Guía de programación básica: Instrucciones de selección

Este documento ha sido creado como material extra para el curso nivel 1 de arduino.

Instrucciones de selección.

En programación, podemos alterar el flujo de un programa y elegir qué partes de nuestro código fuente deben ser ejecutadas cuando se cumpla cierta condición.

¿Para qué nos puede servir una instrucción de selección?
Por ejemplo, podemos tener dos porciones de código que revisen el valor de una variable, que almacena la temperatura leída por un sensor, si la temperatura rebasa los 30 °C debemos encender un ventilador para enfriar el ambiente; en caso contrario, es probable que decidamos mantenerlo apagado si la temperatura no ha rebasado los 30 °C.

En Arduino, tenemos 2 instrucciones de selección: *if* y *switch*

Instrucción if

La instrucción if nos permite ejecutar un bloque de código cuando cierta condición se cumple, su sintaxis es la siguiente

```
if (condición) {
	Código fuente que debe ejecutarse si condición es verdadera
}
```

La instrucción if se compone de 3 partes principales:

- La palabra reservada *if*
- Una expresión *booleana* (condición) la cual debe evaluarse como *true* (verdadero) *false* (falso)

- Un bloque de código delimitado por llaves, que debe contener el código fuente que ha de ser ejecutado cuando condición es *true*

Expresiones booleanas

Una expresión booleana es una operación que puede resultar en uno de dos valores: true (verdadero) o false (falso) una expresión booleana en el mayor de los casos requiere de un operador y dos operandos.

Por ejemplo en la siguiente expresión booleana:

5 > 4

5 y 4 son los operandos y > es el operador, esta expresión se lee como “cinco mayor que 4”, el resultado de esta expresión booleana es *true* ya que en efecto, 5 es mayor que 4.

Para poder escribir expresiones booleanas contamos con distintos operadores los cuales pueden ser clasificados en dos categorías: relacionales y lógicos.

Operadores relacionales (o de comparación)

Los operadores relacionales nos sirven para comparar dos valores, un operador relacional fue utilizado en el ejemplo visto anteriormente: *5 > 4*

Los operadores relacionales disponibles en arduino son:

Operador	Descripción	Ejemplo de uso
>	Retorna true si el primer operando es mayor que el segundo, se lee como “mayor que”.	*78 > 66* Retorna true ya que 78 es mayor que 66.
<	Retorna true si el primer operando es mejor que el segundo operando, se lee como “menor que”.	*128.12 < 0* Retorna false ya que 128.12 no es menor que 0.
>=	Retorna true si el primer operando es mayor o igual que el segundo operando, se lee como “mayor o igual que”.	*68 >= 67* Retorna true ya que 68 es mayor o igual que 67
<=	Retorna true si el primer operando es menor o igual que el segundo operando, se lee como “menor o igual que”.	*45 <= 4* Retorna false ya que 45 no es ni menor mi igual que 4
==	Retorna true si ambos operandos son iguales, se lee como “igual que”. Nota: son dos signos igual, no confundir con el operador de asignación = ya que puede conducir a un comportamiento indeseado en el programa.	*41.5 == 41.5* Retorna true ya que ambos operandos son exactamente iguales

!=	Retorna true si ambos operandos son diferentes, se lee como “diferente qué”	*32 != 32* Retorna false ya que ambos operandos no son diferentes

Operadores lógicos (o booleanos)

Los operadores lógicos nos sirven para unir (o conjugar) dos o más expresiones booleanas, a excepción del operador ! ya que únicamente necesita un operando para funcionar.

Los operadores lógicos en arduino son:

Operador	Descripción	Ejemplo de uso
&&	Operador AND: Retorna true si ambos operandos son true, en cualquier otro caso, retorna false Se lee como “y”	*(78 > 66) && (35 == 35)* Retorna true ya que 78 es mayor que 66 y 35 es igual que 35. La operación equivalente en este ejemplo es *true && true.*
||	Operador OR: Retorna true si al menos uno de los operandos es true, si ambos operandos son false, retorna false. Se lee como “o”	*(128.12 < 0) || (45 > 4)* Retorna true ya que 128.12 no es menor que 0, pero 45 es mayor que 4. La operación equivalente en este ejemplo es *false || true.*
!	Operador NOT: Invierte el resultado de una expresión booleana. Se lee como “no”	*!(3 > 2)* Retorna true, la expresión interna 3 > 2 retorna false, pero al aplicarle el operador NOT el resultado es invertido. La operación equivalente en este ejemplo es !false.

Nota: los paréntesis en los ejemplos de uso no son estrictamente necesarios, fueron utilizados para resaltar la agrupación de las operaciones booleanas.

El uso de los operadores lógicos también puede ser desglosado en lo que se llama “Tabla de la verdad” este tipo de tablas también son utilizadas en electrónica al estar trabajando con compuertas lógicas.

Operación AND

Operando 1	Operando 2	Resultado
false	false	*false*
false	true	*false*
true	false	*false*
true	true	*true*

Podemos observar que para que el resultado de una operación AND sea verdadero (true) ambos operandos deben ser verdaderos (true).

Operación OR

Operando 1	**Operando 2**	**Resultado**
false	false	*false*
false	true	*true*
true	false	*true*
true	true	*true*

Podemos observar que para el resultado de una operación OR sea verdadero (true), al menos uno de sus operandos tiene que ser verdadero (true).

Operación NOT

Operando 1	**Resultado**
false	true
true	false

La operación NOT sólo requiere un operando, cuyo valor booleano es invertido.

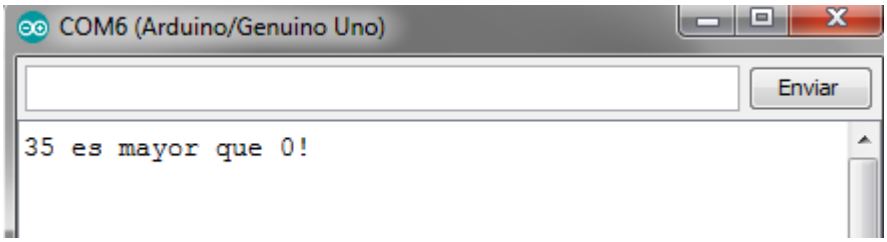
Uso de instrucciones if en Arduino

Basta de tanta teoría, hagamos un poco de práctica, comencemos por escribir nuestra primer instrucción *if* en arduino

Analicemos este programa

```
1 void setup() {
2   Serial.begin(9600);
3 
4   if(35 > 0){
5     Serial.println("35 es mayor que 0!");
6   }
7 }
8 
9 void loop() {}
```

Si cargamos este programa a nuestro arduino, deberíamos ver el siguiente mensaje en el terminal serial.



El mensaje se muestra porque la condición escrita en el bloque if en la línea 4 se cumple, efectivamente, 35 es mayor que 0, por lo tanto, el bloque de código entre las líneas 4 y 6 es ejecutado.

Ahora, ¿Qué pasa si modificamos la condición de la línea 4?

```
1 void setup() {
2   Serial.begin(9600);
3
4   if(35 < 0){
5     Serial.println("35 es mayor que 0!");
6   }
7 }
8
9 void loop() {}
```

Al cargar el programa notamos que no aparece nada.
Esto es debido a que la condición de la línea 4 es falsa (false) 35 no es menor que 0 (al menos en esta dimensión)

Por supuesto, también podemos utilizar instrucciones *if* con variables, supongamos que la variable *temperatura* almacena (convenientemente) la temperatura en °C leída por un sensor.

En este ejemplo hemos movido el bloque if a la función loop

```
1  void setup() {
2    Serial.begin(9600);
3  }
4
5  void loop() {
6    float temperatura = 45.25;
7
8    if(temperatura >= 30){
9      Serial.println("La temperatura es muy alta, encender ventilador");
10   }
11
12   delay(1000);
13 }
```

Si cargamos este programa, veremos en el terminal serial la frase *"La temperatura es muy alta, encender ventilador"* ya que 45.25 (el valor de temperatura) es mayor que 30.

En el programa anterior definimos el momento en el que un ventilador se debería encender en caso de que la temperatura sea muy alta, pero ¿Cuándo lo apagaremos? Una respuesta obvia podría ser: "Tenemos que apagar el ventilador cuando la temperatura no sea mayor o igual a 30" Dada esta respuesta, podríamos modificar nuestro código fuente de la siguiente manera.

```
1  void setup() {
2    Serial.begin(9600);
3  }
4
5  void loop() {
6    float temperatura = 45.25;
7
8    if(temperatura >= 30){
9      Serial.println("La temperatura es muy alta, encender ventilador");
10   }
11
12   Serial.println("La temperatura no es muy alta, apaga el ventilador!");
13
14   delay(1000);
15 }
```

Si cargamos este programa al Arduino, veremos algo así en el terminal serial:

```
La temperatura es muy alta, encender ventilador
La temperatura no es muy alta, apaga el ventilador!
La temperatura es muy alta, encender ventilador
La temperatura no es muy alta, apaga el ventilador!
```

Nos está diciendo que… ¿La temperatura es muy alta y no muy alta a la vez?

Evidentemente tenemos un problema aquí, la instrucción de la línea 12 se ejecuta independientemente del valor de la temperatura; en este caso, podemos utilizar un bloque *else* para que la frase “La temperatura no es muy alta, apaga el ventilador!” aparezca únicamente cuando la temperatura no sea mayor o igual a 30.

```
1  void setup() {
2    Serial.begin(9600);
3  }
4
5  void loop() {
6    float temperatura = 45.25;
7
8    if(temperatura >= 30){
9      Serial.println("La temperatura es muy alta, encender ventilador");
10   } else {
11     Serial.println("La temperatura no es muy alta, apaga el ventilador!");
12   }
13
14   delay(1000);
15 }
```

Para lograr que la frase “La temperatura no es muy alta, apaga el ventilador!” sea mostrada, tenemos que modificar el valor de temperatura de tal manera que sea menor que 30.

Con el uso del bloque *else* podemos definir un bloque de código que ha de ejecutarse si la condición de un bloque *if* no se cumple.

Un bloque *else* **siempre** debe estar acompañado por un bloque if, un bloque *else* no puede existir por sí solo.

Hasta el momento nuestro programa que revisa (hipotéticamente) la temperatura de un sensor sólo puede reconocer 2 rangos de temperatura: temperaturas menores a 30 °C y temperaturas mayores o iguales que 30 °C (hey, hey, calma, durante el curso realizaremos esta práctica con un sensor real :D).

¿Qué tal si ahora queremos reconocer 4 rangos distintos de temperatura ambiente, por ejemplo: frio, agradable, cálido y “Me estoy quemando”?

Esto lo podemos realizar utilizando varios bloques if con condiciones un poco más complejas ¿No es así?

Probemos el siguiente programa

```
1  void setup() {
2    Serial.begin(9600);
3  }
4
5  void loop() {
6    float temperatura = 16;
7
8    if(temperatura >0  && temperatura < 15){
9      Serial.println("Hace frio!");
10   }
11
12   if(temperatura >= 15 && temperatura < 25){
13     Serial.println("El clima es agradable");
14   }
15
16   if(temperatura >= 25 && temperatura < 32){
17     Serial.println("El clima es cálido");
18   }
19
20   if (temperatura >= 32)
21     Serial.println("Me estoy quemando!");
22
23   delay(1000);
24 }
```

Ahora, con una serie de bloques if, hemos agregado el código necesario para detectar 4 niveles de temperatura, impresionante ¿No?, carga este programa en tu arduino y modifica el valor de la variable temperatura para ver en pantalla todos los mensajes para cada nivel de temperatura.

Los 4 niveles de temperatura funcionan de la siguiente manera

- Si la temperatura tiene un valor mayor a 0 **y** si tiene un valor menor que 15, la frase “Hace frio!” será mostrada en pantalla.

- Si la temperatura es mayor o igual a 15 **y** si es menor que 25, la frase “El clima es agradable” será mostrada en pantalla.

- Si la temperatura es mayor o igual a 25 **y** si es menor que 32, la frase “El clima es cálido” será mostrada en pantalla.

- Si la temperatura es mayor o igual a 32, la fase “Me estoy quemando” será mostrada en pantalla.

Nota: en el último bloque if las llaves fueron omitidas, esto es porque si un bloque *if* **sólo cuenta con una instrucción**, las llaves pueden omitirse.

En cada iteración de la función loop se realizan 4 comparaciones correspondientes a cada nivel de temperatura, esto parece funcionar correctamente, pero puede llegar a ser un problema de rendimiento en un programa más grande.

Por ejemplo, si la temperatura cae en el primer nivel *(temperatura = > 0 && temperatura < 15)* ¿Por qué molestarnos en revisar los otros 3 niveles? Si ya sabemos que sus respectivas condiciones no se van a cumplir.

Al saber esto, nos damos cuenta que nuestro programa no es tan impresionante después de todo, ya que está ejecutando instrucciones if innecesarias ☹…

Pero podemos hacerlo mejor :-D

El bloque else if

Para evitar la revisión de bloques if innecesarios, podemos transformarlos en bloques *else if*:

```
1  void setup() {
2    Serial.begin(9600);
3  }
4
5  void loop() {
6    float temperatura = 16;
7
8    if(temperatura >0  && temperatura < 15){
9      Serial.println("Hace frio!");
10   } else if(temperatura >= 15 && temperatura < 25){
11     Serial.println("El clima es agradable");
12   } else if(temperatura >= 25 && temperatura < 32){
13     Serial.println("El clima es cálido");
14   } else if(temperatura >= 32)
15     Serial.println("Me estoy quemando!");
16
17   delay(1000);
18 }
19
```

Este programa parece funcionar exactamente igual que antes, y así es, pero internamente funciona de manera distinta ya que, una vez que el valor de la temperatura cumpla con las condiciones de cualquier bloque if, Arduino no se molestará en revisar los bloques restantes.

Por ejemplo, si la temperatura tiene un valor de 25:

1. Se revisará el primer nivel definido (temperatura > 0 && temperatura < 15)
   La condición del primer bloque if no se cumple, así que se procede a revisar el segundo bloque
2. Se revisa el segundo nivel definido (temperatura >= 15 && temperatura < 25)
   La condición del bloque else if no se cumple, así que se procede a revisar el siguiente bloque
3. Se revisa el tercer nivel definido (temperatura >= 25 && temperatura < 32)
   Esta condición se cumple, por lo tanto la frase “El clima es cálido” será mostrada en pantalla, y la ejecución de nuestra estructura else if finalizará aquí, una condición se cumplió, así que los demás niveles definidos de temperatura no serán revisados.

Ejercicio: Agregar un bloque else al programa anterior, dentro del bloque else poner la siguiente instrucción *Serial.println("Congelado?");*

Algo como esto:

```
1  void setup() {
2    Serial.begin(9600);
3  }
4
5  void loop() {
6    float temperatura = 5;
7
8    if(temperatura >0  && temperatura < 15){
9      Serial.println("Hace frio!");
10   } else if(temperatura >= 15 && temperatura < 25){
11     Serial.println("El clima es agradable");
12   } else if(temperatura >= 25 && temperatura < 32){
13     Serial.println("El clima es cálido");
14   } else if(temperatura >= 32)
15     Serial.println("Me estoy quemando!");
16   else
17     Serial.println("Congelado?");
18
19   delay(1000);
20 }
```

¿Para qué valores de temperatura será mostrado ese mensaje?
¿Puedes deducir el comportamiento de un bloque *else* agregado al final de una estructura *else if*?

Instrucción switch

La instrucción switch la podemos ver como una versión comprimida de un bloque *else if* el cual ejecutará sus respectivos bloques de código si la variable que estamos probando es igual a alguno de los valores que especifiquemos dentro de la estructura switch.

La sintaxis de una estructura switch es la siguiente

```
switch (variable){
	case a:
	//codigo a ejecutar si variable == a
	break;

	case b:
	//codigo a ejecutar si variable == b
	break;

	case c:
	//codigo a ejecutar si variable == c
	break;

	case n:
	//codigo a ejecutar si variable == n
	break;


	default:
		//codigo a ejecutar si variable no es igual a ninguno de los valores
		//especificadas en cada case
	break;
}
```

Básicamente se compone de 4 partes

- La palabra reservada *switch*, seguido de la variable que queramos probar entre paréntesis.
- Un bloque de código que contendrá una serie de instrucciones *case*.
- Una serie de instrucciones case las cuales deben estar acompañadas de un posible valor para la variable que estemos probando, un bloque de código fuente que ha de ejecutarse si *variable = valor_a_probar* y una instrucción *break* que indica el fin de un bloque de código para una instrucción case.
- Una instrucción *default* seguida de un bloque de código el cual se ha de ejecutar cuando la variable que estemos probando no cumpla con las condiciones de ninguno de las instrucciones case que hayamos definido anteriormente y la palabra reservada *break*.
  La instrucción ***default*** **no es necesaria para una estructura switch, es opcional.**

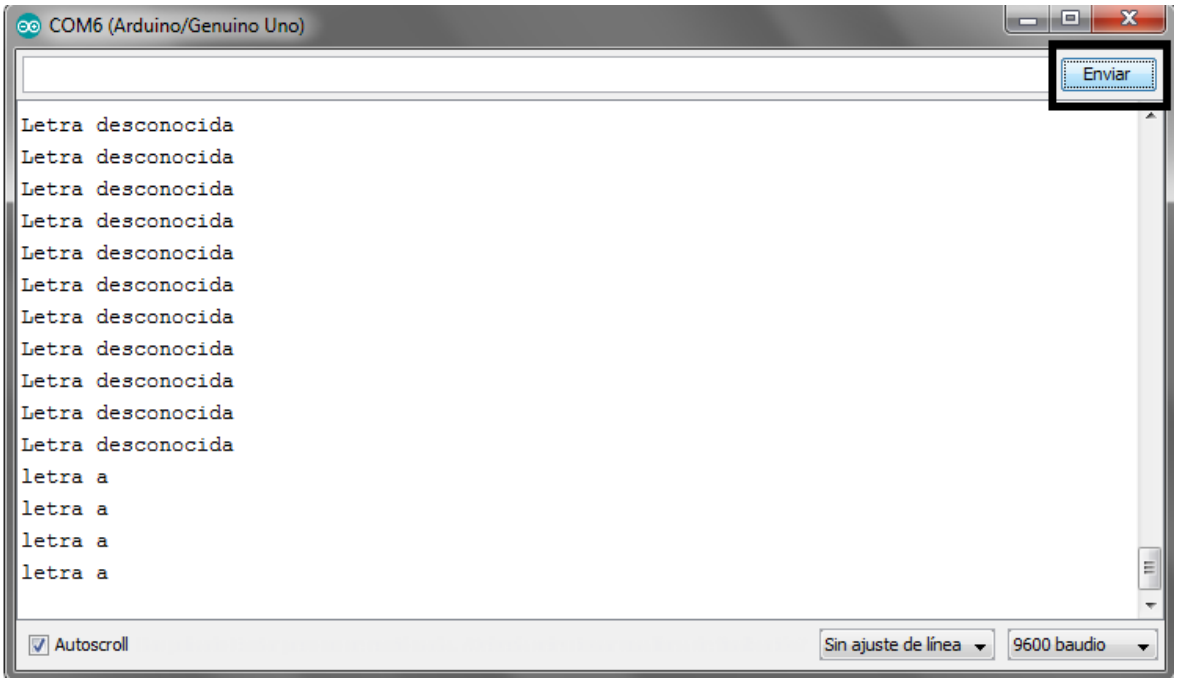
Todo esto se explica mejor con un ejemplo práctico.

```
1  void setup() {
2    Serial.begin(9600);
3  }
4
5  void loop() {
6    int botonesPresionados = 21;
7
8    switch(botonesPresionados){
9      case 1:
10       Serial.println("Un boton presionado");
11     break;
12
13     case 20:
14       Serial.println("20 botones presionados... es posible?");
15     break;
16
17     case 4:
18       Serial.println("4 botones presionados!");
19     break;
20
21     default:
22       Serial.println("Ni 1, ni 20, ni 4");
23     break;
24   }
25
26   delay(1000);
27 }
```

También podemos usar otros tipos de dato con la instrucción switch, no solamente int, podemos tener bloques switch con variables byte, float, double, o incluso char.

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

char letra = 'a';

void loop() {

  //Abre la terminal serial y envia una letra o una serie de letras
  if(Serial.available())
    letra = Serial.read();

  switch(letra){
    case 'a':
      Serial.println("letra a");
    break;

    case 'b':
      Serial.println("letra b");
    break;

    case 64: //Internamente un char tambien es un numero
      Serial.println("esto es una @?");
    break;

    default:
      Serial.println("Letra desconocida");
    break;
  }

  delay(1000);
```

Para ver mejor los efectos de este último programa, abre la terminal serial y envía algo de texto



Recuerda que puedes encontrar el código fuente utilizado para los ejemplos de esta guía en https://github.com/sct999/introduccion-programacion-arduino

Esperamos que esta guía haya sido de ayuda ;)
¡Nos vemos en clase!