(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 646 888**

(51) Int. CI.:

H04Q 9/00 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **06.10.2011 E 11184236 (5)**

(97) Fecha y número de publicación de la concesión europea: **27.09.2017 EP 2442585**

(54) Título: **Método para transmitir valores numéricos de una unidad de detección a una unidad de control**

(30) Prioridad:

12.10.2010 IT TO20100828

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
18.12.2017

(73) Titular/es:

LEONARDO S.P.A. (100.0%)
Piazza Monte Grappa 4
00195 Roma, IT

(72) Inventor/es:

CATANIA, GIROLAMO y
GALIZIA, GIOVANNI

(74) Agente/Representante:

ARIAS SANZ, Juan

DESCRIPCIÓN

Método para transmitir valores numéricos de una unidad de detección a una unidad de control

La presente invención se refiere a un método para transmitir valores numéricos de una unidad de detección a una unidad de control.

Como es conocido, los sistemas de detección, particularmente extendidos en la actualidad, que cada uno comprende una pluralidad de unidades de detección y al menos una unidad de control, típicamente están establecidos a una distancia de las unidades de detección. Cada unidad de detección comprende una respectiva unidad de procesamiento y uno o más sensores, que están conectados a la unidad de procesamiento y detectan la evolución en tiempo de las respectivas cantidades, que en lo que sigue se denominarán como “parámetros”. Las evoluciones detectadas se comunican a continuación por las unidades de detección a la unidad de control, que supervisa las operaciones realizadas por las unidades de detección y pueden llevar a cabo acciones específicas basándose en las evoluciones detectadas.

En detalle, considerando por simplicidad de descripción un sistema de detección formado por una única unidad de detección que comprende un único sensor, la respectiva unidad de procesamiento obtiene y almacena valores numéricos que corresponden al parámetro monitorizado por el sensor. Suponiendo que el sensor individual suministra una señal (no necesariamente una señal analógica) que indica los valores numéricos supuestos en tiempo por el parámetro, dichos valores numéricos pueden obtenerse de una manera periódica (síncrona), por ejemplo muestreando la señal suministrada por el sensor o si no consultando el sensor, como es de manera frecuente el caso de sensores de un tipo digital. Los valores numéricos pueden obtenerse además de una manera asíncrona; en este caso, típicamente es el mismo sensor el que notifica a la unidad de procesamiento variaciones del parámetro, tan pronto como se detectan.

Independientemente de cómo se obtienen los valores numéricos, la unidad de detección transmite a la unidad de control los valores numéricos obtenidos de esta manera de tal manera que los pone a disposición de la misma unidad de control. La transmisión de los valores numéricos obtenidos pueden resultar problemática; de hecho, los valores numéricos se obtienen por la unidad de procesamiento en un formato que es apropiado para el mismo sensor. Por ejemplo, en el caso donde el sensor es de un tipo digital y suministra valores numéricos expresados en un cierto formato, entendido como número de bits y tipo de codificación (por ejemplo, codificación de coma flotante), la unidad de procesamiento obtiene valores numéricos en ese formato dado. En consecuencia, de acuerdo con la precisión del sensor, puede ser necesario transmitir, para cada valor numérico, un número elevado de bits, con el consecuente aumento del tráfico intercambiado en el sistema de detección.

En el caso de un sistema simplificado que comprende una única unidad de detección, la transmisión, para cada valor numérico, de un número elevado de bits conlleva una pérdida de eficacia. Además, a medida que el número de unidades de detección aumenta, una transmisión de valores numéricos de este tipo puede resultar problemática, en particular en el caso de sistemas de detección en los que las unidades de detección comunican con la unidad de control por medio de canales de comunicación con baja tasa de bits, es decir, con una tasa de bits inferior a 256 Kbit/s.

En particular, la transmisión de valores numéricos puede resultar problemática en el caso donde la unidad de control y las unidades de detección comunican entre sí usando canales conmutados, es decir, en el caso donde las comunicaciones entre la unidad de control y cada unidad de detección tienen lugar antes de la configuración dinámica de un respectivo canal de comunicación de un tipo de punto a punto. De hecho, en este caso ocurre que, teniendo en cuenta la baja tasa de bits, la transmisión de un valor numérico genérico por una unidad de detección genérica requiere un respectivo tiempo de conexión durante el cual la unidad de detección genérica debe permanecer conectada a la unidad de control por medio de un respectivo canal conmutado. Durante el tiempo de conexión, que tiene una duración no despreciable, la unidad de control no puede conectarse hasta cualquier otra unidad de detección, con el consecuente retardo en la transmisión de valores numéricos obtenidos por otras unidades de detección, o incluso la pérdida de dichos valores numéricos.

En los contextos anteriormente descritos, puede sentirse por lo tanto la necesidad de limitar el tráfico transmitido por las unidades de detección a la unidad de control para transmisión de los valores numéricos obtenidos por las respectivas unidades de procesamiento.

El documento US6487695 desvela un sistema de prevención frente a fallos para conectar a unidad de entrada móvil con una unidad de control de máquina, que implementa un procesamiento previo redundante de las entradas de usuario en la unidad de entrada móvil. Las señales desde los medios de entrada se encaminan a dos procesadores de la unidad de entrada, y ambos procesadores codifican las entradas de usuario antes de transmitir las señales a la unidad de control. En la unidad de control, las entradas de usuario recibidas y codificadas se decodifican por dos procesadores y se comparan posteriormente entre sí para concordancia. Cuando se proporciona concordancia, los datos recibidos se procesan adicionalmente en el control. Si no hay concordancia, se ejecuta una rutina de error, apagando la herramienta de máquina.

El objetivo de la presente invención es proporcionar un método para transmitir valores numéricos que resolverá al menos en parte las desventajas de la técnica conocida.

De acuerdo con la presente invención, se proporciona un método para transmitir valores numéricos, una unidad de detección y una unidad de señor, como se define, respectivamente, en las reivindicaciones 1, 13 y 14.

Para un mejor entendimiento de la invención, las realizaciones de la misma se describen ahora, simplemente por medio de ejemplo no limitante y con referencia a los dibujos adjuntos, en los que:

- La Figura 1 es una ilustración esquemática de un sistema de detección;
- Las Figuras 2, 3, 5, 6, 8, 9, 11 y 14 muestran diagramas de flujo de operaciones de acuerdo con el presente método;
- Las Figuras 4, 7, 10 y 12 muestran esquemáticamente estructuras proporcionadas de acuerdo con la presente invención; y
- La Figura 13 muestra en formato hexadecimal una cadena de datos binarios.

La Figura 1 muestra un sistema de detección 1, que comprende una unidad de control 2, un medio de comunicación 4 y una pluralidad de unidades de detección 6, que no son necesariamente las mismas entre sí y están conectadas a la unidad de control 2 por medio del medio de comunicación 4. En particular, la Figura 1 muestra, a modo de ejemplo simplemente, tres unidades de detección 6. Además, una vez más de nuevo a modo de ejemplo, solamente una de las tres unidades de detección ilustradas se muestra en mayor detalle.

La unidad de detección 6 y la unidad de control 2 se sincronizan con respecto a uno y el mismo reloj de sistema. Además, cada unidad de detección 6 comprende un respectivo dispositivo electrónico periférico 8 y una pluralidad de aparatos de sensor 10, conectados al dispositivo electrónico periférico 8. A modo de ejemplo, la Figura 1 muestra tres aparatos de sensor 10. Incluso aunque no se muestre, cada unidad de detección 6 puede tener un número diferente de aparatos de sensor 10.

A su vez, cada dispositivo electrónico periférico 8 comprende una unidad de procesamiento 12, una memoria volátil 14, una memoria no volátil 16 y un módulo de transceptor 18. La memoria volátil 14, la memoria no volátil 16 y el módulo de transceptor 18, como de manera análoga a los aparatos de sensor 10, están conectados la unidad de procesamiento 12.

De manera operativa, la unidad de control 2 puede conectarse de manera selectiva a cada unidad de detección 6, estableciendo un respectivo canal de comunicación en el medio de comunicación 4. Por ejemplo, el medio de comunicación 4 puede simplemente ser el espacio libre presente entre la unidad de control 2 y las unidades de detección 6, o si no puede formarse por conexiones de material conductor. En el último caso, el medio de comunicación 4 es de un tipo alámbrico. Además, en un instante de tiempo dado, la unidad de control 2 puede comunicar con solamente una unidad de detección 6, que, de una manera en sí misma conocida, puede comunicar con la unidad de control 2 a través de su propio módulo de transceptor 18.

De manera operativa, cada aparto de sensor 10 monitoriza un parámetro correspondiente y suministra a la unidad de procesamiento 12 a la que está conectado una respectiva señal, ya sea analógica o digital, que indica el parámetro correspondiente monitorizado. En mayor detalle, considerando un aparato de sensor genérico 10 de entre los aparatos de sensor 10 de una unidad de detección genérica 6, coopera con la unidad de procesamiento 12 de la unidad de detección 6 a la que pertenece de tal manera que dicha unidad de procesamiento 12 obtendrá de una manera síncrona y/o asíncrona valores numéricos que corresponden al parámetro monitorizado por el aparato sensor considerado 10. De una manera en sí misma conocida, la adquisición de un valor numérico X puede realizarse, por ejemplo:

i) de una manera síncrona, consultando paródicamente con un periodo de muestreo T (descrito en lo sucesivo) del aparato sensor considerado 10, en la parte de la unidad de procesamiento 12 conectada al aparato sensor considerado 10, y comunicación consecuente del valor numérico X en la parte del aparato sensor considerado 10; y/o

ii) de una manera asíncrona, mediante comunicación del valor numérico X en la parte del aparato sensor considerado 10, siguiendo después una detección de una variación del parámetro correspondiente en la misma parte del aparato sensor considerado 10.

En la práctica, en el caso donde el valor numérico X se obtiene de acuerdo con ii), la adquisición se realiza de una manera no correlacionada a cualquier consulta periódica del aparato sensor considerado 10 por la unidad de procesamiento 12 conectada al mismo.

En lo que sigue, se supondrá que las adquisiciones asíncronas tienen una discretización temporal que no sea superior a Δ_t, es decir, que la distancia mínima en tiempo entre dos adquisiciones asíncronas sea igual a Δ_t.

En mayor detalle, el aparato sensor considerado 10 coopera con la unidad de procesamiento 12 a la que está

conectado de tal manera que la unidad de procesamiento 12 obtiene el valor numérico X en un formato de codificación que es apropiado para el aparato sensor considerado 10 (por ejemplo, formato de coma flotante en un cierto número de bits), o en cualquier caso a un acoplamiento electrónico (por ejemplo, por medio del canal analógico y un convertidor de analógico a digital) presente entre la unidad de procesamiento 12 y el aparato sensor considerado 10, independientemente de si la adquisición se realiza de la manera síncrona i) o si no de la manera asíncrona ii). En lo que sigue dicho formato de codificación se denominará como “formato original”.

Una vez obtenido, el valor numérico X está asociado por la unidad de procesamiento 12 anteriormente mencionada a un instante de adquisición correspondiente, así como, como se describe en el presente documento más adelante, a un correspondiente modo de adquisición, de manera alterna síncrona o asíncrona, de acuerdo con cómo se ha obtenido el valor numérico X.

Esto se dice, por razones de simplicidad y sin implicar esto ninguna pérdida de generalidad, en lo que sigue el presente método se describe con referencia a una situación simplificada en la que el sistema de detección 1 comprende una única unidad de detección, que se denominará como la “unidad de detección 6”. Además, se supone que la unidad de detección 6 comprende solamente uno de los aparatos de sensor, que se denominará como el “aparato de sensor 10”. Con dicha hipótesis de simplificación, los valores numéricos hacen referencia a un único parámetro. La descripción subsiguiente puede en cualquier caso generalizarse a un sistema de detección que comprende cualquier número de unidades de detección, cada una de las cuales a su vez comprende cualquier número de aparatos de sensor.

En lo que sigue se supone además que el parámetro, además de ser expresable en un formato numérico, tiene una dinámica comprendida entre un respectivo valor mínimo Vmin y un respectivo valor máximo Vmax, y tiene una etapa de discretización Etapa. El valor mínimo Vmin, el valor máximo Vmax, y la etapa de discretización Etapa son características del aparato sensor 10 y son conocidas tanto para la unidad de procesamiento 12 como para la unidad de control 2. En particular, la etapa de discretización Etapa del parámetro depende de la precisión del aparato sensor 10 y coincide con la variación mínima del parámetro que puede detectarse por el aparato sensor 10, y por lo tanto con la mínima diferencia distinta de cero posible entre dos valores numéricos obtenidos por medio del aparato sensor 10.

De manera operativa, la unidad de procesamiento 12 puede ejecutar además, en cada valor numérico obtenido, un número N (con $N \geq 2$) de algoritmos de codificación, de un tipo en sí mismo conocido, que suministra N correspondientes números enteros codificados, cada uno de los cuales se forma por un múltiplo entero de bytes.

Por ejemplo, en el caso donde N=2, la unidad de procesamiento 12 puede ejecutar un primer algoritmo de codificación y un segundo algoritmo de codificación, que suministran, empezando desde uno y el mismo valor numérico obtenido, un primer número entero codificado y un segundo número entero codificado, que se forman por un primer número de bytes y un segundo número de bytes, respectivamente. Como se aclara en lo sucesivo por medio de ejemplos, el primer y segundo números de bytes pueden depender no únicamente del primer y segundo algoritmos de codificación, sino también de la naturaleza del parámetro, puede entenderse por “naturaleza del parámetro” las dinámicas del parámetro y/o la etapa de discretización Etapa del mismo parámetro.

Como se muestra en mayor detalle en la Figura 2, si designamos por Vpar un valor numérico genérico del parámetro, los algoritmos de codificación prevén convertir (bloque 200) el valor numérico Vpar en un número entero NUM-INT, ya sea positivo o negativo, y a continuación codificar (bloque 210) el número entero NUM-INT en un respectivo número local de bytes para obtener un correspondiente número entero codificado NUM-BIT.

Al codificar el número entero NUM-INT, es posible que uno o más algoritmos de codificación generen una denominada excepción. En ausencia de excepciones (descritas en lo sucesivo), todos los algoritmos de codificación prevén la codificación del número entero NUM-INT en uno y el mismo número teórico de bytes de tal manera que el número local de bytes coincide con el número teórico de bytes. Como alternativa, en la presencia de excepciones, es posible que el número local de bytes sea diferente del número teórico de bytes, como se describe en el presente documento más adelante por medio de ejemplos.

Además, es posible distinguir entre los denominados algoritmos de codificación de clase a) y algoritmos de codificación de clase b), basándose en si los resultados de las respectivas etapas de conversión del valor numérico Vpar en un número entero NUM-INT, y por lo tanto los mismos números enteros NUM-INT suministrados de esta manera, a) pueden únicamente ser positivos, o si no b) pueden ser tanto positivos como negativos.

En mayor detalle, en el caso de los algoritmos de codificación de clase a), el número entero NUM-INT se codifica de acuerdo con una codificación A, que es una codificación binaria convencional de números enteros positivos (mediante la cual, por ejemplo, pueden codificarse en ocho bits doscientos cincuenta y seis números enteros positivos diferentes), modificados de tal manera que se prohíbe usar, para los fines de la codificación, las cadenas de bits de los bits del cuarteto más significativo del cual son iguales a “1”.

En su lugar, en el caso de los algoritmos de codificación de clase b), el número entero NUM-INT se codifica de

acuerdo con una codificación B, que se implementa de acuerdo con las siguientes reglas:
- el bit más significativo del byte más significativo del número entero codificado NUM-BIT se establece a “0” si el número entero NUM-INT es positivo o cero, y a “1” si el número entero NUM-INT es negativo;
- los tres bits menos significativos del cuarteto más significativo del número entero codificado NUM-BIT no pueden ser todos iguales a “1”, por razones que se aclararán en el presente documento más adelante.

En la práctica, si suponemos que el número teórico de bytes es igual a uno, los algoritmos de codificación de clase a) puede codificar, es decir, discriminar, hasta doscientos cuarenta números enteros diferentes, que corresponden a doscientos cuarenta valores numéricos diferentes, mientras que los algoritmos de codificación de clase b) pueden codificar hasta ciento doce números enteros positivos diferentes y tantos números enteros negativos, para un total de doscientos veintitrés valores numéricos diferentes. Además, evitando la generación de números enteros codificados con el cuarteto más significativo igual a “1111”, quedan disponibles dieciséis diferentes valores de escape, que se codifican en un byte y que puede expresarse en notación hexadecimal como “0xFw” con $0 \leq w \leq F$. Los valores de escape son conocidos para la unidad de control 2.

A continuación se describen algunos ejemplos de posibles algoritmos de codificación, proporcionados simplemente por medio de ilustración no limitante. Además, en lo que sigue, por razones de brevedad, los números enteros codificados NUM-BIT suministrados por los algoritmos de codificación se denominarán como “códigos”.

En detalle, un primer algoritmo de codificación, que en lo que sigue se denominará como “algoritmo de valor mínimo”, prevé, dado el valor numérico Vpar, establecer el correspondiente número entero NUM-INT igual a (Vpar-Vmin)/Etapa, es decir, igual al número de etapas de discretización a añadirse al valor mínimo Vmin para obtener el valor numérico Vpar. Dicho número entero NUM-INT puede ser positivo o cero; por lo tanto el código correspondiente NUM-BIT se obtiene codificando el número entero NUM-INT en el número teórico de bytes, de acuerdo con la codificación anteriormente mencionada A.

En la recepción, la unidad de control 2 puede determinar de nuevo el valor numérico Vpar. Para este fin, la unidad de control 2 decodifica el código NUM-BIT para obtener el número entero NUM-INT, y a continuación calcula Vpar=Vmin+Num*Etapa. El algoritmo de valor mínimo no prevé la posibilidad de generar excepciones.

Un segundo algoritmo de codificación, que en lo que sigue se denominará como “algoritmo diferencial”, prevé determinar el número entero NUM-INT que corresponde al valor numérico Vpar como una función del valor numérico obtenido inmediatamente antes del valor numérico Vpar, designado en lo que sigue como Vpar_prec. En detalle, el algoritmo diferencial prevé, dado el valor numérico Vpar, establecer el correspondiente número entero NUM-INT igual a (Vpar-Vpar_prec)/Etapa. Dicho número entero NUM-INT puede ser negativo o positivo, y en consecuencia el código NUM-BIT que corresponde al valor numérico Vpar se obtiene codificando el número entero NUM-INT en el número teórico de bytes (menos para excepciones), de acuerdo con la codificación anteriormente mencionada B.

El algoritmo diferencial prevé la posibilidad de generar excepciones. En particular, en el caso donde el valor numérico Vpar_prec no existe, por ejemplo debido a que el valor numérico Vpar es el primer valor numérico que se obtiene por la unidad de procesamiento 12, se genera una excepción. En este caso, el valor numérico Vpar se codifica usando un algoritmo alternativo, conocido de antemano para la unidad de procesamiento 12 y para la unidad de control 2, tal como, por ejemplo, el algoritmo de valor mínimo anteriormente descrito. En particular, en el caso de una excepción, el código NUM-BIT se establece igual a una secuencia binaria que puede expresarse en notación hexadecimal como “0xFBwwxxFF”, no necesariamente formada por el número teórico de bytes. En detalle, “FB” es uno de los valores de escape anteriormente mencionados; “ww” identifica el algoritmo alternativo; “xx” es el código NUM-BIT como puede obtenerse aplicando el algoritmo alternativo al número entero NUM-INT que corresponde al valor numérico Vpar (en el caso en cuestión, se supone que el número teórico de bytes es igual a uno); y “FF” es un valor de escape adicional.

En recepción, la unidad de control 2 puede determinar de nuevo el valor numérico Vpar. Para este fin, la unidad de control 2 decodifica el código NUM-BIT para obtener el número entero NUM-INT, y a continuación calcula Vpar=Vpar_prec+Num*Etapa.

Un tercer algoritmo de codificación, que en lo que sigue se denominará como “algoritmo de valor nominal”, prevé determinar el número entero NUM-INT que corresponde al valor numérico Vpar como una función de un valor de referencia Vrif, que puede estar comprendido entre el valor mínimo Vmin y el valor máximo Vmax, y puede ser igual al valor numérico que es estadísticamente más probable para el parámetro. Además, el valor de referencia Vrif es conocido para la unidad de control 2.

En detalle, el algoritmo de valor nominal prevé, dado el valor numérico Vpar, establecer el correspondiente número entero NUM-INT igual a (Vpar-Vrif)/Etapa, es decir, igual a el número de etapas de discretización a añadirse o restarse del valor de referencia Vrif para obtener el valor numérico Vpar. Dicho número entero NUM-INT puede ser positivo o negativo; por lo tanto, el correspondiente código NUM-BIT se obtiene codificando el número entero NUM-BIT en el número teórico de bytes, de acuerdo con la codificación anteriormente mencionada B. El algoritmo de valor nominal no prevé la posibilidad de generar excepciones.

En recepción, la unidad de control 2 puede determinar una vez más el valor numérico Vpar. Para este fin, la unidad de control 2 decodifica el código NUM-BIT para obtener el número entero NUM-INT, y a continuación calcula Vpar=Vrif+Num*Etapa.

Después de haber dicho esto, para posibilitar la transmisión de los valores numéricos obtenidos por la unidad de procesamiento 12 a la unidad de control 2, se llevan a cabo las operaciones mostradas en la Figura 3.

En detalle, inicialmente la unidad de procesamiento 12 establece (bloque 300) el número de bytes teórico anteriormente mencionado. En particular, el número teórico de bytes puede determinarse como una función de un número D de posibles valores numéricos que puede asumir el parámetro, siendo dicho número D igual a (Vmax-Vmin)/Etapa. En la práctica, el número teórico de bytes es al menos igual al número mínimo de bytes suficiente para codificar el número D basándose en la codificación anteriormente mencionada A.

A continuación, la unidad de control 2 envía (bloque 310) a la unidad de detección 6, por ejemplo de una manera periódica, un mensaje de solicitud de recogida de datos para el parámetro monitorizado por el aparato sensor 10.

En general, es decir, con referencia al caso donde hay un número de unidades de detección, un número de aparatos de sensor, y más de un parámetro, el mensaje de solicitud para recogida de datos contiene:
- un identificador de unidad de detección (opcional, en el caso de una única unidad de detección), que identifica una unidad de detección correspondiente;
- un identificador de aparato (opcional, en el caso de un único aparato de sensor), que identifica un correspondiente aparato de sensor;
- un identificador de parámetro (opcional, en el caso de un único parámetro), que identifica un parámetro correspondiente;
- un identificador de sesión, que identifica el mismo mensaje de solicitud de recogida de datos, o en su lugar una correspondiente sesión de recogida de datos, iniciada por dicho mensaje de solicitud de recogida de datos;
- un periodo de observación, definido por una fecha/hora de inicio, que se denominará en lo sucesivo como “INICIO”, y una fecha/hora de fin; y
- un periodo de muestreo T.

Una vez más con referencia a la situación simplificada, tras la recepción del mensaje de solicitud de recogida de datos, la unidad de procesamiento 12 crea (bloque 320) en la memoria no volátil 16 un correspondiente fichero de recuperación, por ejemplo del denominado tipo de “Lenguaje de Marcas Extensible” (XML), en el que almacena el mensaje de solicitud de recogida de datos.

A continuación, la unidad de procesamiento 12 determina un número convencional de bytes de tiempo (bloque 330), que es al menos igual al número de bytes necesarios para codificar (por ejemplo, de acuerdo con la codificación anteriormente mencionada A) un número entero positivo igual a la relación entre el periodo de muestreo T y la discretización temporal Δ_t. Por ejemplo, en el caso donde T=1 minutos y Δ_t=1 segundos, el número entero positivo anteriormente mencionado es igual a sesenta; por lo tanto, el número convencional de bytes de tiempo es igual a uno.

Además, la unidad de procesamiento 12 determina (bloque 340) periódicamente si es necesario iniciar una recogida de datos. En este sentido, debería observarse que las operaciones del bloque 340 se muestran, por razones de simplicidad, como sucesivas con respecto a las operaciones de los bloques 300-330, incluso aunque se lleven a cabo por la unidad de procesamiento 12 de una manera independiente de las operaciones de los bloques 300-330.

En detalle, para determinar el posible inicio de una recogida de datos, la unidad de procesamiento 12 comprueba periódicamente si se almacena algún mensaje de solicitud de recogida de datos en la memoria no volátil 16. En el caso donde se almacene al menos un mensaje de solicitud de recogida de datos, la unidad de procesamiento 12, que se proporciona con un respectivo reloj de sistema, compara su propio reloj de sistema con las fechas/horas de inicio contenidas en los mensajes de solicitud de recogida de datos almacenados, iniciando una recogida de datos correspondiente para cada uno de los mensajes de solicitud de recogida de datos almacenados, la respectiva fecha/hora de inicio la cual es igual a la suma de la fecha/hora de inicio de un tiempo de guarda arbitrario. En la práctica, la unidad de procesamiento 12 puede gestionar múltiples recogidas de datos también de una manera concurrente, es decir, de una manera total o parcialmente solapante en el tiempo. Como se describe en el presente documento más adelante, en general cada recogida de datos se refiere a un parámetro correspondiente y conlleva la gestión, en la parte de la unidad de procesamiento 12, de respectivas estructuras de datos (tablas). Además, en el caso de recogidas de datos concurrentes, las estructuras de datos de cada única recogida de datos son independientes de las estructuras de datos de las otras recogidas de datos. La descripción subsiguiente se refiere, por simplicidad, al caso de una única recogida de datos.

En el caso donde la unidad de procesamiento 12 determina que es necesario iniciar una recogida de datos, crea (bloque 350) en la memoria volátil 14 una tabla primaria 20 (Figura 4), que está inicialmente vacía. La tabla primaria 20 se forma por un número N+4 de columnas, como se describe en el presente documento más adelante, y permanece almacenada en la memoria volátil 14 durante la duración completa de la recogida de datos.

A continuación, después de esperar hasta el INICIO, la unidad de procesamiento 12 obtiene (bloque 360) valores numéricos y actualiza (bloque 370) la tabla primaria 20 en cada adquisición de un nuevo valor numérico. En particular, la unidad de procesamiento 12 obtiene valores numéricos tanto de una manera síncrona con un periodo igual al periodo de muestreo T y (posiblemente) de una manera asíncrona.

Como se muestra en detalle en la Figura 5, para actualizar la tabla primaria 20 después de la adquisición, en un instante t0, de un valor numérico Y, la unidad de procesamiento 12 determina (bloque 400) si se ha obtenido el valor numérico Y de una manera síncrona o asíncrona, y a continuación aplica (bloque 410) los N algoritmos de codificación al valor numérico Y para obtener N correspondientes códigos NUM-BIT.

A continuación, la unidad de procesamiento 12 inserta (bloque 420), en la tabla primaria 20, una nueva fila, que corresponde al valor numérico Y. En el caso donde se ha obtenido el valor numérico Y de una manera síncrona, los N+4 elementos de la fila insertada contienen, respectivamente:
- un identificador de intervalo de tiempo, que es de un tipo progresivo y que se denominará como “identificador de intervalo”;
- un valor de tiempo igual a t0;
- una indicación de tipo, indicando de manera alterna una adquisición síncrona, en el caso donde el valor numérico Y se haya obtenido de una manera síncrona, o si no una adquisición asíncrona, en el caso donde el valor numérico Y se haya obtenido de una manera asíncrona;
- un valor de parámetro, igual al valor numérico Y y codificado de acuerdo con el formato original; y
- los N códigos NUM-BIT que corresponden al valor numérico Y, obtenidos aplicando los N algoritmos de codificación.

En su lugar, en el caso donde el valor numérico Y se haya obtenido de una manera asíncrona, los N+4 elementos de la fila insertada son iguales a aquellos descritos para el caso de adquisición síncrona, pero para el identificador de intervalo, que está ausente.

Como se muestra a modo de ejemplo en la Figura 4, que se refiere al caso N=2 y la hipótesis de inicio de las adquisiciones síncronas en un instante de INICIO igual a 15:20:00, la tabla primaria 20 se forma por lo tanto por:

- una columna de elementos (en el ejemplo mostrado en la Figura 4, la primera columna) vacía de manera alterna, en el caso de filas que corresponden a adquisiciones asíncronas, o si no, en el caso de filas que corresponden a adquisiciones síncronas, que contienen identificadores de intervalo, proporcionándose la última por ejemplo por un contador implementado por la unidad de procesamiento 12 e incrementado en una unidad en cada adquisición síncrona;
- una columna de elementos (en el ejemplo mostrado en la Figura 4, la segunda columna) que contiene valores de tiempo, por ejemplo en la forma correspondiente horas:minutos:segundos;
- una columna de elementos (en el ejemplo mostrado en la Figura 4, la tercera columna) que contiene indicaciones de tipo, por ejemplo formadas de manera altera por el carácter “C”, en el caso de adquisición síncrona, y por el carácter “R”, en el caso de adquisición asíncrona;
- una columna de elementos (en el ejemplo mostrado en la Figura 4, la cuarta columna) que contiene los valores de parámetro, codificados en el formato original; y
- N columnas (en el ejemplo mostrado en la Figura 4, la quinta y la sexta columnas), conteniendo cada una los códigos NUM-BIT que se obtienen aplicando correspondientes algoritmos de codificación.

Debería observarse que, en la Figura 4, los valores de parámetro de la cuarta columna se muestran, para facilitar el entendimiento, en formato ASCII, en lugar de en el formato original. Además, en la Figura 4, como de manera análoga en las figuras posteriores, los códigos NUM-BIT suministrados por los N algoritmos de codificación se muestran en notación hexadecimal. Apareciendo en paréntesis se encuentran los correspondientes números enteros NUM-INT, según se generan por los correspondientes algoritmos de codificación. El número teórico de bytes es igual a uno.

En la práctica, debería observarse que, si designamos por s1 y s2 dos identificadores de intervalo contenidos en dos filas diferentes de la tabla primaria 20 y de manera que s2-s1 = 1, la diferencia entre los dos valores de tiempo contenidos en estas dos filas diferentes es igual al periodo de muestreo T. Además, entre las dos filas de la tabla primaria 20 que corresponden a s2 y a s1 puede haber un número (posiblemente cero) de filas que corresponden a adquisiciones asíncronas, que se han realizado entre los instantes de INICIO+(s1-1)*T e INICIO+(s2-1)*T.

Un valor de escape particular, por ejemplo formado por la secuencia binaria que puede expresarse en notación hexadecimal como “0xFC”, puede además almacenarse en las columnas de la tabla primaria 20 que corresponden a los N algoritmos de codificación en el caso donde, por cualquier razón, la unidad de procesamiento 12 no gestiona, en un instante INICIO+s*T (con s = 1, 2, ...) , para obtener de una manera síncrona un correspondiente valor numérico.

Además de actualizar (bloque 370, Figura 3) la tabla primaria 20, la unidad de procesamiento 12 actualiza (bloque 380) el fichero de recuperación, insertando en el mismo, para cada valor numérico, el correspondiente identificador

de intervalo (si está presente), el correspondiente valor de tiempo, la correspondiente indicación de tipo y el correspondiente valor de parámetro. De esta manera, en el caso donde tenga lugar una interrupción del proceso de adquisición, por ejemplo debido a un fallo de alimentación en la unidad de detección 6, la unidad de procesamiento 12 puede, en el momento cuando se restaura el suministro, recrear, basándose en los contenidos del fichero de recuperación, la tabla primaria 20, como están disponibles antes de la interrupción del proceso de adquisición. La unidad de procesamiento 12 puede a continuación empezar de nuevo a obtener valores numéricos cooperando con el aparato sensor 10, así como reanudando la transmisión de los valores numéricos a la unidad de control 2.

En mayor detalle, en el caso donde el proceso de adquisición de los valores numéricos en la parte de la unidad de procesamiento 12 se interrumpe durante un cierto intervalo de tiempo de interrupción, tras la reanudación del proceso de adquisición, la unidad de procesamiento 12 recrea la tabla primaria 20 y añade filas correspondientes, en los instantes INICIO+pT (con p=1, 2, ...) que caen dentro del intervalo de tiempo de interrupción. Conteniendo dichas filas correspondientes, además del respectivo identificador de intervalo (igual a p), indicaciones de tipo que identifican la adquisición síncrona y, en las columnas que corresponden a los N algoritmos de codificación, la secuencia binaria que puede expresarse en notación hexadecimal como “0xFC”.

El fichero de recuperación puede transmitirse además por la unidad de procesamiento 12 a la unidad de control 2, tras la solicitud por la misma unidad de control 2. La transmisión puede tener lugar, por ejemplo, estableciendo una denominada sesión de protocolo de transferencia de ficheros (FTP). La unidad de control 2 puede a continuación usar el fichero de recuperación para verificar la corrección de valores numéricos comunicados previamente por la unidad de detección 6.

Independientemente de la gestión del fichero de recuperación, la unidad de procesamiento 12 transmite (bloque 390, Figura 3) a la unidad de control 2 los valores numéricos obtenidos, almacenados en la tabla primaria 20. Para este fin, la unidad de control 2 y la unidad de procesamiento 12 llevan a cabo las operaciones mostradas en la Figura 6.

En particular, para rápida transmisión, en la parte de la unidad de detección 6, de los valores numéricos obtenidos, la unidad de control 2 envía (bloque 500), por ejemplo periódicamente con un periodo de control T_{ctr}, un mensaje de consulta a la misma unidad de detección 6.

El mensaje de consulta contiene:
- el mismo identificador de sesión contenido en el mensaje de solicitud de recogida de datos, para identificar la sesión de recogida de datos iniciada por el mismo mensaje de solicitud de recogida de datos; y
- un último identificador de intervalo de tiempo s_inf, usado como se describe en el presente documento más adelante.

Por ejemplo, el primer mensaje de consulta enviado por la unidad de control 2 contiene un identificador de último intervalo de tiempo s_inf igual a cero.

La unidad de procesamiento 12 recibe (bloque 510) el mensaje de consulta, y a continuación selecciona (bloque 520) un conjunto de filas de la tabla primaria 20. En particular, la unidad de procesamiento 12 selecciona las filas de la tabla primaria 20 comprendidas entre la fila que tiene el identificador de intervalo igual a s_inf+1 (incluido) y la última fila que tiene indicación de tipo el carácter “C” (excluido). En la práctica, si usamos el término “información de intervalo k” para designar la información contenida en las filas de la tabla primaria 20 comprendidas entre la fila que tiene el identificador de intervalo k (incluido) y la fila que tiene el identificador de intervalo k+1 (excluido), es decir, toda la información comprendida entre la adquisición síncrona de orden k (incluida) y la adquisición síncrona de orden k+1 (excluida), la selección implica toda la información de intervalo posterior al intervalo más bajo s_inf excepto la máxima información de intervalo. En la práctica, dicha información de intervalo máxima es la información que corresponde a las filas de la tabla primaria 20 que tienen valores de tiempo iguales o posteriores al valor de tiempo incluido en la fila que contiene un identificador de intervalo s_max, que es el identificador de intervalo máximo de entre todos los identificadores de intervalo contenidos en la tabla primaria 20. De hecho, el proceso de adquisición de dicha información de intervalo máxima puede no terminarse aún.

A continuación, la unidad de procesamiento 12 genera (bloque 530) una tabla secundaria 22, basándose en el conjunto de filas de la tabla primaria 20 previamente seleccionadas. La Figura 7 muestra un ejemplo de la tabla secundaria 22, que corresponde al ejemplo de la tabla primaria 20 mostrada en la Figura 4 y al caso donde el número convencional de bytes de tiempo es igual a dos.

La tabla secundaria 22 se forma por N+2 columnas. En detalle, la tabla secundaria 22 contiene una columna de elementos (en el ejemplo mostrado en la Figura 7, la primera columna) vacíos de manera alterna o si no conteniendo identificadores de intervalo, una columna de desplazamientos de tiempo (en el ejemplo mostrado en la Figura 7, la segunda columna, descrita en lo sucesivo) y N columnas (en el ejemplo mostrado en la Figura 7, la tercera y cuarta columnas), que cada una corresponde a un respectivo algoritmo de codificación. En la práctica, la columna de desplazamientos de tiempo define una secuencia de desplazamientos de tiempo, mientras que las N columnas que corresponden a los algoritmos de codificación cada una define una respectiva secuencia de código.

Para generar la tabla secundaria 22, la unidad de procesamiento 12 lleva a cabo las operaciones mostradas en la Figura 8.

En detalle, para cada fila seleccionada durante las operaciones del bloque 520, la unidad de procesamiento 12 verifica (bloque 600) si la fila seleccionada corresponde a una adquisición de un tipo síncrono o de un tipo asíncrono, basándose en la correspondiente indicación de tipo.

Si la fila seleccionada corresponde a una adquisición síncrona (salida SÍ desde el bloque 600), la unidad de procesamiento 12 inserta (bloque 610), en la tabla secundaria 22, una nueva fila, que contiene un identificador de intervalo igual al identificador de intervalo de la fila seleccionada, un desplazamiento de tiempo cero codificado en el número convencional de bytes de tiempo, y N valores de escape, que corresponden por ejemplo a la secuencia binaria que puede expresarse en notación hexadecimal como “0xF6”.

A continuación, la unidad de procesamiento 12 verifica (bloque 620) si existe en la tabla primaria 20 una fila que precede la fila seleccionada, y si los elementos de dicha fila precedente que corresponden a los N algoritmos de codificación son iguales a los correspondientes elementos de la fila seleccionada. Si es así (salida SÍ desde el bloque 620), es decir, en el caso donde existe dicha fila precedente y esta es igual, con respecto a los elementos que corresponden a los algoritmos de codificación, a la fila seleccionada, la unidad de procesamiento itera las operaciones del bloque 600 en una nueva fila seleccionada de la tabla primaria 20. De lo contrario (salida NO desde el bloque 620), es decir, en el caso donde dicha fila precedente no existe o no es igual a la fila seleccionada, la unidad de procesamiento inserta (bloque 630), en la tabla secundaria 22, una fila adicional sin identificador de intervalo, que contiene un desplazamiento de tiempo cero codificado en el número convencional de bytes de tiempo, y en el que cada elemento que corresponde a un algoritmo de codificación dado contiene el código NUM-BIT que se almacena en el elemento de la fila seleccionada que corresponde al algoritmo de codificación dado. A continuación, la unidad de procesamiento 12 itera las operaciones del bloque 600 en una nueva fila seleccionada de la tabla primaria 20.

De otra manera, si la fila seleccionada corresponde a una adquisición asíncrona (salida NO desde el bloque 600), la unidad de procesamiento 12 determina (bloque 640) una diferencia de tiempo DIFF. En particular, si designamos por TAS1 el valor de tiempo presente en la fila seleccionada y con TS1 el valor de tiempo presente de una fila correspondiente de inicio de intervalo, es decir, en la fila de la tabla primaria 20 que corresponde a una adquisición síncrona y el valor de tiempo TS1 el cual es de manera que TS1<TAS1 y TAS1-TS1<T, tenemos DIFF=TAS1-TS.

A continuación, la unidad de procesamiento 12 codifica (bloque 650) la diferencia de tiempo DIFF en el número convencional de bytes de tiempo para obtener una desplazamiento de tiempo correspondiente. En la práctica, el número convencional de bytes de tiempo determinados por medio de las operaciones del bloque 330 es de manera que es posible de manera efectiva codificar, sin pérdida de información, alguna diferencia entre los valores de tiempo presentes en la fila seleccionada y la correspondiente fila de intervalo.

A continuación, la unidad de procesamiento 12 inserta (bloque 660), en la tabla secundaria 22, una nueva fila sin identificador de intervalo, que contiene el desplazamiento de tiempo obtenido previamente y en el que cada elemento que corresponde a un algoritmo de codificación dado contiene el código NUM-BIT que se almacena en el elemento de la fila seleccionada que corresponde al algoritmo de codificación dado.

A continuación, la unidad de procesamiento itera las operaciones del bloque 600 en una nueva fila seleccionada de la tabla primaria 20.

Debería observarse que es posible considerar el desplazamiento de tiempo cero insertado durante las operaciones del bloque 610 según se obtiene codificando una diferencia de tiempo cero DIFF. Debería observarse además que por conveniencia de lectura del ejemplo de la tabla secundaria 22 mostrada en la Figura 7, lo que aparece en paréntesis en las columnas que corresponden a los N algoritmos de codificación son los correspondientes números enteros NUM-INT, excepto en los casos donde los códigos NUM-BIT son iguales a “0xF6”. Además, lo que aparece en la columna que contiene los desplazamientos de tiempo no son solo los mismos desplazamientos de tiempo, mostrados en formato hexadecimal; de hecho, en el caso de desplazamientos de tiempo distintos de cero, también aparece en paréntesis las diferencias de tiempo correspondientes DIFF.

Una vez que se pasa por las operaciones del bloque 530, la unidad de procesamiento 12 determina (bloque 540, Figura 6), para cada columna de las N columnas de la tabla secundaria 22 que corresponden a los N algoritmos de codificación y la columna que contiene los desplazamientos de tiempo, un correspondiente número efectivo de bytes L_{ott}.

En mayor detalle, si las columnas de la tabla secundaria 22 que corresponden a los N algoritmos de codificación y a la columna de desplazamientos de tiempo se denominan como una totalidad como “columnas a optimizar”, y considerando una columna de las N+1 columnas a optimizar, la unidad de procesamiento 12 lleva a cabo las operaciones mostradas en la Figura 9, que hacen referencia, a modo de ejemplo, al caso donde la columna considerada es una de las N columnas que corresponden a los N algoritmos de codificación. La descripción

subsiguiente puede en cualquier caso generalizarse al caso donde la columna considerada es la columna de los desplazamientos de tiempo.

En detalle, la unidad de procesamiento 12 inicializa (bloque 700) a cero una pluralidad de contadores normales, indexados con un índice j, y un contador de excepción. Además, la unidad de procesamiento 12 selecciona de manera individual (bloque 710) cada código NUM-BIT presente en la columna considerada, y a continuación determina (bloque 720) un correspondiente número mínimo de bytes Nmin, siendo dicho número mínimo de bytes Nmin el número mínimo de bytes suficiente para codificar el número entero NUM-INT que corresponde al código NUM-BIT seleccionado.

En particular, en el caso donde el número entero NUM-INT que corresponde al código NUM-BIT seleccionado ha generado una excepción en la parte del algoritmo de codificación que corresponde a la columna considerada, el número mínimo de bytes Nmin es igual al número de bytes que forma dicho código NUM-BIT seleccionado; es decir, es igual al número de bytes local correspondiente. De otra manera, en el caso donde dicho número entero NUM-INT no ha generado una excepción, el número mínimo de bytes Nmin es el número mínimo de bytes suficiente para codificar dicho número entero NUM-INT, de manera alterna de acuerdo con la codificación A o, si no, la codificación B, dependiendo de si el algoritmo de codificación que corresponde a la columna considerada es, respectivamente, de clase a) o si no b). En el caso donde el código NUM-BIT seleccionado es igual a “0xF6”, el número mínimo de bytes es igual a uno.

En la práctica, en el caso donde el código NUM-BIT seleccionado no corresponde a una excepción, es posible que el número mínimo correspondiente de bytes Nmin sea menor que el respectivo número local de bytes, y por lo tanto también que el número teórico de bytes.

A continuación, la unidad de procesamiento 12 verifica (bloque 725) si el código NUM-BIT seleccionado es una excepción, caso en el que incrementa (bloque 726) el contador de excepción en un número igual al número de bytes de la excepción. De otra manera, si el código NUM-BIT seleccionado no es una excepción, la unidad de procesamiento incrementa (bloque 730) en una unidad el contador normal indexado por el índice j igual al número mínimo de bytes Nmin determinado (j=Nmin).

Después de seleccionar todos los códigos NUM-BIT presentes en la columna considerada, la unidad de procesamiento 12 tiene disponible una pluralidad de contadores normales y el contador de excepción. En un contador normal de orden j genérico, el número contenido en el miso es igual a M(j) (posiblemente, M(j) puede ser igual a cero), y es igual al número de apariciones de códigos NUM-BIT que no corresponden a excepciones y los números mínimos de bytes Nmin de los cuales son iguales a j.

A continuación, si jmax es el índice máximo al que corresponde un respectivo M(j) distinto de cero, la unidad de procesamiento 12 determina (bloque 740) un número jmax de longitudes S(z), indexadas por un índice z. En detalle, la unidad de procesamiento 12 calcula, para z = 1, ..., jmax:

$$S(z) = \sum_{i=1}^{j\max} factor(i) \cdot M(i) + ECC$$

donde ECC es el contador de excepción, y donde factor(i)=z si i≤z≤i+2; de otra manera, factor(i)=i+2.

En la práctica, las operaciones del bloque 740 son equivalentes, para cada longitud S(z) que corresponde a un número hipotético HYPO=z de entre los números mínimos de bytes Nmin determinados anteriormente, para seleccionar cada código NUM-BIT de la columna considerada que no corresponde a una excepción, y verificar si:

CASO 1) HYPO<Nmin o sino HYPO>Nmin+2;
CASO 2) Nmin≤HYPO≤Nmin+2.

A continuación, en CASO 1) se añade Nmin+2, mientras que en CASO 2) se añade HYPO. Adicionalmente, para cada excepción, se añade el número de bytes del código que ha generado la misma excepción.

A continuación, la unidad de procesamiento 12 selecciona (bloque 750) la longitud mínima S_{min}(zmin) de entre las longitudes S(z), y por lo tanto selecciona también el índice zmin al que corresponde dicha longitud mínima S_{min} (zmin) . En la práctica, zmin representa el número efectivo anteriormente mencionado de bytes L_{ott}, específico de la columna considerada, y por lo tanto del algoritmo de codificación considerado.

Como se ha mencionado anteriormente, las operaciones de los bloques 700-750 se llevan a cabo también en la columna de la tabla secundaria 22 que corresponde a los desplazamientos de tiempo en orden para obtener un correspondiente número efectivo de bytes L_{ott}. Debería observarse que, en este caso, dado un desplazamiento de tiempo, el número mínimo correspondiente de bytes Nmin es el número mínimo de bytes suficiente para codificar la diferencia de tiempo correspondiente DIFF, o sino el valor cero (en caso de desplazamiento de tiempo), de acuerdo

con la codificación anteriormente mencionada A.

Al final de las operaciones de los bloques 700-750, la unidad de procesamiento 12 tiene disponibles N+1 números efectivos de bytes L_{ott}, así como N+1 longitudes mínimas $S_{min}(zmin)$, de las cuales N corresponden a los N algoritmos de codificación. Una vez más con referencia a la Figura 6, después de llevar a cabo las operaciones del bloque 540, la unidad de procesamiento 12 selecciona (bloque 550) un algoritmo óptimo. En particular, el algoritmo óptimo es el algoritmo de codificación al que corresponde la más pequeña de entre las N longitudes mínimas $S_{min}(zmin)$ que corresponden a los N algoritmos de codificación.

A continuación, la unidad de procesamiento 12 genera (bloque 560) una tabla terciaria 24, un ejemplo de la cual se muestra en la Figura 10. En particular, en el ejemplo mostrado en la Figura 10, los números efectivos de bytes L_{ott} que corresponden a los desplazamientos de tiempo y al primer algoritmo de codificación y al segundo algoritmo de codificación son iguales a uno.

En detalle, la tabla terciaria 24 se forma por N+2 columnas y es similar a la tabla secundaria 22; por lo tanto, contiene una columna de elementos que están vacíos de manera alterna o si no contienen identificadores de intervalos, una columna de desplazamientos de tiempo y N columnas que corresponden a los N algoritmos de codificación. En la práctica, la columna de desplazamientos de tiempo define una secuencia comprimida de desplazamientos de tiempo, mientras que las N columnas que corresponden a los algoritmos de codificación cada una define una respectiva secuencia de código comprimido.

Para generar la tabla terciaria 24, la unidad de procesamiento 12 lleva a cabo las operaciones mostradas en la Figura 11.

En detalle, la unidad de procesamiento 12 selecciona (bloque 800) de manera individual cada fila de la tabla secundaria 22, y verifica (bloque 810) si la fila seleccionada pertenece a un grupo de filas formado por al menos cuatro filas de la tabla secundaria 22 establecidas adyacentes y que contienen, en los respectivos elementos de las N columnas que corresponden a los N algoritmos de codificación, el valor de escape que corresponde a la secuencia binaria que puede expresarse en notación hexadecimal como “0xF6” (son, por lo tanto, filas que corresponden a adquisiciones síncronas).

En el caso donde la fila seleccionada de la tabla secundaria 22 no pertenezca a un grupo de filas (salida NO desde el bloque 810), la unidad de procesamiento 12 inserta una nueva fila en la tabla terciaria 24.

En detalle, en el caso donde la fila seleccionada de la tabla secundaria 22 contenga un respectivo identificador de intervalo, la unidad de procesamiento 12 inserta (bloque 820), en dicha nueva fila de la tabla terciaria 24, el identificador de intervalo de la fila seleccionada de la tabla secundaria 22.

Además, la unidad de procesamiento 12 selecciona (bloque 830) individualmente los desplazamientos de tiempo y los códigos NUM-BIT de la fila seleccionada de la tabla secundaria 22.

A continuación, la unidad de procesamiento 12 verifica (bloque 840) si el desplazamiento de tiempo/código NUM-BIT seleccionado es un código NUM-BIT que corresponde a una excepción, y por lo tanto si es del tipo “0xFBwwxxFF”, o sino al valor de escape “0xF6”, caso en el que (salida SÍ desde el bloque 840) la unidad de procesamiento 12 inserta (bloque 850) el código NUM-BIT seleccionado en el elemento correspondiente de la nueva fila de la tabla terciaria 24. A continuación, la unidad de procesamiento 12 verifica (bloque 860) si hay otros elementos (desplazamiento de tiempo o códigos NUM-BIT) de la fila seleccionada de la tabla secundaria 22 a seleccionarse, caso en el que (salida SÍ desde el bloque 860) itera las operaciones del bloque 830; de lo contrario (salida NO desde el bloque 860), itera las operaciones del bloque 800 en una fila posterior de la tabla secundaria 22, hasta el agotamiento de la tabla secundaria 22.

De otra manera, es decir, en el caso donde la unidad de procesamiento 12 haya seleccionado un desplazamiento de tiempo o un código NUM-BIT que no corresponde a una excepción ni al valor de escape “0xF6” (salida NO desde el bloque 840), verifica (bloque 870) si el número mínimo de bytes Nmin que corresponde al desplazamiento de tiempo/NUM-BIT de código seleccionado es de manera que $Nmin \leq L_{ott} \leq Nmin+2$, donde L_{ott} es el correspondiente número efectivo de bytes determinado anteriormente. En la práctica, dicho número efectivo de bytes correspondiente L_{ott} es el número efectivo de bytes L_{ott} que corresponde a la columna a la que pertenece el desplazamiento de tiempo/código NUM-BIT seleccionado.

En el caso donde $Nmin > L_{ott}$ o si no $Nmin < L_{ott}\_2$ (salida NO desde el bloque 870), la unidad de procesamiento 12 inserta (bloque 880), en la nueva fila de la tabla terciaria 24, en particular en el elemento de la nueva fila de la tabla terciaria 24 que corresponde al desplazamiento de tiempo/código NUM-BIT seleccionado, una secuencia binaria que puede expresarse en notación hexadecimal como “0xF9mmyyyy”. En detalle, “F9” es un valor de escape que indica el inicio de un tipo de salida desde el correspondiente número efectivo de bytes L_{ott}; “mm” indica (por ejemplo, de acuerdo con la codificación A) el número mínimo de bytes Nmin que corresponde al desplazamiento de tiempo/código NUM-BIT seleccionado; y “yyyy” es el desplazamiento de tiempo/código NUM-BIT seleccionado,

expresado en el número mínimo de bytes Nmin (En este caso, “mm”=“0x02”, debido a que la secuencia proporcionada a modo de ejemplo “yyyy” ocupa dos bytes).

Debería observarse que por la frase “desplazamiento de tiempo/código NUM-BIT seleccionado, expresado en el número mínimo de bytes Nmin” se entiende una secuencia binaria obtenida codificando la diferencia de tiempo/número entero que corresponde al desplazamiento de tiempo/código NUM-BIT seleccionado en el respectivo número mínimo de bytes Nmin, de acuerdo con la codificación A o la codificación B, dependiendo de si es un desplazamiento de tiempo o un código NUM-BIT y, en el último caso, dependiendo de la clase del algoritmo correspondiente. En la práctica, si designamos por TEO el número convencional de bytes de tiempo/número teórico de bytes, si TEO>Nmin, la secuencia binaria anteriormente mencionada “yyyy” es igual al desplazamiento de tiempo/código NUM-BIT seleccionado, menos un número TEO-Nmin de cero bytes, siendo lo último vaciado de información.

A continuación, la unidad de procesamiento 12 itera las operaciones del bloque 860.

De otra manera, en el caso donde $Nmin \leq L_{ott} \leq Nmin+2$ (salida SÍ desde el bloque 870), la unidad de procesamiento 12 inserta (bloque 890), en la nueva fila de la tabla terciaria 24 (en particular en el elemento de la nueva fila de la tabla terciaria 24 que corresponde al desplazamiento de tiempo/código NUM-BIT seleccionado), el desplazamiento de tiempo/código NUM-BIT seleccionado, expresado en el número efectivo correspondiente anteriormente mencionado de bytes L_{ott}. En otras palabras, la unidad de procesamiento 12 inserta la secuencia binaria obtenida codificando la diferencia de tiempo/número entero que corresponde al desplazamiento de tiempo/código NUM-BIT seleccionado en el correspondiente número efectivo de bytes L_{ott}, de acuerdo con la codificación A o la codificación B, dependiendo de si es un desplazamiento de tiempo o un código NUM-BIT, y, en el último caso, dependiendo de la clase del algoritmo correspondiente.

A continuación, la unidad de procesamiento 12 itera las operaciones del bloque 860.

Como alternativa, en el caso donde la fila seleccionada pertenece a un grupo de filas (salida SÍ desde el bloque 810), la unidad de procesamiento 12 inserta una nueva fila en la tabla terciaria 24.

En particular, la unidad de procesamiento 12 inserta (bloque 900), en dicha nueva fila de la tabla terciaria 24, el mismo identificador de intervalo como el presente en la fila seleccionada. Además, la unidad de procesamiento 12 inserta (bloque 910), en el elemento de la nueva fila que pertenece a la columna de los desplazamientos de tiempo, un desplazamiento de tiempo cero (expresado de acuerdo con la codificación A, en el número efectivo de bytes L_{ott} que corresponde a los desplazamientos de tiempo), así como, en cada elemento de las N columnas que corresponden a los N algoritmos de codificación, la secuencia binaria que puede expresarse en notación hexadecimal como “0xFAnnF6”, donde “FA” es un valor de escape que indica el inicio de los grupos de filas, y “nn” indica el número de filas (y por lo tanto de intervalo) presentes en el grupo de filas.

A continuación, la unidad de procesamiento 12 itera las operaciones del bloque 860 en la primera fila de la tabla secundaria 22 posterior al grupo de filas anteriormente mencionado, hasta el agotamiento de la tabla secundaria 22.

En la práctica, la presencia de un grupo de filas dentro de la tabla secundaria 22 indica que, durante los correspondientes intervalos, el parámetro no ha variado. En consecuencia, las operaciones de los bloques 800-910 posibilitan una reducción del tráfico implicado en transmitir esta información. En otras palabras, nn*T es igual a la duración del intervalo de tiempo en el que el parámetro permanece constante. El umbral que corresponde a la presencia de cuatro filas es específico para el caso particular en el que el número teórico de bytes es igual a uno y es debido al hecho de que, con las codificaciones descritas, las operaciones de los bloques 800-910 posibilitan la reducción del tráfico únicamente si cada grupo de filas se forma por al menos cuatro filas, puesto que la secuencia binaria “0xFAnnF6” conlleva una tara de tráfico de dos bytes con respecto al número teórico de bytes. Las modificaciones con respecto a las codificaciones descritas pueden realizarse en cualquier caso, con variaciones consecuentes del umbral.

Una vez más con referencia a las operaciones mostradas en la Figura 6, la unidad de procesamiento 12 crea (bloque 570), si está ausente, o de lo contrario actualiza, si está presente, una tabla de resultados 26, que se forma por N+1 columnas, N de las cuales corresponde a los N algoritmos de codificación. Un ejemplo de la tabla de resultados 26 se muestra en la Figura 12.

En detalle, la unidad de procesamiento 12 inserta una fila en la tabla de resultados 26, que contiene, como el primer elemento, el identificador de sesión contenido en el mensaje de solicitud de recogida de datos. Además, cada elemento de entre los N elementos restantes de la fila insertada corresponde a un respectivo algoritmo de codificación y contiene un número igual al número global de bytes contenidos en la columna de la tabla terciaria 24 que corresponde a dicho algoritmo respectivo.

A continuación, la unidad de procesamiento 12 envía (bloque 580) un mensaje de respuesta a la unidad de control 2.

El mensaje de respuesta contiene:
- el identificador de sesión contenido en el mensaje de solicitud de recogida de datos y en el mensaje de consulta;
- un código que identifica el algoritmo óptimo;
- un identificador de intervalo inicial, igual a s_inf+1;
- un identificador de intervalo igual a s_max-1;
- el número efectivo de bytes L_{ott} que corresponde al algoritmo óptimo;
- el número efectivo de bytes L_{ott} que corresponde a los desplazamientos de tiempo; y
- una cadena de datos optimizada.

En detalle, la cadena de datos optimizada se forma por la unidad de procesamiento 12 basándose en los bytes presentes en las columnas de la tabla terciaria 24 que corresponden a los desplazamientos de tiempo y al algoritmo óptimo.

En mayor detalle, la unidad de procesamiento 12 explora las filas de la tabla terciaria 24 en secuencia, empezando desde la primera fila hasta la última. Un ejemplo de cadena de datos optimizada que corresponde a la tabla terciaria 24 mostrada en la Figura 10 se proporciona en la Figura 13.

En el caso donde la fila explorada contiene, en la columna que corresponde al algoritmo óptimo, el valor de escape “0xF6” o si no una secuencia binaria que puede expresarse en notación hexadecimal como “0xFAnnF6”, la unidad de procesamiento añade a la cadena de datos optimizada (que está inicialmente vacía) el valor de escape “0xF6” o si no la secuencia binaria que puede expresarse en notación hexadecimal como “0xFAnnF6”, respectivamente.

En su lugar, en el caso donde la fila explorada no contiene, en la columna que corresponde al algoritmo óptimo, el valor de escape “0xF6” ni una secuencia binaria que puede expresarse en notación hexadecimal como “0xFAnnF6”, la unidad de procesamiento 12 añade a la cadena de datos optimizada en primer lugar el desplazamiento de tiempo de la fila explorada y a continuación la secuencia binaria contenida en el elemento de la fila explorada que corresponde al algoritmo óptimo.

A continuación, la unidad de procesamiento 12 explora una fila posterior de la tabla terciaria 24, hasta el agotamiento de la tabla terciaria.

Finalmente, la unidad de procesamiento 12 elimina (bloque 590) la tabla secundaria 22 y la tabla terciaria 24 de la memoria volátil 14. La tabla primaria 20 permanece, en su lugar, inalterada.

Las operaciones descritas por lo tanto posibilitan la transmisión de los valores numéricos obtenidos por la unidad de detección 6, usando un número limitado de bytes.

Después de la recepción del mensaje de respuesta, la unidad de control 2 puede reconstruir la evolución en tiempo del parámetro, aplicando hacia atrás el algoritmo óptimo para obtener, empezando desde los códigos, los correspondientes números enteros, y por lo tanto los correspondientes valores numéricos.

En particular, la unidad de control 2 puede discriminar correctamente entre desplazamientos de tiempo y códigos NUM-BIT, basándose en los valores de escape, el número efectivo de bytes L_{ott} que corresponde al algoritmo óptimo, y el número efectivo de bytes L_{ott} que corresponde a los desplazamientos de tiempo.

En mayor detalle, la cadena de datos optimizada se inicia con una secuencia binaria “0xF6”. La unidad de control 2, como se muestra en la Figura 14, detecta (bloque 1000) dicha secuencia binaria “0xF6” y calcula (bloque 1010) un instante de intervalo igual a INICIO+recuento*T, donde “recuento” es un contador que se encuentra inicialmente en el valor cero. A continuación, la unidad de control 2 decodifica y analiza (bloque 1020) el byte posterior de la cadena de datos optimizada.

En el caso donde el byte posterior, que en lo que sigue se denominará como “primer byte de tiempo”, es el valor de escape “0xF9” (salida SÍ desde el bloque 1020), significa que estamos en la presencia de un desplazamiento de tiempo codificado como “0xF9mmyyyy”, es decir, de una “salida”. Por lo tanto, la unidad de control 2 decodifica (bloque 1030) el byte posterior al primer byte de tiempo (“0xmm”), que se denominará como “segundo byte de tiempo”, y a continuación selecciona (bloque 1040) un número de bytes posterior al segundo byte de tiempo (“0xyyyy”), indicándose dicho número por el mismo segundo byte de tiempo. La unidad de control 2 decodifica (bloque 1050) dicho número de bytes posterior al segundo byte de tiempo para obtener una diferencia de tiempo, en este punto designada por Δ. A continuación, la unidad de control 2 determina (bloque 1060) un instante INICIO+recuento*T+Δ.

De otra manera, si el primer byte de tiempo es diferente de “0xF9” (salida NO desde el bloque 1020), la unidad de control 2 decodifica (bloque 1055) un grupo de bytes que incluye el primer byte de tiempo y un número de bytes (posterior al primer byte de tiempo) igual al número efectivo de bytes L_{ott} que corresponde a los desplazamientos de tiempo, decrementados en uno. De esta manera, la unidad de control 2 obtiene una correspondiente diferencia de tiempo Δ. A continuación, la unidad de control 2 lleva a cabo las operaciones del bloque 1060.

Después de realizar las operaciones del bloque 1060, la unidad de control 2 decodifica y analiza (bloque 1070) el byte posterior al número posterior de bytes “0xyyyy”, que en lo que sigue se denominará como “primer byte de datos”, y verifica si el primer byte de datos es igual de manera alterna a “0xF9”, o si no a “0xFA”, o si no a “0xFB”, o si no es diferente de “0xF9”, de “0xFA” y de “0xFB”.

Si el primer byte de datos es igual a “0xF9” (salida 0xF9 del bloque 1070), significa que estamos en presencia de un código codificado como “0xF9mmyyyy”, es decir, de una “salida”; por lo tanto, la unidad de control 2 decodifica (bloque 1080) el byte posterior al primer byte de datos, que en lo que sigue se denominará como “segundo byte de datos”, y selecciona (bloque 1090) un número de bytes posterior al segundo byte de datos, indicándose dicho número por el mismo segundo byte de datos. La unidad de control 2 decodifica (bloque 1100) dicho número de bytes posterior al segundo byte de datos, basándose en el algoritmo óptimo para obtener el correspondiente número entero y, por lo tanto, el correspondiente valor numérico.

Si el primer byte de datos es igual a “0xFB” (salida 0xFB del bloque 1070), significa que estamos en presencia de una excepción; por lo tanto, la unidad de control 2 decodifica (bloque 1110) el byte posterior al primer byte de datos, para determinar (bloque 1120) el algoritmo alternativo, y a continuación decodifica (bloque 1130) por medio del algoritmo alternativo el byte comprendido entre el segundo byte de datos (excluido) y el primer byte igual a “0xFF” (excluido).

Si el primer byte de datos es igual a “0xFA” (salida 0xFA del bloque 1070), significa que el parámetro ha permanecido constante durante un cierto intervalo de tiempo. En consecuencia, la unidad de control 2 decodifica (bloque 1150) el segundo byte de datos y determina (bloque 1160) un número de intervalo en el que el parámetro no ha cambiado. Además, la unidad de control elimina (bloque 1170) de la cadena de datos optimizada el byte “0xF6” posterior al segundo byte de datos, incrementa “recuento” en un número igual al número de intervalos en el que el parámetro no ha cambiado, y a continuación itera las operaciones del bloque 1020.

Si el primer byte de datos es diferente de “0xF9”, de “0xFA” y de “0xFB” (salida del bloque 1070), la unidad de control 2 decodifica (bloque 1180) un grupo de bytes formado por el primer byte de datos y por un número de bytes (posterior al primer byte de datos) igual al número efectivo de bytes L_{ott} que corresponde al algoritmo óptimo, decrementado en uno. De esta manera, la unidad de control 2 obtiene un correspondiente valor numérico.

Al final de las operaciones de los bloques 1100, 1130 y 1180, la unidad de control 2 asocia (bloque 1190) el valor numérico obtenido en el instante INICIO+recuento*T+Δ.

A continuación, la unidad de control selecciona y decodifica (bloque 1200) un byte adicional y verifica (bloque 1210) si dicho byte adicional es igual a “0xF6”.

Si el byte adicional es igual a “0xF6” (salida SÍ desde el bloque 1210), la unidad de control 2 incrementa (bloque 1220) en una unidad el contador “recuento”, y a continuación itera las operaciones del bloque 1010 y posteriores bloques. De otra manera, si el byte adicional es diferente de “0xF6” (salida NO desde el bloque 1210), la unidad de control 2 itera las operaciones del bloque 1020.

Las ventajas que el presente método permite surgen de manera evidente a partir de la descripción anterior. En particular, la transmisión de valores numéricos se optimiza basándose en los algoritmos de codificación disponibles y de los mismos valores numéricos, según se obtiene durante un cierto intervalo de tiempo. Se transmiten de manera efectiva reducciones adicionales en el número de bytes basándose en la exploración de tiempo con lo que se obtienen los valores numéricos, así como posibles periodos de ausencia de adquisición.

El método descrito es por lo tanto adecuado para aplicaciones en las que es de particular importancia reducir tanto como sea posible el número de bytes transmitidos, tales como por ejemplo aplicaciones de control por satélite, o si no aplicaciones en el campo de control remoto por medio de ondas transportadas, es decir, aplicaciones donde los datos se transmiten a través de líneas de suministro eléctrico. En estos casos, el sistema de detección 1 funciona como un sistema de lectura remota, en el que el medio de comunicación 4 se forma de manera alterna por espacio libre, o si no por líneas de suministro eléctrico.

Adicionalmente, incluso aunque el presente método se ha descrito con referencia particular al caso donde está presente un único parámetro, puede aplicarse al caso donde se han de monitorizar diferentes parámetros con diferentes dinámicas y etapas de discretización, y por lo tanto con diferentes números teóricos de bytes.

Finalmente, es evidente que pueden realizarse modificaciones y variaciones al presente método, sin alejarse de esta manera del alcance de la presente invención, según se define por las reivindicaciones adjuntas.

Por ejemplo, la información contenida en las tablas primaria, secundaria y terciaria puede organizarse de una manera diferente de lo que se ha descrito en el presente documento. Además, es posible almacenar en las tablas anteriormente mencionadas información diferente de la descrita; por ejemplo, es posible insertar identificadores de intervalos también en las filas de la tabla primaria 20 que corresponden a adquisiciones asíncronas.

Además, es posible aplicar algoritmos de codificación que son adicionales a y/o diferentes de los descritos.

Es además posible ejecutar una o más de las operaciones mostradas en las Figuras 2, 3, 5, 6, 8, 9, 11 y 14 en un orden diferente del descrito. Es además posible que una o más de las operaciones mostradas en las Figuras 2, 3, 5, 6, 8, 9, 11 y 14 estén ausentes. Además, es posible, durante la ejecución de los algoritmos de codificación, para cada número entero NUM-INT, como de manera análoga para cada diferencia de tiempo DIFF, que se codifique de una manera dinámica directamente en un número mínimo correspondiente Nmin de bytes. En este caso, las operaciones del bloque 890 prevén insertar, en la nueva fila de la tabla terciaria 24, exactamente el desplazamiento de tiempo/código NUM-BIT seleccionado; además, con respecto a las operaciones del bloque 880, la cadena “yyyy” coincide con el desplazamiento de tiempo/código NUM-BIT seleccionado.

Una vez más, es posible aplicar el presente método también en el caso donde las operaciones de codificación se realizan no en bytes, sino en grupos de bits de cualquier longitud.

Finalmente, el presente método puede aplicarse también a valores de tipo no numérico (por ejemplo, de un tipo alfabético, o de lo contrario valores que corresponden a imágenes o películas, es decir, píxeles), después de la transformación apropiada anterior, de una manera en sí conocida, en correspondientes valores numéricos.

REIVINDICACIONES

1. Un método para transmitir valores numéricos de una unidad de detección (6) a una unidad de control (2), estando configurada la unidad de detección para obtener valores numéricos en respectivos instantes de adquisición, de una manera síncrona y/o asíncrona, y basándose en una señal que indica una cantidad a monitorizarse; comprendiendo dicho método, para cada valor considerado de entre los valores numéricos obtenidos, realizar, por dicha unidad de detección, las etapas de:
- aplicar (410) a dicho valor considerado al menos un primer algoritmo de codificación y un segundo algoritmo de codificación, para obtener, respectivamente, un primer código y un segundo código, estando dicho primer y segundo códigos en formato binario;
- almacenar (420) el primer código, el segundo código y el instante de adquisición que corresponden al valor considerado;
comprendiendo el método adicionalmente realizar, por dicha unidad de detección, la etapa de generar (530) una primera secuencia de código y una segunda secuencia de código, asociadas respectivamente al primer algoritmo de codificación y al segundo algoritmo de codificación y que comprende, respectivamente, el primer y segundo códigos almacenados;
caracterizado por que comprende adicionalmente realizar, por dicha unidad de detección, las etapas de:
- seleccionar (550) un algoritmo óptimo de dicho al menos un primer algoritmo de codificación y un segundo algoritmo de codificación, basándose en la primera y segunda secuencias de código;
- generar (560) una secuencia comprimida de datos basándose en, de manera alterna, la primera secuencia de código o la segunda secuencia de código, dependiendo del algoritmo óptimo seleccionado; y
- transmitir (390) a la unidad de control la secuencia comprimida de datos.

2. El método de transmisión de acuerdo con la reivindicación 1, que comprende adicionalmente llevar a cabo, por la unidad de detección (6), las etapas de:
- determinar una pluralidad de instantes de muestreo periódicos;
- determinar (640) una pluralidad de diferencias de tiempo basándose en los instantes de adquisición almacenados y en instantes de muestreo correspondientes;
- determinar (650) una pluralidad de intervalos de tiempo en formato binario, basándose en las diferencias de tiempo; y
- generar (560) una secuencia comprimida de intervalos de tiempo, basándose en los intervalos de tiempo determinados;
y en el que dicha etapa de transmisión (390) a la unidad de control (2) de la secuencia comprimida de datos comprende las etapas de generar una cadena binaria basándose en la secuencia comprimida de datos y en la secuencia comprimida de intervalos de tiempo, y transmitir la cadena binaria.

3. El método de transmisión de acuerdo con la reivindicación 2, en el que la etapa de selección de un algoritmo óptimo (550) comprende, para cada secuencia considerada de entre la primera y segunda secuencias de código, las etapas de:
- para cada código de la secuencia considerada, determinar (720) un correspondiente número mínimo de bits suficiente para codificar el correspondiente valor numérico obtenido de acuerdo con el algoritmo de codificación que corresponde a la secuencia considerada;
- seleccionar individualmente cada número mínimo de bits de entre los números mínimos de bits determinados;
- para cada número mínimo de bits seleccionado individualmente, determinar (740) una longitud correspondiente de datos relacionada con la secuencia considerada, correspondiendo dicha etapa de determinación a la longitud de datos que comprende añadir, para cada código considerado de entre los códigos de la secuencia considerada:
a) el número mínimo de bits seleccionados individualmente, si dicho número mínimo de bits seleccionados individualmente está comprendido entre el número mínimo de bits relacionados con el código considerado y la suma del número mínimo de bits relacionados con el código considerado y un número adicional; de lo contrario
b) el número mínimo de bits relacionado con el código considerado más el número adicional, si el número mínimo de bits relacionado con el código considerado es mayor que dicho número mínimo de bits seleccionados individualmente o si no, es menor que dicho número mínimo de bits seleccionados individualmente decrementados por el número adicional;
y
- determinar (750) una duración mínima correspondiente de datos, siendo dicha longitud mínima de datos la más pequeña de entre las longitudes determinadas de datos;
comprendiendo adicionalmente dicha etapa de selección (550) de un algoritmo opcional seleccionar el algoritmo asociado a la secuencia de código que tiene la longitud más corta de entre las longitudes mínimas de datos determinados para la primera y segunda secuencias de código.

4. El método de transmisión de acuerdo con la reivindicación 3, en el que la etapa de aplicar (410) a dicho valor considerado al menos un primer algoritmo de codificación y un segundo algoritmo de codificación comprende generar (200) un primer número entero y un segundo número entero y codificar (210), dicho primer y segundo números enteros respectivamente por medio de una primera codificación binaria y una segunda codificación binaria.

5. El método de transmisión de acuerdo con la reivindicación 4, en el que dicha etapa de determinar (720), para

cada código de la secuencia considerada, un correspondiente número mínimo de bits comprende determinar el número mínimo de bits suficiente para codificar de manera alterna el primer número entero o si no, el segundo número entero relacionado con el valor numérico obtenido que corresponde a dicho código, respectivamente por medio de la primera codificación binaria o si no, la segunda codificación binaria, dependiendo de si la secuencia considerada es de manera alterna la primera secuencia de código o si no, la segunda secuencia de código.

6. El método de transmisión de acuerdo con la reivindicación 4 o la reivindicación 5, que comprende la etapa de determinación (400, 600), para cada valor numérico obtenido, por dicha unidad de detección (6), de si el valor numérico obtenido se ha obtenido de una manera síncrona o asíncrona; y en el que la etapa de determinación (640, 650) de una pluralidad de intervalos de tiempo comprende generar (530) una primera secuencia de intervalos de tiempo, comprendiendo dicha etapa de generación de una primera secuencia de intervalos de tiempo, para cada valor numérico obtenido:
- en el caso donde el valor numérico obtenido se haya obtenido de una manera síncrona, insertar (610), en la primera secuencia de intervalos de tiempo, una primera secuencia de tiempo binaria que corresponde a una diferencia de tiempo cero; de lo contrario
- en el caso donde el valor numérico obtenido se haya obtenido de una manera asíncrona, calcular (640) una diferencia entre el correspondiente instante de adquisición y el instante de muestreo que es el más reciente con respecto a dicho instante de adquisición correspondiente, codificar (650) dicha diferencia en formato binario, e insertar (660) la diferencia codificada en la primera secuencia de intervalos de tiempo;
y en el que la etapa de generación (530) de una primera secuencia de código y una segunda secuencia de código comprende, para cada valor numérico obtenido:
- en el caso donde el valor numérico obtenido se haya obtenido de una manera síncrona, insertar (610), en la primera y segunda secuencias de código, un código igual a una primera secuencia de control binaria; de lo contrario
- en el caso donde el valor numérico obtenido se haya obtenido de una manera asíncrona, insertar (660), en la primera y segunda secuencias de código, respectivamente, el primer código correspondiente y el segundo código correspondiente.

7. El método de transmisión de acuerdo con la reivindicación 6, en el que la etapa de generación (530) de una primera secuencia de código y una segunda secuencia de código comprende, para cada valor numérico obtenido, las etapas de:
- si el valor numérico obtenido se ha obtenido de una manera síncrona, verificar (620) si existe un valor numérico anterior, obtenido inmediatamente antes del valor numérico obtenido e igual al valor numérico obtenido; y
- en el caso donde dicho valor numérico anterior no exista o, si no, sea diferente de dicho valor numérico obtenido, insertar (630), en la primera y segunda secuencias de código, respectivamente, el primer y segundo códigos que corresponden al valor numérico obtenido.

8. El método de transmisión de acuerdo con la reivindicación 6 o la reivindicación 7, en el que la etapa de generación (560) de una secuencia comprimida de datos comprende:
- determinar (700-750) un número efectivo de bits de código, siendo dicho número efectivo de bits de código igual al número mínimo de bits seleccionado individualmente al que corresponde la longitud mínima de datos relacionada con la secuencia de código asociada al algoritmo óptimo;
- seleccionar, de manera alterna, la primera secuencia de código o la segunda secuencia de código, dependiendo del algoritmo óptimo seleccionado;
- seleccionar (800) cada código de la secuencia de código seleccionada y:
- en el caso donde dicho código seleccionado sea igual a la primera secuencia de control binaria, insertar (850) la primera secuencia de control binaria en la secuencia comprimida de datos;
- en el caso donde dicho código seleccionado sea diferente de la primera secuencia de control binaria, verificar (870) si el número efectivo de bits de código está comprendido entre el número mínimo de bits que corresponde al código seleccionado y la suma de dicho correspondiente número mínimo de bits y el número adicional, y:
- si el número efectivo de bits de código está comprendido entre dicho correspondiente número mínimo de bits y la suma de dicho correspondiente número mínimo de bits y el número adicional, determinar una primera versión del código seleccionado, codificando el número entero que corresponde al código seleccionado de acuerdo con el algoritmo óptimo y a dicho número efectivo de bits de código, e insertar (890) dicha primera versión en la secuencia comprimida de datos; de lo contrario
- si el número efectivo de bits de código es menor que dicho correspondiente número mínimo de bits o, si no, es mayor que la suma de dicho correspondiente número mínimo de bits y del número adicional, determinar una segunda versión del código seleccionado, codificando el número entero que corresponde al código seleccionado de acuerdo con el algoritmo óptimo y de dicho correspondiente número mínimo de bits, y a continuación insertar (880), en la secuencia comprimida de datos, una segunda secuencia de control binaria, indicando una secuencia binaria dicho correspondiente número mínimo de bits, y dicha segunda versión.

9. El método de transmisión de acuerdo con la reivindicación 8, que comprende adicionalmente las etapas de:
- para cada intervalo de tiempo de la primera secuencia de intervalos de tiempo, determinar (720) un correspondiente número de bits inferior, siendo dicho número inferior de bits el número mínimo de bits suficiente para codificar la correspondiente diferencia de tiempo;
- seleccionar individualmente cada número inferior de bits de entre los números inferiores de bits determinados;

- para cada número inferior de bits seleccionado individualmente, determinar (740) una duración de tiempo correspondiente relacionada con la primera secuencia de intervalos de tiempo, correspondiendo dicha etapa de determinación a una duración de tiempo que comprende añadir, para cada intervalo de tiempo considerado de entre los intervalos de tiempo de la primera secuencia de intervalos de tiempo;
a) el número inferior de bits seleccionado individualmente, si dicho número inferior de bits seleccionado individualmente está comprendido entre el número inferior de bits relacionado con el intervalo de tiempo considerado y la suma del número inferior de bits relacionada con el intervalo de tiempo considerado y el número adicional; de lo contrario
b) el número inferior de bits relacionado con el intervalo de tiempo considerado más el número adicional, si el número inferior de bits relacionado con el intervalo de tiempo considerado es mayor que dicho número inferior de bits seleccionado individualmente o si no, es menor que dicho número inferior de bits seleccionado individualmente decrementado por el número adicional;
y
- determinar (750) un número efectivo de bits de tiempo, igual al número inferior de bits seleccionado individualmente al que corresponde la más corta de entre las duraciones de tiempo determinadas.

10. El método de transmisión de acuerdo con la reivindicación 9, en el que la etapa de generación (560) de una secuencia comprimida de intervalos de tiempo comprende las etapas de:
- seleccionar (800) cada intervalo de tiempo de la primera secuencia de intervalos de tiempo y:
- si el número efectivo de bits de tiempo está comprendido entre el número mínimo de bits que corresponde al intervalo de tiempo seleccionado y la suma de dicho correspondiente número inferior de bits y el número adicional, determinar una primera variante del intervalo de tiempo seleccionado, codificando la diferencia de tiempo que corresponde al intervalo de tiempo seleccionado en dicho número efectivo de bits de tiempo, e insertar (890) dicha primera variante en la secuencia comprimida de intervalos de tiempo; de otra manera
- si el número efectivo de bits de tiempo es menor que dicho correspondiente número inferior de bits o si no, es mayor que la suma de dicho correspondiente número inferior de bits y el número adicional, determinar una segunda variante del intervalo de tiempo seleccionado, codificando la diferencia de tiempo que corresponde al intervalo de tiempo seleccionado en dicho correspondiente número inferior de bits, y a continuación insertar (880), en la secuencia comprimida de intervalos de tiempo, la segunda secuencia de control binaria, una secuencia binaria que indica dicho correspondiente número inferior de bits, y dicha segunda variante.

11. El método de acuerdo con la reivindicación 10, en el que dicha etapa de generación (560) de una secuencia comprimida de intervalos de tiempo y dicha etapa de generación (560) de una secuencia comprimida de datos comprenden:
- verificar (810) si, en la secuencia de código asociada al algoritmo óptimo, están presentes conjuntos de código formados por un número de códigos consecutivos igual a la primera secuencia de control binaria, siendo dicho número de códigos consecutivos al menos igual a un número umbral;
- para cada conjunto de códigos, insertar (910), en la secuencia comprimida de datos, una tercera secuencia de control binaria y una secuencia binaria que indica el número de códigos presentes en el conjunto de códigos.

12. El método de transmisión de acuerdo con la reivindicación 11, que comprende adicionalmente la etapa de:
- discriminar, por la unidad de control (2), entre primeros grupos de bits de la cadena binaria relacionada con códigos y segundos grupos de bits de la cadena binaria relacionada con intervalos de tiempo, basándose en la primera, segunda y tercera secuencias de control binarias, del número efectivo de bits de código, y del número efectivo de bits de tiempo.

13. Una unidad de detección configurada para ejecutar el método de transmisión de acuerdo con una cualquiera de las reivindicaciones 1 a 11.

14. Una red de sensores que comprende una unidad de control (2) y una unidad de detección (6) de acuerdo con la reivindicación 13 cuando depende de la reivindicación 11, estando configurada la unidad de control para discriminar entre primeros grupos de bits de la cadena binaria relacionada con códigos y segundos grupos de bits de la cadena binaria relacionada con intervalos de tiempo, basándose en la primera, segunda y tercera secuencias de control binarias, del número efectivo de bits de código y del número efectivo de bits de tiempo.

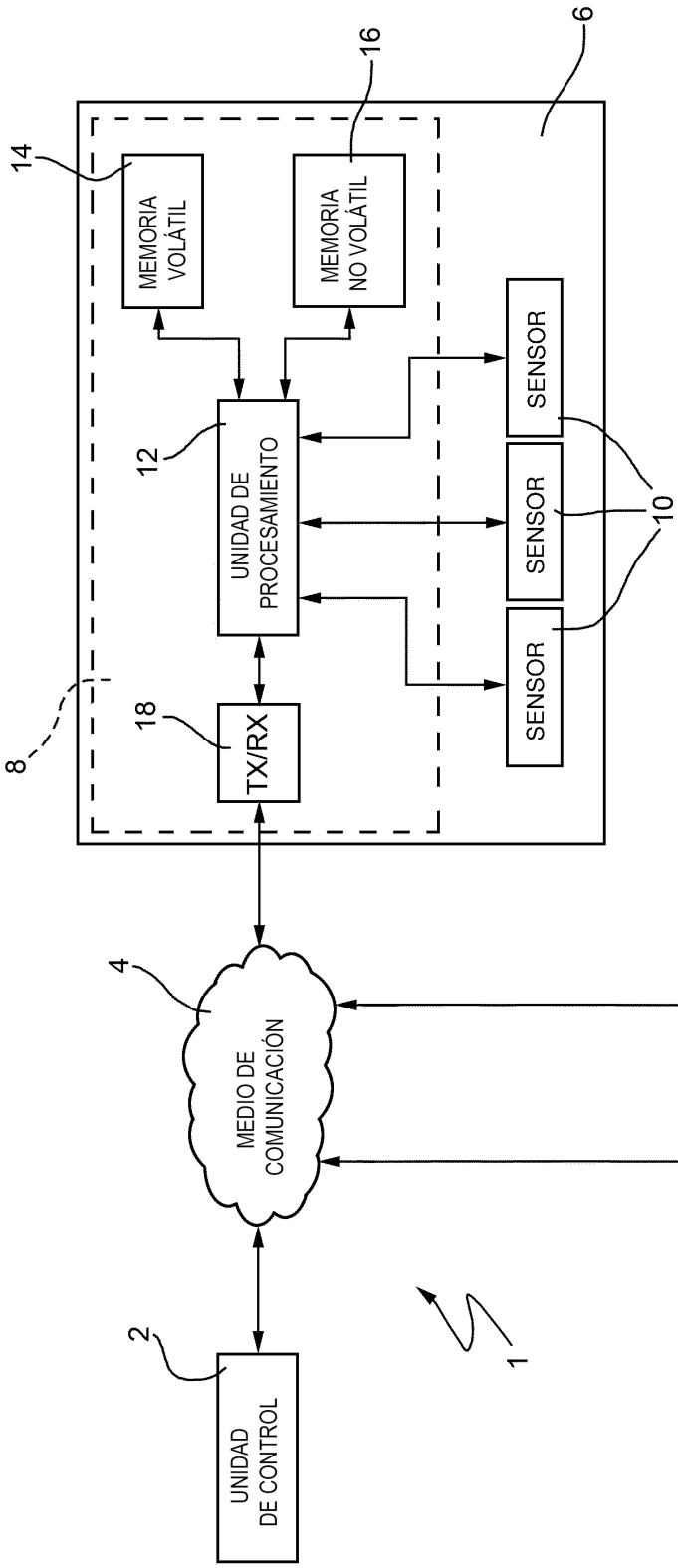
6
16
14
MEMORIA VOLÁTIL
MEMORIA NO VOLÁTIL
SENSOR
SENSOR
SENSOR
10
12
UNIDAD DE PROCESAMIENTO
8
18
TX/RX
UNIDAD DE DETECCIÓN
6
UNIDAD DE DETECCIÓN
6
4
MEDIO DE COMUNICACIÓN
2
UNIDAD DE CONTROL
1

FIG. 1

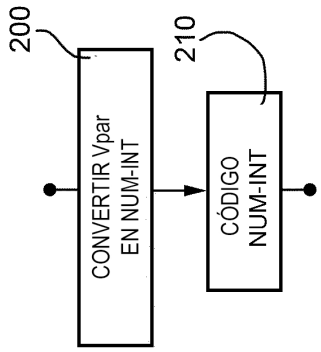
200
CONVERTIR Vpar EN NUM-INT
210
CÓDIGO NUM-INT

FIG. 2

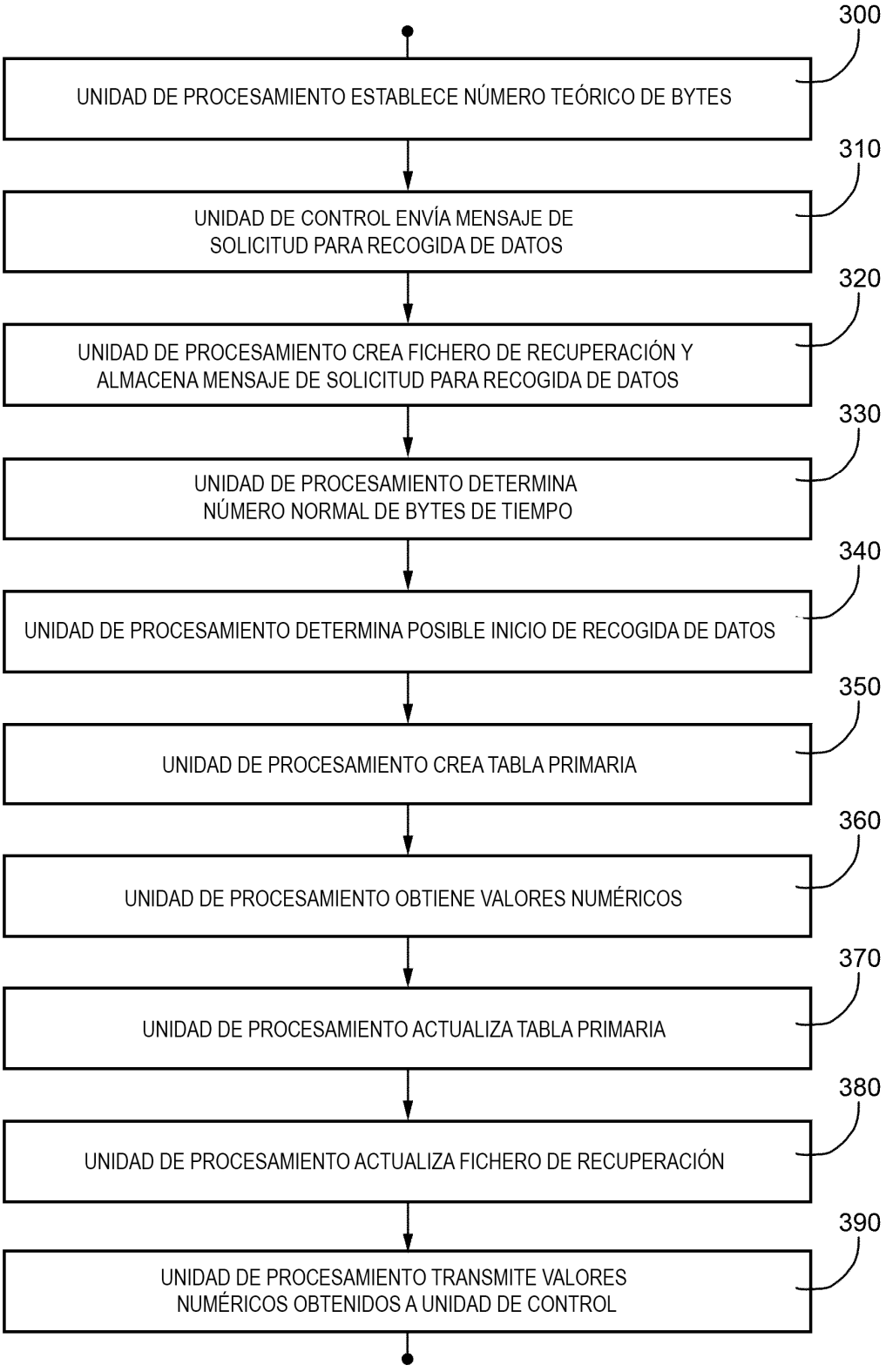
300
UNIDAD DE PROCESAMIENTO ESTABLECE NÚMERO TEÓRICO DE BYTES
310
UNIDAD DE CONTROL ENVÍA MENSAJE DE SOLICITUD PARA RECOGIDA DE DATOS
320
UNIDAD DE PROCESAMIENTO CREA FICHERO DE RECUPERACIÓN Y ALMACENA MENSAJE DE SOLICITUD PARA RECOGIDA DE DATOS
330
UNIDAD DE PROCESAMIENTO DETERMINA NÚMERO NORMAL DE BYTES DE TIEMPO
340
UNIDAD DE PROCESAMIENTO DETERMINA POSIBLE INICIO DE RECOGIDA DE DATOS
350
UNIDAD DE PROCESAMIENTO CREA TABLA PRIMARIA
360
UNIDAD DE PROCESAMIENTO OBTIENE VALORES NUMÉRICOS
370
UNIDAD DE PROCESAMIENTO ACTUALIZA TABLA PRIMARIA
380
UNIDAD DE PROCESAMIENTO ACTUALIZA FICHERO DE RECUPERACIÓN
390
UNIDAD DE PROCESAMIENTO TRANSMITE VALORES NUMÉRICOS OBTENIDOS A UNIDAD DE CONTROL

FIG. 3

20

IDENTIFICADOR DE INTERVALO	VALOR DE TIEMPO	TIPO	PARÁMETRO	ALGORITMO 1	ALGORITMO 2
1	15:20.00	C	5,2	0x34 (52)	0xFB0034FF (52)
2	15:21.00	C	5,2	0x34 (52)	0x00 (0)
	15:21.12	R	5,3	0x35 (53)	0x01 (1)
	15:21.27	R	5,4	0x36 (54)	0x01 (1)
3	15:22.00	C	5,4	0x36 (54)	0x00 (0)
4	15:23.00	C	5,4	0x36 (54)	0x00 (0)
5	15:24.00	C	5,4	0x36 (54)	0x00 (0)
6	15:25.00	C	5,4	0x36 (54)	0x00 (0)
	15:25.10	R	5,7	0x39 (57)	0x03 (3)

FIG. 4

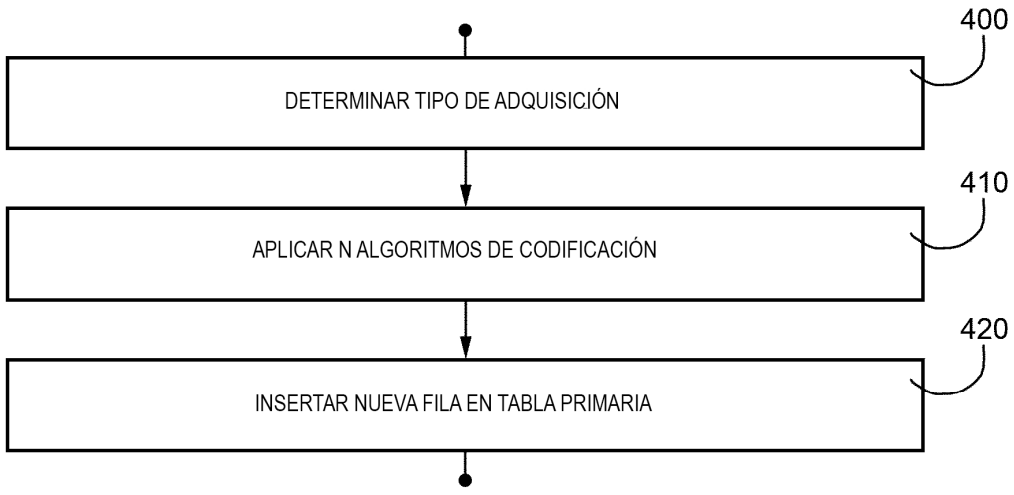

FIG. 5

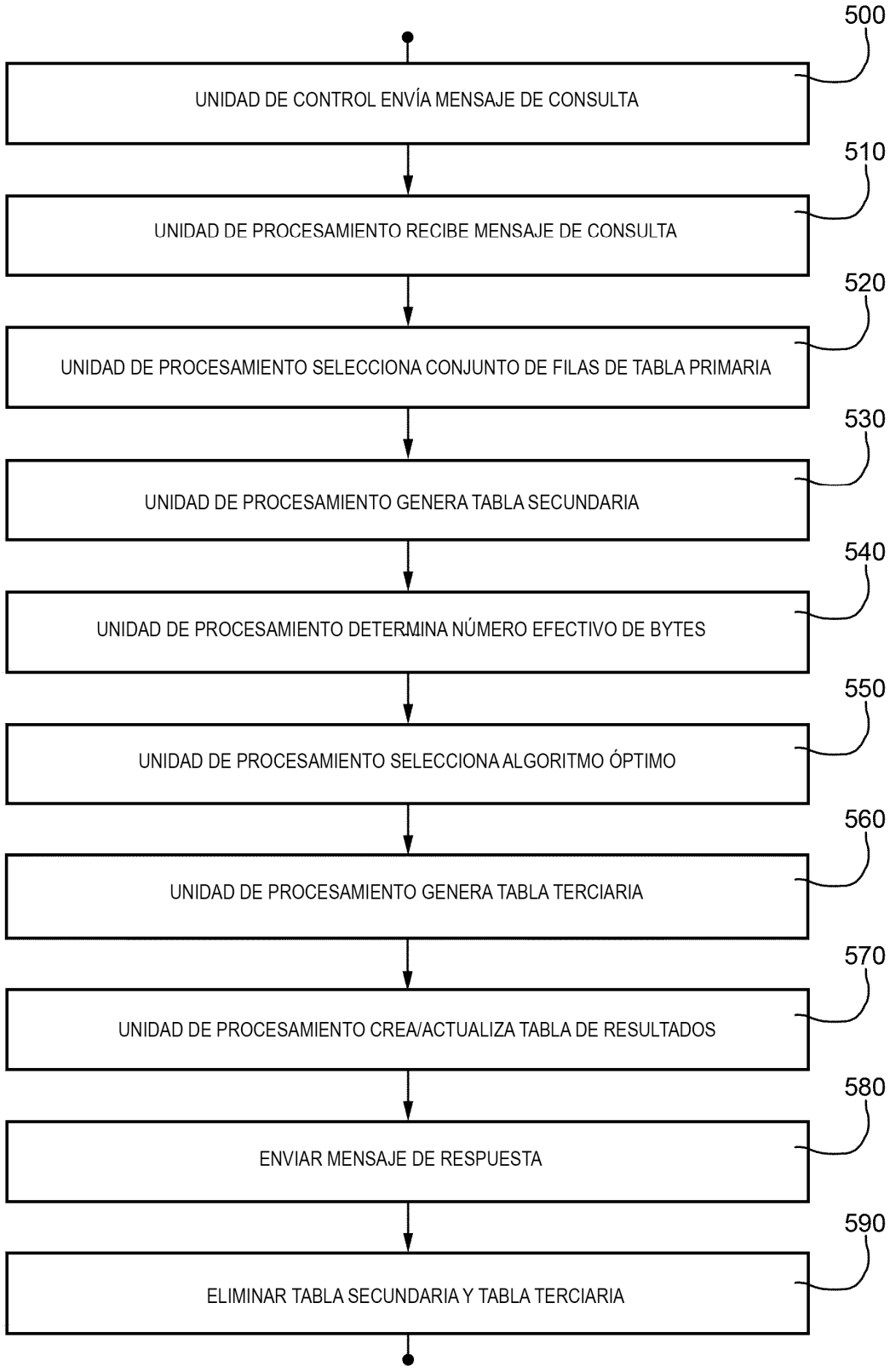
500
UNIDAD DE CONTROL ENVÍA MENSAJE DE CONSULTA
510
UNIDAD DE PROCESAMIENTO RECIBE MENSAJE DE CONSULTA
520
UNIDAD DE PROCESAMIENTO SELECCIONA CONJUNTO DE FILAS DE TABLA PRIMARIA
530
UNIDAD DE PROCESAMIENTO GENERA TABLA SECUNDARIA
540
UNIDAD DE PROCESAMIENTO DETERMINA NÚMERO EFECTIVO DE BYTES
550
UNIDAD DE PROCESAMIENTO SELECCIONA ALGORITMO ÓPTIMO
560
UNIDAD DE PROCESAMIENTO GENERA TABLA TERCIARIA
570
UNIDAD DE PROCESAMIENTO CREA/ACTUALIZA TABLA DE RESULTADOS
580
ENVIAR MENSAJE DE RESPUESTA
590
ELIMINAR TABLA SECUNDARIA Y TABLA TERCIARIA

FIG. 6

22

IDENTIFICADOR DE INTERVALO	DESPLAZAMIENTO DE TIEMPO	ALGORITMO 1	ALGORITMO 2
1	0x0000 (0)	0xF6	0xF6
	0x0000 (0)	0x34 (52)	0xFB0034FF (52)
2	0x0000 (0)	0xF6	0xF6
	0x000C (12)	0x35 (53)	0x01 (1)
	0x001B (27)	0x36 (54)	0x01 (1)
3	0x0000 (0)	0xF6	0xF6
4	0x0000 (0)	0xF6	0xF6
5	0x0000 (0)	0xF6	0xF6
6	0x0000 (0)	0xF6	0xF6
	0x000A (10)	0x39 (57)	0x03 (3)

FIG. 7

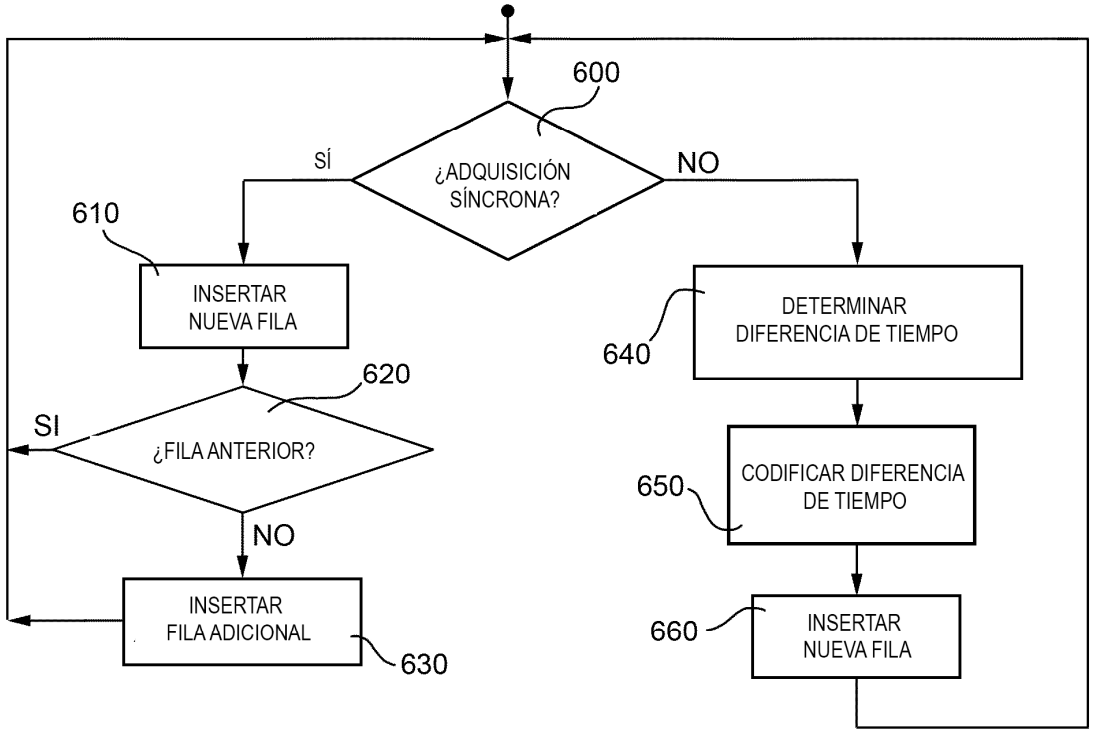

FIG. 8

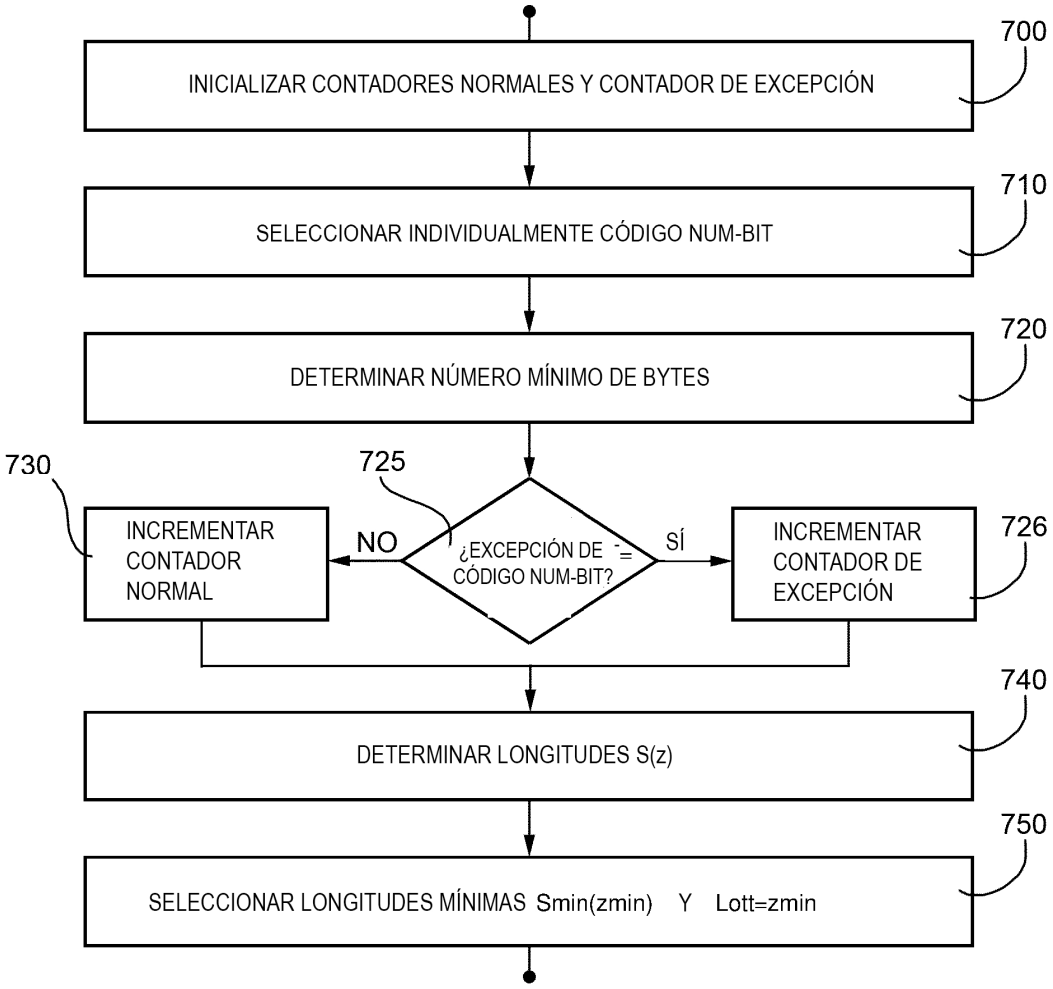

FIG. 9

24

IDENTIFICADOR DE INTERVALO	DESPLAZAMIENTO DE TIEMPO	ALGORITMO 1	ALGORITMO 2
1	0x00 (0)	0xF6	0xF6
	0x00 (0)	0x34 (52)	0xFB0034FF (52)
2	0x00 (0)	0xF6	0xF6
	0x0C (12)	0x35 (53)	0x01 (1)
	0x1B (27)	0x36 (54)	0x01 (1)
3	0x00 (0)	0xFA04F6	0xFA04F6
	0x0A (10)	0x39 (57)	0x03 (3)

FIG. 10

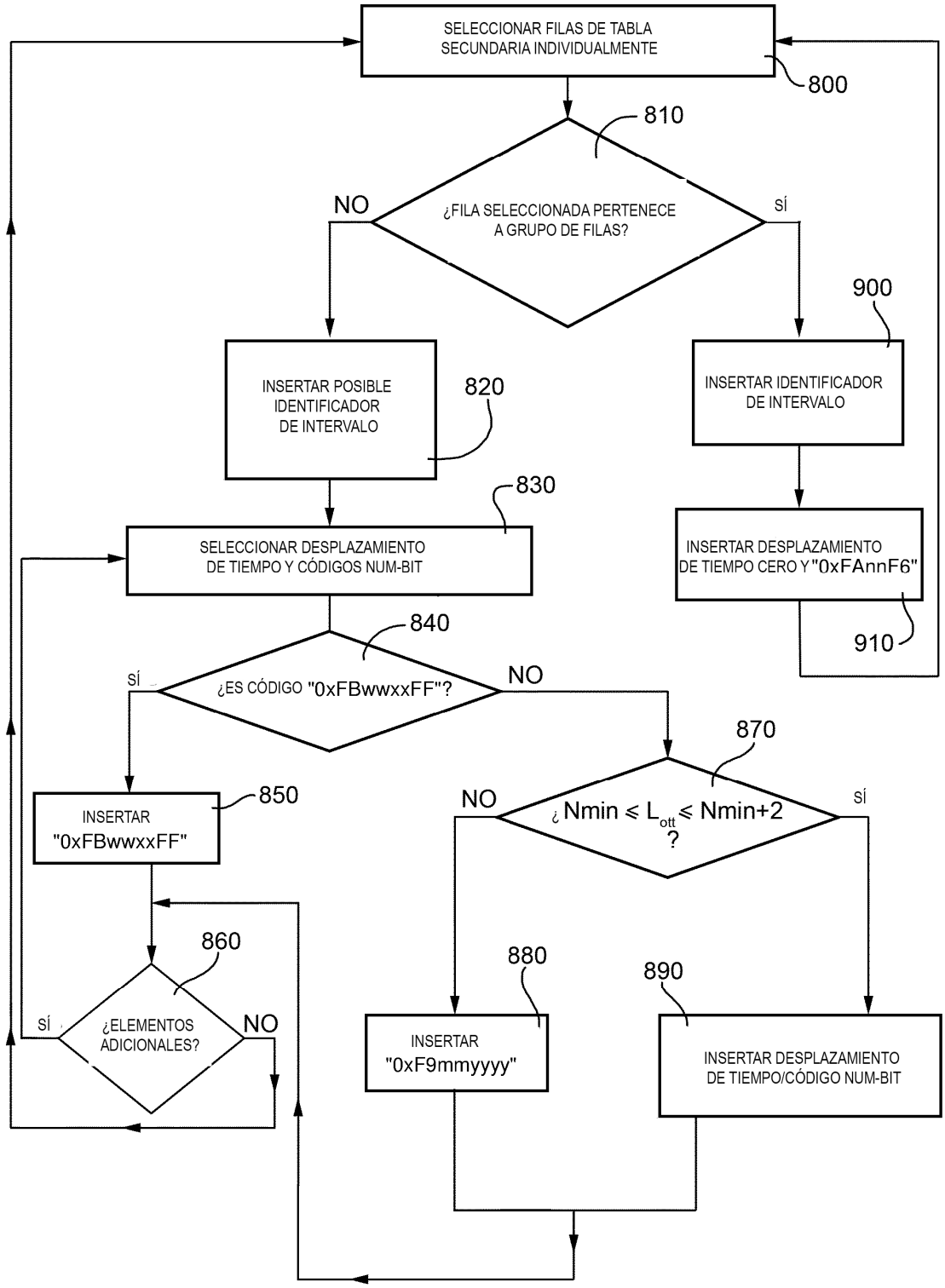
SELECCIONAR FILAS DE TABLA SECUNDARIA INDIVIDUALMENTE
800
810
¿FILA SELECCIONADA PERTENECE A GRUPO DE FILAS?
NO
SÍ
INSERTAR POSIBLE IDENTIFICADOR DE INTERVALO
820
900
INSERTAR IDENTIFICADOR DE INTERVALO
830
SELECCIONAR DESPLAZAMIENTO DE TIEMPO Y CÓDIGOS NUM-BIT
INSERTAR DESPLAZAMIENTO DE TIEMPO CERO Y "0xFAnnF6"
910
840
¿ES CÓDIGO "0xFBwwxxFF"?
SÍ
NO
870
INSERTAR "0xFBwwxxFF"
850
NO
¿ Nmin ⩽ L ott ⩽ Nmin+2 ?
SÍ
860
¿ELEMENTOS ADICIONALES?
SÍ
NO
880
INSERTAR "0xF9mmyyyy"
890
INSERTAR DESPLAZAMIENTO DE TIEMPO/CÓDIGO NUM-BIT

FIG. 11

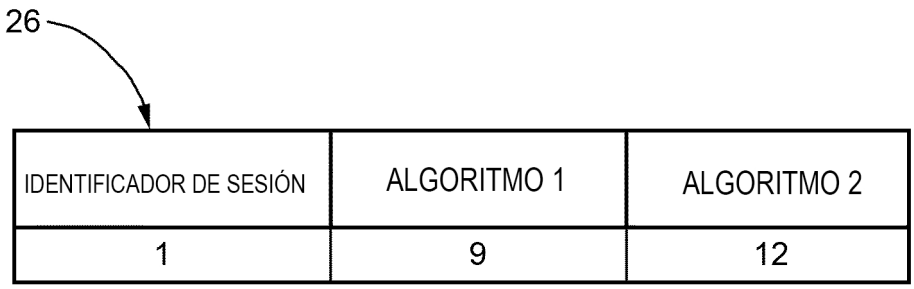

FIG. 12

0xF60034F60C351B36FA04F60A39

FIG. 13

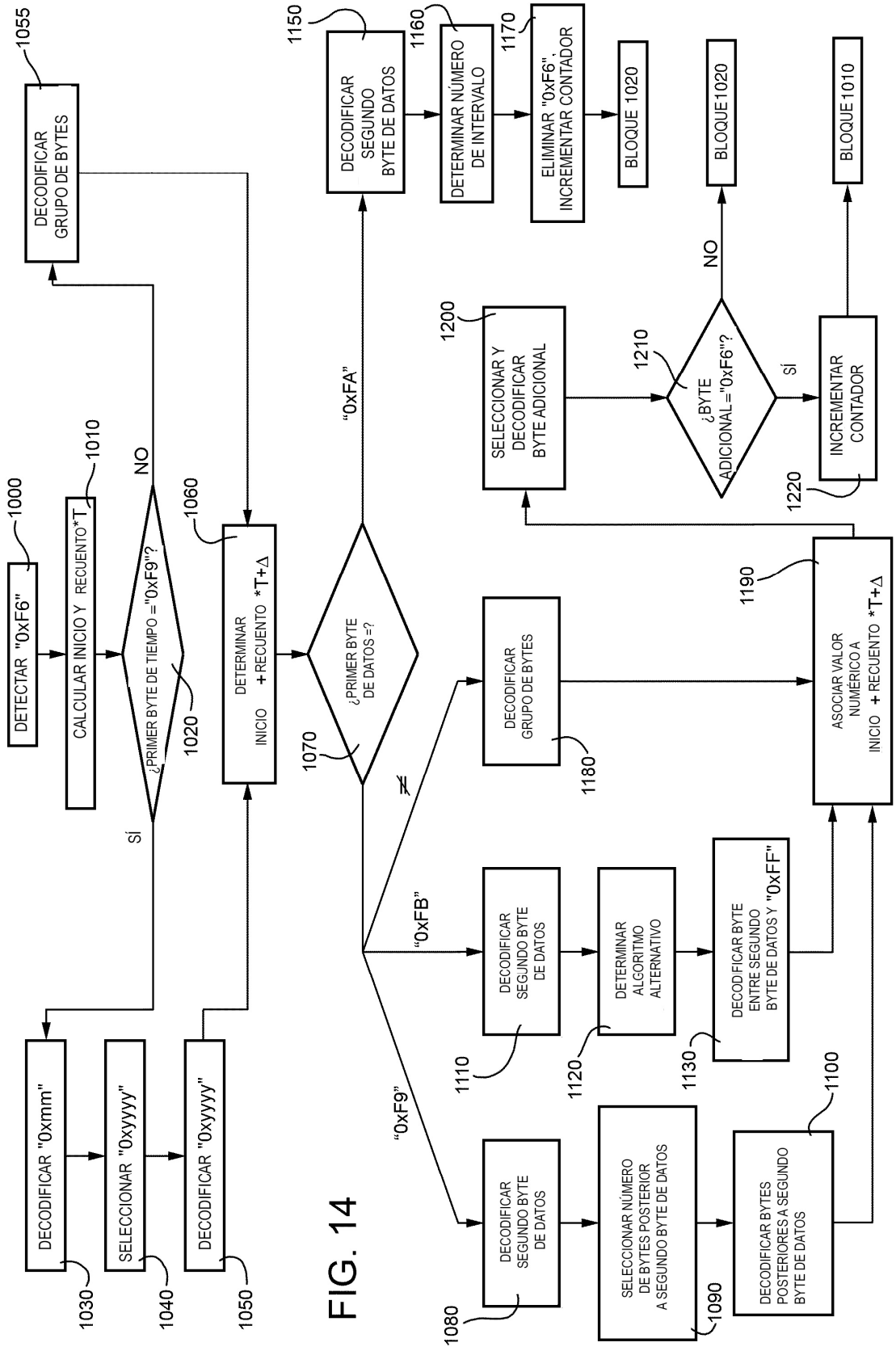
1000
DETECTAR "0xF6"
1010
CALCULAR INICIO Y RECUENTO *T
¿PRIMER BYTE DE TIEMPO = "0xF9"?
NO
Sí
1020
1030
DECODIFICAR "0xmm"
1040
SELECCIONAR "0xyyyy"
1050
DECODIFICAR "0xyyyy"
1055
DECODIFICAR GRUPO DE BYTES
1060
DETERMINAR INICIO + RECUENTO *T+Δ
1070
¿PRIMER BYTE DE DATOS =?
"0xFA"
"0xFB"
"0xF9"
≠
1080
DECODIFICAR SEGUNDO BYTE DE DATOS
1090
SELECCIONAR NÚMERO DE BYTES POSTERIOR A SEGUNDO BYTE DE DATOS
1100
DECODIFICAR BYTES POSTERIORES A SEGUNDO BYTE DE DATOS
1110
DECODIFICAR SEGUNDO BYTE DE DATOS
1120
DETERMINAR ALGORITMO ALTERNATIVO
1130
DECODIFICAR BYTE ENTRE SEGUNDO BYTE DE DATOS Y "0xFF"
1150
DECODIFICAR SEGUNDO BYTE DE DATOS
1160
DETERMINAR NÚMERO DE INTERVALO
1170
ELIMINAR "0xF6", INCREMENTAR CONTADOR
BLOQUE 1020
1180
DECODIFICAR GRUPO DE BYTES
1190
ASOCIAR VALOR NUMÉRICO A INICIO + RECUENTO *T+Δ
1200
SELECCIONAR Y DECODIFICAR BYTE ADICIONAL
1210
¿BYTE ADICIONAL = "0xF6"?
NO
BLOQUE 1020
Sí
1220
INCREMENTAR CONTADOR
BLOQUE 1010

FIG. 14