UART (Universal Asynchronous Receiver/Transmitter)

UART es una forma de comunicación serial debido a que los datos son transmitidos en una secuencia de bits, usando una línea de transmisión (TX) y otra de recepción (RX).

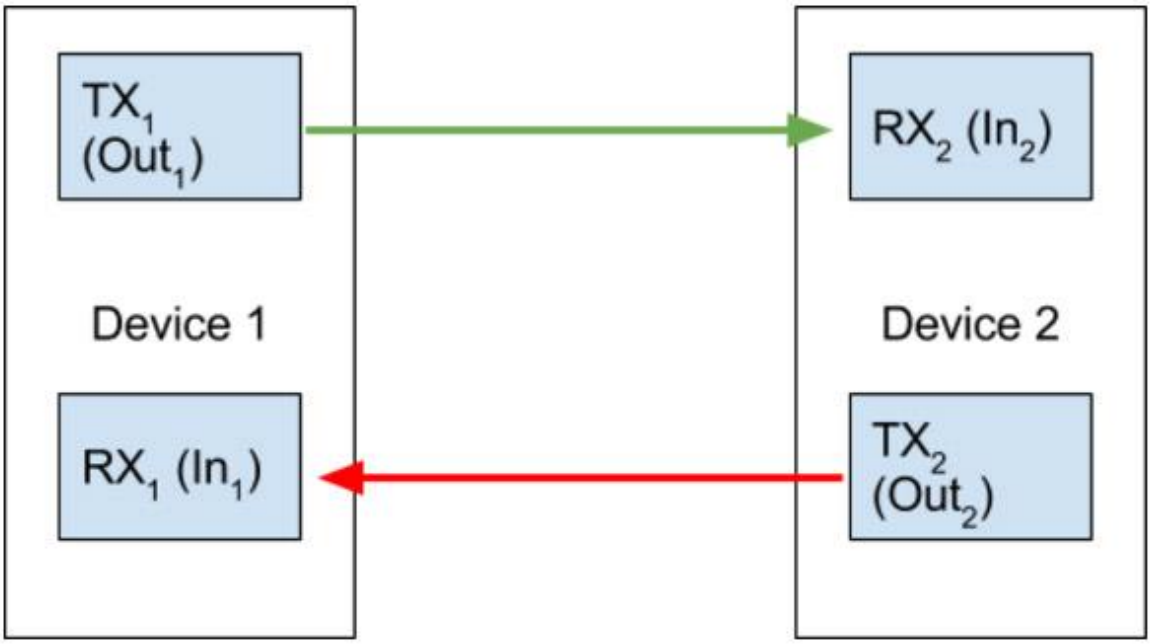

Imagen 1 - Conexiones UART

En vez de ser un protocolo, UART es uno o una serie de circuitos que implementan una comunicación serial asíncrona. El protocolo RS-232 es uno de muchos que operan en estos circuitos. La forma más conocida de comunicación serial entre periféricos y componentes es por medio del puerto USB.

El software de Arduino contiene la librería SoftwareSerial que implementa este protocolo en software utilizando dos pines digitales. Cada tarjeta Arduino con capacidad de comunicación serial tiene estos pines ya definidos; en caso de no contar con dicho circuito, se necesita tener un convertidor de señal RS-232 o TTL a USB.

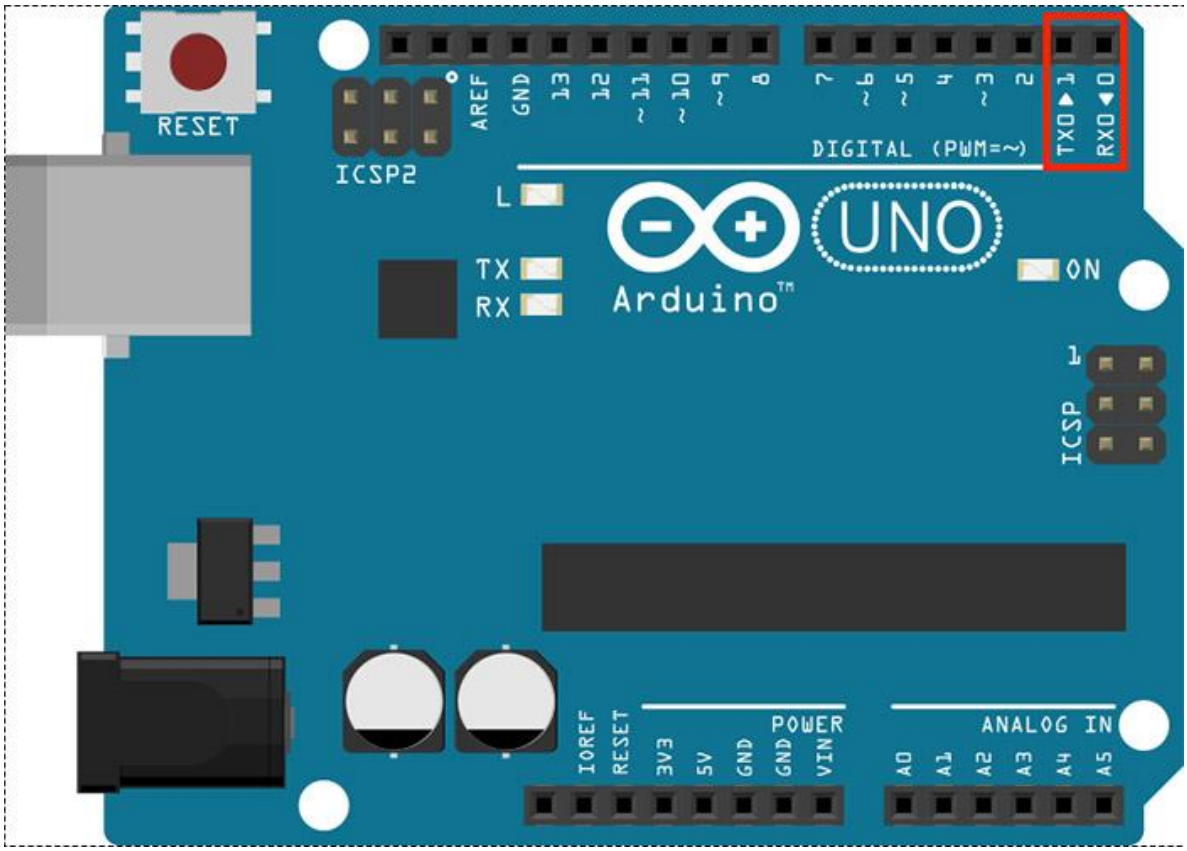

Imagen 2 - Arduino UNO - Ubicación de las líneas de transmisión y recepción

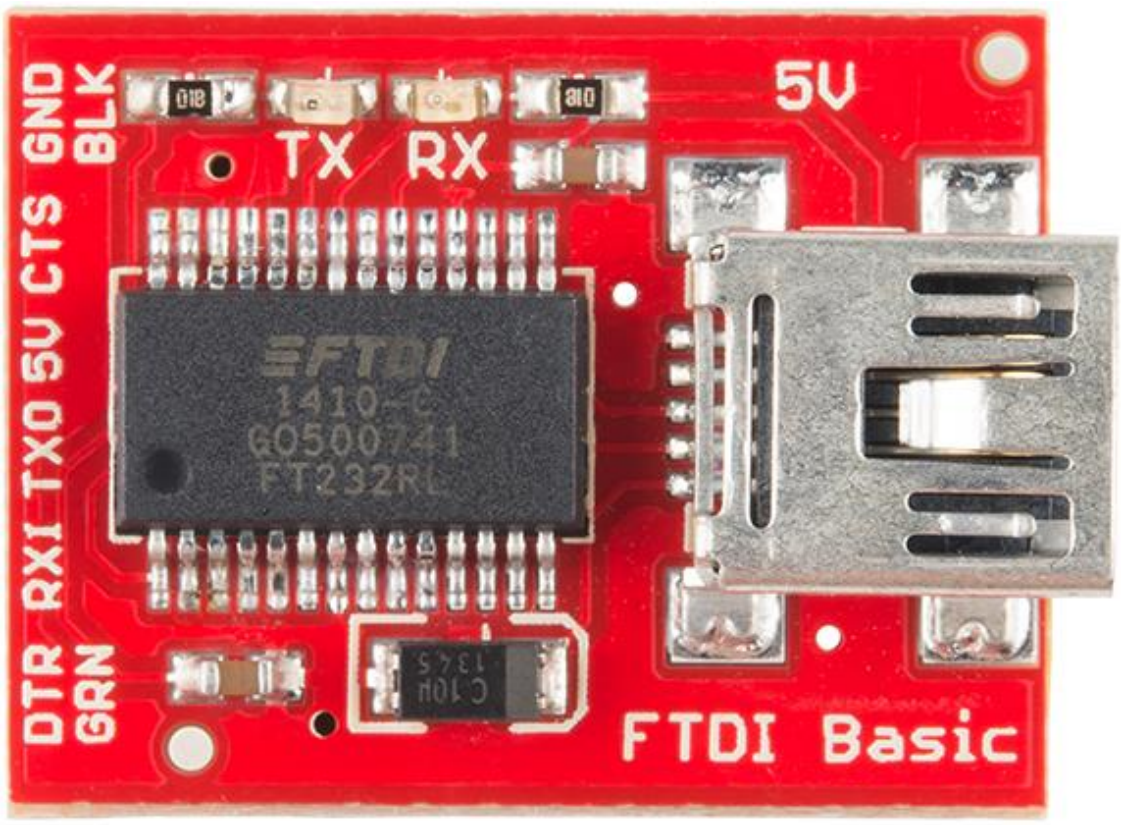

Imagen 3 - Convertidor FTDI - RS232 a USB

I²C (Inter-Integrated Circuit)

Es un protocolo síncrono que facilita la comunicación entre dispositivos utilizando solo dos líneas de conexión: una línea para una señal de reloj (SCL) y otra para los datos (SDA). Al igual que la interfaz SPI, estas líneas no necesitan revertirse.

Cada dispositivo utilizado en este protocolo tiene su propia dirección, lo que permite conectar hasta 128 dispositivos hacia un solo micro-controlador. Algunos ejemplos son:

- Relojes de tiempo real
- Sensores de adquisición de datos
- Expansiones de entradas/salidas
- Circuitos driver para LED's
- Circuitos driver para pantallas o displays

El software de Arduino contiene una librería que implementa este protocolo, que es compatible con las funciones de flujo de entrada y salida de C++.

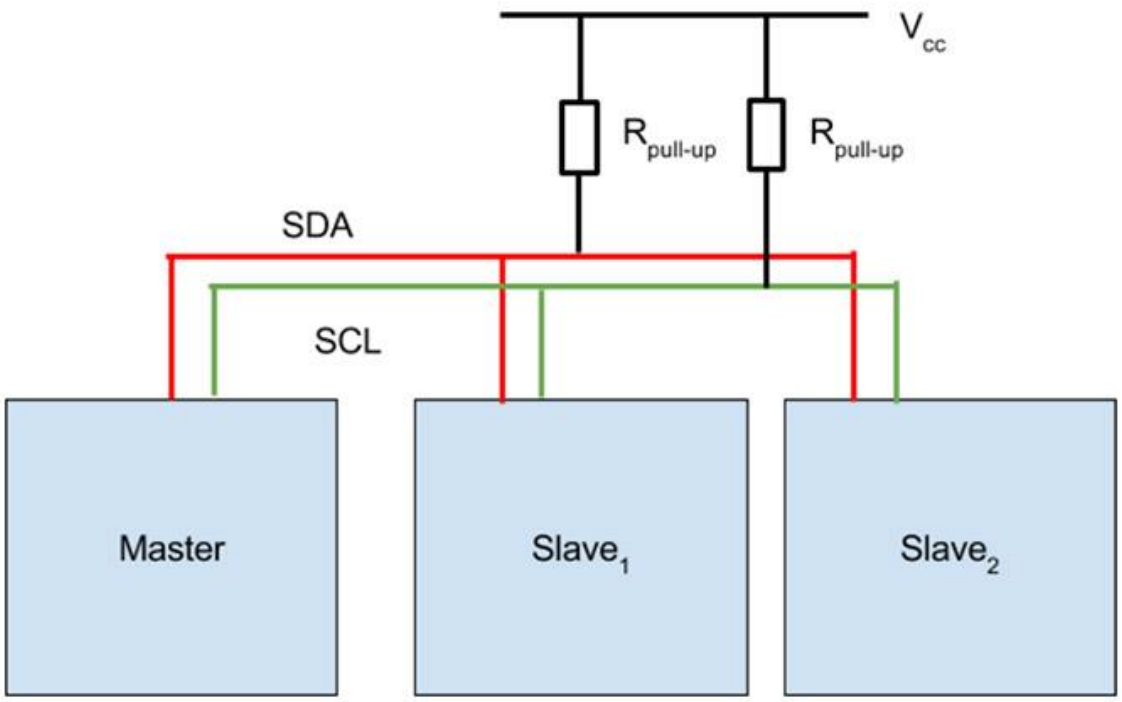

Imagen 4 - Diagrama de conexiones I2C

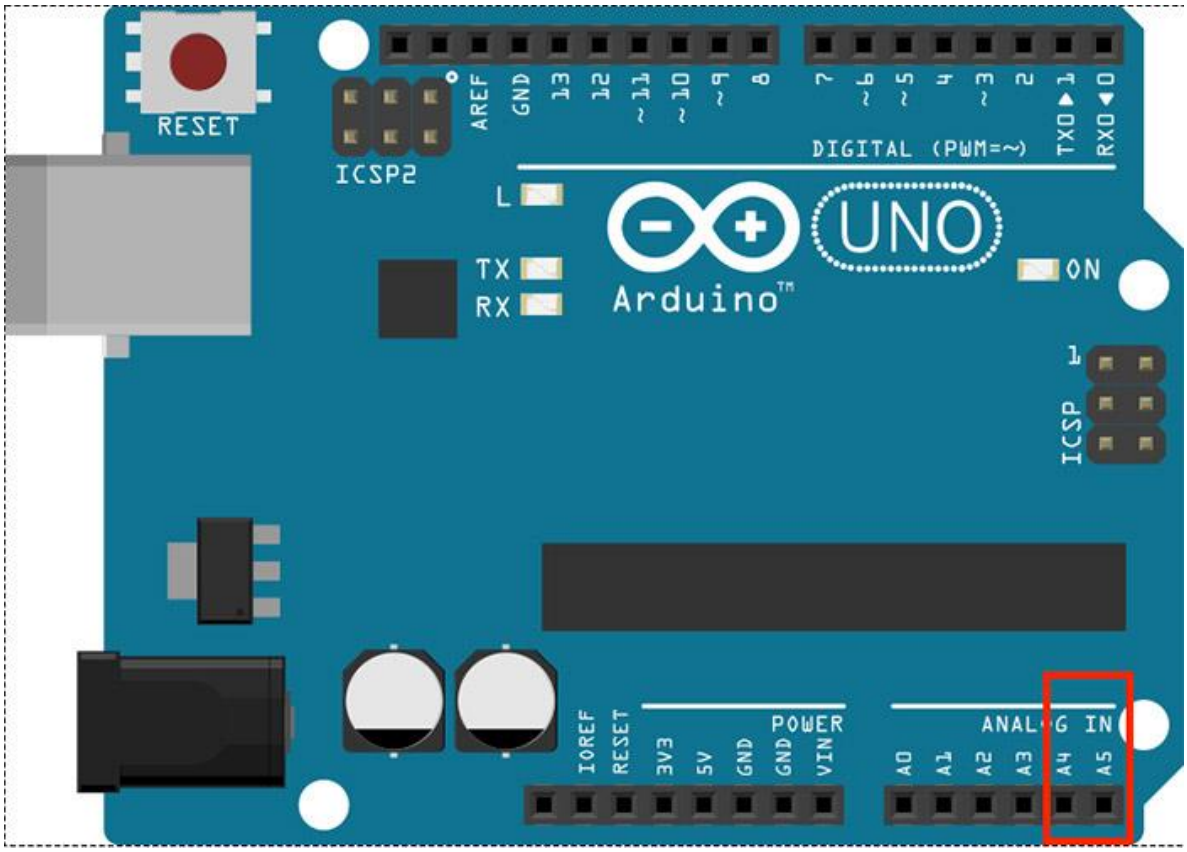

Imagen 5 - Ubicación de los pines I²C en Arduino UNO

SPI (Serial Peripheral Interface)

El protocolo SPI es un bus de datos síncrono que opera en modo full dúplex. Los dispositivos que utilizan este protocolo se configuran en modo maestro/esclavo, donde el dispositivo maestro inicia la transmisión de datos. Se pueden conectar varios dispositivos esclavo con líneas de selección (chip select).

Este protocolo utiliza cuatro líneas de conexión para establecer comunicación con dispositivos, las cuales son:

- MOSI (“Master Out Slave In”): Línea de transmisión de maestro (salida) - esclavo (entrada).
- SCK (“Clock”): Línea de reloj que define la velocidad, inicio y final de transmisión.
- SS (“Slave Select”): Línea de selección para elegir el dispositivo esclavo.
- MISO (“Master In Slave Out”): Línea de transmisión de esclavo (salida) - maestro (entrada).

El software de Arduino contiene una librería que implementa este protocolo, aunque ciertos periféricos pueden tener su propia librería para tomar ventaja del hardware del Arduino reasignando pines distintos a los predefinidos en la librería ya incluida.

Una característica de este protocolo es que las líneas de selección deben inicializarse en HIGH para prevenir fallos al momento de realizar la primera selección. Esto regularmente es establecido por la librería que se utilice para desarrollar el programa.

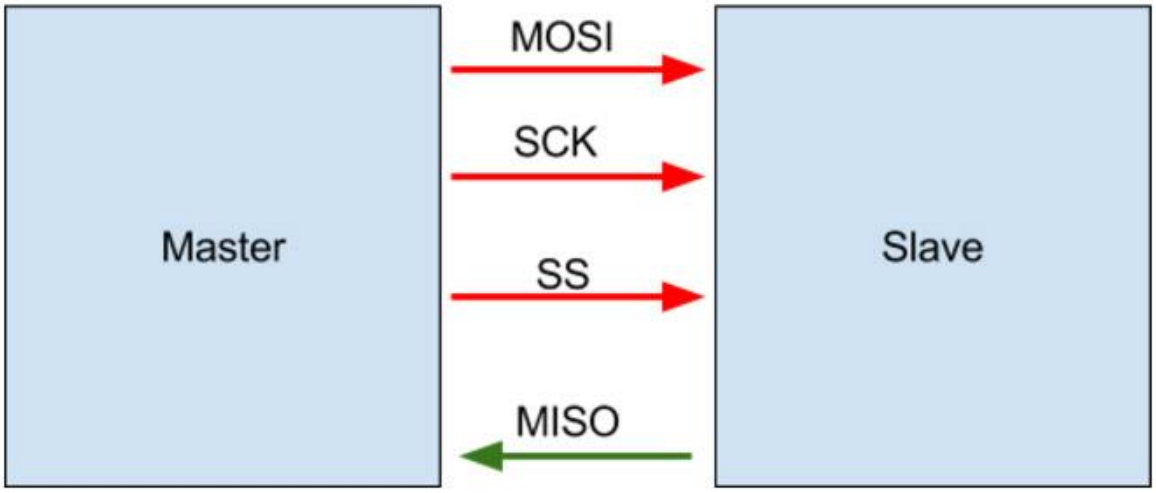

Imagen 6 - Diagrama de conexiones para SPI

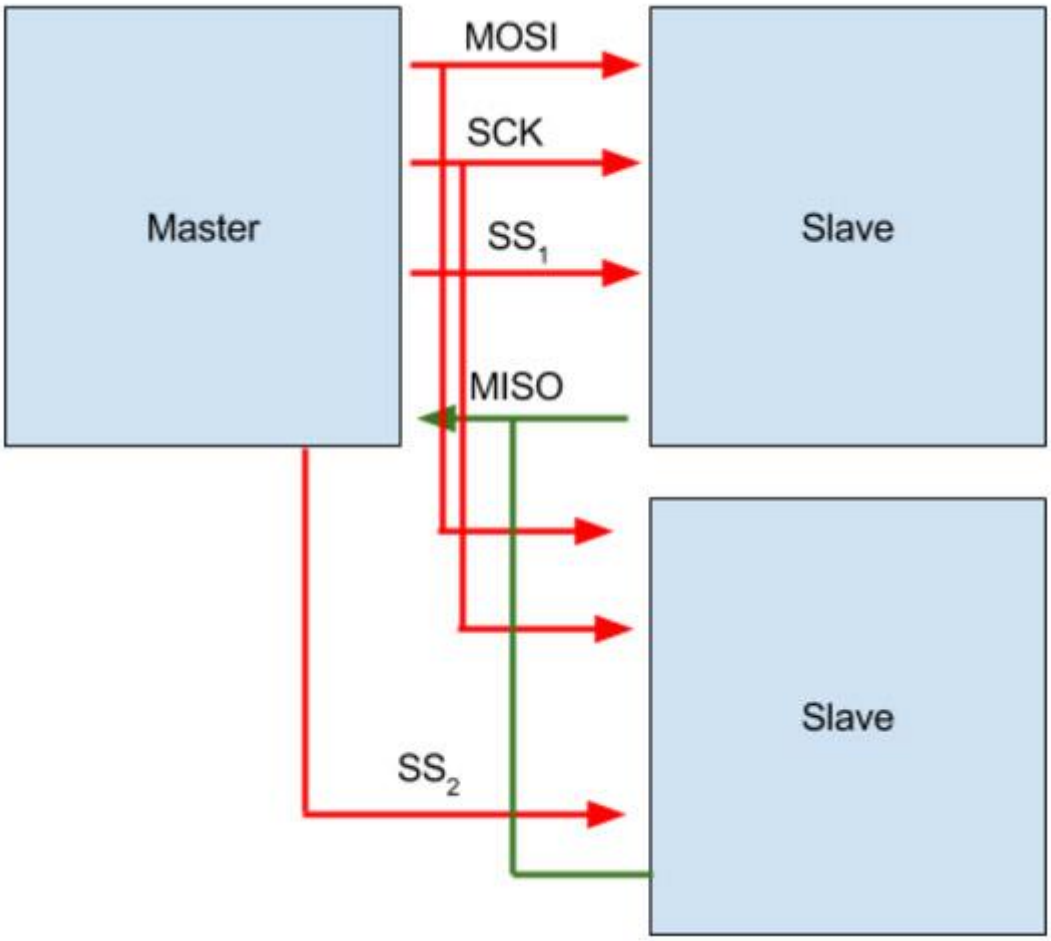

Imagen 7 - Múltiples esclavos conectados a un maestro

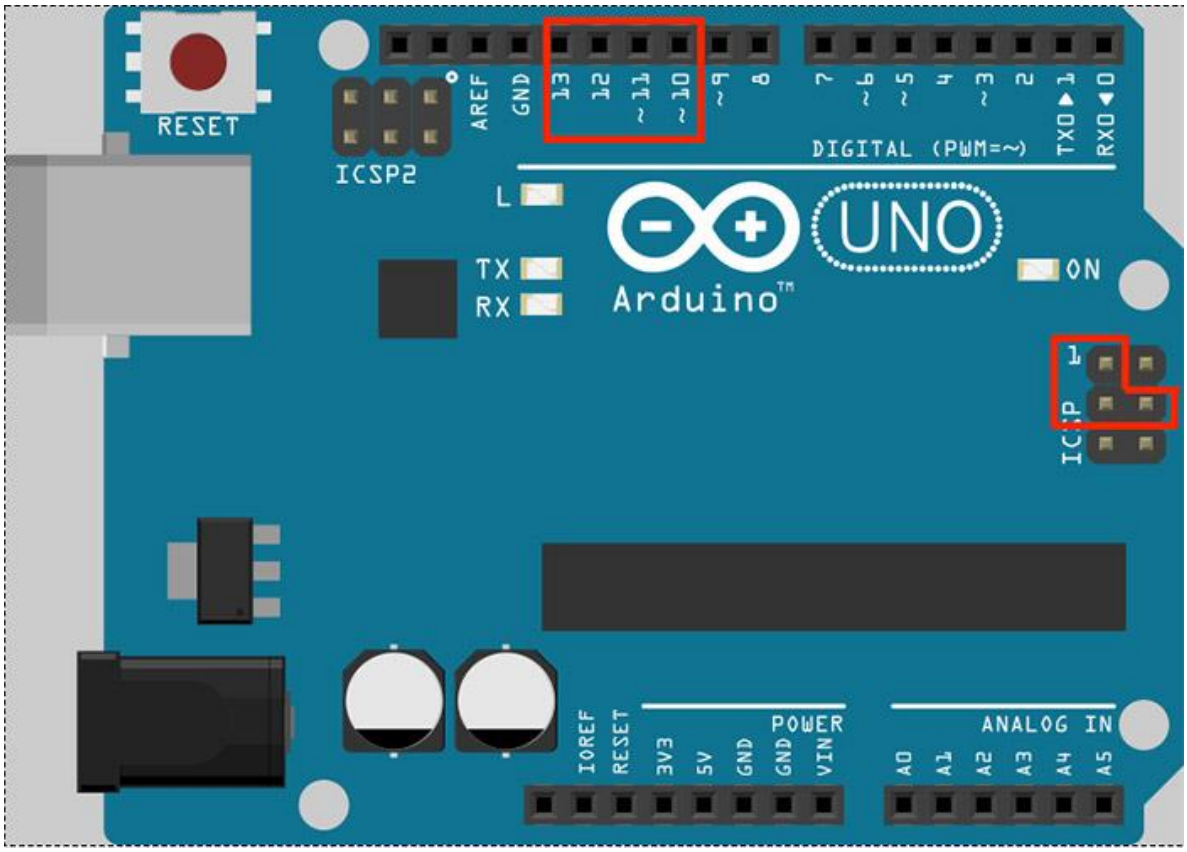

Imagen 8 - Ubicación de los pines SPI en el Arduino UNO

Referencias:

Iyer, R. Arduino Communication Protocols Tutorial (2016, Noviembre 29). Recuperado de: https://www.deviceplus.com/how-tos/arduino-guide/arduino-communication-protocols-tutorial/