Universidad Francisco de Vitoria UFV Madrid

Fundamentos de la Programación con Arduino

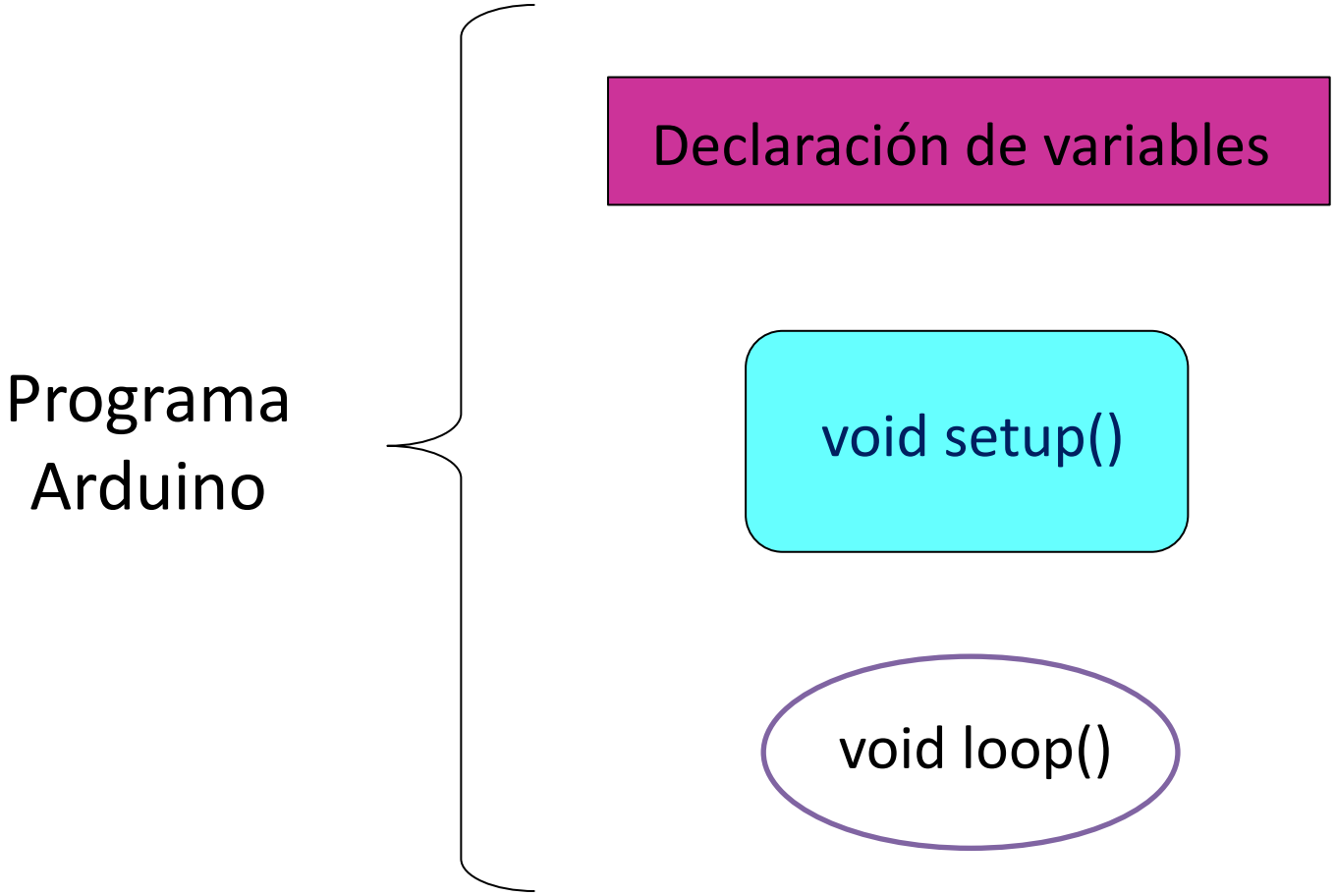
Tema 3 – Lenguaje de Programación Arduino

Universidad Francisco de Vitoria UFV Madrid

Grado en Ingeniería Informática
Escuela Politécnica Superior

Estructura de un programa

Universidad Francisco de Vitoria UFV Madrid



Estructura de un programa

Universidad Francisco de Vitoria UFV Madrid

- La estructura de un programa escrito en el lenguaje de programación Arduino se compone de al menos dos funciones. Estas funciones encierran bloques que contienen declaraciones o instrucciones.

```
void setup()
{
   instrucciones;
}
void loop()
{
   instrucciones;
}
```

La función setup()

Universidad Francisco de Vitoria UFV Madrid

- En la función de configuración o setup() nos encargaremos de establecer la configuración de la placa Arduino.
- Es la primera función a ejecutar y se ejecuta una sóla vez cuando el programa comienza su ejecución.
- La utilizamos para inicializar los pines como entrada o salida, así como el funcionamiento del puerto serie.
- **Aunque el bloque se encuentre vacío debe ser incluido en el programa.**

```
void setup()
 {
    pinMode(pin, OUTPUT); //configura 'pin' como salida
 }
```

La función loop()

Universidad Francisco de Vitoria UFV Madrid

- La función loop() es la que contienen el programa que se ejecutará cíclicamente.
- Es el núcleo de todos los programas de Arduino y la que realiza la mayor parte del trabajo.

```
void loop()
{
   digitalWrite(pin, HIGH); // pone a ‘pin’ en 1 (on, 5v)
   delay(1000); // espera un segundo (1000 ms)
   digitalWrite(pin, LOW); // pone a ‘pin’ 0 (off, 0v.)
   delay(1000); espera un segundo (1000 ms)
}
```

- Una vez que finaliza el bucle, vuelve a comenzar, de forma cíclica.

Variables

Universidad Francisco de Vitoria UFV Madrid

- Una variable es una manera de nombrar y almacenar un valor para su uso posterior por el programa.
- Las variables son espacios en memoria cuyo contenido puede cambiar, al contrario de lo que ocurre con las constantes, cuyo valor nunca cambia.
- Una variable debe ser declarada y, opcionalmente, inicializarla con un valor.
- Las variables deben tomar nombres descriptivos, para hacer el código más legible.
- Una variable puede ser cualquier nombre o palabra que no sea una palabra reservada en las instrucciones de Arduino.

Tipos de datos

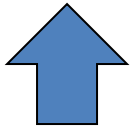
Universidad Francisco de Vitoria UFV Madrid

Tipo	Tamaño	Rango	Descripción
boolean	1 bit	0 a 1	Entero de 1 bit
byte	8 bit	0 a 255	Entero
unsigned char	8 bit	0 a 255	Entro sin signo
char	8 bit	-128 a 127	Entero con signo
int	8 bit	-32,768 a 32,767	Entero de 8 bit con signo
unsigned int	16 bit	0 - 65,535	Entero de 16 bit
long	4 byte	-2,147'483,648 a 2,147'483,647	Entero de 16 bit con signo
unsigned long	4 byte	0 a 4,294'967,295	Entero sin signo de 4 bits
float / double	4 byte	-3.4028235e+37 a 3.4028235e+38	Decimal con signo de 4 bytes

Constantes

Universidad Francisco de Vitoria UFV Madrid

- Existen constantes propias del lenguaje Arduino:
  - **HIGH | LOW** – Al leer o escribir en un pin solo hay estas dos posibilidades. Su significado es diferente si el pin está configurado como INPUT o OUTPUT.
  - **INPUT | OUTPUT | INPUT_PULLUP** – Modo en el que pueden configurarse los pines digitales.
  - **LED_BUILTIN** – La mayoría de las placas Arduino tienen un pin conectado a un led en la placa y esta constante devuelve el número de pin en función de la placa.
  - **true | false** – Representación de las constantes booleanes en arduino
- Podemos definir nuevas constantes con la instrucción **const**
  - `const float pi = 3,14159;`



Entradas/Salidas digitales

Universidad Francisco de Vitoria UFV Madrid

- Mediante las entradas / salidas digitales podemos controlar dos valores o estados: abierto/cerrado, encendido/apagado, etc.
- En sistemas digitales, utilizamos dicha lógica de dos estados y nos referiremos a ellos como high (H) o low (L)

Entradas/Salidas digitales

Universidad Francisco de Vitoria UFV Madrid

- En el lenguaje Arduino para tratar las entradas y salidas digitales usamos las siguientes funciones:
  - **pinMode(pin, modo)** – configura el pin especificado para que se comporte como entrada o como salida. Lo utilizamos en la función setup()
    - pinMode(8, OUTPUT)
  - **digitalWrite(pin, valor)** – Escribe un valor HIGH o LOW en el pin digital especificado.
    - digitalWrite(8, HIGH)
  - **digitalRead(pin)** – lee el valor del pin correspondiente como HIGH o LOW.
  - Tanto digitalRead como digitalWrite lo utilizaremos en la función loop()

Entradas/salidas analógicas

Universidad Francisco de Vitoria UFV Madrid

- Los pines analógicos de la placa arduino son exclusivamente de entrada.
- Para leer el valor recibido por un pin analógico utilizaremos la función:
  - `analogRead(pin) // Rango 0 - 1023`
- Podemos almacenarlo en una variable de la siguiente forma:
  - `valor = analogRead(pin);`

Entradas/salidas analógicas

Universidad Francisco de Vitoria UFV Madrid

- Para poder enviar un valor ‘analógico’ utilizaremos los pines indicados en la placa como PWM.
- Si enviamos el valor 0 generaremos una salida de 0 voltios en el pin especificado; un valor de 255 genera una salida de 5 voltios de salida en el pin especificado. Para valores de entre 0 y 255, el pin enviará una tensión entre 0 y 5 voltios
  - `analogWrite(pin, valor); // escribe 'valor' en el 'pin' definido como analógico`

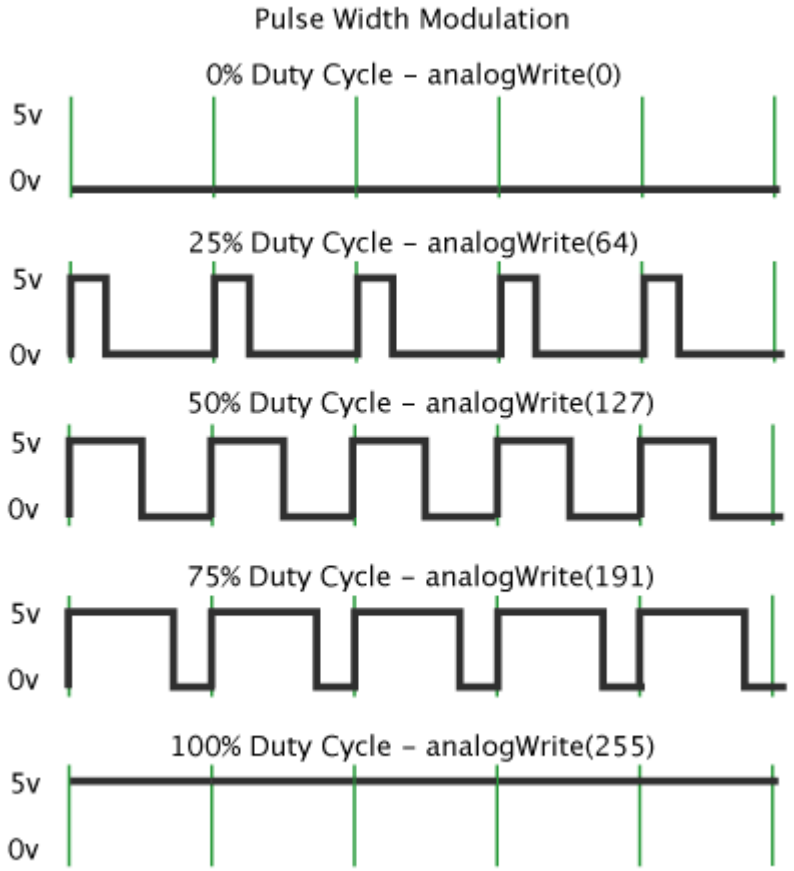
Entradas/salidas analógicas

Universidad Francisco de Vitoria UFV Madrid

64 es aprox.= 25% de 255 → 25% de 5V = 1,25 V

127 es aprox.= 50% de 255 → 50% de 5V = 2,5 V

192 es aprox.= 75% de 255 → 75% de 5V = 3,75 V



Funciones de Tiempo

Universidad Francisco de Vitoria UFV Madrid

- **delay(ms)**: pausa la ejecución del programa durante el tiempo indicado como parámetro (en milisegundos)
  - `delay(1000); // espera 1 segundo`
- **delayMicroseconds(µs):** similar a la función anterior pero utilizando microsegundos
- **millis(ms)**: devuelve el número de milisegundos que han transcurrido desde que la placa Arduino comenzó la ejecución del programa actual (volverá a cero en aproximadamente 50 días). El tipo de dato devuelto es un `unsigned long`
  - `tiempo = millis();`

Puerto serie

Universidad Francisco de Vitoria UFV Madrid

- **Serial.begin(ratio)**: abre el puerto serie y fija la velocidad (en baudios) para la transmisión de datos en serie. El valor típico de velocidad para comunicarse con el ordenador es 9600, aunque pueden ser usadas otras velocidades.

```
void setup() {
    Serial.begin(9600); // abre el Puerto serie
}
```

- **Serial.println(dato)**: Imprime los datos en el puerto serie, seguido por un retorno de carro automático y salto de línea

```
Serial.println("Hola");
```

- **Serial.print(dato)**: similar a la función anterior, pero no realiza salto de carro y línea.

El resultado podemos verlo en el monitor serie, tanto en el IDE de Arduino como en el simulador Tinkercad.

Función map()

Universidad Francisco de Vitoria UFV Madrid

- La función map(), que se define de la siguiente forma:
  - `map(dato, origen_min, origen_max, destino_min, destino_max)`

  nos permite realizar equivalencias entre rangos.

  Por ejemplo, dada una variable de nombre *value* que puede tener un valor comprendido entre 0 y 1023, queremos un valor equivalente en el rango 0 a 255:

  - `nuevovalor = map(value, 0, 1023, 0, 255);`