Carica di un Condensatore

Obiettivo : Analizzare la carica di un condensatore mediante una lettura analogica.

Componenti elettronici:

- Arduino UNO
- Breadboard
- 1 Condensatore (1000 microFarad)
- 1 Resistenza (100 Ohm) per carica condensatore
- 1 Pulsante
- 1 Resistenza (1 kOhm) per resistenza pulsante

Teoria: Il condensatore è un dispositivo elettronico che ha la capacità di immagazzinare carica elettrostatica. Tale dispositivo modifica il suo comportamento in funzione del segnale di alimentazione utilizzato nell'applicazione in questione (e.g., segnale continuo, segnale alternato, etc …).

In regime di tensione costante (corrente continua), il condensatore si carica inizialmente (**regime transitorio**) fino a raggiunge una situazione di equilibrio dove la carica sulle armature corrisponde esattamente alla tensione applicata mediante il generatore (**regime stazionario**). Una volta carico, il condensatore si comporta come un circuito aperto ovvero interrompe ogni flusso di corrente all'interno del circuito.
Al cessare dell'eccitazione sul circuito l'energia elettrica accumulata nel condensatore torna a scaricarsi sotto forma di corrente elettrica rilasciata nel circuito.

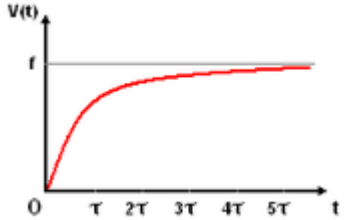
Il tempo di carica e di scarica dipende alla costante di tempo *Tau* definita come:

$$Tau = R*C$$

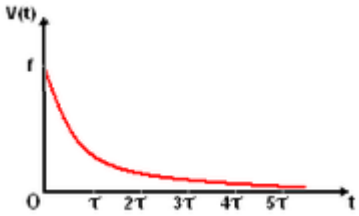
Nelle seguenti figure sono riportati i grafici relativi alla carica e alla scarica del condensatore. Nell'immagine di sinistra il condensatore, inizialmente scarico, si carica fino a raggiungere il livello di tensione finale *f*. Differentemente, nell'immagine di destra il condensatore, inizialmente carico e con una tensione *f* ai suoi capi, si scarica fino a raggiungere un livello di tensione finale pari a zero volt.

In questa esperienza si propone l'utilizzo del monitor seriale per visualizzare i tempi di carica del condensatore. E' importante considerare che i dati riportati sul monitor seriale (tempo e tensione) possono essere facilmente copiati in un foglio excel per ottenere una migliore rappresentazione grafica dell'evoluzione della tensione nel tempo.

L'utilizzo di un pulsante permette di avviare la carica del condensatore mediante l'utilizzo di una istruzione di digitalWrite.



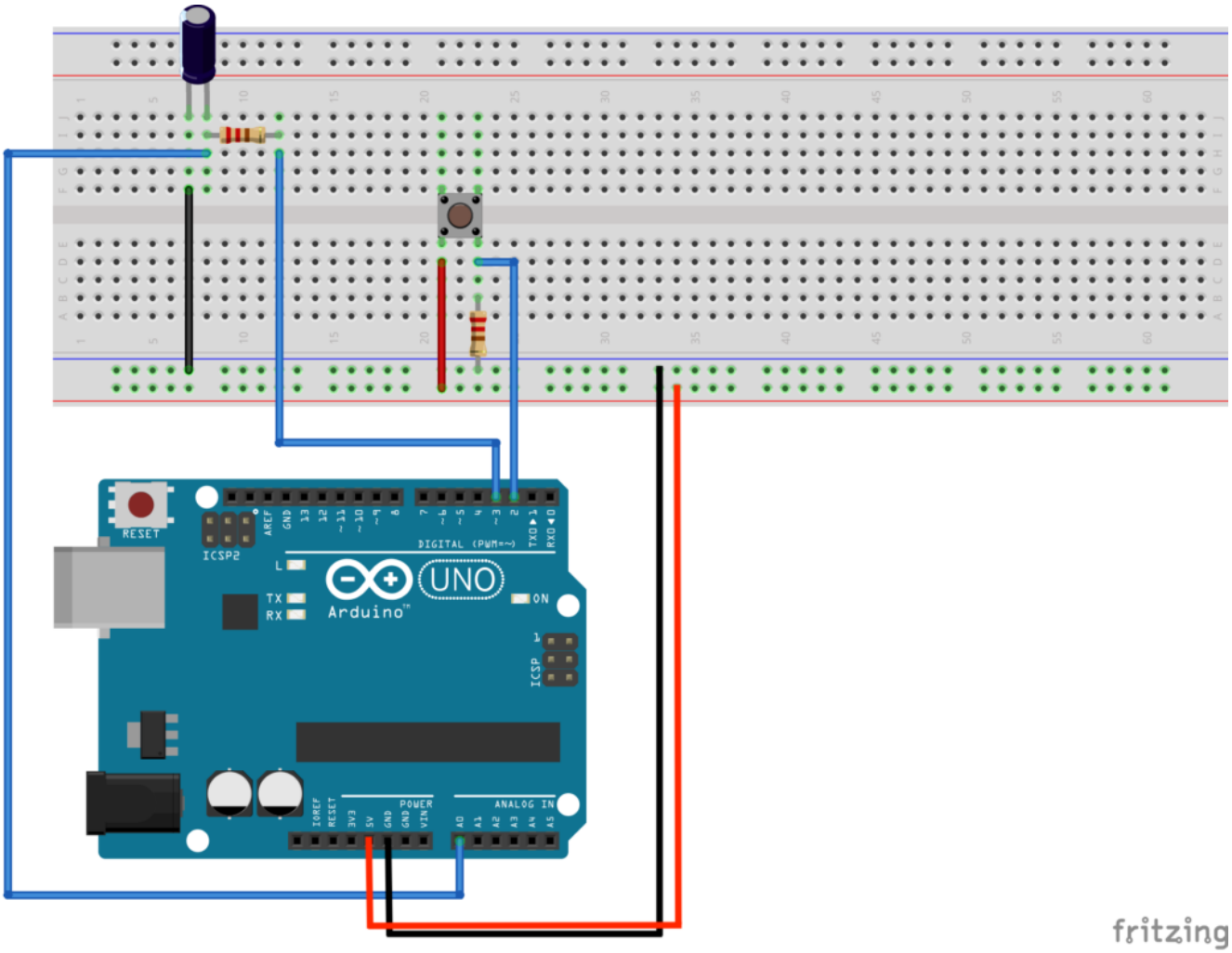
- Carica di un condensatore



Scarica di un condensatore

Collegamento Circuitale:

Viene in seguito riportato lo schema elettrico utilizzato per analizzare il tempo di carica di un condensatore. Nel dettaglio il tempo di carica viene monitorato utilizzando la lettura analogica del controllore Arduino.



Collegamento Circuitale

Codice:

Personalizzazioni:

E’ possibile modificare il codice e l’hardware per simulare anche la scarica del condensatore.

Creare funzioni con Arduino … per un Display a 7 Segmenti

Obiettivo: Creare una serie di funzioni con Arduino per utilizzare un display a 7 segmenti .

Componenti elettronici:

- Arduino UNO
- Breadboard
- 1 Display a 7 Segmenti
- 7 Resistenza (100 Ohm)

Pre-Requisiti:

1..2..3.. Il Display a 7 Segmenti

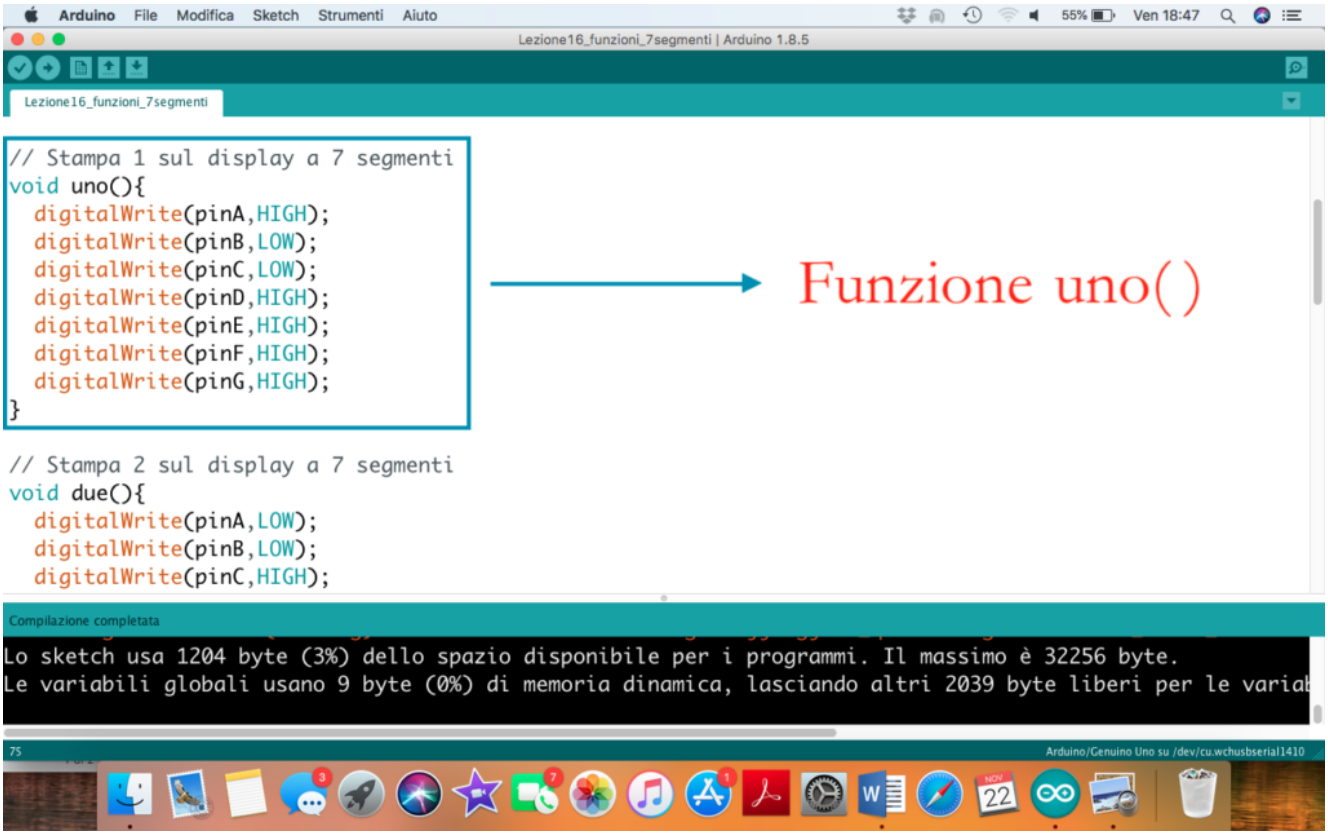
Teoria: La possibilità di riutilizzare il codice precedentemente scritto (**code re-use**) rappresenta una delle pratiche più comuni nella programmazione ovvero richiamare/invocare parti di codice precedentemente già sviluppate, ogni qualvolta risulta necessario, senza doverle riscrivere daccapo.

Nello specifico, questa possibilità si concretizza attraverso la scrittura di funzioni che possono essere richiamate/invocate all'interno dello stesso programma.

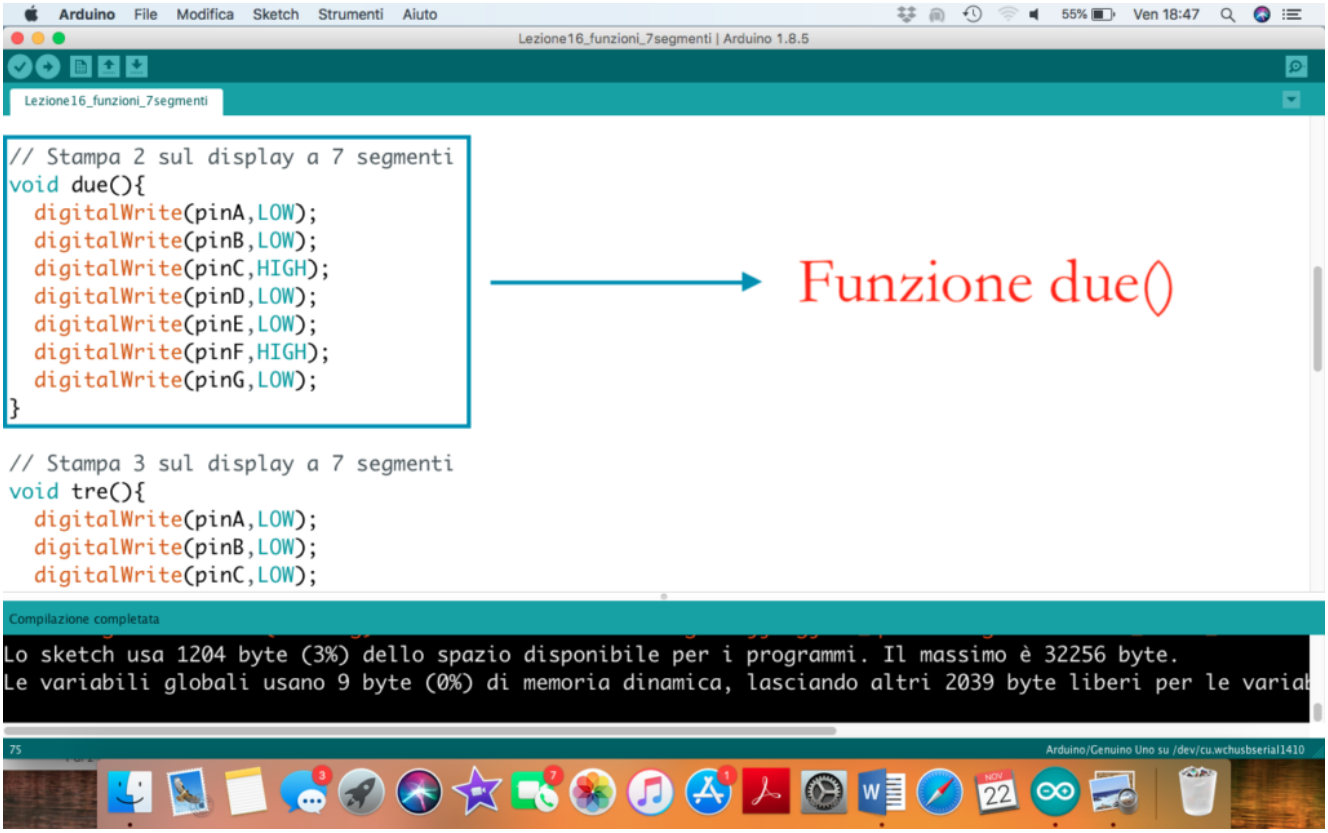
Queste funzioni possono svolgere diverse operazioni, che permettono di manipolare degli input dati ed eventualmente restituire degli output desiderati.

Nel caso in questione le tre funzioni implementate (denominate, uno, due e tre) non prevedono né il passaggio specifico di parametri di input né la restituzione di un output dato; queste tre funzioni svolgono solamente una serie di azioni sequenziali volte ad accendere alcuni elementi di un display a 7 segmenti. E' importante considerare che le tre funzioni sono tutte definite prima del loro utilizzo (ovvero prima delle due funzioni principali di setup e loop).

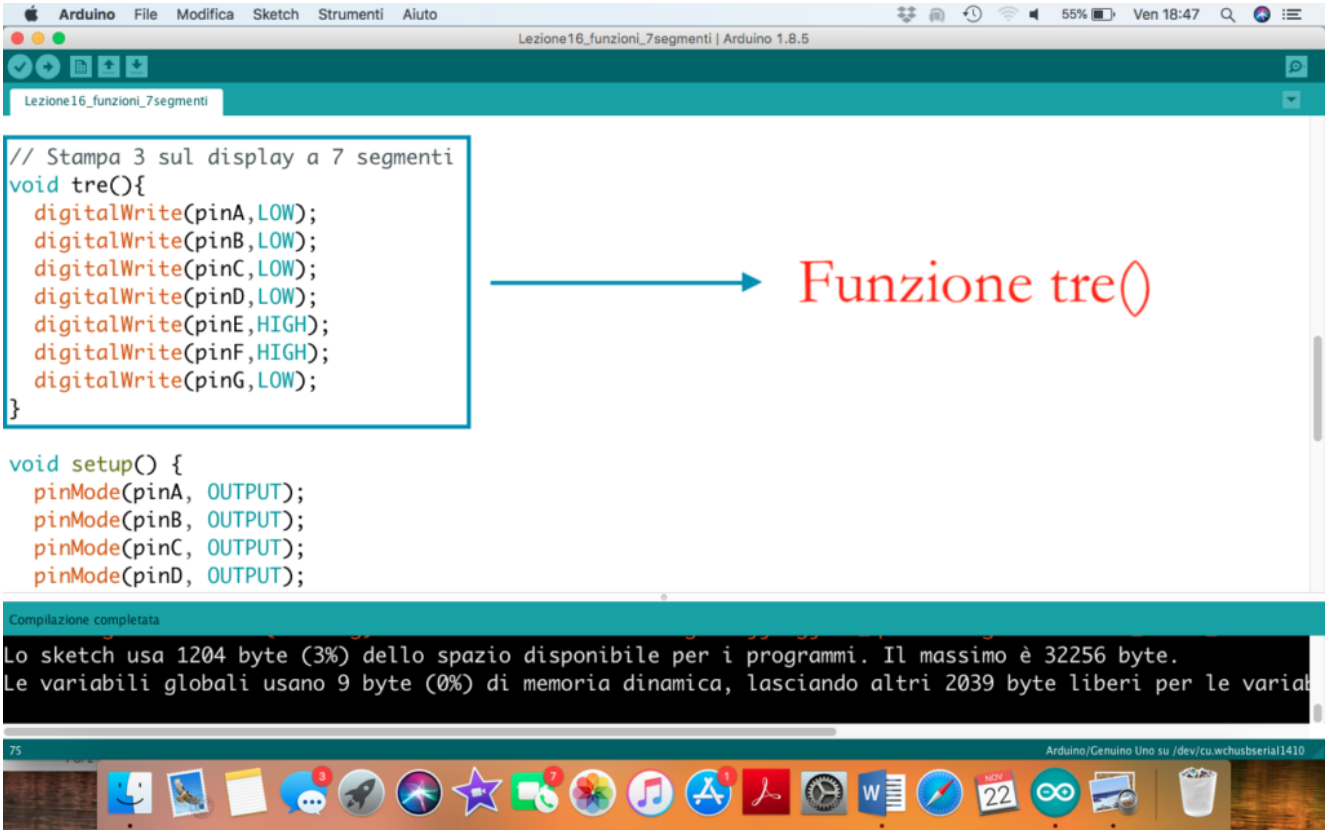
- **Funzione Uno()**: Stampa il carattere 1 sul display a 7 segmenti



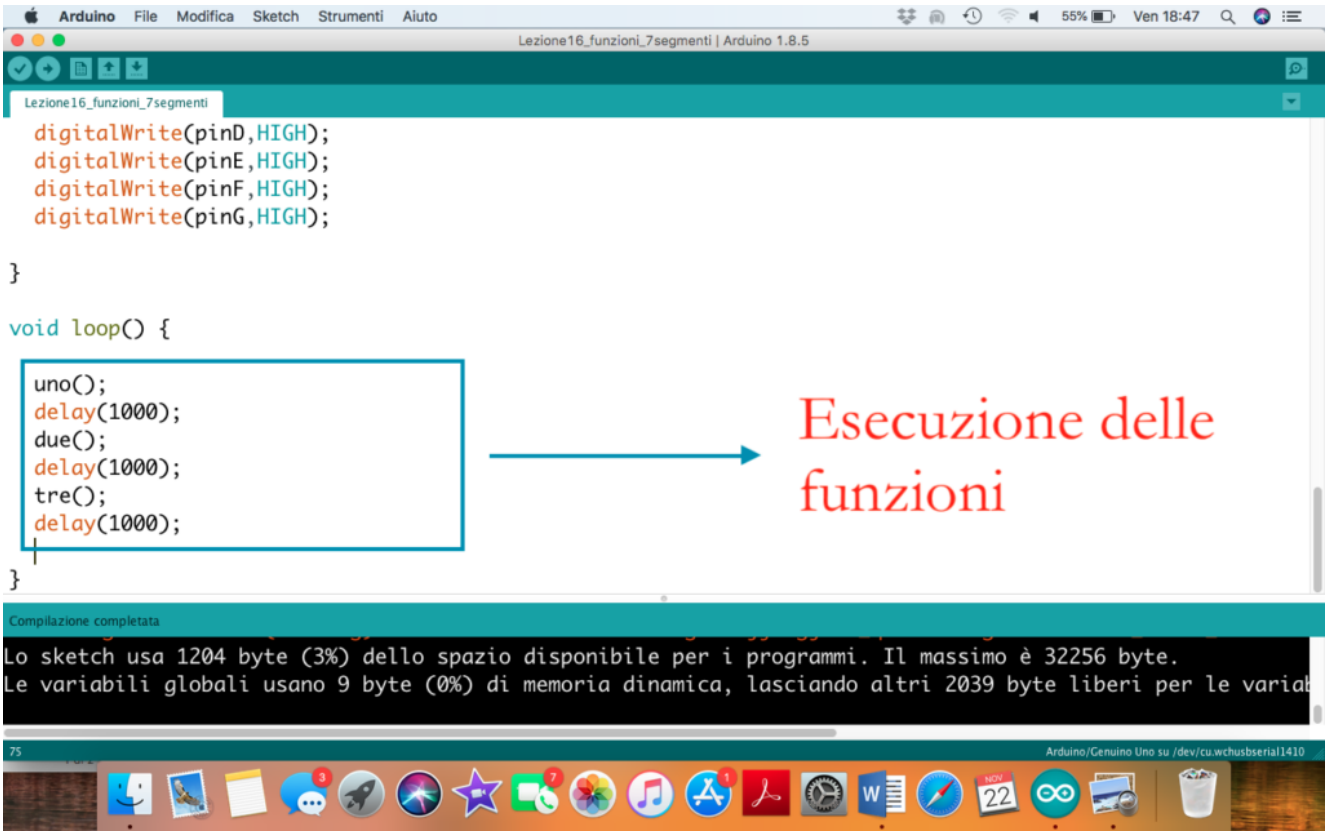
- **Funzione Due()**: Stampa il carattere 2 sul display a 7 segmenti



- **Funzione Tre()**: Stampa il carattere 3 sul display a 7 segmenti

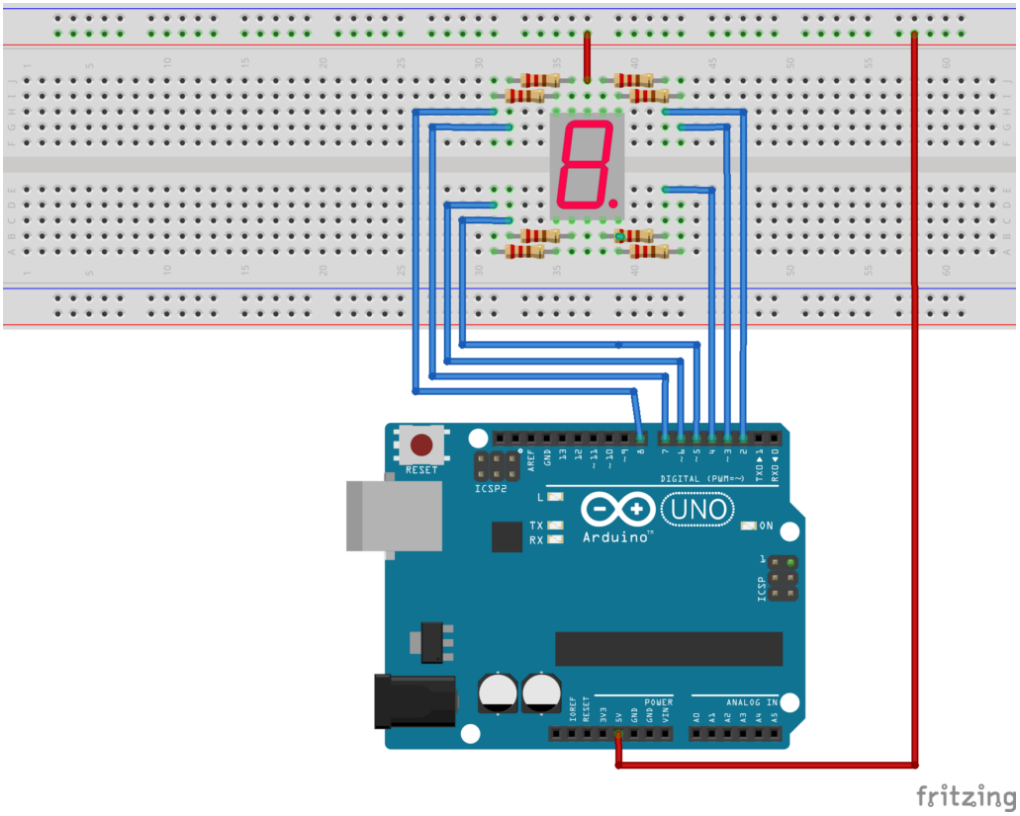


- **Invocazione delle Funzioni**: Le funzioni possono essere invocate semplicemente richiamandole nel punto in cui devono essere eseguite. Nel caso in questione le funzioni sono richiamate all'interno del blocco loop().



Collegamento Circuitale:

Viene in seguito riportato lo schema elettrico utilizzato per comandare un display a 7 segmenti mediante funzioni. Nel dettaglio il display impiegato è della modalità anodo comune.



Collegamento Circuitale

Codice:

Considerando l’impiego di un display ad anodo comune i singoli LED possono essere accesi mediante l’istruzione digitalWrite(pinLed,LOW);

Tre differenti funzioni sono state create per gestire il display. La funzioni sono, rispettivamente denominate, uno(), due() e tre(). Queste funzioni devono essere definite prima dei blocchi setup e loop, non prevedono l’impiego di parametri in ingresso né la restituzione di output in uscita (funzioni di tipo void).

Personalizzazioni:

E’ possibile modificare il software aggiungendo altre funzioni per la rappresentazione dei vari numeri sul display a 7 segmenti.

1..2..3.. Il Display a 7 Segmenti

Obiettivo: Utilizzo di un display a 7 segmenti .

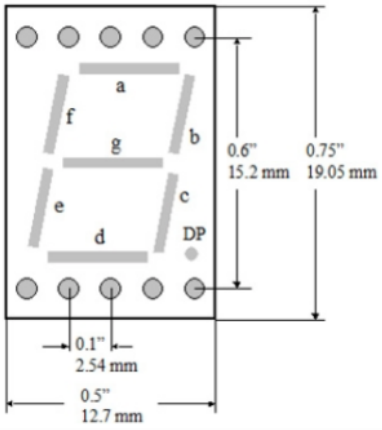
Componenti elettronici:

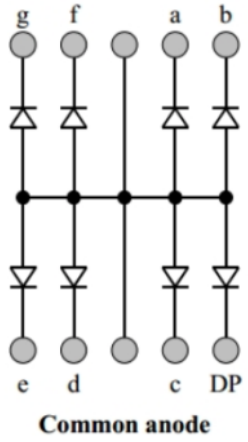
- Arduino UNO
- Breadboard
- 1 Display a 7 Segmenti
- 7 Resistenza (100 Ohm)

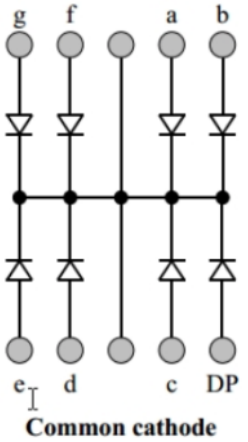
Teoria: Il display a 7 segmenti è un dispositivo elettronico in grado di visualizzare, attraverso l’accensione di combinazioni di sette differenti segmenti luminosi, le 10 cifre numeriche, alcune lettere alfabetiche e simboli grafici.

I display a 7 segmenti sono dotati di 10 differenti pin e sono così classificati:

- **Display 7 segmenti ad Catodo Comune:** il pin centrale del LED deve essere collegato a massa (GND) mentre gli altri 8 differenti PIN sono utilizzati per comandare i 7 segmenti ed il punto DP. Nel caso in questione i singoli LED sono accesi attraverso l'istruzione Arduino digitalWrite(pinLed, HIGH);
- **Display 7 segmenti ad Anodo Comune:** il pin centrale del LED deve essere collegato a 5V (VCC) mentre gli altri 8 differenti PIN sono utilizzati per comandare i 7 segmenti ed il punto DP. Nel caso in questione i singoli LED sono accesi attraverso l'istruzione Arduino digitalWrite(pinLed, LOW);

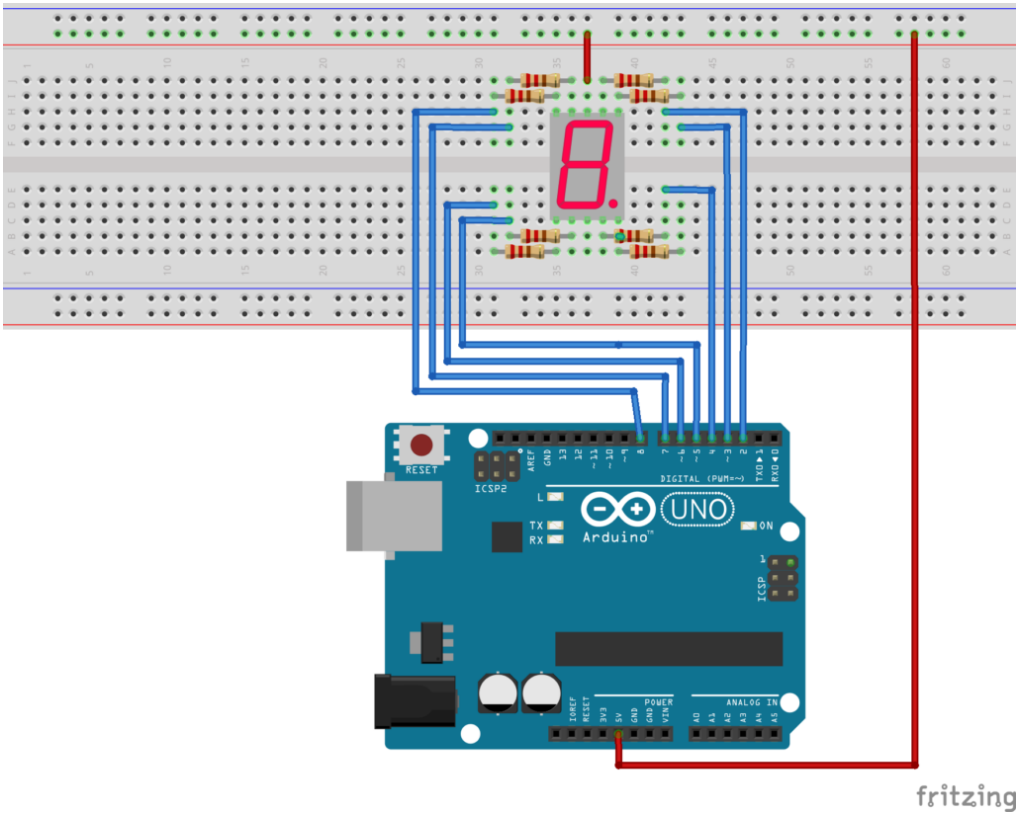






<u>Collegamento Circuitale</u>:

Viene in seguito riportato lo schema elettrico utilizzato per comandare un display a 7 segmenti. Nel dettaglio il display impiegato è della modalità anodo comune.



Collegamento Circuitale

Codice:

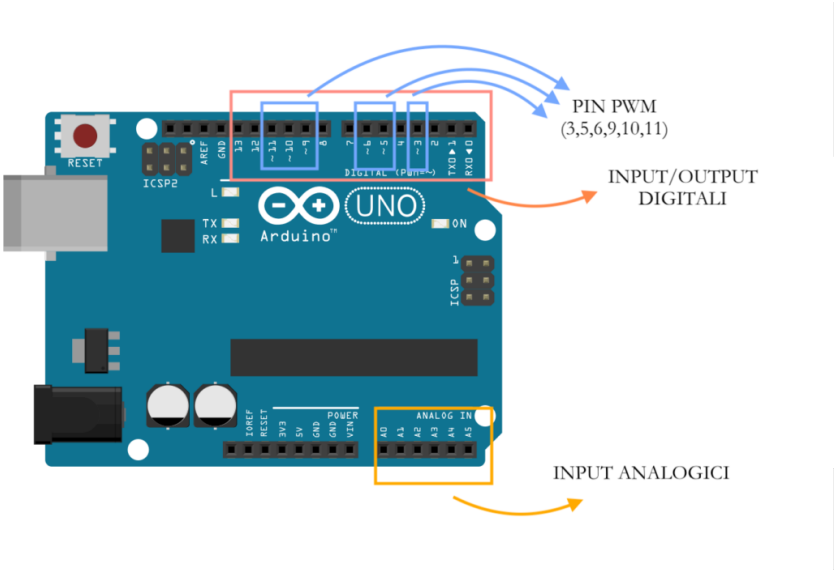
Considerando l’impiego di un display ad anodo comune i singoli LED possono essere accesi mediante l’istruzione digitalWrite(pinLed,LOW);

Personalizzazioni:

E’ possibile modificare il codice aggiungendo la possibilità di visualizzare altri numeri e non solo 1, 2 e 3.

Le Funzioni digitalWrite, digitalRead, analogWrite e analogRead

Obiettivo: Imparare ad utilizzare le principali funzioni di Arduino



Arduino Porte (Input, Output, Digitali, Analogici)

Teoria:

Le principali funzioni utilizzate da Arduino per comunicare con il mondo esterno sono quattro e si dividono in base alla **tipologia di azione:**

- Lettura: utilizzate per acquisire i dati dai differenti sensori (luminosità, temperatura, umidità, etc)
- Scrittura: utilizzate per comandare i differenti attuatori (motori, buzzer, display, etc)

ed in base alla **tipologia di segnale trattato:**

- Digitale: utilizzate per trattare segnali digitali che possono assumere solamente valori logici (i.e., LOW e HIGH)
- Analogico: utilizzate per trattare segnali analogici con valori compresi tra 0 e 5V.

Nello specifico queste quattro funzioni sono così definite:

Codice:

- **digitalWrite:** Funzione utilizzata per comandare attuatori mediante una logica LOW/HIGH come ad esempio motori, led o buzzer. Questa funzione prevede l'impiego di due parametri di input: il **PIN** (0-13) ed il **VALORE** (LOW/HIGH)

```
digitalWrite(pin, valore);
```

- **analogWrite:** Funzione utilizzata per comandare attuatori mediante una logica analogica (valori compresi tra 0V e

5V) come ad esempio motori o led. Questa funzione prevede l'impiego di due parametri di input: il **PIN** (0-13) ed il **VALORE** (0-255). Nel caso specifico il valore 0 corrisponde a 0V mentre 255 a 5V. Per tutti gli altri VALORI si può attuare la proporzione lineare. (Ad esempio volendo generare un riferimento di tensione pari a 3Volt il VALORE di input può essere così calcolato: (3/5)*255. E' importante considerare che i valori di tensione non sono "realmente" analogici ma generati attraverso la tecnica **PWM**. Inoltre, l'istruzione analogWrite può essere utilizzata solamente su alcuni pin digitali di output: i pin PWM (3,5,6,9,10,11).

analogWrite(pin, valore);

- **digitalRead:** Funzione utilizzata per leggere dati da sensori basati su una logica LOW/HIGH come ad esempio i pulsanti. Questa funzione prevede l'impiego di un parametro di input: il **PIN** (0-13) ed un parametro di output: il **VALORE** (LOW/HIGH) che viene restituito dalla funzione.

valore= **digitalRead**(pin);

- **analogRead:** Funzione utilizzata per leggere dati da sensori di tipo analogico (valori compresi tra 0V e 5V) come ad esempio fotoresistenze, sensori di temperatura,

umidità etc. Questa funzione prevede l’impiego di un parametro di input: il **PIN** (A0-A5) ed un parametro di output: il **VALORE** (0-1023). Nel caso specifico il valore 0 corrisponde a 0V mentre 1023 a 5V. Per tutti gli altri VALORI si può attuare la proporzione lineare. (Ad esempio se viene letto il VALORE 512, la tensione di riferimento può essere così calcolata: (512/1023)*5.

```
valore = analogRead(pin);
```

Quiz:

A che valore di tensione corrisponde l'intero 818 letto attraverso la funzione analogRead

4V
3V
2V
5V
Correct! Wrong!
(818/1023)*4 = 1V

A che valore di tensione corrisponde l'intero 204 letto attraverso la funzione analogRead

2V
4V
3V
1V
Correct! Wrong!
(204/1023)*5 = 1V

A che valore di tensione corrisponde l'intero 511 letto attraverso la funzione analogRead

5V
4V
2,5V
1V
Correct! Wrong!
(511/1023)*5 = 2.5V

A che valore di tensione corrisponde l'intero 255 generato utilizzando la funzione analogWrite

5V
0V
2.5V
1V
Correct! Wrong!
(255/255)*5 = 5V

A che valore di tensione corrisponde l'intero 100 generato utilizzando la funzione analogWrite

1V
2V
1.66V
1.96
Correct! Wrong!
(100/255)*5 = 1.96V

La funzione Valore = digitalWrite(11) è corretta?

Vero
Falso
Correct! Wrong!
La funzione digitalWrite non ha tipi di ritorno

La funzione digitalWrite(11,HIGH) è corretta?

Vero
Falso
Correct! Wrong!

La funzione analogWrite(11,1023) è corretta?

Vero
Falso
Correct! Wrong!
Il valore massimo di tensione per la funzione digitalWrite è pari a 255

Il LED RGB

Obiettivo: Accensione di un LED RGB (Red Green Blue) .

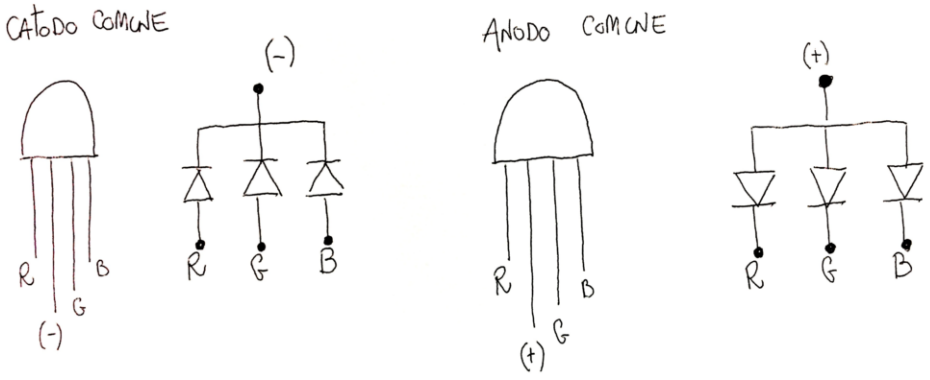
Componenti elettronici:

- Arduino UNO
- Breadboard
- 1 Led RGB
- 3 Resistenza (100 Ohm)

Teoria: Gli RGB rappresentano una particolare sottocategoria di LED capaci di riprodurre differenti colori attraverso la combinazione dei colori fondamentali Rosso Verde e Blu. Nel dettaglio, questi dispositivi sono costituiti da tre differenti led azionati in funzione del colore da riprodurre.

I LED RGB sono dotati di 4 differenti morsetti e sono così classificati:

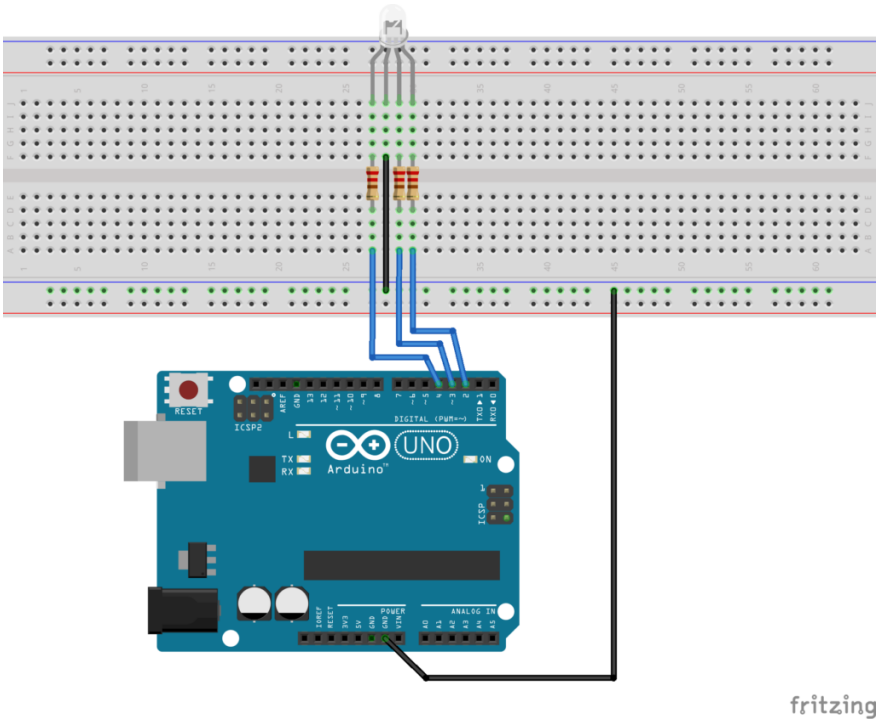
- **RGB ad Anodo Comune:** il pin più lungo del LED deve essere collegato a massa (GND) mentre gli altri 3 differenti PIN sono utilizzati per comandare i 3 LED (rosso, verde, blue). Nel caso in questione i singoli LED sono accesi attraverso l'istruzione Arduino digitalWrite(pinLed, HIGH);
- **RGB a Catodo Comune:** il pin più lungo del LED deve essere collegato a VCC (5V) mentre gli altri 3 differenti PIN sono utilizzati per comandare i 3 LED (rosso, verde, blue). Nel caso in questione i singoli LED sono accesi attraverso l'istruzione Arduino digitalWrite(pinLed, LOW);



LED RGB (Anodo e Catodo comune)

Collegamento Circuitale:

Viene in seguito riportato lo schema elettrico utilizzato per comandare il LED RGB. Nel dettaglio il LED impiegato è della modalità catodo comune.



Collegamento Circuitale

Codice:

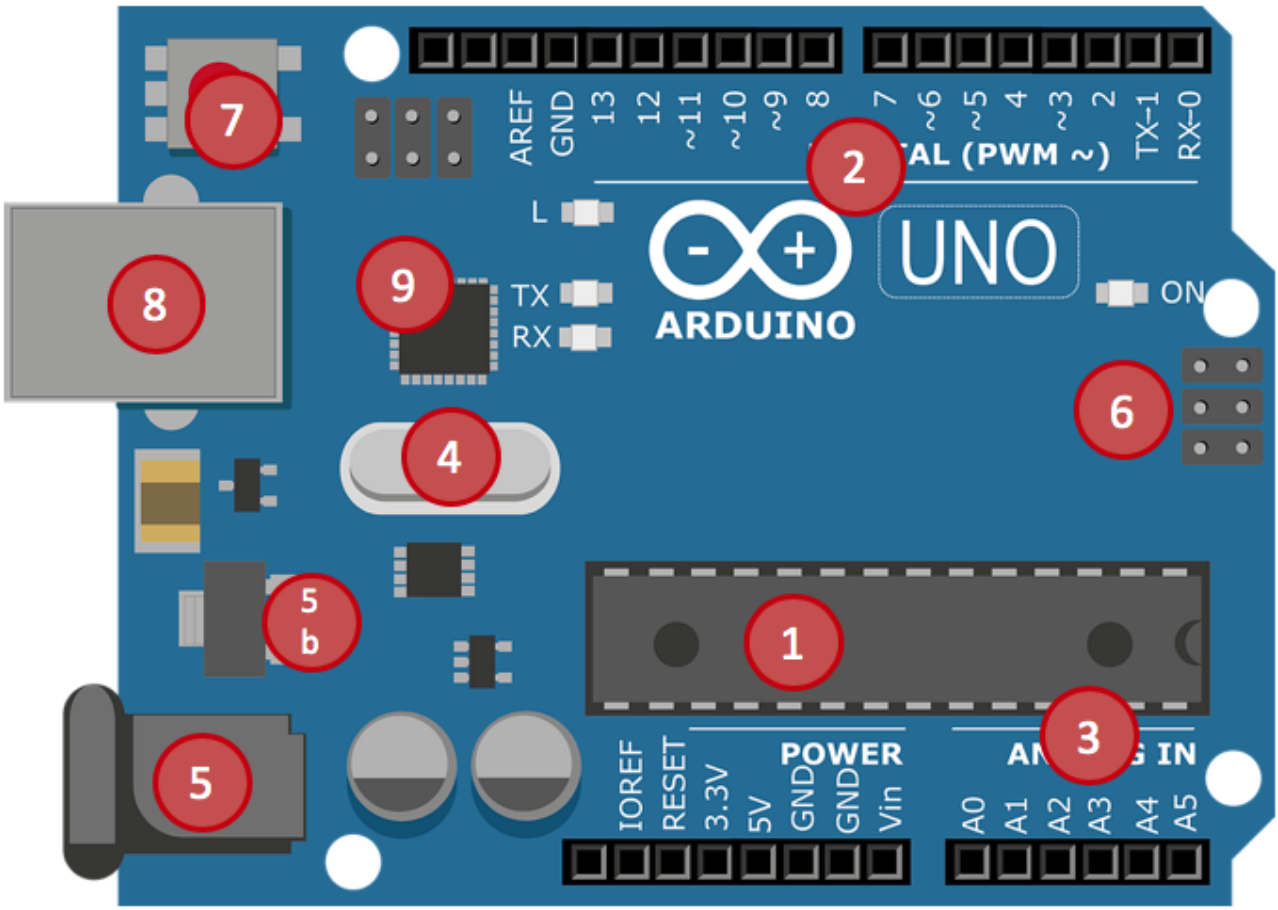
Considerando l'impiego di un LED a catodo comune i singoli LED (R, G, B) possono essere accesi mediante l'istruzione digitalWrite(pinLed,HIGH);

Personalizzazioni:

E' possibile modificare il circuito aggiungendo pulsanti

Caratteristiche Hardware

Obiettivo: Conoscere le principali caratteristiche Hardware di Arduino



Principali Componenti Hardware

Teoria:

Arduino UNO è una piattaforma HW dotata di Microcontrollore **ATMega328P**

1. **MCU Microcontrollore ATmega328P**. È un microcontrollore a 8 bit, in formato PDIP a 28 pin. L'MCU è dotata di 3 differenti tipologie di **memoria**: FLASH 32 KB (che includono anche il bootloader e la memoria programma), SRAM 2 KB (usata per memorizzare le variabili e le costanti del software) ed EEPROM 1KB (utilizzata per memorizzare le configurazioni)
2. Un totale di **14 pin di input/output digitali** programmabili, di cui 6 utilizzabili per fornire in uscita segnali modulati PWM. I pin PWM sono identificabili grazie al carattere tilde presente vicino al pin (3,5,6,9,10,11).
3. Un totale di **6 pin di input analogici**
4. **Oscillatore** a frequenza 16 MHz
5. **Jack per alimentazione** esterna (5b: **Regolatore di tensione**)
6. **Connettore ICSP** (*In Circuit Serial Programmer*) per effettuare la programmazione diretta del microcontrollore.
7. **Pulsante di Reset**
8. **Connettore USB** utilizzabile sia per alimentare la scheda che per programmare il microcontrollore
9. **Dispositivo per la comunicazione 16U2:** si occupa della conversione dei dati provenienti dall'USB in dati seriali (e viceversa) adatti per il microcontrollore.

Modalità di alimentazione:

- Alimentazione da pc con cavo usb (corrente max500mA): Se oltre al cavo usb alimentiamo Arduino anche tramite un connettore o dal pin Vin, verrà bypassata automaticamente l'alimentazione da usb e verrà utilizzata quella esterna.

- Connettore di alimentazione(corrente max800mA): La tensione nel rangetra 7 e 12 volte viene stabilizzata dall’integrato NCP1117
- Collegamento diretto al PIN Vin: Anche in questo caso la tensione è stabilizzata. Non è presente però il diodo di protezione non invertire la polarità.
- Collegamento diretto al PIN 5V: Tensione non stabilizzata. P**ERICOLO!!!**