Arduino

José Luis Poza Luján. Sergio Sáez Barona

Actividad 5. Control reactivo

Contenido

1 Introducción

1.1 Justificación

Esta actividad describe un sistema reactivo sencillo, para implementar con un Arduino, pero lo suficientemente complejo como para valorar las bondades del diseño e implementación de sistemas reactivos utilizando patrones.

1.2 Objetivos

Los objetivos que se pretenden conseguir con este ejercicio son:

- Plantear un ejemplo sencillo que contenga los elementos básicos de un sistema reactivo.
- Valorar el uso de patrones de diseño para implementar sistema reactivos.

1.3 Descripción de la actividad

La actividad se divide en dos bloques: el primero trata de revisar conceptos básicos del diseño de un sistema reactivo, mientras que el segundo plantea la creación de un sistema de sensorización inteligente a partir de código ya trabajado.

1.4 Lista de material

El material que se va a utilizar durante la realización de este ejercicio es el siguiente:

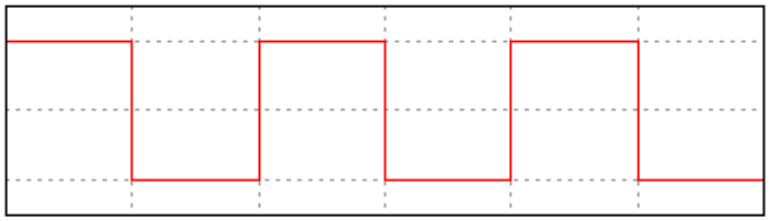
- Placa Arduino.
- 2 pulsadores.
- Un diodo led.
- Resistencias del pull-down, etc.

2 Conceptos básicos de sistemas reactivos

El desarrollo del presente bloque se va a dividir en diversas etapas. En cada etapa se irán añadiendo nuevas funcionalidades al ejercicio, siempre partiendo de los resultados de la etapa anterior.

2.1 Montaje básico

El montaje que se pretende realizar es un sistema de control de un led. El sistema, tras la llegada de un cierto evento, deberá generar una ráfaga intermitente en el led indicador. La ráfaga consistirá en tres pulsos del led encendido, con un ciclo de trabajo de un segundo, es decir, un segundo encendido, un segundo apagado, etc. Para generar el evento de activación de la ráfaga se utilizará inicialmente un pulsador.



Para llevar a cabo este ejercicio se conectará un pulsador a una entrada digital del Arduino. Denominaremos **ENCENDIDO** al pulsador conectado en la entrada digital y **ACTIVAR** a la entrada a la que se encuentra conectado dicho pulsador. Dicha entrada deberá ofrecer un valor **HIGH** cuando el pulsador **ENCENDIDO** esté pulsado. Para que el valor de la entrada sea **LOW** cuando el pulsador esté libre se deberá utilizar una resistencia de *pull-down*.

Ejercicio 1. Realizar el montaje con el pulsador **ENCENDIDO** y comprobar su correcto funcionamiento utilizando el puerto serie para mostrar el estado de la entrada digital **ACTIVAR**. Al *sketch* resultante se le denominará `sistema_basico.ino`

Se sugiere que cada entrada o salida digital (pin) se le asigne un nombre en el *sketch* mediante un **#define**, por ejemplo:

```
#define PIN_ACTIVAR 5
```

A continuación, se pretende conectar un led, que denominaremos **ILUMINACIÓN** a una salida digital de la placa Arduino. Dicha salida digital, que denominaremos **LED**, se deberá activar inicialmente siempre que el pulsador **ENCENDIDO** este activo.

Ejercicio 2. Realizar el montaje del led **ILUMINACIÓN** conectándolo a la salida digital **LED**. A continuación se deberá modificar el *sketch* `sistema_basico.ino` para que active la salida **LED** siempre que esté activa la entrada **ACTIVAR**.

A continuación, con el fin de aislar la lectura de los sensores de la lógica de control, vamos a proceder a encapsular la lectura de la entrada digital en una función. Si en futuro el mecanismo para leer la señal de activar el encendido del led se modifica, será esta función la única que haya que modificar.

Recordar que la definición de una función tiene el siguiente aspecto:

```
int nombre_de_la_función(/* parámetros */) {
      /* código */
      return VALOR;
}
```

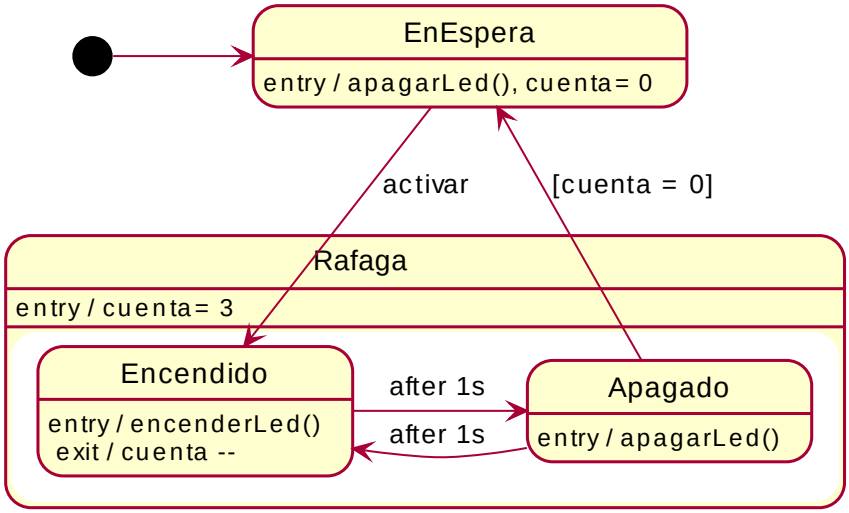
Ejercicio 3. Encapsular la lectura de la entrada **ACTIVAR** en una función que denominaremos **leerActivar()** y que devolverá un valor entero (**int**) con el estado de la entrada. Modificar el código del bucle **loop()** para que utilice la función **leerActivar()** en vez de la lectura directa de la entrada **ACTIVAR**.

Ejercicio 4. Encapsular la escritura en la salida **LED** en dos funciones que denominaremos **encenderLed()** y **apagarLed()**. Modificar el código del bucle **loop()** para que utilice dichas funciones en vez de la escritura directa de la salida **LED**.

Comprobad que al pulsar el botón **ENCENDIDO** se enciende el led **ILUMINACIÓN** y que el led se apaga al soltar el pulsador.

2.2 Sistema reactivo básico

A continuación se pretende modificar el montaje básico para que la detección de una pulsación en el **ENCENDIDO** inicie una ráfaga de 3 tres parpadeos de un segundo de duración en el led **ILUMINACIÓN**. Para ello se deberá implementar la máquina de estados que se muestra a continuación:



Vamos a proceder a su implementación paso a paso. Para ello aprovecharemos parte del código del *sketch* `sistema_basico.ino` que copiaremos en un nuevo *sketch* `sistema_reactivo.ino`. Mantendremos las definiciones de las entradas (**#define**s) y la función **leerActivar()**. El código de **loop()** se deberá descartar.

Vamos a proceder a la implementación progresiva del sistema reactivo. Comenzaremos por la definición de los estados en los que se puede encontrar el sistema.

Ejercicio 5. Añadir al fichero `sistema_reactivo.ino` la definición de los estados del sistema reactivo mediante un tipo enumerado (**EN_ESPERA**, **ENCENDIDO**, **APAGADO**). Utilizad letras mayúsculas para los nombres de los mismos y el prefijo **EST_** para diferenciarlos de los eventos.

A continuación introduciremos la definición de los eventos que se pueden producir en el sistema.

Ejercicio 6. Añadir al fichero `sistema_reactivo.ino` la definición de los eventos del sistema reactivo mediante un tipo enumerado (**ACTIVAR**, **AFTER_1S**). Utilizad letras mayúsculas para los nombres de los mismos y el prefijo **EV_** para diferenciarlos de los estados.

Una vez definidos los estados y eventos del sistema, vamos a incorporar a la implementación del sistema el código de captura de los eventos. La

captura de eventos se realizará mediante funciones similares a la que se muestra a continuación:

```
evento_t comprobarEvento() {
  evento_t evento = EV_NINGUNO;

  /* Código para comprobar el estímulo correspondiente */
  /* Se asignará el evento resultante a la variable 'evento' */

  return evento;
}
```

Ejercicio 7. Añadir al fichero `sistema_reactivo.ino` una función por cada posible evento (**comprobarActivar()**, **comprobarAfter1S()**). Cada función deberá devolver el evento en cuestión (**EV_ACTIVAR, EV_AFTER_1S**) ó **EV_NINGUNO** si el evento no se ha producido.

Para la implementación de la función **comprobarAfter1S()** supondremos que existe una variable **comienza_after_1s** que tiene el valor en milisegundos del instante en que comenzó la cuenta. El código de comprobación sería similar al que aparece a continuación:

```
if (estado == EST_ENCENDIDO || estado == EST_APAGADO) {
      if (millis() - comienza_after_1s >= 1000) {
            evento= EV_AFTER_1S;
      }
}
```

A continuación incorporaremos la detección de los eventos al bucle principal.

Ejercicio 8. Añadir en el bucle **loop()** la detección de los eventos mediante la invocación de las funciones oportunas. Si la función invocada devuelve algo distinto de **EV_NINGUNO**, dicho evento se deberá añadir a la cola de eventos.

Finalmente, trasladaremos el comportamiento del sistema al bucle principal mediante dos estructuras **switch** anidadas como las mostradas en las transparencias.

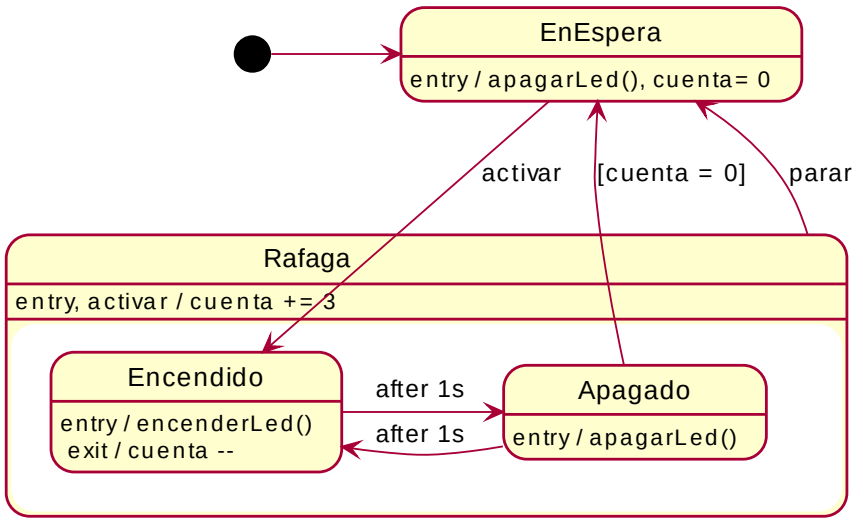
Ejercicio 9. Añadir en el bucle **loop()** el comportamiento del sistema en función del estado actual (**switch** externo) y del evento producido (**switch** interno). No olvidar que las transiciones deben modificar la variable de estado si el sistema lo requiere.

Comprobad que al pulsar el botón **ENCENDIDO** se produce una ráfaga de tres destellos en el led iluminación.

2.3 Completando el sistema reactivo

Una vez implementado el sistema básico vamos a proceder a completar el comportamiento deseado con la incorporación del sistema de parada y el de reactivación.

La siguiente figura muestra el esquema completo del sistema reactivo.



Lo primero que deberemos hacer es incorporar un nuevo pulsador al montaje original. Denominaremos a este nuevo pulsador **PARADA** y lo conectaremos a una nueva entrada digital que denominaremos **PARAR**. El esquema de conexiones será similar al del pulsador original.

Ejercicio 10. Añadir al fichero `sistema_reactivo.ino` la definición del nuevo pin.

Vamos a añadir a continuación el manejo de los nuevos eventos.

Ejercicio 11. Añadir la definición del los nuevos eventos y de las funciones de detección correspondientes (**`comprobarXXX()`**). No olvidaros de actualizar el número de eventos en la constante **`NUM_EVENTOS`**.

Ejercicio 12. Añadir la comprobación de los nuevos eventos al bucle **`loop()`**.

Finalmente, añadiremos las transiciones y las acciones correspondientes en el bucle principal.

Ejercicio 13. Modificar la construcción **`switch`** del bucle principal para añadir el código de tratamiento de los nuevos eventos. Recordad que las transiciones internas que no tienen estado destino no deben modificar el estado del sistema (variable **`estado`**).

Comprobad que al pulsar el botón **`ENCENDIDO`** se produce una ráfaga de tres destellos en el led iluminación, que con el botón **`PARADA`** de se interrumpe la ráfaga y que pulsando el botón de **`ENCENDIDO`** durante una ráfaga ésta continua durante tres destellos adicionales.

3 Conclusiones

En esta actividad se han revisado los conceptos básicos de control reactivo y se ha creado un bucles de control reactivo de dificultad progresiva.

Ejercicio 13. Autoevaluación.

Determinar el grado de utilidad (1: nada, 2: poco, 3: alguno, 4: mucho, 5: imprescindible) obtenido para cada apartado de los desarrollados y el nivel adquirido en el aprendizaje de los mismos (1: ninguno, 2: poco, 3: suficiente, 4: mucho, 5: experto)

Apartado	Utilidad					Nivel adquirido				
	1	2	3	4	5	1	2	3	4	5
Conceptos de control										
Entorno de aplicación										

Ejercicio 14. Ampliaciones y trabajo personal. Determinar qué posibles inquietudes personales se pueden realizar relacionadas con la actividad.