Diseño con microcontroladores

Comunicación I2C y SPI

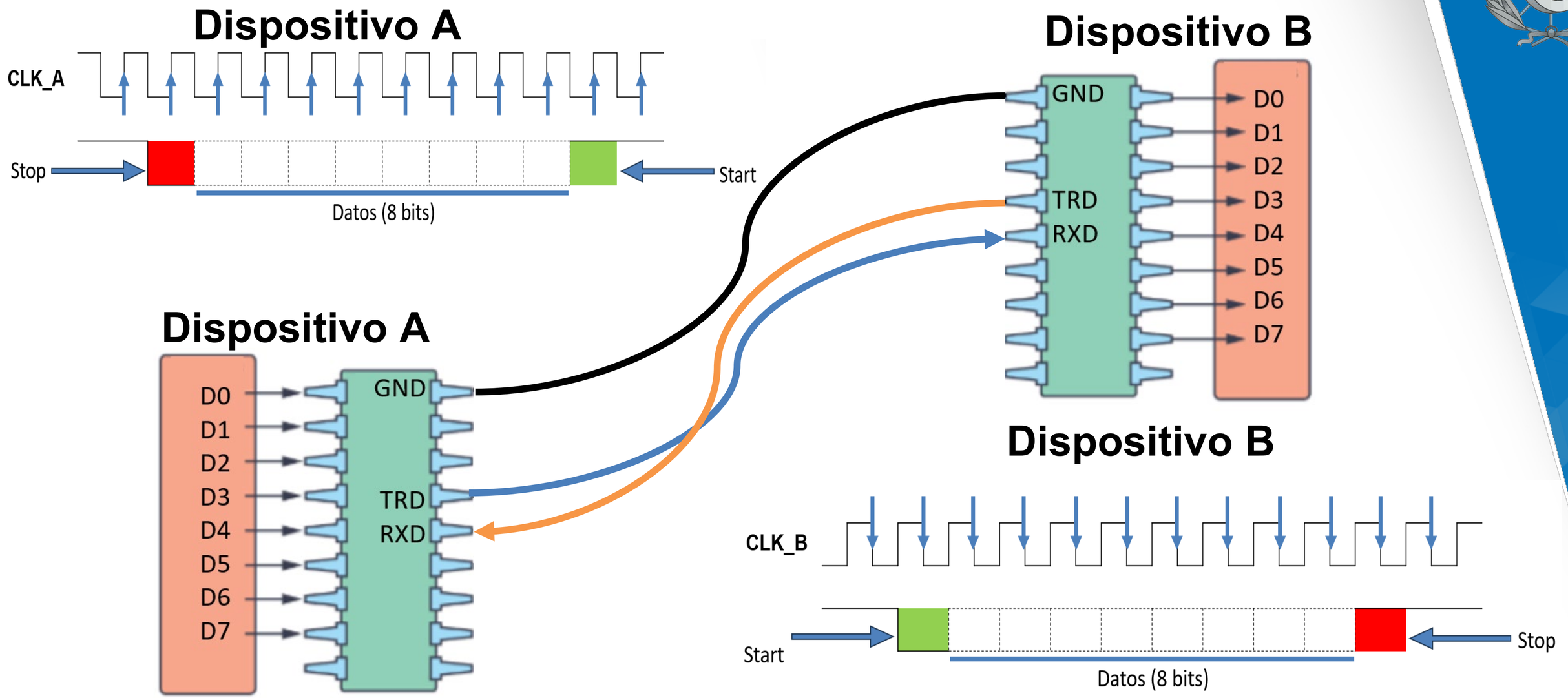
Marco Xavier Rivera González
marco.rivera@upm.es

Escuela Técnica Superior de Ingenieros Informáticos
Universidad Politécnica de Madrid

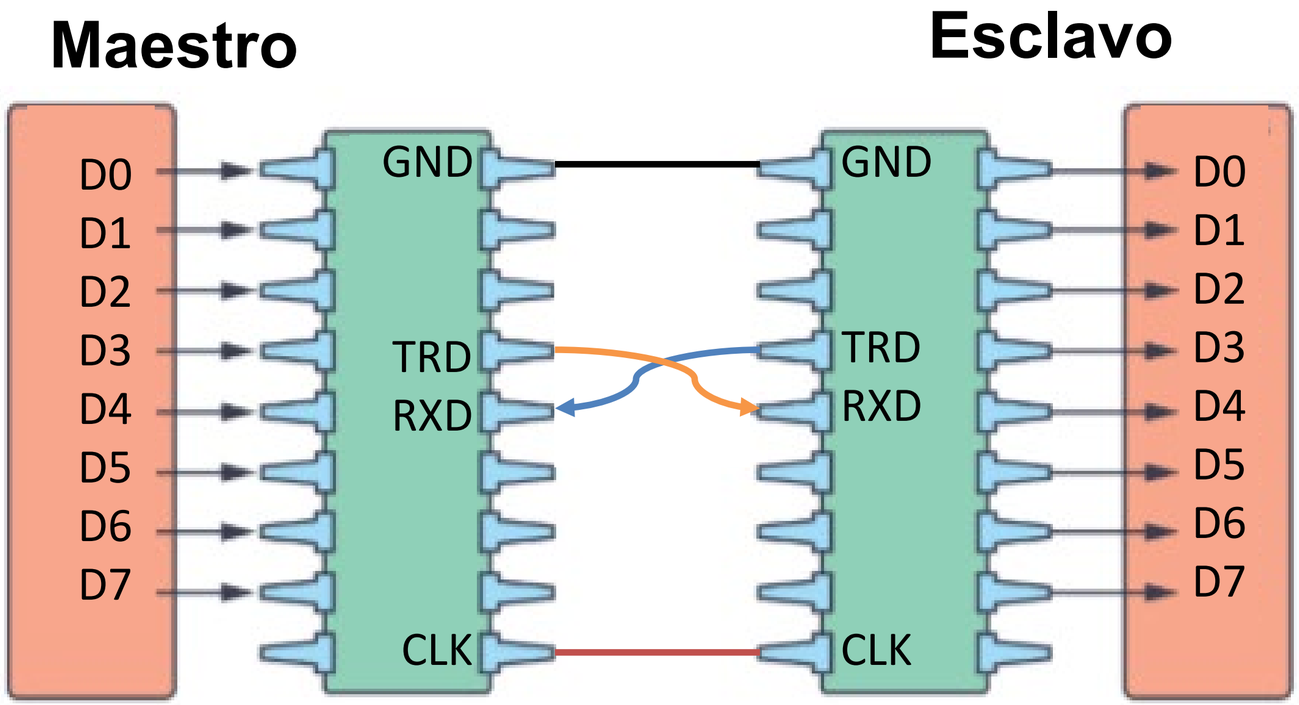
Escuela Técnica Superior de
Ingenieros Informáticos

UNIVERSIDAD POLITÉCNICA DE MADRID

Comunicación Síncrona vs Asíncrona



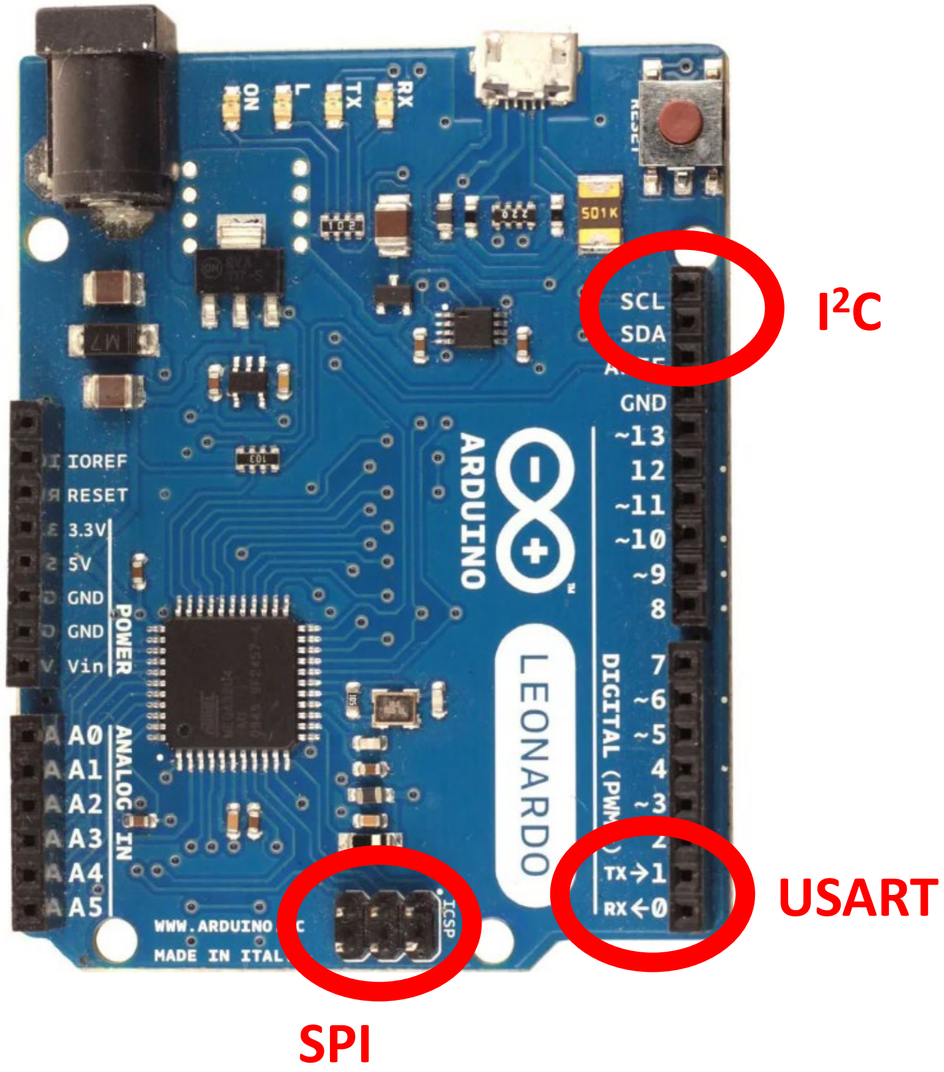
Comunicación Síncrona vs Asíncrona



Comunicación Síncrona:

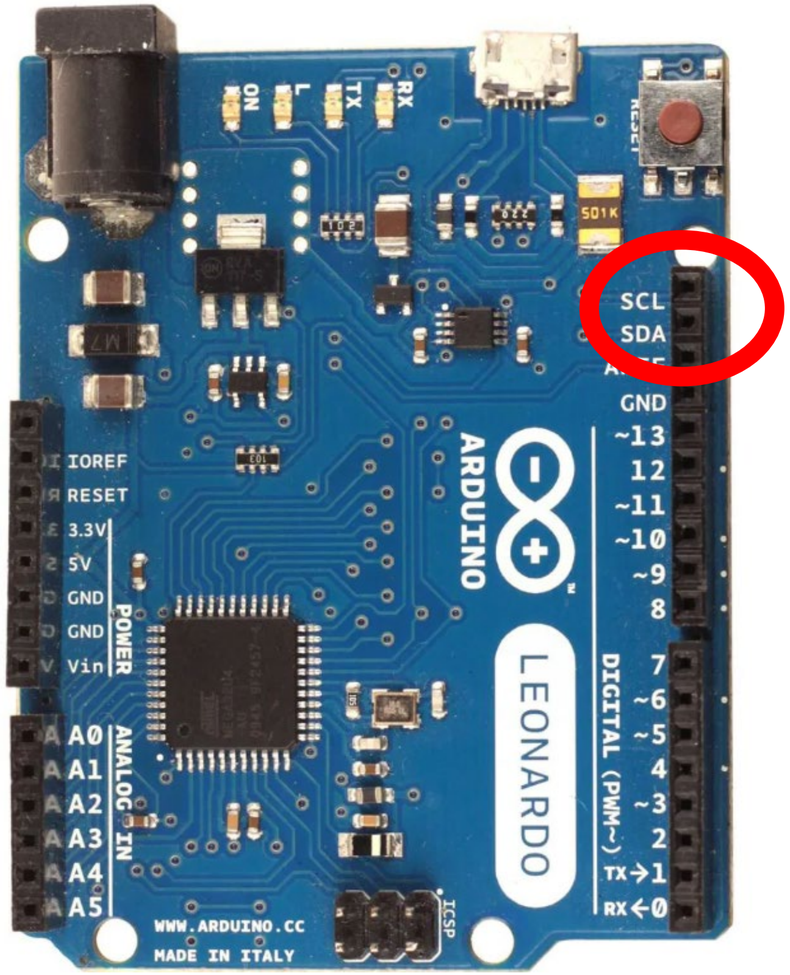
- **Con reloj (Clock/CLK):** Hay un cable extra dedicado exclusivamente a marcar el ritmo (tic-tac-tic-tac).
- **Maestro-esclavo:** El Maestro dicta la velocidad. El Esclavo solo obedece el ritmo. No hace falta configurar "baudios" en el esclavo. Si el Maestro deja de emitir pulsos de reloj, la comunicación se congela y no se pierden datos.

Modos de transmisión:



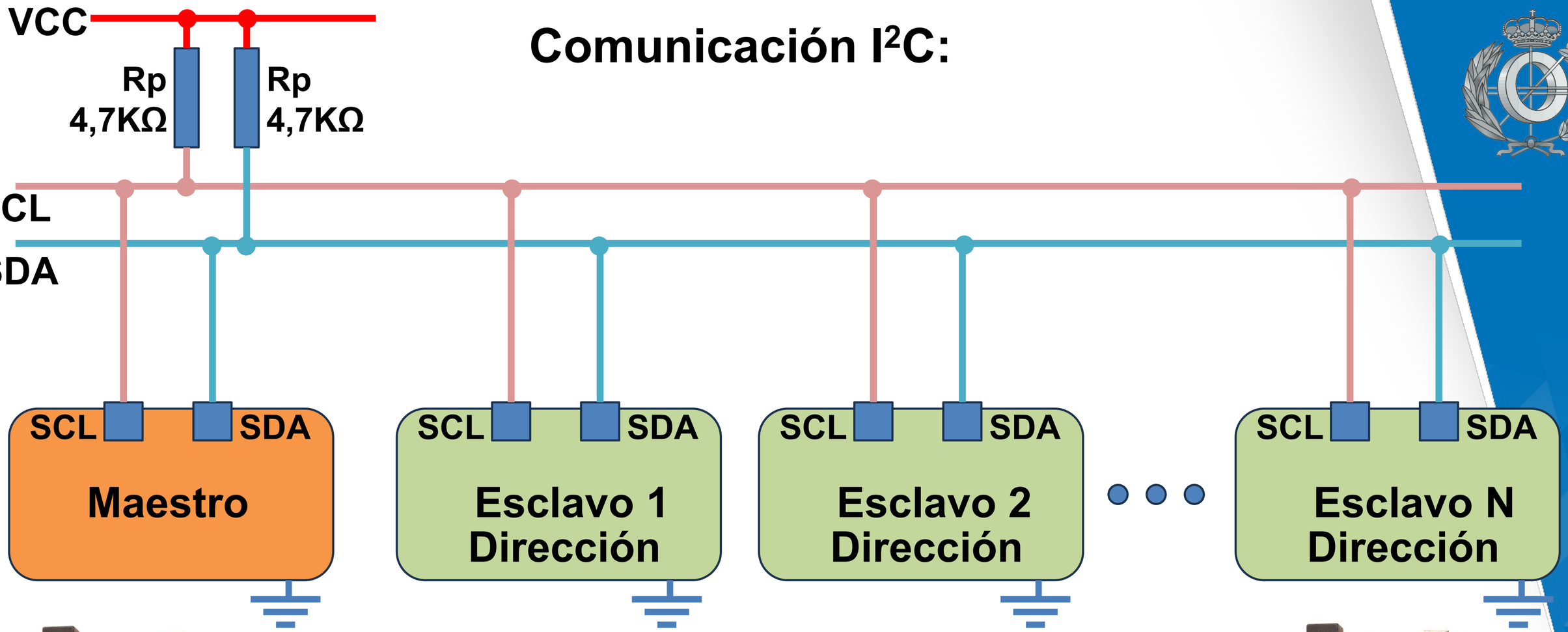
- **Simplex:** Comunicación en un solo sentido (ej. Radio FM, o un sensor que solo envía datos y nunca recibe órdenes).
- **Half-Duplex (Ej. I2C):** La información viaja en ambas direcciones, pero **no al mismo tiempo**. O hablas, o escuchas. Usan el mismo cable para ir y volver.
- **Full-Duplex (Ej. SPI):** Puedes enviar y recibir datos **simultáneamente**. Requiere cables separados para ida (MOSI) y vuelta (MISO).

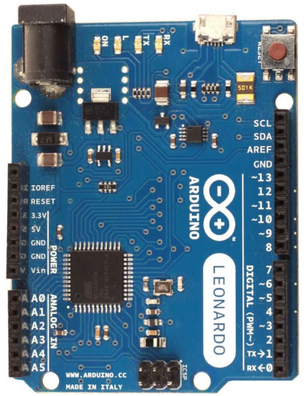
Comunicación I²C:

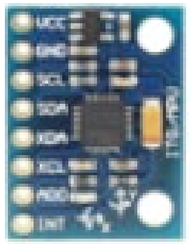


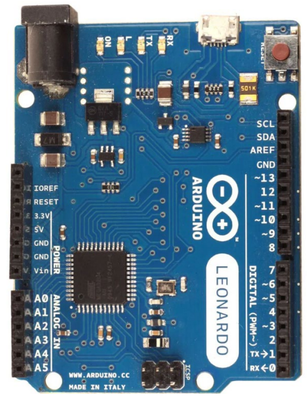
- **I²C** (Inter-Integrated Circuit) o **TWI** (Two Wired Interface).
  - Comunicación Maestro-Esclavo
- **El Bus:**
  - Consiste en solo dos líneas compartidas por todos los dispositivos:
    - *SDA (Serial Data):* Por aquí viajan los bits (0s y 1s).
    - *SCL (Serial Clock):* La señal de reloj que marca cuándo leer el bit.
- **Lógica Negativa (Active Low):**
  - Las líneas I2C están "normalmente en ALTO" (High, 5V).
    - *Para enviar un **0***, el dispositivo conecta la línea a Tierra (GND).
    - *Para enviar un **1***, el dispositivo simplemente "suelta" la línea y deja que la resistencia de pull-up la suba ponga a 5V.

Comunicación I²C:

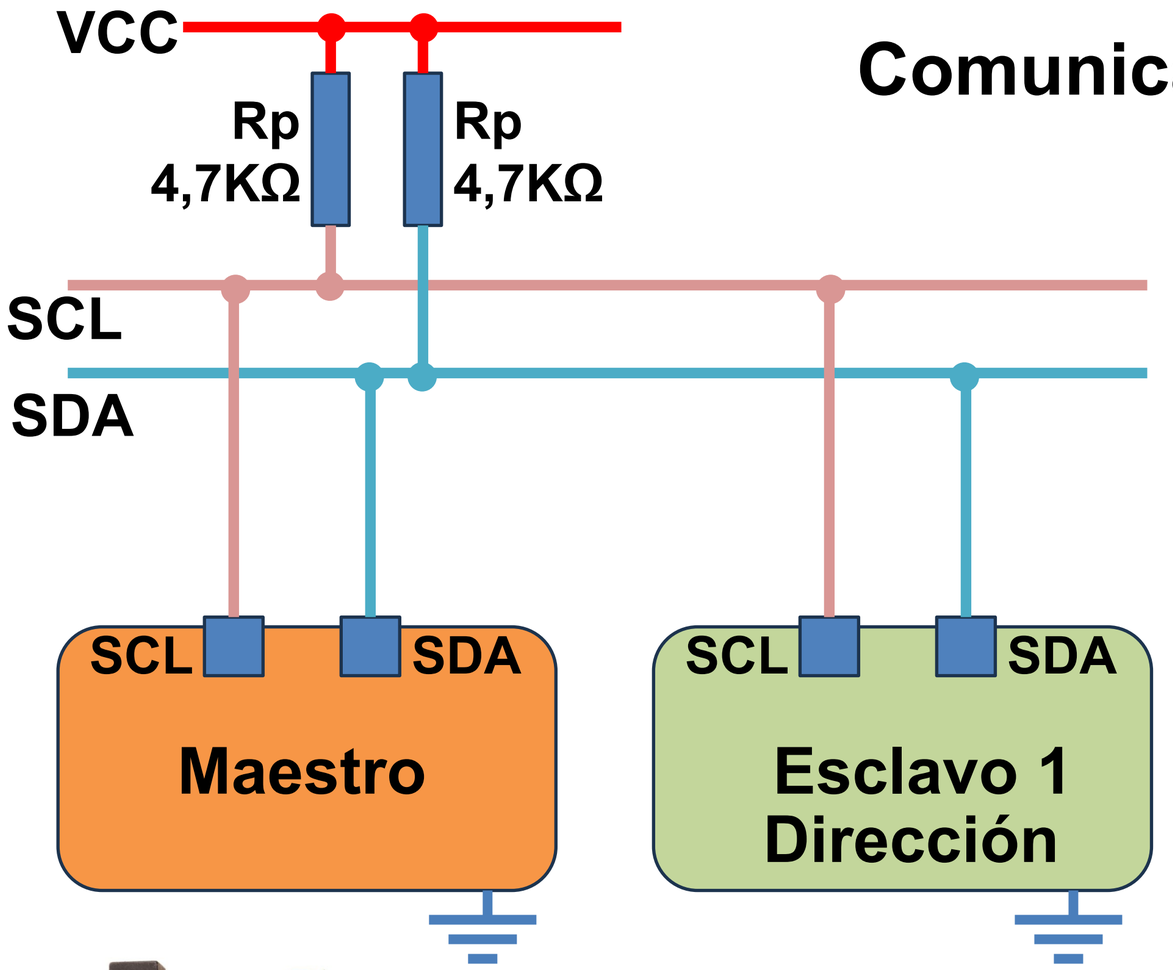


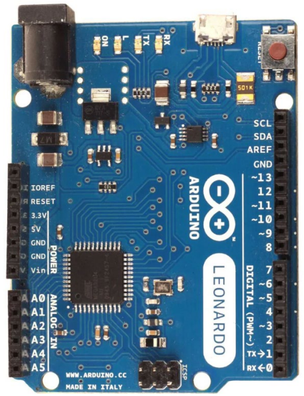






Comunicación I²C:

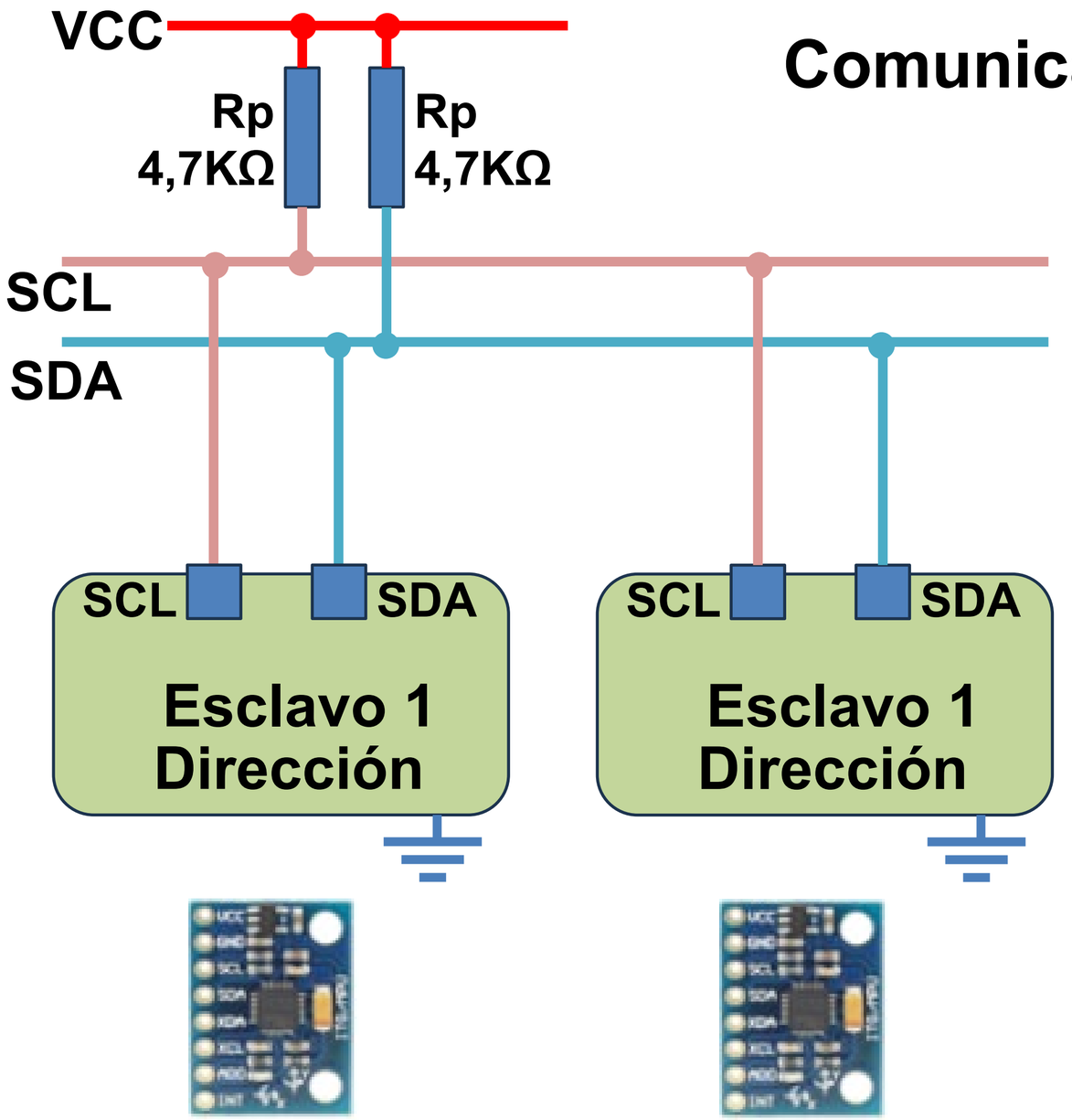




Las Resistencias Pull-Up (Imprescindibles)

- **Función:** Mantener las líneas en 5V cuando nadie está hablando.
- **Valor Típico:** 4.7kΩ (para 5V estándar). Las resistencias de pull up internas de Arduino están en el rango de 20kΩ-50kΩ, lo cual permitirían conexiones máximas de entre 10 y 20 cm

Comunicación I²C:



Direccionamiento

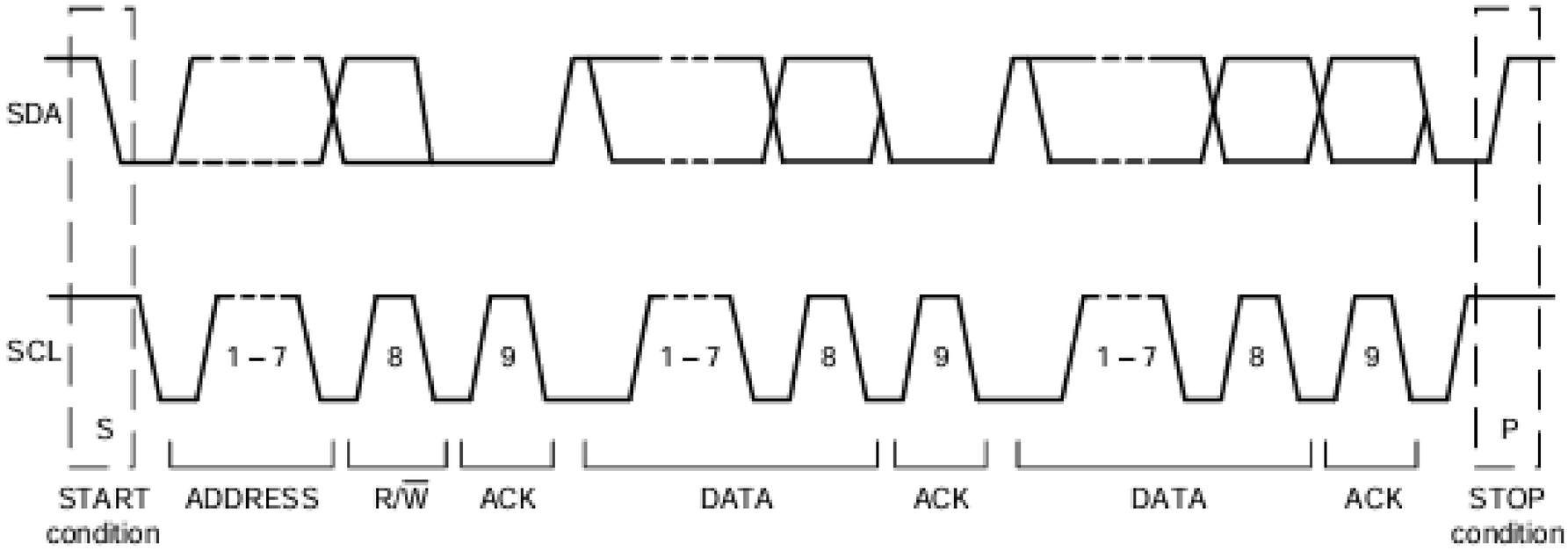
- **Espacio de direcciones:** 7 bits permiten 128 direcciones (0 a 127).
- **Direcciones reservadas:** Las direcciones 0 a 7 y 120 a 127 suelen estar reservadas para propósitos especiales.
- **Direcciones Fijas vs. Configurables:**
- Sensores tienen una dirección base, pero tienen accesible un o mas pines físicos para conectar a GND (‘0’) VCC (‘1’) para ajustar su dirección.

Comunicación I^2C:

La Estructura del Mensaje (La Trama de Datos)

1. **Start Condition (Inicio):**
   - El Maestro da una señal de Inicio (‘0’).
2. **Byte de Dirección (7 bits + 1 bit R/W):**
   - El Maestro envía 7 bits que identifican al esclavo.
   - El **8º bit** indica la acción:
     - **0 (LOW):** Write (Maestro quiere escribir datos).
     - **1 (HIGH):** Read (Maestro quiere leer datos).
3. **El Bit ACK/NACK (El Acuse de Recibo):**
   - **Crucial:** Después de cada 8 bits enviados, el que envía (sea Maestro o Esclavo) suelta la línea SDA.
   - El que recibe debe bajar la línea SDA durante el 9º pulso de reloj.
     - **ACK (Low):** "Te escuché y entendí".
     - **NACK (High):** "No te escuché", "No existo" o "Estoy ocupado".
4. **Datos:** Se envían bytes de 8 bits seguidos de sus respectivos ACKs.
5. **Stop Condition (Fin):**
   - El Maestro da una señal de parada (‘1’).

Comunicación I²C: escritura:



Single-Byte Write Sequence

Master	S	AD+W		RA		DATA		P
Slave			ACK		ACK		ACK	

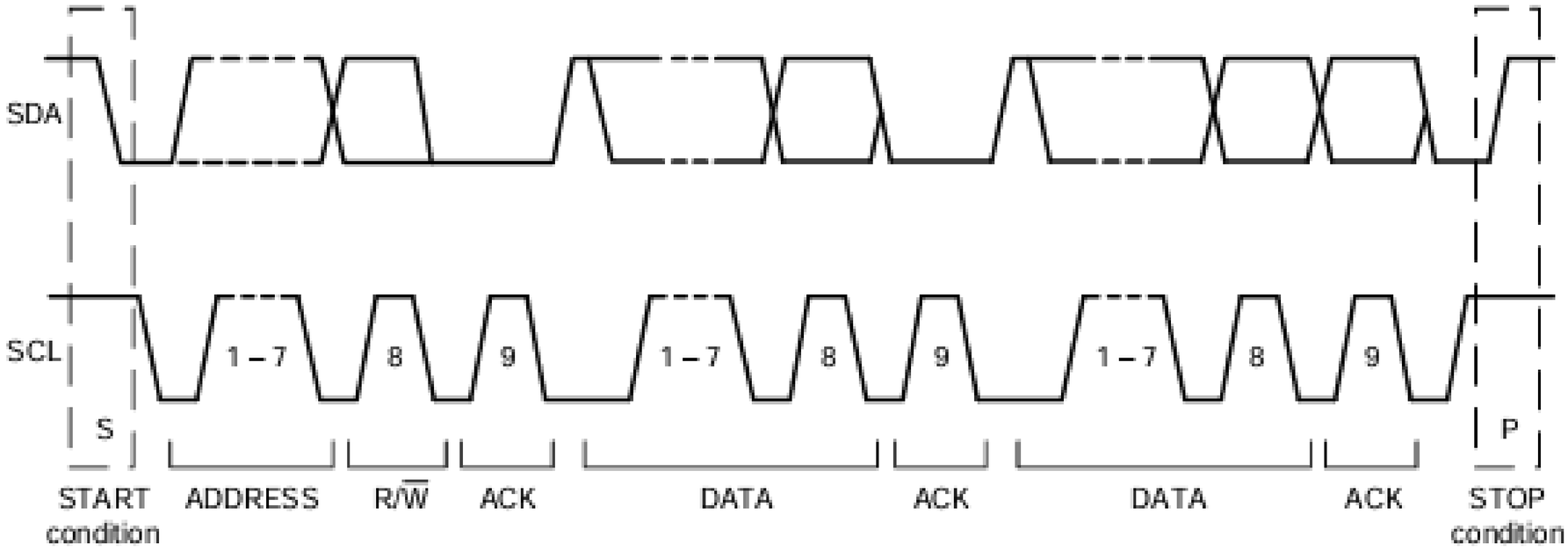
Burst Write Sequence

Master	S	AD+W		RA		DATA		DATA		P
Slave			ACK		ACK		ACK		ACK	

Signal	Description
S	Start Condition: SDA goes from high to low while SCL is high
AD	Slave I²C address
W	Write bit (0)
R	Read bit (1)
ACK	Acknowledge: SDA line is low while the SCL line is high at the 9th clock cycle
NACK	Not-Acknowledge: SDA line stays high at the 9th clock cycle
RA	MPU-60X0 internal register address
DATA	Transmit or received data
P	Stop condition: SDA going from low to high while SCL is high

Datos tomados de las hojas de datos del sensor MPU6050

Comunicación I²C: lectura:



Single-Byte Read Sequence

Master	S	AD+W		RA		S	AD+R			NACK	P
Slave			ACK		ACK			ACK	DATA		

Burst Read Sequence

Master	S	AD+W		RA		S	AD+R			ACK		NACK	P
Slave			ACK		ACK			ACK	DATA		DATA		

Signal	Description
S	Start Condition: SDA goes from high to low while SCL is high
AD	Slave I²C address
W	Write bit (0)
R	Read bit (1)
ACK	Acknowledge: SDA line is low while the SCL line is high at the 9th clock cycle
NACK	Not-Acknowledge: SDA line stays high at the 9th clock cycle
RA	MPU-60X0 internal register address
DATA	Transmit or received data
P	Stop condition: SDA going from low to high while SCL is high

Datos tomados de las hojas de datos del sensor MPU6050

Comunicación I²C: Funciones librería Wire.h: Lectura

```
void setup() {
  Wire.begin();
  Wire.setClock(bauds);
}
```

Single-Byte Write Sequence

Master	S	AD+W		RA		DATA		P
Slave			ACK		ACK		ACK	

Burst Write Sequence

Master	S	AD+W		RA		DATA		DATA		P
Slave			ACK		ACK		ACK		ACK	

```
Wire.beginTransmission(Add);
Wire.write(register/data);
Wire.write(data);
Wire.endTransmission(true);
```

```
Wire.beginTransmission(Add);
Wire.write(register/data);
Wire.write(data);
Wire.write(data);
  .
  .
  .
Wire.write(data);
Wire.endTransmission(true);
```

Comunicación I²C: Funciones librería Wire.h: Escritura

```
void setup() {
  Wire.begin();
  Wire.setClock(bauds);
}
```

Single-Byte Read Sequence

Master	S	AD+W		RA		S	AD+R			NACK	P
Slave			ACK		ACK			ACK	DATA		

Burst Read Sequence

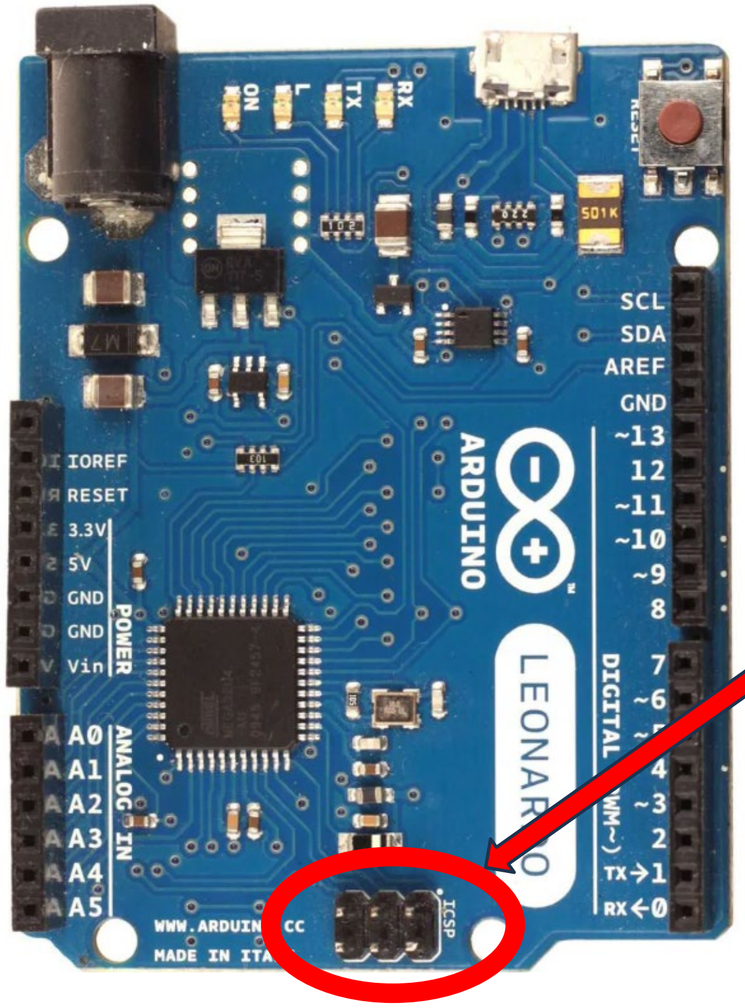
Master	S	AD+W		RA		S	AD+R			ACK		NACK	P
Slave			ACK		ACK			ACK	DATA		DATA		

```
Wire.beginTransmission(Add);
Wire.write(register/data);
Wire.endTransmission(false);

Wire.requestFrom(add, N, parada);

for (char i=0;i<n;i++)
{
  dato[j]=Wire.read();
}
```

```
byte error = Wire.endTransmission();
```

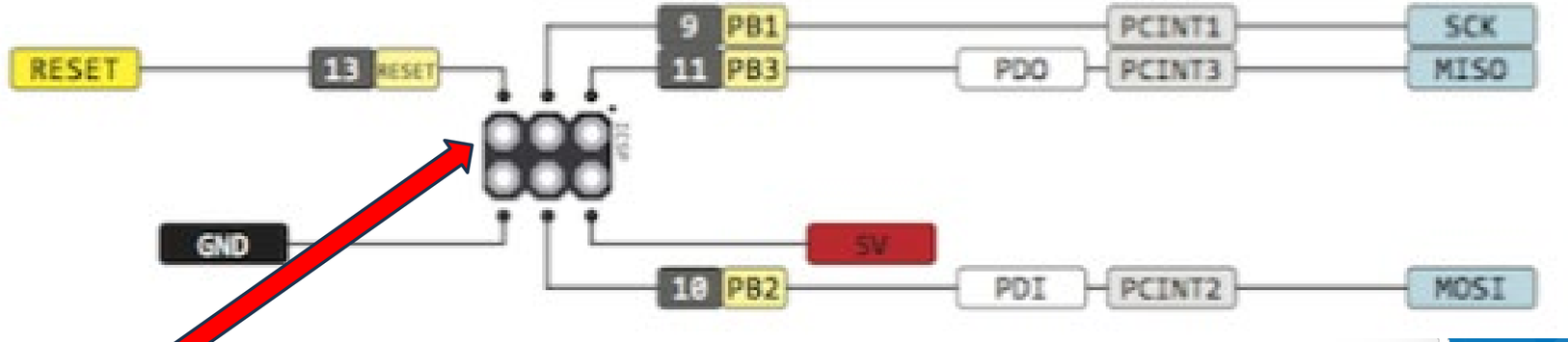
- **0:** Éxito, se recibió el ACK
- **1:** Buffer lleno
- **2:** NACK en dirección.
- **3:** NACK en Datos
- **4:** Otro error

Comunicación SPI:



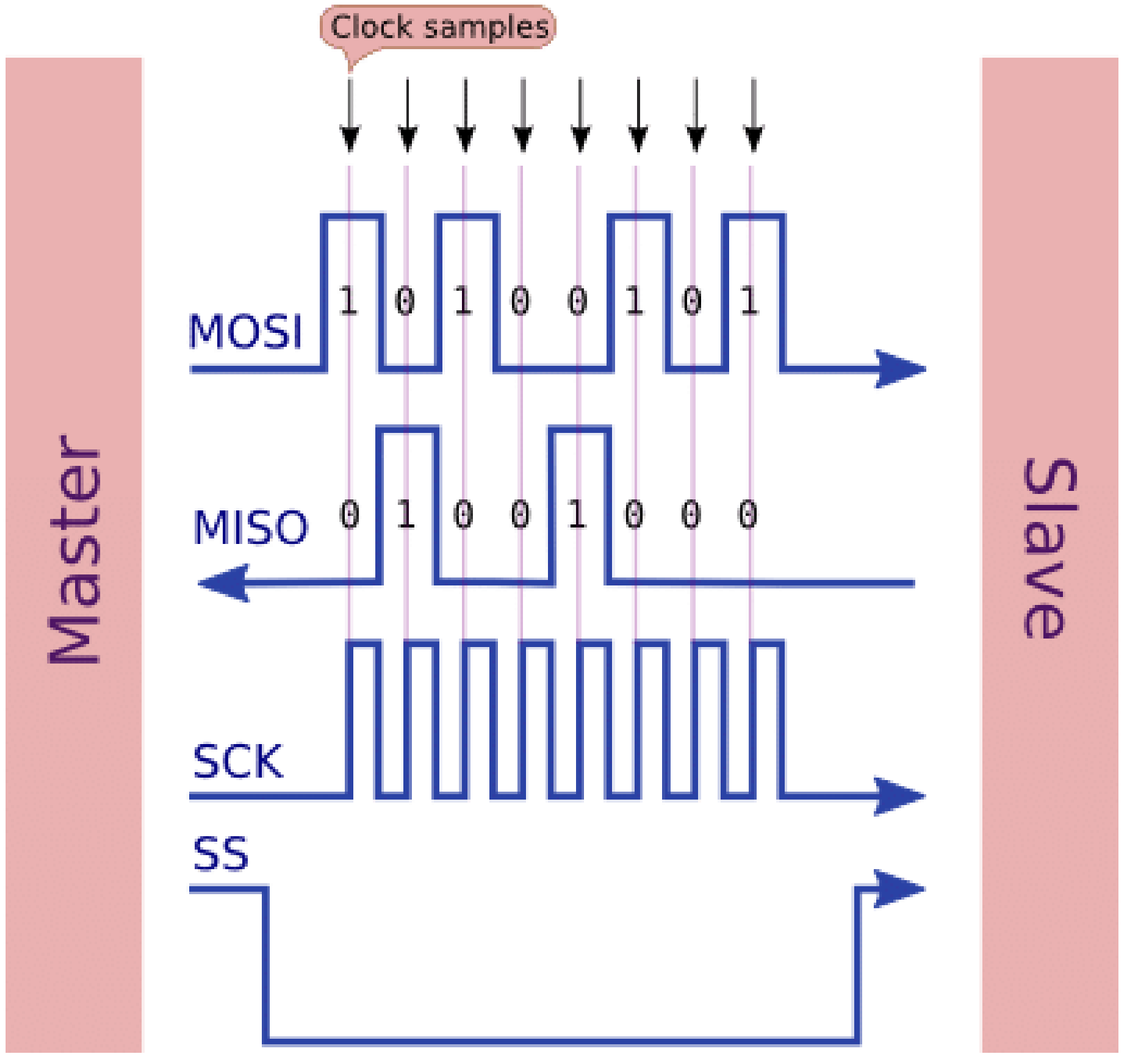
SPI

- **SPI** (Serial Periferial Interface)
  - Mayor velocidad
  - Grandes cantidades de datos



- **Comunicación Full dúplex**
  - **MISO:** Master Input, Salve Output
  - **MOSI:** Master Output, Slave Input
  - **SCK:** Serial Clock
  - **SS/CS:** Slave Select / Chip Select

Comunicación SPI:

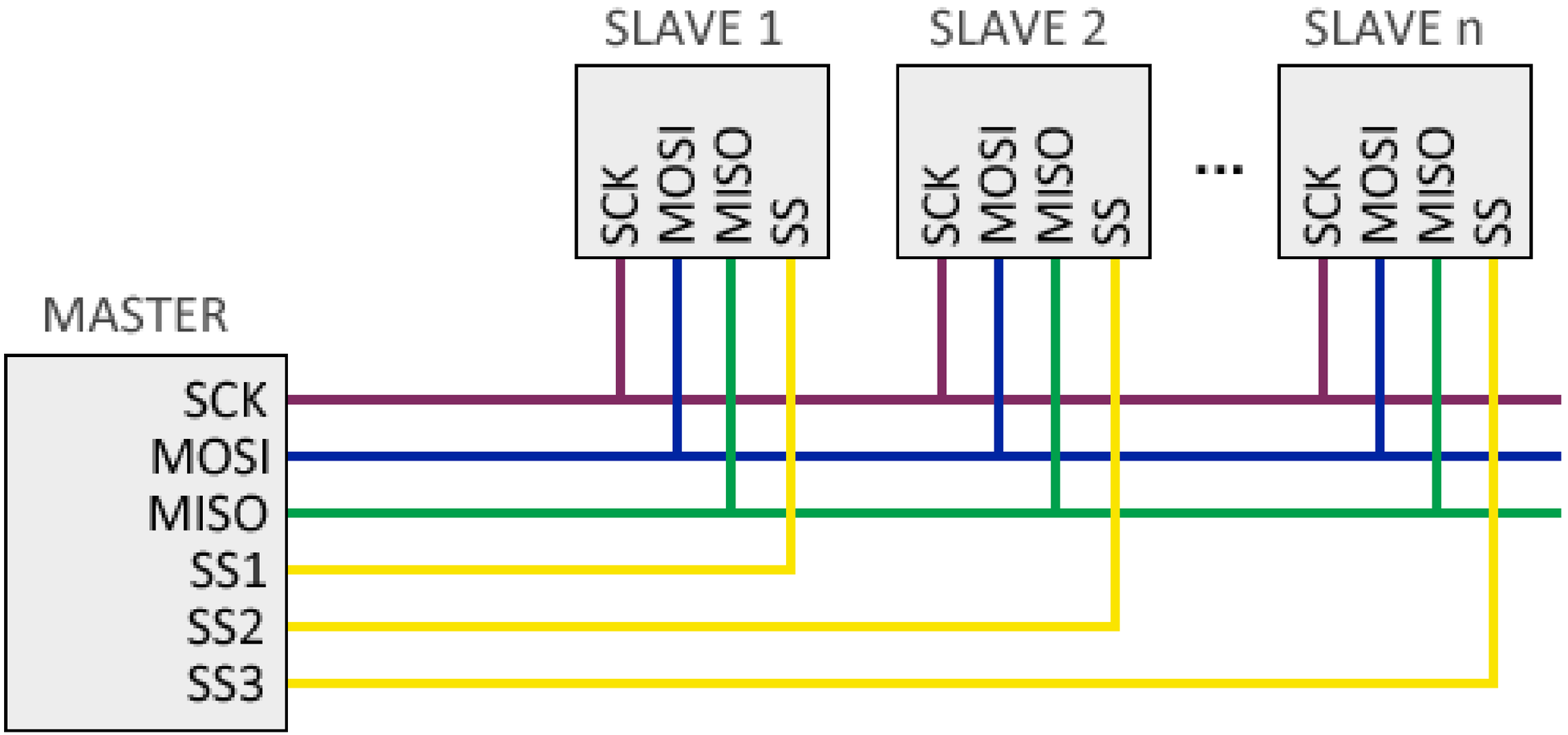


No existe "solo leer" o "solo escribir". Siempre es un intercambio

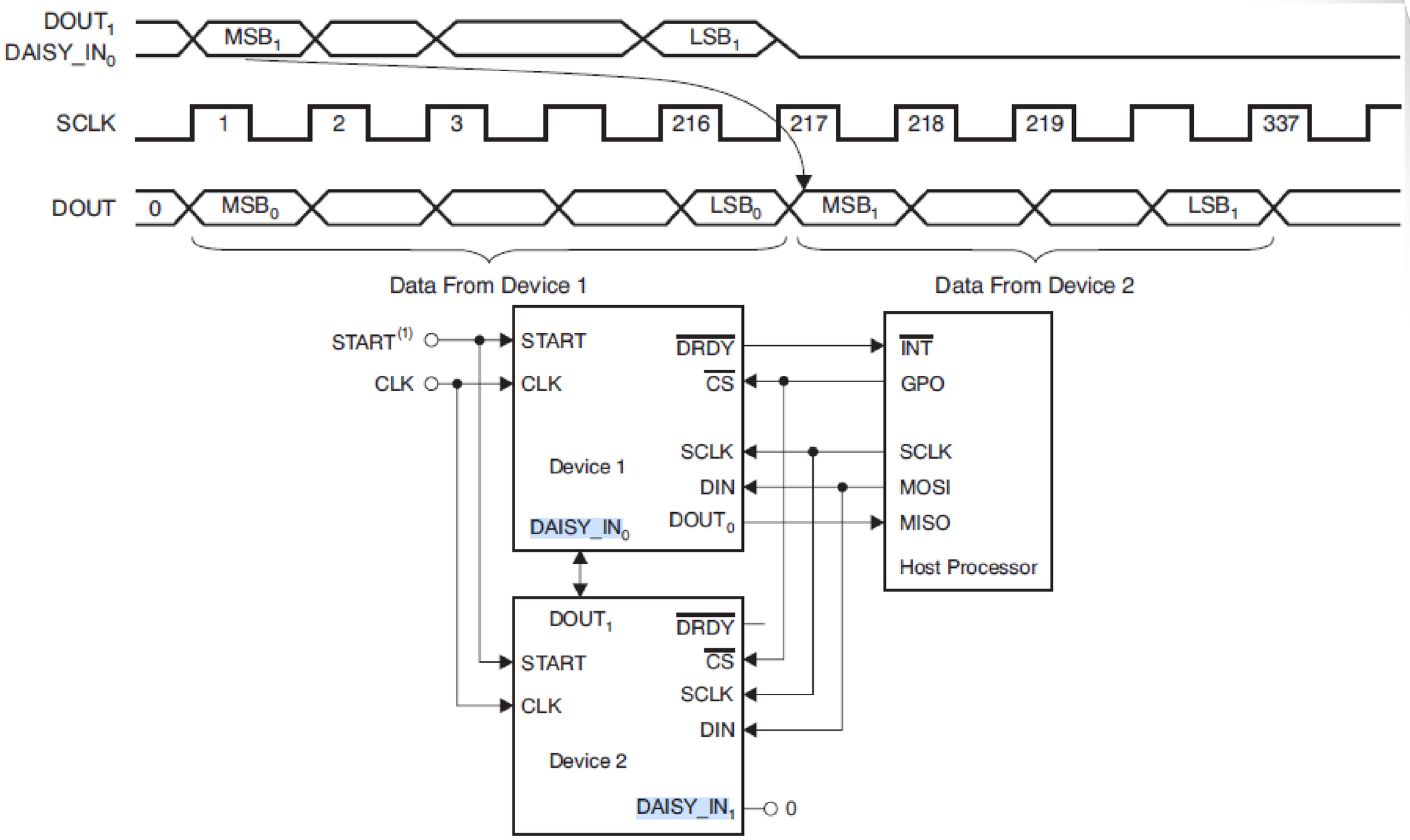
PULSO 1 DE RELOJ:

- El [0] del Maestro ---> Entra en el [7] del Esclavo
- El [0] del Esclavo ---> Entra en el [7] del Maestro
- Todos los demás bits se corren un puesto

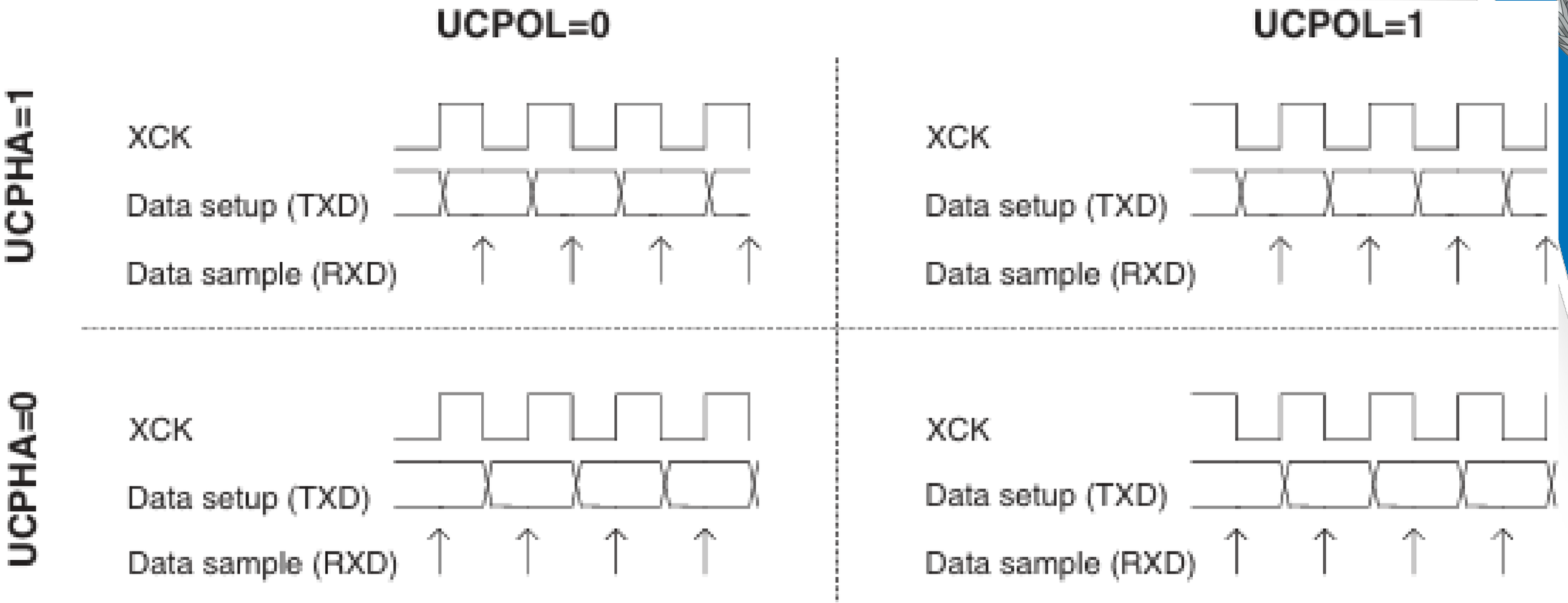
Comunicación SPI: conexión paralelo



Comunicación SPI: Conexión en cadena (Daisy Chain)



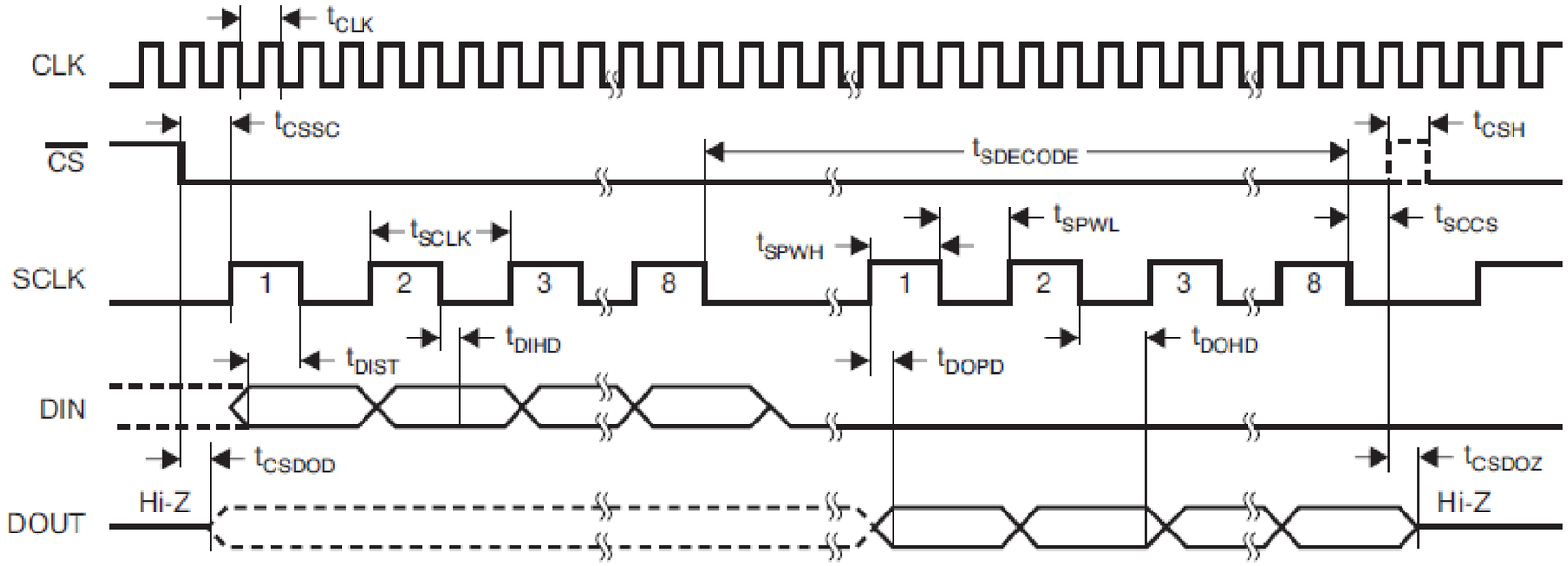
Comunicación SPI: Modos



- **CPOL (Polaridad):** ¿El reloj descansa en 0V o en 5V?
- **CPHA (Fase):** ¿Flanco de subida, o flanco de bajada?

Comunicación SPI: Modos

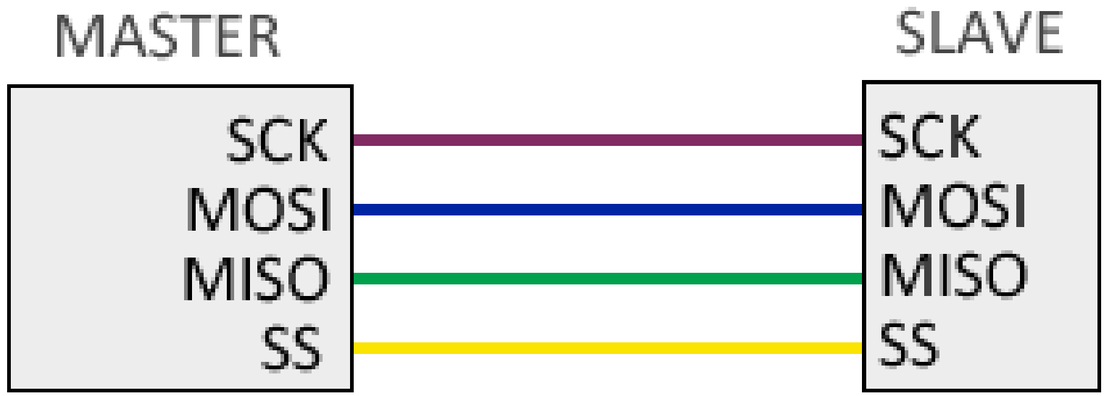
PARAMETER		2.7 V ≤ DVDD ≤ 3.6 V		1.8 V ≤ DVDD ≤ 2.0 V		UNIT
		MIN	MAX	MIN	MAX	
t_{DOHD}	Hold time, SCLK falling edge to invalid DOUT	10		10		ns
t_{DOPD}	Propagation delay time, SCLK rising edge to DOUT valid		17		32	ns
t_{CSDOD}	Propagation delay time, $\overline{CS}$ low to DOUT driven	10		20		ns
t_{CSDOZ}	Propagation delay time, $\overline{CS}$ high to DOUT Hi-Z		10		20	ns

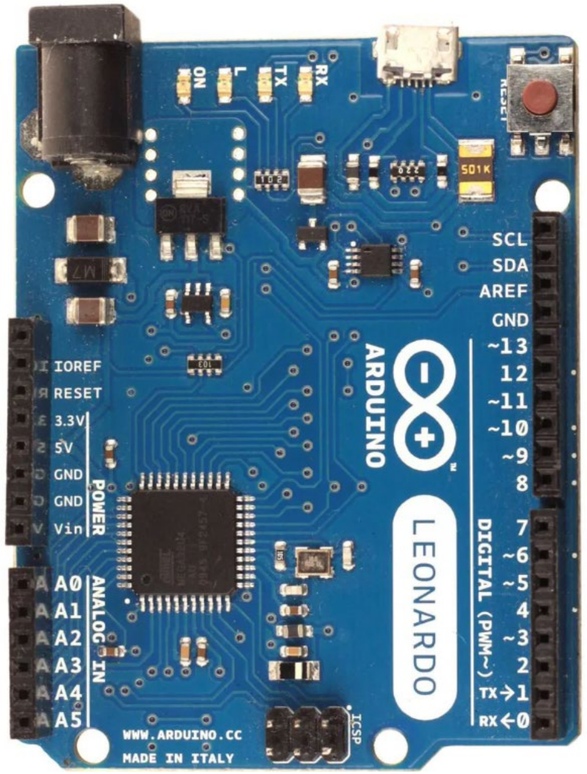


NOTE: SPI settings are CPOL = 0 and CPHA = 1.

Características sensor AD1299

Comunicación SPI: conexiones

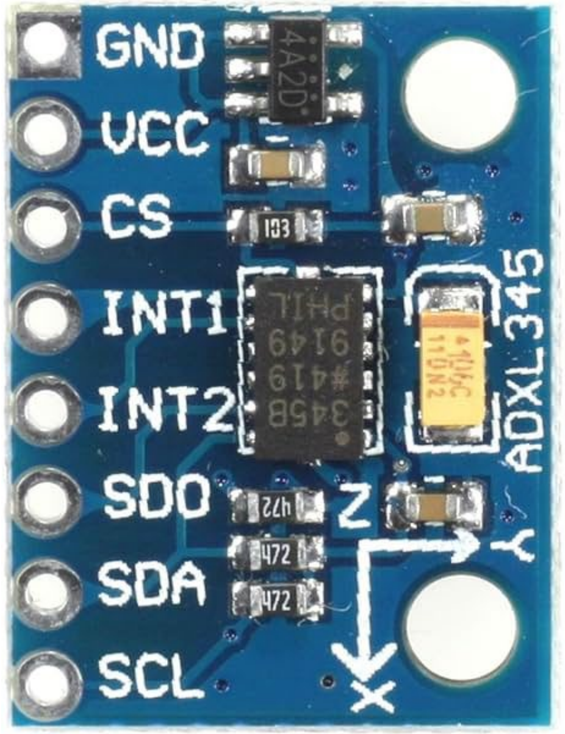




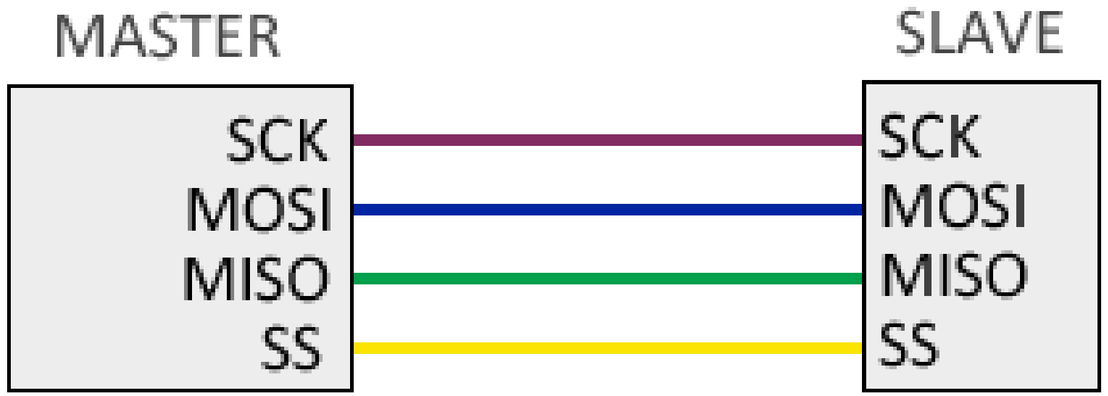
5V

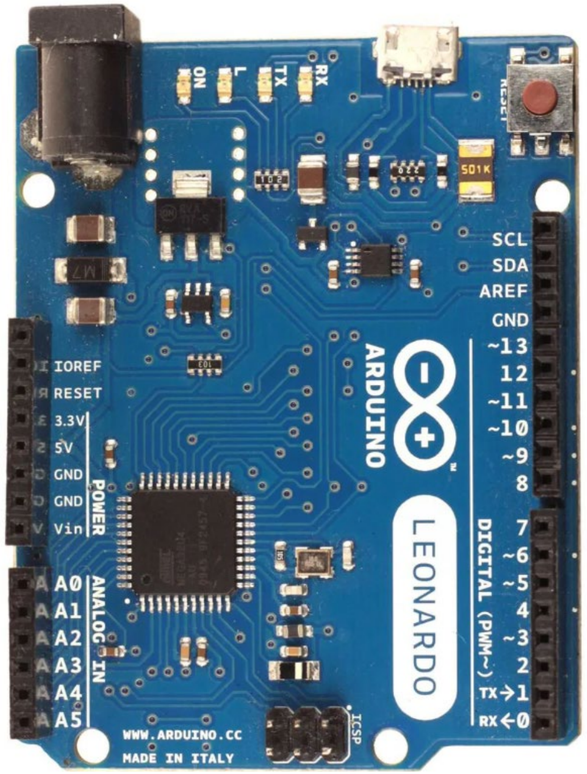
Interface, por voltaje

3.3V



Comunicación SPI: conexiones



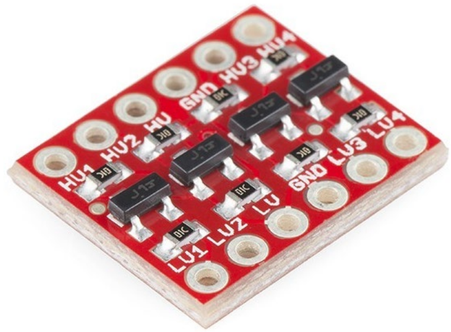


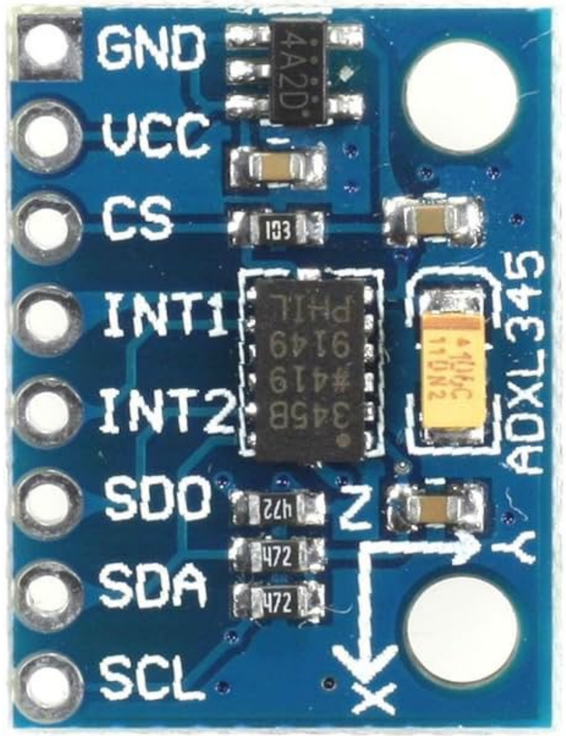
5V

Interface, por voltaje

3.3V

Conversor de niveles lógicos 3.3/5V





Comunicación SPI: Funciones librería SPI.h

```
#include <SPI.h>
void setup() {

  pinMode(slaveSelectPin, OUTPUT);
  SPI.begin();
}
void loop()
{
  cs_low;
  SPI.beginTransaction(SPISettings(Speed(HZ), DataOrder, SPI_MODE));

  SPI.transfer(data);

  data_IN=SPI.transfer(data);

  SPI.endTransaction();
  cs_high;
}
```

Orden_Datos

MSBFIRST: Bit mas significativo primero

MSBFIRST: Bit menos significativo primero

Mode	CPOL	CPHA
SPI_MODE0	0	0
SPI_MODE1	0	1
SPI_MODE2	1	0
SPI_MODE3	1	1

Referencias:

- [1] ATMEL. ATmega16/32U4 datasheet. https://www.datsi.fi.upm.es/docencia/Informatica_Industrial/DMC/pdf/atmega32u4.pdf
- [2] Elecia White. Making Embedded Systems, 2nd Edition. O'Reilly Media, Inc. https://learning.oreilly.com/library/view/making-embedded-systems/9781098151539/
- [3] DATSI. Ejemplos para programar la USART. Página web de la asignatura https://www.datsi.fi.upm.es/docencia/Informatica_Industrial/DMC/eje_usart_vmlab.rar
- [4] DATSI. Resumen y guía para programar el modulo LCD por sobre I2C. https://www.datsi.fi.upm.es/docencia/Informatica_Industrial/DMC/pdf/lcd_i2c.pdf
- [5] Arduino. API de la librería I^2C Wire. https://www.arduino.cc/reference/en/language/functions/communication/wire/
- [6] Arduino. API de la librería I^2C Wire. https://docs.arduino.cc/learn/communication/spi/