Azzolini Riccardo 2020-12-11

Interfacce seriali sincrone

1 Interfacce seriali sincrone

Le interfacce seriali sincrone fanno uso di un segnale di sincronizzazione, chiamato **clock**, che determina la cadenza con cui i dati transitano sul bus. Così, a differenza di quanto avviene per le interfacce seriali asincrone, non è necessario configurare separatamente e ugualmente tutti i dispositivi connessi: uno dei dispositivi, il **master**, determina come avviene la comunicazione, mentre gli altri dispositivi, gli **slave**, si adeguano (purché siano compatibili con i parametri di comunicazione scelti). Questa caratteristica rende le interfacce seriali sincrone adatte anche alla comunicazione con dispositivi molto semplici, come ad esempio i sensori integrati (a uscita digitale).

Nei microcontrollori, le interfacce seriali sincrone (così come quelle asincrone) sono tipicamente implementate a livello hardware, ad esempio da un dispositivo configurabile chiamato *USART* (*Universal Synchronous Asynchronous Receiver Transmitter*, uno UART che in più supporta anche interfacce sincrone).

2 SPI

SPI (*Serial Peripheral Interface*) è un'interfaccia seriale sincrona full-duplex, che collega un master a uno o più slave usando tre linee (fili) condivise e una linea separata per ogni slave.

Le tre linee condivise sono:

- **SCLK**, *Serial Clock*: il segnale di clock, generato dal master;
- **MOSI**, *Master Out Slave In*: la linea per l'invio dei dati dal master agli slave;
- **MISO**, *Master In Slave Out*: la linea per l'invio dei dati dagli slave al master.

Invece, la linea separata che connette un singolo slave al master è chiamata **SS**, *Slave Select*, e serve a selezionare lo slave con cui il master vuole comunicare: quando il master attiva la linea SS di un determinato slave, quest'ultimo comunica con il master sulle linee condivise, mentre gli altri slave, le cui SS sono inattive, rimangono "in silenzio" per evitare di interferire con la comunicazione. Le linee SS funzionano solitamente in modo **attivo basso**: uno slave è selezionato quando la sua linea SS è bassa (a 0 V), e deselezionato (inattivo) quando invece la sua SS è alta (a 3.3 V o 5 V). In generale, negli

schemi elettrici e nei datasheet, gli input/output attivi bassi vengono indicati mettendo una barra sul nome: $\overline{\text{SS}}$ o /SS.

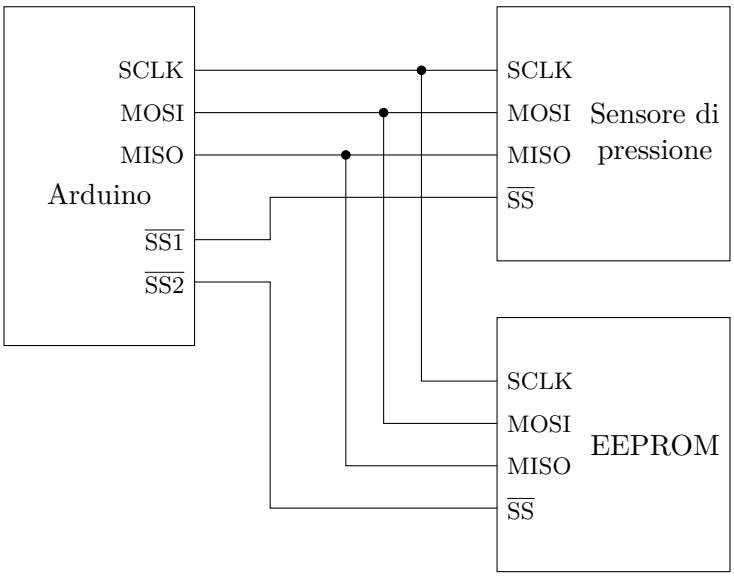
Il meccanismo di selezione degli slave a livello hardware, tramite le linee SS, rende semplice e veloce la comunicazione, e permette un'elevata flessibilità (ad esempio, se uno degli slave è più lento, si può diminuire la frequenza del clock quando esso è selezionato, e tenerla alta per gli altri slave), ma ha lo svantaggio di limitare il numero di slave che possono essere collegati al numero di piedini disponibili per l'uso come SS sul microcontrollore che fa da master: ad esempio, per comunicare con 10 slave servono 13 piedini del microcontrollore (SCLK, MOSI, MISO e 10 SS).

Nota: In alcuni datasheet vengono usati nomi/abbreviazioni diversi per le varie linee. Ad esempio:

- SCK invece di SCLK;
- DIN (Data In) e DOUT (Data Out) invece di MOSI e MISO (DIN corrisponde a MOSI se il dispositivo è uno slave, e a MISO se invece è un master, mentre per DOUT vale l'opposto);
- CS (Chip Select) invece di SS.

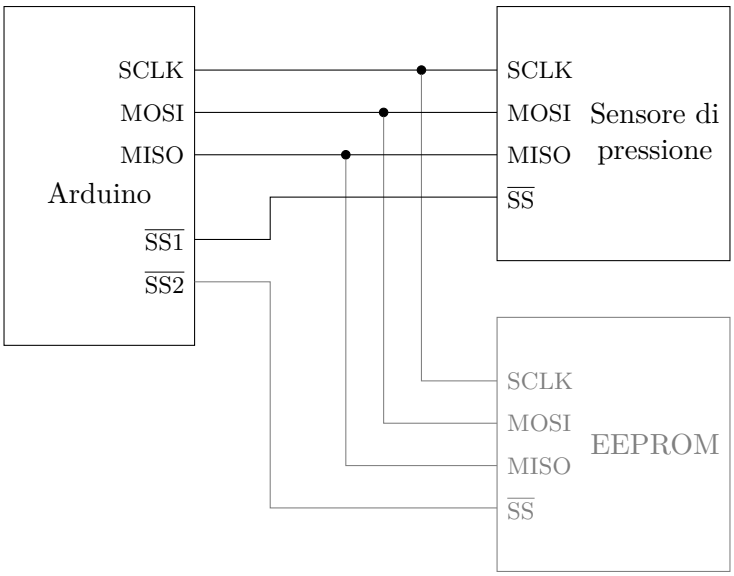
2.1 Esempio

Si supponga di voler mettere in comunicazione una scheda Arduino con due dispositivi dotati di interfacce SPI: un sensore di pressione, e una memoria non volatile (EEPROM) in cui salvare i valori di pressione letti. Le linee dell'interfaccia SPI vanno collegate come mostrato in questo schema:

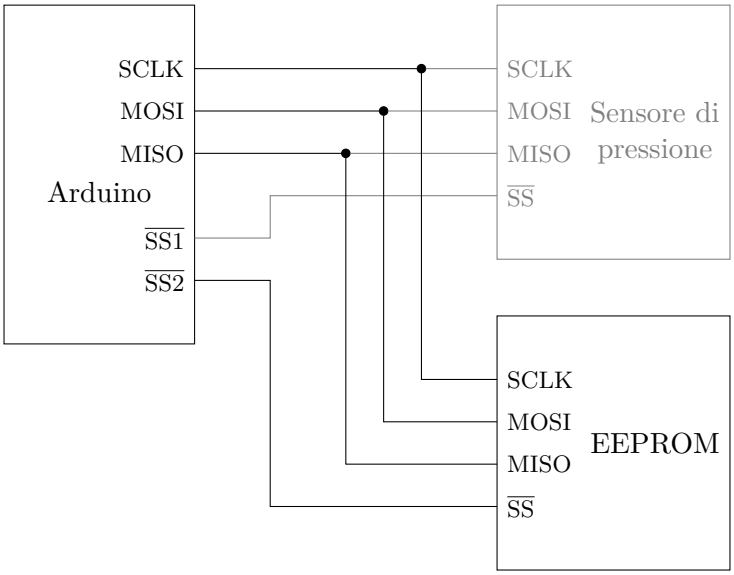


Tipicamente, su Arduino, le linee condivise (SCLK, MOSI e MISO) vanno collegate a dei piedini specifici, sui quali il microcontrollore implementa l'interfaccia SPI, mentre le linee SS vengono collegate a normali piedini di I/O digitale, gestiti con il solito comando `digitalWrite`.

Per leggere un valore di pressione, il programma caricato su Arduino deve portare bassa la linea SS1 e alta la linea SS2, in modo da abilitare la comunicazione con il sensore di pressione. A questo punto, usando nel codice un'apposita libreria SPI, si invia un comando di lettura sulla linea MOSI, e si riceve il valore di pressione sulla linea MISO (le modalità di comunicazione esatte dipendono dal sensore). Anche la EEPROM riceve il comando di lettura rivolto al sensore, ma lo ignora perché la sua linea SS è alta: in pratica, è come se alle linee condivise fosse collegato solo il sensore.



Quando poi si vuole salvare nella EEPROM il valore letto, il programma deve rimettere SS1 alta, portare invece bassa SS2, e inviare il dato da memorizzare sulla linea MOSI (secondo il formato di comunicazione indicato nel datasheet della specifica EEPROM utilizzata). In questo caso, è invece il sensore a ignorare i dati inviati, perché non è lo slave selezionato.



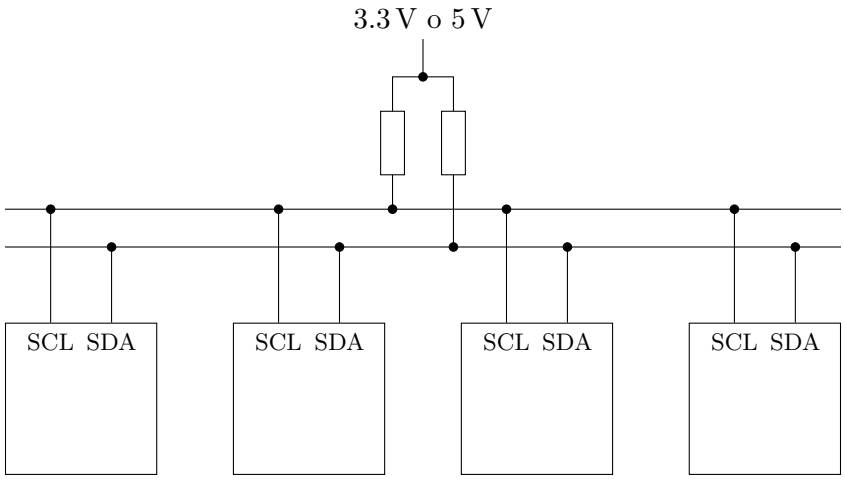
Infine, una volta conclusa la comunicazione, si rimette alta anche SS2.

3 I²C

Un'altra interfaccia seriale sincrona molto comune nell'ambito dei microcontrollori è **I²C** (*Inter-Integrated Circuit*; l'acronimo si pronuncia "I quadro C"), chiamata anche **TWI** (Two Wire Interface). Essa permette una comunicazione half-duplex tra dispositivi multipli (di solito un master e uno o più slave, ma è anche possibile avere master multipli), usando solo due linee condivise:

- **SCL**, *Serial CLock*, per il segnale di clock;
- **SDA**, *Serial DAta*, per i dati.

Queste due linee devono essere collegate alla tensione di alimentazione utilizzando delle resistenze di pull-up, il cui valore è spesso suggerito nei datasheet dei dispositivi, ma tipicamente è nell'ordine di qualche kΩ, e deve essere tanto più basso quanti più sono i dispositivi collegati alle linee.



Nel caso dei microcontrollori, queste resistenze potrebbero a volte essere già integrate nella periferica hardware che implementa l'interfaccia I^2C, oppure, nel caso di Arduino, potrebbero essere montate sulla scheda; nel dubbio, è consigliabile mettere comunque delle resistenze esterne.

Siccome non c'è un meccanismo hardware per la selezione dello slave, questa avviene tramite un meccanismo di indirizzamento nel protocollo di comunicazione, con indirizzi a 7 bit o a 10 bit. Ciascuno degli slave collegati a una stessa interfaccia (la stessa coppia di linee SCL e SDA) deve avere un indirizzo diverso dagli altri, quindi il numero massimo di slave collegati è limitato dagli indirizzi disponibili:

- con l'indirizzamento a 7 bit, che è il più usato, si hanno $2^7 = 128$ indirizzi (ma alcuni di questi sono riservati per codificare informazioni di controllo);
- con l'indirizzamento a 10 bit, si hanno a disposizione $2^{10} = 1024$ indirizzi.

L'indirizzo I^2C di un dispositivo è indicato nel suo datasheet. Tipicamente, i costruttori cercano di assegnare indirizzi diversi a dispositivi di tipi diversi, in modo da evitare conflitti. Inoltre, molti dispositivi hanno un indirizzo almeno in parte configurabile tramite appositi piedini di input, il che permette l'uso di più esemplari identici sulla stessa interfaccia (ad esempio, si potrebbero usare più sensori di temperatura per zone diverse di un ambiente, oppure più moduli di memoria per ottenere una maggiore capacità, ecc.). Se, nonostante tutto ciò, si dovesse comunicare con due dispositivi che hanno lo stesso indirizzo, l'unica possibilità sarebbe collegarli a due interfacce I^2C separate, ma per fortuna questo succede raramente in pratica.

Il grande vantaggio di I^2C rispetto a SPI è la possibilità di comunicare con numerosi dispositivi utilizzando solo due piedini del microcontrollore, mentre gli svantaggi sono il funzionamento half-duplex e la minore velocità di comunicazione (esistono diverse modalità, con velocità da 100 kbps a 3.4 Mbps), dovuta in parte all'overhead introdotto dall'indirizzamento (all'inizio di ogni comunicazione, bisogna trasmettere l'indirizzo dello slave da selezionare).