Arduino comunicazione tra schede I2C

Premessa

Il protocollo I2C (I^2C) sviluppato da Philips nel 1980 permette las comunicazione seriale tra dispositivi digitali utilizzando solo 2 fili (uno per i dati SDA, e uno per il clock SCL) + i altri due per l'alimentazione e per la massa) . Lo svantaggio rispetto ad altri protocolli quale l'SPI e' la velocita' e che i dati possono viaggiare solo in una direzione alla volta. Un dispositivo sul bus I2C si considera il master (che deve essere unico, ma che puo' cedere il ruolo per permettere ad un altro di diventare master) incaricato di trasferire le informazioni con altri dispositivi (slave), identificati da un indirizzo numerico univoco sul bus. Arduino Uno offre sul pin analogico 4 il segnale SDA e sul 5 l'SCL. Arduino Uno Rev. 3 ha dei pin aggiuntivi per l'I2C posti dopo il pin 13. L'indirizzo per lo slave (di 7 o 10 bit) puo' essere assegnato parzialmente in fase di produzione piu' una parte impostabile tramite pin, predefinito come gli orologici in tempo reale, o impostato via SW. Per sapere l'indirizzo slave di un dispositivo non ci sono meccanismi di 'scoperta'. O si ricava dai data sheet oppure si prova in sequenza fino a che qualcuno non risponde (a questo proposito si puo' usare lo sketch I2C_scanner (https://gist.github.com/tfeldmann/5411375). Dato che le due linee di comunicazione sono bidirezionali, il master se vuole attivarsi controlla che il bus sia libero testando che SDA e SCL siano entrambi ad alto livello. Le resistenze di pull-up (2KΩ-10KΩ) permettono che in assenza di dispositivi attivi entrambe le linee si presentino con livello alto. Il master inizia La sequenza start mettendo bassa la linea SDA occupando cosi' il bus. Viene quindi invia il clock sul pin SCL. E solo dopoil primo fronte di discesa su SCL la linea SDA torna alta e si puo' inviare sempre sull'SDA, l'indirizzo dello slave con cui il master vuole dialogare. Dopo aver inviato l'indirizzo allo slave, il master invia anche il bit che segnala il tipo di operazione lettura o scrittura. Lo slave a quell'indirizzo risponde al master indicando che ha ricevuto la richiesta. A questo punto parte lo scambio di dati a 8 bit tra master e slave.

Link per approfondire I2C:
http://www.crivellaro.net/wp-content/uploads/2017/03/Protocollo-di-comunicazione-I2C.pdf

La libreria Wire

Questa libreria permette ad Arduino di assumere il ruolo di master o slave e scambiare dati.
Per approfondire:
https://www.critics-corporation.com/RaspberryPi/libreria-wire-arduino-i2c

Comunicazione tra 2 Arduino

http://www.istitutobartolo.it/didattica/arduino/Arduino%2019-parte.pdf

Arduino può espandere le sue funzioni tramite l'uso di schede aggiuntive (shield). La maggior parte delle shield sono però progettate per compiere una funzione specifica, non programmabile. 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 secondo lo schema di figura.1 che illustra una configurazione che utilizza lo standard I2C.

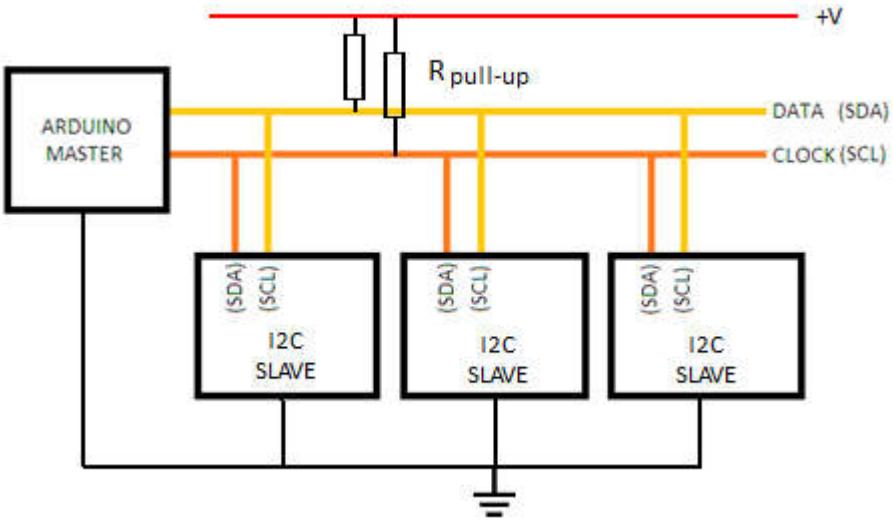

Figura.1

IL bus è costituito da due sole linee la linea SDA che consente al Master di inviare ricevere dati mentre e la linea SCL fornisce il segnale di clock che consente agli Slave di sincronizzarsi al Master.

Questa architettura di sistema si definisce ad *"intelligenza distribuita"* che sostituisce i tradizionali sistemi dove una sola logica programmabile svolge tutte le funzioni richieste dal sistema.

Il bus I2C, 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:

1. Il Master invia sul bus un bit di start
2. Il Master invia sul bus l'indirizzo dello slave con cui vuole comunicare
3. Il Master decide se scrivere o leggere dal dispositivo
4. 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.

Si propone, nella figura.2 , lo schema dell'architettura appena illustrata, il Master è collegato ad un dispositivo di uscita (Led) mentre lo Slave è collegato ad un dispositivo di input (pulsante)

Il master richiede un byte allo slave, attraverso la funzione *Wire.requestFrom(,)*, quando lo slave riceve la richiesta, utilizzando la funzione *requestEvent()*, legge lo stato del pulsante e se è premuto invia "a" altrimenti invia "s". Quindi il master legge il byte ricevuto e se è "a" accende il led, altrimenti se è "s" lo lascia spento. Sotto sono riportati i due SKETCH.

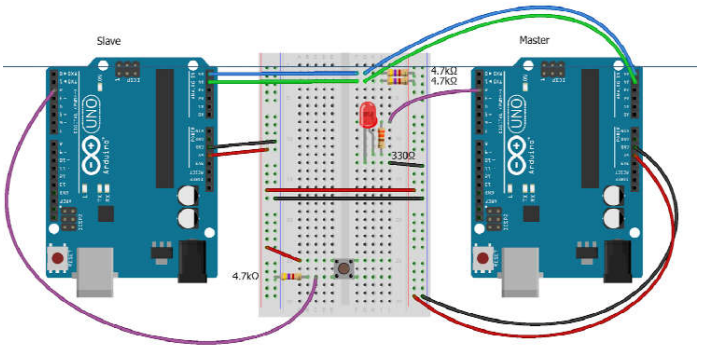

Figura.2

Sketch per il Master

```
#include <Wire.h>

void setup()
{
  Wire.begin();        // avvia il bus i2c
  pinMode (2,OUTPUT);
}

void loop()
{
  // richiedi 1 byte al dispositivo presente all'indirizzo 2.
  Wire.requestFrom(2, 1);
  String b = "";
  char c = Wire.read(); // leggi il byte ricevuto
  b = b + c;
  delay (100);

  if (b == "a")
  {    digitalWrite(2, HIGH);  }
  else
  {    digitalWrite(2, LOW);  }
  delay(10);
}
```

Sketch per lo Slave

```
#include <Wire.h>

void setup()
{  pinMode(2, INPUT);
    pinMode(13,OUTPUT);
    Wire.begin(2);      // entra nel bus I2C con indirizzo 2
    // esegue la funzione quando ha una richiesta
    Wire.onRequest(requestEvent);
}

void loop()
{
  delay(100);
}

void requestEvent()
{
  if (digitalRead(2) == HIGH){
    Wire.write("a");   // se il pulsante è premuto invia "a"
    digitalWrite(13,HIGH);
  }
  else
  {    Wire.write("s");      // altrimenti invia "s"
       digitalWrite(13,LOW);  }
}
```

Il seguente codice permette di inviare un dato numerico allo slave, che provvederà ad incrementarlo per poi rispedirlo al Master.

```
//MASTER
#include <Wire.h>

byte x = 0;
byte num = 0;

void setup()
{
//inizializzo la libreria Wire come Master
Wire.begin();
```

```
//init seiale
Serial.begin(9600);
//avviso che il programma è avviato
Serial.println("START");

}

void loop()
{
//invio sul bus I2C un byte al device
//che ha come indirizzo il valore 0x04
//start trasmissione
Wire.beginTransmission(0x04);
//invio un byte
Wire.write(x);
//fine trasmissione
Wire.endTransmission();

delayMicroseconds(500);

//richiedo un byte allo slave che ha indirizzo 0x04
Wire.requestFrom(0x04, 1);

//attendo la disponibilità di dati sul bus i2c
while(Wire.available())
{
//quando è presente un dato avvia
//la lettura
num = Wire.read();
}

//incrementa il valore del byte
x++;

//verifico che il byte letto dallo slave sia stato
//incrementato
if(num != x)
Serial.println("ERRORE");

delay(5);

}
```

Il codice seguente è relativo allo Slave

```
//SLAVE
#include <Wire.h>

byte x = 0;

void setup()
{
//inizializzo la libreria
//imposto l'indirizzo dello slave
Wire.begin(0x04);

//eventi per la ricezione del dato
//e per la richiesta del dato
Wire.onReceive(receiveEvent);
Wire.onRequest(requestEvent);
}
```

```
void loop()
{
//esegui qualcosa
delay(1000);
}

void receiveEvent(int data)
{
//questo evento viene generato quando sul bus
//è presente un dato da leggere

//eseguo la lettura
x = Wire.read();

//elaboro il dato letto
x++;
}

void requestEvent()
{
//questo evento viene generato quando il master
//richiede ad uno specifico slave
//una richiesta di dati

//spedisco il dato al Master
Wire.write(x);

}
```

A questo punto dopo la lettura del documento e il test degli sketch proposti potete sperimentare vostre applicazioni di intelligenza distribuita.

Novelli Claudio
Data di creazione: 18 mar18 mar
Ho verificato, nell'ambiente tinkercad potete condividere il vostro progetto con me, dovete inviarmi il link che si genera con il comando "condividi". Copiatelo e mettetelo nel documento che consegnate in classroom