Riassumiamo i tipi di variabili che possiamo trovare:

Tipo	Lunghezza	Descrizione
void		(vuoto) è un tipo speciale che possiamo usare solo come prefisso per le funzioni ed indica che queste non restituiscono nulla
boolean	1 byte	(boleano) valore logico può valere solamente 0 (zero) o 1 (uno) ovvero vero o falso
char	1 byte	(carattere) è usato per memorizzare il valore numerico di un carattere, Se viene assegnato in formato letterale si usa il singolo apice come questo: 'A'. Per caratteri multipli (string) si usa il doppio apice: “ABC”. I caratteri vengono memorizzati in formato numerico secondo la tabella ASCII. Nel tipo char possono essere memorizzati numeri da -128 a +127
unsigned char	1 byte	(carattere senza segno) tipo carattere ma senza segno memorizza numeri da 0 a 255 . Sconsigliato l'utilizzo, usare byte
byte	1 byte	(byte) per memorizzare numeri senza segno a 8 bit, da 0 a 255
int	2 bytes	(intero) il tipo principale di variabile per memorizzare i numeri su questi microcontrollori memorizza valori a 16 bit da -32.768 a +32.767 (da -2^15 a +2^15-1)
unsigned int	2 bytes	(intero senza segno) memorizza numeri da 0 a 65.535 ((2^16)-1)
word	2 bytes	(lett. Parola) uguale a unsigned int
long	4 bytes	(intero lungo) memorizza numeri a 32 bit da -2.147.483.648 a +2.147.483.647
unsigned long	4 bytes	(intero lungo senza segno) per numeri da 0 a 4.294.967.295
short	2 bytes	(corto) identico ad int
float	4 bytes	(a virgola mobile) memorizza numeri con virgola da -3,4028235^38 a +3,4028235^38 la precisione è a 6 cifre (compresa anche la parte a sinistra della virgola). Attenzione nell'uso in quanto 6,0 / 3,0 può non essere uguale a 2,0. I calcoli eseguiti con i float sono molto più lenti e per applicazioni dove i tempi sono critici andrebbero evitati.
double	4 bytes	(doppia precisione) sulle schede con microcontroller ATMEGA sono esattamente come i float senza nessun guadagno in precisione
String		(Stringa) Attenzione!! con la S maiuscola, è un oggetto stringa e può contenere caratteri alfanumerici. La sua lunghezza (e occupazione di memoria) è variabile e dipende dal numero di caratteri memorizzati
array		(serie o collezione) un array è una collezione di variabili accessibili tramite un indice numerico

Facciamo ora un passo avanti, creiamo due funzioni cioè due “sottoprogrammi” che eseguono dei compiti specifici che possiamo richiamare ed eseguire tutte le volte che ci servono, separiamo quindi dal ciclo loop i lampeggi necessari per i caratteri S e O in questo modo:

```
/* Sos codice morse

const int pinled = 13;          // pin collegato al led
const int punto = 200;          // tempo led acceso per punto
const int linea = 600;          // tempo led acceso per linea
const int pausaBreve = 300;     // tempo led spento
const int pausaLunga = 800;     // tempo led spento tra 2 lettere

//definisco la procedura per generare la lettera S
void blinkS(){
  digitalWrite(pinled,HIGH); // accendo il led mettendo high l'uscita 13
  delay(punto);              //pausa 200 millisec.
  digitalWrite(pinled,LOW);  // spengo il led mettendo low l'uscita 13
  delay(pausaBreve);         //pausa 300 millisec.
  digitalWrite(pinled,HIGH); // accendo il led mettendo high l'uscita 13
  delay(punto);              //pausa 200 millisec.
  digitalWrite(pinled,LOW);  // spengo il led mettendo low l'uscita 13
  delay(pausaBreve);         //pausa  200 millisec.
  digitalWrite(pinled,HIGH); // accendo il led mettendo high l'uscita 13
  delay(punto);              //pausa 200 millisec.
  digitalWrite(pinled,LOW);  // spengo il led mettendo low l'uscita 13
  delay(pausaLunga);        //pausa 800 millisec.
}

//definisco la procedura per generare la lettera O
void blinkO(){
  digitalWrite(pinled,HIGH); // accendo il led mettendo high l'uscita 13
  delay(linea);              //pausa 600 millisec.
  digitalWrite(pinled,LOW);  // spengo il led mettendo low l'uscita 13
  delay(pausaBreve);         //pausa 300 millisec.
  digitalWrite(pinled,HIGH); // accendo il led mettendo high l'uscita 13
  delay(linea);              //pausa 600 millisec.
  digitalWrite(pinled,LOW);  // spengo il led mettendo low l'uscita 13
  delay(pausaBreve);         //pausa 300 millisec.
  digitalWrite(pinled,HIGH); // accendo il led mettendo high l'uscita 13
  delay(linea);              //pausa 600 millisec.
  digitalWrite(pinled,LOW);  // spengo il led mettendo low l'uscita 13
  delay(pausaLunga);         //pausa 800 millisec.
}
void setup() {
  pinMode(pinled,OUTPUT); // definisco il pin 13 come uscita
}

void loop() {
  //lettera S
 blinkS();
  //lettera O
 blinkO();
  //lettera S
 blinkS();
}
```

Qui ho creato due nuove funzioni chiamate ***blinkS*** e ***blinkO,*** il nome ovviamente l'ho scelto in modo che sia semplice ricordare quello che fanno, all'interno della funzione ***blinkS*** ci sono le istruzioni che fanno lampeggiare il led per comporre la lettera S e lo stesso la funzione ***blinkO*** per la lettera O. Queste funzioni come si vede sono esterne sia alla sezione ***setup*** che alla ***loop***, sono due pezzetti di programma che vengono eseguiti solo su richiesta, tutte le volte che vengono invocati. Si invocano scrivendone il nome e le parentesi tonde ***nomeFunzione()*** in questo caso, nelle parentesi non va scritto nulla in quanto non vi sono parametri da passare alla funzione (vedremo in seguito cosa significa).
E' facile intuire che posso codificare in questo modo tutto l'alfabeto morse e poi comporre le parole e frasi che voglio semplicemente mettendo nel loop le lettere che voglio ad esempio nello sketch qui sotto lampeggia la parola “osso”.

```
/* Sos codice morse
 
const int pinled = 13;          // pin collegato al led
const int punto = 200;          // tempo led acceso per punto
const int linea = 600;          // tempo led acceso per linea
const int pausaBreve = 300;     // tempo led spento
const int pausaLunga = 800;     // tempo led spento tra 2 lettere

//definisco la procedura per generare la lettera S
void blinkS(){

//definisco la procedura per generare la lettera O
void blinkO(){

void setup() {

void loop() {
  //genero OSSO
 blinkO();
 blinkS();
 blinkS();
 blinkO();
}
```

In questo caso, ho compresso le tre funzioni che non mi interessa visualizzare premendo il meno che c'è a sinistra della parola ***void***, per visualizzarle di nuovo basta cliccare su quello che ora è diventato un più.

Ora passiamo alla realizzazione di questo semplice circuito sulla breadboard usando i componenti del kit, così introduciamo qualche informazione di elettronica.

Cominciamo parlando dei LED (*Light Emitting Diode*) ovvero diodo emettitore di luce. Sono dei componenti da sempre molto usati e attualmente per l'illuminazione anche in ambito domestico. Il simbolo per rappresentarlo in elettronica è il seguente:

Questo simbolo che è lo stesso del diodo, tranne le due freccette che indicano l'emissione di luce, ci dice che l'alimentazione viaggia solo in un senso, dal positivo (anodo) al negativo (catodo). Se colleghiamo il led al contrario non si accenderà. Esistono led di molti colori e forme e praticamente ogni led ha le sue caratteristiche che dobbiamo conoscere per poterlo usare in maniera corretta. Vediamo le specifiche relative ai led rosso e verde del kit andando a leggere le informazioni sul datasheet fornito dal venditore:

Part Number: RL5-R5015 - Super-Red LED (AlGaInP)
absolute maximum ratings: (TA=25°C)

PARAMETER	SYMBOL	RATING	UNIT
Power Dissipation	PD	150	mW
Continuous Forward Current	IF	50	mA
Peak Forward Current (1/10th duty cycle, 0.1ms pulse width)	IFM	100	mA
Reverse Voltage	VR	5.0	V
Operating Temperature	TA	-25~+85	°C
Storage Temperature	TSTG	-40~+85	°C
Reverse Current (VR=5V)	IR	10	microA

Lead Soldering Temperature (3mm from body) 260C (for 3 seconds)

optoelectric characteristics:

PARAMETER	SYMBOL	MAX	TYP	UNIT	TEST
View Angle of Half Power	2ø1/2	±3deg	15	Degree	
Forward Voltage	VF	2.5	2.0	V	IF=20mA
Peak Emission Wavelength	ÿ P	631		nm	IF=20mA
Dominant Emission Wavelength	ÿ D	±3nm	624	nm	IF=20mA
Spectral Line Half Width (FWHM)	DI	13.49		nm	IF=20mA
Luminous Intensity	IV	±20%	5000	mcd	IF=20mA

Part Number: RL5-G8020 - Super-Green LED (GaInN/GaN)
absolute maximum ratings: (TA=25°C)

PARAMETER	SYMBOL	RATING	UNIT
Power Dissipation	PD	100	mW
Continuous Forward Current	IF	20	mA
Peak Forward Current (1/10th duty cycle, 0.1ms pulse width)	IFM	50	mA
Reverse Voltage	VR	5	V
Operating Temperature	TA	-40~+85	°C
Storage Temperature	TSTG	-40~+85	°C
Reverse Current (VR=5V)	IR	10	microA

Lead Soldering Temperature (3mm from body) 260C (for 3 seconds)

optoelectric characteristics:

PARAMETER	SYMBOL	MAX	TYP	UNIT	TEST
View Angle of Half Power	2ø1/2		20	Degree	
Forward Voltage	VF	4.0	3.6	V	IF=20mA
Peak Emission Wavelength	ÿ P		525	nm	
Luminous Intensity	IV		8000	mcd	IF=20mA

Sono un sacco di informazioni ma a noi ne servono essenzialmente 2

per il led rosso:
1 Continuous forward current: corrente continua di alimentazione (IF) 50 mA
2 Forward Voltage: tensione di alimentazione (VF) 2.0 V max 2.5 V usando 20 mA

per il led verde:
1 Continuous forward current: corrente continua di alimentazione (IF) 20 mA
2 Forward Voltage: tensione di alimentazione (VF) 3,6 V max 4.0 V usando 20 mA

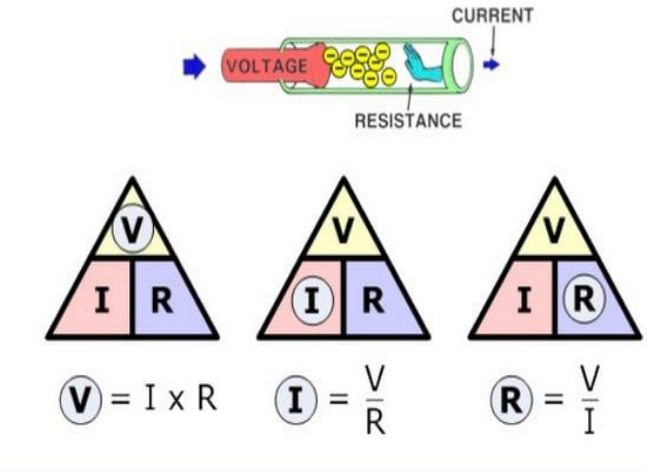
queste 2 informazioni ci dicono come dobbiamo alimentare il nostro led.

Questi datasheet secondo me non sono quelli originali dei led forniti, bensì come appare nella descrizione, sono relativi a led super luminosi, per questo motivo faremo riferimento ai valori “standard” applicati ai led che possiamo trovare in rete e precisamente per quanto riguarda la tensione di alimentazione:

Tipologia LED	tensione di giunzione Vf (volt)
Colore infrarosso	1,3
Colore rosso	1,8
Colore giallo	1,9
Colore verde	2
Colore arancione	2
Flash blu/bianco	3
Colore Blu	3,5
Colore Ultravioletto	4 ÷ 4,5

Arduino ha due opzioni di prelievo della tensione: 5.0 V e 3.3 V. Useremo l'uscita a 5 V, in entrambi i casi però 5V sono troppi quindi dovremo limitare la tensione aggiungendo una resistenza al circuito. La resistenza come dice il nome, oppone resistenza al passaggio della tensione, si comporta come una strozzatura su un tubo per l'acqua. Per calcolare il valore della resistenza da utilizzare dobbiamo usare la legge di Ohm

La prima legge di Ohm recita che in un circuito la corrente "I" è direttamente proporzionale alla tensione "V" e inversamente proporzionale alla resistenza "R", vediamo questa figura



I tre termini sono disposti in un triangolo per aiutare a memorizzare l'operazione da compiere a seconda di quello che vogliamo ottenere. Nel nostro caso siamo interessati al terzo esempio e cioè il calcolo della resistenza necessaria:

per il led rosso

alimentazione da Arduino =	5 V
tensione usata dal led =	1,8 V
tensione da limitare con la resistenza =	5 V – 1,8 V = 3,2 V
corrente massima da usare per il led =	50 mA per sicurezza usiamo 15 mA = 0.015 A

per cui

R = V / I = 3,2 / 0.015 = 213 Ω questo valore non è reperibile quindi useremo quello più vicino da 220 Ω

per il led verde
alimentazione da Arduino = 5 V
tensione usata dal led = 2 V
tensione da limitare con la resistenza = 5 V – 2 V = 3 V
corrente massima da usare per il led = 20 mA per sicurezza usiamo 15mA = 0.015A
per cui

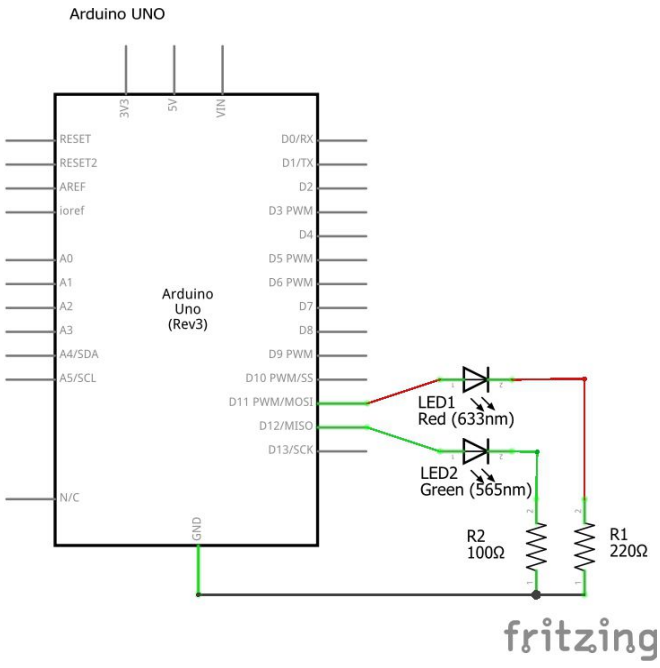
R = V / I = 3 / 0.015 = 200 Ω questo valore non è reperibile quindi ne useremo una da 220 Ω

Le resistenze opponendosi al passaggio di corrente si riscaldano (come le stufette a corrente elettrica), quindi devono essere di potenza adeguata per non bruciarsi nel nostro caso avremo che:
la Potenza in Watt (W) = tensione in Volt (V) x corrente in Ampere (I) quindi

per il led rosso
W= 3,2 V * 0.015 A = 0.045W = 48 mW

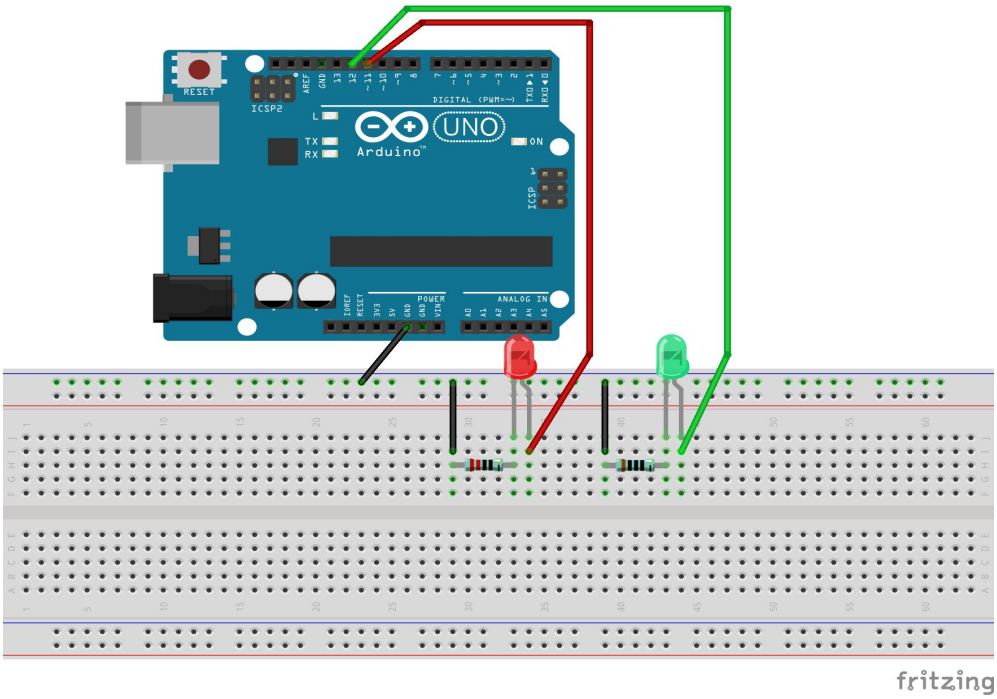
per il led verde
W= 2 V * 0.015 A = 0.021W = 30 mW

In entrambi i casi potremo usare una resistenza da ¼ di Watt (0,250W) che sono quelle fornite nel kit.
Prepariamo dunque il nostro circuito con:
1 led rosso
1 led verde
2 resistenze da 220 Ω (colori rosso rosso nero nero)
la breadboard
cavetti vari
scheda Arduino
e realizziamo il seguente circuito collegando gli anodi (+) del led verde al pin 12 e il rosso al pin 11 di Arduino (sono i terminali lunghi dei led). I 2 catodi (-) (i terminali corti dei led) li colleghiamo alla relativa resistenza e l'altro capo della resistenza lo colleghiamo ad 1 dei pin GND di arduino come da schema qui sotto



* Il circuito è stato realizzato con una resistenza da 100 Ohm seguendo il datasheet fornito dal kit per la tensione del led verde

Che sulla nostra breadboard diventa:



ora colleghiamo la scheda al pc e scriviamo il programma.

```
/* 2 led collegati ad arduino programma 1: lampeggio contemporaneo
 *  led rosso collegato a pin 11
 *  led verde collegato a pin 12
 */

const int ledRosso=11;
const int ledVerde=12;

int tAcceso=200;
int tSpento= 300;

void setup() {
  pinMode(ledRosso,OUTPUT);
  pinMode(ledVerde,OUTPUT);
}

void loop() {
  digitalWrite(ledRosso,HIGH);
  digitalWrite(ledVerde,HIGH);
  delay(tAcceso);
  digitalWrite(ledRosso,LOW);
  digitalWrite(ledVerde,LOW);
  delay(tSpento);
}
```

in questo modo i due led si accendono e spengono assieme
con una semplice modifica possiamo fare in modo che quando 1 è acceso l'altro si spenga e viceversa:

```
/* 2 led collegati ad arduino programma 2: lampeggio alternato
 *  led rosso collegato a pin 11
 *  led verde collegato a pin 12
 */

const int ledRosso=11;
const int ledVerde=12;

int tAcceso=200;
int tSpento= 300;

void setup() {
  pinMode(ledRosso,OUTPUT);
  pinMode(ledVerde,OUTPUT);
}

void loop() {
  digitalWrite(ledRosso,HIGH);
  digitalWrite(ledVerde,LOW);
  delay(tAcceso);
  digitalWrite(ledRosso,LOW);
  digitalWrite(ledVerde,HIGH);
  delay(tSpento);
}
```

Se invece vogliamo far lampeggiare con frequenza diversa ogni led, dobbiamo cambiare approccio, infatti non possiamo usare in questo caso, la funzione ***delay(millisecondi)***, in quanto questa blocca l'esecuzione del programma per un certo tempo rendendo impossibile compiere due azioni in momenti diversi.
Dobbiamo lasciare scorrere il tempo e controllare ad ogni ciclo quanto tempo è trascorso, se è passato abbastanza tempo per accendere i vari led procediamo altrimenti aspettiamo ancora un ciclo e ricontrolliamo. Lo stesso facciamo per spegnerli.
Useremo un nuovo tipo di variabile e precisamente: ***unsigned long*** che corrisponde ad un intero lungo senza segno, in questo tipo di variabili si possono memorizzare numeri che vanno da 0 (zero) a 4.294.967.295 cioè 2^32-1 (usa 4 byte di memoria). In una variabile di questo tipo possiamo memorizzare il tempo trascorso in millisecondi per un periodo di quasi 50 giorni, più che sufficiente per i nostri scopi

ecco lo sketch:

```
1  /* 2 led collegati ad arduino programma 3: lampeggio a frequenza diversa
2   *  per fare ciò uso la funzione interna millis() che ad ogni istante restituisce
3   *  il valore in millisecondi del tempo trascorso dall'ultima
4   *  accensione o reset della scheda
5   *
6   *  led rosso collegato a pin 11
7   *  led verde collegato a pin 12
8   */
9
10 const int ledRosso=11;
11 const int ledVerde=12;
12
13 const unsigned long intervalloRosso=800; //definisco i tempi di lampeggio di ogni led
14 const unsigned long intervalloVerde=300;
15
16 unsigned long timerRosso; // variabile per memorizzare il tempo trascorso
17 unsigned long timerVerde; // 1 per ogni led
18
19 void setup() {
20   pinMode(ledRosso,OUTPUT);
21   pinMode(ledVerde,OUTPUT);
22   timerRosso= millis();     // inizializzo le due variabili con il tempo correntein millisecondi
23   timerVerde= millis();
24 }
25
26 void loop() {
27   //Gestisco il primo led (rosso)
28   if ((millis()- timerRosso) >= intervalloRosso) // se è passato + tempo di quello previsto per l'attesa
29     invertiRosso();                              // chiamo la funzione invertiRosso
30
31   //gestisco il secondo led (verde)
32   if ((millis()- timerVerde) >= intervalloVerde)  // se è passato + tempo di quello previsto per l'attesa
33     invertiVerde();                               // chiamo la funzione invertiVerde
34 }
35
36 // scrivo qui le funzioni esterne che richiamo dal loop
37 void invertiRosso() {
38   if (digitalRead(ledRosso) ==LOW)  // se il led è spento
39       digitalWrite(ledRosso,HIGH);  // lo accendo
40   else                              // altrimenti
41       digitalWrite(ledRosso,LOW);   // lo spengo
42   timerRosso=millis();              // memorizzo quando ho cambiato lo stato del led
43 }
44
45 void invertiVerde() {
46   if (digitalRead(ledVerde) ==LOW)  // se il led è spento
47       digitalWrite(ledVerde,HIGH);  // lo accendo
48   else                              // altrimenti
49       digitalWrite(ledVerde,LOW);   // lo spengo
50   timerVerde=millis();              // memorizzo quando ho cambiato lo stato del led
51 }
```

Qui introduciamo una delle istruzioni fondamentali nella programmazione, l'istruzione ***if... else.*** Con questa istruzione che tradotta in italiano significa: se...oppure, possiamo decidere se compiere una determinata azione o meno a seconda del risultato dell'ipotesi formulata dopo il “se”. Alla riga 28, controlliamo se il tempo attuale meno il tempo memorizzato precedentemente da un risultato maggiore a quanto abbiamo stabilito di attendere per il led rosso, “se” il risultato è positivo allora eseguiamo la riga di codice seguente, altrimenti non facciamo nulla e continuiamo con il programma alla istruzione successiva (prossima if). La stessa cosa nelle due routines, usiamo il se per decidere se dobbiamo accendere o spegnere il led in questione.