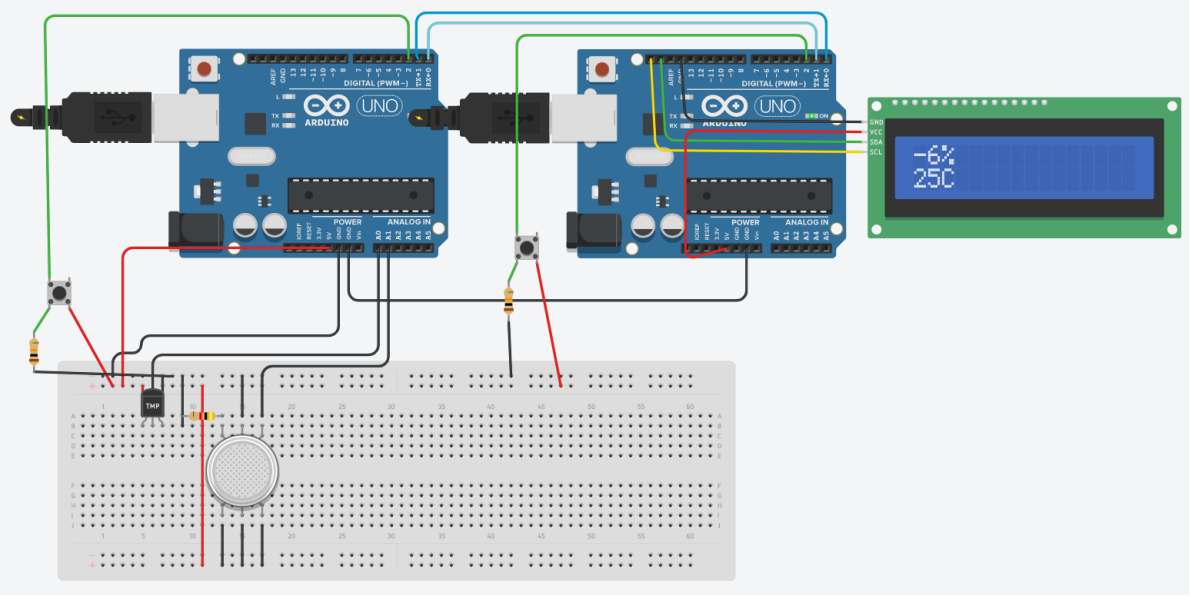


Diseño de hardware (Esquemático del circuito) 1

Name	Quantity	Component
UArduino de sensores U2	2	Arduino Uno R3
U3	1	MCP23008-based, 32 LCD 16 x 2 (I2C)
U4	1	Temperature Sensor [TMP36]
GAS1	1	Gas Sensor
R1	1	4 kΩ Resistor
R2 R4	2	10 kΩ Resistor
S3 S1	2	Pushbutton



Funcionamiento del circuito 1 en Tinkercad



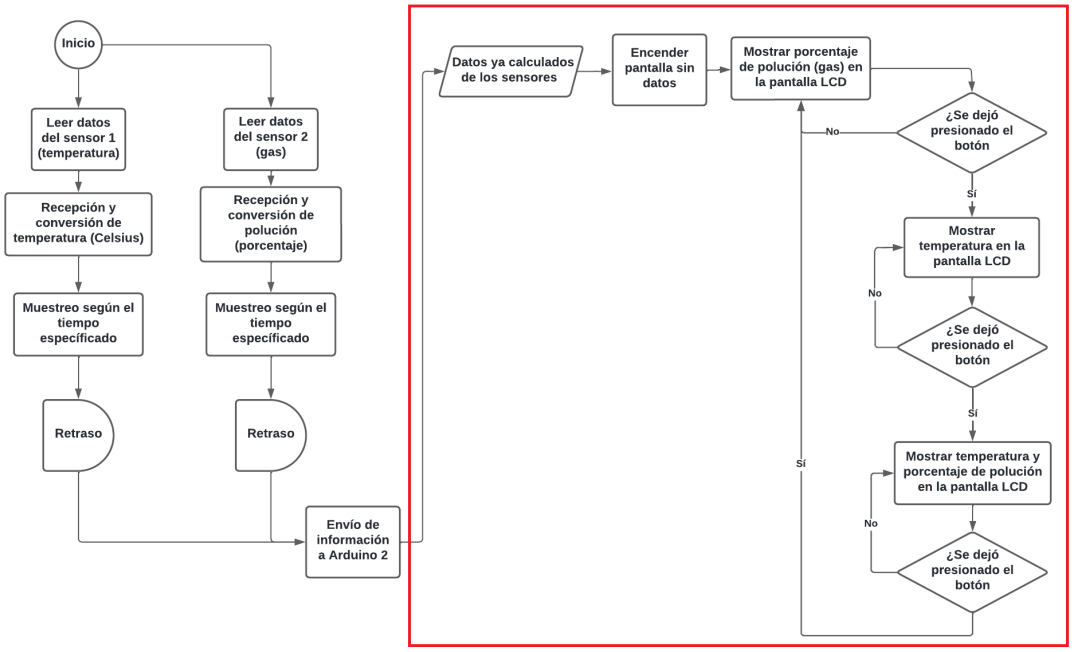
Al iniciarse la simulación,el arduino 1 lee los datos de cada sensor y hace la conversión necesaria para luego enviar estos datos al Arduino 2.

Elaboración del circuito 1

La simulación del circuito fue elaborada totalmente en Tinkercad, donde se añadieron elementos hardware, como: dos dispositivos Arduino, una pantalla LCD 16x2, un breadboard, botones, dos sensores (uno de temperatura y otro de gas) y el cableado correspondiente.

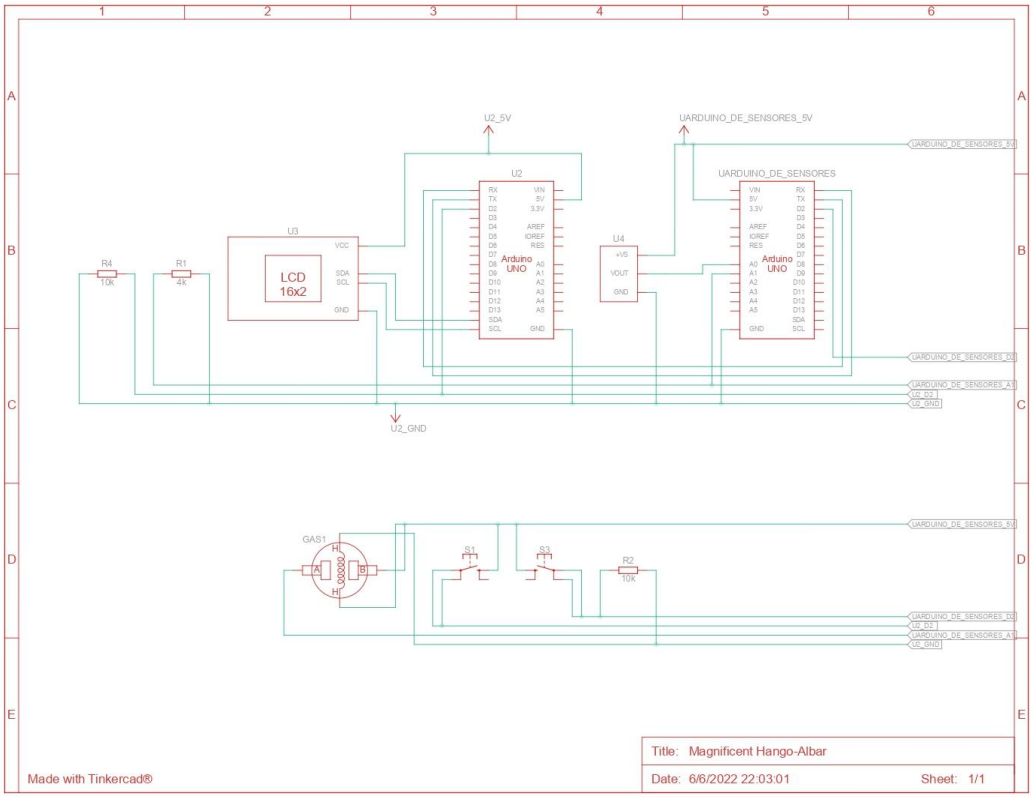
Con respecto al software tenemos la programación de cada Arduino que se muestra al final del documento como parte de los anexos.

Diseño de software (Diagrama de flujo) 2

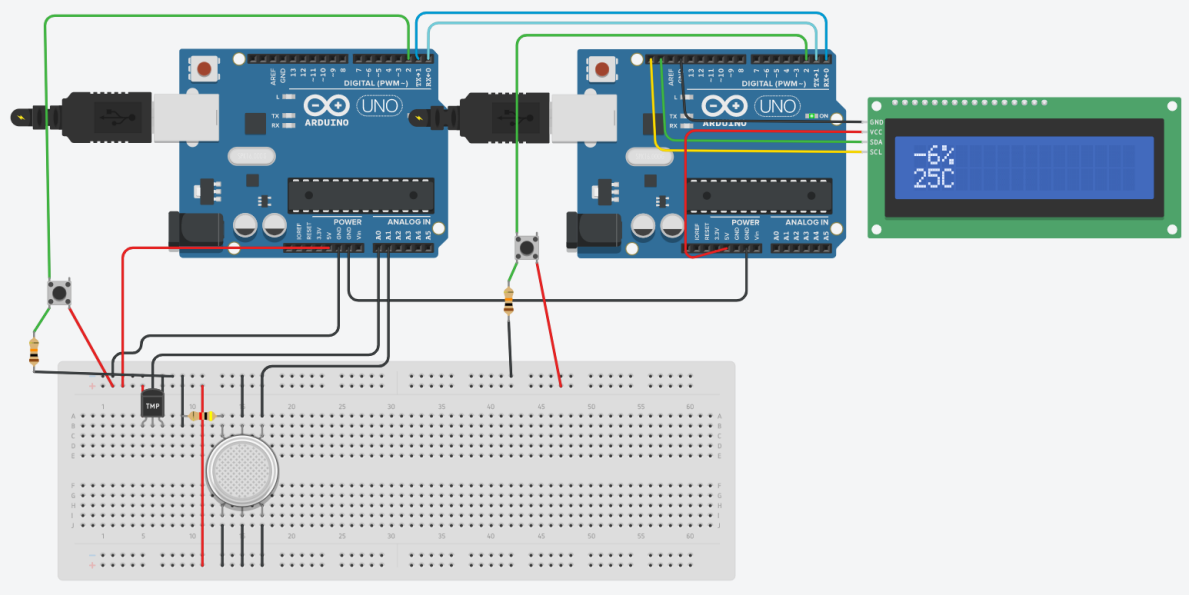


Diseño de hardware (Esquemático del circuito)

Name	Quantity	Component
UArduino de sensores U2	2	Arduino Uno R3
U3	1	MCP23008-based, 32 LCD 16 x 2 (I2C)
U4	1	Temperature Sensor [TMP36]
GAS1	1	Gas Sensor
R1	1	4 kΩ Resistor
R2 R4	2	10 kΩ Resistor
S3 S1	2	Pushbutton



Funcionamiento del circuito en Tinkercad



Al iniciarse la simulación, la pantalla se enciende y se muestra el porcentaje de polución (los datos del sensor de gas). Si se mantiene presionado el botón que encontramos por el lado izquierdo podemos configurar el tiempo para obtener el muestreo deseado; por otro lado, si se hace lo mismo con el del lado derecho (en medio de los arduinos) este nos permite cambiar cuál sensor deseamos visualizar en la pantalla LCD (gas, temperatura o ambos a la vez). Cabe mencionar que no basta con hacer click en los botones, si no, hay que mantenerlos presionados. También, al ser una simulación, se puede cambiar la temperatura

haciendo click en el sensor de temperatura y arrastrando lo que aparece hasta donde se desee; en el caso del sensor de gas, con hacer un click encima del sensor nos va a aparecer como una nube de gas que podemos arrastrar para visualizar los cambios.

Elaboración del circuito

La simulación del circuito fue elaborada totalmente en Tinkercad, donde se añadieron elementos hardware, como: dos dispositivos Arduino, una pantalla LCD 16x2, un breadboard, botones, dos sensores (uno de temperatura y otro de gas) y el cableado correspondiente.

Con respecto al software tenemos la programación de cada Arduino que se muestra al final del documento como parte de los anexos.

Bibliografía

H. Rashid, Muhammad. Editorial Pearson Educación, ed. Electrónica de potencia: circuitos, dispositivos y aplicaciones (2004 edición)

Peña Millahual, Claudio. Editorial RedUsers, ed. Descubriendo Arduino (2020 edición)

Stroustrup, Bjarne. AT&T Bell Laboratories, ed. An Overview of C++ (2018 edición)

Hernandez Muñoz, Andrea Yiseth. Universidad Antonio Nariño, Facultad de Educación, ed. Estrategias pedagógicas a través de la Plataforma Tinkercad para fortalecer las competencias tecnológicas por medio del funcionamiento de los circuitos básicos y diseño en 3D (2020 edición)

Anexos

Enlace al proyecto en Tinkercad
https://www.tinkercad.com/things/2nZG8VWivPq?sharecode=f1E989bUdxAKbdbog1GwnWvM6z7eALmmXlSJhFknQ4Y

Código de Fuente para cada Arduino

- Arduino 1 (Sensores)

```
// C++ code

String temp;

String gas;

int muestra = 1000;

void setup()

{

  Serial.begin(9600);

  pinMode(2, INPUT);

}


void loop()

{

  temp = String((int)map(((analogRead(A0) - 20) * 3.04), 0, 1023, -40, 125));                    //
Recepción y conversión de la temperatura (C°)

  gas = String((int)((analogRead(A1) - 300)/450.0*100));                                         //
Recepción y conversión de la polución (%)

  Serial.write((temp + "s" + gas + "e").c_str(), temp.length() + 2 + gas.length());

  if (digitalRead(2) == HIGH) {                    // Muestreo configurable

    if (muestra == 1000) {

          muestra = 5000;                          // De 5 segundos

    } else if (muestra == 5000) {

          muestra = 10000;                         // De 10 segundos

    } else if (muestra == 10000) {

          muestra = 20000;                         // De 20 segundos

    } else {

          muestra = 1000;                          // De 1 segundo
```

```
    }
  }
  delay(muestra);
}
```

- Arduino 2 (Para pantalla LCD)

```
// C++ code
// Importar librería LiquidCrystal
#include <Adafruit_LiquidCrystal.h>
Adafruit_LiquidCrystal lcd_1(0);  // Configuración del LCD

// Confiuración de los estados previos
String temp_prev;
String gas_prev;
int modo = 0;

void setup()
{
  Serial.begin(9600);
  lcd_1.begin(16, 2);
  pinMode(2, INPUT);
}

void mostrar() {
  lcd_1.clear();   // Limpiar pantalla
  // Mostrar gas
  if (modo == 0 || modo == 2) {
            lcd_1.setCursor(0, 0);
            lcd_1.print(gas_prev);
            lcd_1.print("%");
  }
```

```
  // Mostrar Temperatura
  if (modo == 1 || modo == 2) {
            lcd_1.setCursor(0, 1);
            lcd_1.print(temp_prev);
            lcd_1.print("C");
  }
  lcd_1.setBacklight(1);
}

void loop()
{
  char temp[5] = {};
  char gas[5] = {};
  // Leer "historial de bytes"
  char ver = 0;
  int cnt = 0;
  int resp = 0;
  if (digitalRead(2) == HIGH) {
    if (modo == 0) {
            modo = 1;
    } else if (modo == 1) {
      modo = 2;
    } else {
            modo = 0;
    }
    mostrar();
  }
  if (Serial.available() > 1) {          // Serial.read() siempre adelanta una posición.
            ver = char(Serial.read());
  while (Serial.available() && ver != 'e') {
    if (ver == 's') {
```

```
      Serial.println(resp);
      cnt++;
      resp = 0;
    } else if (cnt == 0) {
      temp[resp] = ver;
      resp++;
    } else {
      gas[resp] = ver;
      resp++;
    }
    ver = char(Serial.read());
  }
  // Se muestra el cambio a la pantalla y se cuardan los estado previos.
  if (String(gas) != gas_prev || String(temp) != temp_prev) {
    temp_prev = String(temp);
    gas_prev = String(gas);
    mostrar();
  }
  }
  delay(1000);      // Delay arbitrario
}
```