UNIDAD 1: PRIMEROS PROGRAMAS

OBJETIVOS

Realizar los primeros programas que te permitan manejar periféricos de entradas y salidas (E/S) digitales de forma rápida y sencilla.

Para ello analizaremos el aspecto que tiene un programa escrito en el lenguaje Arduino de programación. Se trata de un lenguaje de alto nivel con sus propias reglas, sentencias y sintaxis. A continuación, vas a estudiar las sentencias elementales que deberás usar para el control básico de los periféricos de entrada y salida digital.

TEORÍA

- ASPECTOS DE UN PROGRAMA
  - Zona de comentarios
  - Zona de declaración de variables y funciones.
  - Zona para tareas de configuración
  - Zona para el cuerpo principal del programa.
  - Zona de mensajes.
- SENTENCIAS EN UN PROGRAMA
  - Función: setup()
  - Función: loop()
  - Función: pinMode()
  - Función: digitalRead()
  - Función: digitalWrite()
- OPERADORES LÓGICOS
  - Operador NO (NOT)
  - Operador Y (AND)
  - Operador O (OR)
  - Combinando operadores

PRÁCTICAS

- EJEMPLO 1: Encendido de un led v.1
- EJEMPLO 2: Encendido de un led v.1
- EJEMPLO 3: Encendido de un led v.1
- EJEMPLO 4: Encendido invertido de un led
- EJEMPLO 5: Control desde dos pulsadores
- EJEMPLO 6: Trabajo personal

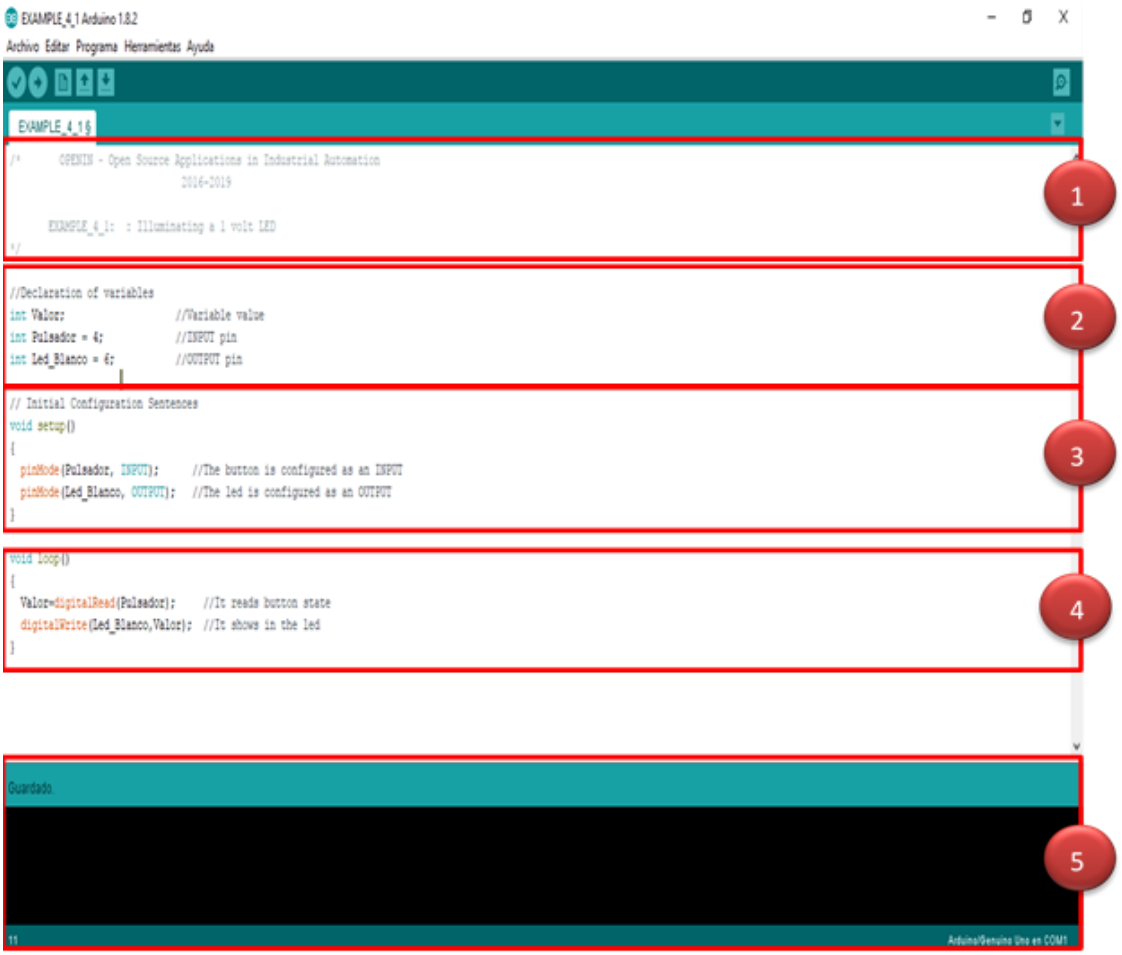
PRACTICE MATERIALS

- *Ordenador portátil o de sobremesa.*
- *Entorno de trabajo Arduino IDE que se incluye en el material complementario y que se supone instalado y configurado.*
- *Tarjeta controladora Arduino UNO*
- *Cable USB.*

TABLE OF CONTENTS

TEORÍA

1. ASPECTO DE UN PROGRAMA



A. ZONA DE COMENTARIOS

A Todo programa debiera de empezar dando una serie de información. Por ejemplo, en este caso, el nombre del curso, fecha, autor, empresa, etc… También sería interesante una explicación sobre en qué consiste o qué hace el programa

Reciben el nombre de “Comentarios de cabecera”. Puedes poner todos los comentarios que quieras y donde quieras, pero siempre deben ir encerrados entre los caracteres “/*” y “*/”. Fíjate bien en Figure 1.

También puede poner comentarios más sencillos, de una sola línea o renglón. En este caso basta con que vayan precedidos de los caracteres "//". Los verás muy a menudo..

Figure 1

Recuerda y no seas perezoso. Acostúmbrate a comentar tus programas. Quizá en un futuro necesites de ellos para recordar lo qué hacías, cómo y por qué.

B. ZONA DE DECLARACIÓN DE VARIABLES Y FUNCIONES

Esta segunda zona te recomiendo que la uses para hacer la declaración de variables y funciones. (Figure 2). Todavía no sabes lo qué son, pero ya las estudiarás más adelante. Imagina a una variable como una especie de caja o receptáculo a la que le asignas un nombre y quizá también le des un valor. Se guarda en la memoria RAM del controlador y la podrás usar posteriormente tantas veces como la necesites. Una función a su vez está formada por un conjunto de otras sentencias, instrucciones o funciones. A ese conjunto le asignas un nombre y lo usas también tantas veces como necesites.

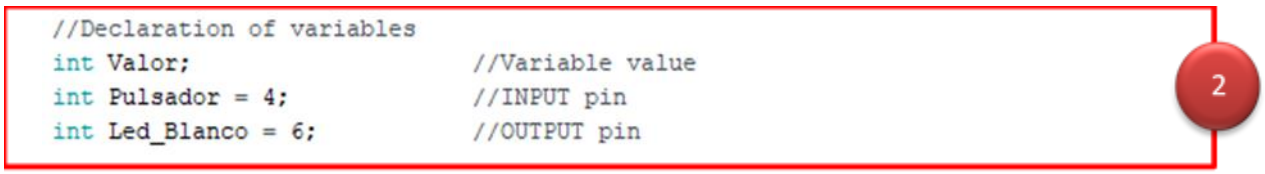


Figure 2

En el ejemplo de la figura se han creado tres variables: "value", "Pulsador" y "Led_Blanco". En la primera se va a guardar el valor que resulta de leer una entrada digital. A la segunda, la variable "Pulsador", se le asigna el valor 4. Corresponde con el número de la patilla de entrada donde está conectado el pulsador cuyo estado vamos a leer. A la variable "Led_Blanco" le hemos asignado el valor 6.

En general tanto las variables como las funciones han de declararse ANTES de ser utilizadas posteriormente por el programa.

C. ZONA PARA TAREAS DE CONFIGURACIÓN

Normalmente todo programa escrito en el lenguaje Arduino, empieza ejecutando unas cuantas instrucciones o funciones de configuración. Con ellas se establece, entre otras cosas, qué patillas deben actuar como entradas y quiénes como salidas.

```
// Initial Configuration Sentences
void setup()
{
  pinMode(Pulsador, INPUT);      //The button is configured as an INPUT
  pinMode(Led_Blanco, OUTPUT);   //The led is configured as an OUTPUT
}
```

3

Figure 3

En el ejemplo de la Figure 3 la patilla de la variable “Pulsador”, a la que anteriormente se le asignó el valor 4, se le configura como entrada (INPUT).

La patilla correspondiente a la variable “Led_Blanco”, es decir la patilla 6, se le configura como salida (OUTPUT).

Normalmente las sentencias o funciones de configuración sólo se ejecutan una vez, cada vez que reinicias el sistema (RESET) o conectas la alimentación

D. ZONA PARA EL CUERPO PRINCIPAL DEL PROGRAMA

Aquí es donde debes escribir todas las instrucciones, sentencias o funciones que forman parte de tu programa. En el ejemplo que nos ocupa únicamente hay dos funciones, pero piensa que un programa puede estar compuesto de cientos o miles.

```
void loop()
{
  Valor=digitalRead(Pulsador);     //It reads button state
  digitalWrite(Led_Blanco,Valor);  //It shows in the led
}
```

4

Figure 4

La primera función, “Valor=digitalRead(Pulsador);”, lee el estado digital del pulsador conectado en la patilla 4 y que anteriormente se configuró como entrada. El estado de ese pulsador se guarda en la memoria RAM, en la variable “value”.

La segunda función, “digitalWrite(Led_Blanco,Valor);”, escribe sobre la patilla 6 (“Led_Blanco”), el contenido de la variable “value”, el que se leyó en la función previa. Recuerda que esa patilla, la 6, previamente se había configurado como salida.

Todas las funciones que forman el cuerpo principal del programa se ejecutan de forma repetida, de la primera a la última, e indefinidamente, a la máxima velocidad que pueda ofrecer el controlador.

Importante: Observa que al declarar las variables, así como todas las funciones de configuración y del programa principal, finalizan SIEMPRE con ";"

E. ZONA DE MENSAJES

En esta última zona, el entorno IDE de trabado de Arduino te irá mostrando diferentes mensajes: si estás guardando un programa, o compilándolo, o grabándolo sobre la memoria del controlador. También te informa si ha habido errores de compilación, tipo de error y en dónde se ha producido.

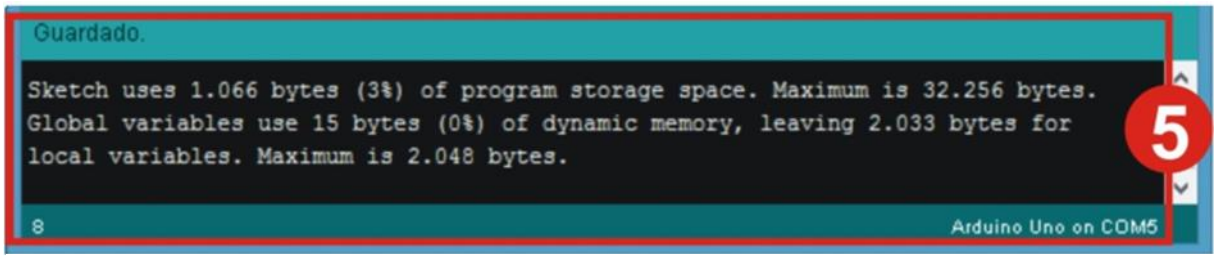


Figure 5

En este caso todo ha ido perfectamente. Nos informa de que el programa ocupa un total de 1066 bytes de la memoria FLASH de programa, de los 32256 disponibles. Se han consumido 15 bytes de la memoria RAM para datos (por las variables que hemos creado) de un total de 2048. Quedan por lo tanto disponibles 2033 bytes.

Recuerda: "Compilar" un programa es traducir el programa que has escrito empleando el lenguaje Arduino de alto nivel, a su equivalente en códigos binarios (código máquina) que es lo que realmente se graba en la memoria FLASH del controlador. Es un proceso automático y transparente para ti.

2. SENTENCIAS EN UN PROGRAMA

Ya has visto, al menos teóricamente, el aspecto de un programa. Habrás comprobado que está compuesto de varias instrucciones o sentencias también llamadas "FUNCIONES". Un programa puede estar compuesto de miles de funciones.

Cada lenguaje de programación tiene sus propias funciones con sus propias sintaxis y sus propias reglas de programación. Vamos a estudiar el lenguaje Arduino de programación, del cual ya hemos empleado cinco instrucciones o funciones diferentes. Vas a verlas ahora con más detalle.

A. FUNCIÓN: SETUP()

Esta función y todas las funciones contenidas en ella, se ejecutan al reiniciar el sistema. Esto ocurre cada vez que pulsas el botón RESET o bien conectas la alimentación.

Dentro de esta función se suelen colocar las funciones para la configuración de las patillas de entrada o de salida, inicio de ciertas variables, librerías, etc… En general, dentro de la función setup() puedes colocar las sentencias o funciones que quieras, siempre que estén encerradas entre las llaves

"{ … }". Incluso puedes poner una función setup() vacía, sin funciones entre sus llaves, pero debes de ponerla obligatoriamente.

Lo importante que debes recordar es que todas las funciones que contiene la función setup() sólo se ejecutan una única vez, en el reinicio.This function and all the other functions contained in it are executed when the system is reset. This happens every time you push the RESET button or power up the system.

Sintaxis:

Void setup()

{

.....}

Ejemplo

Void setup()

{

pinMode(INPUT, Button); *// La patilla del pulsador se configura como entrada*

pinMode(White_LED,OUTPUT); *// La patilla del led se configura como salida*

}

B. FUNCIÓN: LOOP()

Aquí, dentro de esta función y cerradas entre las llaves "{ … }", es donde vas a colocar todas tus instrucciones y funciones. Se dice que contiene el "cuerpo principal" del programa.

Sintaxis:

void loop()

{

.....

}

Ejemplo:

void loop()

{

Valor=digitalRead(Pulsador); *//Lee el estado del pulsador*

digitalWrite(Led_Blanco,Valor); *//Lo saca por el led*

}

Un programa puede estar compuesto por decenas, cientos o miles de funciones que se incluirán dentro de la función loop() principal. En el ejemplo anterior sólo dos. Incluso puede darse el caso de que la función loop() esté vacía, sin funciones encerradas entre sus llaves ({ … }). Es igual, es obligatorio que la pongas siempre.

Todas las funciones incluidas entre las llaves de la función loop() se ejecutan una tras otra, desde la primera hasta la última, de forma secuencial e indefinida, hasta que desconectes el sistema.

C. THE PINMODE() FUNCTION

Configura cualquiera de las patillas digitales del controlador Arduino para que actúe como entrada o como salida. Normalmente se suele poner al principio de un programa, incluida en la función setup().

Sintaxis:

pinMode(patilla,modo);

patilla: Expresa el número de patilla que vamos a configurar. En el caso de Arduino UNO puede ser entre la 0 y la 13.

modo: Establece si se comporta como entrada (INPUT) o salida (OUTPUT).

Ejemplo:

```
int Pulsador=4;
int Led_Blanco=6;
pinMode(Led_Blanco,OUTPUT);        //La patilla 6 es salida
pinMode(9,OUTPUT);                 //La patilla 9 es salida
pinMode(Pulsador,INPUT);           //La patilla 4 es entrada
pinMode(12,INPUT);                 //La patilla 12 es entrada
```

Por defecto, cada vez que se reinicia el sistema, todas las patillas digitales quedan configuradas como entradas.

D. THE DIGITALREAD() FUNCTION

Lee y devuelve el estado lógico binario (“1” o “0”, “HIGH” o “LOW”) de cualquiera de las patillas del controlador Arduino.

Sintaxis:

digitalRead(pin);

pin:. Expresa el número de patilla que vamos a leer. En el caso de Arduino UNO puede ser entre la 0 y la 13.

Ejemplos

int Pulsador=4;

Valor=digitalRead(Pulsador); *//Lee el estado de la patilla 4 y lo guarda en "value"*

X=digitalRead(8); *//Lee el estado de la patilla 8 y lo guarda en "X"*

E. FUNCIÓN: DIGITALWRITE()

Escribe o saca por una patilla de salida un valor binario ("1" o "0", "HIGH" o "LOW").

Sintaxis:

digitalWrite(pin, value);

pin: Expresa el número de patilla por la que vamos a sacar el valor. En el caso de Arduino UNO puede ser entre la 0 y la 13.

Value: Indica el valor a sacar ("1" o "0", "HIGH" o "LOW").

Ejemplos

int White_LED=6;

DigitalWrite(White_LED,Value); *// Saca el contenido de "value" ("1" o "0") por la patilla 6*

DigitalWrite(11, LOW); *// Saca nivel "0" por la patilla 11 de salida*

DigitalWrite(7,1); *// Saca nivel "1" por la patilla 7 de salida*

3. LOGICAL OPERATORS

Llegados a este punto no puedo evitar hablarte de ciertos operadores, concretamente tres, capaces de relacionar entre sí a diferentes funciones, operaciones, expresiones, etc… Se llaman "operadores lógicos" y se emplean mucho en los sistemas digitales. Espero explicártelos de la forma más sencilla posible, sin entrar en tecnicismos.

A. THE NOT OPERATOR

Expresa una negación (NO o NOT) y se simboliza mediante el signo de admiración "!". Lo entenderás mejor con un ejemplo. Mira esta función que ya te resultará conocida:

Value=digitalRead(Button);

La variable “value” será de nivel “1” si al leer “Pulsador” éste se encuentra también a nivel “1”. En caso contrario “value” será de nivel “0”. Ahora bien, fíjate en la siguiente función:

Value = ! digitalRead(Button);

La variable “value” será de nivel “1” si al leer “Pulsador” éste NO se encuentra a nivel “1”, es decir está a nivel “0”. En caso contrario, si el pulsador se encuentra a nivel“1”, la variable “value” será de nivel “0”. Mira el ejemplo y responde.

Value = ! digitalRead(12);

“*Value*” es de nivel “1” o “*true*” si: _____

“*Value*” es nivel “0” o “*false*” si: _____

B. OPERADOR Y (AND)

Esta operación genera un nivel “1”, también llamado “verdadero”, si TODOS los elementos que relacionas están también a nivel “1” o son verdaderos. Se representa mediante los signos “&&”. Mira el ejemplo:

Value =digitalRead(4) && digitalRead(12);

La variable “value” será de nivel “1”, “verdadero”, si al leer las entradas 4 Y 12 ambas están también a nivel “1”. Si cualquiera de las entradas está a nivel “0”, el resultado en la variable “value” también será de nivel “0” o “falso”. Mira el ejemplo siguiente y responde.

Value =digitalRead(4) && digitalRead(8) && digitalRead(12);

“*Value*” es de nivel “1” o “*true*” si: _____

“*Value*” es de nivel “0” o “*false*” si: _____

C. OPERADOR O (OR)

Esta operación genera un nivel “1”, también llamado “verdadero”, si CUALQUIERA de los elementos que relacionas están también a nivel “1” o son verdaderos. Se representa mediante los signos “||”.

Value =digitalRead(4) || digitalRead(12);

La variable “value” será de nivel “1”, “verdadero”, si las entradas 4 O 12 O ambas están a nivel “1”. Si toda la entrada está a nivel “0”, el resultado en la variable “value” también será de nivel “0” o “falso”. Mira el ejemplo siguiente y responde.

Value =digitalRead(4) || digitalRead(8) || digitalRead(12);

“*Value*” equals level “1” or “*true*” if: _____

“*Value*” equals level “0” or “*false*” if: _____

D. COMBINANDO OPERADORES

Por supuesto que también es posible combinar diferentes operadores lógicos en una misma función. Debes emplear los paréntesis para establecer el orden en que se deben evaluar y calcular.

Value = ! digitalRead(4) && **digitalRead(7);**

La variable “value” será de nivel “1” o verdadera si la entrada 4 NO está a “1” Y la entrada 7 si lo está. Mira el ejemplo siguiente y responde.

Value = (digitalRead(8) && **! digitalRead(12)) || digitalRead(4));**

“*Value*” equals level “1” or “*true*” if: _____

“*Value*” equals level “0” or “*false*” if: _____

PRÁCTICA

1. EJEMPLO 1: Encendido de un led v.1

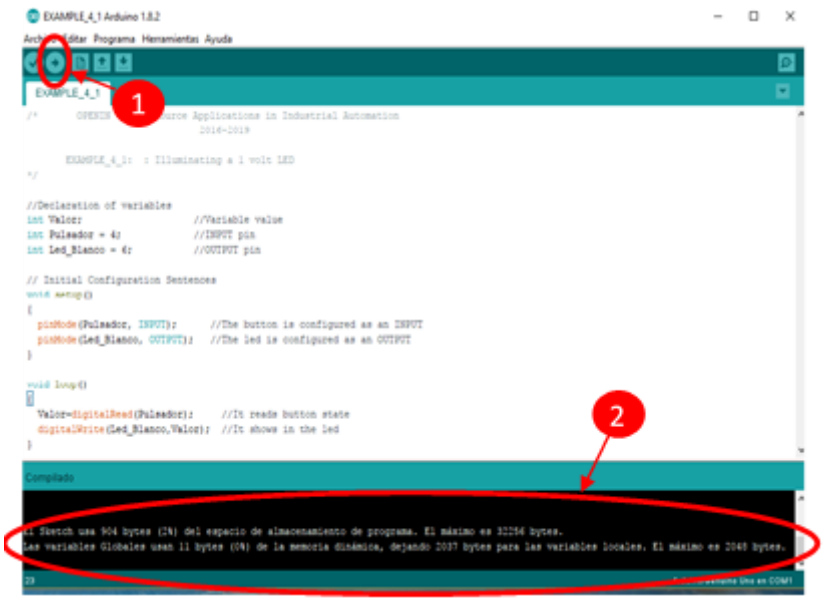
Este es el mismo ejemplo que el que has estudiado en el área de teoría, por lo que no te voy a aburrir repitiendo su funcionamiento. Simplemente recordarte que consiste en leer el estado del pulsador conectado en la patilla 4, y reflejar su valor en el led blanco, conectado en la patilla 6 del controlador.

- **Compilación y grabación**

Como se muestra en la Figure 6, basta con pulsar el botón marcado con 1. En un proceso bastante rápido y totalmente automático, el entorno IDE de Arduino se encargará de compilar tu programa traduciéndolo a código máquina o código binario. Para ti no tiene importancia el cómo lo hace.

Si el programa está bien escrito y no tiene errores, verás unos mensajes como los que se muestran en la ventana marcada con 2. Ya los conoces. Indican el tamaño que ocupará el programa dentro de la memoria FLASH del controlador, así como la cantidad de memoria RAM consumida por las variables que has empleado. Además, si tu programa está correctamente escrito, inmediatamente se grabará sobre la memoria FLASH del controlador.

En esa misma ventana también te aparecerán, en caso de que los hubiera, los diferentes tipos de errores y en qué lugar del programa se han producido. Una vez que los corrijas vuelves a compilar el programa.



- **Comprobación**

Figure 6

Aunque al compilar un programa, el entorno IDE de Arduino no te informe de errores, no quiere decir que el programa funcione correctamente. El entorno sólo puede detectar errores de tipo sintáctico: una función mal escrita o inexistente, emplear variables que no se han declarado previamente, usar datos incorrectos, etc...

La prueba de fuego consiste en comprobar que el controlador hace exactamente aquello para lo que le has programado. Este ejemplo es muy sencillo, basta con que acciones el pulsador D4 para observar que el led blanco conectado en la patilla D6 se ilumina. En caso contrario se apaga.

Una última prueba. Cuando grabas la memoria FLASH con un programa, éste se mantiene indefinidamente aunque desconectes la alimentación. Un programa sólo se borrará cuando grabes otro nuevo. Otro detalle. Una vez grabado un programa, tarjeta Arduino es totalmente independiente y autónoma respecto al PC. Todo esto lo puedes comprobar de forma muy sencilla. Retira el cable USB de la tarjeta Arduino UNO y ahora la alimentas con un alimentador de red o una pila como se muestra en la figura. Verás que el funcionamiento sigue siendo el mismo.

Quizá pienses que no has hecho una gran cosa, que para encender un led con un pulsador no hace falta un controlador. Te equivocas. Has dado el primer paso. Si has comprendido el concepto de lo que es un programa y sus funciones, te aseguro de que podrás llegar a hacer cosas mucho más complicadas e interesantes. Tiempo al tiempo.

2. EJEMPLO 2: Encendido de un led v.2

Aquí tienes una nueva versión del mismo programa anterior. Controla el led blanco conectado con la patilla 6, mediante el pulsador conectado en la patilla 4 de la misma tarjeta. Observa el programa de la figura y compáralo con el del ejemplo anterior.

La única diferencia es que no hemos empleado variables para definir en qué patillas están conectados el pulsador y el led blanco. En las propias funciones pinMode(), digitalRead() y digitalWrite() se han indicado directamente las patillas a la que hacen referencia (4 y 6).

Se dice que en este caso el número de las patillas se expresan como “constantes”, y no como “variables”. Como hiciste con el ejemplo anterior, compila y graba el programa. Luego verifica que funciona correctamente.

- **Conclusiones**

El empleo de variables o constantes para definir una patilla del controlador depende fundamentalmente de ti. Por ejemplo, si el pulsador va a estar permanentemente conectado en la patilla 4, emplea este número como constante, como lo acabamos de hacer en este ejemplo. Ahorrarás en memoria RAM. Ahora bien, si ese pulsador (o cualquier otro periférico) puede estar sujeto a posibles cambios en su conexión con las patillas del controlador, mejor defines esa conexión como una variable, por ejemplo: “int Pulsador=4;”. De esta forma si en un futuro la conexión del pulsador pasa a ser, por ejemplo, con la patilla 12, sólo tendrás que cambiar la declaración de la variable: “int Pulsador=12;”. No tendrás que modificar las funciones del programa que hacen uso de ese pulsador.

3. EJEMPLO 3: Encendido de un led v.3

Una nueva (y última) versión del programa para controlar un led mediante un pulsador. En primer lugar, compila el programa y grábalo en el controlador. Comprueba su correcto funcionamiento. También puedes comprobar que si desconectas el cable USB y alimentas al conjunto de tarjeta Arduino de forma independiente y autónoma respecto al PC, con una pila, todo debiera seguir funcionando correctamente. Entonces … ¿dónde están las diferencias?

- **Conclusiones**

Si estudias con cuidado el programa verás dónde se encuentran las diferencias entre este ejemplo y los anteriores EJEMPLO 1 y EJEMPLO 2. De ello podemos sacar las siguientes conclusiones:

✓ No se ha empleado ninguna variable, con el consiguiente ahorro de memoria RAM. En un gran número de ocasiones, el empleo de variables y constantes depende más bien de lo que a ti te interese.

✓ Las patillas que vayan a actuar como entradas digitales no necesitan ser configuradas mediante pinMode(). Todas las patillas están configuradas como entrada por defecto.

✓ El resultado de una función puede usarse como entrada de otra función sin pasar por la variable intermedia “Valor” como en los otros ejemplos. Observa la siguiente expresión:

digitalWrite(6,digitalRead(4));

Hay una función dentro de otra. Siempre se ejecuta primero la función más interna: digitalRead(4). El resultado de ésta se pasa a la siguiente función digitalWrite(6,…);

✓ Un mismo programa se puede resolver de diferentes formas. De momento te debes ocupar de que funcione correctamente. Es lo más importante. Con la práctica irás consiguiendo hacer que el mismo programa consuma menos memoria FLASH de programa y también menos memoria RAM de datos, haciéndolo por tanto más “EFICIENTE”.

Tú mismo puedes hacer una sencilla comprobación. Empieza de nuevo por compilar y grabar EJEMPLO 1, EJEMPLO 2 y EJEMPLO 3. En cada caso anota la cantidad de memoria de programa (FLASH) y de memoria de variables (RAM) que consumes. Esta información te la da el propio entorno IDE de Arduino, en la parte de abajo, cada vez que se termina de grabar un programa. Completa la siguiente tabla.

EJEMPLO	FLASH Program Memory	RAM Data Memory
EJEMPLO 1		
EJEMPLO 2		
EJEMPLO 3		

4. EJEMPLO 4: Encendido invertido de un led

Empleando el operador lógico NO, se trata de hacer un programa en el que el led blanco, conectado con la patilla 6, se active siempre que el pulsador conectado con la patilla 4, NO esté activado. Aquí tienes el programa.

En amarillo se ha resaltado el cuerpo principal del programa contenido en la función loop(). Es muy similar al del ejemplo anterior. La única diferencia es que la lectura del pulsador conectado con la patilla 4 se invierte mediante la función NO, símbolo "!".

Este ejemplo Figure 7 podría simular una máquina cuyo motor está siempre en marcha excepto si se acciona un pulsador de emergencia.The idea is to use the NOT logic operator and make a program to illuminate the white LED connected to pin 6 whenever the button connected to pin 4 is NOT on.

Figure 7

5. EJEMPLO 5: Control desde dos pulsadores

Imagina que deseas controlar una alarma desde dos pulsadores que están alejados entre sí. La alarma debe activarse cuando se acciona un pulsador O el otro (O ambos). La alarma se simula mediante el led rojo, conectado en la patilla 11. Los dos pulsadores se encuentran conectados en las patillas 4 y 12. En Figure 8 tienes el programa resuelto. Estúdialo.

Figure 8

6. EJEMPLO 6: Trabajo personal

Emplea el EJEMPLO 6 como programa base o “plantilla”. Sobre él vas a hacer las modificaciones necesarias para resolver los diferentes casos que te voy a proponer. Espero que emplees todo lo que hemos visto hasta el momento, aprendas y, sobre todo, te diviertas. Te darás cuenta de que a medida que vamos avanzando en la programación, el controlador es capaz de hacer mucho más que encender un simple led, y ¡¡ no has hecho más que empezar!!

Exercise	Description
1	El led ámbar se debe iluminar cuando los pulsadores 7 Y 12 están activados.
2	El led verde se ilumina si los pulsadores 12 y 8 están activados y el 4 no lo esté.
3	El led blanco se ilumina si el pulsador 4 está activado y el 7 no. El led rojo se ilumina si se activa el pulsador 8 o el pulsador 12.
4	El led verde se ilumina si el pulsador 4 está activado, y el 7 no, y el 8 sí y el 12 no.
5	El led rojo se enciende cuando los pulsadores 4 y 7 y 8 estén activados, o bien lo esté el pulsador 12.

Recuerda. Cada modificación que hagas debes escribirla en el entorno IDE de Arduino, garbarla en el controlador y asegurarte de que funciona según las especificaciones de cada caso.

Completa la siguiente tabla. En la columna setup() anotas todas las sentencias que hayas usado en la configuración. Lo mismo haces con la columna loop(). Escribes todas las funciones que formen parte del programa principal.

Exercise	Function	Your program’s instructions
1	setup()	
	loop()	
2	setup()	
	loop()	
3	setup()	
	loop()	
4	setup()	
	loop()	

5	**setup()**	
	loop()	

REFERENCIAS

BOOKS

[1]. **EXPLORING ARDUINO**, Jeremy Blum, Ed.Willey
[2]. Practical ARDUINO, Jonathan Oxer & Hugh Blemings, Ed.Technology in action
[3]. Programing Arduino, Next Steps, Simon Monk
[4]. Sensores y actuadores, Aplicaciones con ARDUINO, Leonel G.Corona Ramirez, Griselda S. Abarca Jiménez, Jesús Mares Carreño, Ed.Patria
[5]. ARDUINO: Curso práctico de formación, Oscar Torrente Artero, Ed.RC libros
[6]. 30 ARDUINO PROJECTS FOR THE EVIL GENIUS, Simon Monk, Ed. TAB
[7]. Beginning C for ARDUINO, Jack Purdum, Ph.D., Ed.Technology in action
[8]. ARDUINO programing notebook, Brian W.Evans

WEB SITES

[1]. https://www.arduino.cc/
[2]. https://www.prometec.net/
[3]. http://blog.bricogeek.com/
[4]. https://aprendiendoarduino.wordpress.com/
[5]. https://www.sparkfun.com