Práctica final:

UNIVERSITAT DE LES ILLES BALEARS

22422-Xarxes de Comunicacions Industrials

Darvoy Espigulé, Carlos 43 474 067 L

Garcías Ripoll, Sergio 43 232 885 S

Curso 23-24

GX512

Pareja 14

ÍNDICE

1. OBJETIVO

Nuestro objetivo en esta práctica es implementar un sistema de control distribuido sencillo, y para ello, tendremos que aplicar los conceptos adquiridos en clase sobre la programación de microcontroladores en arduino.

El lenguaje arduino tendrá sus dispositivos internos y periféricos (botones, timer, LCD y ADC). Además, mediante el bus de campo CAN y el puerto de serie, en la que utilizaremos pila de protocolos, capa física, capa de enlace y capa de aplicación y tendremos redes de comunicación.

En esta práctica tenemos que diseñar un depósito de agua con una válvula de entrada (VE) y una válvula de salida (VS), además tendremos dos sensores uno de nivel de líquido, que nos marcará la cantidad de agua que queremos que haya dentro del depósito, y otro de temperatura. Nuestro objetivo es que el depósito se mantenga en el nivel establecido en el sensor de nivel, mientras que la temperatura podrá ir desde el rango [-511,512], pero para que el sistema funcione tiene que mantenerse entre el rango de [-250,250].

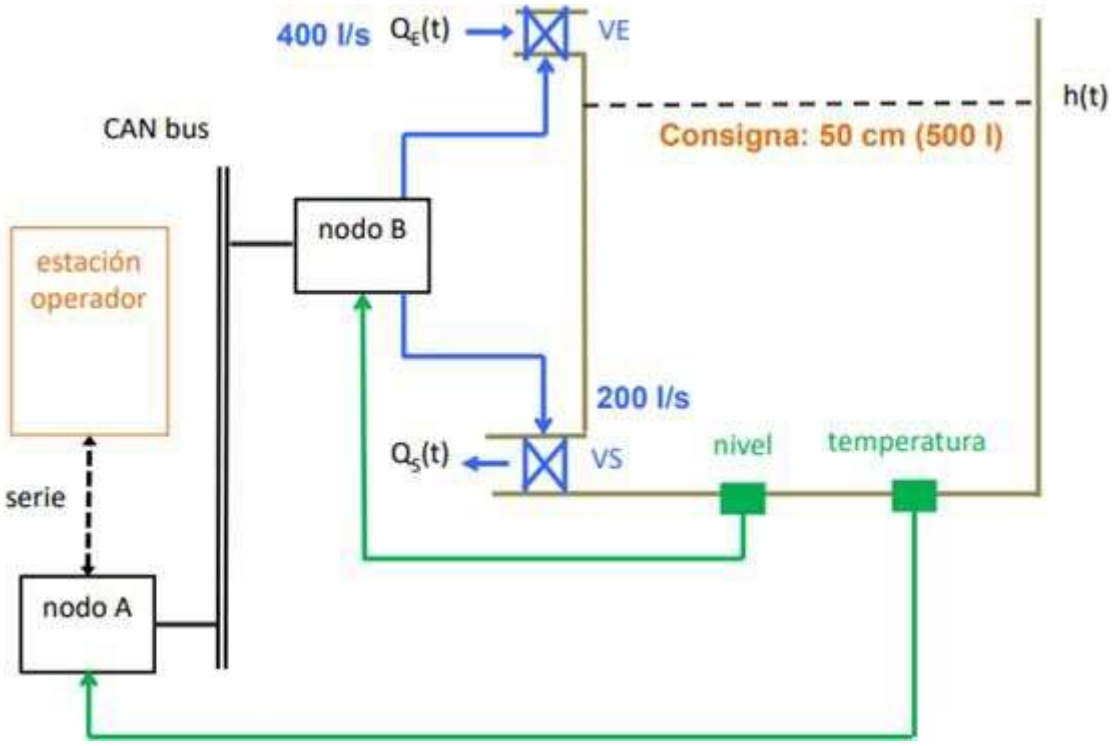

Figura 1. Esquema del sistema de control

La figura 1 de nuestro informe, ha sido copiada de la siguiente página web: https://ad.uib.es/estudis2324/pluginfile.php/80125/mod_resource/content/5/slidesPractFinal-2023-24.pdf

2. ARQUITECTURA DEL SISTEMA

Utilizaremos dos placas Arduino Mega 2560 que tienen un software de desarrollo Arduino IDE. Estas placas tienen una memoria flash ROM de 256 KB, una RAM de 8 KB, 54 E/S, además de tener una pantalla LCD, LEDS y un teclado.

Este tipo de placas presentan posibilidad de transmitir por bus CAN.

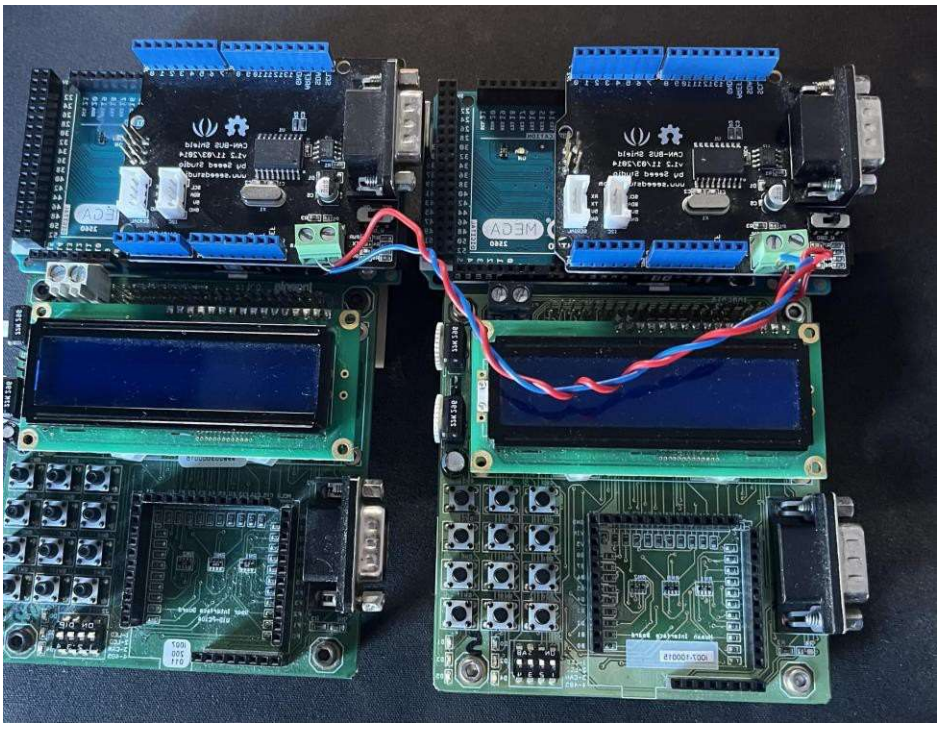

Figura 2: Imagen de las dos placas conectadas

3. HARDWARE

El hardware de estas placas consta de un puerto USB, una fuente de alimentación (entre 3.3V y 5V), un microcontrolador Atmel AVR de 8 bits y un conector GPIO (que sirve para que las placas puedan comunicarse)

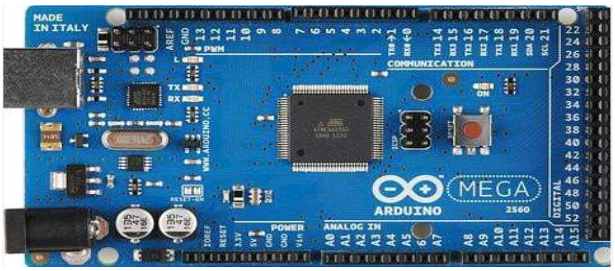

Figura 3: Imagen del hardware de la placa

4. SOFTWARE

El software que utilizaremos será el IDE Arduino. Además del código en arduino, tendremos que utilizar los siguientes ficheros y librerías:

SPI: Permitirá la comunicación entre el master y el slave.

HIB: Contiene las funciones relacionadas con la pantalla LCD, el ADC, el Timer5 y el teclado de las placas.

timer_config: Contiene los parámetros necesarios para cambiar la duración del slot, en esta práctica se ha tenido que recalcular para obtener los tiempos deseados.

mcp_CAN_UIB: Contiene las funciones para transmitir y recibir los mensajes a través de CAN.

Terminal: Contiene las funciones para interactuar con la UART.

defComunes: Contiene los ID de todas las funciones que se transmiten vía CAN.

5. TRAMAS CAN

En la siguiente tabla vamos a mostrar cómo se transmiten las tramas a través de CAN:

Identificador	Nodo origen	Nodo destino	Datos
0	A	B	Indica cómo se encuentran las válvulas
1	A	B	Indica que valor tiene que tener la válvula de entrada
2	A	B	Indica que valor tiene que tener la válvula de salida
3	A	B	Indica el valor del nuevo periodo de muestreo de nivel
4	B	A	Valor del nivel
5	B	A	Valor de la válvula de entrada
6	B	A	Valor de la válvula de salida

6. INTERFAZ

En la figura 4, podemos observar la UART, en la que podremos observar los siguientes valores:

Valor de consigna Operador: Podrá variar la consigna si pulsamos diferentes valores por teclado, si pulsamos la “a” valdrá 500mm, si pulsamos “b” valdrá 750mm, si pulsamos “c” valdrá 1000mm y si pulsamos “d” valdrá 1250mm.

Valor de consigna Objetivo: El valor que se busca alcanzar de nivel.

Nivel actual del depósito: La cantidad de agua que hay en nuestro depósito.

Temperatura actual del depósito: Nos da el valor de la temperatura.

Estado VE: Nos indica si la válvula está abierta “A” o cerrada “C”.

Estado VS: Nos indica si la válvula está abierta “A” o cerrada “C”.

Alarma: Nos indica si la temperatura del sistema se encuentra entre el rango de seguridad. Si se encuentra en el rango de seguridad, el modo de operación será normal, sino será alarma.

Duración Slot: Podrá variar el tiempo si pulsamos diferentes valores por teclado, si pulsamos la “h” valdrá 10ms, si pulsamos “j” valdrá 20ms, si pulsamos “k” valdrá 50ms y si pulsamos “l” valdrá 100ms, esto también cambiará en relación 0.5 el periodo de muestreo de la temperatura.

Periodo Muestreo: Nos indica el tiempo del periodo de muestreo en ms.

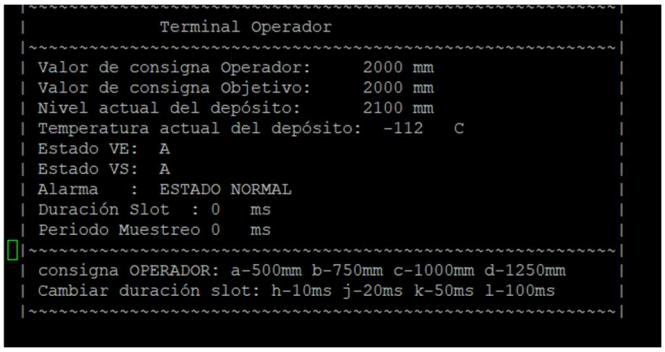

Figura 4: Imagen de la interfaz

Figura 5: Imagen de las dos placas en funcionamiento

7. PROTOCOLO

Emplearemos un TDMA (Time Division Multiple Access) en el cual cada turno comprende un slot (intervalo de tiempo). Nuestros nodos se comportan como un Master/Slave, en los cuales el nodo A será el Maestro y el nodo B el esclavo. Tal como hemos visto en clase, el nodo A enviará una trama al B, este lo recibirá y contestará al nodo A si tiene algo que transmitir.

Utilizaremos interrupciones, que son señales que permiten que la CPU pueda cambiar el flujo de ejecución, para así poder controlar los sucesos que no se encuentran bajo su mando. Una interrupción ejecuta la ISR (rutina de servicio de interrupción), que puede ser mediante hardware o software. En nuestro caso, cada interrupción tiene una ISR asociada mediante el overflow del timer, el cual estará programado para que pase cada 100ms.También utilizaremos el Hook, que es una función que se invocará desde la ISR

Un ADC (Analog to Digital Converter) es un dispositivo electrónico que nos permite digitalizar una señal analógica para que pueda ser procesada digitalmente. Nos permitirá procesar la información del entorno mediante sensores. Tendremos 16 entradas analógicas, ya que tenemos 16 pines. Utilizaremos una de estas entradas analógicas para determinar la temperatura del depósito.

Ahora vamos a explicar los diferentes pasos para llegar a la solución:

Lo primero que hay que hacer es que el nodo A transmita un mensaje a B para sincronizarse mediante el bus CAN. Para esto, hemos tenido que habilitar las interrupciones y recepciones de A y B y también hemos tenido que habilitar la transmisión de las tramas. También hemos tenido que inicializar las variables y los indicadores, donde se guardaran los datos de las tramas que necesitamos. Para acabar tenemos que fijarnos que el nodo A transmite correctamente y que el nodo B la recibe bien.

Luego hemos transmitido del nodo B vía CAN un slot con una trama con el valor del nivel del agua, con otra trama con el estado de la válvula VE y otra trama con el valor de la válvula VS. Y tenemos que fijarnos que el nodo A los reciba y los guarde correctamente.

Después el nodo A tendrá que calcular el estado de VE y VS teniendo en cuenta diferentes factores como son la consigna y el nivel actual del agua. A continuación, como el nodo B es que se encarga de abrir o cerrar las válvulas de entrada y de salida, el nodo A deberá enviar una trama con el estado actual de ambas válvulas para que así el nodo B pueda abrirlas o cerrarlas en el momento adecuado.

Lo siguiente que hemos realizado es la interfaz y la hemos hecho mediante los comandos de la UART. Hemos impreso cada uno de los valores antes mencionados

de los diferentes parámetros, como se puede observar en la figura 4. Además, hemos tenido que habilitar el teclado del ordenador para poder cambiar la consigna y la duración de paso. Lo que, para poder cambiar la duración de paso, primero tendremos que registrar la orden, después calcular el nuevo valor del sensor de nivel, y por último tendremos que solicitar vía CAN que el nodo B cambie el valor del muestreo de nivel al que hayamos calculado, ya que desde el nodo A no podemos hacerlo.

8. DECISIONES RELEVANTES

1. Delay en el tiempo de cierre de válvulas.

Cabe destacar que como el nivel se incrementa muy rápidamente, el nodo A tiene un delay a la hora de reaccionar y mandar una trama para terminar el nivel por lo que este sigue incrementando hasta recibir la orden y se pasa una distancia considerable (>30% aproximadamente 160 mm)

Hemos pensado que esto se podría solucionar implementando una condición en el slave de fin de cuenta pasando la consigna como una trama, pero esto haría el programa redundante puesto que no tendría sentido mandar cerrar las válvulas si ya conocemos el límite y no podríamos probar este protocolo tipo Master-Slave.

2. ISR:

Hemos buscado al máximo usar la ISR para todo lo relacionado con la comunicación a través de CAN entre nodos, evitando hacer prints innecesarios que podrían generar problemas indeseados.

3. LCD y UART:

Se han creado subrutinas para limpiar las variables muestreadas y que al sobrescribir estás en la interfaz de usuario no generase valores basura alterando el valor real de las variables.

9. PRUEBAS:

Prueba 1: Estado inicial y Temperatura:

Valor inicial:
El nivel empieza en 2000mm, al empezar la consigna es de 500mm, por lo que comenzaremos decrementando el nivel.

Temperatura:
Si excedemos el rango permitido [-250,250], entramos en estado de alarma, esto implica cerrar las válvulas hasta que se alcance una temperatura apropiada.

Leds:
Aprovechamos la prueba para comprobar que los leds se encienden/apagan correctamente, esto es una representación de una posible acción sobre unos actuadores o válvulas que se accionasen por el Arduino Slave o Nodo B.

Enlace: https://youtu.be/E8LsMMeHySQ

Prueba 2: Cambios de Consigna:

Consigna objetivo:
Se realiza este ejemplo para ver que el operador puede cambiar correctamente la consigna, esta se ve actualizada en la terminal

Enlace: https://youtu.be/WRH-HsuymeY

Prueba 3: Cambios de Tiempo y Periodo:

Duración de paso y tiempo de ronda:
Se ve modificada por el operador, esto se nota ya que la velocidad con la que varía el nivel cambia, así como la duración de los pasos
En el video empezamos pulsando la “L” por teclado → 50ms de periodo y 100 ms de duración de ronda lo cuál es muy lento y produce mucho error en el offset ± 100mm.
Luego pulsamos “H” → 5 ms y 10 ms respectivamente lo cual se nota porque hay menor error y mayor velocidad de incremento de nivel.

Una manera de ver esto es a través de el cambio de estado de los leds o augmentar la velocidad del video a video$_x$2

Enlace: https://youtu.be/XOBuqs0KVm8

[Se ha realizado videos de manera que quede explicado gráficamente]

10. CONCLUSIÓN:

A través de la práctica hemos mejorado nuestra comprensión sobre como funcionan los sistemas de comunicaciones y sus múltiples aplicaciones, concretamente en el sector industrial que es el que más nos concierne.

Es un hecho que conseguir comunicar al máximo nuestros sistemas a controlar permite una automatización eficiente y centralizado.

Por nuestra parte ha sido muy interesante ver como funciona el lenguaje Arduino, basado en C el cual ya conocemos y el uso de librerías que nos dan funcionalidades avanzadas si sabemos leerlas correctamente.

El bus de campo CAN y la comunicación vía puerto serie UART con el Operador nos ha parecido de real interés , puesto que por mi parte he tenido la suerte de conocer sistemas centralizados en barcos con buses de datos tales como NMEA2000 6 pines, SealTalk …

IMAGEN: NMEA 2000 Bus Cable

Tener un bus de datos donde converger todos los sistemas y luego aprovechar un display como interfaz con el usuario para que este controle todo el barco de gran eslora desde varios puntos es un factor muy interesante, hablando de temas como la seguridad y reducir consumos.

Por último, cabe destacar que el esquema utilizado lo hemos visto en la parte de teoría, TDMA (Time división multiple access) ha sido una manera muy práctica de entenderlo, ya que la simulación de un depósito de agua es un caso realista.