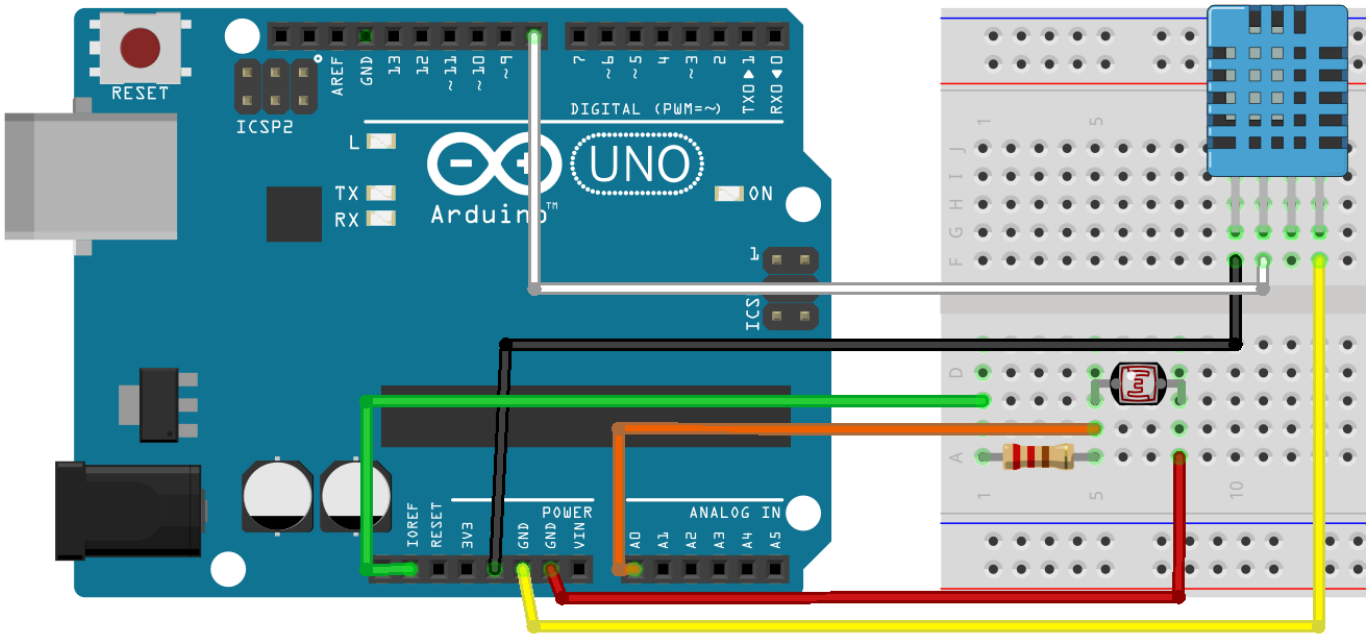
Artículo
Jose Tomas Salvador · Mayo 1, 2020 Lectura de 8 min

Estación meteorológica con Arduino

Era el momento del InterSystems hackathon y nuestro equipo, Artem Viznyuk y Eduard Lebedyuk tenian una placa Arduino (one) y varios componentes (de sobra). Así que su curso de acción estaba marcado - como todos los aprendices de Arduino, decidieron construir una estación meteorológica. Voy a aprovechar este artículo para seguir sus pasos y hacer lo mismo que ellos hicieron en 2016, pero con la persistencia de datos en IRIS y la visualización en Business Intelligence (antiguo DeepSee)! (Digamos que además de hacer la traducción del artículo original, aprovecho y nos actualizamos todos ;-))



Trabajar con dispositivos

InterSystems IRIS puede trabajar directamente con muchos tipos de dispositivos físicos y lógicos:

- Terminal
- TCP
- Al Spooler
- Impresoras
- Cinta magnética
- **Puertos COM**
- y muchos otros

Puesto que Arduino utilizar el puerto COM para las comunicaciones, tenemos todo lo que necesitamos. Generalmente, trabajar con dispositivos puede dividirse en 5 pasos:

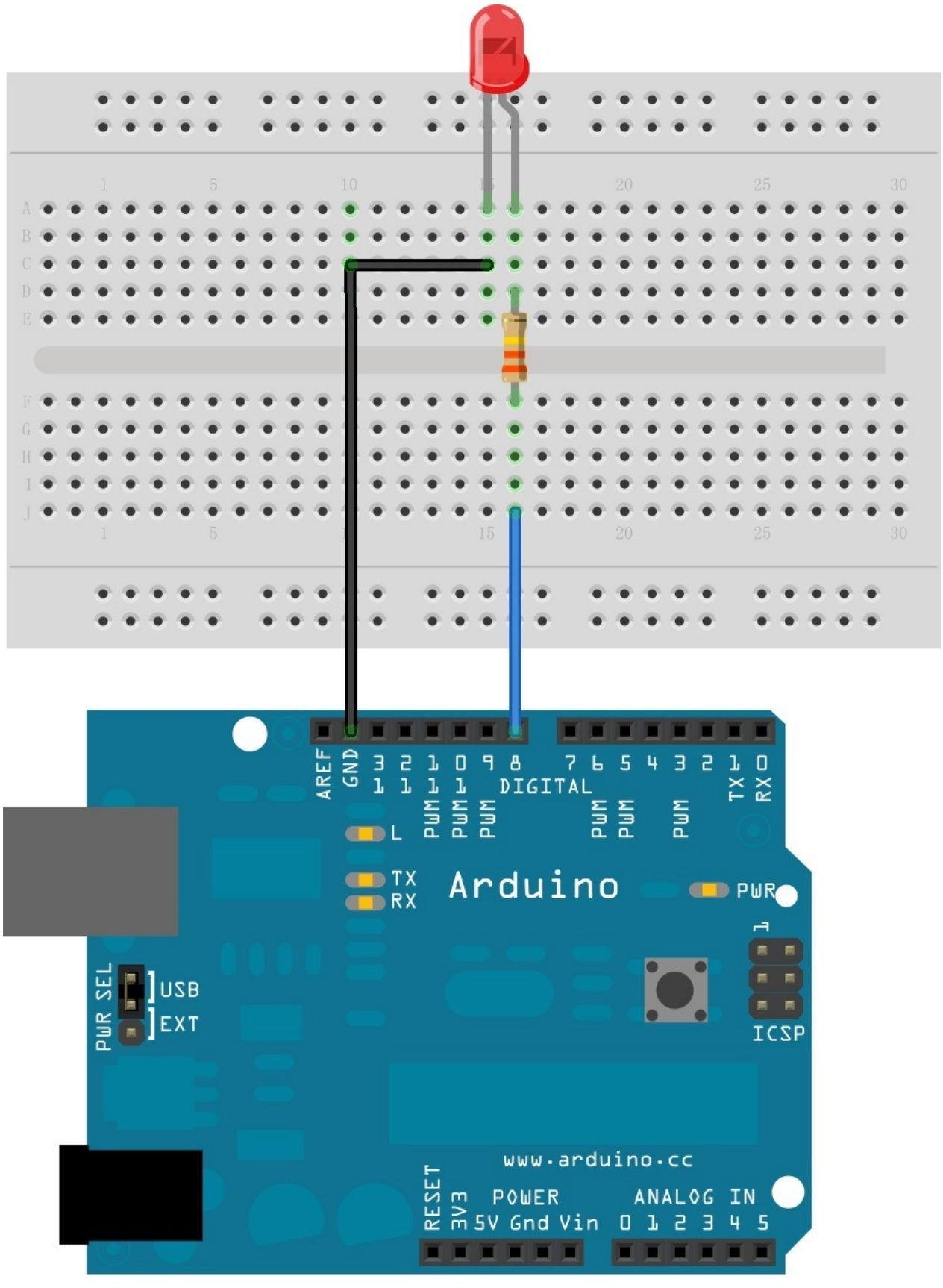
1. Comando OPEN para registrar el dispositivo con el proceso actual y acceder

2. Comando USE para hacer que sea el primario
3. Hacer algún trabajo. READ para recibir datos de un dispositivo, y WRITE para enviar datos
4. USE de nuevo para conmutar el dispositivo primario
5. Comando CLOSE para liberar el dispositivo

Bien, esa es la teoría, ¿ cómo es en la práctica?

Un parpadeo desde InterSystems IRIS

Primeramente, construimos un dispositivo con Arduino que lee un número del puerto COM y enciende un led durante unos milisegundos.
Circuito:



C code (for Arduino)

```
/* Led.ino
 * Receive data on a COM port
 * Connect your led to ledPin
```

```
 */
// Pin, to connect your led
#define ledpin 8
// Received data buffer
String inString = "";

// Execute once at the beginning
void setup() {
 Serial.begin(9600);
 pinMode(ledpin, OUTPUT);
 digitalWrite(ledpin, LOW);
}

// Execute indefinetly
void loop() {
 // Get data from com
 while (Serial.available() > 0) {
 int inChar = Serial.read();
 if (isDigit(inChar)) {
  // one character at a time
  // and append it to data buffer
  inString += (char)inChar;
 }

 // Encounter new line
 if (inChar == '\n') {
  // Power on the led
  digitalWrite(ledpin, HIGH);
        // be aware: toInt() converts a string that starts with a number to number...

  int time = inString.toInt();
  delay(time);
  digitalWrite(ledpin, LOW);
  // Flush the buffer
  inString = "";
 }
  }

}
```

Y por último un método en InterSystems IRIS que envía el string 1000 \ n (acabado en nueva línea) al puerto COM:

```
/// Send 1000\n to a com port
ClassMethod SendSerial()
{
 // Pon el puerto que te haya asignado el sistema a tu Arduino
    set port = "COM3"
 open port:(:::" 0801n0":/BAUD=9600)     // Open device
 set old = $IO // Record current primary device
 use port  // Switch to com port
 write $Char(10) // Send some test data
 hang 1
 write 1000 _ $Char(10) // Send 1000\n
 use old // Back to old terminal
```

```
 close port // Free the device
}
```

« 0801n0 » es una cadena con parámetros para acceder al puerto COM, tal y como se describe en la documentación (busca portstate cuando estés en la página). Y /BAUD=9600 es, por supuesto, la velocidad de conexión.

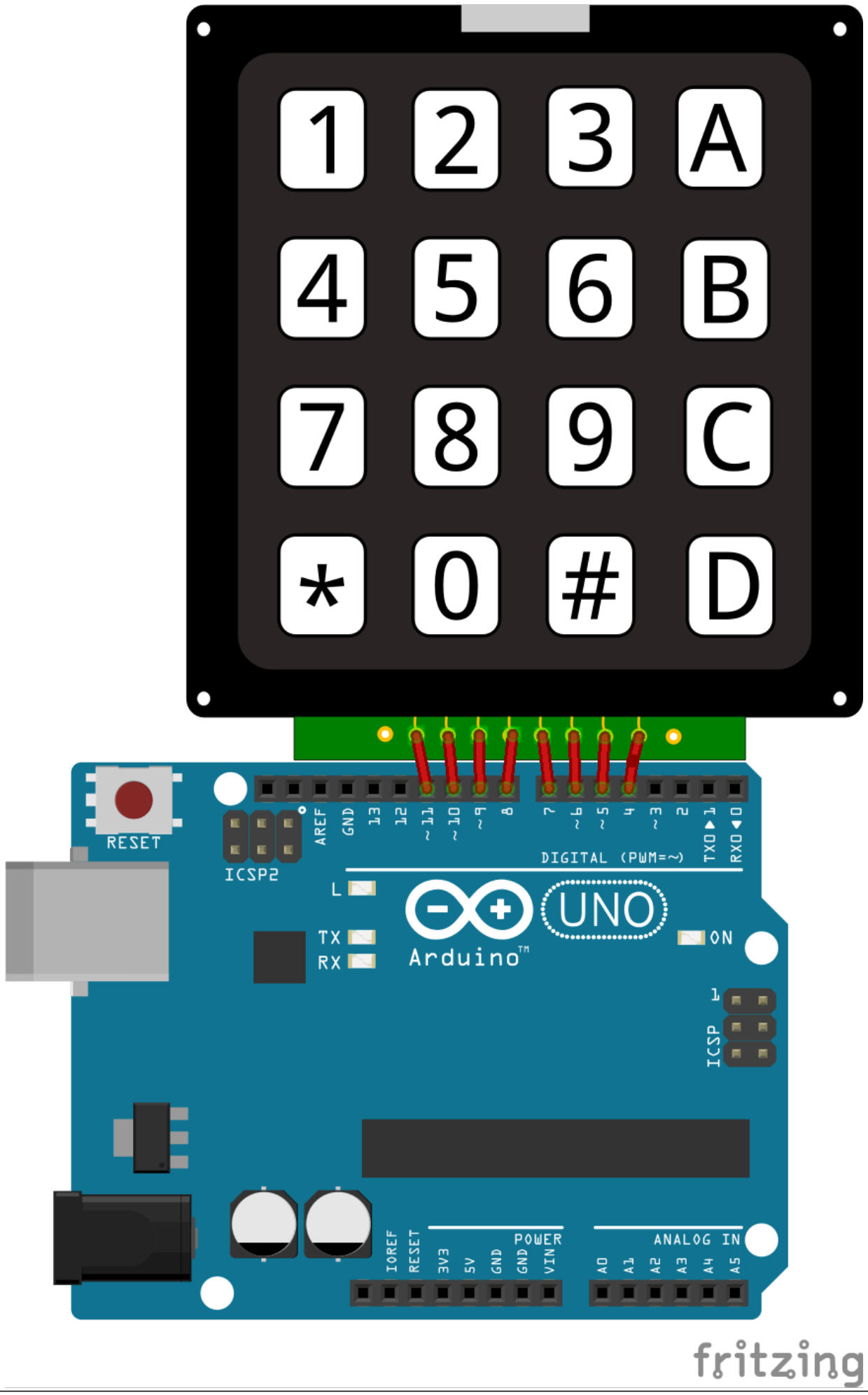
Si ejecutamos este método en un terminal:

```
do ##class(Arduino.Habr).SendSerial()
```

No devolverá nada, pero un led parpadeará por un segundo.

Recepción de datos

Ahora vamos a conectar un keypad a InterSystems IRIS y recibir datos de entrada. Esto podría utilizarse para autenticar a un usuario con autenticación delegada, implementando también la rutina ZAUTHENTICATE.mac.
Circuito:


1
2
3
A
4
5
6
B
7
8
9
C
*
0
#
D
RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM=~)
L
TX
RX
Arduino™
UNO
ON
ICSP
POWER
IOREF
RESET
3V3
5V
GND
GND
VIN
ANALOG IN
A0
A1
A2
A3
A4
A5
fritzing

C code

```
/* Keypadtest.ino *
 * Uses Keypad library,
 * Connect Keypad to Arduino pins
 * as specified in rowPins[] and colPins[].
 *
 */
// Repository:
// https://github.com/Chris--A/Keypad
#include <Keypad.h>
const byte ROWS = 4; // Four rows
const byte COLS = 4; // Three columns
// Map symbols to keys
char keys[ROWS][COLS] = {
 {'1','2','3','A'},
 {'4','5','6','B'},
 {'7','8','9','C'},
 {'*','0','#','D'}
};
// Connect keypad pins 1-8 (up-down) to Arduino pins 11-4: 1->11, 2->10, ... , 8->4
// Connect keypad ROW0, ROW1, ROW2 ? ROW3 to this Arduino pins
byte rowPins[ROWS] = { 7, 6, 5, 4 };

// Connect keypad COL0, COL1 and COL2 to this Arduino pins
byte colPins[COLS] = { 8, 9, 10, 11 };

// Keypad initialization
Keypad kpd = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );

void setup() {
 Serial.begin(9600);
}

void loop() {
 char key = kpd.getKey(); // Receive key pressed
 if(key)
 {
  switch (key)
 {
  case '#':
   Serial.println();
  default:
   Serial.print(key);
 }
 }
}
```

Y aquí tenemos un método en InterSystems IRIS utilizado para recoger los datos desde un puerto COM, una línea cada vez:

```
/// Receive one line till we encounter line terminator from COM1
ClassMethod ReceiveOneLine() As %String
{
 set port = "COM3"
 set str=""
 try {
  open port:(:::" 0801n0":/BAUD=9600)
  set old = $io
  use port
  read str // Read till we encounter line terminator
  use old
  close port
 } catch ex {
  close port
 }
 return str
}
```

Lo ejecutamos en un terminal:

```
write ##class(Arduino.Habr).ReceiveOneLine()
```

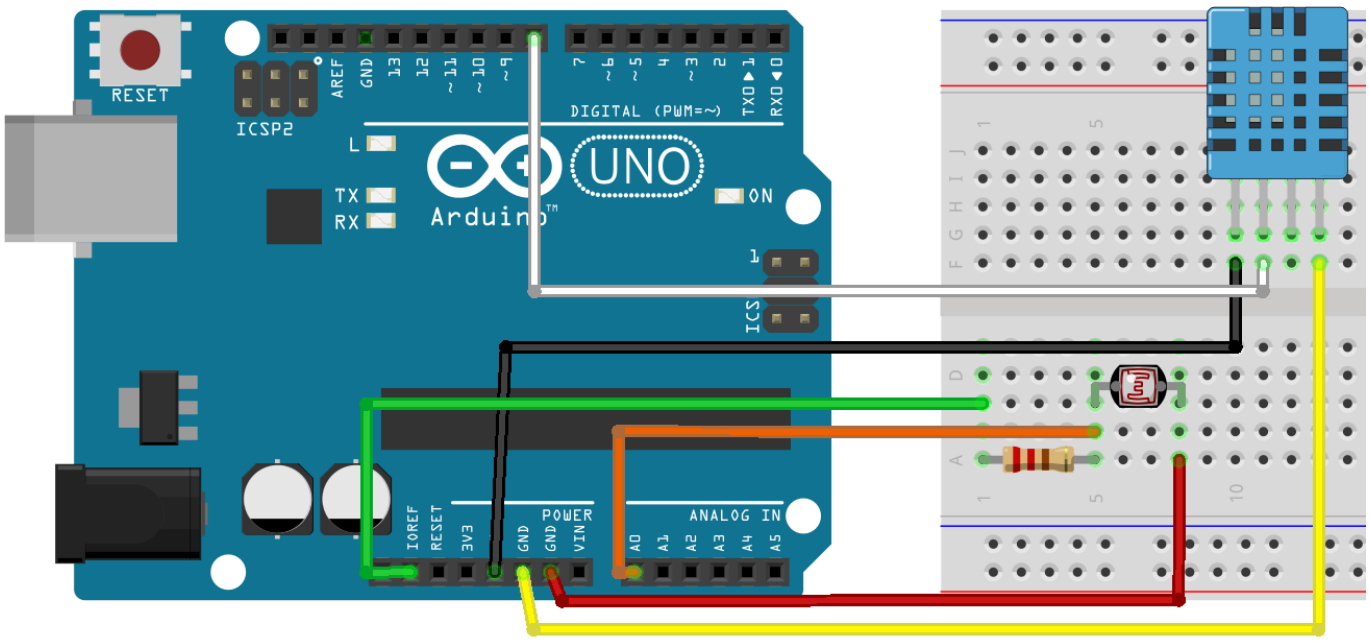
Y se quedará esperando recibiendo datos de entrada hasta que se presione # (que sería enviado como un terminador de línea), tras lo cual los datos introducidos se mostrarán en el terminal.

Esto era la I/O básica Arduino-IRIS, y ahora ya estamos preparados para construir nuestra propia estación meteorológica.

Estación Meteorológica

¡ Finalmente estamos llegando a la estación meteorológica! Utilizamos un fotoresistor y un sensor de Humedad y Temperatura DHT11 para obtener los datos.

Circuito:



C code

```
/* Meteo.ino *
 * Register humidity, temperature and light level
 * And send them to a COM port
 * Output sample: H=1.0;T=1.0;LL=1;
 */
// Photoresistor pin (analog)
int lightPin = 0;

// DHT-11 pin (digital)
int DHpin = 8;

// Array to store DHT-11 temporary data
byte dat[5];

void setup() {
 Serial.begin(9600);
 pinMode(DHpin,OUTPUT);
}

void loop() {
 delay(1000); // measure everything once per second
 int lightLevel = analogRead(lightPin); //Get brightness level
 temp_hum(); // Get temperature and humidity into dat variable
 // And output the result
 Serial.print("H=");
 Serial.print(dat[0], DEC);
 Serial.print('.');
 Serial.print(dat[1],DEC);
 Serial.print(";T=");
 Serial.print(dat[2], DEC);
 Serial.print('.');
 Serial.print(dat[3],DEC);
 Serial.print(";LL=");
 Serial.print(lightLevel);
 Serial.println(";");
}

// Get DHT-11 data into dat
void temp_hum() {
 digitalWrite(DHpin,LOW);
 delay(30);
 digitalWrite(DHpin,HIGH);
 delayMicroseconds(40);
 pinMode(DHpin,INPUT);
 while(digitalRead(DHpin) == HIGH);
 delayMicroseconds(80);
 if(digitalRead(DHpin) == LOW);
 delayMicroseconds(80);
 for(int i=0;i<4;i++)
 {
   dat[i] = read_data();
 }
 pinMode(DHpin,OUTPUT);
 digitalWrite(DHpin,HIGH);
}

// Get a chunk of data from DHT-11
byte read_data() {
 byte data;
```

```
 for(int i=0; i<8; i++)
 {
  if(digitalRead(DHpin) == LOW)
  {
   while(digitalRead(DHpin) == LOW);
   delayMicroseconds(30);
   if(digitalRead(DHpin) == HIGH)
   {
    data |= (1<<(7-i));
   }
   while(digitalRead(DHpin) == HIGH);
  }
 }
 return data;
}
```

Una vez que hemos cargado este código en Arduino, va a comenzar a enviar datos desde el puerto COM con el siguiente formato:

H=34.0;T=24.0;LL=605;

Donde:

- H — humedad (de 0 al 100 por ciento)
- T — temperatura (grados Celsius)
- LL — Luminosidad (de 0 a 1023)

Vamos a almacenar estos datos en InterSystems IRIS. Para eso, escribimos una nueva clase Arduino.Info:

```
Class Arduino.Info Extends %Persistent
{

/// Puerto COM asociado - cambialo según tu sistema
Parameter SerialPort As %String = "com3";
Property DateTime As %DateTime;
Property Temperature As %Double;
Property Humidity As %Double(MAXVAL = 100, MINVAL = 0);
Property Brightness As %Double(MAXVAL = 100, MINVAL = 0);
Property Volume As %Double(MAXVAL = 100, MINVAL = 0);
ClassMethod AddNew(Temperature = 0, Humidity = 0, Brightness = 0, Volume = 0)
{
 set obj = ..%New()
 set obj.DateTime=$ZDT($H,3,1)
 set obj.Temperature=Temperature
 set obj.Humidity=Humidity
 set obj.Brightness=Brightness/1023*100
 set obj.Volume=Volume
 write $SYSTEM.Status.DisplayError(obj.%Save())
}
```

Despues, escribimos un método que recibirá los datos desde Arduino y los transformará en objetos de tipo

Arduino.Info:

```
/// Receive a RAW data in this format: H=34.0;T=24.0;LL=605;\n
/// Convert into Arduino.Info objects
ClassMethod ReceiveSerial(port = {..#SerialPort})
{
    try {
        open port:(:::" 0801n0":/BAUD=9600)
        set old = $IO
        use port
        for {
            read x //read one line
            if (x '= "") {
                set Humidity = $Piece($Piece(x,";",1),"=",2)
                set Temperature =  $Piece($Piece(x,";",2),"=",2)
                set Brightness =  $Piece($Piece(x,";",3),"=",2)
                do ..AddNew(Temperature,Humidity,Brightness) // Add data
            }
        }
    } catch anyError {
        close port
    }
}
```

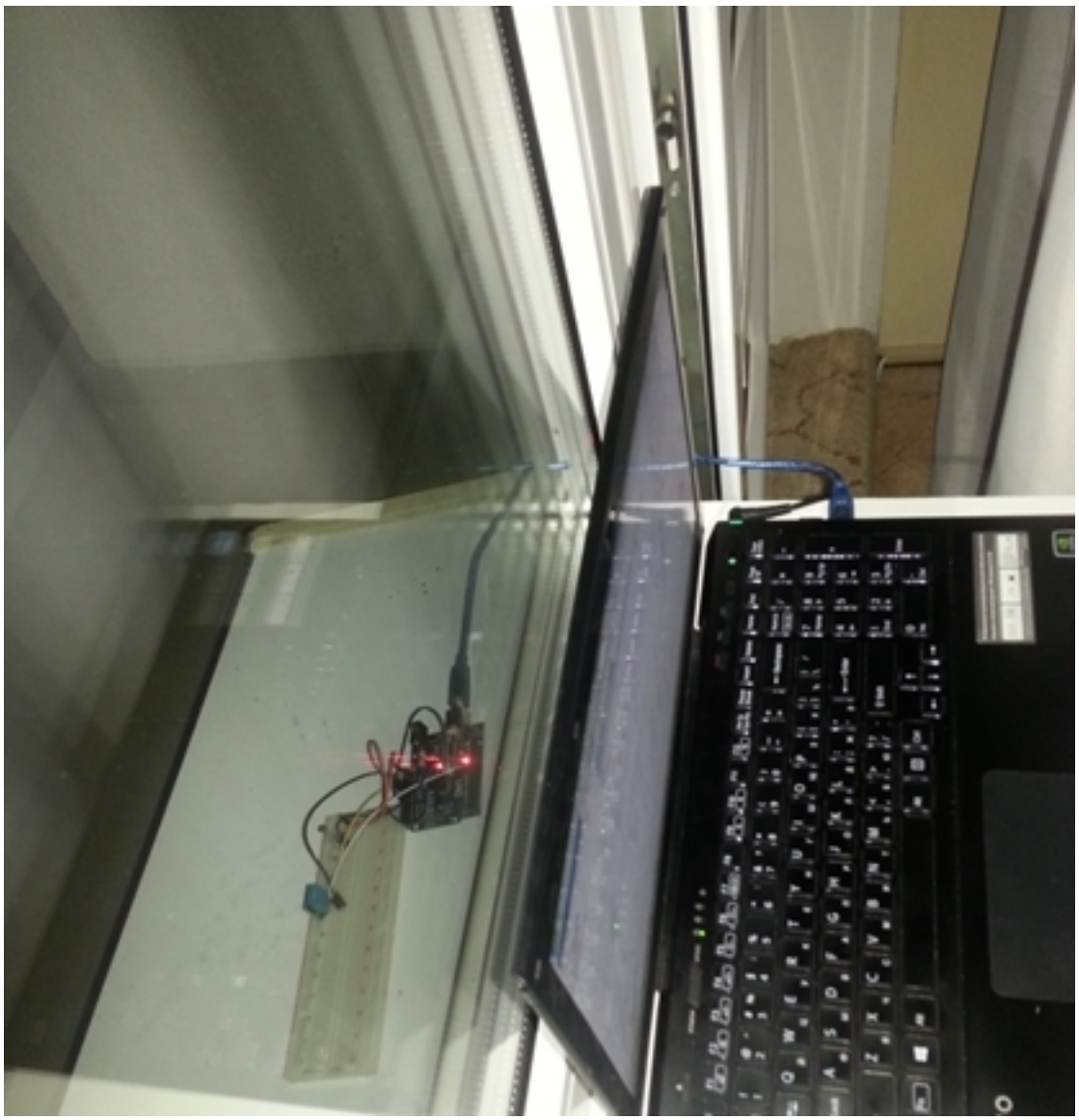
Y finalmente conectamos Arduino y ejecutamos el método ReceiveSerial:

```
write ##class(Arduino.Info).ReceiveSerial()
```

Este método recibirá y almacenrá datos desde Arduino ininterrumpidamente.

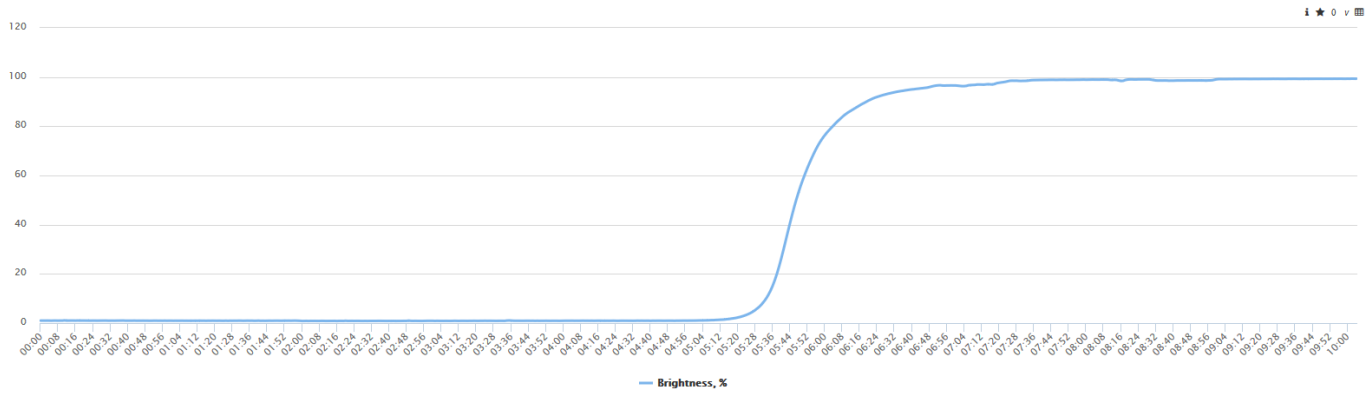
Visualización de Datos

Tras construir nuestro dispositivo lo podemos poner fuera por ejemplo, para recoger datos durante una noche:

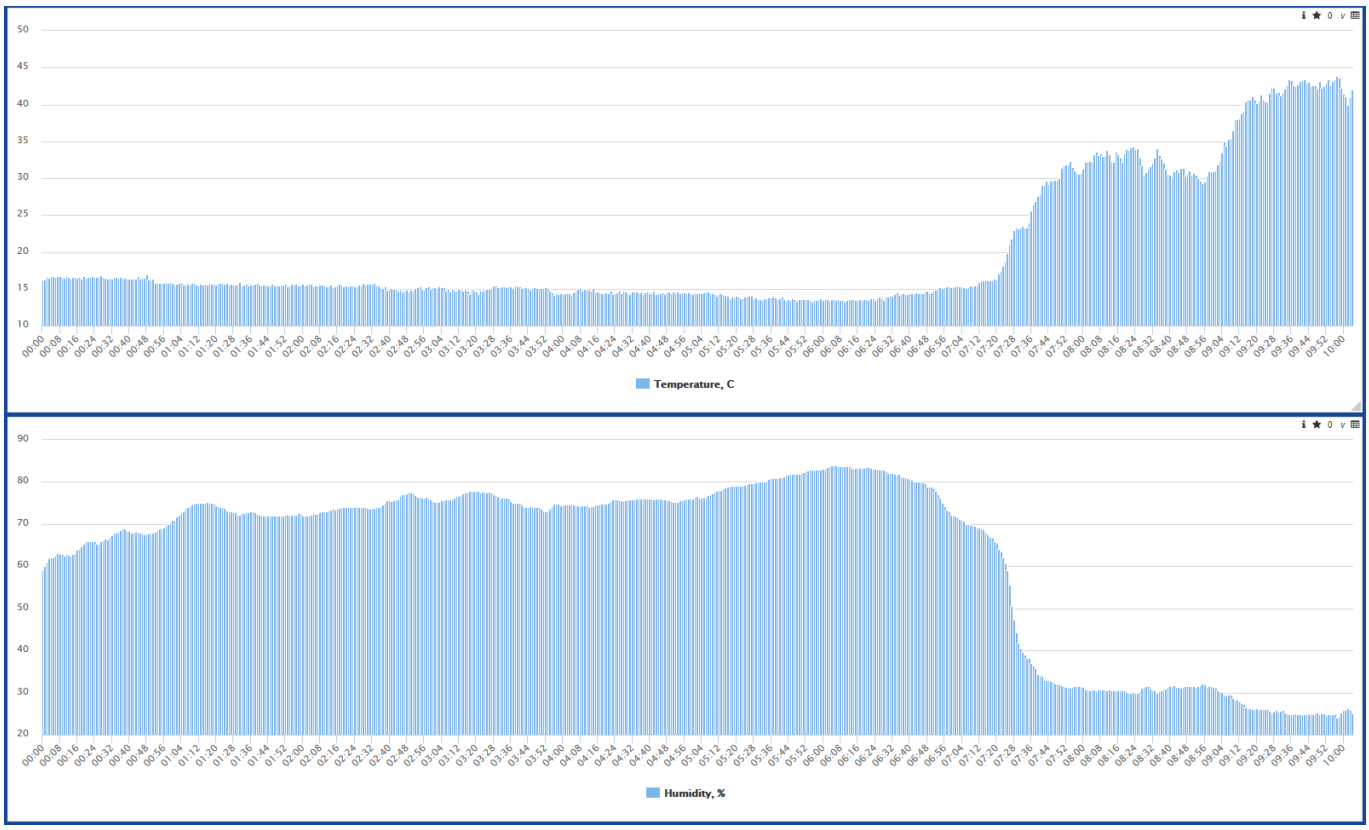


Al llegar la mañana tendremos miles de registros (en este caso más de 36.000), que podemos visualizar con las herramientas de Business Analytics utilizando por ejemplo herramientas disponibles en Open Exchange como el API REST de servidor MDX2JSON y el generador de cuadros de mando DeepSeeWeb , y aquí tenéis una muestra de los resultados:

Niveles de Luminosidad. En esta serie de datos la salida del Sol es claramente visible alrededor de las 5:50:



Gráficos de temperatura y humedad:



La correlación negativa entre la humedad y la temperatura es claramente visible.

Conclusión

Con InterSystems IRIS puedes comunicar directamente con una gran cantidad de dispositivos diferentes. Puedes desarrollar muy rápidamente tanto tus soluciones para procesamiento de datos como de visualización - a nuestro equipo le llevó sobre 4 horas construir nuestra propia estación meteorológica, conectarla a IRIS y visualizar los resultados y la mayor parte del tiempo se empleó en el diseño del circuito y en escribir y ajustar el código C.

Links

»Documentation
»GitHub - Repositorio del proyecto Original de Eduard Lebedyuk

#Terminal #InterSystems IRIS

URL de fuente: https://es.community.intersystems.com/post/estaci%C3%B3n-meteorol%C3%B3gica-con-arduino