CLASSE 4 B
A.S. 2017/18

DIARIO DI BORDO

PROGETTO ALTERNANZA SCUOLA LAVORO LIMITATORE DI VELOCITA'

F.C.A.

Sommario

Introduzione

ll progetto Alternanza Scuola-Lavoro che stiamo seguendo consiste nel simulare una situazione lavorativa reale come la gestione dei carrelli elevatori in transito nei magazzini di FCA.

Poiché la nostra classe appartiene ad un Istituto Tecnico Informatico, come compito ci è stato affidato un problema di natura elettronica e informatica. Occorre quindi proporre una soluzione e applicarla concretamente.

Per questo fine, è stato necessario sviluppare delle competenze di base per la programmazione digitale e per le architetture elettroniche semplici. In particolare abbiamo lavorato attraverso l'uso della scheda Arduino.

In questo documento abbiamo inserito i riferimenti al percorso d'apprendimento che ci ha permesso di progettare il lavoro commissionato dalla FCA, e soprattutto ci ha consentito di imparare nuove nozioni.

Questo percorso è costituito da una serie di relazioni su progetti realizzati con componenti di Arduino:

- Simulazione di due semafori ad un incrocio stradale;

- Accensione alternata di led;

- Controlli iterativi di attuatori;

- Controllo di un led con un pulsante;

- Utilizzo di un monitor seriale.

Arduino, non solo una "board"

Uno dei principali fenomeni degli ultimi anni che ha dato maggiore visibilità al software open source ma soprattutto al concetto di hardware open source è sicuramente quello dei "maker". Tale fenomeno va inquadrato come un vero e proprio movimento che può essere considerato una rivisitazione in chiave moderna del "fai da te" (DIY, Do It Yourself) per tutti coloro che sono fan dell'elettronica, della robotica e del software/firmware che permette di animare un qualsiasi oggetto di questo tipo.

È assolutamente innegabile che in questo campo, la scheda elettronica maggiormente utilizzata per scopi hobbistici e didattici ma in moltissimi casi anche per usi professionali sia Arduino.

Parlare di Arduino come di un'unica board è riduttivo, in quanto con questo nome viene considerato un intero progetto completamente italiano, nato ad Ivrea nel 2005 con lo scopo di ridurre i tempi di prototipizzazione dei prodotti elettronici ad un costo estremamente basso.

Si va infatti dalla "Arduino Uno" considerata come la entry level della famiglia alla "Arduino Yun" che attualmente rappresenta il top della gamma (con a bordo Linux) passando per tante altre board differenti.

ARDUINO UNO

ARDUINO YUN

Arduino Uno

La "Arduino Uno" è una board basata su un microcontrollore della Atmel, ATmega328, che ha a bordo tutta la memoria e lo storage necessario per i nostri programmi (che come vedremo sono chiamati "sketch"); in particolare abbiamo a disposizione:

- *32 KB di memoria Flash (di cui 0,5 KB sono già occupati dal bootloader), la memoria Flash è utilizzata per il salvataggio del nostro "sketch" che sarà lanciato in esecuzione ad ogni avvio della board;*
- *2 KB di SRAM (Static RAM), la memoria RAM è utilizzata a runtime (ad esempio per le variabili) ;*
- *1 KB di EEPROM (Electrically Erasable Programmable ROM), che serve per salvare eventuali dati e parametri di configurazione utili e/o necessari al nostro programma.*
  È da sottolineare che, in termini di capacità di calcolo, stiamo parlando di un microcontrollore con architettura RISC (Reduced Instruction Set Computer) ad 8-bit con una frequenza di clock di 16 Mhz (dimenticatevi i Ghz dei vostri PC).

I/O

I principali pin del microcontrollore sono messi a disposizione del "maker" attraverso due connettori posti ai bordi della scheda, attraverso i quali è banale collegare un qualsiasi altro dispositivo esterno che possa essere pilotato attraverso il nostro programma.
In particolare, sono disponibili: 14 pin digitali e 6 pin analogici, in quanto l'Atmega328 è dotato anche di un AD Converter (Analog to Digital) a 6 canali e con 10 bit di risoluzione.
Ciascuno dei pin digitali può essere pilotato in maniera indipendente da software, impostandone il livello 1 (alto) o 0 (basso), ed utilizzato come pin di GPIO (General Purpose Input Output); alcuni di questi pin hanno comunque delle funzionalità aggiuntive se opportunamente configurati ed in particolare :

- *Una porta seriale di tipo TTL (attenzione non RS232) caratterizzata dai soli segnali RX (Ricezione) e TX (trasmissione);*
- *Due pin per segnalare ("triggerare") un interrupt verso il microcontrollore, in corrispondenza del quale deve essere magari eseguita una specifica elaborazione;*
- *Segnali di output di tipo PWM (Pulse Width Modulation) per ottenere una tensione media variabile (che non sia solo alta o bassa) strettamente legata alla durata di un impulso nel tempo;*
- *Una porta SPI (Serial Peripheral Interface) utilizzata per l'interconnessione con device esterni attraverso appunto il bus sincrono di tipo SPI;*
- *Una porta I2C (Inter Integrated Circuit) che alla pari dell'SPI è utilizzata su un bus sincrono per la connessione ad altri device. Essa è anche nota come TWI (Two Wires Interface) essendo costituita da due soli segnali contro i quattro previsti dall'SPI;*
- *Il pin che stabilisce la tensione di riferimento (AREF, Analog REFerence) per tutti gli input analogici. Essa è necessario per stabilire il range di valori che l'AD Converter produce sulla base dei 10 bit a disposizione (in pratica esso fornirà un valore numerico compreso tra 0 e 1023, in corrispondenza ai valori di tensione da 0 a AREF Volts);*

Inoltre, la board mette a disposizione due segnali di alimentazione in uscita a 3,3V e 5V che possono essere utili per alimentare uno o più dispositivi esterni con i quali comunicare, un segnale di RESET e soprattutto un LED a bordo che utilizzeremo per la nostra prima applicazione "Hello World".

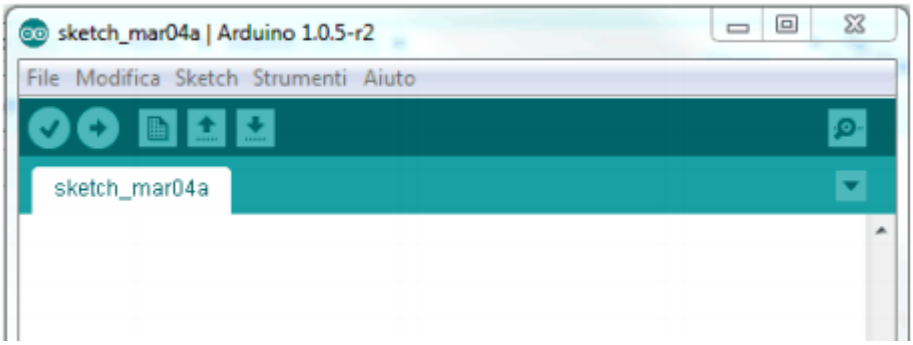
L'unica connettività diretta con il mondo esterno è caratterizzata da un connettore USB che possiamo sfruttare per il collegamento al nostro PC di sviluppo. Il PC avrà a disposizione una porta seriale per poter comunicare con la "Arduino Uno", grazie a degli opportuni driver da installare ed al convertitore "USB-to-serial" di cui la board è dotata (in pratica il microcontrollore non ha un'interfaccia nativa USB ma utilizza sempre la comunicazione di tipo seriale).

Un'importante considerazione da fare, per evitare di bruciare la nostra scheda al primo "esperimento", è che il microcontrollore ha una tensione operativa di 5V (massimo tollerabile su tutti i pin) ma possiamo utilizzare un alimentatore esterno tra i 6 ed i 20V. Inoltre, la corrente massima supportata su tutti i pin è pari a 40mA. La cosa interessante è che, trattandosi di hardware open source, tutti gli schemi elettrici sono disponibili online sul sito ufficiale di Arduino, per cui la scheda è assolutamente "replicabile" in casa; esistono, infatti, tantissime board "Arduino-like" non ufficiali.

In primo luogo, bisogna sottolineare che il linguaggio di programmazione utilizzato per Arduino è il C/C++; si può parlare di entrambi i linguaggi, in quanto è possibile utilizzare o meno il supporto per le classi che il C++ mette a disposizione. Inoltre, il compilatore utilizzato è ovviamente dedicato per i microcontrollori della famiglia AVR della Atmel e fa parte di una ben definita toolchain GCC.
Con un compilatore ed un semplice editor di testo è sempre possibile scrivere e compilare un programma ma, per aumentare la produttività e ridurre i tempi di sviluppo, si preferisce sempre utilizzare un IDE dedicato (se disponibile); per fortuna con Arduino abbiamo questa possibilità!

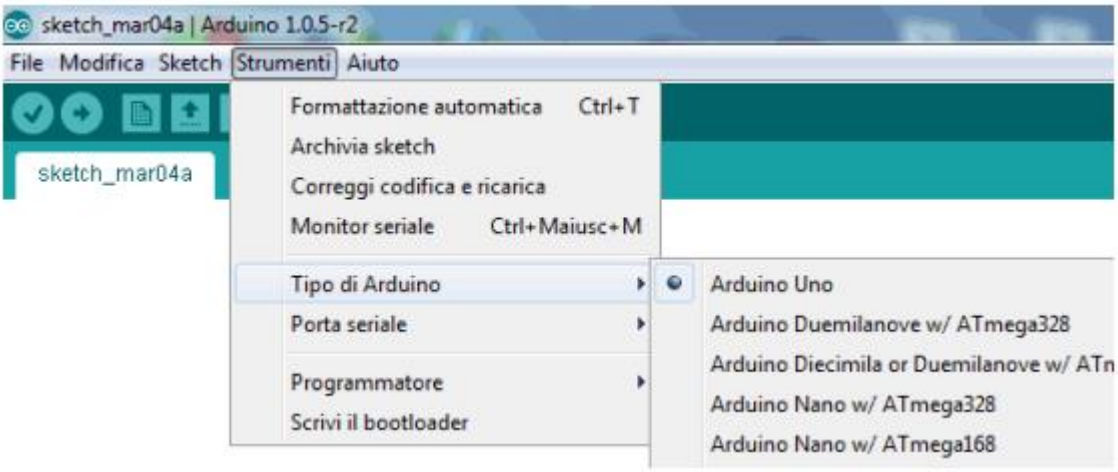
Sul sito ufficiale di Arduino è disponibile la sezione Download , dalla quale possiamo scaricare l'IDE per il sistema operativo che utilizziamo (sono supportati Windows, Linux e Mac OS X).
Una volta scaricato ed estratto il contenuto del file ZIP, clicchiamo sul file eseguibile arduino.exe che si trova nella cartella principale e dopo pochi secondi ci ritroveremo davanti ai nostri occhi l'unica finestra semplice ma essenziale dell'IDE completamente sviluppato in Java.

Arduino IDE



La prima operazione da fare consiste nel selezionare quale board della famiglia Arduino sarà utilizzata (nel nostro caso la "Arduino Uno") attraverso il menu "Strumenti e Tipo di Arduino".

Selezione del tipo di board Arduino da utilizzare



Prima di connettere la nostra board al PC attraverso il cavo USB, diamo uno sguardo all'IDE per visionare le funzionalità che ci mette a disposizione. Sempre nel menu Sketch è disponibile la voce Verifica/Compila che ci permette di verificare la correttezza del codice (ovviamente da un punto di vista sintattico e dei riferimenti alle funzioni utilizzate) e quindi compilarlo ma senza caricarlo immediatamente sulla board.
Infine, nel menu Modifica troviamo una serie di funzioni strettamente legate all'editor per poter eseguire semplicemente le operazioni di commento del codice, di ricerca, di copia/incolla e di indentazione.

Alcune delle operazioni più comuni sono accessibili attraverso una serie di pulsanti posti immediatamente al di sotto alla barra dei menu e che rispettivamente indicano :

- *Verifica : esegue la verifica del codice scritto e relativa compilazione;*
- *Carica : esegue il caricamento sulla board del firmware compilato;*
- *Nuovo : permette di creare un nuovo sketch;*
- *Apri : permette di aprire uno sketch esistente;*
- *Salva : permette di salvare lo sketch correntemente aperto;*

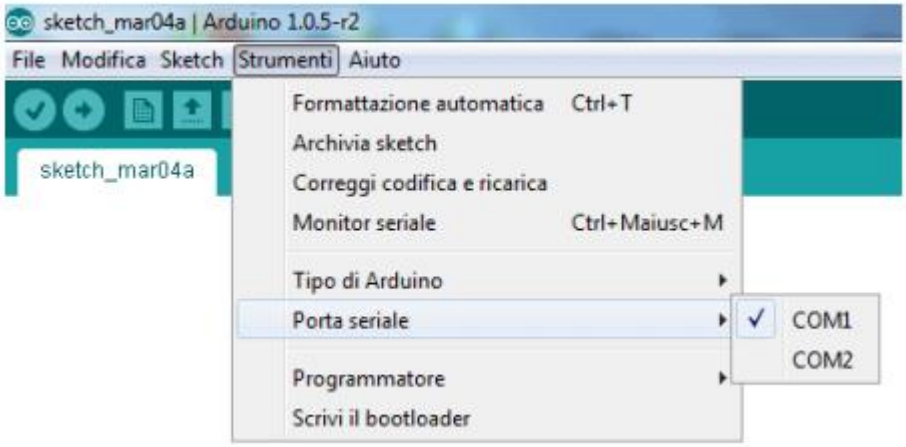
Pulsati per accesso diretto alle principali funzioni

Collegare la scheda A questo punto possiamo passare al collegamento della nostra board al PC attraverso il cavo USB. Una volta collegata ed alimentata la scheda (anche solo attraverso il cavo USB), il PC dovrebbe riconoscere la presenza della board (es. "Arduino Uno") ma necessita dei driver per una corretta installazione.

All'interno della cartella drivers dell'archivio che abbiamo scaricato è possibile trovare i due file eseguibili (dpinst-x86.exe e dpinst-amd64.exe), che permettono l'installazione della scheda rispettivamente su un sistema operativo a 32 e 64 bit.

Una volta completata l'installazione, le porte seriali già disponibili sul PC con in più quella appena installata e relativa alla board Arduino, saranno visibili nel menu Strumenti > Porta seriale; dobbiamo ovviamente selezionare la porta giusta alla quale è collegata la scheda. Per verificare quale sia la porta seriale giusta, possiamo individuare tutte le porte installate nel sistema all'interno del pannello di controllo, dove troviamo una porta del tipo Arduino Uno (COMx), che altro non è che una porta seriale virtuale associata al convertitore USBseriale a bordo della scheda.

Impostazione porta seriale relativa alla board Arduino



Struttura tipica di uno sketch

Prima di procedere allo svolgimento della relazione posta in esame, è fondamentale essere a conoscenza di due elementi della piattaforma Arduino: la scheda e il software di programmazione.

La scheda è la parte fisica nella quale vengono assemblati i vari componenti di Arduino.

Il software ci permette di programmare e inviare istruzioni alla scheda mediante la connessione tramite cavo.

Quando si apre un nuovo sketch verranno visualizzati due spazi con la seguente configurazione:

void setup(){

...ISTRUZIONI...
}
void loop(){

...ISTRUZIONI...
}

*ll termine **void** indica delle procedure che non devono restituire nulla e che servono solo a introdurre delle funzioni. Le void "fondamentali" nella scrittura di uno sketch sono due: **void setup** e **void loop**.*

Tra le parentesi graffe sarà necessario scrivere il codice con i comandi da fare eseguire alla scheda. Per capire il suo funzionamento ci soffermeremo sulle due procedure sopra citate e le spiegheremo brevemente qui di seguito:

- *voidsetup ()*

  *In questa procedura diamo alla scheda tutte le informazioni necessarie prima dell'esecuzione di un programma. Per esempio impostiamo un mode su alcune porte della scheda come **INPUT** oppure **OUTPUT** con l'istruzione "pinmode(pin,mode)". Una particolarità del void setup è che può contenere anche dei comandi da fare eseguire la scheda, ma vengono **ripetuti solo una volta** appena la scheda viene collegata ad una fonte di energia.*

- *void loop()*

  *Nel **void loop** forniamo alla scheda tutte le informazioni relative all'esecuzione del programma, e tutti i comandi ad esso relativi. La particolarità del void loop è che i comandi vengono **ripetuti sempre in ordine**, e quando il codice termina, il void loop riparte da capo.*

Le istruzioni che ci è possibile usare, vanno in base al tipo di dato da inviare o ricevere che può essere digitale, analogico o seriale:

- ***digitalRead (pin)**: ci permette di leggere lo stato di un pin digitale;*

- ***digitalWrite(pin,valore)****: ci permette di impostare un pin digitale con i valori booleaniHIGH o LOW;*

- ***analogread(pin)****:ci permette di leggere lo stato di un pin analogico;*

- ***analogWrite(pin,valore)****: ci permette di impostare un pin anologico con un valore che oscilla tra 0 (equivalente a 0 volt) e 255 (equivalente a 5 volt) e che può essere ottenuto facendo una semplice proporzione.*

-***Serial.begin(9600)*** *= vuol dire la scheda arduino uno, dopo esser stata collegata alla porta usb, dovrà trasferire I dati ad una velocità di 9600 byte;*

-***Serial.println("TESTO")*** *= questo ci permetterà di stampare sul monitor del nostro pc desktop il testo che noi vorremmo visualizzare all'utente;*

-***Serial.available()*** *= funzione che legge il numero di caratteri inseriti da tastiera nel monitor seriale;*

-***Serial.read()*** *= funzione che r*estituisce il codice ASCII del carattere ricevuto nel monitor seriale.

Simulazione di due semafori ad un incrocio stradale

Titolo:

Simulazione di due semafori ad un incrocio stradale

Obiettivo:

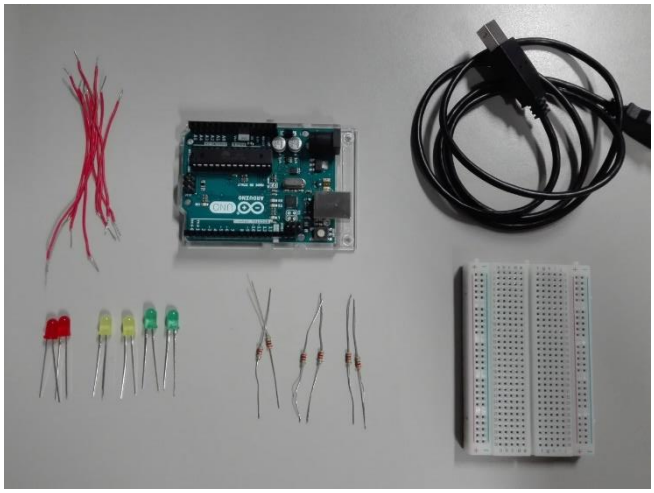
L'obiettivo di tale esperienza è stato quello di apprendere la progettazione, con l'ausilio di una scheda Arduino, di un semplice programma che simulasse un reale incrocio semaforico stradale.

Materiali e strumenti utilizzati **:**

Per la realizzazione e progettazione in oggetto, sono stati usati i seguenti materiali:

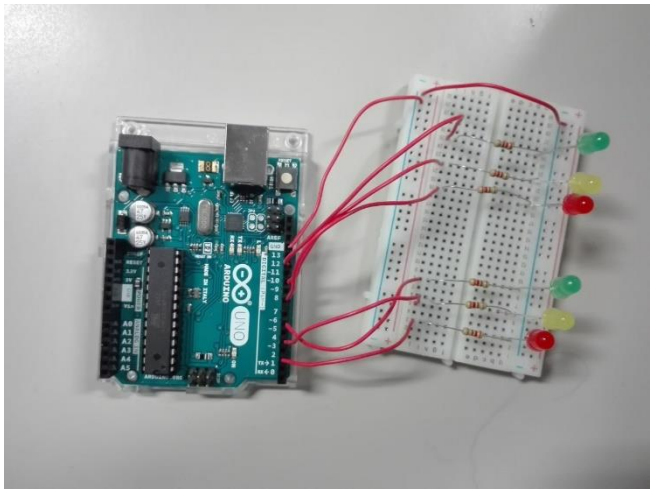
1. *Un Pc desktop (utilizzato per varie compilazioni e test);*
2. *IDE Arduino (applicazione multi-piattaforma in Java, installato sul Pc desktop);*
3. *Scheda Arduino Uno;*
4. *Breadboard (basetta sperimentale);*
5. *Cavo USB ti tipo A/B M-M (Il connettore A è usato per collegarlo ad un PC, mentre l'altro connettore B per collegarlo alla scheda di Arduino)*
6. *Cavi ponticello per effettuare collegamenti e/o ponti traBreadBoard e scheda Arduino;*
7. *Un numero di led necessari pari a:*
    a. *2 rossi;*
    b. *2 gialli;*
    c. *2 verdi;*
8. *Un numero di resistenze necessarie pari a 6 di 120Ω ;*
9. *Pinze;*
10. *Spella-cavi.*

***Immagini di materiali, fase di progettazione e test*:**



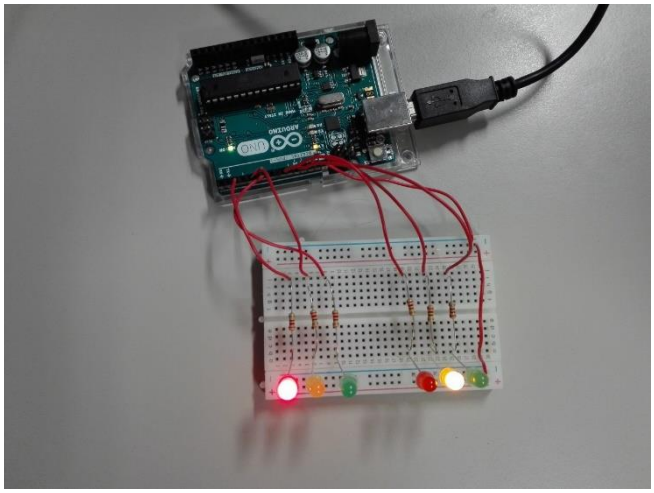
₁Fase di preparazione dei componenti:

Preparazione dei componenti necessari per lo sviluppo dell'esperienza in laboratorio



₂Fase di progettazione:

Tutti i componenti vengono assemblati secondo lo schema tecnico fornito dal docente



₃Fase di test:

Il codice viene caricato sulla scheda per la verifica di eventuali errori commessi nella scrittura del codice e nell'assemblaggio dei componenti

Parte Sperimentale :

Questa sezione può essere divisa in semplici fasi:

1. Fase di progettazione

Collegamenti fisici:

La prima operazione da svolgere per l'esecuzione del progetto è stata quella di:

- *riprodurre lo schema indicato dal Prof. Sicca sulla nostra BreadBoard consegnata precedentemente (inserendo led e resistenze secondo la pianta prestabilita).*
- *Successivamente si è passato al collegamento tra la BreadBoard e la scheda Arduino UNO, grazie al supporto dei cavi ponticello*

Stesura del Codice:
Per la stesura del codice ci siamo serviti del IDE Arduino.

Sono state inizialmente dichiarate in globale le variabili dei led collegati ai Pin di riferimento.

Successivamente siamo passati all'inizializzazione del Void Setup dove sono stati impostati i diversi pinMode in modalità OUTPUT mentre per il Void Loop sono stati impostati i diversi digitalWrite e delay a seconda dei casi proposti per la giusta esecuzione del progetto

2. Fase di test

In questa fase, dopo aver effettuato il collegamento tramite cavo USB tra Pc desktop e scheda Arduino Uno, abbiamo verificato e compilato, attraverso l'opzione prevista dal software "Verifica/Compila" che il codice trascritto non avesse errori.

Per ultimo, verificato se il codice fosse privo di errori, abbiamo caricato sulla scheda ARDUINO UNO, il codice precedentemente Verificato e compilato.

CODICE

```
int R = 2, G = 3, V = 4, r = 8, g = 9, v = 10;

void setup()
{

pinMode (R,OUTPUT);
pinMode (G,OUTPUT);
pinMode (V,OUTPUT);
pinMode (r,OUTPUT);
pinMode (g,OUTPUT);
pinMode (v, OUTPUT);
}

void loop() {

digitalWrite (R,HIGH);
digitalWrite (G, LOW);
digitalWrite (V, LOW);
digitalWrite (r,LOW);
digitalWrite (g, LOW);
digitalWrite (v, HIGH);
delay (10000);

digitalWrite (R,HIGH);
digitalWrite (G, LOW);
digitalWrite (V, LOW);
digitalWrite (r,LOW);
digitalWrite (g, HIGH);
digitalWrite (v, LOW);
delay (7000);

digitalWrite (R, LOW);
digitalWrite (G, LOW);
digitalWrite (V, HIGH);
digitalWrite (r,HIGH);
digitalWrite (g, LOW);
digitalWrite (v, LOW);
delay (10000);

digitalWrite (R, LOW);
digitalWrite (G, HIGH);
digitalWrite (V, LOW);
digitalWrite (r,HIGH);
digitalWrite (g, LOW);
digitalWrite (v, LOW);
delay (7000);
}
```

Elaborazione dati :

Nella sottostante tabella di verità sono stati riportati i risultati di tale progetto:

	Stato led semaforo 1			***Stato led semaforo 2***		
Delay (seconds)	Rosso	Giallo	Verde	Rosso	Giallo	Verde
10	**HIGH**	LOW	LOW	LOW	LOW	**HIGH**
7	**HIGH**	LOW	LOW	LOW	**HIGH**	LOW
10	LOW	LOW	**HIGH**	**HIGH**	LOW	LOW
7	LOW	**HIGH**	LOW	**HIGH**	LOW	LOW

Accensione alternata dei led

Titolo:

Accensione alternata di led

Obiettivo:

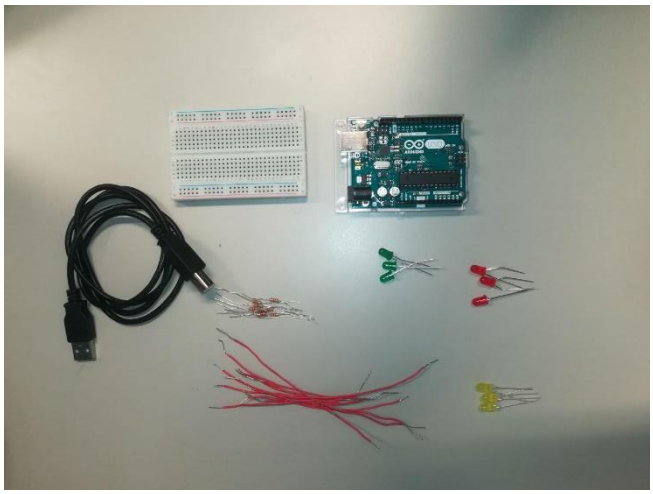
L'obiettivo di tale esperienza è stato quello di apprendere la progettazione, con l'ausilio di una scheda Arduino, di un semplice programma che simulasse un accensione alternata di led.

Materiali e strumenti utilizzati **:**

Per la realizzazione e progettazione in oggetto, sono stati usati i seguenti materiali:

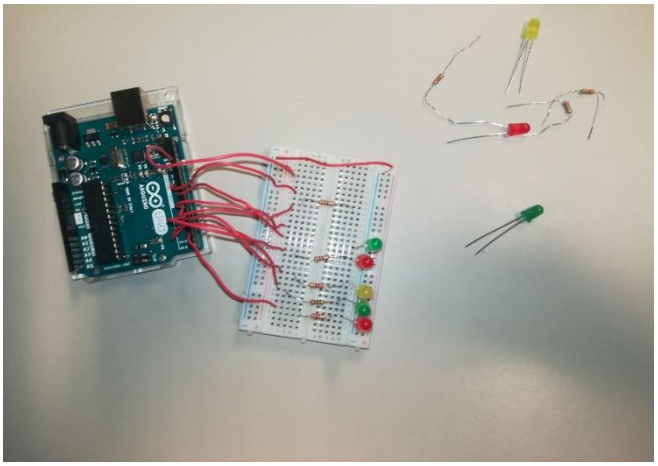
1. *Un Pc desktop (utilizzato per varie compilazioni e test);*
2. *IDE Arduino (applicazione multi-piattaforma in Java, installato sul Pc desktop);*
3. *Scheda Arduino Uno;*
4. *Breadboard (basetta sperimentale);*
5. *Cavo USB ti tipo A/B M-M (Il connettore A è usato per collegarlo ad un PC, mentre l'altro connettore B per collegarlo alla scheda di Arduino)*
6. *Cavi ponticello per effettuare collegamenti e/o ponti tra BreadBoard e scheda Arduino;*
7. *Un numero di led necessari pari a 8:*
    a. *3 rossi;*
    b. *2 gialli;*
    c. *3 verdi;*
8. *Un numero di resistenze necessarie pari a 8 di 120Ω ;*
9. *Pinze;*
10. *Spella-cavi.*

***Immagini di materiali, fase di progettazione e test*:**



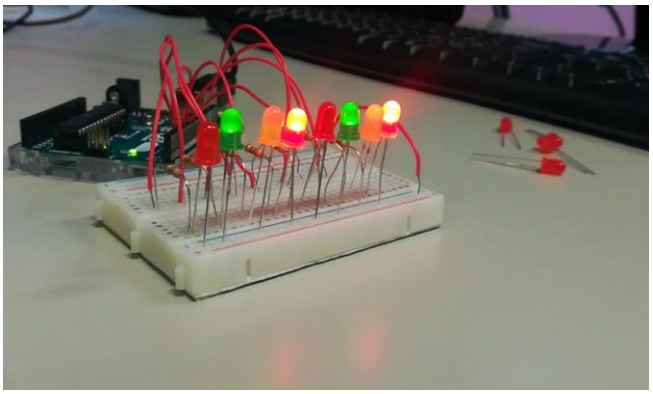
$_{1}$Fase di preparazione dei componenti:

Preparazione dei componenti necessari per lo sviluppo dell'esperienza in laboratorio



$_{2}$Fase di progettazione:

Tutti i componenti vengono assemblati secondo lo schema tecnico fornito dal docente



$_{3}$Fase di test:

Il codice viene caricato sulla scheda per la verifica di eventuali errori commessi nella scrittura del codice e nell'assemblaggio dei componenti

Parte Sperimentale

Questa sezione può essere divisa in semplici fasi:

1. Fase di progettazione

Collegamenti fisici:

La prima operazione da svolgere per l'esecuzione del progetto è stata quella di:

- *riprodurre lo schema indicato dal Prof. Sicca sulla nostra BreadBoard consegnata precedentemente (inserendo led e resistenze secondo la pianta prestabilita).*
- *Successivamente si è passato al collegamento tra la BreadBoard e la scheda Arduino UNO, grazie al supporto dei cavi ponticello*

Stesura del Codice:
Per la stesura del codice ci siamo serviti del IDE Arduino.

Sono state inizialmente dichiarate in globale dei vettori di riferimento.

Successivamente siamo passati all'inizializzazione del void setup dove sono stati inizializzati i vettori con l'ausilio di cicli per caricarli poi con i numeri dei pin utilizzati e poi dopo dichiarati come OUTPUT. Nel void loop sono stati utilizzati nuovamente i cicli per far eseguire le istruzioni di accensione e spegnimento(con i delay associati) secondo la tabella di verità prestabilita dal docente.

2. Fase di test

In questa fase, dopo aver effettuato il collegamento tramite cavo USB tra Pc desktop e scheda Arduino Uno, abbiamo verificato e compilato, attraverso l'opzione prevista dal software "Verifica/Compila" che il codice trascritto non avesse errori.

Per ultimo, verificato se il codice fosse privo di errori, abbiamo caricato sulla scheda ARDUINO UNO, il codice precedentemente Verificato e compilato.

CODICE

```
int led_pari  [4];
int led_dispari [4];


void setup() {

 for (int i = 0; i < 4; i++){
  led_pari[i] =(i+1)*2;
  led_dispari[i] = (( i+1)*2) +1;
  pinMode (led_pari [i], OUTPUT);
  pinMode (led_dispari[i], OUTPUT);
 }

}
void loop() {

     for (int i=3; i >= 0; i--)
    {
      digitalWrite (led_dispari [i], HIGH);
      digitalWrite (led_pari[i], LOW);
                    digitalWrite (led_dispari[i], LOW);
      digitalWrite (led_pari[i], HIGH);
    }
    delay (4000);


    for (int i=3; i >= 0; i--)
    {
      digitalWrite (led_dispari[i], LOW);
      digitalWrite (led_pari[i], LOW);
    }
    delay (1000);
/* ---------------------------------------------------------------- */
    for (int i=3; i >= 0; i--)
    {
      digitalWrite (led_dispari[i], HIGH);
      digitalWrite (led_pari[i], LOW);
      digitalWrite (led_dispari[i], LOW);
      digitalWrite (led_pari[i], HIGH);
    }
    delay (800);

}
```

Controlli iterativi di attuatori

Titolo:

Controlli iterativi di attuatori

Obiettivo:

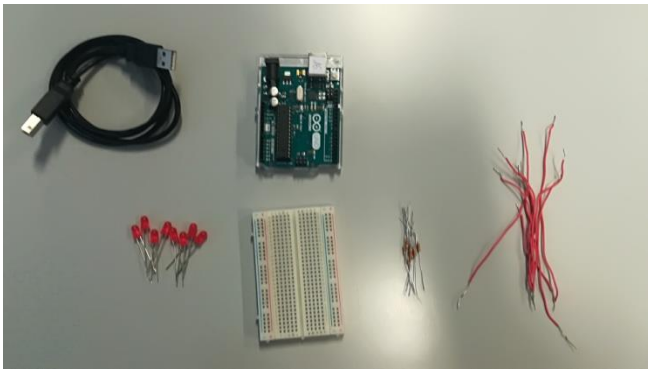
L'obiettivo di tale esperienza è stato quello di apprendere la progettazione, con l'ausilio di una scheda Arduino, di un semplice programma che permettesse di eseguire dei controlli iterativi di attuatori.

Materiali e strumenti utilizzati :

Per la realizzazione e progettazione in oggetto, sono stati usati i seguenti materiali:

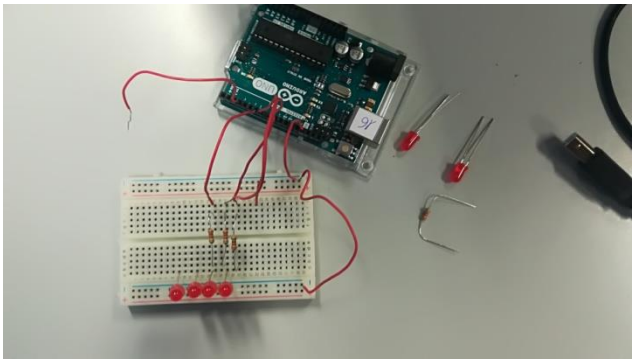
1. *Un Pc desktop (utilizzato per varie compilazioni e test);*
2. *IDE Arduino (applicazione multi-piattaforma in Java, installato sul Pc desktop);*
3. *Scheda Arduino Uno;*
4. *Breadboard (basetta sperimentale);*
5. *Cavo USB ti tipo A/B M-M (Il connettore A è usato per collegarlo ad un PC, mentre l'altro connettore B per collegarlo alla scheda di Arduino);*
6. *Cavi ponticello per effettuare collegamenti e/o ponti tra BreadBoard e scheda Arduino;*
7. *Un numero di led necessari pari a 1;*
8. *Un numero di resistenze necessarie pari a 1 di 120Ω;*
9. *Pinze;*
10. *Spella-cavi.*

***Immagini di materiali, fase di progettazione e test*:**



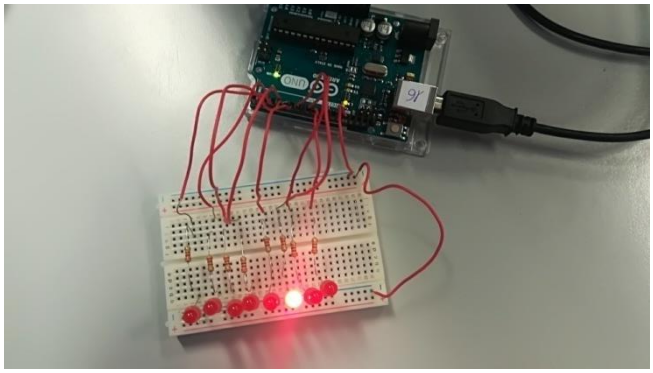
₁Fase di preparazione dei componenti:

Preparazione dei componenti necessari per lo sviluppo dell'esperienza in laboratorio



₂Fase di progettazione:

Tutti i componenti vengono assemblati secondo lo schema tecnico fornito dal docente



₃Fase di test:

Il codice viene caricato sulla scheda per la verifica di eventuali errori commessi nella scrittura del codice e nell'assemblaggio dei componenti

Parte Sperimentale :

Questa sezione può essere divisa in semplici fasi:

1. Fase di progettazione

Collegamenti fisici:

Nella parte iniziale, è necessario riprodurre il progetto cartaceo fornito sulla basetta sperimentale e sulla scheda arduino uno, effettuando I vari collegamenti e inserimenti dei componenti necessari in modo tale da ottenere la stessa "pianta".

Stesura del Codice:
Per la stesura del codice ci siamo serviti del IDE Arduino.

E' stata inizialmente dichiarata in globale un variabileintera che verrà poi usata per eseguire delle operazioni sui led.

Successivamente siamo passati all'inizializzazione del Void Setup dove sono stati impostati i diversi pinMode in modalità OUTPUT con l'ausilio di un ciclo for usando la variabile dichiarata inizialmente. Dopo ciò, abbiamo proceduto a scrivere ilVoid Loop. In esso sono stati impostati, in due cicli for, i diversi digitalWrite e delay a seconda dei casi proposti per la giusta esecuzione del progettoin modo tale da ottenere: in un ciclo un'accensione di led da sinistra a destra e nell'altro ciclo for il caso inverso, in modo tale da ottenere un'accensione sincronizzata di led che si incroci.

2. Fase di test

In questa fase, dopo aver effettuato il collegamento tramite cavo USB tra Pc desktop e scheda Arduino Uno, abbiamo verificato e compilato, attraverso l'opzione prevista dal software "Verifica/Compila" che il codice trascritto non avesse errori.

Per ultimo, verificato se il codice fosse privo di errori, abbiamo caricato sulla scheda ARDUINO UNO, il codice precedentemente Verificato e compilato.

CODICE

```
int ledpin;
void setup() {

for (ledpin = 10; ledpin > 1; ledpin--)
 {
 pinMode (ledpin, OUTPUT);
 }
}

void loop() {

 for (ledpin = 2; ledpin < 11; ledpin++)
{
  digitalWrite (ledpin, HIGH);
  delay (50);
  digitalWrite (ledpin, LOW);
}

for (ledpin = 9; ledpin > 1; ledpin--)
{
  digitalWrite (ledpin, HIGH);
  delay (50);
  digitalWrite (ledpin, LOW);
}
}
```

Controllo di un led con un pulsante

Titolo:

Controllo di un led con un bottone

Obiettivo:

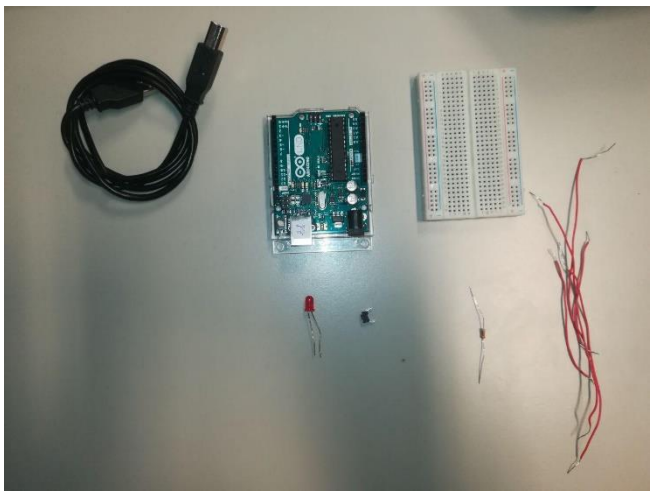
L'obiettivo di tale esperienza è stato quello di apprendere la progettazione, con l'ausilio di una scheda Arduino, di un semplice programma che permettesse di controllare un led con un bottone.

Materiali e strumenti utilizzati :

Per la realizzazione e progettazione in oggetto, sono stati usati i seguenti materiali:

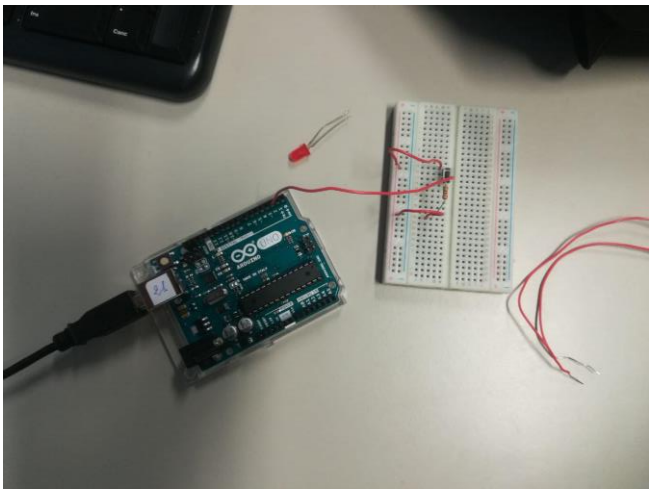
1. *Un Pc desktop (utilizzato per varie compilazioni e test);*
2. *IDE Arduino (applicazione multi-piattaforma in Java, installato sul Pc desktop);*
3. *Scheda Arduino Uno;*
4. *Breadboard (basetta sperimentale);*
5. *Cavo USB ti tipo A/B M-M (Il connettore A è usato per collegarlo ad un PC, mentre l'altro connettore B per collegarlo alla scheda di Arduino);*
6. *Cavi ponticello per effettuare collegamenti e/o ponti tra BreadBoard e scheda Arduino;*
7. *Un numero di led necessari pari a 1;*
8. *Un numero di resistenze necessarie pari a 1 di 120Ω;*
9. *Un bottone;*
10. *Pinze;*
11. *Spella-cavi.*

***Immagini di materiali, fase di progettazione e test*:**



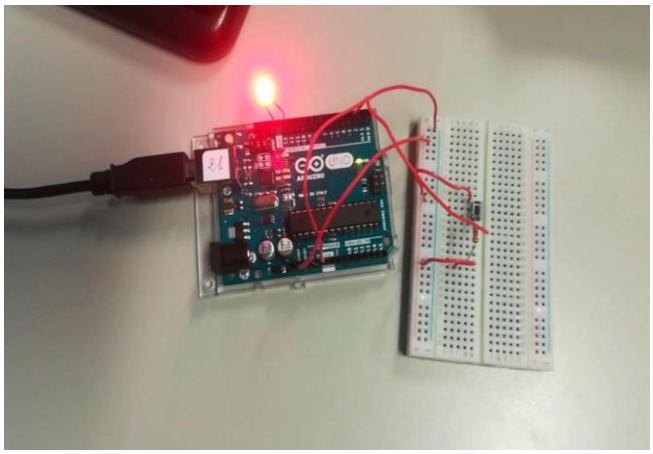
$_{1}$Fase di preparazione dei componenti:

Preparazione dei componenti necessari per lo sviluppo dell'esperienza in laboratorio



$_{2}$Fase di progettazione:

Tutti i componenti vengono assemblati secondo lo schema tecnico fornito dal docente



$_{3}$Fase di test:

Il codice viene caricato sulla scheda per la verifica di eventuali errori commessi nella scrittura del codice e nell'assemblaggio dei componenti

Parte Sperimentale :

Questa sezione può essere divisa in semplici fasi:

1. Fase di progettazione

Collegamenti fisici:

Nella parte iniziale, è necessario riprodurre il progetto cartaceo fornito sulla basetta sperimentale e sulla scheda arduino uno, effettuando I vari collegamenti e inserimenti dei componenti necessari in modo tale da ottenere la stessa "pianta".

Stesura del Codice:

Sono state inizialmente dichiarate in globale le variabili del bottone, del led, dello stato attuale, del precedente stato(inizializzata con LOW) ed una variabile dove verrà salvato il valore letto in entrata dal bottone. Sono state in seguito dichiarate le due variabili del bottone e del led rispettivamente con input ed output. Il codice da ciclare inizia con la lettura del bottone (se viene premuto il bottone abbiamo in entrata 1. In caso contrario, abbiamo 0) e sarà poi assegnato alla variabile che salva lo stato attuale del componente. Abbiamo poi una serie di controlli: il primo controllo dopo aver confrontato lo stato del led attuale con quello precedente(se il valore attuale e quello precedente saranno rispettivamente HIGH E LOW) esegue con un ritardo di 5s il secondo controllo che incrementa o diminuisce il valore dello stato. Dopo aver eseguito fuori il controllo l'assegnazione dello stato attuale alla variabile dello stato precedente, abbiamo il terzo ed ultimo controllo che in baso allo stato attuale, accenderà o spegnerà in modo permanente il led fino a che non verrà ripremuto nuovamente il bottone o verrà scollegato il cavo USB che collega la scheda arduino con l'alimentazione del pc desktop.

2. Fase di test

In questa fase, dopo aver effettuato il collegamento tramite cavo USB tra Pc desktop e scheda Arduino Uno, abbiamo verificato e compilato, attraverso l'opzione prevista dal software "Verifica/Compila" che il codice trascritto non avesse errori.

Per ultimo, verificato se il codice fosse privo di errori, abbiamo caricato sulla scheda ARDUINO UNO, il codice precedentemente Verificato e compilato.

CODICE

```
int but=2;
int led=13;
int var= LOW;
int stato=0; // 0 corrisponde a SPENTO; 1 corrisponde ad ACCESO
int  vecchio_val= LOW;

void setup() {

     pinMode (led, OUTPUT);
     pinMode (but, INPUT);

}

void loop() {

  var= digitalRead (but);

  if (var == HIGH && vecchio_val==LOW)
    {
    delay (5);
    stato= 1- stato;
    if (stato==1)
    stato=0;
    else
    stato=1; //sia se metto stato= 1-stato e sia se faccio l'if è la stessa cosa
   }

   vecchio_val =var;

if (stato == 0)
    digitalWrite (led, LOW);
else
    digitalWrite (led, HIGH);
}
```

Elaborazione dati :

Nella sottostante tabella di verità sono stati riportati i risultati di tale progetto:

	OutPut
Stato Bottone	**Led**
1 (High)	**High**
0 (LOW)	Low

Nel caso in cui il bottone verrà tenuto premuto e non verrà pertanto rilasciato, il led manterrà lo stato che ha già ottenuto in input in uno dei due casi precedenti e non verrà pertanto ciclato fino ad ottenere così un led che lampeggia ad intermittenza.

Utilizzo di un monitor seriale

Titolo:

Gestione del monitor seriale

Obiettivo:

L'obiettivo di questa prima esperienza con il monitor seriale è destinata a comprendere il suo funzionamento. Per fare ciò, è stato progettato in classe un semplice programma che accendesse sulla scheda Arduino ,uno o più led dopo aver inserito una lettera da tastiera.

Materiali e strumenti utilizzati :

Per la realizzazione e progettazione in oggetto, sono stati usati i seguenti materiali:

1. *Un Pc desktop (utilizzato per varie compilazioni e test);*
2. *IDE Arduino (applicazione multi-piattaforma in Java, installato sul Pc desktop);*
3. *Scheda Arduino Uno;*
4. *Breadboard (basetta sperimentale);*
5. *Cavo USB ti tipo A/B M-M (Il connettore A è usato per collegarlo ad un PC, mentre l'altro connettore B per collegarlo alla scheda di Arduino)*
6. *Cavi ponticello per effettuare collegamenti e/o ponti tra BreadBoard e scheda Arduino;*
7. *Un numero di led necessari pari a 3(Rossi);*
8. *Un numero di resistenze necessarie pari a 3 di 120Ω ;*
9. *Pinze;*
10. *Spella-cavi.*

Parte Sperimentale :

Questa sezione può essere divisa in semplici fasi:

1. *Fase di progettazione*

   Collegamenti fisici:

   Nella parte iniziale, è necessario riprodurre il progetto proiettato a muro sulla basetta sperimentale e sulla scheda arduino uno, effettuando I vari collegamenti fino ad ottenere la "pianta" prestabilita.

   Stesura del Codice:

   Sono state inizialmente dichiarate in globale le variabili input, I pin dei led (l1, l2, l3) edunavariabile "i" che avrà la funzione di contatorenelcodice. Nella procedura void setup, dopo aver dichiarato pin dei 3 led come OUTPUT, sono state inseritealcunedellenuoveistruzioni per questoparticolareprogramma: la prima è Serial.begin(9600) cherappresenta la velocià di trasmissionedatiche la schedaarduinodovràacquisire; la secondainvece è Serial.println("TESTO"). In questocaso, sono state inserite 2 funzioni di Serial.println per dire all'utente/programmatorecheaveva 2 scelte di eseguireilprogramma. Al posto di TESTO all'internodellevirgolettepostodentroalleparentesi, sono state scritteinformazionichedescrivonoquelloche la schedaavrebbeeseguitonelcaso in cui quelpulsante da tastiera fosse statoinviatonel monitor seriale. Si è passati poi allafase di loop; successivamentesonostatiusatideicontrolli IF per stabilireilregolaresvolgimento del programma in base allaselezionedell'utente. Il primo ciclocontrolla se sonostatiinseritideivaloriall'interno del monitor seriale. Se sonopresentideicaretteri, serial.avaiable() saràmaggiore di 0. Verrannoeseguiteinfine le istruzionicontenutenellacondizionevera, ovveroaltri 2 cicli. Dopo aver assegnatoallavariabile input ilvalore di Serial.read() (ovvero un codice ASCII), siparte con il primo IF annidato a quelloprecedente. In questacondizione, I led lampeggeranno 2 volte, se la letterainseritadall'utente è uguale ad "A", grazie all' ausilio di un ciclo for. Nel secondo if contenutonel primo IF di partenzaall'inizio del loop, è presente la secondaopzione. I led lampeggeranno 3 volte se l'utenteinserisce da tastiera la lettera "B" sempregrazieall'ausilio di un ciclo for. L'usodeicicli for in questoprogrammasonoindispensabili per limitare le numeroserighe di codiceinutili.

2. *Fase di test*

   In questa fase, dopo aver effettuato il collegamento tramite cavo USB tra Pc desktop e scheda Arduino Uno, abbiamo verificato e compilato, attraverso l'opzione prevista dal software "Verifica/Compila", che il codice trascritto non avesse errori.

   Per ultimo, verificato se il codice fosse privo di errori, abbiamo caricato sulla scheda ARDUINO UNO, il codice precedentemente Verificato e compilato.

CODICE

```
int input;
int l1 = 2;
int l2 = 4;
int l3 = 6;
int i = 0;
void setup() {

pinMode(l1,OUTPUT);
pinMode(l2,OUTPUT);
pinMode(l3,OUTPUT);
Serial.begin(9600);
Serial.println ("Premi A per far lampeggiare due volte");
Serial.println ("Premi B per far lampeggiare tre volte");
}

void loop() {

if (Serial.available()>0){
   input = Serial.read();
   if (input == 'A')
   for (i=0; i < 2; i++){
     digitalWrite (l1, HIGH);
     digitalWrite (l2, HIGH);
     digitalWrite (l3, HIGH);
     delay (500);
     digitalWrite (l1, LOW);
     digitalWrite (l2, LOW);
     digitalWrite (l3, LOW);
     delay (500);
   }

     if (input == 'B')
   for (i=0; i <3 ; i++){
     digitalWrite (l1, HIGH);
     digitalWrite (l2, HIGH);
     digitalWrite (l3, HIGH);
     delay (500);
     digitalWrite (l1, LOW);
     digitalWrite (l2, LOW);
     digitalWrite (l3, LOW);
     delay (500);
   }


}
}
```

Elaborazione dati :

Nella sottostante tabella sono stati riportati i risultati dell'esperienza in base al codice scritto dal programmatore.

	Gruppo di led		
SCELTA	**LED 1**	**LED 2**	**LED 3**
Nel caso in cui dovesse essere inserita la lettera "A", verrà eseguito il primo ciclo del void loop per 2			
A(prima parte ciclo 300s)	High	High	High
A(secondaparte ciclo 300s)	High	High	High
In caso contrario, se dovesse essere inserita la lettera "B", verrà eseguito il secondo ciclo del void loop per 3 volte			
B(prima parte ciclo 300s)	High	High	High
B(seconda parte ciclo 300s)	High	High	High

Legenda:
-A (I led lampeggiano 2 volte);
-B (I led lampeggiano 3 volte).

Conclusione

Il progetto FCA ci ha permesso di affrontare la progettazione con la scheda Arduino che ha un utilizzo pratico nel mondo del lavoro. Prima di poter sviluppare nel concreto quanto richiesto abbiamo dovuto studiare i concetti teorici relativi alla programmazione con Arduino ed abbiamo dovuto affrontare in laboratorio la creazione di circuiti specifici tramite dei progetti che ci guidassero nell'applicazione pratica di tali concetti.

In particolare le funzioni necessarie alla progettazione con Arduino viste a lezione sono state le seguenti:

pinMode()

Sintassi,

pinMode(pin,mode) ; // pin, numero del pin di cui si vuole impostare la modalità ingresso o uscita

//mode, sempre INPUT o OUTPUT

// I pin analogici possono essere usati come pin digitali purché ci si riferisca con i nomi A0, A1, ..., A5 Esempio,

intledPin

ledPin = 10; // led connesso al pin digitale 10

voidsetup( ){ pinMode(ledPin, OUTPUT)}; // imposta il pin digitale come uscita (output)

digitalWrite()

Scrive un valore HIGH o LOW su un pin impostato come digitale. Se il pin è stato impostato come OUTPUT con il pinMode(), la sua tensione sarà impostata al corrispondente valore di 5V per HIGH e 0V per LOW. Se il pin è impostato come INPUT tramite il pinMode(), scrivendo un valore HIGH con digitalWrite() si abiliterà un resistore interno di pull-up da 20K. Scrivendo basso sarà disabilitato il pull-up. Il resistore di pull-up è sufficiente per far illuminare un led debolmente, per cui sembra che il diodo lavori anche se debolmente (questa è una probabile causa). Il rimedio è di impostare il pin su un'uscita con il metodo pinMode Attenzione: il pin 13 è più difficile da utilizzare come input digitale rispetto agli altri pin poiché ha già collegati un diodo led e un resistore sulla board. Se si abilita il resistore di pull-up esso viene portato a 1,7V invece dei 5V attesi poiché si deve tener conto della serie di questi due componenti e questo significa che restituisce sempre LOW (Arduino legge sempre 0). Per ovviare a questo inconveniente occorrerebbe inserire esternamente un resistore di pull-down.

Sintassi,

digitalWrite(pin,valore);

// pin, è il pin di cui si vuole impostare

// il valore,valore, HIGH o LOW

// I pin analogici possono essere usati come pin digitali purché ci si riferisca con i nomi A0, A1, ..., A5

Esempio,

```
intledPin=13;                 // led connesso al pin digitale 13
voidsetup( )
{
pinMode(ledPin, OUTPUT);      // imposta il pin digitale come output
}
voidloop( )
 {
digitalWrite(ledPin, HIGH);  // accende il LED
delay(1000);                 // attende per 1 secondo
digitalWrite(ledPin, LOW);   // spegne il LED
delay(1000);                 // attende 1 secondo
}
```

digitalRead()

Legge il valore da un pin digitale specifico (se il pin non è connesso per un qualsiasi motivo digitalRead() può restituire sia HIGH che LOW in modo del tutto casuale).

Sintassi,

// pin, è il numero intero del pin di cui si vuole leggere il valore

digitalRead(pin); alto o basso.

// I pin analogici possono essere usati come pin digitali purché ci si riferisca con i nomi A0, A1, ..., A5

Esempio,

```
intledPin = 13;                 // LED connesso al pin digitale 13
intinPin = 7;             // pulsante connesso al pin digitale 7
int val = 0;              // variabile che memorizza il valore letto
voidsetup( )
{
pinMode(ledPin, OUTPUT);   // imposta il pin digitale 13 come uscita
pinMode(inPin, INPUT);       // imposta il pin digitale 7 come ingresso
}
voidloop( )
{
 val = digitalRead(inPin); // legge il pin di ingresso e lo memorizza in val
digitalWrite(ledPin, val); // imposta il LED a seconda dell'azione svolta sul pulsante
}
```

Comunicazione seriale (comandi Serial)

Usata per la comunicazione tra Arduino e il computer o altri dispositivi. Tutte le schede Arduino hanno almeno una porta seriale (Serial, conosciuta come UART o USART). Essa comunica con i pin 0 (RX) e 1 (TX) alla stessa maniera con cui comunica con il computer via USB (la porta usb è collegata a questi pin vedi schema). Se si stanno utilizzando le funzioni legate a Serial, non si possono usare i pin 0 e 1 per gli I/O digitali. Si può utilizzare l'ambiente monitor seriale definito nell'ambiente di sviluppo di Arduino per comunicare con la scheda Arduino, cliccando sul bottone Serial monitor (in menù opzione Strumenti) della toolbar dello sketch e selezionando la stessa velocità in baud (bit al secondo) utilizzato nella chiamata di begin(). L'Arduino Mega ha tre porte seriali aggiuntive: Serial1 sul pin 19 (RX) e 18 (TX), Serial2 sul pin 17 (RX) e 16 (TX), Serial3 sul pin 15 (RX) e 14 (TX). Per usare questi pin per comunicare con il personal computer, sarà necessario aggiungere un adattatore USB-Seriale, poiché questi non sono connessi con l'adattatore USB-Seriale del Mega. Per utilizzare questi per comunicare con dispositivi esterni seriali in logica TTL, connettere il pin TX con il pin RX del dispositivo, e la massa del Mega alla terra del dispositivo. Non connettere questi pin direttamente con la porta seriale RS232 del computer; essa opera a +/- 12V e può danneggiare la scheda Arduino.

begin()

Imposta la velocità di trasmissione dati in bit al secondo (baud) per la trasmissione seriale. Per comunicare con il computer ci sono diverse velocità ammissibili: 300, 1200, 2400, 4800, 9600, 14400, 57600, 115200. Si può comunque specificare altri valori, per componenti che richiedono particolari velocità.

Sintassi,

Serial.begin(velocità);

Solo per Arduino Mega

Serial1.begin(velocità);

Serial2.begin(velocità);

Serial3.begin(velocità);

il parametro velocità rappresenta la velocità in bit per secondi (baud) ed è dichiarata come tipo long. La funzione non restituisce alcun valore

Esempio,

```
voidsetup( )
 {
Serial.begin(9600);          // apre la porta seriale e imposta la velocità a 9600 bps
}
voidloop( ) { }
```

end( )

Disabilita la comunicazione seriale, permettendo ai pin RX e TX di essere utilizzati come I/O generali. Per riabilitare la comunicazione seriale basta richiamare la funzione Serial.begin().

Sintassi,

Serial.end( );

available( )

Restiuisce il numero di byte disponibili per leggere dalla porta seriale.

Sintassi,

Serial.available( )

Solo per Arduino Mega,

Serial1.available( );

Serial2.available( );

Serial3.available( );

La funzione non richiede alcun parametro e restituisce un valore di tipo byte necessari per la lettura

read( )

Legge i dati in entrata dalla comunicazione seriale.

Sintassi,

Serial.read( )

Esempio

```
intbyteInArrivo = 0;                    // per iniziare la trasmissione seriale
voidsetup( )
{
        Serial.begin(9600);   //apertura porta seriale, impostazione velocità di trasmissione dati a
                              9600bps
}
voidloop( )
 {
        if (Serial.available( )> 0)
         {
        byteInArrivo =Serial.read();          // leggi i byte che arrivano Serial.print("Ricevuto: ");
        Serial.println(byteInArrivo, DEC);    // scrive i dati ricevuti
        }
}
```

flush( )

Svuota il buffer dei dati seriali in entrata. Cioè, ogni chiamata di Serial.read() o Serial.available() restituirà solo i dati ricevuti dopo tutte le più recenti chiamate di Serial.flush().

Sintassi,

Serial.flush( )

print( )

Questa funzione stampa sulla porta seriale un testo ASCII leggibile dall'uomo. Questo comando può assumere diverse formati: i numeri sono stampati usando un carattere ASCII per ogni cifra, i float sono scritti allo stesso modo e per default a due posti decimali, i byte vengono inviati come caratteri singoli mentre i caratteri e le stringhe sono inviati come sono.

Sintassi,

Serial.print(valore)

Serial.print(valore,formato) valore indica il valore da scrivere di qualsiasi tipo, mentre formato indica la base del sistema numerico (per i dati di tipo integer) o il numero di posti decimali per i tipi floating point.

La funzione non restituisce alcun valore.

Esempi,

Serial.print(78); // dà come risultato "78"

Serial.print(1.23456); // dà come risultato "1.23"

Serial.print(byte(78)); // dà come risultato "N" (il valore ASCII di N è 78)

Serial.print('N'); // dà come risultato "N"

Serial.print("Ciao mondo."); // dà come risultato "Ciao mondo."

Un secondo parametro opzionale specifica la base, il formato in uso: i valori permessi sono BYTE, BIN (binario e base 2), OCT(ottale o base 8), DEC (decimale o base 10), HEX (esadecimale o base 16).Per i numeri a virgola mobile, questo parametro specifica il numero di posti decimali da utilizzare.

Esempi,

Serial.print(78, BYTE); // stampa "N"

Serial.print(78, BIN); // stampa "1001110"

Serial.print(78, OCT); // stampa "116"

Serial.print(78, DEC); // stampa "78"

Serial.print(78, HEX); // stampa "4E"

Serial.println(1.23456, 0); // stampa "1"

Serial.println(1.23456, 2); // stampa "1.23"

Serial.println(1.23456, 4); // stampa "1.2346"

Nota: l'ultimo carattere che deve essere scritto è trasmesso sulla porta seriale dopo che Serial.print() è terminato. Questo potrebbe essere un problema in casi particolari. 39

println( )

Scrive i dati sulla porta seriale come testo leggibile dall'uomo in codice ASCII seguito da un carattere di ritorno a capo (ASCII 13 oppure '\r') e un carattere di nuova linea (ASCII 10, oppure '\n'). Questo comando assume la stessa forma di Serial.print().

Sintassi,

Serial.println(valore);

Serial.println(valore, formato);

Il parametro valore indica il valore da scrivere di qualsiasi tipo, mentre formato indica la base del sistema numerico (per i dati di tipo integer) o il numero di posti decimali per i tipi floating point. La funzione non restituisce alcun valore.

write( )

Scrive i dati binari sulla porta seriale. Questi dati sono inviati come byte o serie di byte; per inviare i caratteri rappresentanti le cifre di un numero usare invece la funzione print().

Sintassi,

Serial.write(valore); // il parametro valore è il valore da trasmettere come singolo byte

Serial.write(stringa); // il parametro stringa è una stringa da spedire come serie di byte

Arduino Mega supporta anche,

Serial.write(buf,len); // il parametro buf è un array da spedire come serie di byte
// il parametro len rappresenta la lunghezza del buf Serial1, Serial2, Serial3 al posto di Serial.