Trabajo Práctico 1
Antiguo Reloj Despertador

Grupo 1

Julio Kriger
L.U.: 207/69
jkriger@dc.uba.ar

6 de mayo de 2002

Índice

1. El modelo conceptual

El sistema elegido a modelar en un antiguo reloj despertador. Este reloj tiene un péndulo que marca el ritmo de un segundo al oscilar. El reloj muestra a través de sus agujas la hora y minuto del día. Tiene también una alarma con agujas que muestra la hora y minuto en que se activará, emitiendo un sonido.

El usuario del reloj, que no se modelará en este trabajo, puede setear la hora y minutos girando en sentido horario (incrementar) o antihorario (decrementar) la aguja de los minutos, y ésta al completar un giro incrementará o decrementará según corresponda la aguja de la hora. También puede realizar la misma acción para la alarma girando la aguja de los minutos de la alarma. Claro está que se pueden realizar las habituales operaciones de encender y apagar el reloj, y activar y desactivar la alarma.

1.1. Modelos atómicos

A continuación se explicarán los modelos atómicos observados a partir del sistema elegido.

1.1.1. Péndulo

Este modelo mantiene el estado del péndulo: parado y oscilando de derecha a izquierda, a través de una variable de estado. Acepta eventos de entrada: arrancar y parar; tiene eventos de salida: tic cada 1 segundo. Ver figura 1.

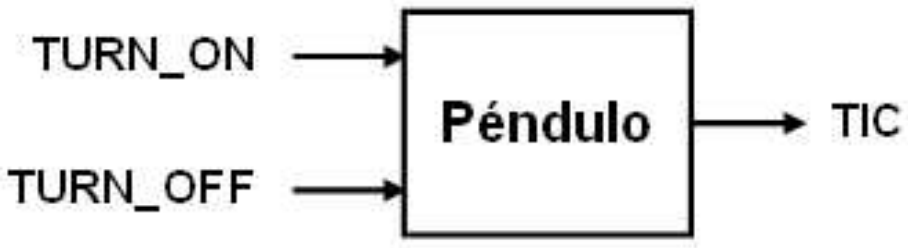

Figura 1: Péndulo

1.1.2. Control de Encendido

Este modelo mantiene el estado estado del reloj: encendido y apagado, a través de una variable de estado. Acepta eventos de entrada: encender y apagar; tiene eventos de salida: encendido y apagado. Ver figura 2.

1.1.3. Alarma

Este modelo mantiene el estado de la alarma: activa, inactiva, hora y minuto del reloj y de la alarma, a través de contadores y variables de estado. Acepta eventos de entrada: valor de la aguja de la hora y minuto del reloj, valor de la aguja de la hora y minuto de la alarma, activar y desactivar; tiene eventos de salida: activa, inactiva y sonido despertador. Ver figura 3.

Figura 2: Control de Encendido

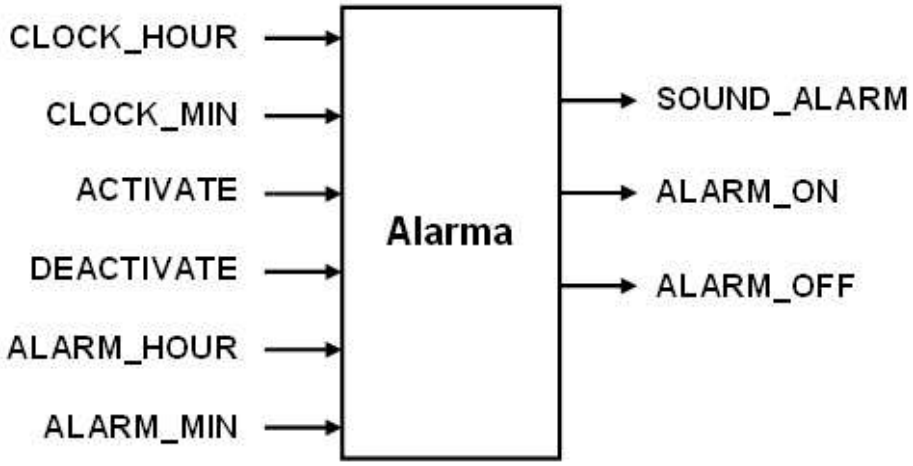

Figura 3: Alarma

1.1.4. Aguja de la Hora

Este modelo mantiene el estado de la aguja de la hora, a través de contadores. Acepta eventos de entrada: incrementar y decrementar, tiene un evento de salida: valor de la aguja. Ver figura 4.

Figura 4: Aguja de Hora

1.1.5. Aguja del Minuto

Este modelo mantiene el estado de la aguja del minuto, a través de contadores. Acepta eventos de entrada: incrementar, decrementar y tic; tiene eventos de salida: incrementar, decrementar y el valor de la aguja. Ver figura 5.

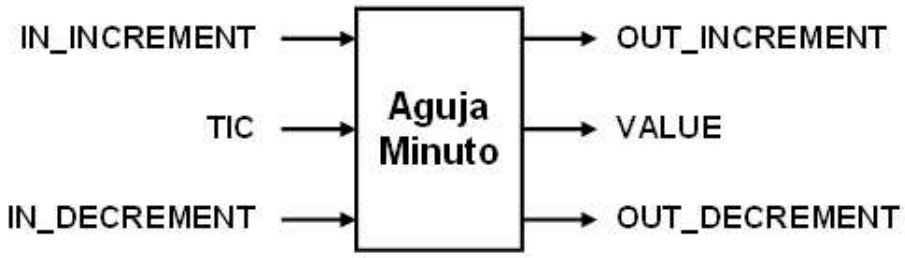

Figura 5: Aguja de Minuto

1.2. Modelos acoplados

A continuación se explicarán los modelos acoplados observados a partir del sistema elegido.

1.2.1. Motor

Este modelo acopla los modelos Péndulo (figura 1) y Control de Encendido (figura 2). Este modelo es el *motor* que hace funcionar a todos los componentes del reloj. Acepta eventos de entrada: encender y apagar; tiene eventos de salida: tic, encendido y apagado. Ver figura 6.

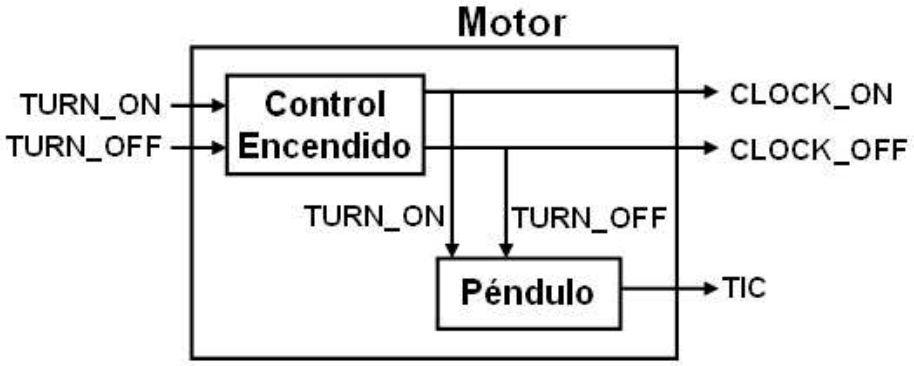

Figura 6: Motor

1.2.2. Control de Agujas

Este modelo acopla los modelos Aguja Minuto (figura 5) y Aguja Hora (figura 4). Este modelo controla las agujas que marcan la hora y minuto, sirve tanto

para el reloj como para la alarma. Acepta eventos de entrada: incrementar y decrementar la aguja del minuto y tic; tiene eventos de salida: valor de la aguja de la hora y minuto. Ver figura 7.

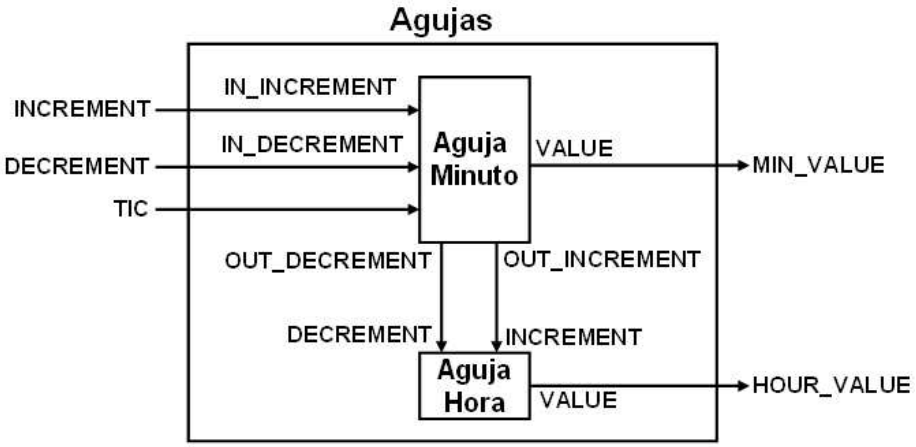

Figura 7: Control de Agujas

1.2.3. Control de Alarma

Este modelo acopla los modelos Alarma (figura 3) y Agujas (figura 7). Este modelo lleva el control de la alarma, la hora a la que debe sonar y si está activa o no. Acepta eventos entrantes: valor de la aguja de la hora y minuto del reloj, incrementar o decrementar la aguja del minuto de la alarma, activar y desactivar alarma; tiene eventos de salida: alarma activa e inactiva, y sonido despertador, valor de la hora y minuto de la alarma. Ver figura 8.

1.2.4. Reloj

Este modelo acopla los modelos Motor (figura 6), Control de Agujas (figura 7) y Control de Alarma (figura 8). Este es el modelo principal, *el modelo top*, es el que junta a los otros modelos y los hace interactuar entre sí par llevar a cabo las funciones de un antiguo reloj despertador. Acepta eventos entrantes: encender y apagar el reloj, incrementar y decrementar la aguja del minuto del reloj, activar y desactivar alarma, incrementar y decrementar la aguja del minuto de la alarma; tiene eventos de salida: reloj encendido y apagado, alarma activa e inactiva, valor de la aguja de la hora y minuto del reloj, alarma activa e inactiva, valor de la aguja de la hora y minuto de alarma, sonido despertador. Ver figura 9.

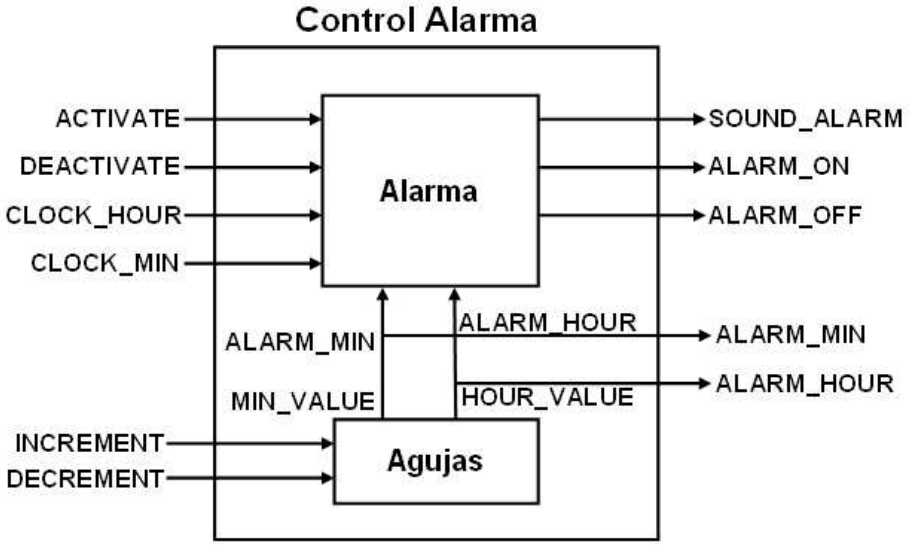

Figura 8: Control de Alarma

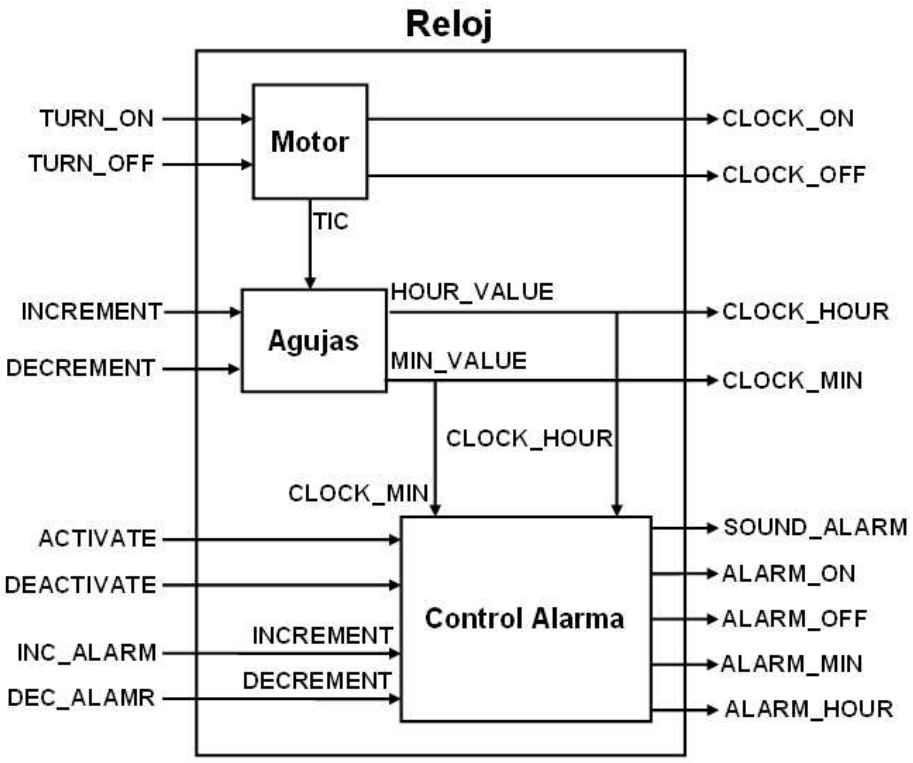

Figura 9: Reloj

2. Especificación de los modelos atómicos

A continuación se especificaran los modelos atómicos descritos anteriormente. Si bien no se realizará una especificación formal como la dada en clase, se utilizará un especificación alternativa para una fácil y comprensible lectura.

Primeramente se declaran los *eventos entrantes y salientes*. Luego, los *estados* se declaran como nombre de variable y entre "(" y ")" sus posibles valores. Por último, sigue un pseudocódigo de las *funciones de transición externa, interna y de salida*.

2.1. Péndulo

Eventos Entrantes

```
TURN_ON, TURN_OFF
```

Eventos Salientes

```
TIC
```

Estados

```
estado (OFF, RIGHT, LEFT)
```

Función de transición externa

```
Si llega evento TURN_ON
    Si estado == OFF
        estado = RIGHT
        Esperar 1 segundo y pasar a fase Activa
Si llega evento TURN_OFF
    Si estado == ON
        estado = OFF
        Pasivar
```

Función de transición interna

```
Si estado == RIGHT
    estado = LEFT
        Esperar 1 segundo y pasar a fase Activa
Si estado == LEFT
    estado = RIGHT
        Esperar 1 segundo y pasar a fase Activa
```

Función de Salida

```
Enviar evento TIC
```

2.2. Control de encendido

Eventos Entrantes

```
TURN_ON, TURN_OFF
```

Eventos Salientes

```
CLOCK_ON, CLOCK_OFF
```

Estados

```
estado (ON, OFF)
```

Función de transición externa

```
Si llega evento TURN_ON
    Si estado == OFF
        estado = ON
        Esperar 0 segundos y pasar a fase Activa
Si llega evento TURN_OFF
    Si estado == ON
        estado = OFF
        Esperar 0 segundos y pasar a fase Activa
```

Función de transición interna

```
Pasivar
```

Función de Salida

```
Si estado == ON
    Enviar evento CLOCK_ON
Si estado == OFF
    Enviar evento CLOCK_OFF
```

2.3. Alarma

Eventos Entrantes

```
CLOCK_HOUR_1 ... CLOCK_HOUR_12, CLOCK_MIN_0 ... CLOCK_MIN_59,
ALARM_HOUR_1 ... ALARM_HOUR_12, ALARM_MIN_0 ... ALARM_MIN_59,
ACTIVATE, DEACTIVATE
```

Eventos Salientes

```
SOUND_ALARM, ALARM_ON, ALARM_OFF
```

Estados

```
estado (ACTIVE, INACTIVE), habilitar_salida (TRUE, FALSE),
hora_reloj (1 ... 12), minuto_reloj (0 ... 59), hora_alarma (1 ...
12), minuto_alarma (0 ... 59)
```

Función de transición externa

```
Si llega evento CLOCK_HOUR_1 ... CLOCK_HOUR_12
    hora_reloj = CLOCK_HOUR_i
Si llega evento CLOCK_MIN_0 ... CLOCK_MIN_59
    minuto_reloj = CLOCK_MIN_i
Si llega evento ALARM_HOUR_1 ... ALARM_HOUR_12
    hora_alarma = ALARM_HOUR_i
Si llega evento ALARM_MIN_0 ... ALARM_MIN_59
    minuto_alarma = ALARM_MIN_i

Si llega evento ACTIVATE
    Si estado == INACTIVE
        habilitar_salida = TRUE
        estado = ACTIVE
        Esperar 0 segundos y pasar a fase Activa
Sino
    Si llega evento DEACTIVATE
        Si estado == ACTIVE
```

```
            habilitar_salida = TRUE
            estado = INACTIVE
            Esperar 0 segundos y pasar a fase Activa
    Sino
        Si estado == ACTIVE y hora_reloj == hora_alarma y
        minuto_reloj == minuto_alarma
            Esperar 0 segundos y pasar a fase Activa
```

Función de transición interna

```
Pasivar
```

Función de Salida

```
Si habilitar_salida == TRUE
    habilitar_salida = FALSE
    Si estado == ACTIVE
        Enviar evento ALARM_ON
    Si estado == INACTIVE
        Enviar evento ALARM_OFF
Sino
    Si estado == ACTIVE
        Enviar evento SOUND_ALARM
```

2.4. Aguja hora

Eventos Entrantes

```
INCREMENT, DECREMENT
```

Eventos Salientes

```
VALUE_1 ... VALUE_12
```

Estados

```
hora (1 ... 12), minutos (0 ... 59)
```

Función de transición externa

```
Si llega evento INCREMENT
    Incrementar minutos
    Si minutos >= 60
        minutos = 0
        Incrementar hora
        Si hora >= 12
            hora = 1
        Esperar 0 segundos y pasar a fase Activa
Si llega evento DECREMENT
    Decrementar minutos
    Si minutos < 0
        minutos = 59
        Decrementar hora
        Si hora <= 0
            hora = 12
        Esperar 0 segundos y pasar a fase Activa
```

Función de transición interna

Pasivar

Función de Salida

Enviar evento VALUE_i (donde i corresponde a hora)

2.5. Aguja minuto

Eventos Entrantes

IN_INCREMENT, IN_DECREMENT, TIC

Eventos Salientes

OUT_INCREMENT, OUT_DECREMENT, VALUE_0 ... VALUE_59

Estados

evento_saliente (INC, DEC), minutos (0 ... 59), tics (0 ... 59)

Función de transición externa

```
Si llega evento TIC
    Incrementar tics
    Si tics >= 60
        tics = 0
        Incrementar minutos
        evento_saliente = INC
        Si minutos >= 60
            minutos = 0
        Esperar 0 segundos y pasar a fase Activa
Si llega evento IN_INCREMENT
    Incrementar minutos
    evento_saliente = INC
    Si minutos >= 60
        minutos = 0
    Esperar 0 segundos y pasar a fase Activa
Si llega evento IN_DECREMENT
    Decrementar minutos
    evento_saliente = DEC
    Si minutos < 0
        minutos = 59
    Esperar 0 segundos y pasar a fase Activa
```

Función de transición interna

Pasivar

Función de Salida

```
Si evento_saliente == INC
    Enviar evento OUT_INCREMENT
Si evento_saliente == DEC
    Enviar evento OUT_DECREMENT
Enviar evento VALUE_i (donde i es minutos)
```

3. Especificación de los modelos acoplados

A continuación se especificaran los modelos acoplados descritos anteriormente. Si bien no se realizará una especificación formal como la dada en clase, se utilizará un especificación alternativa para una fácil y comprensible lectura.

Primeramente se declaran los *eventos entrantes y salientes*. Luego, los *componentes* se declaran como nombre del componente y entre "(" y ")" su tipo. Por último, se declara la forma en que los componentes se acoplan, *acoplamiento de eventos entrantes, salientes e internos*. No se especifica la función de selección debido a que fue necesaria implementarla.

3.1. Motor

Ver figura 6.

Eventos Entrantes

```
TURN_ON, TURN_OFF
```

Eventos Salientes

```
CLOCK_ON, CLOCK_OFF, TIC
```

Componentes

```
control (Control de Encendido), pendulo (Pendulo)
```

Acoplamiento de eventos entrantes

```
TURN_ON --> control.TURN_ON

TURN_OFF --> control.TURN_OFF
```

Acoplamiento interno

```
control.CLOCK_ON --> pendulo.TURN_ON

control.CLOCK_OFF --> pendulo.TURN_OFF
```

Acoplamiento de eventos salientes

```
pendulo.TIC --> TIC

control.CLOCK_ON --> CLOCK_ON

control.CLOCK_OFF --> CLOCK_OFF
```

3.2. Control de agujas

Ver fingura 7.

Eventos Entrantes

```
TIC, INCREMENT, DECREMENT
```

Eventos Salientes

```
HOUR_VALUE_1 ... HOUR_VALUE_12, MIN_VALUE_0 ... MIN_VALUE_59
```

Componentes

```
min (Aguja Minuto), hora (Aguja Hora)
```

Acoplamiento de eventos entrantes

```
TIC --> min.TIC

INCREMENT --> min.IN_INCREMENT

DECREMENT --> min.IN_DECREMENT
```

Acoplamiento interno

```
min.OUT_INCREMENT --> hora.INCREMENT

min.OUT_DECREMENT --> hora.DECREMENT
```

Acoplamiento de eventos salientes

```
hora.VALUE_i --> HOUR_VALUE_i (donde i es un entero entre [1..12])

min.VALUE_i --> MIN_VALUE_i (donde i es un entero entre [0..59])
```

3.3. Control de alarma

Ver figura 8.

Eventos Entrantes

```
ACTIVATE, DEACTIVATE, CLOCK_HOUR, CLOCK_MIN, INCREMENT, DECREMENT
```

Eventos Salientes

```
ALARM_ON, ALARM_OFF, SOUND_ALARM, ALARM_MIN, ALARM_HOUR
```

Componentes

```
agujasAlarma (Control de Agujas), alarma (Alarma)
```

Acoplamiento de eventos entrantes

```
ACTIVATE --> alarma.ACTIVATE

DEACTIVATE --> alarma.DEACTIVATE

CLOCK_HOUR --> alarma.CLOCK_HOUR

CLOCK_MIN --> alarma.CLOCK_MIN

INCREMENT --> agujasAlarma.INCREMENT

DECREMENT --> agujasAlarma.DECREMENT
```

Acoplamiento interno

```
agujasAlarma.HOUR_VALUE_i --> alarma.ALARM_HOUR_i (donde i es un
entero entre [1..12])

agujasAlarma.MIN_VALUE_i --> alarma.ALARM_MIN_i (donde i es un
entero entre [0..59])
```

Acoplamiento de eventos salientes

```
alarma->ALARM_ON --> ALARM_ON
```

alarma.ALARM_OFF --> ALARM_OFF

alarma.SOUND_ALARM --> SOUND_ALARM

agujasAlarma.HOUR_VALUE_i --> ALARM_HOUR_i (donde i es un entero entre [1..12])

agujasAlarma.MIN_VALUE_i --> ALARM_MIN_i (donde i es un entero entre [0..59])

3.4. Reloj

Eventos Entrantes

TURN_ON, TURN_OFF, ACTIVATE, DEACTIVATE, INCREMENT, DECREMENT, INC_ALARM, DEC_ALARM

Eventos Salientes

CLOCK_ON, CLOCK_OFF, ALARM_ON, ALARM_OFF, SOUND_ALARM, CLOCK_HOUR, CLOCK_MIN, ALARM_HOUR, ALARM_MIN

Componentes

agujas (Control de Agujas), controlalarma (Control de Alarma), motor(Motor)

Acoplamiento de eventos entrantes

ACTIVATE --> controlalarma.ACTIVATE

DEACTIVATE --> controlalarma.DEACTIVATE

INCREMENT --> agujas.INCREMENT

DECREMENT --> agujas.DECREMENT

INC_ALARM --> controlalarma.INCREMENT

DEC_ALARM --> controlalarma.DECREMENT

TURN_ON --> motor.TURN_ON

TURN_OFF --> motor.TURN_OFF

Acoplamiento interno

motor.TIC --> agujas.TIC

agujas.HOUR_VALUE --> controlalarma.CLOCK_HOUR

agujas.MIN_VALUE --> controlalarma.CLOCK_MIN

Acoplamiento de eventos salientes

motor.CLOCK_ON --> CLOCK_ON

motor.CLOCK_OFF --> CLOCK_OFF

```
agujas.HOUR_VALUE_i --> CLOCK_HOUR_i (donde i es un entero entre
[1..12])

agujas.MIN_VALUE --> CLOCK_MIN_i (donde i es un entero entre
[0..59])

controlalarma.SOUND_ALARM --> SOUND_ALARM

controlalarma.ALARM_ON --> ALARM_ON

controlalarma.ALARM_OFF --> ALARM_OFF

controlalarma.ALARM_HOUR_i --> ALARM_HOUR_i (donde i es un entero
entre [0..59])

controlalarma.ALARM_MIN_i --> ALARM_MIN_i (donde i es un entero
entre [0..59])
```

4. Testing de los modelos

Se adjunta con el código fuente una serie de scripts para la ejecución de los tests individuales de cada modelo atómico y acoplado descrito en el presente trabajo.

Estos tests se realizaron por medio de archivos *.ev*. En los mismo, dependiendo de cada modelo, se trató de buscar posibles errores en la codificación. Al pasar el test correctamente, se analizaron los resultados de la simulación para saber que *tan fehacientemente* refleja la realidad.

Se introdugieron errores a sabiendas para observar el comportamiento del sistema trabajando en condiciones anormales.

Aunque no se muestra en el trabajo, se han realizados los tests a todos los modelos. Gracias a éstos se corrigieron varios errores de lógica y se detectaron errores de la herramienta[1] a los cuales se le encontró un camino alternativo. Primeramente se testearon todos los modelos atómicos, y luego de que se corrigieran y hubiesen pasado nuevamente los tests correctamente se ejecutaron los tests de los modelos acoplados. A estos se los corrigió y se volvieron a testear hasta que corrieran exitosamente.

Todos los modelos, atómicos y acoplados, han pasado todos los tests satisfactoriamente.

A continuación se mostrará un ejemplo de la ejecución de un script de test del modelo acoplado Reloj[2].

Eventos entrantes

```
00:00:01:00 TURN_OFF -1.0
00:00:02:00 TURN_ON 1.0
00:00:03:00 INC_ALARM 1.0
00:00:04:00 INC_ALARM 1.0
00:00:05:00 ACTIVATE 1.0
00:00:06:00 DEACTIVATE -1.0
00:00:07:00 DEACTIVATE -1.0
00:00:08:00 ACTIVATE 1.0
00:00:08:00 ACTIVATE 1.0
00:00:09:00 INC_ALARM 1.0
00:00:10:00 INC_ALARM 1.0
00:00:11:00 INC_ALARM 1.0
00:00:12:00 INC_ALARM 1.0
00:00:13:00 INC_ALARM 1.0
00:00:14:00 INC_ALARM 1.0
00:00:15:00 INC_ALARM 1.0
00:00:16:00 INC_ALARM 1.0
00:00:17:00 INC_ALARM 1.0
00:00:18:00 INC_ALARM 1.0
00:00:19:00 DEC_ALARM -1.0
00:00:20:00 DEC_ALARM -1.0
00:00:21:00 DEC_ALARM -1.0
00:00:22:00 DEC_ALARM -1.0
```

[1] En conclusiones ver posibles mejoras a la herramienta.

[2] Se mostrará este ejemplo solamente, ya que los listados de los modelos restantes no aporta más información útil.

```
00:00:23:00 INCREMENT 1.0
00:00:24:00 INCREMENT 1.0
00:00:25:00 INCREMENT 1.0
00:00:26:00 INCREMENT 1.0
00:00:27:00 INCREMENT 1.0
00:00:28:00 INCREMENT 1.0
00:00:29:00 INCREMENT 1.0
00:00:30:00 DECREMENT -1.0
00:00:31:00 DECREMENT -1.0
00:10:00:00 TURN_OFF -1.0
```

Resultado

```
00:00:00:000 clock_min 0
00:00:00:000 clock_hour 1
00:00:00:000 alarm_min 0
00:00:00:000 alarm_hour 1
00:00:00:000 alarm_off 0
00:00:00:000 clock_off -1
00:00:02:000 clock_on 1
00:00:03:000 alarm_min 1
00:00:04:000 alarm_min 2
00:00:05:000 alarm_on 0
00:00:06:000 alarm_off 0
00:00:08:000 alarm_on 0
00:00:09:000 alarm_min 3
00:00:10:000 alarm_min 4
00:00:11:000 alarm_min 5
00:00:12:000 alarm_min 6
00:00:13:000 alarm_min 7
00:00:14:000 alarm_min 8
00:00:15:000 alarm_min 9
00:00:16:000 alarm_min 10
00:00:17:000 alarm_min 11
00:00:18:000 alarm_min 12
00:00:19:000 alarm_min 11
00:00:20:000 alarm_min 10
00:00:21:000 alarm_min 9
00:00:22:000 alarm_min 8
00:00:23:000 clock_min 1
00:00:24:000 clock_min 2
00:00:25:000 clock_min 3
00:00:26:000 clock_min 4
00:00:27:000 clock_min 5
00:00:28:000 clock_min 6
00:00:29:000 clock_min 7
00:00:30:000 clock_min 6
00:00:31:000 clock_min 5
00:01:02:000 clock_min 6
00:02:02:000 clock_min 7
00:03:02:000 clock_min 8
```

```
00:03:02:000 sound_alarm 0
00:04:02:000 clock_min 9
00:05:02:000 clock_min 10
00:06:02:000 clock_min 11
00:07:02:000 clock_min 12
00:08:02:000 clock_min 13
00:09:02:000 clock_min 14
00:10:00:000 clock_off -1
```

Análisis Según los resultados obtenidos, se puede observar que el reloj se comporto de manera correcta a lo especificado. Se puede observar que los eventos repetidos ACTIVATE y DEACTIVATE, para generar errores, no causan daño alguno que el sistema, ya que son ignorados.

5. Conclusión

Si bien el modelo elegido, un antiguo reloj despertador, no consta de gran dificultad lógica, se puede ver claramente como interactuan entre sí modelos atómicos y acoplados.

Si se comparan la complejidad de los modelos Alarma y Péndulo, es claro que el modelo Alarma es mucho más complejos. Esto indica que le modelo Alarma puede dividirse en otros modelos mas simples. Al tener modelos simple, la implementación y testing de los mismo se torna sencilla. No pasa esto con modelos complejos como Alarma. Queda como trabajo a futuro el replanteo del modelo Alarma.

El tener una buena base de modelos simples y lo más genéricos posibles, atómicos es de gran importancia. El que los modelos sean simples ayuda a crear y testear, de manera simple, modelos acoplados. El que sean genéricos ayuda a poder formar una librería de modelos atómicos, y poder reutilizarlos para crear modelos acoplados. Si los modelos no son genéricos, sino que están orientados un sistema específico no tendrá una gran reutilización ni formar de una buena librería.

Un tema a destacar es la facilidad de creación de modelos acoplados. Esta facilidad está dada gracias a que no es necesario realizar una nueva compilación y linkedición del simulador por cada cambio en los modelos acoplados.

Luego de la implementación del Reloj se observo un serie de *posibles mejoras a la herramienta*:

Manejo de contextos Se observo, al crear el modelo acoplado Reloj, la falta de manejo de contextos. Esto provocó que se que cambiasen nombres de los componentes de modelo acoplado Control de Agujas, ya que provocaba conflicto con el Control de Agujas del modelo acoplado Control de Alarma. Sería óptimo el manejo recursivo de contextos.

Creación del tipo de modelo acoplado Al crear modelos acoplados se deben *reescribir* los componentes que sean del tipo de un modelo acoplado ya especificado. Esto no ocurre con lo componentes del tipo atómico. Sería de gran ayuda que se puedan crear nuevos tipos de componentes a partir no solo de modelos atómicos sino también de modelos atómicos.