Control de una bomba de achique

Pedro Rosselló Perelló, Francisco Manuel Pozo Pérez

Abstract— **En este artículo se presenta la simulación de un sistema de drenaje de una mina. El sistema simula la presencia de una bomba de achique, unos sensores de gases, un sistema de alarmas y una interfaz para el operario. Cada uno de estos componentes es instalado sobre un microcontrolador dsPIC30F4011.Un sistema operativo de tiempo real (RTOS) llamado FreeRTOS es el encargado de dar soporte a las aplicaciones que simulan dichos componentes. A su vez, se utiliza el protocolo de comunicaciones CAN para para el envío y recepción de mensajes entre los diferentes nodos del sistema.**

I. Introducción

En este proyecto se ha emulado un sistema simplificado de control de una bomba para una mina. El sistema se utiliza para bombear a la superficie el agua extraída del sumidero del pozo de una mina. El principal requisito de seguridad es que la bomba no debe funcionar cuando el nivel de gas metano o el nivel de monóxido de carbono en la mina sea alto, debido al riesgo de explosión e intoxicación de los operarios respectivamente.

El sistema está separado en cuatro nodos diferenetes, el Nodo Bomba (Nodo B) encargado de gestionar el funcionamiento de la bomba de achique, el Nodo Nivel de Gases (Nodo NG) que monitoriza los niveles de gases dentro de la mina, el Nodo Alarmas de Gases (Nodo AG) y por último, el Nodo Monitor (Nodo M). Estos nodos mantienen una comunicación constante intercambiando información tal y como muestra la figura siguiente.

A. Nodo Bomba

Este nodo esta formado por un computador que recibe constantemente el nivel del agua del pozo desde el transmisor de una boya. Dicho computador aplica un algoritmo PID *(Proporcional Integral Derivativo)* sobre el nivel del agua para calcular la potencia a la cual debe trabajar la bomba extractora y enviar dicha información directamente a la bomba extractora de agua.

En este nodo también se encuentra un comunicador encargado de recibir órdenes para incrementar o decrementar la potencia de funcionamiento de la bomba por parte del operario desde el nodo M, aplicándose dichos cambios directamente sobre el estado actual de la bomba (relativo), al igual que también una señal de parada de emergencia proveniente del nodo NG.

Siempre y cuando el nivel del agua permanezca por encima del 35% y por debajo del 65% el operario podrá incrementar o decrementar la potencia de la bomba, de lo contrario unicamente ésta se regirá por la potencia calculada por la tarea computador. Todo esto se aplicará mientras los niveles de monóxido de carbono y metano están en unos niveles no críticos. Si alguno de los niveles anteriores alcanza un nivel crítico, la bomba se detendrá automáticamente sea cual sea el resultado del PID o las órdenes del operario.

El comunicador envía en todo momento el nivel actual del agua dentro del sumidero al nodo monitor para que el operario pueda consultarlo.

B. Nodo Nivel de Gases

El nodo NG tiene un computador parecido al que tiene el nodo B, este computador recibe constantemente los niveles actuales de gas metano y de monóxido de carbono de dentro de la mina. Si alguno de estos gases llega a un nivel crítico, el computador informará mediante un mensaje al comunicador del mismo nodo para que envíe un mensaje de parada al nodo B y uno para la activación de las alarmas de alerta al nodo AG.

Mientras alguno de los gases permanezca en nivel crítico, el computador sigue necesitando constantemente el nivel del mismo para que en el momento en el que deje de estar en dicho nivel, permita a la bomba su normal funcionamiento y el mensaje al nodo AG para que dejen de sonar.

C. Nodo Alarma de Gases

Este nodo es el encargado de activar y desactivar las alarmas por nivel crítico de alguno de los gases. Si el nodo NG detecta que alguno de los niveles es crítico, el nodo AG recibe el mensaje correspondiente y activa la alarma adecuada. Cuando recibe el mensaje de apagar las alarmas debido a que los niveles de los gases vuelven a estar en un nivel no crítico, detiene los avisos de alarma.

En el caso de que el operario desde el nodo M decida silenciar las alarmas, el nodo AG silenciará pero no apagará los avisos por nivel crítico de monóxido de carbono o metano.

D. Nodo Monitor

Este nodo únicamente simula una serie de pantallas e indicadores que podría tener un operario en su ordenador. Recibe el nivel del agua del sumidero del nodo B y los niveles de gases junto a un aviso de si alguno está en nivel crítico por parte del nodo NG.

Desde este nodo se envía un mensaje para silenciar, que no detener, las alarmas del nodo AG, y también unas señales para incrementar o decrementar la potencia de succión de la bomba de extracción de agua del nodo B.

II. Plataforma

A. Hardware

La red esta formada por cuatro nodos que se describen a continuación conectados a un solo bus de comunicaciones serie con cuatro conectores DB9. Las placas utilizadas se dividen en tres componentes, cada uno integrado en el siguiente.

Cada uno de los nodos mencionados en el apartado anterior están representados por una placa UIB.PC104 de ingenia-cat S.L. Que permite al módulo iCM4011 actuar sobre las entradas y salidas conocidas comúnmente como interfaces de usuario. En el módulo iCM4011 hay integrado un Microcuntrolador dsPIC30F4011.

B. Software

Para el SOTR se utiliza la librería de FreeRTOS. FreeRTOS es un líder de mercado de los SOTR creado por *Real Time Engineers Ltd.*. Tiene un desarrollo profesional, un control de calidad estricto, robusto, con soporte además de su uso sea gratuito para productos comerciales sin ninguna obligación de exponer el código fuente del propietario del programa.

Es un sistema multitarea apropiativo, es decir, las tareas compiten por ejecutarse en el procesador.

El procesador planifica qué tarea debe ejecutarse en cada momento mediante un componente funcional llamado *Scheduler* el cual reparte el tiempo disponible del microprocesador entre las tareas que están disponibles para su ejecución.

Una tarea puede encontrarse en cuatro posibles estados, entre ellos el de disponible (*Ready*), los cuales son explicados a continuación junto a un esquema con sus posibles transiciones en la figura 2.

1) *Running*: Cuando la tarea se esta ejecutando, está utilizando el procesador.
2) *Ready*: Las tareas en estado *Ready* son las que están preparadas para ejecutarse, pero no están siendo ejecutandas porque una tarea diferente de igual o mayor prioridad se esta ejecutando.
3) *Blocked*: Una tarea se dice que está en el estado *Blocked* si está a la espera de un evento temporal o externo. Las tareas también pueden estar en este estado mientras esperan los eventos de una cola o un semáforo. Estas pueden tener un *timeout*, después del cual son desbloqueadas. Las tareas bloqueadas no compiten por el procesador.
4) *Suspended*: Las tareas en este estado no compiten por el procesador, al igual que en el estado *Blocked*. Las tareas entran al estado *Suspended* cuando reciben el comando *vTaskSuspend()* y salen de este estado con la función *vTaskResume()* ya que no se puede definir un *timeout*.

Una tarea en estado *Running* devuelve el control del procesador al *Scheduler* cuando se llama a la función *vTaskDelay(time)* entonces la tarea pasará a estado *Blocked* el tiempo pasado por parámetro, o con la función *taskYIElD()* que pasa directamente al estado *Ready*. El SO proporciona herramientas de concurrencia y comunicación entre las tareas como son los semáforos, cola de mensajes y *mutex*.

Existe archivo de configuración llamado *FreeRTOSCongif.h* en el que se pueden configurar por el usuario los parámetros necesarios para el hardware en particular y los requerimientos de la aplicación.

III. Desarrollo

En este apartado se detalla las funciones más importantes utilizadas en el proyecto separadas en mensajes definidos y los distintos nodos del sistema.

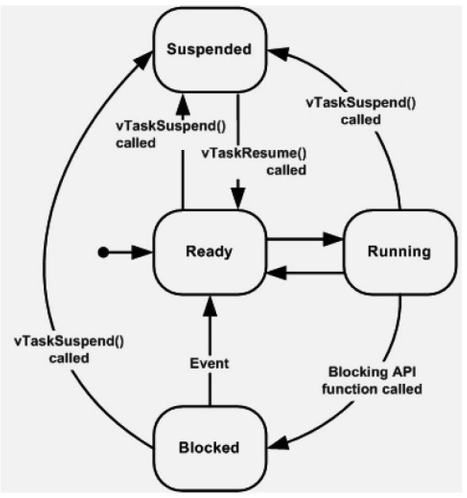

Fig. 1. Estados FreeRTOS

A. Mensaje Definidos

La red de mensjes definidos es una red CAN en la que los mensajes tienen un identificador por el cual todos los nodos conectados a la red pueden saber quién es el emisor y qué datos contiene dicho mensaje.

En esta práctica tenemos siete identificadores. Los consumudores, productores y la información de los mismos, se detallan en la tabla 1:

Productor	Información	Consumidores
Nodo Bomba	Nivel de Agua	Nodo Monitor/Sensores
Nodo Bomba	Potencia de la Bomba	Nodo Monitor/Sensores
Nodo Sensores	Nivel Crítico Metano	Nodo Bomba/Alarmas
Nodo Sensores	Nivel Crítico Monóxido	Nodo Bomba/Alarmas
Nodo Sensores	Niveles de Gases	Nodo Monitor
Nodo Monitor	Silenciar Alarmas	Nodo Alarmas
Nodo Monitor	Incremento Potencia	Nodo Bomba

TABLE I
MENSAJES DEFINIDOS

Gracias a la función *CANConfigure* se configura cada uno de los nodos para recoger la información de únicamente los mensajes cuya información necesiten para su correcto funcionamiento.

B. Introducción a Nodos del Sistema

Todos los nodos del sistema están divididos en una serie de tareas de FreeRTOS que se ejecutan periódicamente controladas por el *Scheduler* de cada nodo.

Todos los nodos del sistema tienen una tarea comunicador, encargada de recibir los mensajes FreeRTOS de su nodo y enviarlos a la red CAN para que los nodos que necesiten la información de dicho mensaje. Los nodos recogerán el mensaje CAN mediante su propia tarea comunicador, que a su vez la enviará mediante mensaje FreeRTOS a la tarea que necesite dicha información.

C. Nodo B

Esta es la aplicación más compleja del sistema. No por la cantidad de tareas que tiene sino por las funciones del computador, en especial la función de cálculo de potencia de la bomba.

Para la función *Do_PID* es necesario que se definan principalmente cuatro variables (Kp, Kd, Ki y Tiempo). La variable Kp (proporcional) determina la reacción del error actual, la variable Kd (derivativo) determina la reacción del tiempo en el que el error se produce y por último, la variable Ki (integral) reduce el error estacionario. Modificando estas variables se ajusta la salida.

1) Tarea Actuador: El actuador o bomba de extracción es la tarea que simula el funcionamiento de la bomba de la mina. Empieza esperando un mensaje de FreeRTOS que proviene de la tarea computador con el nivel de potencia a la que debe trabajar si no ha recibido el mensaje FreeRTOS proveniente de la tarea comunicador de detención por nivel crítico de gases.

2) Tarea Nivel de Agua: Esta tarea lee una variable global que simula el nivel del agua y lo envía mediante un mensaje FreeRTOS al nodo computador.

3) Tarea Computador: La tarea del computador permanece bloqueada hasta que recibe el mensaje con el nivel del agua y lo utiliza como argumento para la función *Do_PID* y la modifica si se recibe la orden de incrementar o decrementar la potencia de la bomba desde la tarea comunicador. Una vez obtenida la potencia total se envía mediante mensaje a la tarea bomba.

D. Nodo NG

La mayor complejidad de este nodo reside en la tarea computador ya que el resto de tareas son triviales.

1) Tarea Sensor CH4: Esta tarea lee una variable global que simula el nivel de CH4 y lo envía mediante un mensaje FreeRTOS al tarea computador.

2) Tarea Sensor CO: Al igual que la tarea anterior, esta tarea lee una variable global que simula el nivel de CO y lo envía mediante un mensaje FreeRTOS al tarea computador.

3) Tarea Computador: Ya que el computador precisa el nivel de ambos gases, empieza su ejecución esperando a recibir los mensajes que provienen de las dos tareas anteriores. Para realizar la simulación de la subida o bajada del nivel de los gases también precisa la potencia actual de la bomba ya que mientras esta está en funcionamiento, lo gases se irán acumulando en el interior de la mina.

Una vez recibidos el nivel de los gases, analizará si alguno de ellos se encuentra en un nivel crítico. En el caso de que anteriormente alguno de estos niveles fuera crítico y ahora deja de serlo, envía el mensaje de cambio de estado a la tarea comunicador.

E. Nodo AG

Este nodo es el más simple del sistema, ya que únicamente recibe mensajes a través del bus CAN y obra en consecuencia.

1) Tarea Alarma CH4: Esta tarea es la encargada de activar o desactivar el *buzzer* mediante la función *BuzzerPlay* simulando el sonido de una alarma. El estado de la alarma también se muestra por el LCD mediante la función *LCDWriteString* si esta activada o desactivada ya que se puede recibir un mensaje FreeRTOS para silenciar la alarma. Silenciar las alarmas no las desactiva, por tanto seguirá mostrándose por pantalla el aviso de activada. Las señales activar y desactivar la alarma esta tarea la recibe por mensaje FreeRTOS.

2) Tarea Alarma CO: La tarea de control de la alarma de monóxido de carbono funciona exactamente igual que la tarea anterior.

F. Nodo M

El nodo operador simula la interfaz usuario-máquina del sistema. No permite realizar grandes cambios en el funcionamiento del sistema, pero si una constante monitorización.

1) Tarea Incrementar Potencia: Espera que se haya apretado el botón correspondiente al incremento de potencia en la bomba mediante la función *PushButtonRead* (que realiza un *polling* de teclado). Si el botón es el correcto, envía un mensaje FreeRTOS a la tarea comunicador con la orden de incremento.

2) Tarea Disminuir Potencia: Esta tarea funciona de manera equivalente a la anterior, la diferencia es el botón pulsado, y en el contenido del mensaje FreeRTOS, en el cual se envía la orden de disminuir la potencia.

3) Tarea Silenciar Alarmas: El funcionamiento de esta tarea es el mismo que las dos tareas anteriores salvo que se espera la pulsación de un botón diferente y que el contenido del mensaje FreeRTOS es el de silenciar alarmas.

IV. Estudio de la Memoria

Este estudio de memoria ha sido necesario después de comprobar las limitaciones que tienen las placas utilizadas. Nos dimos cuenta de dichas limitaciones al crear el primero de los nodos, el nodo B, que a medida que añadíamos tareas nuevas y dichas tareas se volvían más complejas el nodo empezaba a funcionar mal llegando a un punto en el que dejó de funcionar.

Cuando nos encontramos con este problema, analizamos las especificaciones de las placas descubriendo que tienen una memoria RAM de 2 Kbytes. Esto es un problema para FreeRTOS ya que es un sistema operativo de tiempo real caracterizado por su alto consumo de memoria. El mayor consumo de memoria viene dado por la necesidad de reservar una gran cantidad de memoria para las variables de uso interno de FreeRTOS y las variables globales del programa. Este espacio de memoria recibe el nombre de *heap* el cual consume aproximadamente un 75% de la memoria disponible.

En la tabla 2 se detalla el consumo de memoria RAM de cada uno de los nodos del sistema.

Como se puede observar en la tabla, todos los nodos, por simples que sean, necesitan más memoria de la que se dispone. Se podría haber optado por utilizar SALVO OS ya que precisa de mucha menos memoria o un modelo de dsPIC de 4 KB de memoria. Se ha descartado el SALVO OS ya que su versión

Nodo	Heap	Tareas	Colas	Scheduler	Variables	Total
Bomba	1500 B	676 B	300 B	236 B	60 B	2772 B
Gases	1500 B	676 B	200 B	236 B	40 B	2652 B
Alarmas	1500 B	507 B	200 B	236 B	20 B	2463 B
Monitor	1500 B	676 B	100 B	236 B	20 B	2532 B

TABLE II
CONSUMO DE MEMORIA RAM

gratuita SALVO Lite no acepta más de tres tareas en cada nodo y no se disponía de recursos para la versión completa de SALVO OS ni para unas placas con mayor capacidad.

V. PROPUESTA DE MEJORA

El sistema descrito en este artículo, es un claro ejemplo en el que la fiabilidad es uno de los factores más importantes a tener en cuenta en el desarrollo. Ya que el sistema controla una serie de elementos y factores que en caso de mal funcionamiento podría ocasionar grandes daños materiales e incluso la pérdida de vidas humanas.

Hasta el momento, solo se ha descrito el sistema básico en el cual cualquier fallo o parada en el funcionamiento de alguna de las tareas, nodos o comunicaciones derivaría en un fallo global del sistema.

Se han estudiado dos posibles mejoras para mejorar la fiabilidad del sistema, en los cuales, los fallos que actualmente inutilizan el sistema, ahora tienen una solución para que todo siga funcionando correctamente. Dichas mejoras se basan en el uso de redundancia para conseguir tolerancia a fallos.

Los modelos citados a continuación no han sido implementados por falta de tiempo y recursos pero a medida que el proyecto avanzaba, el problema de la fiabilidad era un factor cada vez más importante y por ello creemos que es necesario comentarlo.

A. Modelo Duplicado

Este modelo de fiabilidad se basa en la implementación de un nodo respaldo para cada uno de los nodos del sistema, esto obligaría a tener duplicados todos los sensores y actuadores.

El funcionamiento del nodo respaldo se basa en interrogar constantemente a su equivalente para comprobar si esta presente en el sistema. En el caso de que no se reciba una respuesta, el nodo respaldo entenderá que el nodo principal no esta operativo y tomará su lugar en el sistema.

Un nodo que retorne al sistema tras haber estado inoperativo, tomará el lugar del nodo de respaldo que ahora esta como nodo principal del sistema. Esto hace que si el nodo sustituido se arregla, siga en el sistema como un nodo de respaldo.

Este modelo de funcionamiento no solventa el hecho de que un nodo envíe información errónea, pero si soluciona la desconexión o perdida de cualquier nodo.

B. Model de Funcionamiento Crítico

El modelo de funcionamiento crítico se basa de la implementación de un número impar mayor que uno de replicas de cada uno de los nodos que en lugar de funcionar con nodos de respaldo, funcionan activamente todos ellos intercambiando sus datos y enviando al resto del sistema la moda de dichos datos.

Por ejemplo, en los sensores de gases si se implementa este modelo con 3 nodos, si dos de ellos detectan un nivel de gas metano crítico y el tercero lo detecta normal, se envía al sistema la alerta de nivel de gas metano crítico.

Los nodos implementan un contador de errores el cual se incrementa cada vez que el resultado de este nodo difiera de la mayoría. Este contador se reduce en una menor medida del incremento anteriormente citado por cada coincidencia con la mayoría. En el caso de alcanzar un valor límite de errores, el nodo se desconectará del resto del sistema y su voto no se toma en cuenta. Internamente dicho nodo sigue calculando sus errores y si sus resultados son correctos durante una serie de ciclos consecutivos, el nodo vuelve a las votaciones.

En el caso de que por alguna desconexión anterior el sistema quede con un número par de nodos del mismo tipo, dichos nodos tienen asignadas un peso relativo a ellos que en caso de empate, da un veredicto. Este peso puede ir en función de la calidad de los materiales (sensores, actuadores) del nodo.

Este modelo soluciona las carencias del modelo anterior, ya que no solo corrige la desconexión de un nodo, sino que también controla el mal funcionamiento de cualquiera de ellos.

VI. CONCLUSIONES

En esta práctica se muestra un ejemplo de implementación de un sistema distribuído de tiempo real en el que se ve el funcionamiento de las tareas y la sincronización entre ellas, lo cual es la base de cualquier sistema distribuido de tiempo real.

Este proyecto ha sido realizado utilizando el sistema operativo de tiempo real (SOTR) FreeRTOS, aunque podría haberse utilizado cualquier otro SOTR como SALVO OS, ERIKA Enterprise, Nokia OS, OPENROTS, Real-Time Linux u otros muchos más.

AGRADECIMIENTOS

A los autores les gustaría agradecer a los profesores Julián Proenza y Guillermo Rodríguez-Navas por enseñarnos las bases de la asignatura.

REFERENCES

[1] FreeRTOS - professional grade market leading RTOS. www.freertos.org/
[2] RTOS Concepts. http://www.chibios.org/
[3] *DsPIC30F4011/4012 Data Sheet, High Perfomance, Digital Signal Controllers*, Microship Tecnology Inc, 2005.
[4] Alan Burns y Andy Wellings, *Sistemas de tiempo real y lenguajes de programación*, 3rd ed. Madrid : Addison-Wesley, 2003.

Pedro Rosselló Perelló Estudiante de la titulación de Ingeniería Técnica en Informática de Sistemas en la Universitat de les Illes Balears (UIB).
shodaime@gmail.com

Francisco Manuel Pozo Pérez Estudiante de la titulación de Ingeniería Técnica en Informática de Sistemas en la Universitat de les Illes Balears (UIB).
franciscopozoperez@gmail.com