Computación Ubícua Curso 2014-2015

Practica Modbus

Enunciado

Programar una clase en C++ (`ModbusServer`) que implemente un servidor Modbus-RTU. Esta clase tendrá, al menos el siguientes métodos:

- `ModbusServer(byte id)` constructor donde se especifica el identificador del dispositivo.
- `std::vector<byte> peticion( std::vector<byte> recibido )` que recibe como parámetro un mensaje de petición Modbus-RTU y devuelve la respuesta correspondiente.

donde se ha definido previamente

```
typedef unsigned char byte;
```

El dispositivo completo ha de tener definido:

- Las primeros 20 salidas digitales con un valor inicial de 0 en las posiciones pares y de 1 las posiciones impares.
- Las primeras 20 entradas digitales:
  - Las 14 primeras tendrán un 1 si el valor del registro analógico de entrada correspondiente es par, y un 0 en caso contrario.
  - Los restantes deberán estar inicialmente a 0.
- Los primeros 10 registros analógicos de salida, con un valor inicial de 0, 4, 8, 12, ...
- Los primeros 20 registros analógicos de entrada, que deberán contener en cada momento:
  - Los 6 primeros: el año, el mes, el día, la hora, el minuto y el segundo, respectivamente.
  - Los 4 siguientes: el UID, el GID , el PID , el PPID del proceso.
  - Los 2 siguientes: los segundos y milisegundos de cómputo del proceso.
  - Los 3 siguiente: el contador de peticiones recibidas, en numero de bytes recibidos , el número de bytes enviados.
  - El resto debe estar inicialmente a 0 los pares y a 1111 los impares.

Tras la recepción de cada mensaje se deberá indicar, por la salida estándar, el valor de todos los registros implementados.

Parte 1

Implementar la funcionalidad de los registros (digitales y analógicos) de salida (funciones: 01, 05, 15, 03, 06 y 16). Se variará el contenido de los mismos como consecuencia de las peticiones de

escritura recibidas.

Parte 2

Implementar la respuesta de error adecuada cuando se reciba paquete:

- ID que no corresponde: no devolver nada.
- CRC incorrecto: no devolver nada.
- Solicite función no implementada: devolver error 01
- Solicite registro fuera de rango: devolver error 02
- Paquete tenga datos inválidos: devolver error 03.

Parte 3

Implementar la funcionalidad de los registros, digitales y analógicos, de entrada (funciones 02 y 04).

Parte 4

Implementar un procedimiento mediante el cual se pueda modificar, a través de la entrada estándar, el contenido de los 6 últimos registros, digitales y analógicos, de entrada. Este debe de funcionar concurrentemente con la atención a los mensajes recibidos.

Parte 5

Implementar Modbus/TCP.