Trabajo Final Concurrente

Goldman, Ignacio

Papp, Nicolás

Introducción

El problema propuesto refiere al control de un sistema de bombeo de agua que es extraída del sumidero de un pozo de una mina. Dicho problema se debe abordar de una manera concurrente debido a la multitud de factores que inciden en su funcionamiento y a las características concurrentes del sistema. La técnica utilizada para abordar el problema fue a través del diseño de una Red de Petri que interpreta el funcionamiento del sistema y de un Monitor que sincroniza los eventos de la misma.

Requerimientos

Los requerimientos necesarios para el funcionamiento del sistema se dividen en dos tipos: funcionales y no funcionales.

Requerimientos funcionales:

* Debe poder accionar automáticamente la bomba en base a los niveles medidos en los sensores de agua y metano.
* Debe poder procesar eventos introducidos por los usuarios (prender, apagar).
* Señalización del entorno, si tienen valores críticos se deben señalizar con una alarma.
* Monitorización del Sistema (Logging). Todos los eventos del sistema son guardados para luego poder visualizarlos

Requerimientos no funcionales:

* Seguridad: La bomba no debe funcionar cuando el nivel de gas metano en la mina es alto.
* Confiabilidad: Los valores medidos deben ser exactos y el sistema debe responder de manera adecuada a toda situación posible.
* Eficiencia: El desempeño del programa debe ser lo suficiente rápido para poder disparar los eventos según los tiempos límites especificados.

Diagrama de Actividades

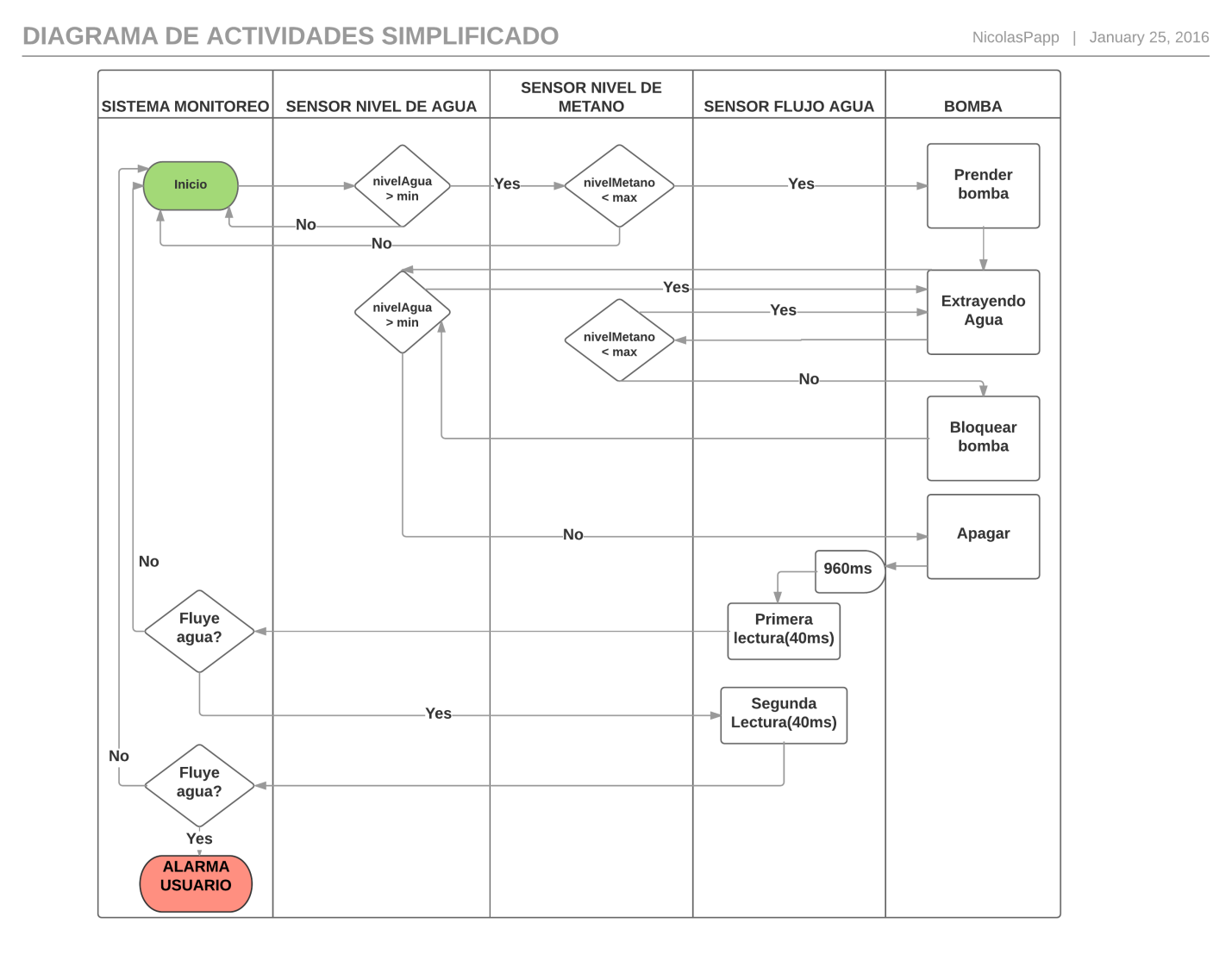
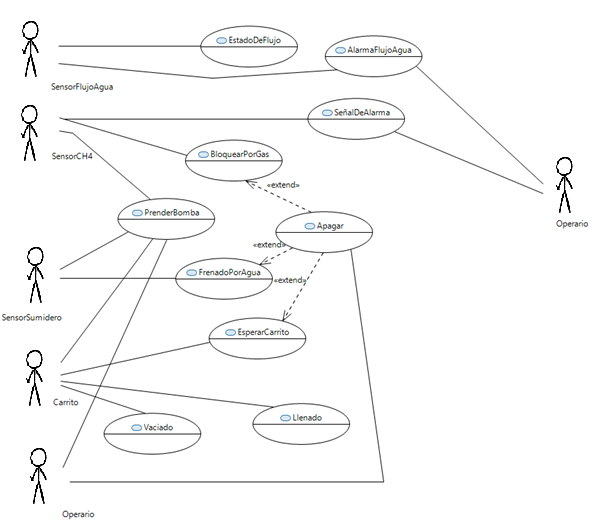


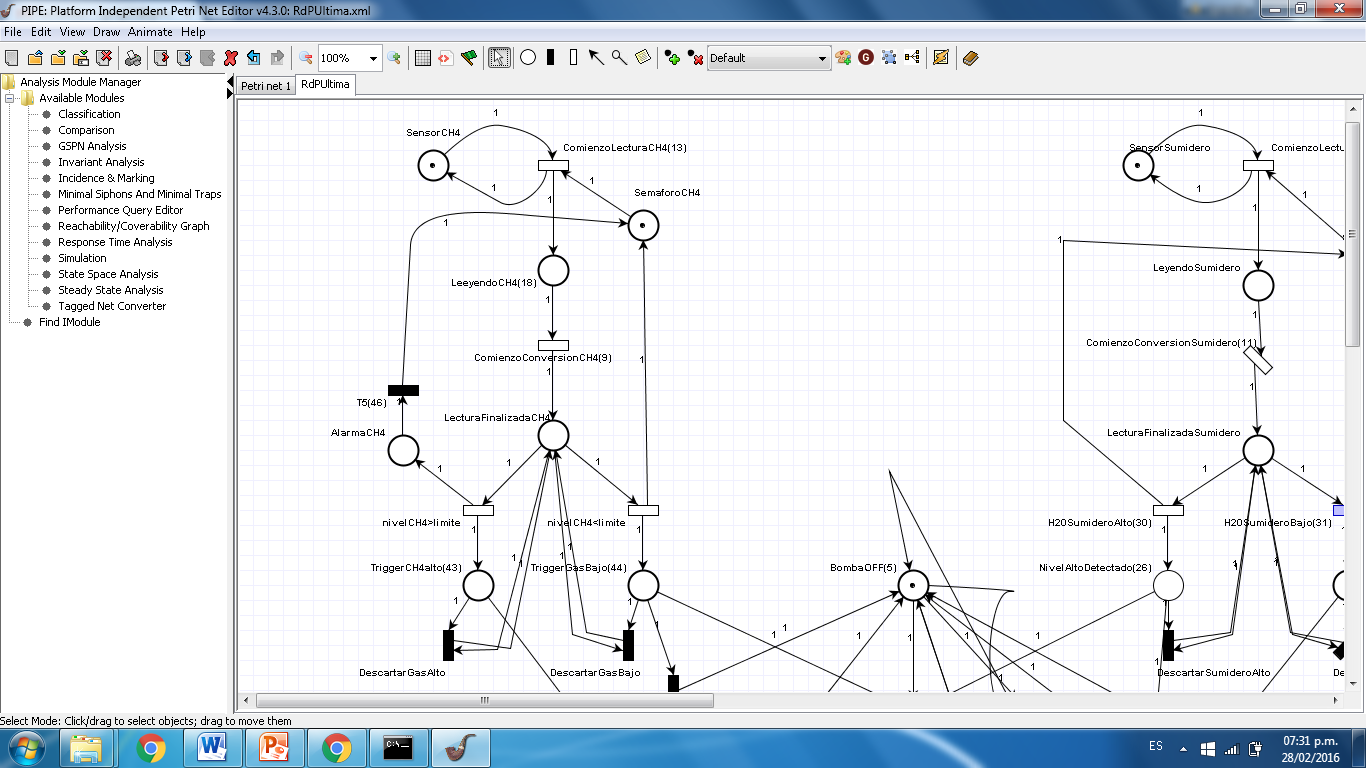
Diagrama de casos de Uso

****

*Caso de uso de prendido de bomba:*

|  |  |
| --- | --- |
| **Caso de Uso** | Prendido de bomba automático |
| **Fuentes** | Petrolera S.A |
| **Actor** | Sensor CH4, Sensor Sumidero, Carrito |
| **Descripción** | Prendido de bomba automático a través de la recolección de datos de sensores |
| **Flujo básico** | **1. Recolección de Dato Sensor CH4**  1.1: Lectura de dato  **1.2: Conversión de dato**  **1.3: Nivel por debajo del límite**  **2. Recolección de Dato de Sensor de Sumidero**  2.1: Lectura de dato  2.2: Conversión de dato  2.3: Nivel por encima del límite 3. Delay de prendido de la bomba |
| **Flujos alternos** | **Un flujo alterno al especificado provocaría que no se produzca el caso de uso especificado** |
| **Pre-condiciones** | 1. **Sensor en Funcionamiento** 2. Carritos en posición |
| **Post-condiciones** | **1. La bomba en funcionamiento** 2. Los carritos en proceso de llenado y vaciado |
| **Requerimientos trazados** | 1. **De funcionamiento** 2. Monitorización del Sistema |

*A partir de los diagramas de Actividades, de Caso de Uso y los requerimientos, se procede con el diseño de la Red de Petri. La misma se descompone en varias partes para una mejor comprensión. En el archivo XML se encuentra la red completa.*

****Red de Petri

**RdP del Sensor de CH4**

*Básicamente en esta parte de la red se generan datos en un cierto tiempo los cuales terminan influyendo en los estados de la bomba*

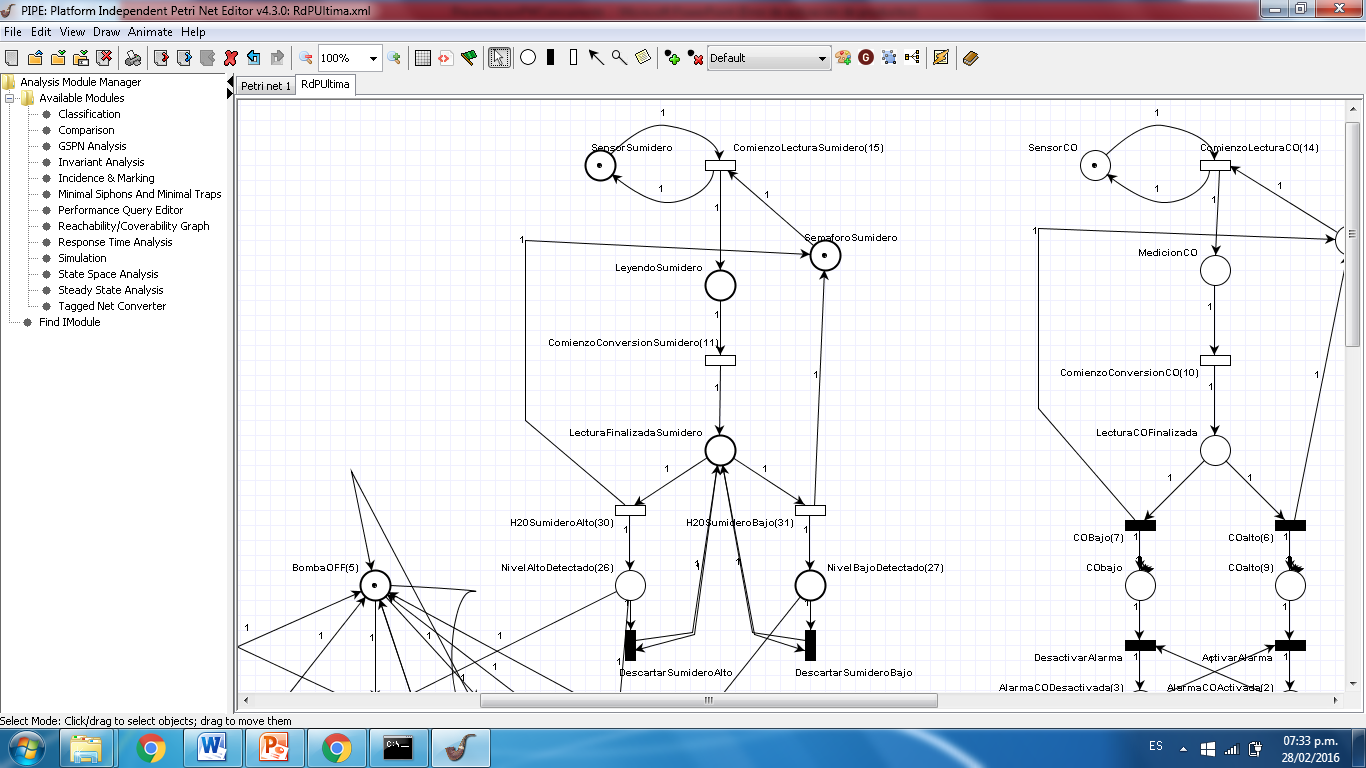
El semáforo permite evitar que se lea un dato nuevo si todavía no se ha procesado el anterior.

Las transiciones que detectan el nivel (alto o bajo) de CH4 se ejecutan posteriormente al descarte del dato anterior.

Si se detecta un nivel de CH4 alto, se activa una alarma y el usuario debe desactivarla.

Se deben tener en cuenta las plazas de GasAlto y GasBajo que influirán en el encendido y apagado de la bomba.

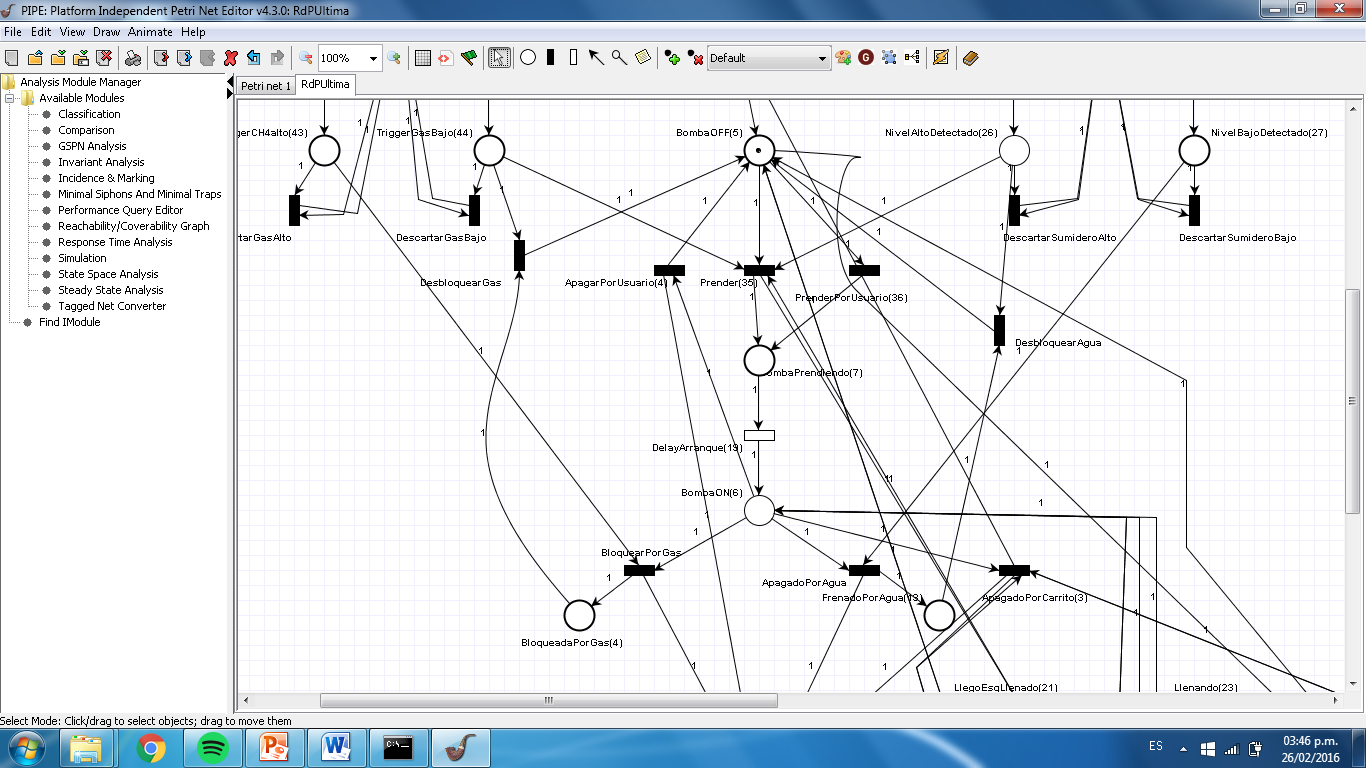
**RdP del Sensor Sumidero**

*****El sensor genera valores del nivel de Agua. Los cuales si son bajos apagan la bomba o si son altos la bomba sigue funcionando normalmente*

Se arma un sensor Sumidero con la misma lógica que el sensor de CH4. La única diferencia con el anterior es que no emite ningún tipo de alarma.

Más adelante veremos cómo influyen los datos que genera dicho sensor en la bomba.

**RdP de la Bomba**

****

La bomba se puede encontrar en 5 estados diferentes: OFF, prendiendo, ON, BloqueadaPorGas o Frenada por Agua. De esta manera, hay 1 token (recurso) que se mueve a través de dichos estados. Dicho de otro modo, la bomba nunca puede tener un token en ON y otro en OFF.

La bomba comienza en OFF y se puede prender:

* Por Usuario
* Automáticamente

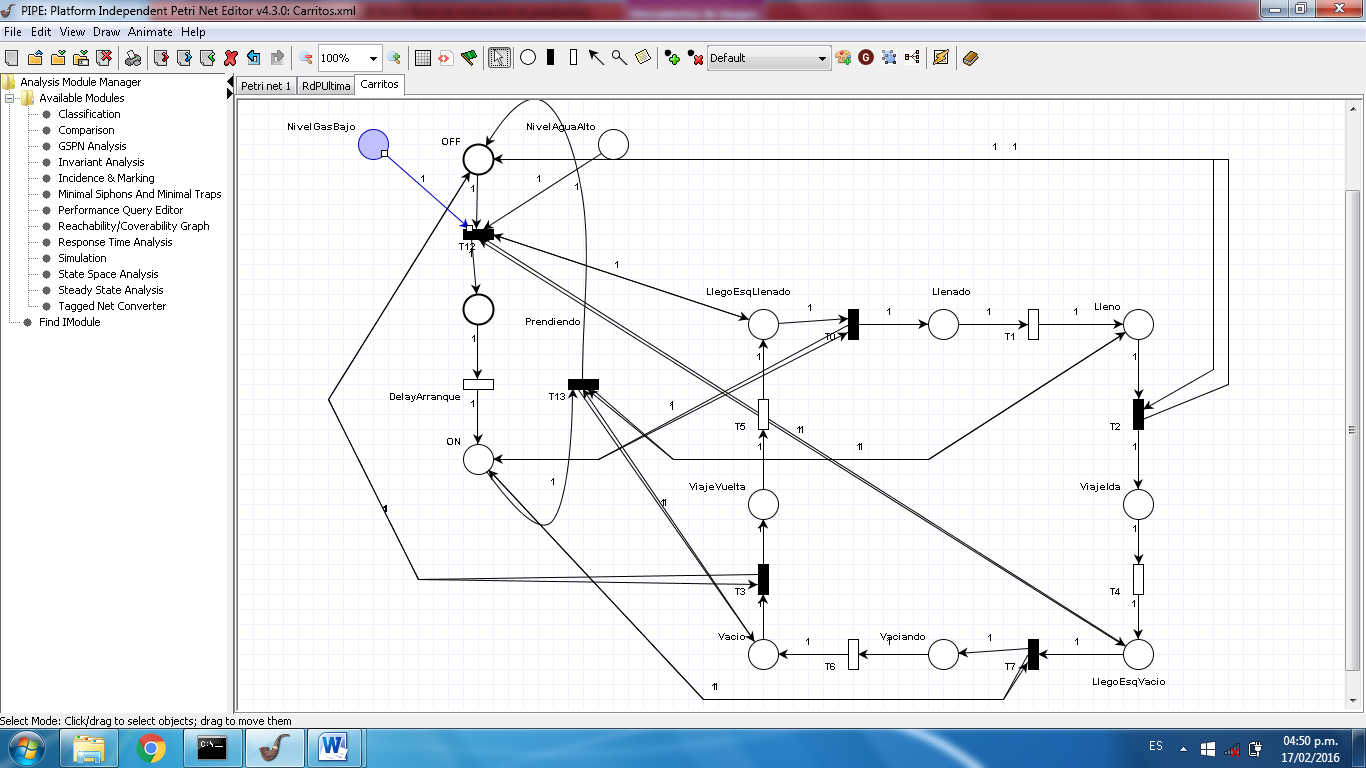
Automáticamente implica que el nivel de CH4 sea bajo, el nivel de agua alto (dichos estados ya los vimos en los sensores) y además los carritos deben estar en la posición adecuada (ya lo explicaremos más adelante).

Una vez que la bomba está encendida, si ocurren ciertos eventos, puede que vuelva a apagarse o puede ser llevada a estados intermedios que aseguran que los niveles vuelvan a la normalidad.

Por ejemplo: si la bomba está en ON y el sensor detecta un nivel alto de CH4, entonces se va a BloquearPorGas que en caso de que el nivel de gas vuelva a ser estable, le permitirá a la bomba volver al estado inicial OFF para que comience a funcionar nuevamente.

Cabe mencionar que el Usuario tiene la posibilidad de apagar la bomba manualmente si lo desea

**RdP Carritos**

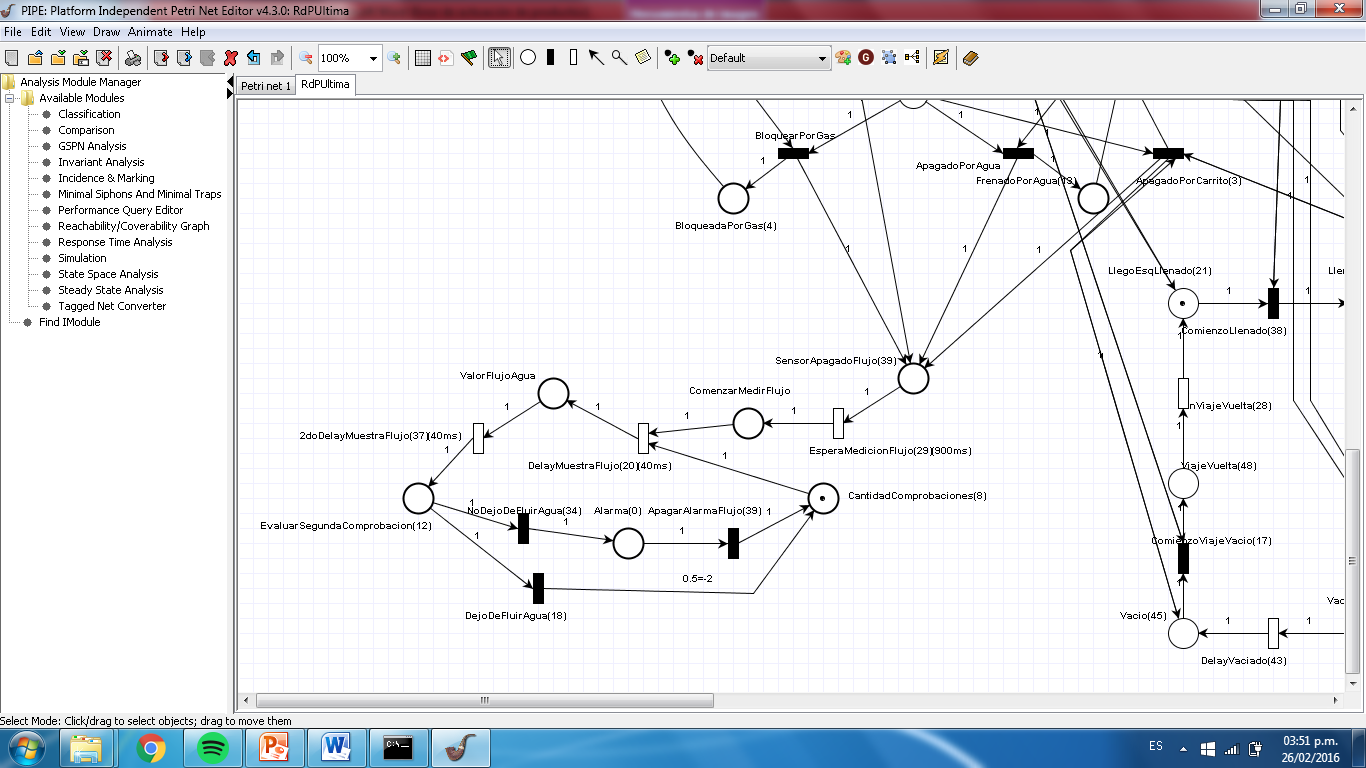
****

Los carritos pueden encontrarse: viajando o en las esquinas. Una esquina es la de vaciado, la cual el carrito puede recién haber llegado, vaciando o vacío. La otra es la esquina en donde se encuentra la bomba, por lo cual el carrito puede haber recién llegado, puede estar llenándose o puede estar lleno.

Hay que mencionar que en todo el circuito tendremos siempre 2 tokens (que representan los carritos) y nunca aumenta ni disminuye la cantidad.

Algunos casos:

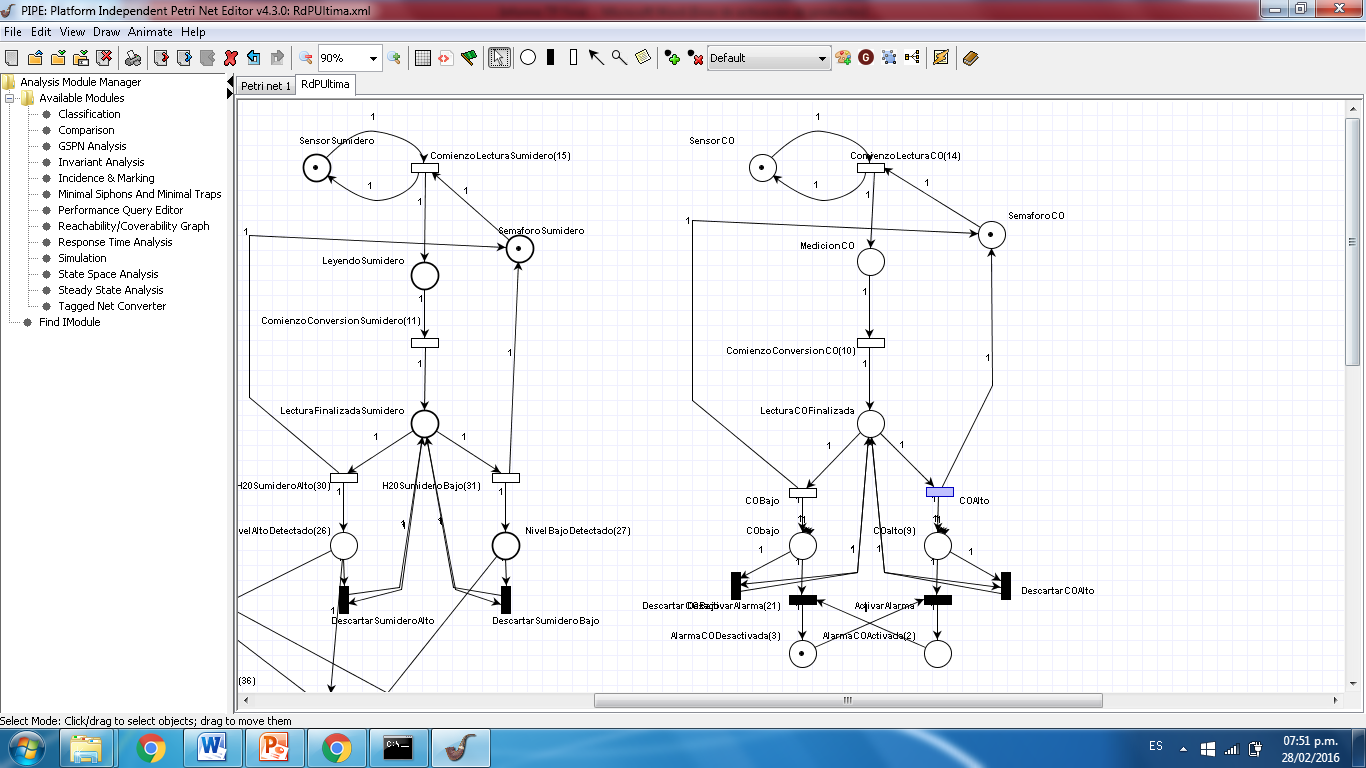
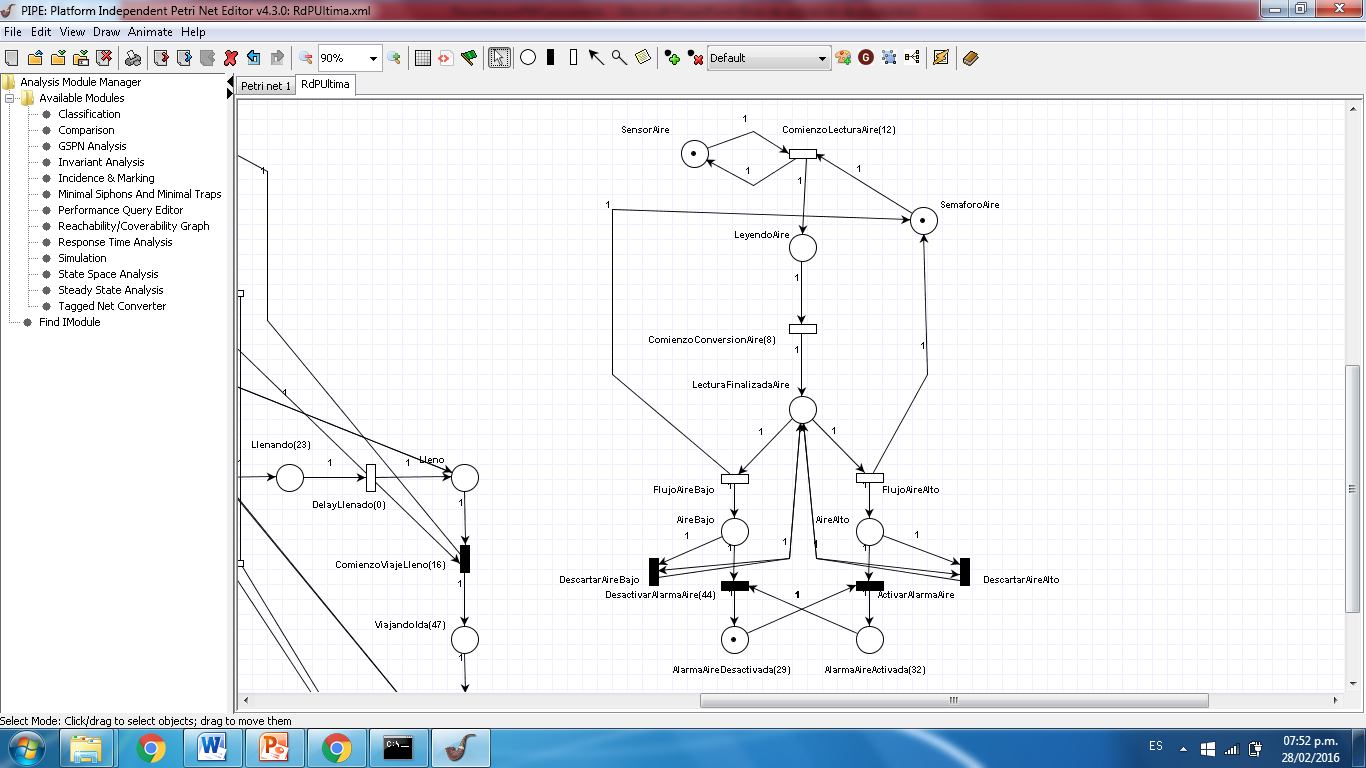
* En el momento que los 2 carritos llegaron a sus respectivas esquinas, la bomba se enciende. Consecuentemente comienza a llenar. Mientras uno se está llenando, el otro se está vaciando (cabe aclarar que se vacía mas rápido de lo que se llena).
* Terminado el vaciado y el llenado, la bomba debe dejar de funcionar, por lo cual se pone automáticamente en OFF y además los carritos comienzan a viajar.
* Los viajes de ida y vuelta a la bomba demoran un cierto tiempo y se vuelve al estado inicial.

**RdP Flujo de Agua**

Cuando la bomba se apaga se debe comprobar si realmente ha dejado de fluir agua.

En caso que no dejo de fluir agua suena una alarma de error. En caso contrario, dejo de fluir agua correctamente.

**RdP Sensor CO y Sensor Aire**

Funcionan de manera similar a los otros sensores anteriormente vistos (CH4 y Sumidero): se leen datos periódicamente del sensor los cuales pueden ser tomados como altos o bajos. Si el nivel de CO o Aire es alto, se activa una alarma. En caso de que sea bajo, se desactiva la alarma.

Aclaración: El programa cuenta con transiciones:

* Externas temporales
* Externas sin tiempo
* Automáticas temporales
* Automáticas sin tiempo

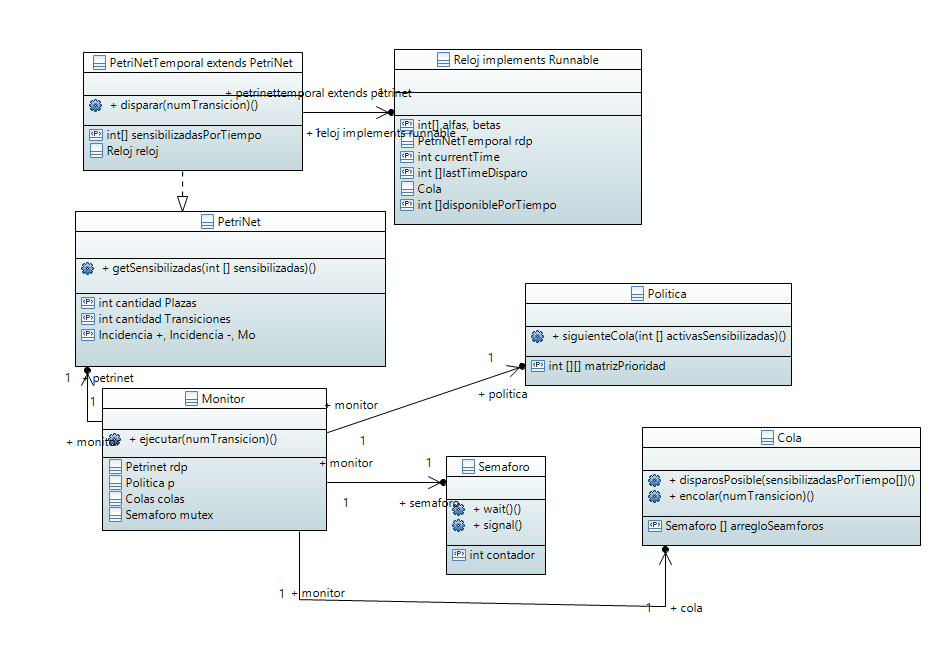
Las transiciones externas son aquellos disparos generados por los sensores y el usuario. Para el caso de los sensores son externas temporales y el caso de un botón para desactivar una alarma es una externa sin tiempo

Las transiciones Automáticas se desencadenan luego de los disparos externos y pueden ser por ejemplo el cambio de estado de la bomba “prender” (sin tiempo) o “delayArranque” (temporal)

*Una vez construida la RdP podemos obtener las matrices de incidencia positiva y negativa y el marcado inicial que serán de utilidad para el desarrollo del código.*

*A continuación se muestran algunos diagramas UML que explican el funcionamiento del programa.*

UML

**Diagrama de Clases** 

El programa cuenta con un monitor que nos permite conseguir la exclusión mutua mediante un Semáforo mutex y la coordinación de procesos.

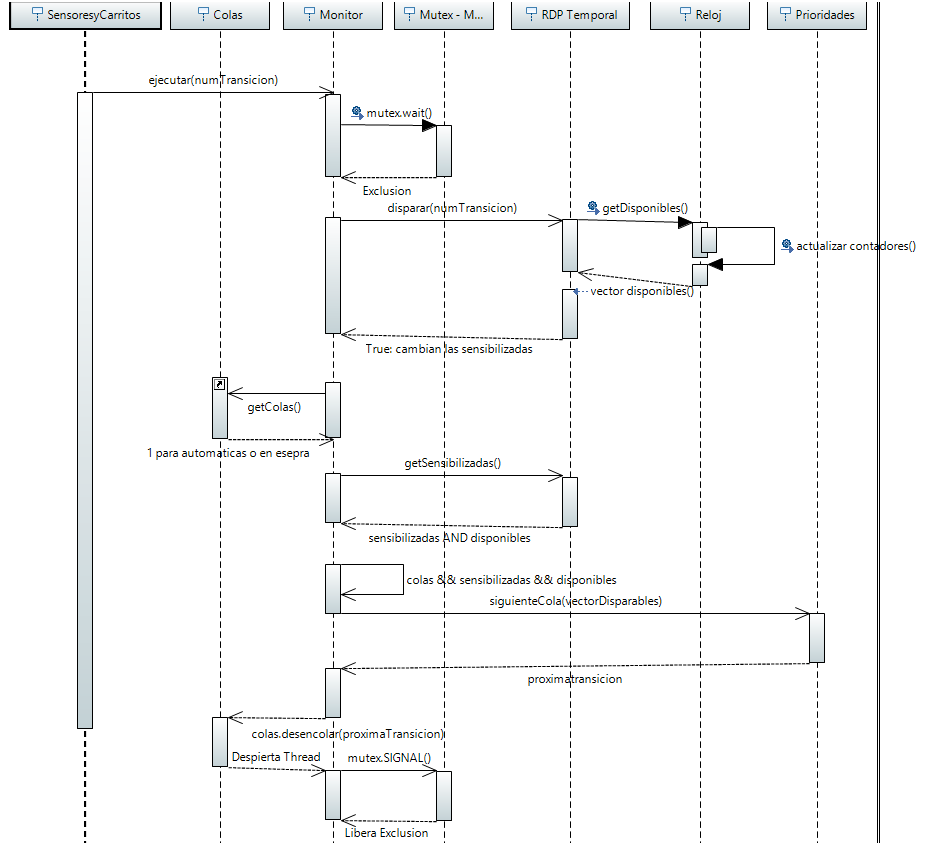
El monitor crea una RdP temporal la cual a medida que se ejecuten las transiciones, esta se va modificando. La petición de ejecución de transiciones es dentro del monitor. Dicho de otro modo, el monitor determina si una transición puede ejecutarse o no a partir de la RdP.

La RdP temporal tiene un reloj, que se encarga de sensibilizar y desensibilizar transiciones en cuanto se cumplan o no los tiempos.

Dentro del monitor tenemos una Politica p, la cual nos indica la prioridad de la transición. Las prioridades pueden ser modificadas por el usuario manualmente y en tiempo de ejecución del programa.

Para cada transición tenemos una cola y para cada transición externa, un semáforo. En caso de que la transición sea automática, no tendremos un semáforo en dicha cola. (ver pag 5: dif entre transiciones externas y automáticas)

**Diagrama de Secuencia: 1 o más de 1 transición sensibilizada**



**Diagrama de secuencia: ninguna transición sensibilizada**

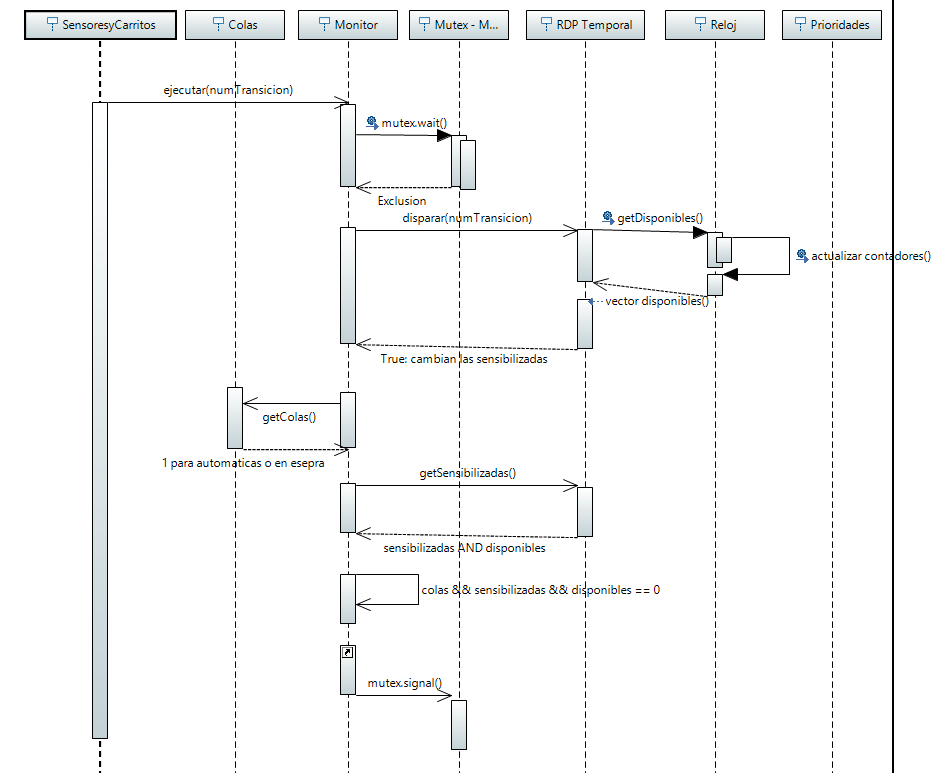


Diagrama de secuencia: No se puede disparar la red de Petri

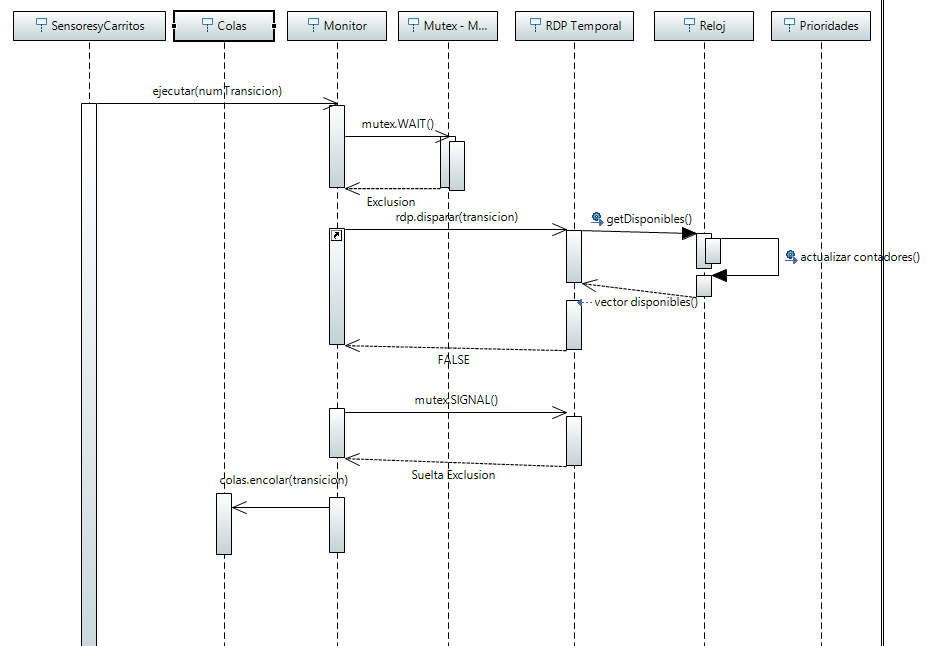


Diagrama de Secuencia: Reloj transición automática

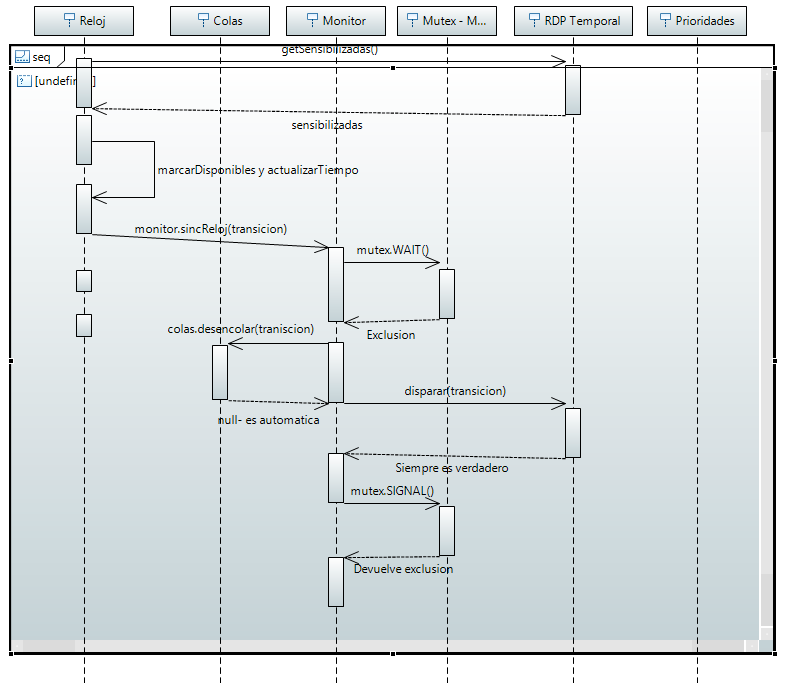
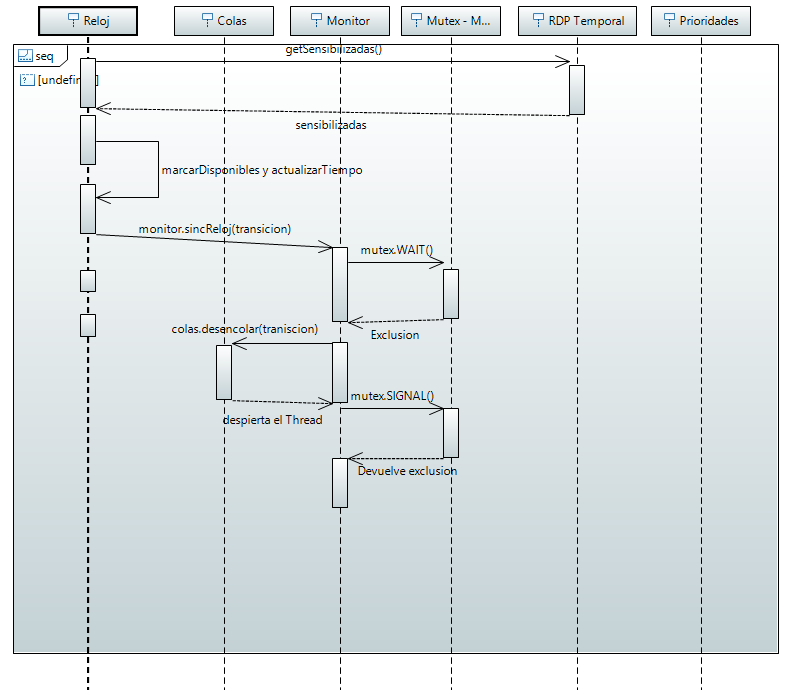
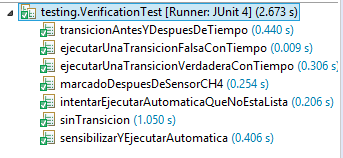


Diagrama de Secuencia: Reloj transición no automática



Tests

*Se realizan tests para comprobar que el programa responda de la misma forma que lo hace la RdP.*

****VerificationTests**

Estos tests verifican la lógica de la red temporal. Son los más básicos, pero si uno de ellos falla, fallará la Red en su totalidad. Se explicaran brevemente lo que hacen algunos test:

transicionAntesYDespuesDeTiempo: la transición no se debe poder ejecutar antes de que se cumpla el tiempo. Una vez llegado el tiempo, se sensibiliza y se puede ejecutar.

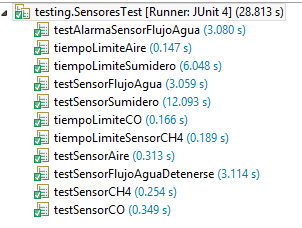
marcadoDespuesDeSensorCH4: vemos si al ejecutar una transición se modifica el marcado de la RdP.

intentarEjecutarAutomaticaQueNoEstaLista: la RdP no debe dejar que se ejecuten transiciones automáticas si no están listas

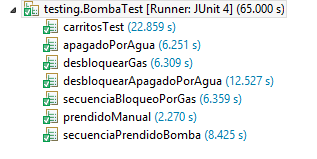
sinTransicion: se comprueba el marcado de la red de Petri si no ejecutamos transiciones (debería mantenerse sin cambios)

sensibilizarYEjecutarAutomatica: probamos que ejecutando una transición, esta puede desencadenar que se ejecuten transiciones automáticas.

**SensoresTest**

Mediante estos tests comprobamos que todos los sensores lean y conviertan datos correctamente, desde el punto de vista lógico y temporal. Son tests mas generales que los anteriores ya que probamos una parte del sistema

Básicamente, lo que hacen los tests es ejecutar ciertos disparos de los sensores y comprobar que el marcado llegue adecuadamente.

**BombaTests**

Una vez comprobada la lógica de la Red y los sensores individualmente, queda probar que ocurre si combinamos ejecuciones de sensores. Esto desencadena que la bomba cambie de estado y que los carritos comiencen moverse. Son los tests más generales de todos y se hacen de la misma manera: ejecutando transiciones, esperando tiempos en el caso que sea necesario y comprobando el marcado que se modifique correctamente.

Conclusión

Haber afrontado el problema inicial a través de las etapas del diseño y utilizando los beneficios de las Redes de Petri fue fundamental a la hora de poder tener una estructura flexible a los cambios, de facilidad de comprensión y mantenimiento. También simplificó a la hora de escribir el código debido a que la utilización de diagramas UML aporta las soluciones en cuanto a la estructura funcionalidad, y secuencia del mismo.