CONCETTI DI BASE E STANDARD I²C

(REV. 2023)

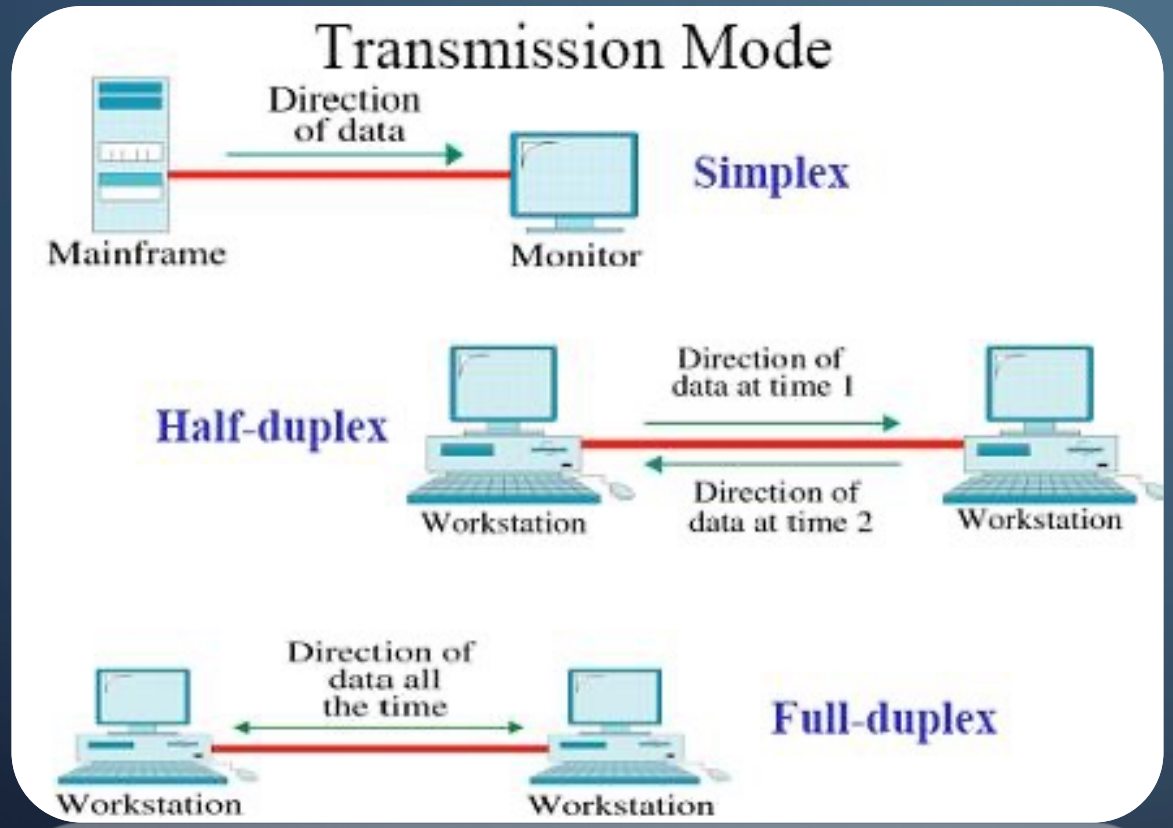
FONTI

- https://it.wikipedia.org/
- https://www.i2c-bus.org/
- https://www.italiantechproject.it/arduino/11-comunicazione-i2c
- http://elettronicadoc.altervista.org/i2c.html

INDICE

- Modi di trasmissione
- Trasmissione seriale e parallela
- Trasmissione sincrona e asincrona
- Protocollo I^2C
  - Introduzione
  - Physical layer
  - Data link layer
- Esempio di comunicazione tra due schede Arduino

FLUSSO DI INFORMAZIONI



La trasmissione delle informazioni tra due dispositivi può avvenire in tre modi:

Simplex: monodirezionale

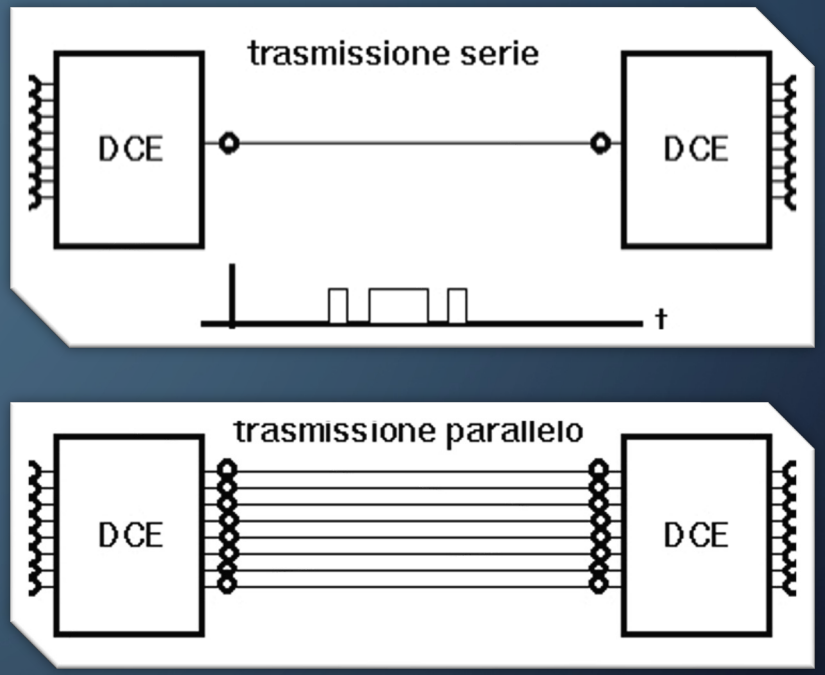
Half duplex: le informazioni viaggiano in entrambe le direzioni, ma non contemporaneamente

Full duplex: i dati possono viaggiare nei due sensi contemporaneamente

TRASMISSIONE SERIALE E PARALLELA

- Nella trasmissione seriale i bit di informazione vengono trasferiti tra i dispositivi (DCE: data communications equipment) uno alla volta, in sequenza

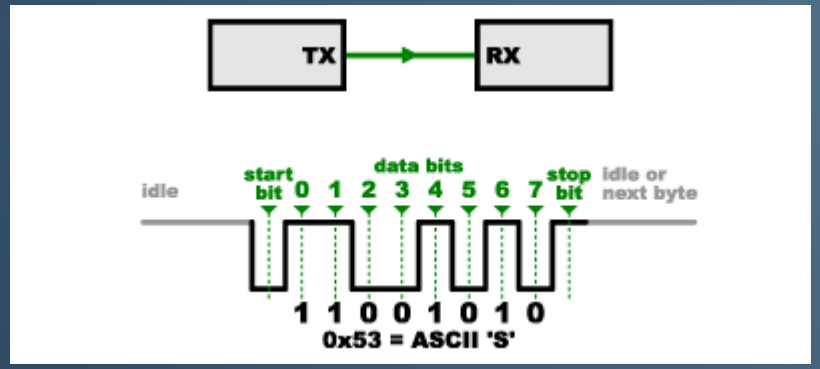
- Nella trasmissione parallela vengono invece trasferiti gruppi di più bit (parole) tutti allo stesso momento



La comunicazione a distanza tra dispositivi diversi avviene ormai solamente in modalità seriale perché:

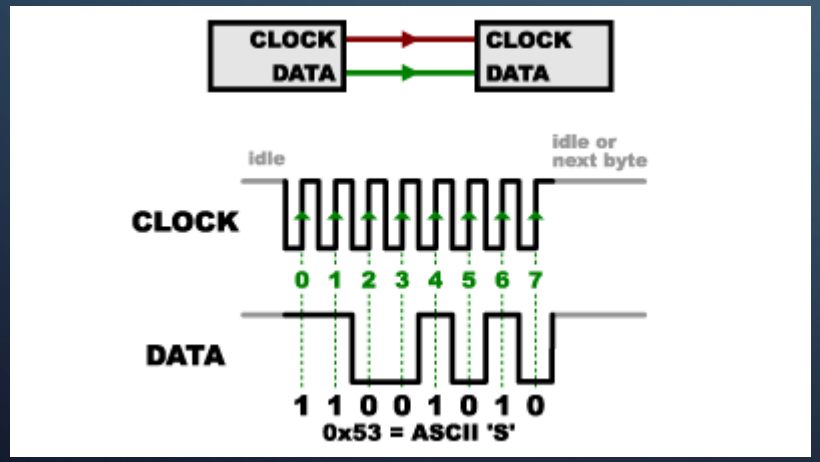
- richiede un minor numero di fili con conseguente **riduzione dei costi**
- è più tollerante rispetto alle **interferenze** e agli **errori di trasmissione**

TRASMISSIONE ASINCRONA E SINCRONA



Trasmissione asincrona:

- Il clock lato TX (trasmettitore) e RX (ricevitore) sono indipendenti e quindi vanno sincronizzati
- Basse velocità di trasmissione (al max 500 kbps)
- Interfaccia UART nei sistemi embedded



Trasmissione sincrona

- I dispositivi si scambiano il clock e quindi sono tra loro sincronizzati
- Elettronica più semplice (semplice shift register nella trasmissione SPI)
- Alte velocità di trasmissione (ordine dei Mbps)
- Interfaccia USART nei sistemi embedded

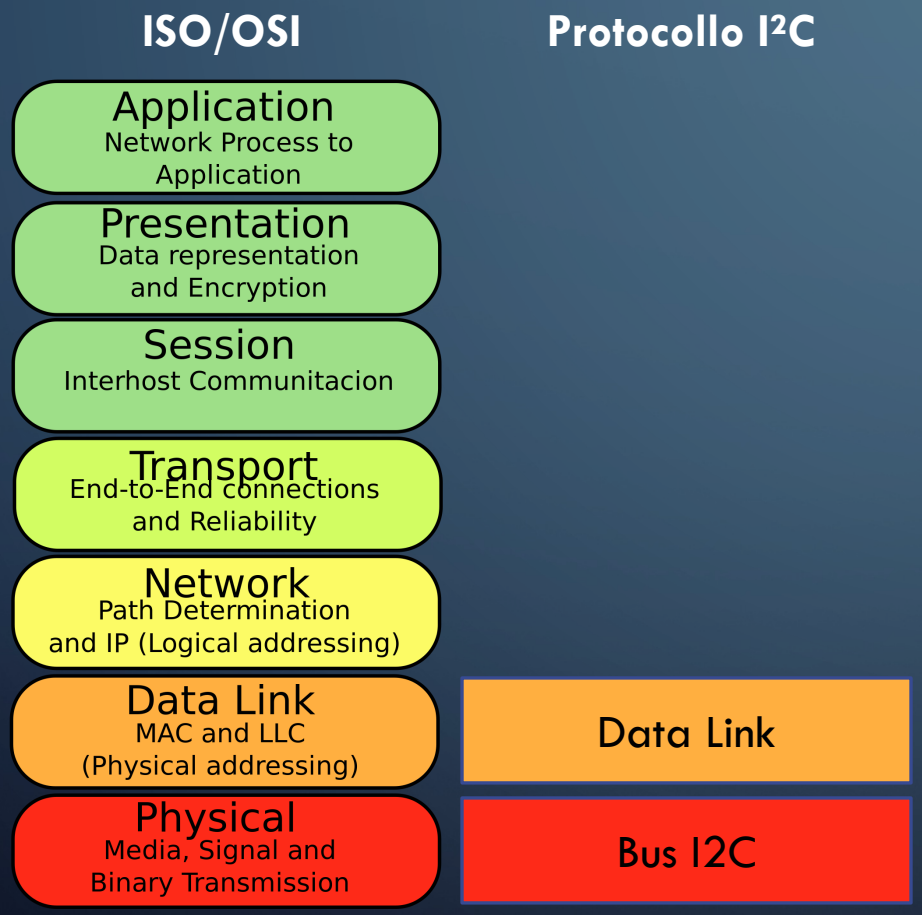
ORIGINI DELL' I²C

- Sviluppato dalla Philips (dal 2006 NXP) nel 1982
- **Attualmente il brevetto è ancora di proprietà NXP**

- Si chiama comunemente anche I2C (pronuncia «i due c»)

- È una trasmissione seriale sincrona con in più una descrizione di come i vari dispositivi connessi devono comunicare tra loro

PROTOCOLLO I2C

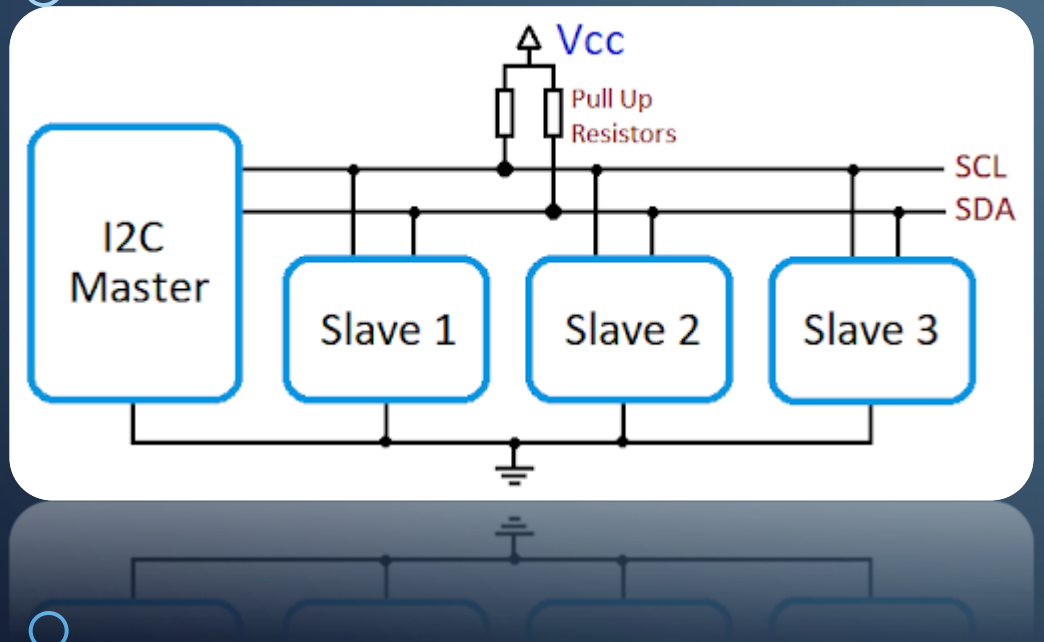


Il protocollo I²C implementa solo i primi due livelli del modello ISO/OSI, ovvero:

- Specifica come deve essere fatto fisicamente il bus I²C e come devono essere collegati i dispositivi
- Specifica come avviene la trasmissione tra i vari dispositivi connessi

Nasce per far comunicare prevalentemente dispositivi che stanno sullo stesso circuito stampato richiedendo un numero limitato di piste di connessione.
Ora si utilizza anche per connettere sensori ad un'unità di elaborazione

LIVELLO FISICO

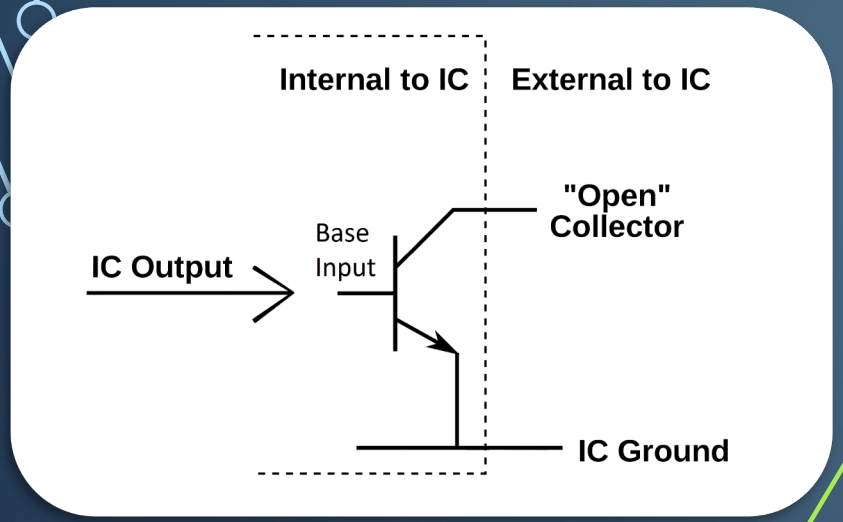


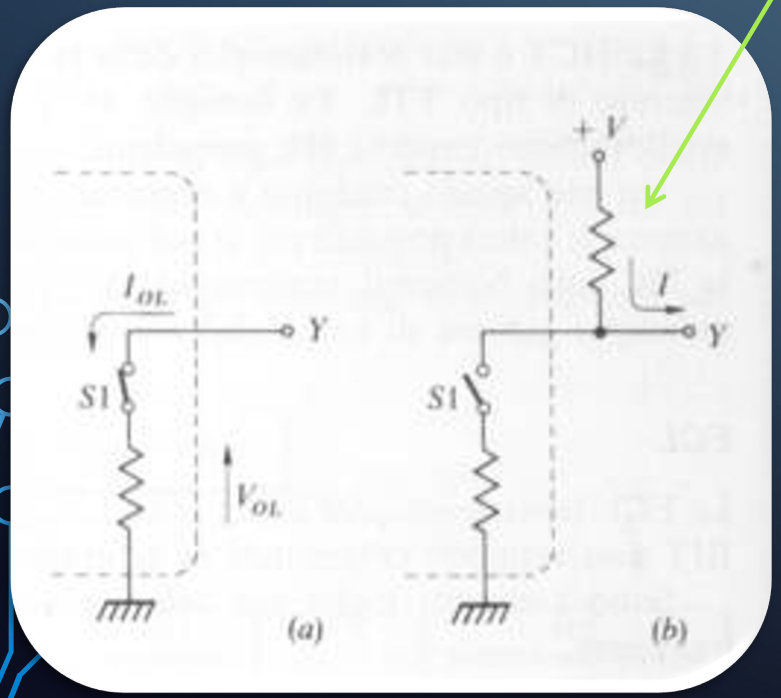
Il protocollo I²C implementa una connessione seriale sincrona, sono quindi necessari **3 fili:**

1. SDA (Serial Data) dove viaggiano i dati
2. SCL (Serial Clock) che trasporta il clock per la sincronizzazione
3. il filo di massa

- Le specifiche originali prevedevano un valore di Vcc = 5 V, ma poiché le specifiche sono vincolate ad un generico valore di Vcc, è comune che i dispositivi che operano a tensioni minori (per esempio 3.3 V) implementino il bus I²C con questi valori di Vcc

LIVELLO FISICO





Le specifiche prevedono che le linee SCL e SDA siano open-collector quindi è indispensabile la presenza di un resistore di pull-up per garantire la presenza del livello logico alto

- Quando un dispositivo open collector attiva la sua uscita (per scrivere uno «0» nel bus) forza bassa la linea portandola a livello logico zero
- Quando la disattiva (per scrivere un «1») lascia libera le linea che viene portata al livello logico alto dal resistore di pull-up
- Nessun dispositivo può forzare il livello logico alto: questa caratteristica viene utilizzata per identificare le collisioni nel caso sia realizzata una comunicazione multimaster

- Valori tipici per le resistenze di pull-up sono compresi tra 2KΩ e 10KΩ.
  - Il valore corretto dipende, oltre che dalla frequenza di trasmissione, anche dalla capacità totale della linea

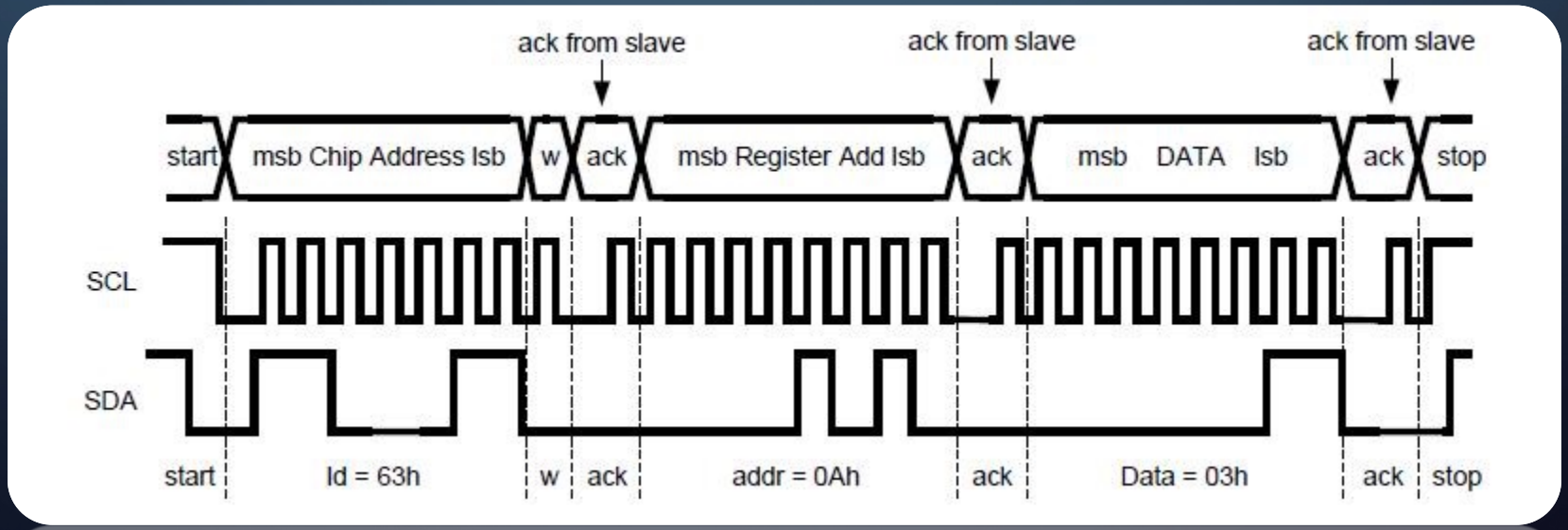
LIVELLO FISICO

Le velocità di trasmissione originali erano:

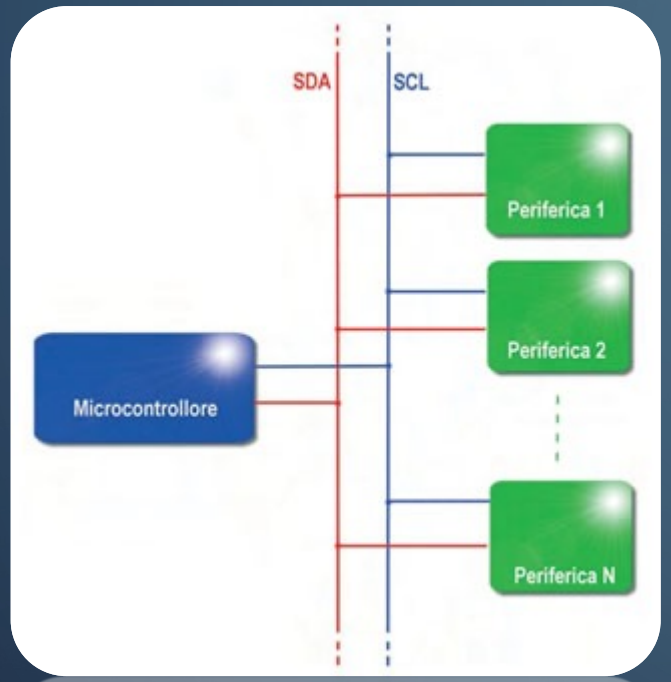
- standard mode: 100 kbit/s
- low-speed mode: 10 kbit/s

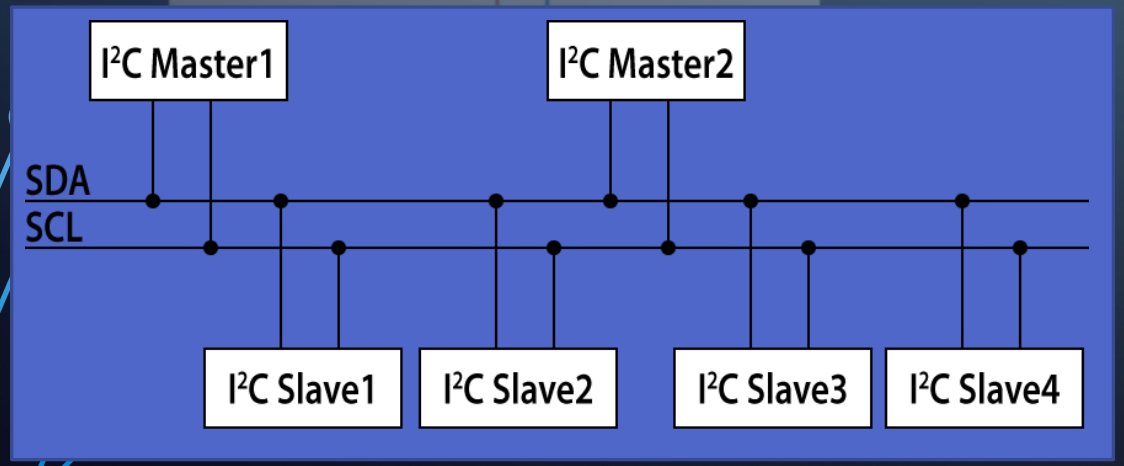
Le revisioni dell' I²C hanno introdotto ulteriori velocità più elevate:

- fast mode: 400 kbit/s
- high speed mode: 3.4 Mbit/s



LIVELLO DATA LINK (CONNESSIONE DATI)





- A livello di data link il bus I²C prevede:
  - una connessione master-slave con più slave
  - una connessione multimaster che prevede l'utilizzo di un arbitraggio delle collisioni (si ha una collisione quando due master vogliono trasmettere contemporaneamente)
    - questo modo di operare è poco diffuso e non viene trattato
- Ogni slave ha un indirizzo a 7 bit che lo identifica univocamente
  - si possono indirizzare fino a 112 slave (16 indirizzi sono riservati)
  - l'indirizzo deve essere assegnato allo slave in fase di costruzione della rete
    - non è possibile un'assegnazione dinamica
  - le nuove versioni dello standard hanno ampliato lo spazio degli indirizzi a 10 bit permettendo il collegamento di più periferiche (l'indirizzamento a 10 bit è comunque poco utilizzato)

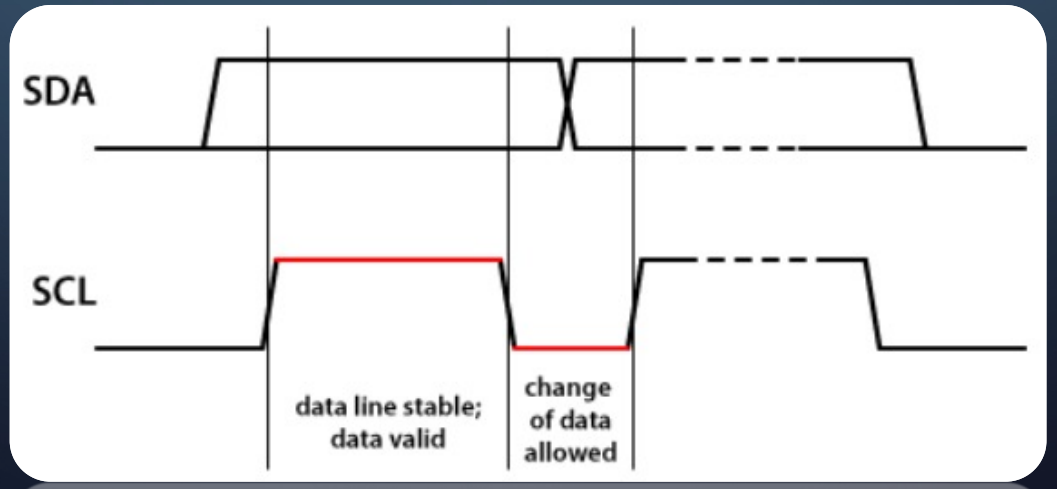
INDIRIZZI RISERVATI

Indirizzi	Utilizzo
0000000	Indirizzo di broadcasting
0000001	Indirizzo utilizzato per permettere in contemporanea una comunicazione sul CBUS (obsoleto)
0000010	Riservato per permettere l'uso dello stesso bus a dispositivi che utilizzano un protocollo diverso
0000011	Riservato per scopi futuri
00001XX0	Riservato per le periferiche ad alta velocità che utilizzano uno schema di indirizzamento diverso
11110XX	Parte iniziale di un indirizzo a 10 bit (usato per l'indirizzamento esteso a 10 bit)
11111XX	Riservato per scopi futuri

- I2C prevede un meccanismo di invio di messaggi a tutti gli slave della rete tramite l'indirizzo di broadcasting
- è prevista anche la possibilità di condividere lo stesso mezzo fisico con protocolli di tipo diverso a livello di data link (segnali elettrici uguali, ma significato dei bit e arbitraggio delle collisioni diversi)

LIVELLO CONNESSIONE DATI

- Il master genera un segnale di clock sulla linea SCL. Per ogni bit trasmesso deve essere generato un impulso di clock
- I dati sulla linea SDA devono essere stabili (non possono variare) durante il periodo ALTO di SCL
- Le modifiche sulla linea SDA (dati) si verificano solo quando la linea SCL (clock) è BASSA
- L'unica eccezione a questa regola sono le condizioni START e STOP
  - in questo caso è il master a far cambiare il valore dei dati sulla linea SDA con SCL alto



LIVELLO CONNESSIONE DATI

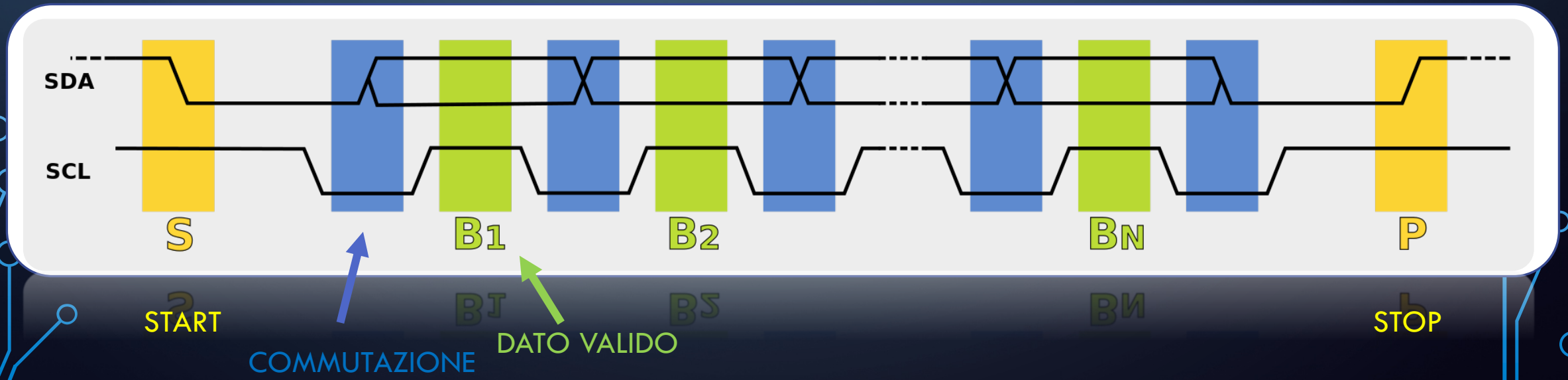
Il meccanismo di trasmissione dei bit è quindi:

- Il master ha sempre il controllo del segnale di clock SCL
- Il controllo di SDA può essere sia del master (se il master sta trasmettendo dati), sia di uno slave alla volta (per brevi periodi se il master deve ricevere dati)

Stato SCL (clock)	Stato SDA (dati)	Evento
LOW	Qualunque	Master o slave stanno modificando il livello logico sulla linea SDA. Dati non validi.
HIGH	HIGH	Dato valido (1 logico) nel bus dati. Può essere il master che sta trasmettendo o uno slave che risponde ad una richiesta del master.
HIGH	LOW	Dato valido (o logico) nel bus dati. Può essere il master che sta trasmettendo o uno slave che risponde ad una richiesta del master.
HIGH	HIGH -> LOW	Transizione da alto a basso del bus SDA con SCL alto. Inizio di una nuova trasmissione. Seguirà l'indirizzo dello slave e poi il comando.
HIGH	LOW -> HIGH	Transizione da basso a alto del bus SDA con SCL alto. Fine della trasmissione. Gli slave si rimettono (tutti) in attesa del prossimo comando.

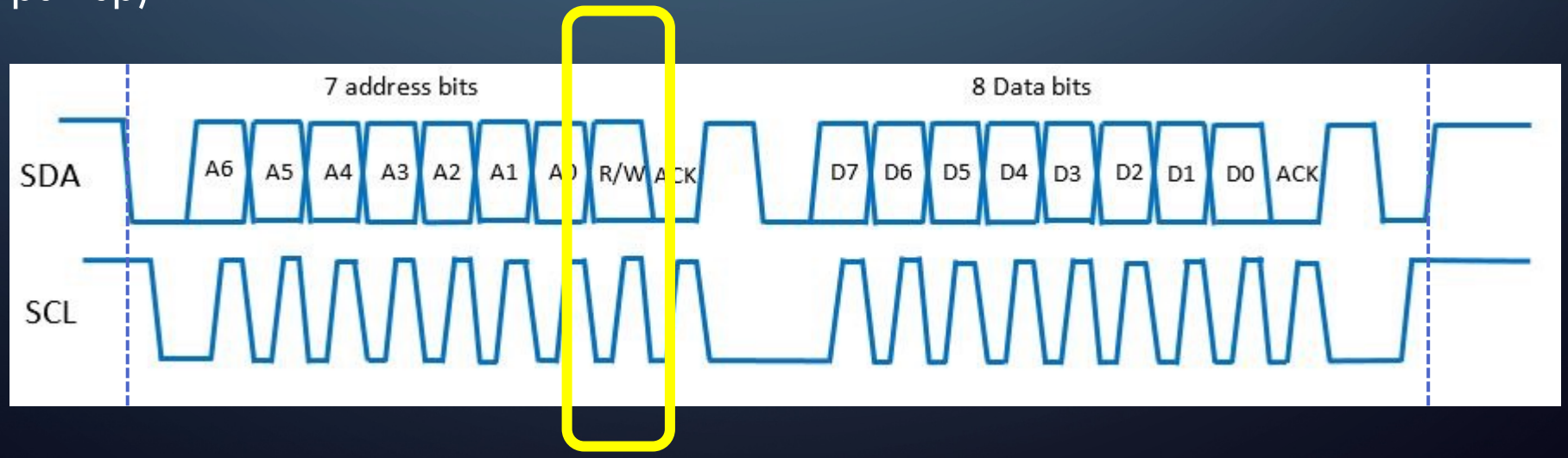
LIVELLO CONNESSIONE DATI

- Trasferimento di una sequenza di bit:
  - S è lo START bit (la linea SDA viene forzata bassa dal *master* mentre il clock SCL è a livello logico alto)
  - Quando SCL è basso avviene il settaggio del primo bit B_1 (in blu) la commutazione di SCL indica agli altri dispositivi che il dato è stabile e può essere letto (verde)
  - La stessa procedura prosegue fino all'ultimo bit B_N
  - La transazione termina con lo STOP bit (P) in giallo in cui SDA viene commutato da basso ad alto quando SCL è alto



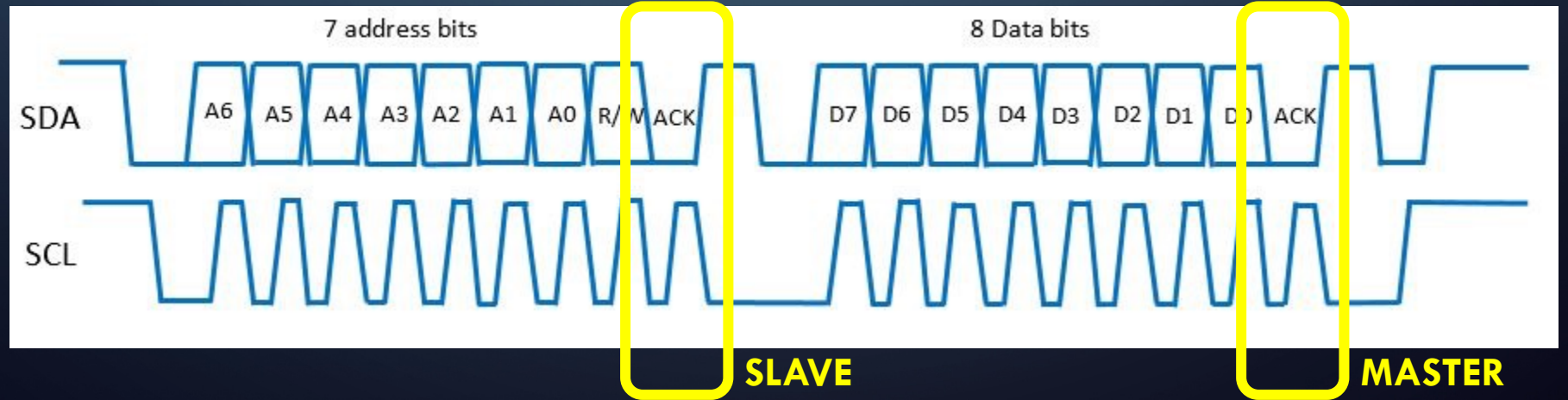
LIVELLO CONNESSIONE DATI: ESEMPIO DI RICHIESTA DI DATI DA PARTE DEL MASTER

- Il master inizia lo scambio di informazioni inviando lo start bit (S) seguito dall'indirizzo dello slave con cui vuole comunicare (A7, ..., A6)
  - Gli indirizzi e i dati per convenzione sono trasmessi iniziando dal bit più significativo e terminando con il bit meno significativo
- Segue un bit (R/W) che indica se il master vuole trasferire informazioni allo slave (scrivere) o ricevere informazioni (leggere)
  - Nel caso voglia scrivere il bit R/W è tenuto basso dal master, nel caso voglia ricevere informazioni il master rilascerà la linea dati (e il bus si porta alto per la presenza del resistore di pull-up)

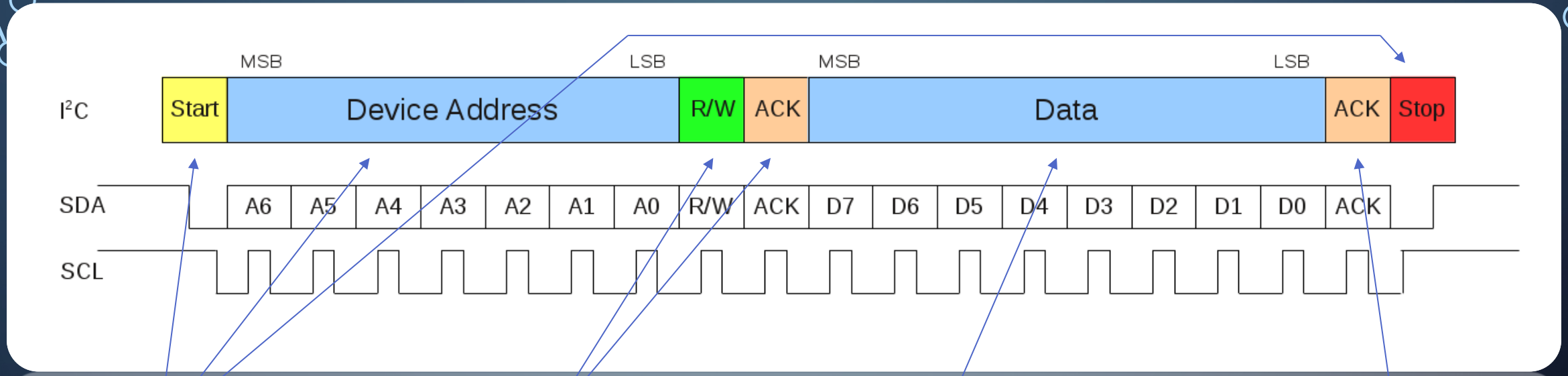


LIVELLO CONNESSIONE DATI: ESEMPIO DI RICHIESTA DI DATI DA PARTE DEL MASTER

- Se lo *slave* indirizzato (A6, ..., A0) esiste prende il controllo della linea dati (SDA) sul successivo impulso alto di SCL (che resta sempre comandato dal master) e la forza bassa (ACK)
- Il *master* accorgendosi che SDA è bassa sa che il dispositivo selezionato ha ricevuto la richiesta ed è in attesa di rispondere e modula il segnale di clock SCL per permettergli di rispondere
  - quando il *master* riceve informazioni da uno *slave*, invia un ACK alla fine di ogni byte ricevuto portando SCL basso se vuole ricevere altri 8 bit o lasciandolo alto se la trasmissione è conclusa
  - dopo l'ultimo byte il master dopo l'ACK invia un bit di STOP bit (P), ma può inviare anche START bit (S) se vuole mantenere il controllo del bus per un altro trasferimento



LETTURA O SCRITTURA DI DATI (SEQUENZA DEI BIT)



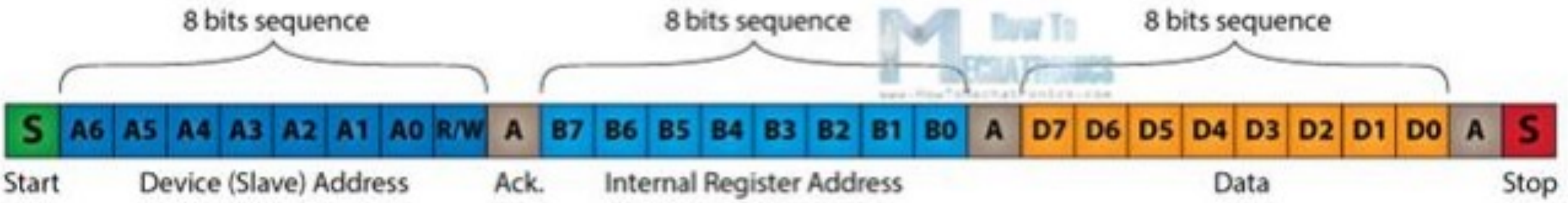
Sempre gestiti dal master

Se in scrittura:
master pone R/W = 0
slave pone ACK = 0
Se in lettura:
master pone R/W = 1
slave pone ACK = 0

Se in scrittura:
gestiti dal master
Se in lettura:
gestiti dallo slave

Se in scrittura:
slave pone ACK = 0 se ok
slave lascia ACK = 1 se ci sono problemi
Se in lettura:
master pone ACK = 0 se ok
master lascia ACK = 1 se è l'ultimo pacchetto che desidera

SEQUENZA TIPICA QUANDO SI INVIANO DATI A UNO SLAVE (PER ESEMPIO UN SENSORE)



La sequenza tipica di invio dati da MASTER a SLAVE è composta di 3 parti

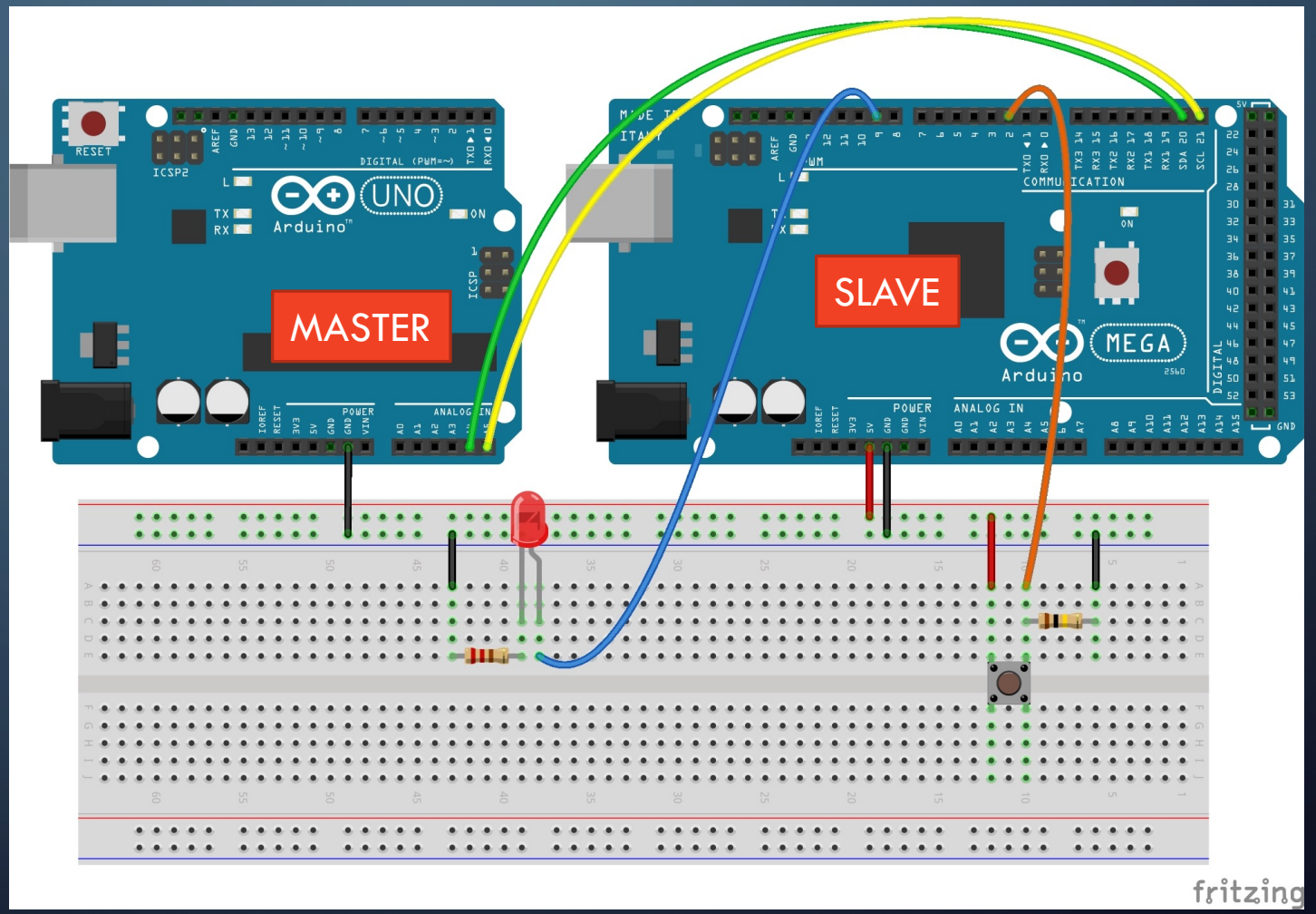
1. Identificazione dello SLAVE tramite trasmissione del suo indirizzo
2. Identificazione del registro interno allo SLAVE nel quele scrivere i dati
3. Invio vero e proprio dei dati. Se il registro ha una dimensione superiore a 1 byte vengono inviati ulteriori byte fino al raggiungimento della dimensione richiesta

ESEMPIO: CONNESSIONE I²C TRA ARDUINO

Schema elettrico
Arduino uno: master
Arduino mega: slave

Cosa fa il sistema?

1. Il led collegato allo SLAVE si accende e spegne ogni mezzo secondo su comando del MASTER
2. Il MASTER chiede allo SLAVE se il tasto è stato premuto e invia l'informazione nella porta seriale



ESEMPIO: CONNESSIONE I²C TRA ARDUINO

Esempio di programma per il master

- La libreria che permette la gestione della trasmissione I2C è la **Wire.h**

- Lo slave è all'indirizzo 7
- Wire.requestFrom(7, 2) chiede 2 byte allo slave all'indirizzo 7

- Il programma:
  1. chiede ricorsivamente 2 byte allo slave (informazioni sullo stato del pulsante)
  2. Accende il led per 500 millisecondi
  3. Spegne il led per 500 millisecondi

```
#include <Wire.h>

void setup(){
  Serial.begin(9600);
  Wire.begin();
}

void loop(){
  Wire.requestFrom(7, 2);
  while(Wire.available()){
    char c = Wire.read();
    Serial.print(c);
  }
  Serial.println();

  Wire.beginTransmission(7);
  Wire.write("accendi");
  Wire.endTransmission();
  delay(500);

  Wire.beginTransmission(7);
  Wire.write("spegni");
  Wire.endTransmission();
  delay(500);
}
```

ESEMPIO: CONNESSIONE I²C TRA ARDUINO

Esempio di programma per lo slave

```
#include <Wire.h>
#define slave_address 7

int pulsantePin = 2;
int ledPin = 9;

void setup(){
  Wire.begin(slave_address);
  Wire.onRequest(rispondi);
  Wire.onReceive(ricevi);
  pinMode(pulsantePin, INPUT);
  pinMode(ledPin, OUTPUT);
}

void loop(){
  delay(1000);
}
```

```
void rispondi(){
  if(digitalRead(pulsantePin) == HIGH){
    Wire.write("si");
  }else{
    Wire.write("no");
  }
}

void ricevi(){
  String comando = "";
  while(Wire.available()){
    comando += char(Wire.read());
  }

  if(comando == "accendi"){
    digitalWrite(ledPin, HIGH);
  }else if(comando == "spegni"){
    digitalWrite(ledPin, LOW);
  }
}
```

CONCLUSIONI

- I²C è l'abbreviazione di «Inter Integrated Circuit», (pronuncia «i-quadro-ci» o «i-due-ci»)
- È un sistema di comunicazione seriale (2 fili più massa) utilizzato tra circuiti integrati
- Il classico bus I²C è composto da almeno un master ed uno o più slave
- La situazione più frequente vede un singolo master e più slave
    - è sempre il master che gestisce la comunicazione
    - possono tuttavia essere usate architetture multimaster e multislave in sistemi più complessi