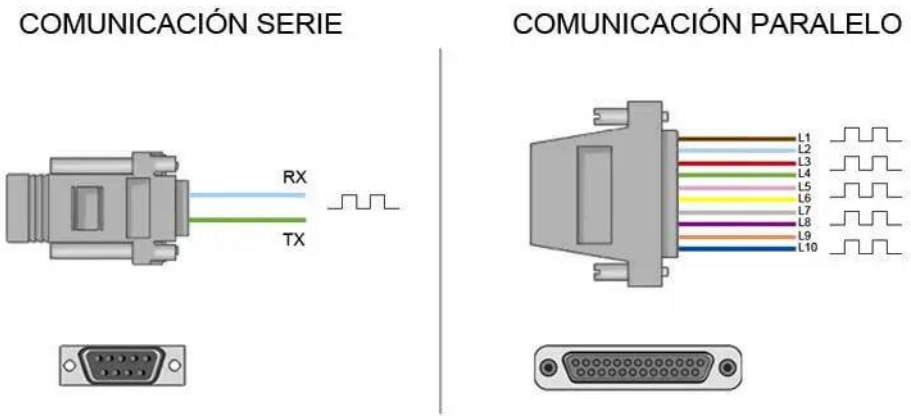
TEMA 7. COMUNICACIÓN SERIAL CON ARDUINO UNO

¿Qué es el puerto serie?

Un puerto es el nombre genérico con que denominamos a los interfaces, físicos o virtuales, que permiten la comunicación entre dos ordenadores o dispositivos.

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 enviaría la información mediante múltiples canales de forma simultánea. Para ello necesitan un número superior de conductores de comunicación, que varían en función del tipo de puerto (igualmente existe la posibilidad de conductores adicionales además de los de comunicación).



Sin embargo, a medida que los procesadores se hicieron más rápidos los puertos de serie fueron desplazando progresivamente a los puertos paralelos en la mayoría de aplicaciones.

Un ordenador convencional dispone de varios puertos de serie. Los más conocidos son el popular USB (Universal Serial Port) y el ya casi olvidado RS-232 (el de los antiguos ratones).

Sin embargo, dentro del ámbito de la informática y automatización existen una gran cantidad adicional de tipos de puertos serie, como por ejemplo el RS-485, I2C, SPI, Serial Ata, Pcie Express, Ethernet o FireWire, entre otros.

Sin embargo, a medida que los procesadores se hicieron más rápidos los puertos de serie fueron desplazando progresivamente a los puertos paralelos en la mayoría de aplicaciones.

Un ordenador convencional dispone de varios puertos de serie. Los más conocidos son el popular USB (Universal Serial Port) y el ya casi olvidado RS-232 (el de los antiguos ratones).

Sin embargo, dentro del ámbito de la informática y automatización existen una gran cantidad adicional de tipos de puertos serie, como por ejemplo el RS-485, I2C, SPI, Serial Ata, Pcie Express, Ethernet o FireWire, entre otros.

Placa Arduino	Puerto Serie	Pin RX	Pin TX
Arduino UNO	Serie 0	0 (RX)	1 (TX)
Arduino Mini Pro	Serie 0	0 (RX)	1 (TX)
Arduino Mega	Serie 0	0 (RX)	1 (TX)
Arduino Mega	Serie 1	19 (RX)	18 (TX)
Arduino Mega	Serie 2	17 (RX)	16 (TX)
Arduino Mega	Serie 3	15 (RX)	14 (TX)
Arduino Due	Serie 0	0 (RX)	1 (TX)
Arduino Due	Serie 1	19 (RX)	18 (TX)
Arduino Due	Serie 2	17 (RX)	16 (TX)
Arduino Due	Serie 3	15 (RX)	14 (TX)

Comunicación Serial con Arduino UNO

La comunicación serial es un método de transferencia de datos entre dos dispositivos que permite enviar y recibir información bit por bit. En el caso de Arduino UNO, la comunicación serial es fundamental para interactuar con otros dispositivos (computadoras, sensores, módulos periféricos, etc.) o depurar proyectos.

---

1. ¿Qué es la Comunicación Serial?

La comunicación serial es un proceso en el que los datos se transfieren un bit a la vez a través de un único canal de comunicación. Este método se usa ampliamente debido a su simplicidad y eficiencia.

Tipos de Comunicación Serial:

1. Asíncrona: No requiere una señal de reloj compartida. Se utiliza un protocolo como UART (Universal Asynchronous Receiver-Transmitter).
2. Síncrona: Requiere una señal de reloj para sincronizar el envío y recepción de datos.

Conceptos Fundamentales de la Comunicación Serial

1.1 Comunicación Serial

- La comunicación serial envía datos **bit por bit** a través de un solo canal, en contraste con la comunicación paralela, que envía varios bits simultáneamente.
- Es **asíncrona**, lo que significa que no requiere una señal de reloj compartida entre los dispositivos.

1.2 UART

- El UART es el protocolo utilizado para manejar la comunicación serial en Arduino.
- Funciona utilizando:
  - **TX (Transmisión):** Enviar datos.
  - **RX (Recepción):** Recibir datos.

1.3 Baud Rate

- Es la velocidad de transmisión de datos, medida en bits por segundo (bps).
- Ambas partes deben operar con la misma velocidad.
- Valores comunes: **9600**, **14400**, **19200**, **38400**, **57600**, **115200** bps.

Características de la Comunicación Serial en Arduino

1. **Hardware UART:**
   - Arduino UNO tiene un solo puerto UART conectado a los pines **TX (1)** y **RX (0)**.
   - El puerto UART también se conecta al USB mediante el chip convertidor USB a serie.
2. **Software Serial:**
   - Si necesitas más puertos UART, puedes usar la biblioteca **SoftwareSerial** para emular la comunicación serial en otros pines digitales.
3. **Voltaje de Señal:**
   - La señal de comunicación serial en Arduino opera a **5V** (compatible con niveles lógicos TTL).

4. **Buffer Serial:**
    - Arduino usa un buffer de entrada y salida (tamaño predeterminado: 64 bytes) para manejar datos en la comunicación serial.

En el caso de Arduino UNO, se utiliza la comunicación serial UART**.**

2. Características de la Comunicación Serial en Arduino UNO

1. Hardware Serial (UART):
    - Pines: TX (Pin 1) para transmisión y RX (Pin 0) para recepción.
    - Lógica: Voltaje de 0V (LOW) para un bit 0 y 5V (HIGH) para un bit 1.
    - Protocolo: UART.
2. Velocidad de Transmisión (Baud Rate):
    - Se define en bits por segundo (bps).
    - Ejemplo: 9600 bps, 115200 bps.
    - Ambos dispositivos deben operar con el mismo baud rate.
3. Buffer de Datos:
    - Arduino tiene un búfer para datos entrantes y salientes (tamaño predeterminado: 64 bytes).
4. Interfaces de Comunicación:
    - Puerto USB: Para comunicación entre Arduino y una computadora.
    - Pines TX/RX: Para conectar otros dispositivos.

Configuración de la Comunicación Serial

La configuración se realiza con la función Serial.begin(baudRate). Esto inicializa la UART con una velocidad de transmisión específica.

Estructura de Configuración:

```
void setup() {
  Serial.begin(9600); // Inicializar comunicación serial a 9600 bps
}
```

Baud Rates Comunes:

- 300, 1200, 2400, 4800, 9600, 14400, 19200, 38400, 57600, 115200.
- La mayoría de los proyectos utilizan 9600 o 115200 bps.

Principales Funciones de la Biblioteca Serial

Arduino utiliza la biblioteca Serial para gestionar la comunicación serial. A continuación, se explican sus funciones principales:

Función	Descripción
`Serial.begin(baudRate)`	Inicializa la comunicación serial con el baud rate especificado.
`Serial.print(data)`	Envía datos al puerto serial como texto sin salto de línea.
`Serial.println(data)`	Envía datos con un salto de línea al final.
`Serial.read()`	Lee un byte del puerto serial.
`Serial.available()`	Devuelve el número de bytes disponibles para leer en el búfer de entrada.
`Serial.write(data)`	Envía datos en forma binaria al puerto serial.
`Serial.flush()`	Espera a que se envíen todos los datos en el búfer de salida.

Flujo de Configuración de la Comunicación Serial

1. **Iniciar la Comunicación:**
    - Configura la velocidad de transmisión con Serial.begin(baudRate).
2. **Enviar Datos:**
    - Usa Serial.print() o Serial.println() para enviar información al puerto serial.
3. **Leer Datos:**
    - Usa Serial.read() para obtener datos desde el búfer de entrada.
4. **Verificar Disponibilidad de Datos:**
    - Usa Serial.available() para asegurarte de que hay datos antes de intentar leerlos.

Ejemplo Básico de Comunicación Serial

6.1 Enviar Datos al Monitor Serie

Este ejemplo envía un mensaje continuo al monitor serie.

Código:

```
void setup() {
  Serial.begin(9600); // Configura el baud rate
  Serial.println("Iniciando comunicación serial...");
}
void loop() {
  Serial.println("Este es un mensaje de prueba.");
  delay(1000); // Espera 1 segundo entre mensajes
}
```

Leer Datos desde el Monitor Serie

Este programa lee un carácter enviado desde el monitor serie y lo responde.

Código:

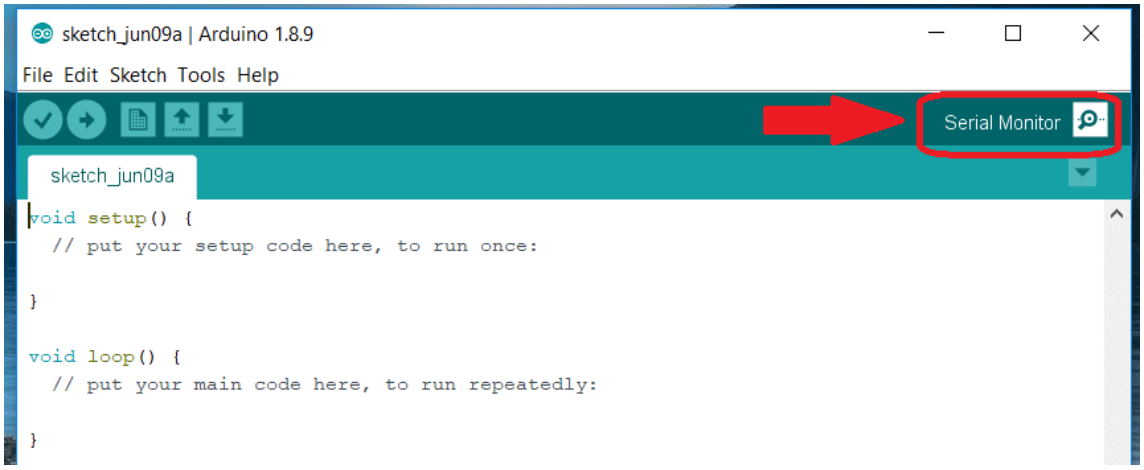
```
void setup() {
  Serial.begin(9600);
  Serial.println("Escribe algo en el monitor serie:");
}
void loop() {
  if (Serial.available() > 0) {          // Si hay datos disponibles
    char receivedChar = Serial.read();   // Leer el carácter recibido
    Serial.print("Recibido: ");
    Serial.println(receivedChar);        // Responder con el carácter recibido
  }
}
```

Comunicación Serial Arduino PC

Para utilizar la comunicación serial en Arduino, primero se debe abrir la comunicación serial mediante el comando Serial.begin(). Este comando establece la velocidad de transmisión en baudios y se utiliza para indicar a la placa Arduino que está lista para enviar y recibir datos. Por ejemplo, el comando Serial.begin(9600) establece la velocidad de transmisión en 9600 baudios.

Para verificar que los datos están siendo transmitidos correctamente, se puede utilizar el Monitor Serial de Arduino. Este es una herramienta incluida en el software de Arduino.

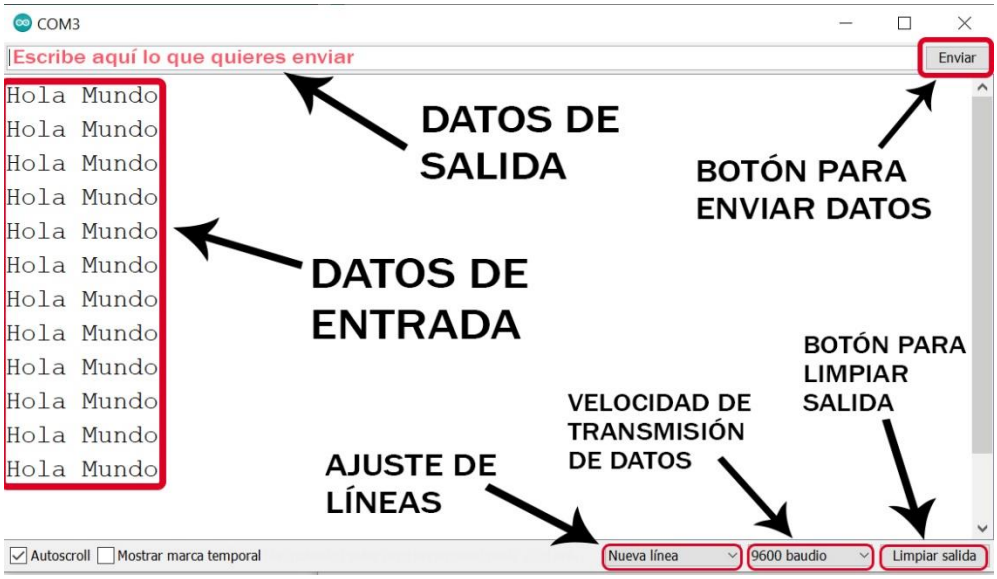
Una vez conectamos nuestro Arduino a nuestro computador a través del cable USB, dentro del IDE de arduino podemos acceder al Monitor Serial



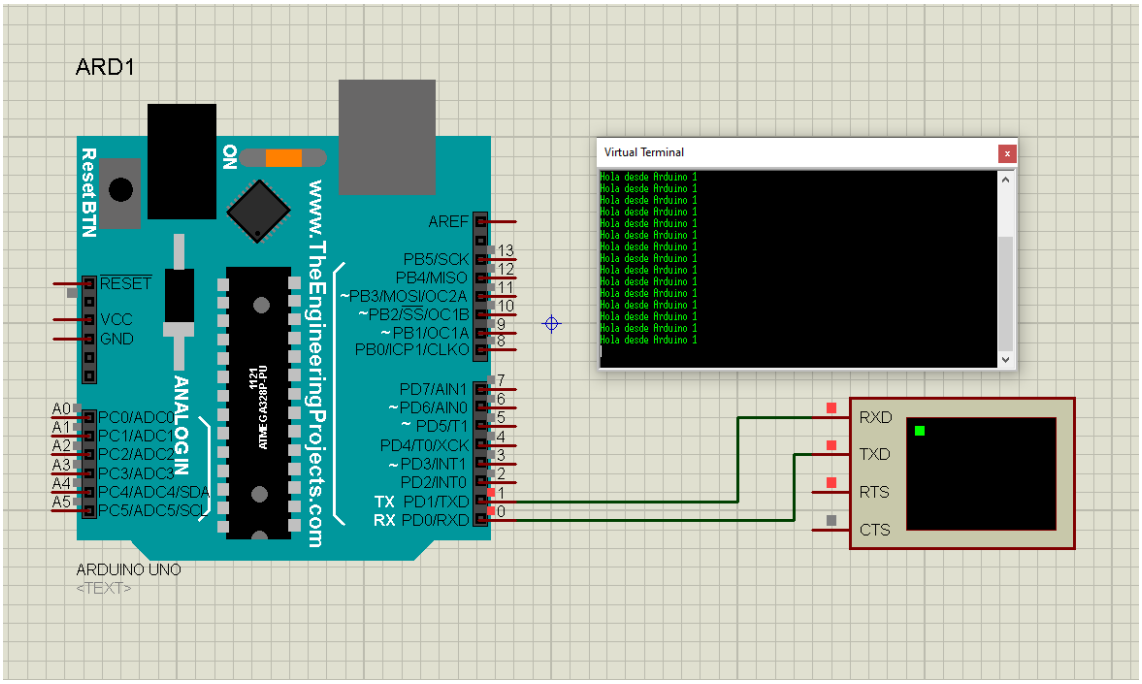
Enviar datos con el monitor serie de Arduino

Posiblemente te estés preguntado cómo puedes ahora enviarle datos desde tu PC a Arduino. Podrías desarrollar un programa en Python, C, C++ o cualquier otro lenguaje de programación que se ejecutara en tu ordenador e hiciera dicha tarea.

Sin embargo, ¡hay una manera mucho más sencilla de hacerlo! Simplemente abre el monitor serie de Arduino. Arriba tienes un espacio donde puedes escribir. Simplemente escribe lo que quieras enviarle a Arduino y dale al botón de enviar que hay arriba a la derecha.



7.1. ENVIAR HOLA MUNDO DESDE ARDUINO



Material necesario

1. 2 Arduinos (pueden ser Arduino Uno, Nano, Mega, etc.).
2. Cables de conexión.
3. Protoboard (opcional para conexiones más organizadas).
4. Resistencias (opcional, para seguridad en comunicación).

Conexiones

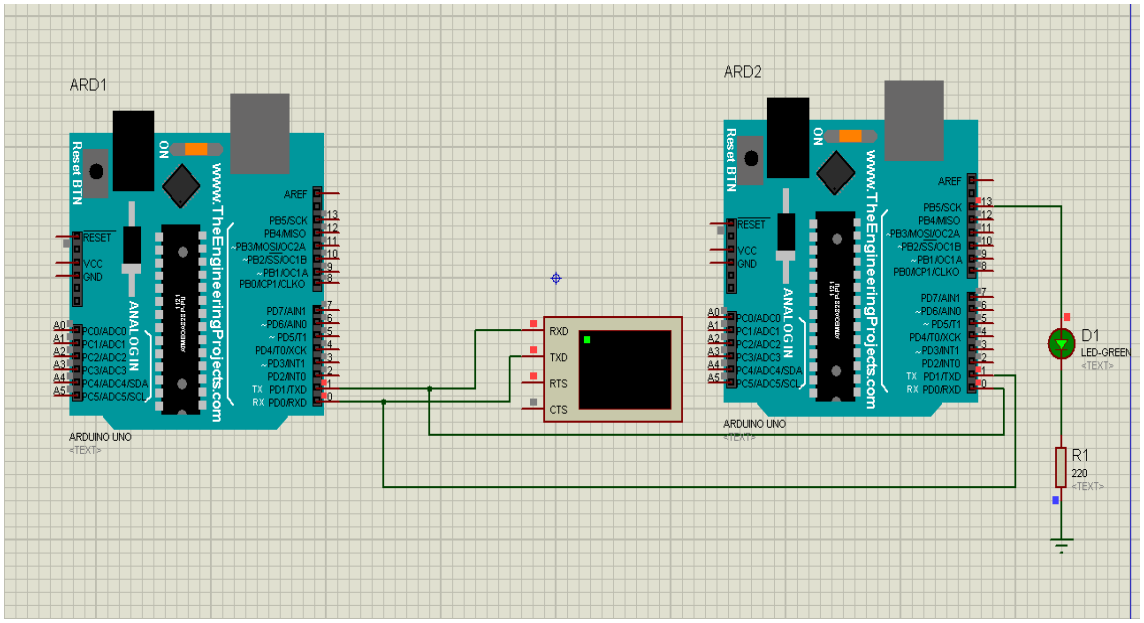
1. Conecta los pines TX y RX de los Arduinos:
   - TX del Arduino 1 -> RX del Arduino 2.
   - RX del Arduino 1 -> TX del Arduino 2.
2. Conecta las tierras (GND):
   - GND del Arduino 1 -> GND del Arduino 2.
3. Si usas Arduinos que operan a diferentes voltajes (por ejemplo, un Uno de 5V y un Nano de 3.3V), utiliza un divisor de voltaje o un convertidor lógico entre TX y RX para proteger los pines.

Código para el Arduino 1 (emisor)

```
void setup() {
  Serial.begin(9600); // Inicializar la comunicación serial
}

void loop() {
  Serial.println("Hola desde Arduino 1"); // Enviar mensaje
  delay(1000); // Esperar 1 segundo antes de enviar de nuevo
}
```

7.2.Encender un led desde un pulsador mediante dos arduinos por comunicación serial



Material necesario

1. 2 Arduinos (pueden ser Arduino Uno, Nano, Mega, etc.).
2. Un pulsador.
3. Un LED.
4. Resistencias (330 ohm para el LED y 10k ohm para el pulsador).
5. Cables de conexión.
6. Protoboard.

Esquema de conexiones

Arduino Emisor (con el pulsador)

1. Conecta un extremo del pulsador al pin digital 2.
2. Conecta el otro extremo del pulsador a GND.
3. Coloca una resistencia de 10k ohm entre el pin digital 2 y VCC (Pull-up).

Arduino Receptor (con el LED)

1. Conecta el ánodo del LED al pin digital 13 (o cualquier otro pin digital disponible).
2. Conecta el cátodo del LED a GND mediante una resistencia de 330 ohm.

Conexión entre los Arduinos

1. TX del Arduino Emisor -> RX del Arduino Receptor.
2. RX del Arduino Emisor -> TX del Arduino Receptor.
3. GND del Arduino Emisor -> GND del Arduino Receptor.

Código para el Arduino Emisor

```
void setup() {
  Serial.begin(9600); // Configura la velocidad de comunicación serie
}

void loop() {
  Serial.println("1"); // Enviar el comando para encender el LED
  delay(1000);        // Esperar 1 segundo
  Serial.println("0"); // Enviar el comando para apagar el LED
  delay(1000);        // Esperar 1 segundo
}
```

Código para el Arduino Receptor

```
const int ledPin = 13; // Pin donde está conectado el LED

void setup() {
  pinMode(ledPin, OUTPUT); // Configura el pin del LED como salida
  Serial.begin(9600);      // Configura la velocidad de comunicación serie
}

void loop() {
  if (Serial.available() > 0) { // Verifica si hay datos disponibles
    char command = Serial.read(); // Lee el comando recibido
    if (command == '1') {
      digitalWrite(ledPin, HIGH); // Enciende el LED
    } else if (command == '0') {
      digitalWrite(ledPin, LOW);  // Apaga el LED
    }
  }
}
```