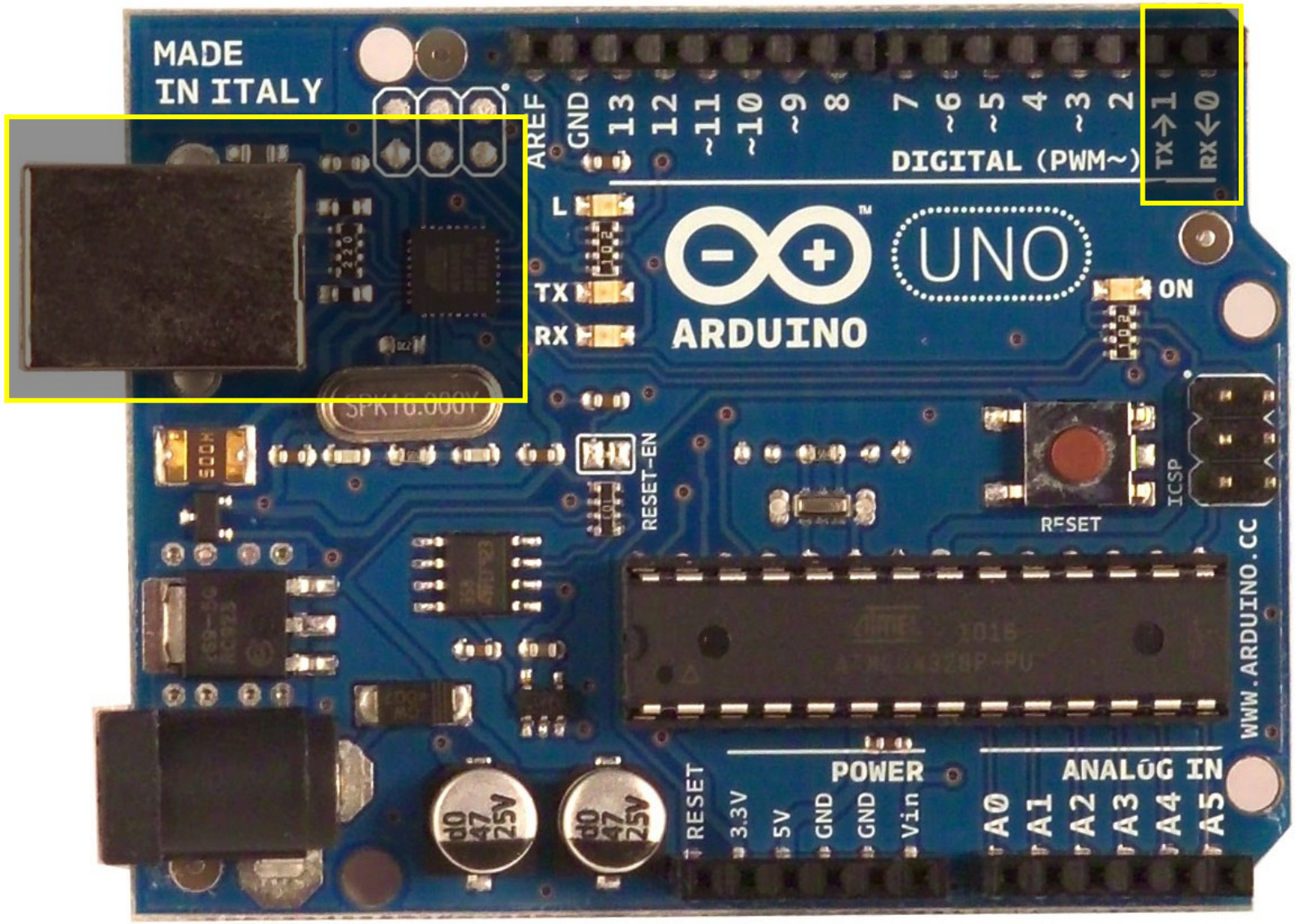
Arduino

Comunicación Serial

UART


MADE
IN ITALY
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
RESET-EN
RESET
ICSP
WWW.ARDUINO.CC
POWER
ANALOG IN
RESET
3.3V
5V
GND
GND
Vin
A0
A1
A2
A3
A4
A5

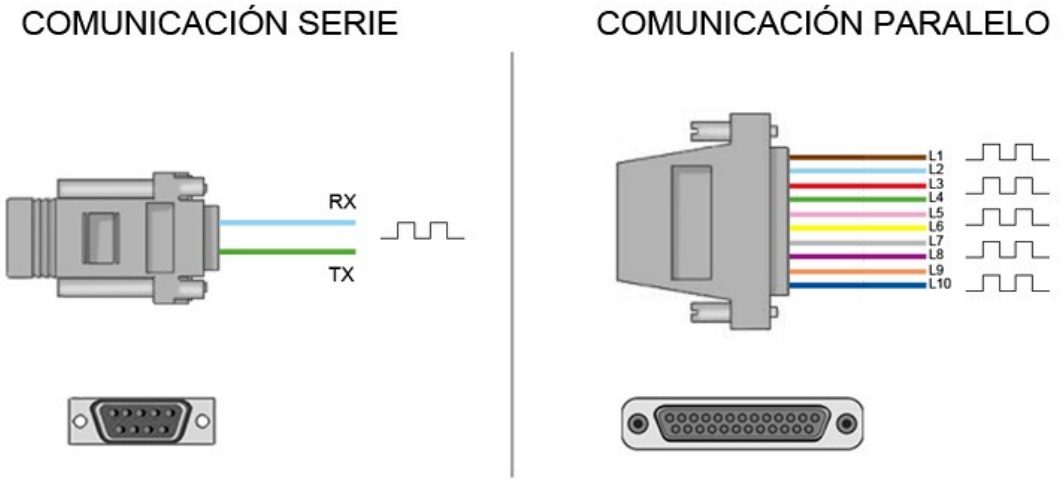
UART: Comunicación Serial

- UART, son las siglas en inglés de Universal Asynchronous Receiver-Transmitter, en español: Transmisor-Receptor Asíncrono Universal, es el dispositivo que controla los puertos y dispositivos serie.
- Los puertos serie son la forma principal de comunicar una placa Arduino con el mundo exterior (un computador, un celular, otro arduino).
- Existen un sin fin de posibilidades en las que se requiere el empleo del puerto serie. Por tanto, es un componente fundamental de una gran cantidad de proyectos de Arduino, y es uno de los elementos básicos que debemos aprender para poder sacar todo el potencial de Arduino.

Puerto Serie / Puerto Paralelo

- Un puerto serie envía la información mediante una secuencia de bits. Para ello se necesitan al menos dos conectores para realizar la comunicación de datos, RX (recepción) y TX (transmisión). No obstante, pueden existir otros conductores para referencia de tensión, sincronismo de reloj, etc.

- Por el contrario, un puerto paralelo envía la información mediante múltiples canales de forma simultánea.



Arduino y el Puerto Serial

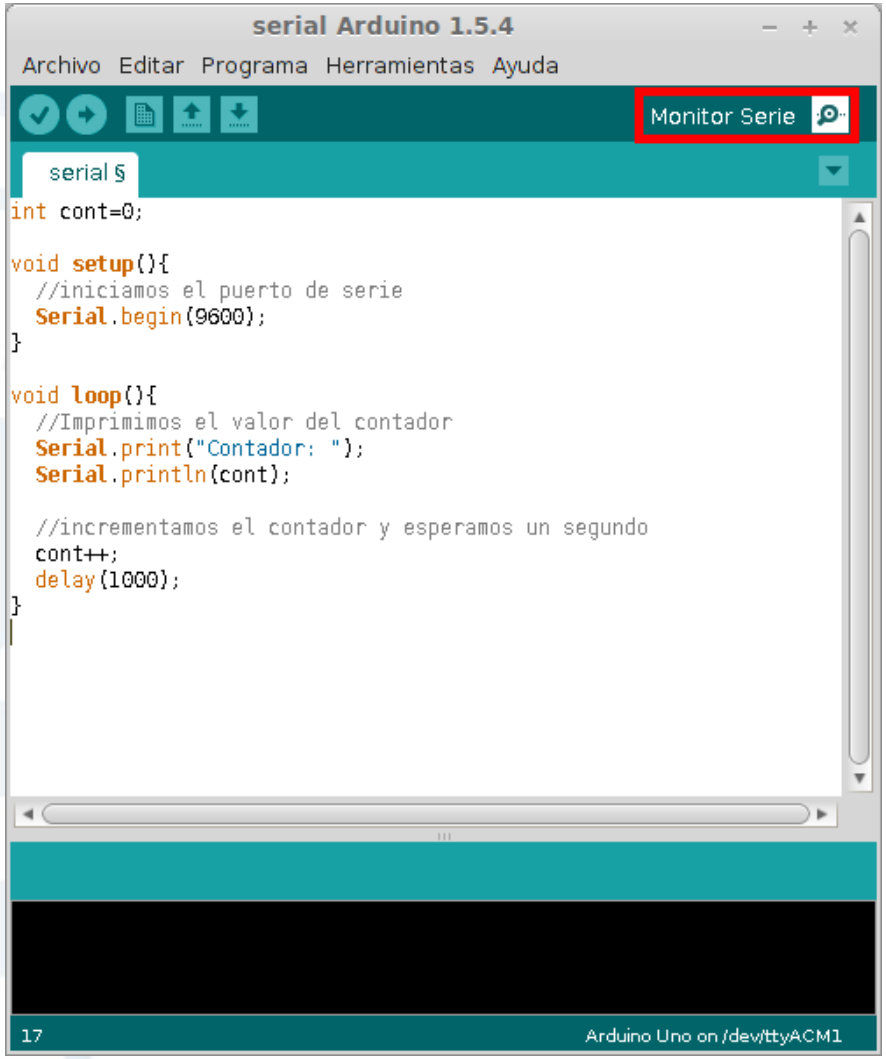
- Prácticamente todas las placas Arduino disponen al menos de una unidad UART. Las placas Arduino UNO y Mini Pro disponen de una unidad UART que operan a nivel TTL 0V / 5V, por lo que son directamente compatibles con la conexión USB. Por su parte, Arduino Mega y Arduino Due disponen de 4 unidades UART TTL 0V / 5V.

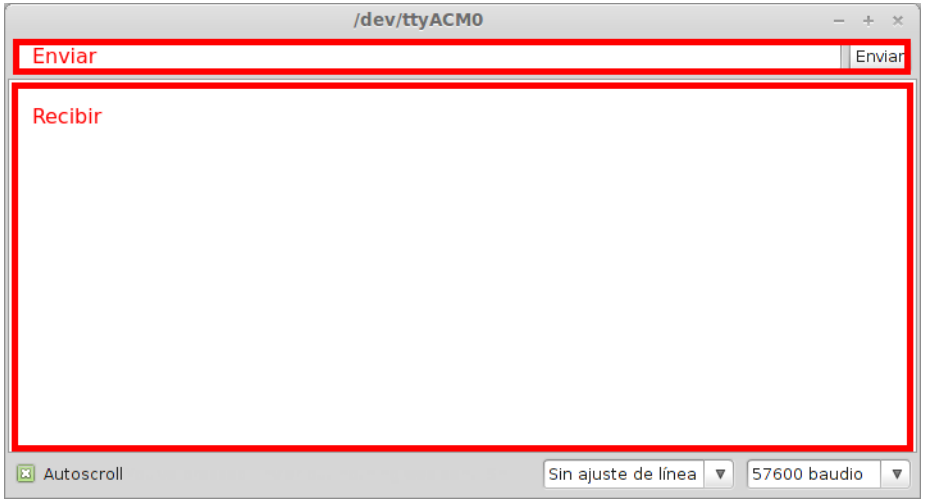
- Los puertos serie están físicamente unidos a distintos pines de la placa Arduino. Lógicamente, mientras usamos los puertos de serie no podemos usar como entradas o salidas digitales los pines asociados con el puerto de serie en uso.

- En Arduino UNO y Mini Pro los pines empleados son 0 (RX) y 1 (TX). En el caso de Arduino Mega y Arduino Due el puerto de serie 0 está conectado a los pines 0 (RX) y 1 (TX), el puerto de serie 1 a los pines 19 (RX) y 18 (TX) el puerto de serie 2 a los pines 17 (RX) y 16 (TX), y el puerto serie 3 a los pines 15 (RX) y 14 (TX).

No debemos acostumbrarnos a usar el puerto de serie si realmente no necesitamos comunicación

Monitor Serie en Arduino IDE





- El monitor de puerto serie es una pequeña utilidad integrada dentro de IDE Standard que nos permite enviar y recibir fácilmente información a través del puerto serie.
- dispone de dos zonas, una que muestra los datos recibidos, y otra para enviarlos.

Comandos para el puerto Serial

Serial.begin(rate)

Serial.begin(rate)

- Abre el puerto serie y fija la velocidad en baudios para la transmisión de datos en serie. El valor típico de velocidad para comunicarse con el computador es 9600, aunque otras velocidades pueden ser soportadas.

```
void setup()
{
   Serial.begin(9600);    // abre el Puerto serie
   // configurando la velocidad en 9600 bps
}
```

- Nota: Cuando se utiliza la comunicación serie los pines digitales 0 (RX) y 1 (TX) no pueden utilizarse para otros propósitos.ad

Serial.println(data)

Serial.println(data)

- Imprime los datos en el puerto serie, seguido por un retorno de carro y salto de línea. Existe también el comando Serial.print(data), sin embargo, este comando no agrega un salto de línea.

El siguiente ejemplo toma de una lectura analógica del pin 0 y envía estos datos al ordenador cada segundo.

```
void setup() {
Serial.begin(9600);    // configura el puerto serie a 9600bps
}

void loop() {
    Serial.println(analogRead(0)); // envía valor analógico
    delay(1000);     // espera 1 segundo
}
```

Serial.println(data, format)

Serial.println(data, data type)

- Vuelca o envía un número o una cadena de carateres al puerto serie, seguido de un caracter de retorno de carro "CR" (ASCII 13, or '\r')y un caracter de salto de línea "LF"(ASCII 10, or '\n'). Toma la misma forma que el comando Serial.print()

- Serial.println(b) vuelca o envía el valor de b como un número decimal en caracteres ASCII seguido de "CR" y "LF".

Serial.println(data, format)

- Serial.println(b, DEC): envía el valor de b como un número decimal en caracteres ASCII seguido de "CR" y "LF".
- Serial.println(b, HEX) envía el valor de b como un número hexdecimal en caracteres ASCII seguido de "CR" y "LF".
- Serial.println(b, OCT) envía el valor de b como un número octal en caracteres ASCII seguido de "CR" y "LF".
- Serial.println(b, BIN) envía el valor de b como un número binario en caracteres ASCII seguido de "CR" y "LF".
- Serial.print(b, BYTE) envía el valor de b como un byteseguido de "CR" y "LF".

Serial.avaible()

int Serial.available()

- Devuelve un entero con el número de bytes (carácteres) disponibles para leer desde el buffer serie, ó 0 si no hay ninguno. Si hay algún dato disponible, SerialAvailable() será mayor que 0. El buffer serie puede almacenar como máximo 128 bytes.

```
int incomingByte = 0; // almacena el dato serie
void setup() {
    Serial.begin(9600); // abre el puerto serie, y le asigna la
velocidad de 9600 bps
}
void loop() {
   // envía datos sólo si los recibe:
    if (Serial.available() > 0) {
    // lee el byte de entrada:
    incomingByte = Serial.read();
    //lo vuelca a pantalla
    Serial.print("He recibido: "); Serial.println(incomingByte,
DEC);
   }
}
```

Serial.Read()

int Serial.Read()

- Lee o captura un byte (carácter) desde el puerto serie. Devuelve :El siguiente byte (carácter) desde el puerto serie, ó -1 si no hay ninguno.

```
int incomingByte = 0; // almacenar el dato serie
void setup() {
   Serial.begin(9600); // abre el puerto serie a 9600 bps
}
void loop() {
   // envía datos sólo si los recibe:
   if (Serial.available() > 0) {
    // lee el byte de entrada:
    incomingByte = Serial.read();
    //lo vuelca a pantalla
    Serial.print("He recibido: ");
    Serial.println(incomingByte, DEC);
    }
}
```

Ejemplos

Recibir datos desde Arduino

- En este primer código vamos a recibir el valor de un contador enviado desde la placa Arduino. Este valor se incrementa cada segundo.

```
1   int cont=0;
2
3   void setup(){
4     //iniciamos el puerto de serie
5     Serial.begin(9600);
6   }
7
8   void loop(){
9     //Imprimimos el valor del contador
10    Serial.print("Contador: ");
11    Serial.println(cont);
12
13    //incrementamos el contador y esperamos un segundo
14    cont++;
15    delay(1000);
16  }
```

Enviar Datos al Arduino

```
1  int option;
2  int led = 13;
3
4  void setup(){
5    Serial.begin(9600);
6    pinMode(led, OUTPUT);
7  }
8
9  void loop(){
10   //si existe datos disponibles los leemos
11   if (Serial.available()>0){
12     //leemos la opcion enviada
13     option=Serial.read();
14     if(option=='a') {
15       digitalWrite(led, LOW);
16       Serial.println("OFF");
17     }
18     if(option=='b') {
19       digitalWrite(led, HIGH);
20       Serial.println("ON");
21     }
22   }
23 }
```

En este ejemplo empleamos el puerto de serie para encender o apagar el LED integrado en la placa Arduino. Para ello enviamos un carácter a la placa Arduino, empleando el monitor serial. En caso de enviar ‘a’ la placa Arduino apaga el LED, y en caso de enviar ‘b’ lo enciende.

Enviar números al Arduino

```
1  int option;
2  int led = 13;
3
4  void setup(){
5    Serial.begin(9600);
6    pinMode(led, OUTPUT);
7  }
8
9  void loop(){
10   //si existe información pendiente
11   if (Serial.available()>0){
12     //leeemos la opcion
13     char option = Serial.read();
14     //si la opcion esta entre '1' y '9'
15     if (option >= '1' && option <= '9')
16     {
17       //restamos el valor '0' para obtener el numeroenviado
18       option -= '0';
19       for(int i=0;i<option;i++){
20          digitalWrite(led, HIGH);
21          delay(100);
22          digitalWrite(led, LOW);
23          delay(200);
24       }
25     }
26   }
27 }
```

- Por último, en este ejemplo enviamos un número entre 1 a 9 a través del monitor serial, y la placa Ardunino hace parpadear el LED integrado el número de veces indicado.