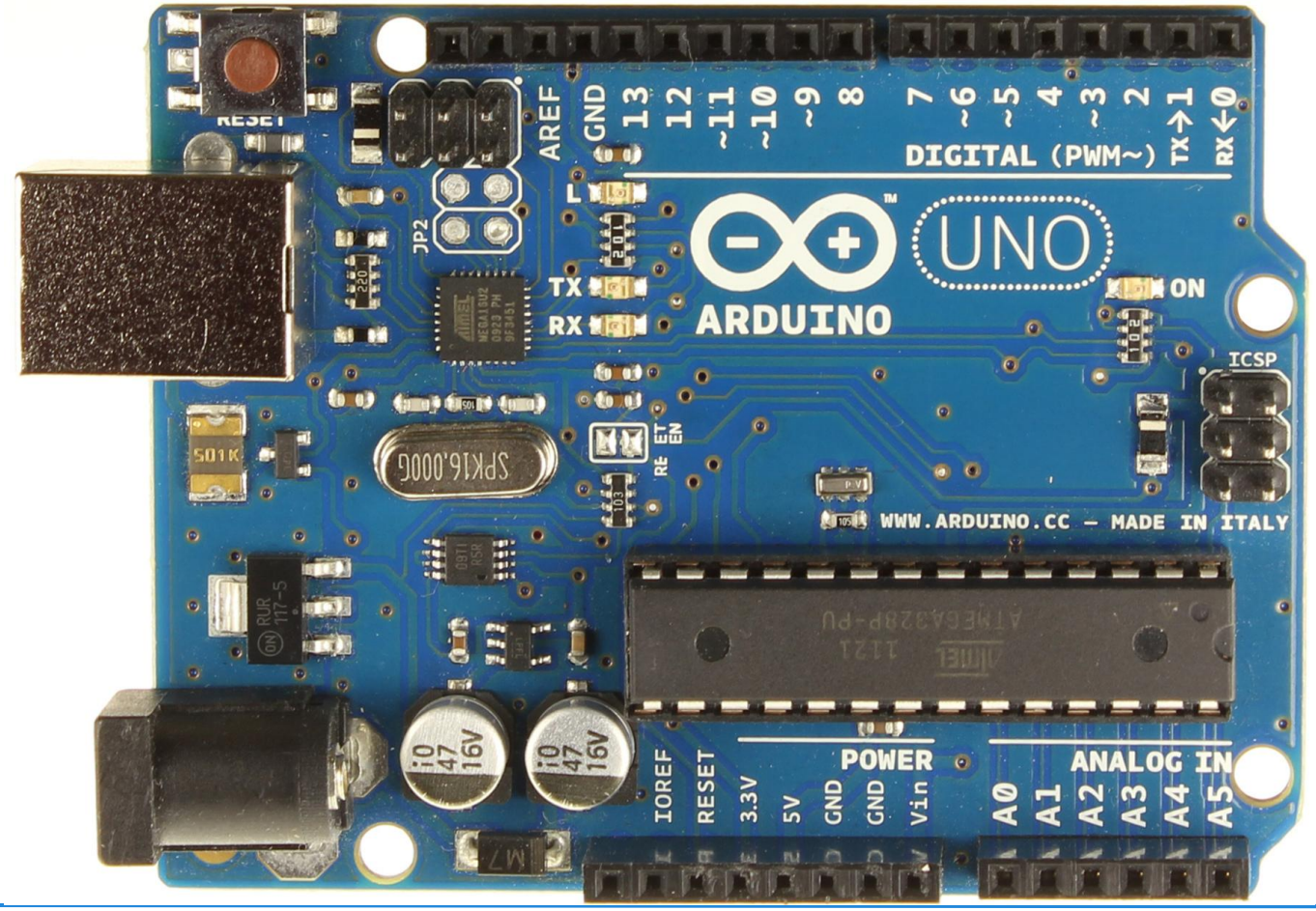
Curso avanzado sobre Arduino: Comunicaciones Serie

elCacharreo.com

Introducción a Arduino: Presente



Introducción a Arduino: Presente

José Antonio Vacas Martínez

blog

javacasm@elcacharreo.com

twitter
linkedin

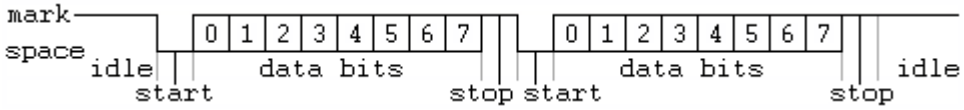
Comunicaciones: tipos

Comunicaciones sencillas entre dispositivos

- Redes 1 a 1 (peer to peer)
- Redes en bus

Comunicaciones: Puerto serie

- Comunicaciones serie sobre 2 (o 4 hilos)
- Facilidad de cableado (hasta 10m)
- Datos serializados



- Velocidades "aceptables": 1200, 9600,... 115200

  ¡¡¡bit por segundo!!!

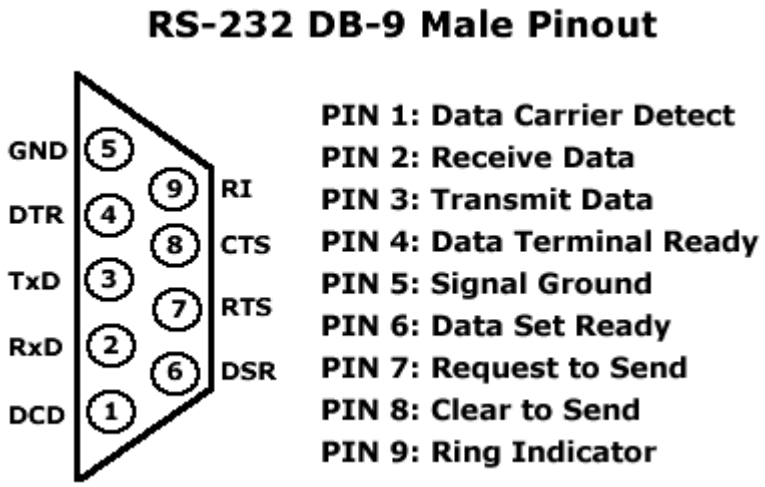
Referencia

Comunicaciones: Usos

Usos

- Ratón
- Impresoras
- modem
- otros dispositivos

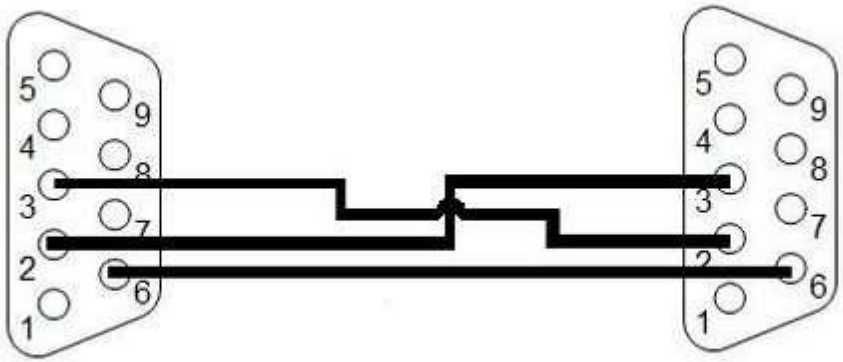
Comunicaciones: Cableado



Comunicaciones: Cableado RS232

Cableado

- RX - TX
- TX - RX
- GND - GND



Comunicaciones: Cableado RS232

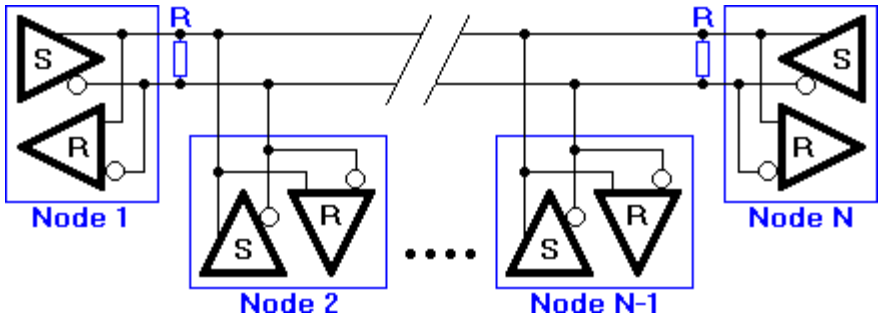
Parámetros

- Velocidad: 1200, 2400, 9600, 19800, 32600, 57900, 115200
- Paridad: par, impar, ninguna
- Bit de parada: 0, 1, 2
- Bit de datos: 7, 8

Comunicaciones: Cableado 485

Cableado (hasta 1.2Km)

- A+
- A-

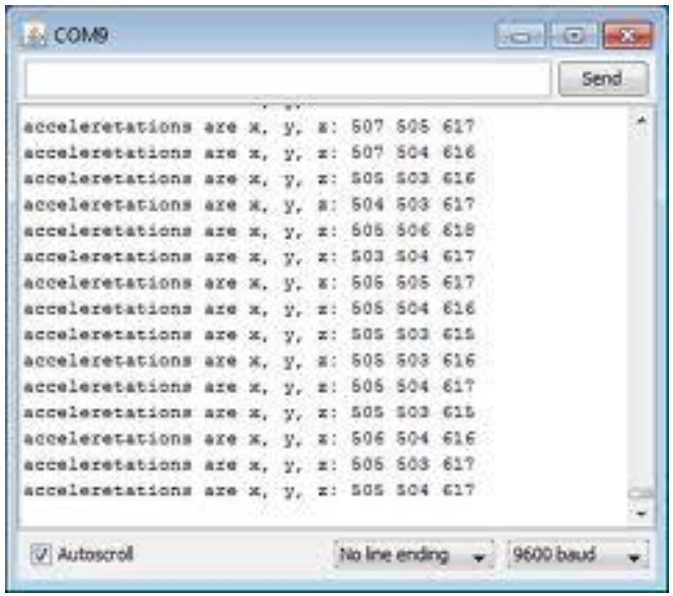


Comunicaciones: introducción

Comunicando con el pc:

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

int i=0;
void loop(){
    Serial.print("hola ");
    Serial.println(i++);
}
```

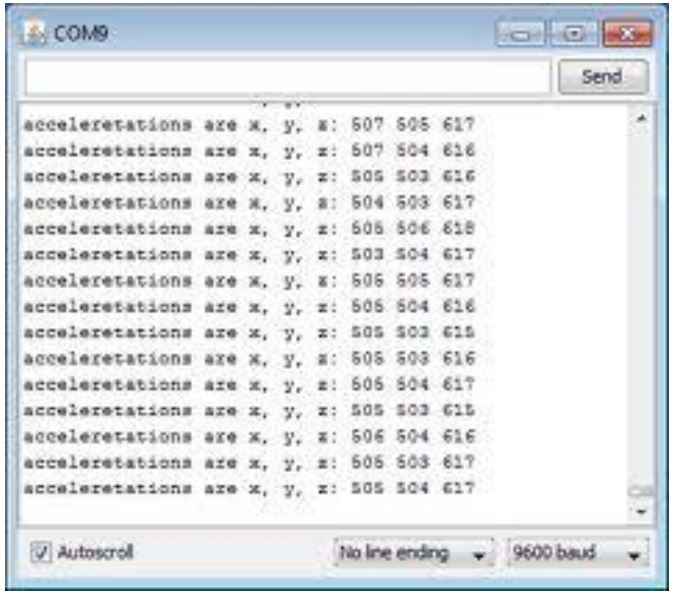


Comunicaciones: introducción

Comunicando con el pc:

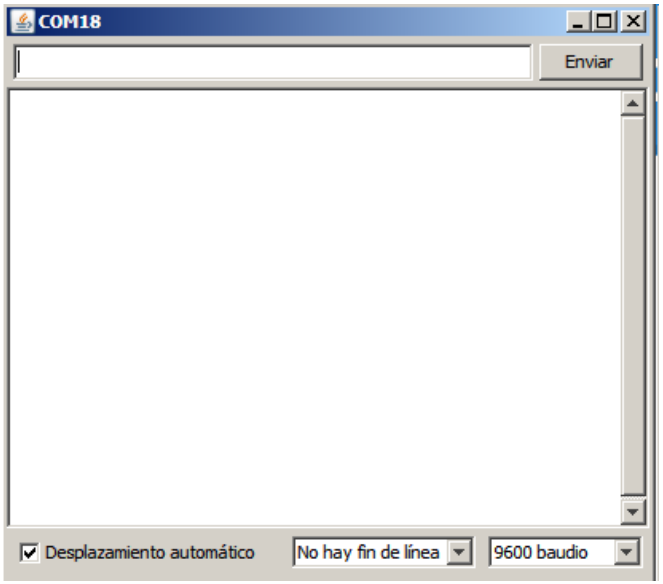
En arduino Mega:

```
Serial1.begin(speed)
Serial2.begin(speed)
Serial3.begin(speed)
```



Comunicaciones: Consola

Abrimos la consola serie



Comunicaciones: el puerto serie

Comandos via serie

Functions

- begin()
- end()
- available()
- read()
- peek()
- flush()
- print()
- println()
- write()
- SerialEvent()
- find()
- findUntil()
- flush()
- parseFloat()
- parseInt()
- readBytes()
- readBytesUntil()
- setTimeout()

http://www.instructables.com/answers/How-to-input-NUMBERS-through-Arduino-serialmonito/

Comunicaciones: Ejemplo de lectura

```
int incomingByte = 0;   // for incoming serial data

void setup() {
        Serial.begin(9600);     // opens serial port, sets data rate to 9600 bps
}

void loop() {
        // send data only when you receive data:
        if (Serial.available() > 0) {
                // read the incoming byte:
                incomingByte = Serial.read();

                // say what you got:
                Serial.print("I received: ");
                Serial.println(incomingByte, DEC);
        }
}
```

Ejercicio: lectura de comandos

Se trata de seleccionar qué diodo vamos a encender por medio de comandos:
Comandos:

r, g, b enciende el correspondiente color
0 apaga todos

Mejora, comando usando salidas analógicas y añadiendo un caracter de intensidad numerico
a5b6g2

Para ampliar el rango numérico podemos usar atoi

Ejercicio: lectura de enteros

```
// pins for the LEDs:
const int redPin = 3, greenPin = 5, bluePin = 6;

void setup() {
  Serial.begin(9600);
  pinMode(redPin, OUTPUT);   pinMode(greenPin, OUTPUT);  pinMode(bluePin, OUTPUT); }

void loop() {
  // if there's any serial available, read it:
  while (Serial.available() > 0) {
    // look for the next valid integer in the incoming serial stream:
    int red = Serial.parseInt(), green = Serial.parseInt(), blue = Serial.parseInt();
    // look for the newline. That's the end of your  sentence:
    if (Serial.read() == '\n') {
      // constrain the values to 0 - 255 and invert
      // if you're using a common-cathode LED, just use "constrain(color, 0, 255);"
      red = 255 - constrain(red, 0, 255);
      green = 255 - constrain(green, 0, 255);
      blue = 255 - constrain(blue, 0, 255);
      // fade the red, green, and blue legs of the LED:
      analogWrite(redPin, red);
      analogWrite(greenPin, green);
      analogWrite(bluePin, blue);
      // print the three numbers in one string as hexadecimal:
      Serial.print(red, HEX);       Serial.print(green, HEX);       Serial.println(blue, HEX);
    }
```

Comunicaciones: SoftwareSerial

¿Y si están ocupados los pines 0 y 1?

¿Cómo depuro?

¿Y si necesito todos los digitales?

(¿Cuántos digitales hay?)

¿Y si necesito comunicar con más un equipo?

Comunicaciones: SoftwareSerial

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(10, 11); // Usara los pines 10 para RX y 11 para TX
void setup()
{
      // Abre el puerto normal de comunicaciones :
      Serial.begin(57600);
      while (!Serial) {
      ; // espera a que el puerto esté disponible, sólo en Leonardo
      }
      Serial.println("hola!");

      // establece la velocidad del puerto SoftwareSerial
      mySerial.begin(4800);
      mySerial.println("Hello, world?");
}

void loop()
{
      if (mySerial.available()) //espera a tener datos disponibles para leer
      Serial.write(mySerial.read()); //escribe por el serie normal lo que lee por el otro
      if (Serial.available()) //espera hasta el serie tenga pendiente, escrito por nos
      mySerial.write(Serial.read()); //lo envía por el softwareserie
}
```

Comunicaciones: Stream

Stream is the base class for character and binary based streams. It is not called directly, but invoked whenever you use a function that relies on it.

Stream defines the reading functions in Arduino. When using any core functionality that uses a read() or similar method, you can safely assume it calls on the Stream class. For functions like print(), Stream inherits from the Print class.

Some of the libraries that rely on Stream include :

- Serial
- Wire
- Ethernet Client
- Ethernet Server
- SD

Conclusiones

Gracias por vuestra atención