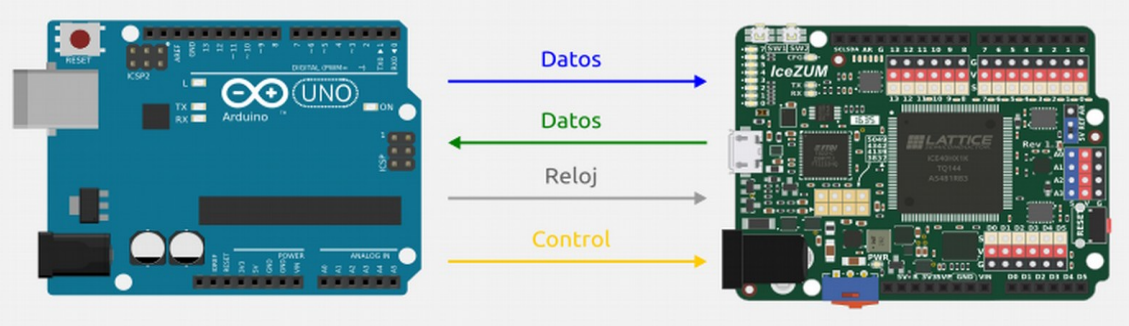
# Comunicaciones: FPGA ↔ Arduino

Mezcla de diseño hardware y software

- Circuito físico, pensamiento hardware realizado con la FPGA.
- Circuito programado, pensamiento software realizado con Arduino.

La forma simple es usar comunicaciones serie síncronas.



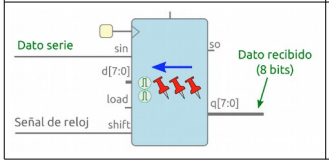
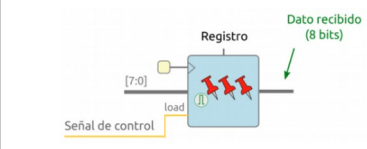
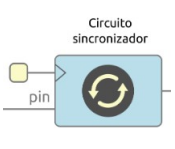
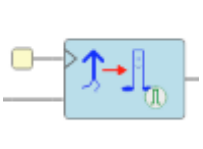

Master Slave

## Comunicación Arduino → FPGA

Esta operación la llamaremos **escritura** y usaremos cuatro cables:

- Envío de datos de Arduino a la FPGA. Usaremos el pin D11 en ambas placas.
- Señal de reloj. D12 en ambas placas.
- Señal de control para poder indicar el fin de la transmisión. D10 en ambas placas.
- Conexión de GND de ambas placas.

Los bloques que usaremos en Icestudio para construir el hardware son:

| | | | |
|---|---|---|---|
| Registro de desplazamiento de 8 bits que recibe los datos por su entrada sin. | Registro para capturar el dato final, cuando se haya recibido completamente. La orden de captura es la que llega por el cable de control. | Sincronizador para minimizar los problemas de metaestabilidad. | Conversor de flancos de subida en tics. Necesario para las entradas shift y load. |
|   |   |   |  |

Los problemas de metaestabilidad suceden cuando la señal exterior cambia en el mismo instante que el flanco de reloj del sistema de la FPGA. En esos casos, el dato capturado no es ni uno ni cero.

El arduino cada vez que realiza una operación de escritura sobre la FPGA está almacenando un dato de 8 bits en el registro de datos. Para realizar esta escritura, el Arduino (software) tiene que realizar estas operaciones:

- Inicialmente las señales de reloj y control deben estar a 0
- Enviar el dato en seríe, empezando por el bit más significativo. Por cada bit enviado debe generar un pulso en la señal de reloj (escritura de un 1, seguida de un 0)
- Una vez enviado el octavo bit, se debe generar otro pulso en la señal de control (un 1 y luego un 0) para que el dato se capture en el registro de datos.



Federico Coca: @fgcoca

Jedi Knight by University of Hardware

Para realizar la escritura desde el Arduino a la FPGA implementamos la función fpga_write(), que envía el dato pasado como parámetro. Se implementa de forma muy fácil usando la función **shiftOut** de la biblioteca de Arduino.

```
//Función fpga_write para pasar el dato como parámetro
void fpga_write(int value) {
  shiftOut(DAT, CLK, MSBFIRST, value); // Envía datos de 8 bits de una vez
  //Generación del pulso de control que indica el final de la transmisión
  digitalWrite(CTRL, HIGH);
  digitalWrite(CTRL, LOW);
}
```

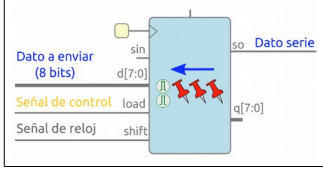
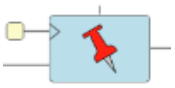
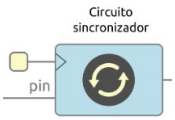
Ejemplo **5-01. Escritura**. Crear el montaje descrito en esta ficha para crear un puerto adicional en Arduino. La idea sería crear un sketch Arduino que escriba los datos 0x55 (0b01010101) y 0xAA (0b10101010) para activar los LEDs on board de la FPGA. Finaliza la transmisión mediante reset de cualquiera de las placas o desconectando el cable de reloj o de datos.

## Comunicación FPGA → Arduino

Esta operación la llamaremos **lectura** y también usaremos cuatro cables:

- Envío de datos de FPGA a Arduino. Usaremos el pin D9 en ambas placas.
- Señal de reloj. D12 en ambas placas. (igual que en escritura)
- Señal de control para poder indicar el fin de la transmisión. D10 en ambas placas. (igual que en escrita)
- Conexión de GND de ambas placas. (igual que en escrita)

Los bloques que usaremos en Icestudio para construir el hardware son:

| Registro de desplazamiento de 8 bits que convierte el dato de paralelo a serie. El dato se envía por salida so. | Biestable D para eliminar los pulsos espúreos, normalmente denominados glitches. | Sincronizador para minimizar los problemas de metaestabilidad. | Conversor de flancos de subida en tics. Necesario para las entradas shift y load. |
|---|---|---|---|
|  |  |  |  |

Para realizar la **lectura**, el Arduino (software) tiene que realizar estas operaciones:

- Inicialmente las señales de reloj y control deben estar a 0
- Envíar un pulso por la señal de control, para que la FPGA capture el dato a enviar
- Recibir el dato en serie, empezando por el bit más significativo. Cada vez que lee un bit debe se genera un pulso en la señal de reloj.
- Al cabo de 8 pulsos de reloj ya tiene el dato completo

Para realizar la lectura de la FPGA desde el Arduino implementamos la función fpga_read(), que devuelve el dato recibido. Se implementa de forma muy fácil usando la función **shiftIn** de la biblioteca de Arduino.

```
byte fpga_read() {
  byte c;
  //Indicar a la FPGA que capture el dato
  digitalWrite(CTRL, HIGH);
  digitalWrite(CTRL, LOW);
  //Recibir el dato
  c =  shiftIn(DAT, CLK, MSBFIRST);
  return c;
}
```

Ejemplo **5-02. Lectura**. Introducir un número de 3 bits mediante interruptores, que se almacena en el registro de datos al pulsar la tecla "ENTER". El arduino lee este registro de datos periódicamente para saber qué dato se ha introducido.
El dato introducido en el registro de datos de la FPGA es mostrado en el monitor serie de Arduino.