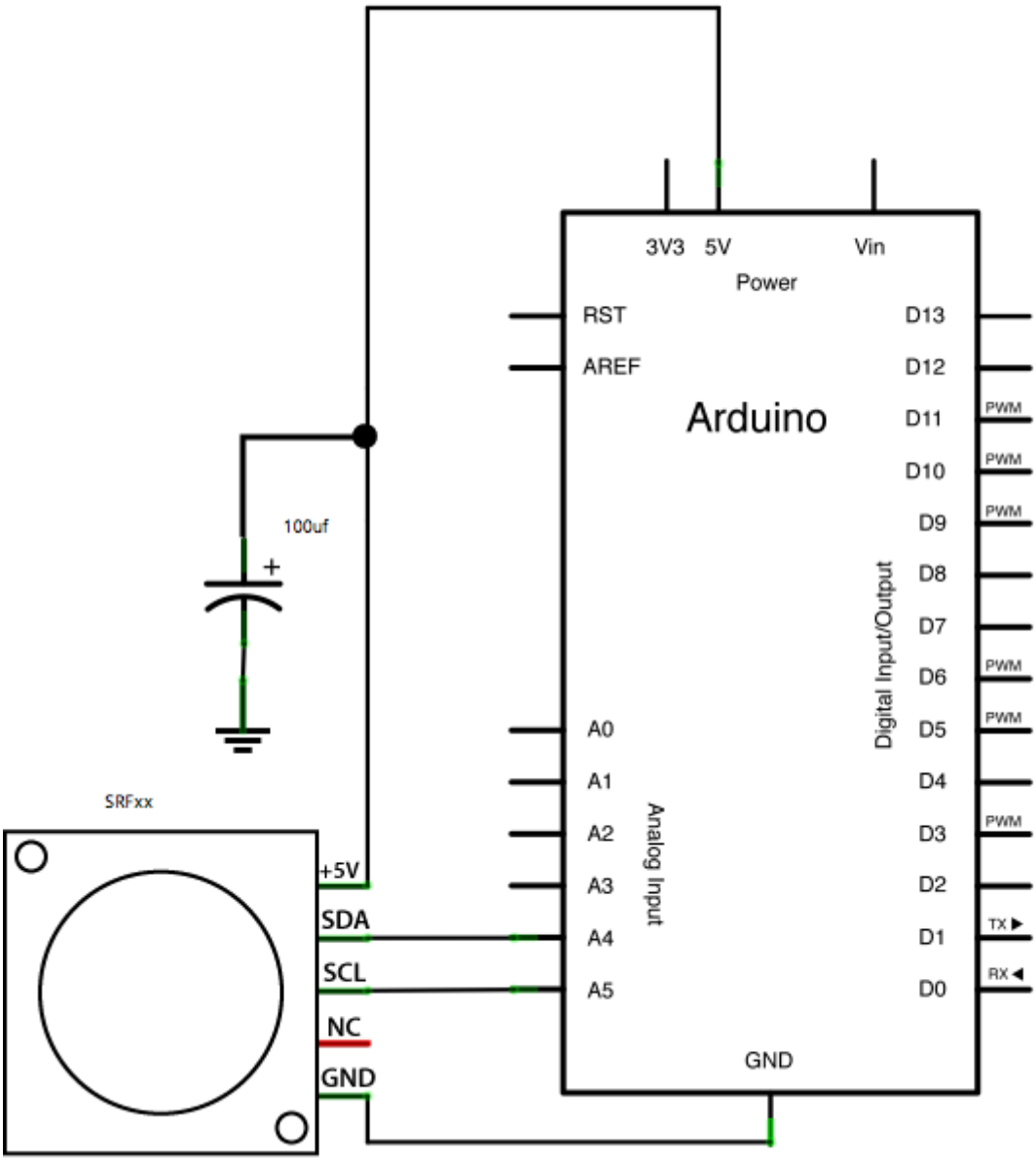


Libreria Wire Arduino i2c

luglio 22, 2014 da Critics

Premessa:

La libreria Wire nasce per la gestione di tutti quei dispositivi elettronici che negli anni sono stati sviluppati con la tecnologia **I2C** per comunicare su due soli fili con altri dispositivi digitali. Si tratta di una tecnologia ormai ben collaudata e nata dai laboratori Philips nel lontano 1980. Lo scopo era quello di ridurre la complessità del cablaggio per il collegamentofra dispositivi digitali e, nello specifico, fra dispositivi master come un microcontrollore, e dispositivi slave come sensori di pressione, memorie EE-PROM, sensori di temperatura, orologi in tempo reale e molto altro. Il collegamento di un bus indirizzi e uno dati richiedeva non poche linee di comunicazione, eventualmente condivise fra dati e indirizzi, ma comunque basate su almeno otto fili. Questi richiedevano altrettanti piedini di collegamento sui circuiti integrati e quindi il tutto comportava costi e complessità non da poco. Se poi il sensore o il dispositivo non era direttamente sul medesimo circuito stampatodel microcontrollore, allora il cablaggio rappresentava un ulteriore costo.

Studio:

Per risolvere tutto questo, Philips ha studiato e promosso un metodo di comunicazione **seriale** capace di gestire con il minor numero possibile di fili un numero di dispositivi significativo. In pratica, a differenza dell'interfaccia seriale che collega due dispositivi direttamente, il collegamento I2C è paragonabile a quello con un bus vero e proprio. I2C gestisce infatti sia l'indirizzamento, sia il trasferimento dati bidirezionale, pur utilizzando solo una linea per i dati (SDA) e una per il clock (SCL). Aggiungiamo una alimentazione e una massa e con quattro piedini è potenzialmente possibile avere tutto quello che serve. Il protocollo I2C permette e prevede che i dispositivi gestiscono l'intera comunicazione direttamente con la loro logica di bordo, mentre il microcontrollore ha a dispozione un meccanismo per il dialogo semplice e lineare. Attraversol'assegnazione di un indirizzo a ciascun dispositivo, sul bus I2C possono conviverenumerosi dispositivi.

Il protocollo I2C:

Per poter utilizzare in modo opportuno i dispositivi I2C è necessario comprendere il funzionamento del protocollo. Inanzitutto, si tratta di una soluzione che prevede sul bus di collegamento dei ruoli precisi, ovvero quelli di **MASTER** che controlla e di n.. **SLAVE** che eseguono i comandi impartiti dal Master; grazie a questo, la gestione della comunicazione sui due fili avvienein modo ordinato e le collisioni sono ridotte al minimo. Sul bus ci puo essere in ogni momento un solo Master, ma una volta terminata la comunicazione, il dispositivo Master può lasciare il suo ruolo e consentire che altri lo prendano. In questo modo, il sistema può essere gestito da più microcontrollori che interrogano e interagiscono con i medesimi dispositivi i2c assumendone la gestione di volta in volta.Il modo per selezionare un dispositivo slave è quello di inviare il suo indirizoo sul bus di comunicazione con le opportune modalità che vedremo più avanti. L'indirizzo può essere di sette o di dieci bit e in un sistema non ci possono essere slave che condividono il medesimo indirizzo. La maggior parte dei dispositivi ha un indirizzo base, assegnato in fase di produzione, più alcuni piedini che permettono di gestire due, tre o quattro bit bassi dell'indirizzo, così da poter avere sul medesimo bus più dispositivi uguali, ma con indirizzo diversificato se necessario. Alcuni dispositivi molto particolari, come gli orologi in tempo reale, hanno indirizzo predefinito, così che sul bus non ce ne possa essere più di uno. Tornando al protocollo, abbiamo detto che solo un master può essere attivo e dialogare con gli slave collegati al bus selezionandoli tramite l'indirizzo. E' previsto che si conosca già l'indirizzo di ciascun slave con cui dialogare e non ci sono meccanismi di scoperta, se non il provare a chiamare in sequenza i vari indirizzi e vedere chi risponde. Questo sistema, però, può creare problemi se ci sonodispositivi con indirizzi a 7 e 10 bit in quanto quelli a 7 bit potrebbero rispondere sulla parte bassa degli indirizzi, creando dei falsi positivi. Una sessione tipica di comunicazione su i2c si svolge attraverso una serie precisa di fai, parte delle quali è gestita dirattamente dalla libreria wire fornita nell'IDE.

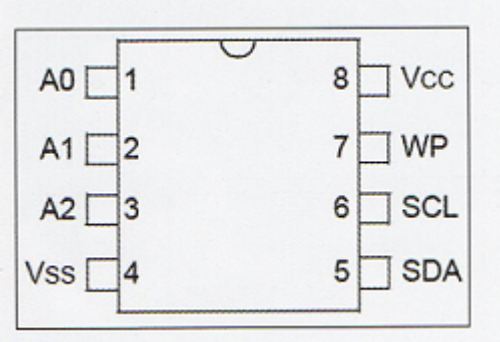
Il funzionamento:

Dato che le due linee di comunicazione sono bidirezionali, il Master inizia la propria attività verificando che non ci sia nessun altro dispositivo che sta usando il bus e nello specifico controlla che **SDA** e **SCL** siano entrambi ad **alto livello**. Delle resistenze di pullup fanno parte della normale configurazione hardware così che in assenza di dispositivi "attivi" entrambe le linee si presentino con un livello alto. I valori vanno da 2 a 10 kohm. Con le linee a livello alto, il Master inizia la propria sequenza di Start mandando **bassa** la linea **SDA**, senza che ci sia un segnale di clock. Gli altri dispositivi interpretano questa variazione e sanno che il bus è occupato. Cona la linea SDA ancora bassa, il Master inizia ad inviare il segnale di clock e solo dipo il primo fronte di discesa su SCL, la linea SDA torna alta e viene predisposto l'invio dell'indirizzo del dispositivo con cui il Master vuole dialogare. Dopo aver inviato l'indirizzo allo Slave, il Master invia anche il **bit** che segnala se l'oprezione da effettuare è di **lettura** o **scrittura**. La periferica a cui è destinata la comunicazione, aquesto punto, risponde al Master indicando che correttamente ricevuto la richiesta. A questo punto parte lo **scambio** di dati a otto bit fra Master e Slave secondo quanto i due dispositivi intendono fare, ovvero leggere e scrivere celle di memoria, recuperare informazioni sui sensori e quant'altro. Le regole che vanno co comunicazione sono che SDA assume un valore valido solo se la linea SCL è a livello

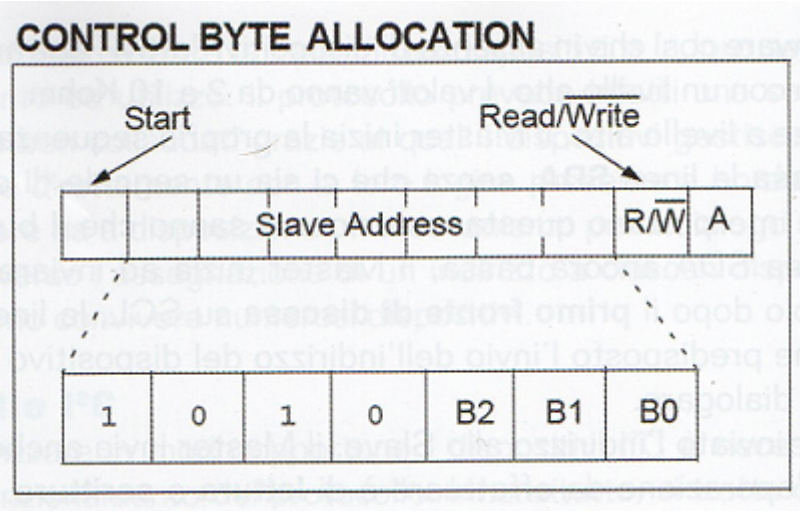
di pulluptiene alta la linea e che quindi il livello basso e necessariamente generato da un dispositivo collegato. Questo meccanismo permette anche di poter liberare il bus con una sequenza di Stop. I cicli di clock generati dal Master sono quindi quelli che scandiscono la comunicazione anche da parte dello Slave che, quando scrive i propri dati e le proprie risposte di acknowledgement, lo deve fare nella parte bassa dell'impulso di clock. l'acnowledgement è un bit che lo Slave invia sul bus al nono impulso di clock, ovvero al primo ciclo di clock dopo la ricezione dell'ottavo bit dal Master. Allo stesso modo lo slave si aspetta un bit di conferma al ciclo di clock successivo al suo ottavo bit trasmesso. Senza questo bit ricevuto da parte di chi ha ricevuto il bit di dati, si ritiene che la trasmissione non sia andata a buon fine generando una condizione di **errore nella comunicazione**. Mentre il Master dialoga con un dispositivo potrebbe avere bisogono di dialogare con un altro Slave e quindi ha due possibilità: chiudere la comunicazione con la sequenza di Stop e poi riprendere con uno Start e un nuovo indirizzo, con il rischio che qualche altro dispositivo assuma il ruolo di Master, oppure può inviare una **nuova sequenza** di Start che viene interpretata dallo Slave corrente come una chiusura della comunicazione. In questo modo il bus non viene mai liberato e il Master non perde la sua posizione di controllo dello stesso. A parte la sequenza di Start e Stop, quello che succede fra Master e Slave durante il loro dialogo dipende in modo specifico dai dispositivi. Il protocollo definisce infatti come la comunicazioneinizia, si instaura fra due dispositivi in modalità Master/Slave e lettura/scrittura e come termina, mentre non dice nulla sul significato dei dati scambiati.

Un esempio pratico:

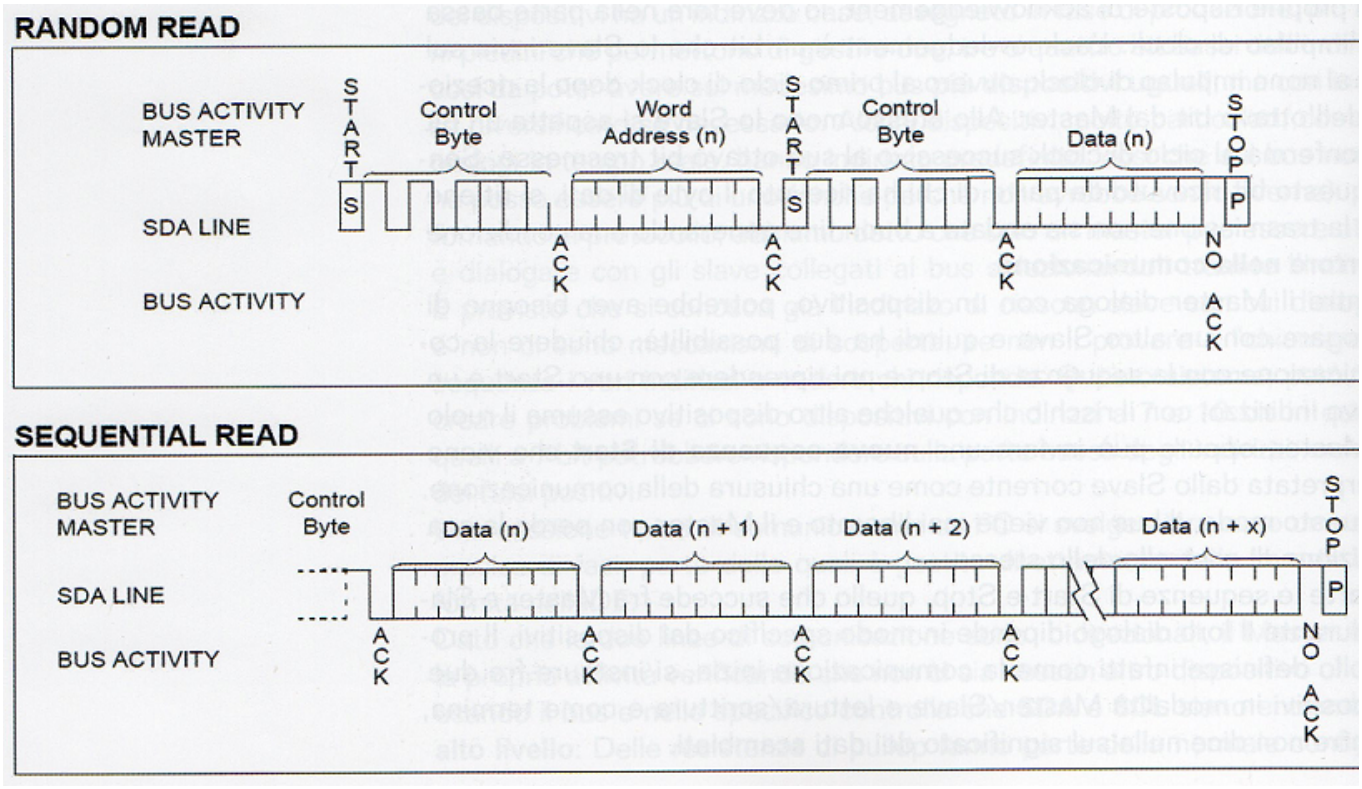
Ecco come è possibile lavorare con una memoria EEPROM di tipo i2c, modello **24LC16B** da 8 pagine e 256 byte Inanzitutto questo dispositivo è solitamente disponibile in un contenitore di tipo DIL a 8 pin e in una serie di varianti SMD. Nella versione Dual in Line gli otto pin hanno i collegamenti disposti in questo modo.

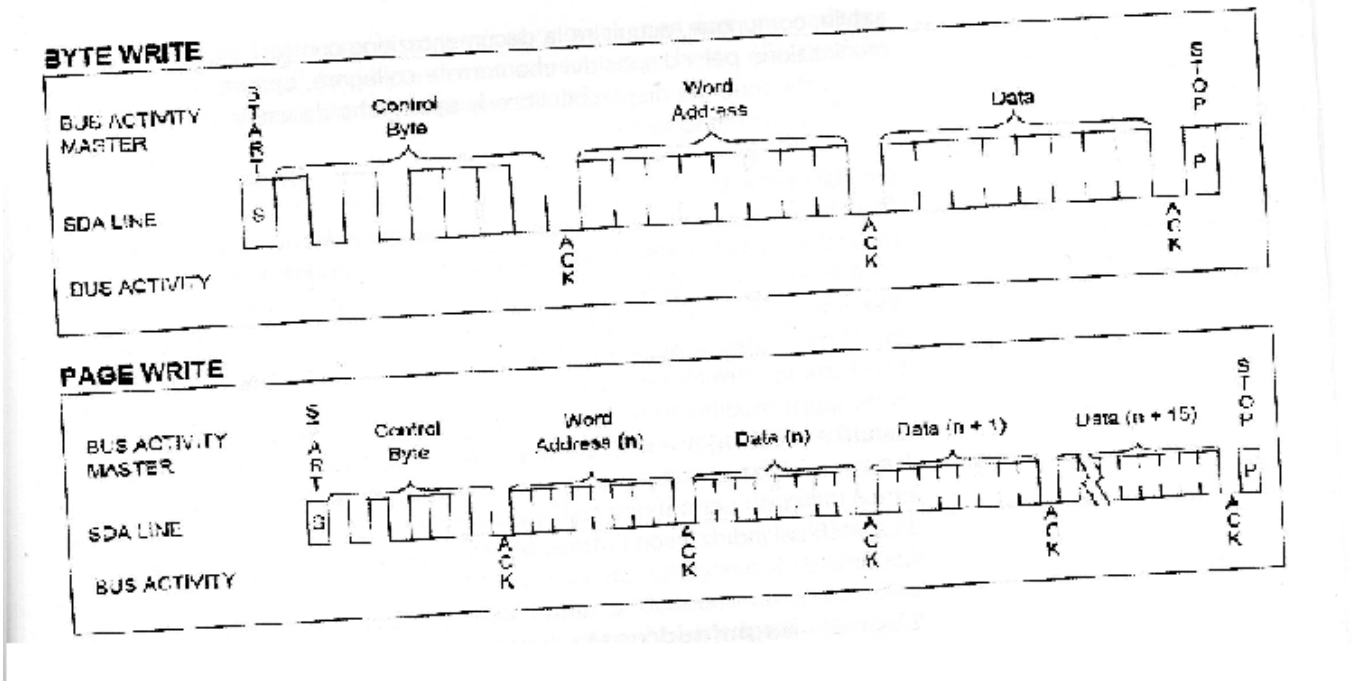


Vss e **Vcc** sono rispettivamente la massa e il positivo dell'alimentazione, poi SCL e SDA son i due pin con cui potremo comunicare con la memoria con il protocollo i2c, mentre WP è un pin che se collegato al positivo inibisce le operazioni di scrittura (write protect). A0,A1 e A2 sono non collegati, ma sono presenti per compatibilità con memorie EEPROM di dimensioni maggiori. Come abbiamo visto, se il master vuole parlare con questo dispositivo, inizia con il comando Start, quindi invia l'indirizzo dello Slave e successivamente comunica con lui. Il primo byte inviato sul bus è definito come **Control Byte** e contiene l'indirizzo del dispositivo assieme all'indirizzo della pagina da utilizzare, secondo lo schema di bit sotto riportati.



Praticamente questa memoria dispone di un indirizzo disso sui primi quattro dei sette bit e lascia tre bit per selezionare le otto pagine, seguiti da un bit che indica se le operazioni sono di lettura o scrittura e infine c'è il nono bit di acknowledgement che sarà generato dallo Slave a conferma della ricezione degli otto bit precedenti. Evidentemente, salvo soluzioni che prevedono l'impiego di circuiti logici aggiuntivi, sul bus i2c è possibile mettereun solo chip di memoria 24LC16 perchè non è possibile assegnarli un indirizzo diverso. Dopo il control byte che instaura la comunicazione, seguono i dati che possono essere letti o scritti dalla memoria secondo i quattro schemi sotto riportati.





Dopo il control byte contenente l'indirizzo di tre bit della pagina e il bit di lettura o scrittura, quello che può avvenire si limita quindi a quattro possibilità: due di lettura e due di scrittura. Da notare che avendo ogni pagina 256 byte, basta un byte per l'indirizzamento, mentre la destinazione fra una modalità di scrittura a byte o a [illegible] viene desunta dal fatto che il Master invia due byte o un flusso di 16 byte prima dello Stop. Ogni componente i2[illegible] propria documentazione circa la modalità di colloquio e programmazione, ma da quanto si può già intuire con l[illegible] seriali, non si tratta mai di situazioni particolarmente complesse.

La libreria wire i2c per Arduino:

Ora che i meccanismi di i2c sono chiari, diventa abbastanza semplice capire le funzioni offerta dalla libreria Wire. Queste vi permettono di far assumere ad arduino il ruolo di Master o Slave, di instaurare la comunicazione per il trasferimento di dati e di gestire il flusso della comunicazione stessa. Questa libreria eredita alcune funzioni dalla libreria **STREAM** e per omogeneità con tutte le altre librerie che gestiscono flussi dati in lettura e scrittura, sono state modificate le istruzioni di lettura e scrittura che cambiano da send e receive a write e read. Il numero di bit gestiti per gli indirizzi dalla libreria sono sette, con l'ottavo che è utilizzato per definire se l'operazione è in lettura o scrittura. Nel caso lo sketch usi indirizzi con l'ottavo bit valorizzato, questo sarà scaricato dalla libreria.

Le funzioni:

begin()/begin(address): Questa funzione permette di inizializzare la libreria e attribuire il ruolo di Master o Slave alla scheda Arduino. Se non si specifica un indirizzo, Arduino acquisirà il ruolo di Master sul bus, mentre se si indica un indirizzo, sarà quello a cui risponde come Slave. Con questo meccanismo, è quindi anche possibile far comunicare fra loro delle schede Arduino che possono assumere entrambe i ruoli.

requestFrom(address, count): Se Arduino è stato configurato come Master, con questa funzione può richiedere al dispositivo Slave con indirizzo address di inviargli un numero count di bytes, che poi andranno letti effettivamente con la funzione read().Per poter leggere i byte, questi dovranno essere anche disponibili nel buffer di comunicazione, da verificare con available().

beginTrasmission(address): Sempre come Master, con questa funzione Arduino inizia la procedura di trasmissione di dati al dispositivo caratterizzato sul bus dall'indirizzo address. Si tratta di un processo che prevede anche il write() dei byte e l'effettiva trasmissione degli stessi quando il pacchetto da trasmettere viene indicato come completo attraverso la funzione endTrasmission().

write(): Con questa funzione si scrive in un buffer un byte alla volta; questo permette di creare l'intero vettore di byte da trasmettere in modo sincrono al dispositivo destinatario senza che ci siano pause fra i byte. Il protocollo i2c prevede infatti che la comunicazione avvenga con una cadenza precisa, senza pause fra i byte di un medesimo pacchetto. L'inizio della scrittura è determinato da beginTrasmission().

endTrasmission(): Quando viene invocata questa funzione, la libreria provvede a inoltrare i byte scritti nel buffer di trasmissione al dispositivo Slaveil cui indirizzo address è stato definito con la fuinzione beginTrasmission(address). Questa funzione restituisce un byte che ha i seguenti significati:

- 0: trasmissione andata a buon fine
- 1: troppi dati per le dimensioni del buffer di comunicazione
- 2: ricevuto un NACK errore di trasmissione dell'indirizzo
- 3: ricevuto un NACK sulla trasmissione dei dati
- 4: altro errore

available(): La funzione va utilizzata per averecome risultato un byte che indica il numero di byte pronti per essere letti dal buffer di ricezione con la funzione read(). I dati possono essere stati ricevuti come Master a seguito di una requestFrom(address) oppure come Slave all'interno della gestione con onReceive(). Il byte permette di dimensionare il ciclo con il quale verrà letto il buffer di ricezione.

read(): Questa è la funzione con cui vengono letti i byte ricevuti come Master o come Slave. In entrambi icasi, i byte sono prelevati dal buffer di ricezione il cui contenuto può essere sempre controllato con available(). Inserendo la funzione read() in una struttura while(available()) continueremo a leggere dati finche il buffer di ricezione ne contiene, dato che solo quando il valore restituito da available() va a zero si ha l'uscita dal loop.

onReceive(handler): Quando Arduino è configurato come Slave, deve aspettarsi comunicazioni dal MAster in qualsiasi momento. Per questo, grazie a onReceive(handler) è possibile definire la funzione "handler" che viene chiamata quando vengono ricevuti dati dal Master. Alla funzione viene passato solo un byte che è il nume[illegible] ricevuti nella trasmissione dal Master. Una volta impostatala funzione nello sketch con una sintassi [illegible]id myHandler(int numBytes), questa verrà chiamata immediatamente alla ricezione dei dati, interrom[illegible]

onRequest(handler): Simile alla precedente, queta funzione definisce la funzione handler da chiamarequando lo Slave riceve una richiesta di dati da parte di un Master. Se Arduino è configurato come Slave, deve rispondere alle richieste del Master e lo fa attraverso il gestore (handler) definito con queta funzione. Come arriva la richiesta, l'esecuzione del codice passa al gestore che deve onorare la richiesta per non creare un errore sul bus i2c.

Nella sezione Home delle guide trovere un esempio di progetto realizzato con la libreria i2c di Arduino.

Buon progetto.

Progetti
Arduino, i2c
Corso Base – Raspberry Pi e Arduino a Roma – Luglio 2014
Corso Base – Raspberry Pi e Arduino a Roma – Agosto 2014

Servizi

Consulenza Progetti

Out of Stock

Carrello

Nessun prodotto nel carrello.

Segnalibri

Consulenza Progetti

Learn Critics

Makersheet