Problema #6

En este ejercicio, se debe plantear una solución para el control del agua de una piscifactoría. Los periféricos a analizar son el temporizador, el teclado, y dos controladores de nivel de suciedad y temperatura del agua.

Apartado A

Estados del sistema:

Para determinar en qué estado se encuentra el sistema y tomar las acciones oportunas, se define la variable global *estado_actual* y se inicializa a *Normal*. Los posibles valores de esta variable son los siguientes estados:

- **Normal**: en este estado, la situación del agua es la correcta para la supervivencia de los peces. Sin embargo, como explica el enunciado, cada 5 minutos hay que comprobar la temperatura del agua. En caso de ser demasiado alta (> 15º C) o demasiado baja (< -5º C), habrá que tomar las medidas necesarias mediante transición de estados. En caso contrario, no se ha de tomar ninguna medida especial.
  Para el control del agua, hay que realizar una encuesta temporizada al controlador K_{temp}. Suponemos declarada una variable de ámbito global llamada *temperatura_actual* que indica la temperatura del agua, y otra llamada *segundos* que indica los segundos que han pasado desde la última lectura de la temperatura.
  Supongamos entonces que el temporizador está configurado para que interrumpa una vez por segundo. La rutina de atención a las interrupciones del temporizador incrementará en uno siempre y cuando nos encontremos en el estado *Normal*:

```
if( estado_actual == Normal || estado_actual == Sucia )
      segundos++;
```

  Deducimos del enunciado que no se hace control mientras se está tratando la temperatura (estados *Caliente* y *Fria*), pero sí mientras se está limpiando (estado *Sucia*).
  Transiciones:
  a) La primera transición posible se deduce del controlador de temporizador que acabo de explicar. Cuando la variable segundos llega al valor equivalente a 5 minutos, esto es, 300 segundos, hay que realizar una comprobación de temperatura (encuesta) y pasar al estado *Caliente* si ésta es mayor que 15º C, o a *Fría* si es menor que -5º C. De ser una temperatura adecuada, se pone el contador de tiempo a cero para repetir el proceso a los 5 minutos.
     Por lo tanto, la rutina de atención a las interrupciones del temporizador hay que completarla de esta forma:

```
if( estado_actual == Normal || estado_actual == Sucia ) {
      segundos++;
      if( estado_actual == Normal && segundos == 300 ) {
            while( !InPort(REST_KTEMP) ) {}
            temperatura_actual = InPort(RDAT_KTEMP);
            strobe(RCONT_KTEMP);
            REST_KTEMP = 0;
            if( temperatura_actual > 15 )
                  estado_actual = Caliente;
            else if( temperatura_actual < -5 )
                  estado_actual = Fria;
            else
                  segundos = 0;
      }
}
```

b) La otra transición posible es la que ocurre cuando el agua está demasiado sucia. El controlador puede interrumpir en cualquier momento, independientemente de la temperatura del agua. Como el enunciado no dice nada al respecto, voy a elegir dar preferencia a las tareas de limpieza del agua sobre las de ajuste de temperatura. Si se lanza una interrupción del controlador de suciedad, la ejecución se mantendrá en su rutina de atención hasta que se haya terminado de limpiar, como se explica más adelante.

- **Caliente**: el sistema entra en este estado cuando la temperatura del agua es superior a 15º C. Surge la necesidad de echar 100 litros de hielo al agua para enfriarla, las veces que haga falta hasta que se alcance una temperatura inferior a 15º C. Esto se consigue mediante la rutina *hielos()* descrita en el enunciado y que se supone implementada. Una vez vertido el hielo, hay que volver a comprobar la temperatura para ver si se ha llegado al punto deseado. Es por esto que la temperatura se analiza constantemente dentro de un bucle *while*:

```
while( temperatura_actual > 15 ) {
      hielos();
      while( !InPort(REST_KTEMP) ) {}
      temperatura_actual = InPort(RDAT_KTEMP);
      strobe(RCONT_KTEMP);
      REST_KTEMP = 0;
}
```

Hay que destacar el bucle *while* anidado que aparece en el código anterior. El objetivo de este bucle es esperar a que el termómetro haga una lectura de la temperatura del agua. Sabremos que la ha realizado y que podemos leer el registro *RDAT_KTEMP* con seguridad cuando en el registro de estado *REST_KTEMP* haya un 1. Suponemos escritas las funciones *InPort(reg)* para leer los datos de un registro y *strobe(reg)* para hacer el *STROBE* que se requiere en el enunciado.

Suponemos correcta la última instrucción, que pone a cero el registro de estado del controlador para así asegurarnos en la próxima lectura de que se trata de una nueva medición de la temperatura, y no de datos obsoletos.

Transiciones:

Una vez que se alcanza una temperatura menor a 15º C, se vuelve al estado correspondiente: *Normal* si la temperatura está entre -5º y 15º C, o *Fría* si la temperatura ahora es menor que -5º C.

```
if(temperatura_actual < -5) {
      estado_actual = Fria;
} else {
      segundos = 0;
      estado_actual = Normal;
}
```

Como se puede observar, si el estado del agua ya es normal se establece el contador de temporizador a cero porque según el enunciado, a partir de este instante se vuelve a comprobar la temperatura a los cinco minutos.

- **Fria**: la situación que describe este estado es similar al estado *Caliente*. La diferencia radica en que la temperatura no es adecuada por ser demasiado baja (en lugar de alta como en el estado *Caliente*), es decir, es menor que -5º C.
  En lugar de llamar a la rutina predefinida *hielos()* se llamará a *agua_caliente()*, pero el resto de acciones son las mismas: medir la temperatura, repetir hasta que se superen los -5º C, y hacer una transición al estado *Normal* si la temperatura está entre -5º C y 15º C o a *Caliente* si supera los 15º C.

Nota: vista la similitud entre estos dos estados, se podrían combinar en un único estado llamado *TemperaturaInadecuada*. Habría que diferenciar si es inadecuada por exceso o por defecto, para así llamar a la subrutina adecuada y comparar con las temperaturas correctas. No obstante, como ya lo he esbozado en estados separados, lo voy a dejar así.

- **Sucia:** cuando el sistema pasa a este estado, se entiende que el agua está demasiado sucia y se requieren los servicios de los operarios de limpieza para que limpien el agua.
  El primer paso es activar las redes con la operación predefinida *activar_redes()*, una alarma sonora con *alarma()* y poner el contador *tiempo_alarma* a cero. El enunciado también exige un *STROBE*.
  Según el enunciado, no se hace nada más hasta que se pulsa la tecla 'D', por lo que en la rutina de atencion a la interrupción se añadirá un bucle

```
while( estado_actual == Sucia ) {}
```

que terminará cuando se cambie de estado, cosa que sólo puede ocurrir al pulsar la tecla 'D'.
  Tras este bucle, se deben llamar las funciones enunciadas para desactivar las redes, cambiar los containers, y parar la alarma si aún no ha sido parada.
  Como la alarma no debe estar más de 30 segundos encendida, el temporizador se debe adaptar a esta nueva función para contar el tiempo que lleva encendida. Por ello, se define una variable global *tiempo_alarma* inicializada a cero al comenzar la rutina de atención a la interrupción del controlador de suciedad, y se va incrementando en uno por la rutina de atención a la interrupción del temporizador cuando el sistema se encuentra en el estado *Sucia*:

```
if( estado_actual == Normal ) {
      segundos++;
      if( segundos > 300 ) {
            […]
      }
} else if( estado_actual == Sucia ) {
      segundos++;
      tiempo_alarma++;
      if( tiempo_alarma > 30 )
            parar_alarma();
}
```

Observar que pueden superarse los 5 minutos del control de la temperatura mientras se está limpiando el agua. Por eso se sustituye el == de la condición por un >, para que en caso de pasar esos 5 minutos, se haga la comprobación de la temperatura tan pronto como se termine de limpiar.

Transiciones:

El sistema vuelve al estado en el que se encontraba previamente (que habrá de guardarse en una variable *estado_anterior*) cuando el operario jefe de limpieza pulse la tecla 'D'.

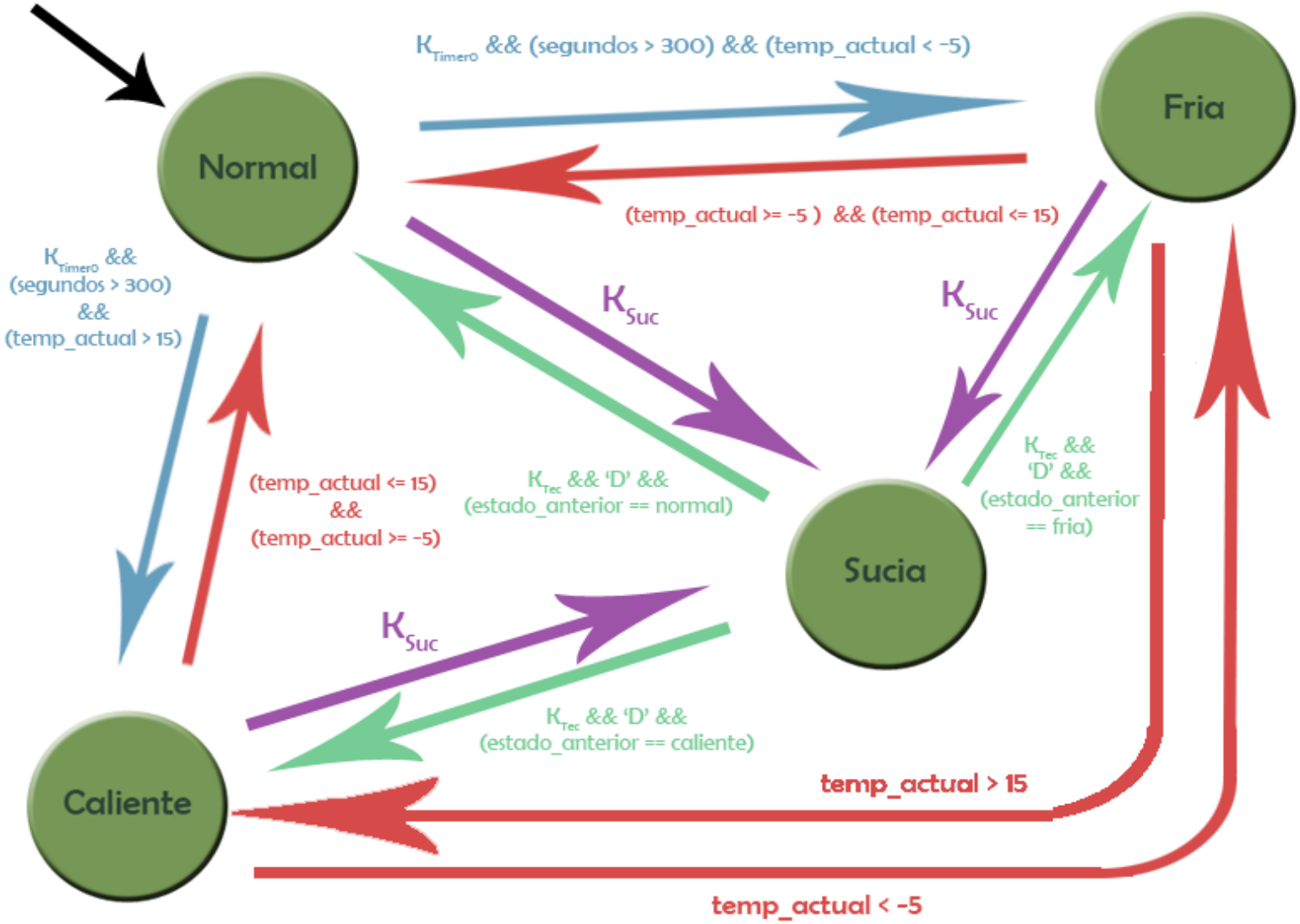
El teclado se controla por interrupción, en la siguiente rutina:

```
codigo_tecla = InPort(RDAT_KTECLADO);
strobe(RCONT_KTECLADO);
if( MAKE(codigo_tecla) && TABLA_ASCII[codigo_tecla]=='D' &&
estado_actual == Sucia ) {
      estado_actual = estado_anterior;
}
```

Se suponen implementadas la función *MAKE()* y la tabla *TABLA_ASCII[]* como en el ejercicio resuelto. Al final de la rutina, se vuelve al estado anterior a la interrupción del controlador de suciedad.

Máquina de estados o autómata

Pese a que el enunciado no la pide, la he elaborado para mayor claridad. No figuran las acciones que se toman en cada transición pues ya han sido explicadas y no encajaban bien en el dibujo. La variable *temperatura_actual* aparece abreviada para ganar espacio.


Normal
Fria
Sucia
Caliente
K Timero && (segundos > 300) && (temp_actual < -5)
(temp_actual >= -5) && (temp_actual <= 15)
K Timero && (segundos > 300) && (temp_actual > 15)
(temp_actual <= 15) && (temp_actual >= -5)
K Suc
K Suc
K Suc
K Tec && 'D' && (estado_anterior == normal)
K Tec && 'D' && (estado_anterior == fria)
K Tec && 'D' && (estado_anterior == caliente)
temp_actual > 15
temp_actual < -5

Rutinas de atención y encuesta

Con todo lo expuesto en el apartado anterior, se construyen las siguientes rutinas:

- Es la única rutina de encuesta, las demás son de atención a interrupciones. Realiza la lectura de la temperatura del agua mediante el controlador K_{Temp}

```
void leer_temperatura() {
      while( !InPort(REST_KTEMP) ) {}
      temperatura_actual = InPort(RDAT_KTEMP);
      strobe(RCON_KTEMP);
      REST_KTEMP = 0;
}
```

Dado que estas líneas aparecen en varias partes del algoritmo, se escribe como una función para reutilizar el código.

- Rutina de atención a las interrupciones de temporizador, K_{Timer0}

```
void RAI_KTIMER() {
      if( estado_actual == Normal ) {
            segundos++;
            if( segundos > 300 ) {
                  leer_temperatura();
                  if( temperatura_actual > 15 )
                        estado_actual = Caliente;
                  else if( temperatura_actual < -5 )
                        estado_actual = Fria;
                  else
                        segundos = 0;
            }
      } else if( estado_actual == Sucia ) {
            segundos++;
            tiempo_alarma++;
            if( tiempo_alarma > 30 )
                  parar_alarma();
      }
}
```

- Rutina de atención a las interrupciones de teclado, $K_{Teclado}$

```
void RAI_KTECLADO() {
      codigo_tecla = InPort(RDAT_KTECLADO);
      strobe(RCON_KTECLADO);
      if( MAKE(codigo_tecla) &&
      TABLA_ASCII[codigo_tecla]=='D' && estado_actual ==
      Sucia ) {
            estado_actual = estado_anterior;
      }
}
```

- Rutina de atención a las interrupciones del controlador de suciedad, K_{Suc}

```
void RAI_KSUC() {
      estado_anterior = estado_actual;
      estado_actual = Sucia;
      strobe(RCON_KSUC);
      tiempo_alarma = 0;
      alarma();
      activar_redes();
      while( estado_actual == Sucia ) {}
      desactivar_redes();
      cambiar_containers();
      if( tiempo_alarma <= 30 )
            parar_alarma();
}
```

Programa principal

En el progama principal se pueden diferenciar varias partes:

- ✓ Declaración e inicialización de variables globales.
- ✓ Asignación de rutinas de atención a interrupciones mediante *irqSet()*.
- ✓ Bucle principal del algoritmo, que entendemos que no acaba nunca.

```
// Variables globales
estado_actual = Normal;
estado_anterior;
segundos = 299; // Primera medición tan pronto como se empieza
temperatura_actual;
tiempo_alarma = 0;

void main() {
	// Rutinas de atención a ejecutar por cada interrupción
	irqSet(IRQ3, RAI_KTIMER);
	irqSet(IRQ12, RAI_KTECLADO);
	irqSet(IRQ26, RAI_KSUC);
	// Bucle principal
	while(true) {
		switch(estado_actual) {
		case Caliente:
			while( temperatura_actual > 15 ) {
				hielos();
				leer_temperatura();
			}
			if(temperatura_actual < -5) {
				estado_actual = Fria;
			} else {
				segundos = 0;
				estado_actual = Normal;
			}
		case Fria:
			while( temperatura_actual < -5 ) {
				agua_caliente();
				leer_temperatura();
			}
			if(temperatura_actual > 15) {
				estado_actual = Caliente;
			} else {
				segundos = 0;
				estado_actual = Normal;
			}

		}
	}
}
```

Apartado B

La sincronización del teclado por encuesta supondría que en la rutina de atención *RAI_KSUC()*, en lugar de tener un bucle *while* vacío, tendríamos que estar constantemente consultando los registros del controlador de teclado para saber si ha sido pulsada la tecla 'D'.

De ser así saldríamos del bucle, y seguiría ejecutándose la rutina con normalidad. Como es evidente, no haría falta la rutina de atención a las interrupciones de teclado.