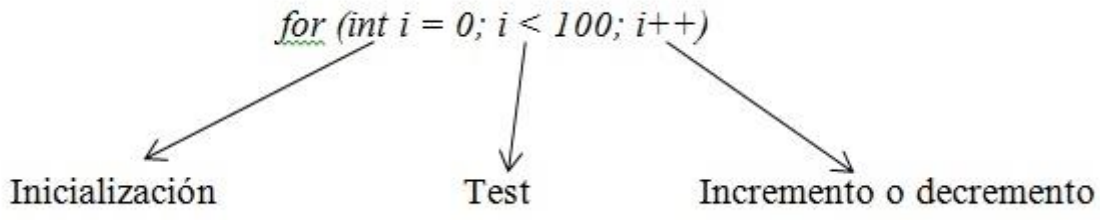
4.1 El bucle "*for*"

Este tipo de bucles se utiliza para ejecutar un bloque de código un cierto número de veces. En general se usan con un contador incremental que va aumentando hasta alcanzar un valor prefijado momento en el que el bucle se da por terminado.

El ejemplo siguiente muestra en típico bucle "*for*" que imprime el valor del contador *i* de 0 hasta 99 hasta que se apaga el Arduino.

```
void setup()
{
   Serial.begin(9600);
}
void loop {
   for (int i = 0; i < 100; i++){
       Serial.println(i);
   }
}
```

Expliquemos cómo funciona el bucle "*for*" basados en el ejemplo anterior:



La variable *i* es inicializada con el valor 0. Al final de cada bucle la variable se incrementa en 1 (*i*++ es una manera abreviada de codificar *i = i + 1*).
El código en el interior del bucle se ejecuta una vez tras otra hasta que alcanza el valor 100. En ese punto el bucle finaliza y recomienza volviendo a poner la variable *i* a 0.
La primera línea del bucle es la instrucción "*for*". Esta instrucción tiene siempre tres partes: inicialización, test y el incremento o decremento de la variable de control o contador.
La inicialización sólo sucede una vez al comienzo del bucle. El test se realiza cada vez que el bucle se ejecuta. Si el resultado del test es "verdadero" (*true*), el bloque de código se ejecuta y el valor del contador se incrementa (++) o decrementa (–) tal como está especificado en la tercera parte del "*for*". El bloque de código (o rutina) continuará ejecutándose hasta que el resultado del test sea "*false*" (es decir, cuando el contador *i* haya alcanzado el valor 100).

4.2 El bucle "*while*"

Muchas veces la condición para que el bucle siga ejecutándose es mucho más compleja que la estudiada en el caso anterior. En estos casos utilizaremos el bucle "*while*". Este tipo de bucle testea una expresión contenida entre paréntesis y ejecuta el bloque de código contenido entre llaves ("*curly braquets*" en inglés) hasta que la expresión es falsa.

```
while (expression) {
   // do statements
}
```

El código del bucle "*for*" que hemos explicado antes se puede reescribir como un bucle "*while*" de la siguiente manera:

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

```
}


void loop(){
    int i = 0;
    while (i < 100){
         Serial.println(i);
         i++;
    }
}
```

Puedes considerar un bucle “*for*” como un caso particular del bucle “*while*”. El bucle ”*while*” se usa cuando no estamos seguros de cuantas veces se va a ejecutar el bucle. El bucle “*while*” se suele usar para testear la entrada de sensores o botones.
La siguiente rutina testea el valor de un sensor:

```
int sensorValue = 0;
while (sensorValue < 2000 {
    sensorValue = analogRead(analogPin);
}
```

El código entre llaves se ejecutará ininterrumpidamente mientras el valor de *sensorValue* sea inferior a 2000.
Cuando programes bucles “*while*” asegúrate de que el código del bucle cambia el valor de la variable test (*sensorValue*). Si no te arriesgas a que el bucle se ejecute indefinidamente (*infinite loop*).

4.3 El bucle “*do while*”

Este tipo de bucle es algo menos usual que el bucle “*while*”. La diferencia con el “*while*” es que el “*do while*” testea la expresión test al final del bucle (es decir después de ejecutar el bucle. De esta manera nos aseguramos que el bucle se ejecuta al menos una vez. El bucle “*do while*” se usa típicamente para leer datos de un fichero

```
do {
    // bloque de código (rutina)
   } while (expresión);
```

Al igual que en el caso anterior, el bloque de código comprendido entre las llaves del bucle se ejecutará hasta que la expresión resulte ser falsa.

4.4 *Continue*. Evitar la ejecución de parte del bloque de código en el bucle.

A veces dentro de un bucle (*for*, *do* o *while*) queremos que sólo parte del bloque de código se ejecute antes de iniciar la siguiente iteración. La instrucción “*continue*” nos permite saltar el resto del bloque de código y volver a la expresión de control del bucle para proceder con las siguientes iteraciones.
Veamos un ejemplo:

```
for (x = 0; x < 255; x ++)
   {
      if (x > 40 && x < 120){  // saltar valores de x en la ejecución del bucle
          continue;
      }
```

```
    digitalWrite(PWMpin, x);
    delay(50);
}
```

En este ejemplo vemos cómo conseguir que se evite la ejecución de la función *digitalWrite()* cuando *x* esté comprendido entre 40 y 120.

4.5 *Goto*. Escapando de un bucle

A veces sucede que se nos “enreda” un poco el código con bucles anidados en otros bucles que están dentro de otros bucles…. y no sabemos cómo salir de este laberinto de bucles. En este tipo de situaciones a veces una instrucción “de escape” puede sernos de gran ayuda.

La instrucción *goto* permite transferir el control a un punto especificado mediante una etiqueta (por ejemplo fuera de la maraña de bucles en la nos hemos metido). Esta es la sintaxis del *goto*:

```
etiqueta:
…
goto etiqueta; // transfiere la ejecución del programa a la etiqueta
```

Normalmente se desaconseja el uso del *goto* en programación ya que el uso del *goto* puede llevarnos a la creación de programas con un flujo de ejecución indefinido, que puede dificultar mucho la detección y eliminación errores (debugging).
Veamos un ejemplo de utilización del *goto:*

```
for(byte r = 0; r < 255; r++){
    for(byte g = 255; g > -1; g--){
        for(byte b = 0; b < 255; b++){
            if (analogRead(0) > 250){ goto escape;}
               // more statements ...
        }
    }
}
escape:
```

En este ejemplo podemos salir “bruscamente” de ese grupo de tres bucles anidados en caso de que la lectura de un pin analógico sea mayor que 250. En ese caso, el flujo de ejecución sería transferido fuera de los tres bucles a la etiqueta “*escape*”.

```
int tiempo = 90;

int led1 = 2;
int led2 = 3;
int led3 = 4;
int led4 = 5;
int led5 = 6;
int led6 = 7;
int led7 = 8;
int led8 = 9;

void setup()
{
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
  pinMode(led3, OUTPUT);
  pinMode(led4, OUTPUT);
  pinMode(led5, OUTPUT);
  pinMode(led6, OUTPUT);
  pinMode(led7, OUTPUT);
  pinMode(led8, OUTPUT);
}
void loop()
{
  for(byte s1 = 0; s1 < 12; s1++)
  {
    digitalWrite(led1, HIGH);
    delay(tiempo);
    digitalWrite(led1, LOW);

    digitalWrite(led2, HIGH);
    delay(tiempo);
    digitalWrite(led2, LOW);

    digitalWrite(led3, HIGH);
    delay(tiempo);
    digitalWrite(led3, LOW);

    digitalWrite(led4, HIGH);
    delay(tiempo);
    digitalWrite(led4, LOW);

    digitalWrite(led5, HIGH);
    delay(tiempo);
    digitalWrite(led5, LOW);

    digitalWrite(led6, HIGH);
    delay(tiempo);
    digitalWrite(led6, LOW);

    digitalWrite(led7, HIGH);
    delay(tiempo);
    digitalWrite(led7, LOW);

    digitalWrite(led8, HIGH);
    delay(tiempo);
    digitalWrite(led8, LOW);
  }

  for( byte s2 = 0; s2 < 12; s2++)
  {
    digitalWrite(led8, HIGH);
    delay(tiempo);
    digitalWrite(led8, LOW);

    digitalWrite(led7, HIGH);
    delay(tiempo);
    digitalWrite(led7, LOW);

    digitalWrite(led6, HIGH);
    delay(tiempo);
    digitalWrite(led6, LOW);

    digitalWrite(led5, HIGH);
    delay(tiempo);
    digitalWrite(led5, LOW);

    digitalWrite(led4, HIGH);
    delay(tiempo);
    digitalWrite(led4, LOW);

    digitalWrite(led3, HIGH);
    delay(tiempo);
    digitalWrite(led3, LOW);

    digitalWrite(led2, HIGH);
    delay(tiempo);
    digitalWrite(led2, LOW);

    digitalWrite(led1, HIGH);
    delay(tiempo);
    digitalWrite(led1, LOW);
  }

  for( byte s3 = 0; s3 < 12; s3++)
  {
    digitalWrite(led1, HIGH);
    delay(tiempo);
    digitalWrite(led1, LOW);

    digitalWrite(led2, HIGH);
    delay(tiempo);
    digitalWrite(led2, LOW);

    digitalWrite(led3, HIGH);
    delay(tiempo);
    digitalWrite(led3, LOW);

    digitalWrite(led4, HIGH);
    delay(tiempo);
    digitalWrite(led4, LOW);
```

```
digitalWrite(led5, HIGH);
delay(tiempo);
digitalWrite(led5, LOW);

digitalWrite(led6, HIGH);
delay(tiempo);
digitalWrite(led6, LOW);

digitalWrite(led7, HIGH);
delay(tiempo);
digitalWrite(led7, LOW);

digitalWrite(led8, HIGH);
delay(tiempo);
digitalWrite(led8, LOW);

digitalWrite(led7, HIGH);
delay(tiempo);
digitalWrite(led7, LOW);

digitalWrite(led6, HIGH);
delay(tiempo);
digitalWrite(led6, LOW);

digitalWrite(led5, HIGH);
delay(tiempo);
digitalWrite(led5, LOW);

digitalWrite(led4, HIGH);
delay(tiempo);
digitalWrite(led4, LOW);

digitalWrite(led3, HIGH);
delay(tiempo);
digitalWrite(led3, LOW);
```

```
  digitalWrite(led2, HIGH);
  delay(tiempo);
  digitalWrite(led2, LOW);
}

for( byte s4 = 0; s4 < 12; s4++)
{
  digitalWrite(led1, HIGH);
  digitalWrite(led2, HIGH);
  digitalWrite(led3, HIGH);
  digitalWrite(led4, HIGH);
  delay(100);
  digitalWrite(led1, LOW);
  digitalWrite(led2, LOW);
  digitalWrite(led3, LOW);
  digitalWrite(led4, LOW);
  delay(100);
  digitalWrite(led1, HIGH);
  digitalWrite(led2, HIGH);
  digitalWrite(led3, HIGH);
  digitalWrite(led4, HIGH);
  delay(100);
  digitalWrite(led1, LOW);
  digitalWrite(led2, LOW);
  digitalWrite(led3, LOW);
  digitalWrite(led4, LOW);
  delay(100);
  digitalWrite(led1, HIGH);
  digitalWrite(led2, HIGH);
  digitalWrite(led3, HIGH);
  digitalWrite(led4, HIGH);
  delay(100);
  digitalWrite(led1, LOW);
  digitalWrite(led2, LOW);
  digitalWrite(led3, LOW);
  digitalWrite(led4, LOW);
  delay(100);
```

```
  digitalWrite(led5, HIGH);
  digitalWrite(led6, HIGH);
  digitalWrite(led7, HIGH);
  digitalWrite(led8, HIGH);
  delay(100);
  digitalWrite(led5, LOW);
  digitalWrite(led6, LOW);
  digitalWrite(led7, LOW);
  digitalWrite(led8, LOW);
  delay(100);
  digitalWrite(led5, HIGH);
  digitalWrite(led6, HIGH);
  digitalWrite(led7, HIGH);
  digitalWrite(led8, HIGH);
  delay(100);
  digitalWrite(led5, LOW);
  digitalWrite(led6, LOW);
  digitalWrite(led7, LOW);
  digitalWrite(led8, LOW);
  delay(100);
  digitalWrite(led5, HIGH);
  digitalWrite(led6, HIGH);
  digitalWrite(led7, HIGH);
  digitalWrite(led8, HIGH);
  delay(100);
  digitalWrite(led5, LOW);
  digitalWrite(led6, LOW);
  digitalWrite(led7, LOW);
  digitalWrite(led8, LOW);
  delay(100);
}

for( byte s5 = 0; s5 < 12; s5++)
{
  digitalWrite(led1, HIGH);
  digitalWrite(led2, HIGH);
  digitalWrite(led3, HIGH);
```

```
 digitalWrite(led4, HIGH);
 delay(500);
 digitalWrite(led1, LOW);
 digitalWrite(led2, LOW);
 digitalWrite(led3, LOW);
 digitalWrite(led4, LOW);

 digitalWrite(led5, HIGH);
 digitalWrite(led6, HIGH);
 digitalWrite(led7, HIGH);
 digitalWrite(led8, HIGH);
 delay(500);
 digitalWrite(led5, LOW);
 digitalWrite(led6, LOW);
 digitalWrite(led7, LOW);
 digitalWrite(led8, LOW);
}

for( byte s6 = 0; s6 < 20; s6++)
{
 digitalWrite(led1, HIGH);
 digitalWrite(led8, HIGH);
 delay(150);
 digitalWrite(led1, LOW);
 digitalWrite(led8, LOW);

 digitalWrite(led2, HIGH);
 digitalWrite(led7, HIGH);
 delay(150);
 digitalWrite(led2, LOW);
 digitalWrite(led7, LOW);

 digitalWrite(led3, HIGH);
 digitalWrite(led6, HIGH);
 delay(150);
 digitalWrite(led3, LOW);
 digitalWrite(led6, LOW);
```

```
 digitalWrite(led4, HIGH);
 digitalWrite(led5, HIGH);
 delay(150);
 digitalWrite(led4, LOW);
 digitalWrite(led5, LOW);
}

for( byte s7 = 0; s7 < 20; s7++)
{
 digitalWrite(led4, HIGH);
 digitalWrite(led5, HIGH);
 delay(150);
 digitalWrite(led4, LOW);
 digitalWrite(led5, LOW);

 digitalWrite(led3, HIGH);
 digitalWrite(led6, HIGH);
 delay(150);
 digitalWrite(led3, LOW);
 digitalWrite(led6, LOW);

 digitalWrite(led2, HIGH);
 digitalWrite(led7, HIGH);
 delay(150);
 digitalWrite(led2, LOW);
 digitalWrite(led7, LOW);

 digitalWrite(led1, HIGH);
 digitalWrite(led8, HIGH);
 delay(150);
 digitalWrite(led1, LOW);
 digitalWrite(led8, LOW);
}

for( byte s8 = 0; s8 < 20; s8++)
{
```

```
 digitalWrite(led1, HIGH);
 digitalWrite(led8, HIGH);
 delay(150);
 digitalWrite(led1, LOW);
 digitalWrite(led8, LOW);

 digitalWrite(led2, HIGH);
 digitalWrite(led7, HIGH);
 delay(150);
 digitalWrite(led2, LOW);
 digitalWrite(led7, LOW);

 digitalWrite(led3, HIGH);
 digitalWrite(led6, HIGH);
 delay(150);
 digitalWrite(led3, LOW);
 digitalWrite(led6, LOW);

 digitalWrite(led4, HIGH);
 digitalWrite(led5, HIGH);
 delay(150);
 digitalWrite(led4, LOW);
 digitalWrite(led5, LOW);

 digitalWrite(led3, HIGH);
 digitalWrite(led6, HIGH);
 delay(150);
 digitalWrite(led3, LOW);
 digitalWrite(led6, LOW);

 digitalWrite(led2, HIGH);
 digitalWrite(led7, HIGH);
 delay(150);
 digitalWrite(led2, LOW);
 digitalWrite(led7, LOW);
  }
}
```