ARDUINO COMUNICAZIONI

(Prof. Fischetti Pietro)

Arduino ha diverse possibilita' facili e flessibii di interagire con PC e altri dispositivi

La Comunicazioni Seriali Asincrona RS232

E' noto come collegare il PC con la porta seriale di Arduino tramite cavo USB. Inoltre le comunicazioni seriali sono uno strumento rudimentale per fare il debug degli sketch. L'IDE di Arduino mette a disposizione il Serial Monitor per visualizzare i dati seriali inviati da Arduino. E' anche possibile inviare dati ad Arduino scrivendo nella casella di testo del Monitor Seriale e cliccando sul pulsante 'Send'. Arduino mette a disposizione la classe Serial (che deriva da Stream) per interagire con la seriale attraverso i seguenti metodi.

available()
read()
flush()
find()
findUntil()
peek()
readBytes()
readBytesUntil()
readString()
readStringUntil()
parseInt()
parseFloat()
setTimeout()

Ad esempio: Dopo aver impostato la velocita con Serial.Begin(),

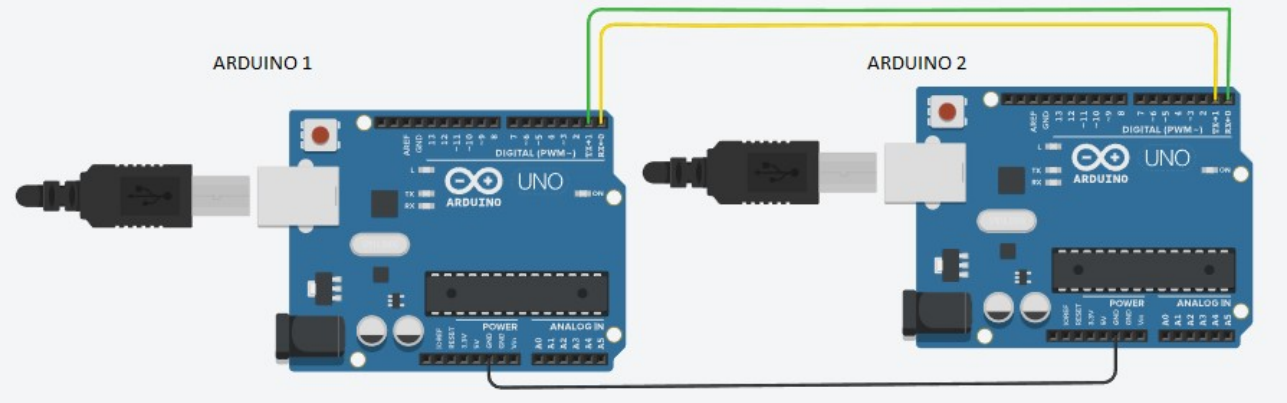
Serial.print e Serial.write permettono di scrivere nel monitor Seriale.

Serial.available() verifica se e' disponibile almeno un carattere in ricezione.

Serial.read() ritorna un numero intero che rappresenta il dato ricevuto.

Serial.available(): Ritorna il numero di byte (caratteri) disponibili per la lettura dalla porta seriale. Questi sono i dati che sono già arrivati e memorizzati nel buffer di ricezione seriale (che contiene 64 byte).

Vediamo un esempio di comunicazione tra due schede Arduino tramite Linea Seriale Presente sul pin 0 (RX) e pin 1(TX)



Nel codice seguente viene inviata una stringa da Arduino 1 a Arduino 2

Arduino 1 – Trasmette una Stringa

```
String str = "Hello";
void setup() {
  Serial.begin(9600);
}

void loop() {
  Serial.println(str);
  delay(1000);
}
```

Arduino 2 – Riceve una Stringa

```
String str;
void setup() {
  Serial.begin(9600);
}

void loop() {
   while(Serial.available()) {
        str = Serial.readStringUntil('\n');
        Serial.println(str);
  }
   delay(1000);
}
```

Codice per inviare un solo byte:

Arduino 1 – Trasmette un byte

```
byte b=0x41;
void setup() {
 Serial.begin(9600);
}

void loop() {
  Serial.write(b);
  delay(1000);
```

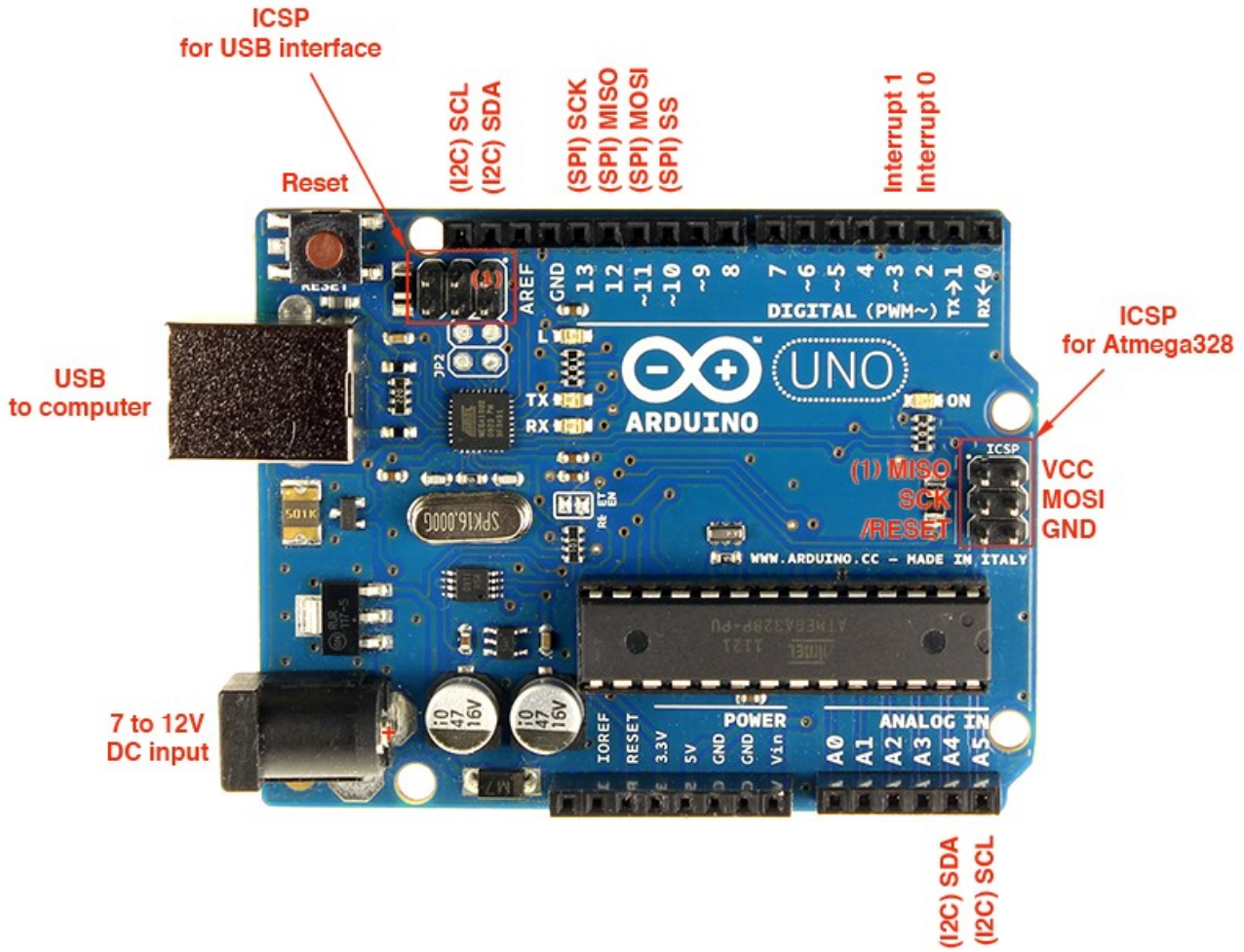
```
}
```

Arduino 2 – Riceve un byte

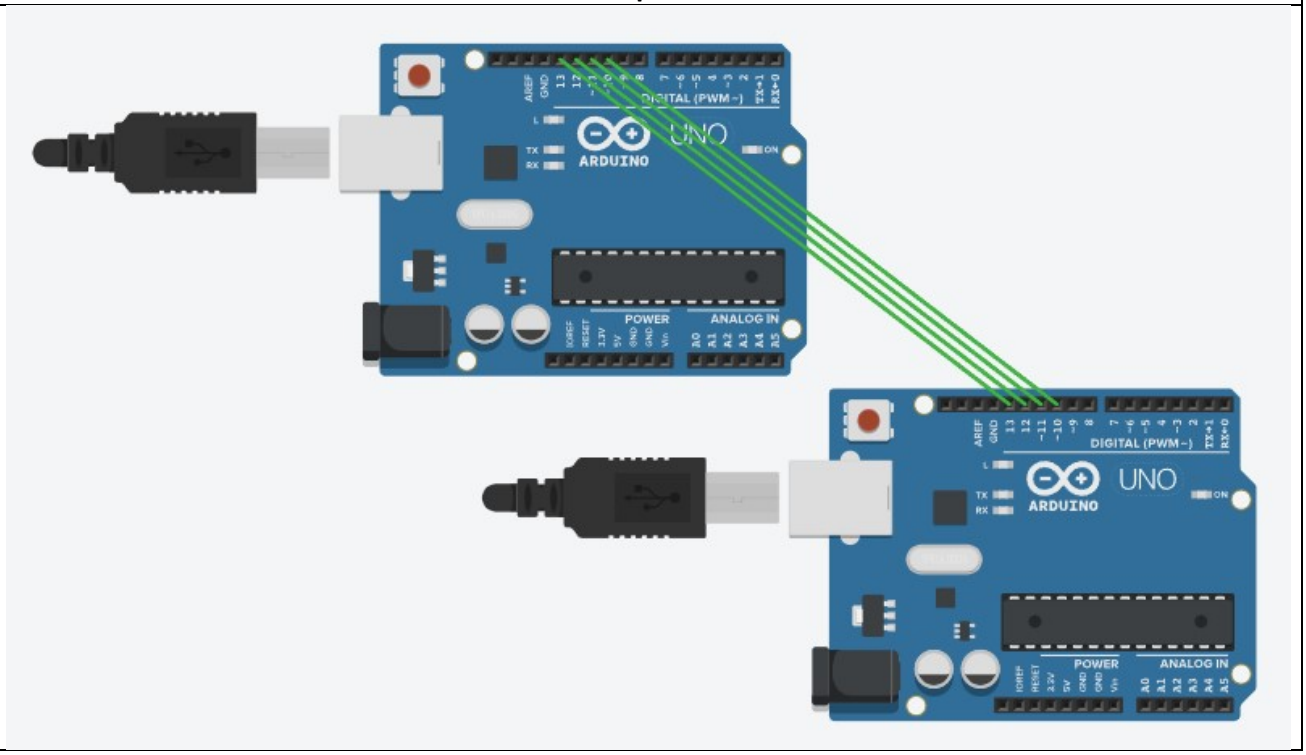
```
char c;
void setup() {
  Serial.begin(9600);
}

void loop() {
   while(Serial.available()) {
     c = Serial.read();
     Serial.println(c);
  }
   delay(1000);
}
```

Comunicare con I2C e SPI (Seriale Sincrona)

Gli standard I2C (Inter-Integrated Circuit) e SPI (Serial Peripheral Interface) facilitano il trasferimento tra sensori e microcontrollori. La scelta tra I2C e SPI dipende dal dispositivo utilizzato. I2C presenta un vantaggio: richiede che si colleghino solo due linee di segnalecon Arduino. Usare diversi dispositivi su questi due collegamenti e' abbastanza facilee inoltre e' possibile verificare che i segnali vengano ricevuti correttamente. Gli svantaggi sono che la velocita' cui viaggiano i dati e' inferiore a quella offerta da SPI e che i dati possono viaggiare solo in una direzione alla volta motivo per cui la loro velocita' si riduce ulteriormente quando e' necessaria una comunicazione a due vie. Per garantire che i segnali vengano trasmessi in maniera affidabile inoltre e' necessario utilizzare delle resistenze di pull-up. I vantaggi di SPI sono che permette ai dati di viaggiare a una velocita' maggiore e che i collegamenti input e output sono separati, per cui e' possibile inviare e ricevere contemporaneamente. Per selezionare il dispositivo attivo, SPI utilizza una linea addizionale per dispositivo, quindi se si devono collegare diversi dispositivi e' necessari un numero maggiore di connessioni. In pratica per poter sfruttare la velocita' di trasferimento dei dati nelle schede Ethernet e nelle schede di memoria si utilizzera' la SPI, mentre I2C si utilizzera' con i sensori che non devono inviare grandi quantita' di dati.


ICSP
for USB interface
Reset
(I2C) SCL
(I2C) SDA
(SPI) SCK
(SPI) MISO
(SPI) MOSI
(SPI) SS
Interrupt 1
Interrupt 0
USB
to computer
ICSP
for Atmega328
VCC
MOSI
GND
(1) MISO
SCK
/RESET
7 to 12V
DC input
(I2C) SDA
(I2C) SCL
DIGITAL (PWM~)
POWER
ANALOG IN
ARDUINO
UNO
WWW.ARDUINO.CC – MADE IN ITALY

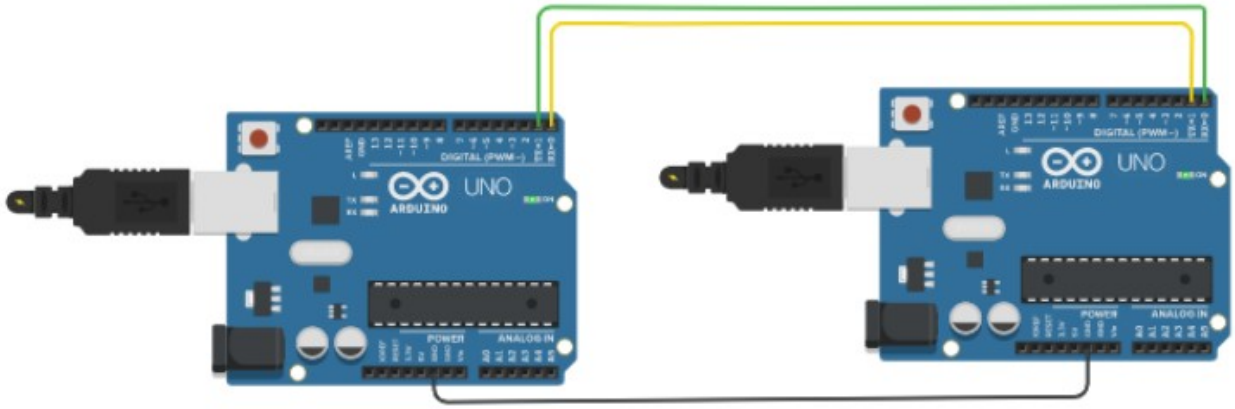
Esempio SPI


DIGITAL (PWM~)
ARDUINO
UNO
POWER
ANALOG IN

Codice .ino - WRITE

```
#include <SPI.h>

char buf [100];
volatile byte pos;
volatile boolean process_it;

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

  // have to send on master in, *slave out*
  pinMode(MISO, OUTPUT);

  // turn on SPI in slave mode
  SPCR |= _BV(SPE);

  // get ready for an interrupt
  pos = 0;   // buffer empty
  process_it = false;

  // now turn on interrupts
  SPI.attachInterrupt();
}


// SPI interrupt routine
ISR (SPI_STC_vect)
{
  byte c = SPDR;  // grab byte from SPI Data Register
  if (pos < sizeof buf)
  {
    buf [pos++] = c;
    if (c == '\n')
      process_it = true;
  }
}

void loop (void)
{
  if (process_it)
  {
    buf [pos] = 0;
    Serial.println (buf);
    pos = 0;
    process_it = false;
  }
}
```

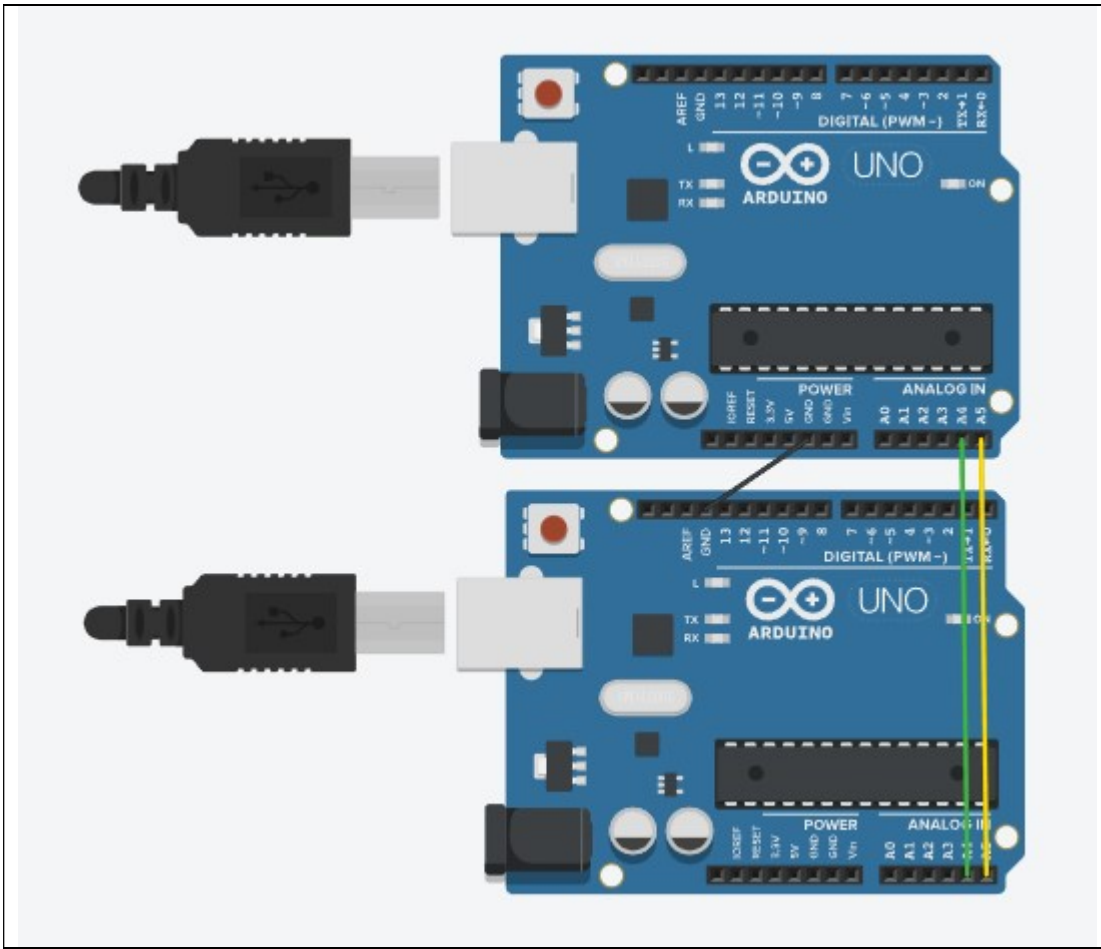
Esempio RS232



Codice .ino - READ

```
char mystr[10]; //Initialized variable to store recieved data
String str;
char b[2];
char c;
void setup() {
  // Begin the Serial at 9600 Baud
  Serial.begin(9600);
}

void loop() {
  //Serial.readBytes(mystr,5); //Read the serial data and store in var
  //Serial.println(mystr); //Print data on Serial Monitor

   while(Serial.available()) {
    //str = Serial.readStringUntil('\n');
    //Serial.println(str);

     //Serial.readBytes(b,1);
     //Serial.println(b);
     c=Serial.read();
     Serial.println(c);
  }
   delay(1000);
}
```

Esempio I2C



Codice .ino – SLAVE

```
// Wire Slave Receiver

// Demonstrates use of the Wire library
// Receives data as an I2C/TWI slave device

#include <Wire.h>

void setup() {
  Wire.begin(4);              // join i2c bus with address #4
  Wire.onReceive(receiveEvent); // register event
  Serial.begin(9600);         // start serial for output
}

void loop() {
  delay(100);
}

// function that executes whenever data is received from master
// this function is registered as an event, see setup()
void receiveEvent(int howMany) {
  while (1 < Wire.available()) { // loop through all but the last
    char c = Wire.read(); // receive byte as a character
    Serial.print©;        // print the character
  }
  int x = Wire.read();    // receive byte as an integer
```

```
  Serial.println(x);        // print the integer
}
```

Codice .ino -MASTER

```
// Wire Master Writer

#include <Wire.h>

void setup() {
  Wire.begin(); // join i2c bus (address optional for master)
}

byte x = 0;

void loop() {
  Wire.beginTransmission(4); // transmit to device #4
  Wire.write("x is ");       // sends five bytes
  Wire.write(x);             // sends one byte
  Wire.endTransmission();    // stop transmitting

  x++;
  delay(1000);
}
```

/////Esempio I2C: Master(Sensore Ultrasuoni)+Slave(Servo)