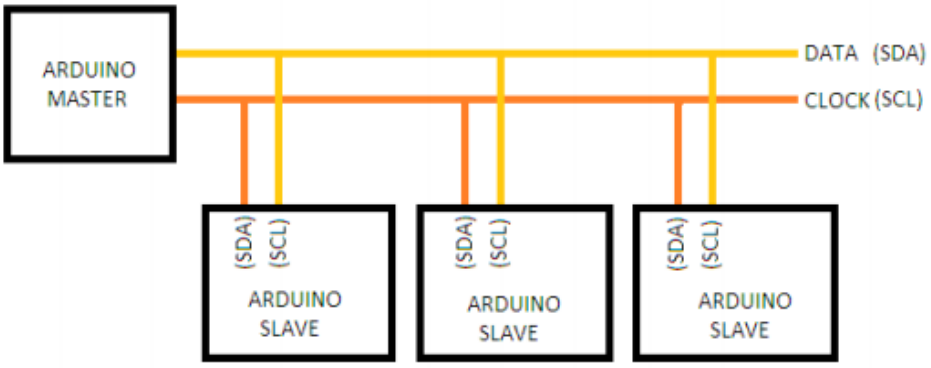
IL PROTOCOLLO DI COMUNICAZIONE I2C

Esistono alcuni metodi che permettono di comunicare con altri dispositivi utilizzando come canale di trasmissione dati i protocolli seriali (I2C, SPI o RS232). Questo rende possibile demandare ad una scheda slave funzioni di elaborazione gravose, che non vogliamo far eseguire alla scheda master.
Il bus I 2C, basandosi su due fili, non permette la comunicazione contemporanea tra Master e Slave. Lo scambio dati deve essere gestito dal Master tramite gli indirizzi (univoci) degli slave Il flusso può essere sintetizzato in questo modo:

- Il Master invia sul bus un bit di start
- Il Master invia sul bus l'indirizzo dello slave con cui vuole comunicare
- Il Master decide se scrivere o leggere dal dispositivo
- Lo Slave legge o scrive in base alla richiesta del Master



La libreria Wire dispone di tutte le funzioni necessarie alla realizzazione Master-Slave tra due schede Arduino.
Vantaggi

1. Il protocollo I 2C consente di mettere in comunicazione sulla stessa coppia di cavi ben 128 dispositivi. Questo è possibile grazie allo schema master-slave (capo-suddito) visto prima.
   Dal punto di vista hardware, il protocollo I2C richiede due linee seriali di comunicazione, che non sono TX e RX ma
   SDA (Serial DAta) ⇒É la linea su cui viaggiano le informazioni (bit).
   SCL (Serial CLock) ⇒É la linea su cui viaggia il segnale di clock, solitamente un'onda quadra periodica che sincronizza la comunicazione (pensate ad un metronomo);la presenza di questo segnale elimina i ritardi che altrimenti potrebbero compromettere la corretta trasmissione dei dati.
2. Un altro enorme vantaggio del protocollo I 2C è che tiene la seriale libera e questo è ottimo per gli Arduino UNO, che dispongono di un'unica seriale comune con la porta USB.
3. Il protocollo I 2C è più veloce rispetto alla classica comunicazione seriale.

Attenzione!
Se dovete utilizzare il protocollo I2C per la comunicazione con dei sensori, ovviamente dovete conoscere l'indirizzo associato a questi ultimi. A seguito trovate il link alla pagina di un programma che, una volta caricato su Arduino, scannerizza tutti i dispositivi collegati e restituisce i loro indirizzi (in codifica esadecimale).
- ARDUINO - I 2C Scanner
Aprendo il monitor seriale, viene segnalato ogni dispositivo trovato sul bus I 2C . È possibile modificare i cavi e i dispositivi I 2C plug-in mentre i2c_scanner è in esecuzione. L'output del monitor seriale sarà simile a questo:

LA LIBRERIA <WIRE.H>

Innanzitutto dobbiamo includerenla libreria , mediante la direttiva #include. L'istruzione necessaria per inizializzare una rete I 2C è Wire.begin(), ma va usata in modo differente a seconda del dispositivo in uso:

- Se l'inizializzazione parte dal MASTER, allora la funzione non richiede alcun parametro.
- Se invece parte da uno SLAVE, allora tra le parentesi tonde bisogna inserire l'indirizzo che si vuole assegnare al dispositivo.

Nel setup dell'Arduino MASTER quindi scriviamo Wire.begin(). Nel setup dell'Arduino SLAVE invece scriviamo Wire.begin(3): in questo modo gli assegniamo arbitrariamente l'indirizzo 3.
I due dispositivi sono pronti a comunicare nella stessa rete!
In una rete I2C la trasmissione dei dati avviene solo quando è il master a richiederla mediante l'istruzione Wire.requestFrom(indirizzo,numero_byte).
I parametri richiesti sono: L'indirizzo del dispositivo a cui va inoltrata la richiesta
Il numero di byte da richiedere: il master infatti deve tenere traccia della quantità di dati che viaggiano sul filo (Serial DAta).
Ogni slave, dal canto suo, ha il compito di rispondere immediatamente a qualsiasi richiesta del master; per questo nel setup del relativo sketch bisogna scrivere Wire.onRequest(funzione)
L'istruzione significa: "Appena ricevi una richiesta da parte dello slave esegui la funzione che è indicata tra parentesi tonde".
Le azioni che lo slave deve eseguire su richiesta del master saranno specificate in una nuova funzione, scritta separatamente dal loop e dal setup.

Creiamo quindi una funzione di nome “manda” che invia al master una sequenza di 4 caratteri (esattamente i 4 byte che esso richiede).

```
void manda()
 { Wire.write(“ciao”); }
```

La funzione Wire.write() funziona allo stesso modo della Serial.write().
Adesso dobbiamo completare il codice del master in modo che possa leggere ciò che lo slave sta trasmettendo e poi stamparlo a video.
Il ragionamento è questo:
“Finché lo slave invia dei caratteri, memorizzali singolarmente in una variabile di tipo char e stampane il contenuto: sarà poi il monitor seriale a comporli in modo che formino la parola.
Dopodiché vai a capo e aspetta un secondo”.

```
while(Wire.available()) { char a = Wire.read(); Serial.print(a); }
Serial.println();
delay(1000);
```

Anche in questo caso la funzione Wire.available() funziona allo stesso modo della Serial.available().
Per il MASTER il codice è il seguente:

```
#include <Wire.h>
void setup()
{ Wire.begin(); Serial.begin(9600); }

void loop()
{
Wire.requestFrom(3,4);
Serial.print(“Lo slave ha scritto: “);
 while(Wire.available())
    { char a = Wire.read(); Serial.print(a); }
Serial.println();
delay(1000);
}
```

ANALISI DELLE ISTRUZIONI

Wire.begin() ⇒ Inizializza una comunicazione I2C e identificami come master della rete.
Serial.begin(9600) ⇒Inizializza una comunicazione seriale a 9600 baud.
Wire.requestFrom(3,4) ⇒Richiedi 4 byte di dati all’indirizzo 3.
while(Wire.available()) ⇒Finchè lo slave invia dei dati.
char a = Wire.read() ⇒Leggi i caratteri e memorizzali uno alla volta nella variabile “a”.
Serial.print(a) ⇒Stampa il contenuto della variabile “a”
Serial.println() ⇒Vai a capo.
delay(1000) ⇒Aspetta un secondo: in questo modo il master invierà richieste di dati ad intervalli di un secondo.

Per lo SLAVE il codice è il seguente:

```
#include <Wire.h>
void setup()
 { Wire.begin(3); Wire.onRequest(manda); }

 void loop()
{ delay(100); }

void manda()
 { Wire.write("ciao"); }
```

ANALISI DELLE ISTRUZIONI

Wire.begin(3) ⇒Richiedi 4 byte di dati all'indirizzo 3.

Wire.onRequest(manda) ⇒Non appena arriva una richiesta dal master esegui la funzione "manda".

Wire.write("ciao") ⇒Invia al master la stringa "ciao".

Attenzione!

Nel loop abbiamo inserito solo un ritardo di 100ms semplicemente perchè adesso vogliamo concentrarci sulla comunicazione I2C, quindi abbiamo bisogno che aspetti senza eseguire alcuna istruzione aggiuntiva. Ovviamente all'atto pratico il loop viene eseguito ciclicamente come accade normalmente, mentre il controllo passa alla funzione "manda" solo in caso di richiesta da parte del master.

MASTER READER

L'esempio che abbiamo appena fatto riguarda solo il caso in cui il master (detto reader) richiede delle informazioni dagli slave, e questa è una pratica che solitamente si applica per la lettura dei sensori.

MASTER WRITER

Nel caso in cui volessimo che fosse il master (detto writer) ad inviare delle informazioni a determinati slave, l'istruzione da usare sarebbe Wire.beginTransmission(indirizzo)

Tale funzione non richiede la quantità di dati da inviare, ma solo l'indirizzo dello slave con cui il master vuole comunicare.

Ricordate che la funzione Wire.beginTransmission() deve essere sempre seguita da Wire.endTransmission(indirizzo) che segnala la fine della trasmissione, altrimenti tutti gli slave in comunicazione bloccherebbero permanentemente l'esecuzione del proprio loop.

Lo schema tipico, quindi è:

```
Wire.beginTransmission(indirizzo);
Wire.write(...);
Wire.endTransmission(indirizzo);
```

Per avere un quadro completo della libreria e approfondirne le funzioni con maggiore rigore, potete recarvi alla pagina della release ufficiale: - ARDUINO - Wire Library