Capítulo 5
La programación en Arduino

1. Estructura de un programa

Como se ha visto en el capítulo anterior, el lenguaje utilizado con Arduino es muy próximo al lenguaje C o C++. Es el caso fundamentalmente de la sintaxis y los tipos de variables. Sin embargo, este lenguaje presenta algunas diferencias, fundamentalmente en la estructura de un programa.

Un programa Arduino se descompone en dos partes principales: una primera parte llamada `setup`, y una segunda llamada `loop`. Estas dos partes son representativas de la manera de funcionar de un microcontrolador y se invocan en este orden. De hecho, un microcontrolador está destinado a ejecutar las instrucciones contenidas en su memoria de manera cíclica e infinita. Antes de ejecutarse, el microcontrolador necesita una fase de « configuración » de su entorno, principalmente para parametrizar los conectores de entrada/salida o simplemente para fijar las variables globales. Esta fase corresponde a la función `setup`.

La función `setup` se presenta de la siguiente manera:

```
void setup(){
    [Instrucciones];
}
```

Las instrucciones contenidas en esta función solo se ejecutan una vez y ocurre cuando la tarjeta se enchufa a la corriente.

Observación

Se aconseja reinicializar la tarjeta cuando se enchufa a la corriente. La función `setup` también se ejecuta después de un reset.

Una vez hecha la configuración, el microcontrolador inicia la ejecución de las instrucciones contenidas en su memoria. Esta fase corresponde a la función `loop`.

La función `loop` se presenta de la siguiente manera:

```
void loop(){
    [Instrucciones];
}
```

Esta función se parece a un bucle que no tiene ninguna condición de parada. El único medio de detenerlo es cortar la alimentación de la tarjeta o reinicializar la tarjeta. Por tanto, las instrucciones contenidas en esta función se repiten indefinidamente todo el tiempo que la tarjeta está conectada a la corriente.

Es obligatorio que estas dos funciones estén presentes en el programa principal. El orden de aparición de estas funciones en el programa no tiene importancia. Sin embargo, es aconsejable mantener la declaración de estas funciones en este orden y evitar separarlas. Esto permite hacer más legible el código y, sobre todo, permite a otro lector localizar más rápidamente el arranque de la función principal.

También es posible declarar las funciones en el mismo archivo en el que está presente el programa Arduino. Estas declaraciones de funciones se podrían insertar en cualquier lugar del programa, mientras se respete la sintaxis. Para terminar es posible, como en los programas en C o C++, incluir librerías de funciones en su programa. Esta inclusión se deberá realizar, si es necesario, antes de la declaración de cualquier función.

Es importante estructurar eficazmente su programa para permitir una mejor relectura y, sobre todo, reparar rápidamente eventuales errores. Un programa Arduino « eficaz » se podría realizar de la siguiente manera:

```
-Inclusión o no de las librerías utilizadas-

-Declaración de las funciones utilizadas-
void setup(){
    [Instrucciones ejecutadas una sola vez cuando se enchufa la tarjeta];
```

© Editions ENI - All rights reserved

```
}

void loop(){
     [Instrucciones ejecutadas en bucle por el microcontrolador];
}
```

2. Variables y constantes específicas

Las variables y constantes utilizas por el lenguaje Arduino también comparten esta similitud con el lenguaje C y C++. Por tanto, todas las variables presentadas en el capítulo anterior son aplicables a la programación en Arduino. Esto también vale para las constantes. Sin embargo, el lenguaje Arduino presenta algunas constantes que le son propias. Estas últimas no están en los demás lenguajes y solo son aplicables a la programación en Arduino.

Estas constantes específicas se utilizan principalmente, para controlar los conectores de entrada/salida de la tarjeta Arduino. De manera más concreta, se trata de constantes que permiten configurar sus modos de funcionamiento.

HIGH y LOW

Estas dos constantes se usan con los conectores digitales. Como hemos visto anteriormente (consulte el capítulo El módulo Arduino, sección Entradas/salidas), los conectores digitales manipulan los datos digitales en forma de señales lógicas. Los dos valores representables se definen por estas constantes. En la práctica, corresponden a los valores de tensión que se utilizarán por los conectores de Arduino. Estas constantes tienen propiedades particulares que dependen del modo de funcionamiento que le hemos asignado a los conectores digitales.

Cuando estos conectores están en modo lectura, la constante `HIGH` corresponde a un valor de tensión superior a 3V. La constante `LOW` corresponde a un valor de tensión inferior a 1,5V. Esto significa que hay que proporcionar los valores de tensión comprendidos entre estos márgenes para que la lectura de estos conectores se pueda entender. No todos los valores de tensión comprendidos entre estos dos valores se pueden interpretar por el microcontrolador. Por tanto, es obligatorio que la tensión no esté comprendida entre estos dos valores si queremos que no haya error en el programa.

Cuando los conectores están en modo escritura, la constante `HIGH` indica al microcontrolador que debe proporcionar a los conectores implicados una tensión de 5V. El microcontrolador proporciona, de manera inversa, una tensión de 0V cuando se usa la constante `LOW`.

OUTPUT y INPUT

Estas dos constantes sirven para determinar si los conectores a los que se aplican funcionan en modo lectura o escritura. Como solo los conectores digitales pueden funcionar en estos dos modos (consulte el capítulo El módulo Arduino, sección Entradas/salidas), estas dos constantes solo se aplican para ellos.

Cuando un conector se declara en modo `INPUT`, es sitúa en un estado llamado de alta impedancia. En este estado, los conectores no consumen casi corriente en los circuitos a los que están conectados. Este estado corresponde al modo de entrada. También es el modo de funcionamiento por defecto de todos los conectores digitales.

Cuando un conector se declara en `OUTPUT`, puede proporcionar o absorber hasta 40 mA de corriente. Se trata del modo de salida de los conectores digitales. En este modo, los conectores generan suficiente corriente para alimentar los LED o pequeños captadores. Recordemos sin embargo, que incuso si todos los conectores pueden proporcionar 40 mA de corriente, existe un límite total acumulado de corriente de 200 mA aproximadamente, en relación a lo que una tarjeta Arduino puede proporcionar.

3. Funciones propias de Arduino

También existe un número de funciones específicas al uso en Arduino. Estas funciones son directamente accesibles en el entorno de desarrollo Arduino, sin tener que utilizar nuevas librerías.

© Editions ENI - All rights reserved

3.1 Entradas/salidas

Las funciones de entrada/salida permiten utilizar los conectores presentes en las tarjetas Arduino. Podemos distinguir dos tipos de funciones, dado que existen dos tipos de conectores. De hecho, los conectores analógicos y digitales no se controlan de la misma manera y existen funciones específicas para estos dos casos.

Conectores digitales

Las siguientes funciones permiten manipular los conectores digitales de las tarjetas Arduino.

– `pinMode()`

  Los conectores digitales pueden funcionar en modo lectura o escritura. Esta función permite indicar este modo de funcionamiento. Idealmente se debe llamar durante la función `setup()` del programa principal.

  ```
  pinMode(número de conector, modo de funcionamiento);
  ```

  El conector que se quiere configurar se debe indicar en los argumentos de la función como un número. Este último se representa por el número que está grabado en la tarjeta. En las tarjetas Arduino Uno, este número va de 0 a 13. Es importante observar que los conectores analógicos también pueden servir de entrada/salida digital. Los números a utilizar en este caso, para una tarjeta Arduino Uno por ejemplo, van de A0 a A5. Para terminar, el modo de funcionamiento se declara utilizando las dos constantes `INPUT` y `OUTPUT`, definidas en la parte anterior. Por defecto, todos los conectores están configurados en modo INPUT. No es técnicamente necesario hacerlo si queremos utilizar un conector en modo lectura, pero es muy aconsejable realizar sistemáticamente este trabajo de configuración para facilitar la legibilidad del programa.

– `digitalWrite()`

  Ahora, la escritura en los conectores digitales es la siguiente etapa. Para esto, la función `digitalWrite()` entra en juego. Un conector configurado en modo escritura debe enviar un valor y esta función indica qué valor utilizar.

  ```
  digitalWrite(número del conector, valor a enviar);
  ```

El número del conector, como en la función anterior, se debe indicar en los argumentos como un número. Después, el valor a enviar corresponde a las dos constantes `HIGH` y `LOW` introducidas en la parte anterior.

– `digitalRead()`

La lectura se realiza por la función `digitalRead()`. Esta función solo recibe como argumento el número del conector sobre el que se desea leer. El valor devuelto solo puede ser una de las dos constantes `HIGH` y `LOW`. Esta constante devuelta, incluso si no se indica en ningún sitio, es de un tipo digital. De esta manera es posible almacenar este valor en una variable local en su programa, para que esta variable sea del tipo correcto.

```
byte valor;
valor = digitalRead(número del conector);
```

Conectores analógicos

Los conectores analógicos tienen un funcionamiento diferente del de su contraparte digital. Por su naturaleza, no se deben poder manipular más de dos valores. De esta manera, los microcontroladores que equipan Arduino disponen de un número de convertidores analógico/digital (6 para Arduino Uno y 16 para Arduino Mega). La función de estos convertidores es traducir una cantidad analógica, en nuestro caso una tensión eléctrica, en un valor digital, que será posible manipular en un programa Arduino. Los convertidores presentes en los microcontroladores utilizados por las tarjetas Arduino pueden manipular valores hasta 10 bits, es decir, números comprendidos entre 0 y 1023. De esta manera, el número 0 corresponde a un valor de tensión nulo o 0V, y el número 1023 corresponde a un valor máximo.

– `analogReference()`

La lectura de un valor analógico se realiza entre dos valores de tensión eléctrica, un valor nulo y un valor máximo. Par defecto, este valor máximo se fija a 5V. Durante una lectura analógica, el número 1023 corresponde con esta tensión umbral. Esto confiere a la tarjeta Arduino una resolución de lectura de 5V/1024 (tensión máxima/número de unidades) o 4.9 mV. Esta resolución corresponde a la tensión mínima necesaria para modificar el valor digital leído. Es posible obtener una precisión de lectura mayor, modificando este valor máximo de tensión.

© Editions ENI - All rights reserved