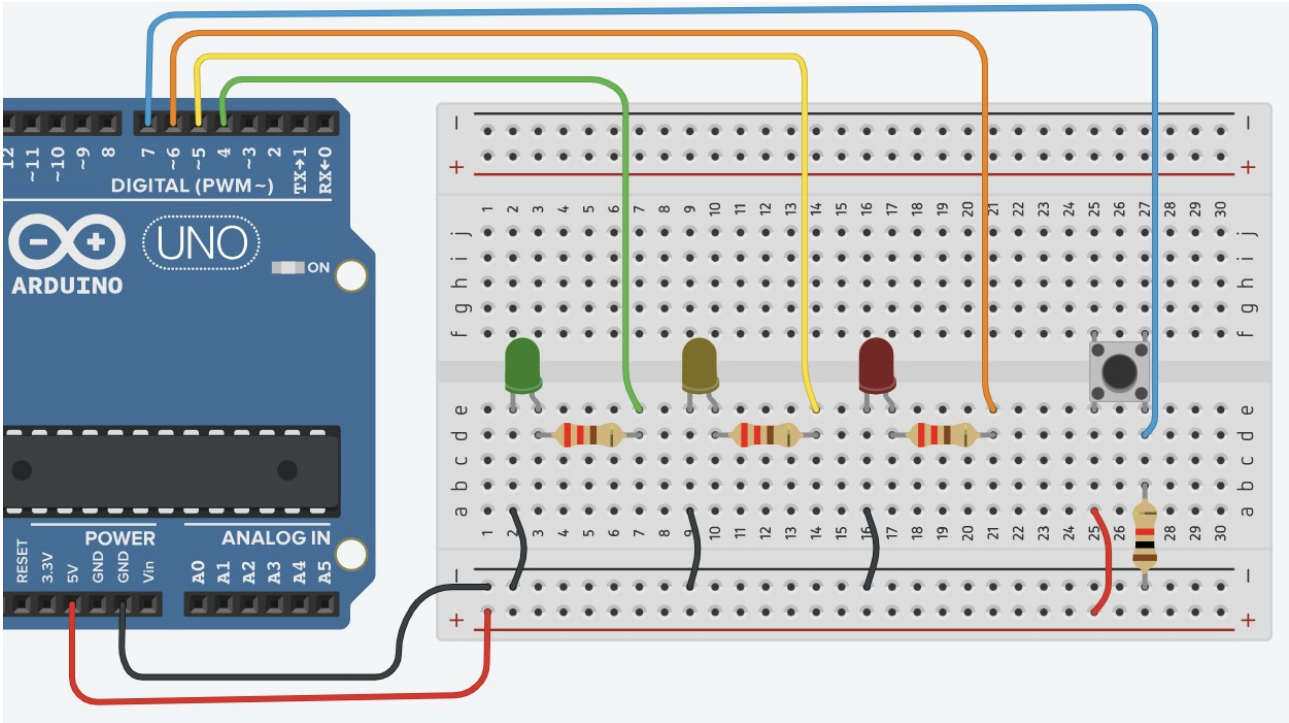
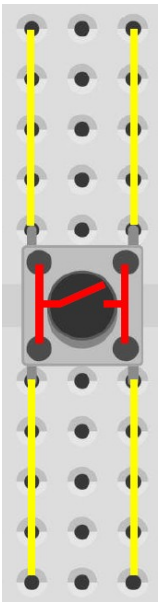
Con la seguente esperienza proviamo a realizzare la simulazione di un semaforo utilizzando una scheda ARDUINO UNO.

Dobbiamo innanzitutto definire i dispositivi che la scheda dovrà controllare che sono:

- le 3 luci del semaforo,
- un pulsante per chiamare l'attraversamento pedonale.



Nel circuito in alto il terminale 4 è collegato al LED VERDE, il terminale 5 al LED GIALLO, il terminale 6 al LED ROSSO ed il terminale 7 al PULSANTE.



Il pulsante è collegato con una resistenza di PULL-DOWN.

Nella parte iniziale del programma potremo associare i nomi dei dispositivi collegati al numero dei terminali a cui sono collegati con la direttiva *#define*.

```
#define LED_VERDE 4
#define LED_GIALLO 5
#define LED_ROSSO 6
#define PULSANTE 7
```

Successivamente possiamo dichiarare le variabili che ci serviranno nel programma.
Possiamo ipotizzare la necessità di una variabile da utilizzare come **contatore** nella gestione del ritardo, ed una variabile per **memorizzare** se è stato premuto il pulsante per chiedere l'attraversamento.
Alla variabile memoria associamo inizialmente il valore 0.

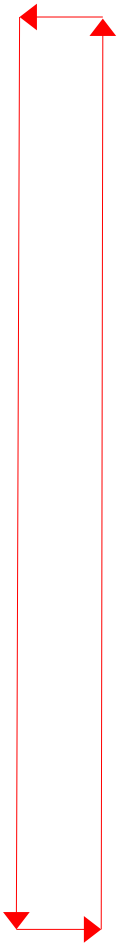
```
int cont;

int memoria=0;
```

Dopo la dichiarazione delle variabili, all'interno del SETUP, dobbiamo indicare quali terminali sono indicati come uscita e quali come ingresso, utilizzando la funzione pinMode

```
void setup() {
  pinMode(LED_VERDE,OUTPUT);
  pinMode(LED_GIALLO,OUTPUT);
  pinMode(LED_ROSSO,OUTPUT);
  pinMode(PULSANTE,INPUT);
}
```

Scriviamo ora all'interno del ciclo loop un programma per accendere in sequenza i 3 led.

```
void loop() {
  digitalWrite(VERDE,HIGH);
  digitalWrite(GIALLO,LOW);
  digitalWrite(ROSSO,LOW);

  delay(5000);

  digitalWrite(VERDE,LOW);
  digitalWrite(GIALLO,HIGH);
  digitalWrite(ROSSO,LOW);

  delay(5000);

  digitalWrite(VERDE,LOW);
  digitalWrite(GIALLO,LOW);
  digitalWrite(ROSSO,HIGH);

  delay(5000);
}
```



Inizialmente viene spento il LED_GIALLO ed il LED_ROSSO e viene acceso il **LED_VERDE**.

Il LED_VERDE rimarrà acceso per **5 secondi**.

Successivamente viene spento il LED_VERDE ed il LED_ROSSO e viene acceso il **LED_GIALLO**.

Il LED_GIALLO rimarrà acceso per **5 secondi**.

Successivamente viene spento il LED_VERDE ed il LED_GIALLO e viene acceso il **LED_ROSSO**.

Il LED_GIALLO rimarrà acceso per ***5 secondi***.

Il programma viene eseguito ciclicamente secondo il flusso indicato dalla freccia rossa a destra.

In questo caso però manca il controllo del pulsante.
Per leggere lo stato del pulsante l'istruzione per leggere lo stato del pulsante è la seguente:

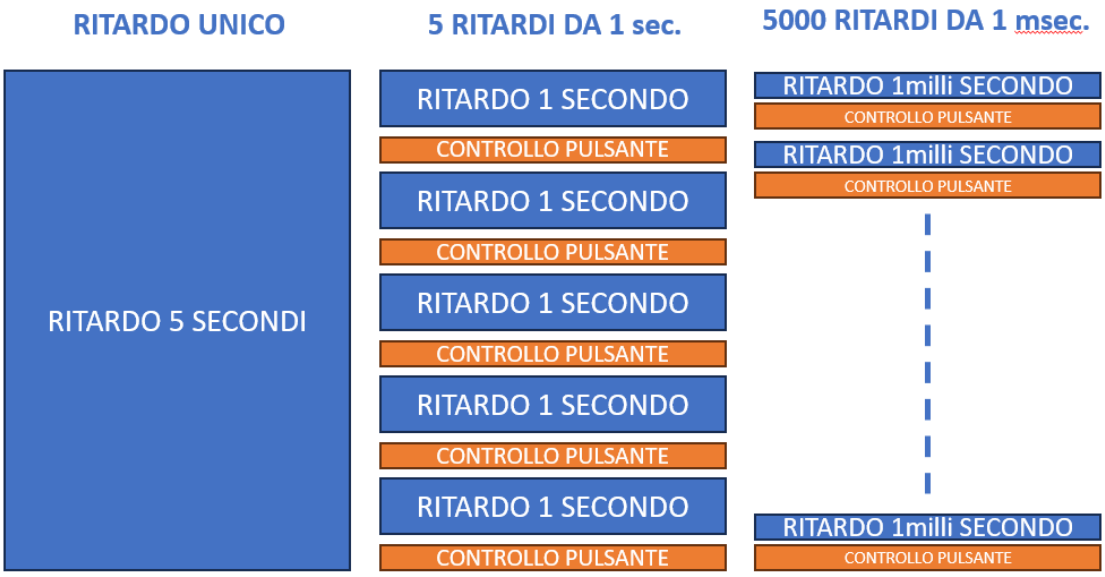
```
digitalRead(PULSANTE);
```

Bisogna però comprendere dove inserire questa istruzione all'interno del ciclo sopra scritto.

Le istruzioni per accendere e spegnere i LED digitalWrite hanno una durata di qualche microsecondo, invece il delay ha la durata di 5 secondi (nella realtà durerà ancora di più).
Per questo è molto probabile anche se inseriamo la lettura del pulsante in qualsiasi punto del programma esso venga letto all'interno del delay .

In questo caso il processore non si accorgerà della pressione del pulsante.

Una volta capito il problema proviamo a proporre una possibile soluzione.
Visto che il problema è il fatto che durante il delay non viene testato il pulsante, allora si potrebbe spezzettare il ritardo in piccoli ritardi tra i quali mettere il test del pulsante.



Pertanto al posto del delay(5000) inserito dopo l'accensione del LED_VERDE e del LED_GIALLO, bisogna mettere un ciclo che esegue per 5000 volte un ritardo di 1msec.
Tra ogni piccolo ritardo e l'altro occorre inserire la lettura del pulsante.

Prima del ciclo while bisogna dare il valore alla variabile cont.

```
cont=5000;
```

Bisogna poi eseguire il ciclo solo se la variabile è maggiore di 0.

```
while(cont>0){
```

Dentro il ciclo occorre inserire il ritardo di 1msec. ed il decremento della variabile.

```
    delay(1);
    cont--;
```

Dopo il ritardo possiamo controllare il pulsante e se esso vale 1 (pulsante premuto) allora mettiamo il valore della variabile memoria ad 1.

```
    if(digitalRead(PULSANTE)==1){
        memoria=1;
    }
}
```

E' importante fare attenzione alle parentesi, ogni volta che viene aperta una parentesi deve essere poi chiusa.

```
}
```

Un'altra cosa importante è **l'indentazione** cioè l'allineamento delle righe di programma.
Si può vedere infatti che le righe all'interno del ciclo while sono allineate più a destra, come anche la riga dentro all'if.

Pertanto in qualsiasi momento verrà premuto il pulsante la scheda se ne accorgerà e la variabile memoria andrà ad 1.

A questo punto il ritardo dopo il LED_ROSSO dipenderà dalla variabile memoria.

Andremo perciò a sostituire il ritardo fisso di 5 secondi inserito prima con un if che controlla il valore della variabile memoria.
Se memoria vale 0 il ritardo sarà di 5 secondi, in caso contrario di 10 secondi.
Al termine memoria viene reinizializzata al valore 0.
Anche in questo caso è importante **indentare** bene le righe di codice.

```
if(memoria==0){
   delay(5000);
}
else{
   delay(10000);
}
memoria=0;
```

Di seguito il programma intero messo in due colonne per poterlo visualizzare in un unica pagina.

```
#define LED_VERDE 4
#define LED_GIALLO 5
#define LED_ROSSO 6
#define PULSANTE 7

int cont;
int memoria=0;

void setup() {
  pinMode(LED_VERDE,OUTPUT);
  pinMode(LED_GIALLO,OUTPUT);
  pinMode(LED_ROSSO,OUTPUT);
  pinMode(PULSANTE,INPUT);
}

void loop() {
  digitalWrite(VERDE,HIGH);
  digitalWrite(GIALLO,LOW);
  digitalWrite(ROSSO,LOW);
  cont=5000;
  while(cont>0){
     delay(1);
     cont--;
     if(digitalRead(PULSANTE)==1){
        memoria=1;
     }
  }
  digitalWrite(VERDE,LOW);
  digitalWrite(GIALLO,HIGH);
  digitalWrite(ROSSO,LOW);
  cont=5000;
  while(cont>0){
     delay(1);
     cont--;
     if(digitalRead(PULSANTE)==1){
        memoria=1;
     }
  }   digitalWrite(VERDE,LOW);
  digitalWrite(GIALLO,LOW);
  digitalWrite(ROSSO,HIGH);
  if(memoria==0){
    delay(5000);
  }
  else{
    delay(10000);
  }
  memoria=0;
}
```

Possiamo inoltre osservare che nel programma c'è la parte di codice relativa al ciclo while che si ripete. Un'ottimizzazione potrebbe essere quella di inserire queste righe di codice all'interno di una **funzione**.
La funzione è una parte di codice a cui viene dato un nome e che può essere richiamata all'interno del programma più volte.

In questo caso la funzione è stata chiamata **ritardo**, ed è stata dichiarata prima del loop.

```
#define LED_VERDE 4
#define LED_GIALLO 5
#define LED_ROSSO 6
#define PULSANTE 7

int cont;
int memoria=0;

void setup() {
  pinMode(LED_VERDE,OUTPUT);
  pinMode(LED_GIALLO,OUTPUT);
  pinMode(LED_ROSSO,OUTPUT);
  pinMode(PULSANTE,INPUT);
}

void ritardo(){
  cont=5000;
  while(cont>0){
     delay(1);
     cont--;
     if(digitalRead(PULSANTE)==1){
        memoria=1;
     }
  }
}

void loop() {
  digitalWrite(VERDE,HIGH);
  digitalWrite(GIALLO,LOW);
  digitalWrite(ROSSO,LOW);
  ritardo();
  digitalWrite(VERDE,LOW);
  digitalWrite(GIALLO,HIGH);
  digitalWrite(ROSSO,LOW);
  ritardo();
  digitalWrite(VERDE,LOW);
  digitalWrite(GIALLO,LOW);
  digitalWrite(ROSSO,HIGH);
  if(memoria==0){
    delay(5000);
  }
  else{
    delay(10000);
  }
  memoria=0;
}
```

All'interno del ciclo loop, il codice scritto nella funzione, può essere eseguito digitando semplicemente il nome della funzione.