4 Implementación

Una vez descritas las funcionalidades del sistema y la arquitectura lógica diseñada para llevar a cabo dichas funcionalidades es el momento de exponer el modo en que todos estos elementos han sido implementados.

En este apartado se mostrará en detalle la estructura interna de cada una de las tareas, funciones e interrupciones que forman parte de la arquitectura del sistema.

También se mostraran otros elementos de software necesarios para el correcto funcionamiento de la arquitectura domótica, como pueden ser las variables globales definidas o los tipos de datos y formatos usados.

4.1 Constantes

Antes de comenzar con la descripción de la implementación del sistema es necesario conocer las constantes definidas en el código para comprender correctamente el resto del mismo.

Las constantes que se corresponden con campos de tramas se describirán en el apartado 4.3. Igualmente, aquellas que definen los modos de funcionamiento y las operaciones con entradas y salidas se mostrarán en el apartado 4.4. Aquellas que se usan para definir los estados de las comunicaciones se verán en el apartado 4.8.3.

4.1.1 Constantes para comunicaciones

Las constantes definidas para las comunicaciones son las siguientes:

- **MAXMENSAJES**: Número máximo de mensajes que pueden albergar los búferes de entrada y de salida.
- **MAXLONGITUD**: Longitud máxima de los mensajes que se pueden almacenar en los búferes de entrada y de salida.
- **HEADER_SIZE**: Tamaño de la cabecera de datos.
- **MAX_RETX**: Número máximo de retransmisiones permitidas.

4.1.2 Constantes generales

- **OVERFLOW**: Valor de desbordamiento de los temporizadores de 16 bits.
- **DIMMER_SCALE**: Factor de multiplicación para convertir los niveles de potencia deseados de los dimmers en los valores de contador que deben escribirse en los registros de los temporizadores.
- **DIMMER_POT_VMAX**: Valor máximo permitido para el nivel de potencia de los dimmers.
- **DIMMER_POT_VMIN**: Valor mínimo permitido para el nivel de potencia de los dimmers.
- **DIMMER_POT_GAP**: Valor umbral con respecto al máximo y al mínimo a partir del cual no se diferencian niveles de potencia. Este umbral se usa para evitar efectos inesperados si el valor de potencia deseado está demasiado cerca de los valores límite (véase 4.7.2).
- **NULL_BYTE**: Byte con todos sus bits a cero.
- **FULL_BYTE**: Byte con todos sus bits a uno.

- **LS_4BITS**: Máscara para extraer los 4 bits menos significativos de un byte.
- **MS_4BITS**: Máscara para extraer los 4 bits más significativos de un byte.
- **PULS_EXTRA_LARGA**: Número de ciclos que debe detectarse una entrada local activa para considerarla una pulsación de longitud mayor a la corta y a la larga.
- **PULS_CORTA**: Número de ciclos de detección de entrada activa por debajo del cual la pulsación se considera corta.
- **MAX_NODOS**: Número máximo de nodos que puede haber en la red domótica. Se usa para dimensionar la tabla de direcciones.
- **MAX_ENTRADAS**: Número máximo de entradas que puede haber en la configuración.
- **NUM_SALIDAS**: Número de salidas locales.
- **NUM_ENTRADAS**: Número de entradas locales.
- **ACKTOUT_CICLOS**: Número de ciclos del temporizador de expiración de asentimiento.
- **RXTOUT_CICLOS**: Número de ciclos del temporizador de expiración de recepción.
- **Sx_DIR**: Direcciones en EEPROM interna donde almacenar los estados actuales de cada salida local.

4.2 Variables globales

Vamos a clasificar las variables globales según el módulo en que se utilizan. Algunas de ellas servirán para intercambiar datos entre varios módulos, de modo que su ámbito de validez se extiende a más de un elemento funcional del sistema.

4.2.1 Variables del módulo principal

- **int8 Ent[MAX_ENTRADAS]**: Tabla donde se guarda la lista de entradas de la configuración una vez leía de la EEPROM e interpretada.
- **int8 Sal[MAX_ENTRADAS]**: Tabla donde se guarda la lista de salidas de la configuración una vez leía de la EEPROM e interpretada.
- **int8 Mod[MAX_ENTRADAS]**: Tabla donde se guarda la lista de modos de la configuración una vez leía de la EEPROM e interpretada.
- **int8 Ope[MAX_ENTRADAS]**: Tabla donde se guarda la lista de operaciones de la configuración una vez leía de la EEPROM e interpretada.
- **int8 Dir[MAX_ENTRADAS]**: Tabla donde se guarda la lista de direcciones de la configuración una vez leía de la EEPROM e interpretada.
- **int8 DireccionPropia**: Variable donde se almacena la dirección propia leída de la EEPROM.

4.2.2 Variables del módulo de comunicaciones

- **enum estados_bus {...} EstadoBus:** Estado actual del bus de datos. Esta variable se usa para saber qué campo de una trama se está esperando recibir del bus en un momento dado. Sus diferentes valores se verán mas adelante, en el apartado 4.8.3.
- **int8 BufferEntrada[MAXMENSAJES][MAXLONGITUD]:** Búfer de entrada de datos del bus.
- **int8 PreBufferEntrada[MAXLONGITUD]:** Búfer donde se almacenan las tramas mientras estas están siendo recibidas. Su objetivo es evitar que la interrupción de recepción de bytes tenga que operar con una tabla bidimensional, ya que ello supone mucho más procesado que el caso de una tabla unidimensional.
- **int8 BufferSalida[MAXMENSAJES][MAXLONGITUD]:** Búfer de salida de datos hacia el bus.

- **int8 Ventana[MAXLONGITUD]:** Variable para almacenar la trama que esté pendiente de asentimiento.
- **int8 SiguienteHuecoE:** Indicador de la posición del búfer de entrada donde se debe escribir la próxima trama que se reciba.
- **int8 SiguienteHuecoS:** Indicador de la posición del búfer de salida donde se debe escribir la próxima trama que se desee enviar.
- **int1 BufferLleno:** Flag para saber cuándo el búfer de entrada está lleno.
- **int1 TramaRecibida:** Flag para indicar a la función de recogida de tramas que existe una trama en el "prebúfer" lista para ser movida al búfer de entrada.
- **int8 UltimoByte:** Variable que permite a la interrupción de recepción de bytes, cuál es el último byte que se espera recibir.
- **int8 Checksum:** Checksum de las tramas recibidas. Se calcula conforme se reciben los bytes de la trama.
- **int1 RxToutCEnable:** Habilitador de la temporización de expiración de recepción de tramas. Se activa al comenzar a recibir una trama y se desactiva una vez que esta se recibe completamente.
- **int1 AckToutCEnable:** Habilitador de la temporización de expiración de espera de asentimiento. Se activa al enviar una trama que requiere de asentimiento y se desactiva al recibir dicho asentimiento.
- **int8 RxToutCValue:** Contador usado para la temporización de la expiración de la recepción de tramas.
- **int8 AckToutCValue:** Contador usado para la temporización de la expiración de espera de asentimiento.
- **int8 NumeroNodos:** Número de nodos diferentes detectados hasta el momento. Esta variable permite conocer el tamaño de la tabla de direcciones donde se almacenan los datos relativos a las comunicaciones con cada nodo de la red (direcciones, números de secuencia y tipos de tramas enviadas).
- **int8 Direcciones[MAX_NODOS][4]:** Tabla donde se almacenan las direcciones de los nodos con los que se ha establecido comunicación hasta el momento. Sus campos contienen información sobre la dirección, el número de secuencia en transmisión (NT) recibido por última vez de dicha dirección, el número de secuencia en recepción (NR) esperado de dicha dirección (que se corresponde con el último NT enviado a dicha dirección) y el tipo de trama enviada por última vez a dicho nodo.

- **int1 VentanaLlena:** Indicador de ventana llena. Este flag se activa cuando se transmite una trama que requiere de asentimiento. Mientras esté activa no se pueden transmitir tramas que requieran de asentimiento, salvo si se trata de una retransmisión. El flag se desactiva cuando se recibe en asentimiento correctamente.
- **int8 Retransmisiones:** Contador de retrasmisiones consecutivas.
- **int1 EsReTx:** Indica a la función de envío de tramas que se pretende realizar una retransmisión.
- **int1 SoloCapturaDirOrig:** Flag para activar la captura de direcciones de origen en caso de se detecte una trama de la que no se es el nodo destinatario. El objetivo de este mecanismo es rellenar la tabla de direcciones antes de que se reciban tramas de todos los nodos.
- **int8 DirAComprobar:** Variable para guardar la dirección que se desea insertar o buscar en la tabla de direcciones.

4.2.3 Variables de módulo de gestión de entradas y salidas

- **int8 EntradaActual:** Valor actual de las entradas locales del nodo.
- **int8 EntradaAnterior:** Valor anterior de las entradas locales del nodo.
- **int8 ContPulsacion[NUM_ENTRADAS]:** Contadores de duración de la activación de las entradas locales.
- **enum eOpCont {Nop, Dec, Dec5, Res} OpCont[NUM_ENTRADAS]:** Operación a realizar con cada contador una vez finalizado el procesado de las entradas locales. Las posibles operaciones con los contadores son: Nop (ninguna operación), Dec (decrementar en una unidad), Dec5 (decrementar en 5 unidades), Res (poner a cero).
- **enum eOperacion {ConmutarL, ApagarL, EncenderL, CPotenciaL, FPotenciaL, ConmutarR, ApagarR, EncenderR, CPotenciaR, FPotenciaR, CPotenciaL5, Ninguna} Operacion[NUM_SALIDAS]:** Array de variables usado por la función de captura de eventos para indicar a la función de activación de salidas la operación que debe realizar con cada salida. Sus posibles valores son: ConmutarL (conmutar una salida local), ApagarL (desactivar una salida local), EncenderL (activar una salida local), CPotenciaL (cambiar la potencia de una salida local regulada en intensidad en una unidad), FPotenciaL (fijar la potencia de una salida local regulada a un valor concreto), ConmutarR, ApagarR, EncenderR, CPotenciaR, FPotenciaR (las terminadas en "R" operan con salidas remotas), CPotenciaL5 (cambiar la potencia de una salida local regulada en intensidad en 5 unidades), Ninguna.

- **enum eEstado {Apagado, Encendido} Estado[NUM_SALIDAS]:** Esta tabla de variables permite conocer el estado actual de cada una de las salidas locales del nodo.
- **int8 SentidoCambio:** Byte que contiene un bit por cada dimmer (sobran 6) y cuyo estado (cero o uno) indica el sentido de cambio de la potencia de dicho dimmer.
- **int8 PotDimmer[2]:** Tabla con los valores actuales de potencia de cada uno de los dimmers.
- **int8 DirEnvio[NUM_SALIDAS]:** Variable usada para el intercambio de información entre la función de captura de eventos y de generación de acciones para indicar, en caso de salida remota, la dirección de envío.
- **int8 Control[NUM_SALIDAS][2]:** Variable usada para el intercambio de información entre la función de captura de eventos y de generación de acciones para indicar, en caso de salida remota, el tipo de trama a enviar.
- **int8 Longitud[NUM_SALIDAS]:** Variable usada para el intercambio de información entre la función de captura de eventos y de generación de acciones para indicar, en caso de salida remota, la longitud de la trama a enviar.
- **int8 Info[NUM_SALIDAS][MAXLONGITUD-HEADER_SIZE]:** Variable usada para el intercambio de información entre la función de captura de eventos y de generación de acciones para indicar, en caso de salida remota, la información que debe contener la trama a enviar.
- **int8 RemoteOp[NUM_SALIDAS]:** Array de indicadores que permiten a la función de captura de eventos informar a la función de activación de salidas que se va a realizar una operación remota. Esto se hace para actualizar la tabla de direcciones con la información relativa a la comunicación que está a punto de establecerse.

4.3 Formato de las tramas

En lo relativo a las comunicaciones, un buen número de parámetros vienen definidos por el estándar RS485 (como pudimos ver en el apartado 2.2.4). La mayor parte de estos parámetros hacen referencia a características de nivel físico tanto del medio de transmisión como de las señales eléctricas que se transmiten por él.

La definición del formato de las tramas de información que se van a transmitir a través del bus es el primer paso a nivel lógico que se debe realizar para definir un protocolo de comunicaciones, uno de los objetivos del presente proyecto.

En este apartado se hará una descripción del formato físico de las tramas de información que se usaran para el intercambio de datos entre los diferentes nodos de la red domótica.

4.3.1 Campos de las tramas

Como es habitual, se han definido varios campos de datos para dar formato a las tramas del protocolo de comunicaciones. Una vez definidos dichos campos, todas las tramas respetarán dicha estructura, variando únicamente la información que contiene cada uno de ellos.

En primer lugar se describirán dichos campos, para, posteriormente, pasar a enumerar los diferentes tipos de tramas y el contenido de dichos campos para cada una de ellas.

4.3.1.1 Bandera inicial

La bandera inicial (definida a través de la constante T_FLAG) es un byte cuya presencia permite a todo nodo que se encuentre en estado de espera la detección del inicio de recepción de una trama.

Su valor es ‘01111110’. Dicho valor puede repetirse dentro de la trama, puesto que sólo se espera recibir una bandera si el nodo se encuentra en estado ocioso en lo que a comunicaciones se refiere.

4.3.1.2 Dirección destino

La dirección destino de la trama es el primer byte que se recibe tras la detección de una bandera válida.

La dirección 0 se reserva para tramas de difusión, de modo que es posible direccionar hasta 255 nodos diferentes.

4.3.1.3 Dirección origen

A continuación aparece la dirección origen de la trama. Esta dirección permite al nodo destino de la trama buscar en la tabla de direcciones los valores de los números de secuencia necesarios para el correcto funcionamiento del mecanismo de asentimiento.

La dirección por defecto de cada nodo se encuentra definida en la constante T_DIR_PROP. La dirección de difusión se encuentra definida a través de la constante T_DIR_DIF.

4.3.1.4 Campo control

Este campo, cuya longitud es de dos bytes, permite distinguir los tipos de tramas. Contiene varios subcampos. Son los siguientes:

- **Tipo:** 2 bits que indican el tipo de trama. En la Tabla 4.1 se pueden ver los tipos de tramas definidos y el valor de este subcampo para cada uno de ellos. Las constantes T_TDAT (datos), T_TACK (asentimiento) y T_TCTL (control) se usan para definir los tipos de tramas.
- **Subtipo:** 6 (o 5) bits que indican el subtipo de trama. Las constantes usadas para definir estos valores son: T_TDAT_DCFG (datos de configuración), T_TCTL_CDIR (comprobación de dirección), T_TCTL_IDRE (indicación de dirección repetida), T_TCTL_ASAL (activación de salida), T_TCTL_DSAL (desactivación de salida), T_TCTL_CSAL (conmutación de salida), T_TCTL_FPOT (fijar potencia) y T_TCTL_CPOT (cambiar potencia). Los diferentes subtipos de tramas se expondrán en apartados posteriores.
- **Extensión:** Sólo es aplicable en tramas de datos, indicando así la última trama de una secuencia segmentada. Si una trama de datos posee E = 1, indica que existen más tramas de datos con el resto de la información a transmitir. Si E = 0, indica que dicha trama es la última de la secuencia segmentada.
- **NT:** 4 bits que contienen el número de secuencia en transmisión.
- **NR:** 4 bits que contienen el número de secuencia en recepción.

Tipo	Valor
Trama de datos	00
Trama de asentimiento	10
Trama de control	01

Tabla 4.1 Tipos de tramas

Los diferentes tipos de tramas se distinguen por el campo de control. A partir de dicho campo, se conoce la estructura del resto de la trama a la hora de ser interpretada.

Los diferentes tipos de tramas que se han definido a priori son: Tramas de **Datos** (TDAT), tramas de **Asentimiento** (TACK) y tramas de **Control** (TCTL).

4.3.1.5 Campo longitud

Número de bytes del campo de datos. Si el campo de datos no aparece, el valor de este campo es cero. Su valor máximo es 255. Cantidades mayores de datos deben segmentarse en varias tramas.

4.3.1.6 Campo de datos

En este campo, cuya longitud viene indicada por el campo anterior, se introducen los datos que debe llevar la trama.

Dichos datos dependen del tipo de trama que se trate. En siguientes apartados se describirán los datos que lleva cada tipo de trama en este campo.

4.3.1.7 Campo de código Cheksum

Como mecanismo de detección de errores se utiliza un código checksum. Dicho código se calcula haciendo la operación lógica XOR entre todos los bytes de la trama a excepción de la bandera inicial.

El código checksum se inicializa con el valor 0xFF antes de proceder con esta operación lógica. Es decir, que se parte del valor 0xFF y se hace el XOR con todos los bytes de la trama a partir de la dirección de destino.

Finalmente, este byte se incluye en las tramas justo detrás del campo de datos, en caso de existir, o detrás del campo longitud.

4.3.2 Tramas de datos

En las tramas de datos, el subcampo tipo tiene el valor '00'. Sólo se ha definido un subtipo dentro del tipo de tramas de datos: Datos de Configuración.

Subtipo	Valor
Datos de Configuración	00000

Tabla 4.2 Tipos de tramas de datos

El campo subtipo tiene cinco bits de longitud en el caso de las tramas de datos, puesto que el sexto bit (bit de extensión) se reserva para la segmentación de tramas, en caso de que estas contengan más datos de lo que se pueden transmitir en una sola trama.

4.3.3 Tramas de asentimiento

Las tramas de asentimiento tienen el valor '10' en el subcampo tipo. Sólo existe un tipo de tramas de asentimiento, por lo que el resto del primer byte del campo de control se encuentra a cero.

4.3.4 Tramas de control

Las tramas de control son las más numerosas en el protocolo definido. Estas tramas son las que se utilizan para enviar instrucciones y comandos a nodos remotos de la red domótica. Cada subtipo de trama define una instrucción a ejecutar de manera remota. En la Tabla 4.3 se pueden observar todas las instrucciones posibles.

Subtipo	Valor
Comprobación de Dirección	000000
Indicación de Dirección Repetida	000001
Activación de Salida	000010
Desactivación de Salida	000011
Conmutación de Salida	000100
Incremento Potencia Dimmer	000101
Decremento Potencia Dimmer	000110

Tabla 4.3 Tipos de tramas de control

4.4 Formato de la configuración

Como ya se dijo en el apartado 3.2.5, la información que contiene la configuración es la siguiente:

- Dirección propia.
- Asignaciones de entradas con salidas.
- Modos de funcionamiento.
- Operaciones a realizar con las salidas.
- Direcciones de envío de datos.

El formato de la configuración es el siguiente:

Dirección	Longitud	Entrada	Salida	Modo	Operación	Dirección
		...	...	...	...	...
		Entrada	Salida	Modo	Operación	Dirección

Tabla 4.4 Formato de la configuración

El primer campo de Dirección es la dirección propia del nodo remoto. El campo longitud contiene la longitud total del resto de la configuración, es decir, un múltiplo de cinco.

Cada uno de los conjuntos de cinco bytes que vienen a continuación define una asignación entre una entrada y una salida ya sea a través de un modo de funcionamiento o de una operación concreta.

- **Entrada**: Su valor puede ser cualquiera de las siguientes constantes: C_IN_AN0, C_IN_AN1, C_IN_AN2, C_IN_AN3, C_IN_AN4 o C_IN_RC0. El valor de estas constantes no solo se usa para diferenciar salidas, si no que también permite comprobar el estado de dicha entrada dentro del byte que representa el estado actual de las entradas del sistema (EntradaActual).
- **Salida**: Su valor puede ser C_OUT_B1, C_OUT_B2, C_OUT_B3, C_OUT_B4, C_OUT_B5, C_OUT_B6, C_OUT_B7 o C_OUT_A4. Estos valores permiten al sistema indexar la tabla de operaciones a realizar con las salidas.
- **Modo**: Los posibles modos (que se vieron en el apartado 3.2.6) son:

- ü M_LOC_DIG_NOR_TOG (local digital normal toggle).
- ü M_LOC_DIG_NOR_PUL (local digital normal pulsador).
- ü M_LOC_DIG_NOR_DIM (local digital normal dimmer).
- ü M_REM_DIG_NOR_TOG (remoto digital normal toggle).
- ü M_REM_DIG_NOR_PUL (remoto digital normal pulsador).
- ü M_REM_DIG_NOR_DIM (remoto digital normal dimmer).

Si este parámetro tiene un valor diferente de cero (que se corresponde con la ausencia de asignación de modo de funcionamiento), el siguiente parámetro (operación) debe ser nulo. Por tanto, no puede haber una asignación de modo y de operación simultáneamente a un mismo par entrada-salida. En caso de darse, prevalece la operación.

- **Operación**: Las diferentes operaciones posible con la salida son O_LOC_ASAL (local de activación de salida), O_LOC_DSAL (local de desactivación de salida), O_REM_ASAL (remota de activación de salida) o O_REM_DSAL (remota de desactivación de salida).
- **Dirección**: En el caso de los modos u operaciones remotos, esta es la dirección del nodo remoto donde se encuentra la salida asignada a la entrada correspondiente.

Con este formato de configuración es posible la asignación de una entrada con varias salidas y viceversa. En caso de asignar una entrada con una salida más de una vez, la operación o modo que prevalece es que aparezca más tarde en la configuración.

De igual modo, si una salida estuviese asignada a más de una entrada y dichas entradas se activasen de manera simultanea, prevalecería aquella asignación que apareciese más cercana al final de la configuración.

Esto se debe al modo en que se trabaja con las tablas de asignación de entradas y salidas. El modo de hacerlo se verá en detalle en siguientes apartados.

4.5 Inicialización

La inicialización del sistema es llevada a cabo por el módulo principal. Dicho módulo hace uso de las funciones Inicialización, Comprobación de existencia de configuración, Lectura de configuración y Generación de configuración por defecto.

La inicialización de variables es llevaba a cabo por la función Inicialización, mientras que la lectura y procesado de la configuración es realizada por el resto de funciones.

4.5.1 Inicialización de variables

La función de inicialización de variables tiene como finalidad asignar un valor por defecto inicial a las variables que están relacionadas con las entradas y las salidas. Concretamente inicializa las variables ContPulsacion, OpCont y RemoteOp.

El resto de variables se inicializan en tiempo de ejecución o durante el proceso de carga de la configuración.

El código comentado de esta función y de todas las descritas en este apartado puede encontrarse en el Anexo I: Código fuente.

4.5.2 Lectura de configuración no volátil

La lectura, interpretación y, en caso necesario, generación de información relativa a la configuración del sistema es algo más compleja que en el caso de las variables relacionadas con las entradas y salidas.

Las funciones involucradas en este proceso son: Comprobación de existencia de configuración, Lectura de configuración y Generación de configuración por defecto.

A continuación puede verse un diagrama de flujo del procesamiento de la configuración:

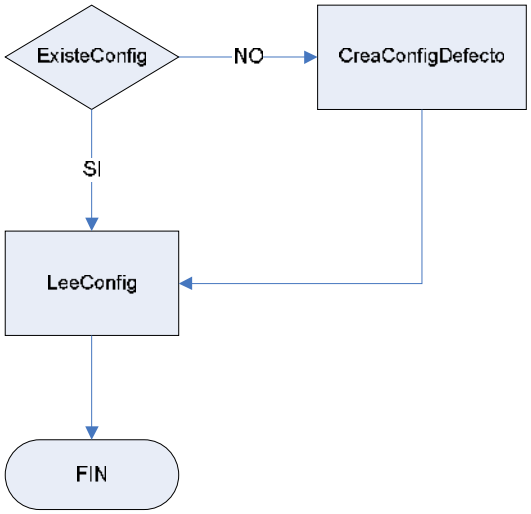

Diagrama 4.1 Diagrama de flujo del procesado de la configuración

4.5.2.1 ExisteConfig

Esta función hace uso de las librerías del compilador para el acceso a la EEPROM interna del microcontrolador, donde se encuentra alojada la configuración actual del nodo siguiendo el formato visto en el apartado 4.4. Para determinar si existe configuración válida alojada en dicha EEPROM, hace una lectura de la dirección cero, donde debe haber un valor diferente de 0x00 o de 0xFF y que coincide con la dirección propia del nodo.

Librerías usadas: read_eeprom(dir).

4.5.2.2 CreaConfigDefecto

Esta función usa las librerías de acceso la EEPROM interna para almacenar en ella la configuración por defecto que viene definida a través de una tabla constante creada en tiempo de programación.

Además, inicializa con el valor 0x00 (apagado) las posiciones de EEPROM donde se almacena el estado de cada una de las salidas del sistema. Esto permitirá al nodo recuperar su estado tras un corte de suministro eléctrico.

Librerías usadas: write_eeprom(dir, dato).

4.5.2.3 LeeConfig

Esta función es la encargada de leer la configuración guardada en la EEPROM interna e inicializar las tablas de asignación de entradas, salidas, modos, operaciones y direcciones a los valores adecuados.

Además, permite leer el estado almacenado de las salidas locales y procesar dicha información para que, posteriormente, las funciones correspondientes, establezcan dichas salidas al valor previamente guardado.

Este es su diagrama de flujo:

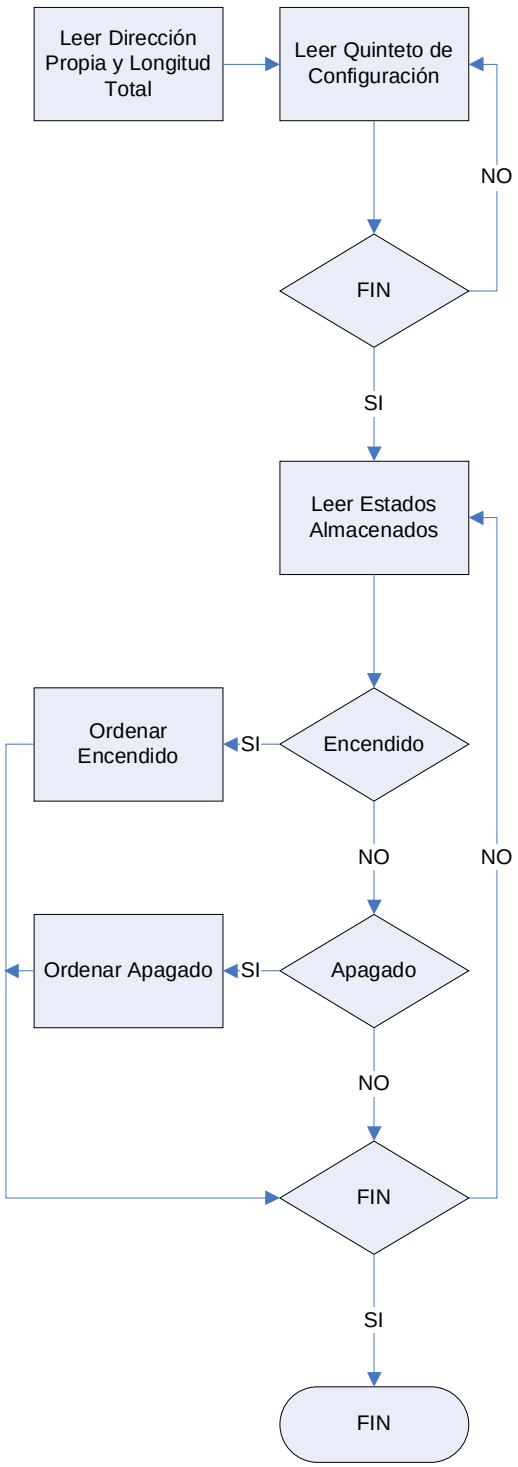

Diagrama 4.2 Diagrama de flujo de LeeConfig

Las variables actualizadas por esta función son: *DireccionPropia, Ent[], Sal[], Mod[], Ope[], Dir[], Estado[]* y *Operacion[]*.

Librerías usadas: read_eeprom(dir).

4.6 Lectura de entradas

Los tipos de entradas considerados han sido definidos en apartados anteriores. Para cada uno de estos tipos existe un procedimiento software diferente de lectura. Las entradas consideradas para su implementación son las entradas digitales simples.

De manera más genérica también se pueden considerar entradas al sistema las tramas recibidas a través del bus de datos.

4.6.1 Lectura de entradas digitales

La captura, interpretación y procesado de las entradas digitales se lleva a cabo a través de la tarea periódica **CapturaEntradas**.

Lo primero que esta tarea hace cada vez que se ejecuta es leer el valor digital de las entradas del sistema. Para ello captura los puertos A y C del microcontrolador, generando un byte (**EntradaActual**) que contiene en valor binario de cada una de las entradas consideradas (un bit por entrada).

A continuación incrementa los contadores de cada una de las entradas en función de si estas están activas. Para ello comprueba el estado de los bits del byte EntradaActual.

A partir de ese momento entran en juego las tablas de entradas, salidas, modos, operaciones y direcciones. Mediante el uso de un bucle, se recorren las tablas de entradas y salidas, leyendo cada vez una pareja de asignaciones. Para cada una de ellas se comprueba si tiene asignado un modo o una operación. Una vez determinado el modo u operación asignado a dicha entrada para dicha salida, se opera en función del estado de la entrada.

Actuando de esta forma, es posible detectar tanto las activaciones como las desactivaciones de las entradas, es decir, tanto los flancos de bajada como de subida de las señales digitales que las representan. Si sólo se considerasen las entradas activas, no sería posible medir la duración de las pulsaciones o detectar la desactivación de las entradas, al ignorar el hecho de que estas dejen de estar activas.

Finalmente, una vez que todos los pares entrada-salida han sido procesados, se actualiza el valor de los contadores de cada entrada dependiendo de los eventos que hayan tenido lugar. Esta operación se hace al final para no alterar el valor del contador entre dos ejecuciones del bucle en caso de que una entrada esté asignada a más de una salida.

Librerías usadas: rtos_yield(), bit_test(byte, bit).

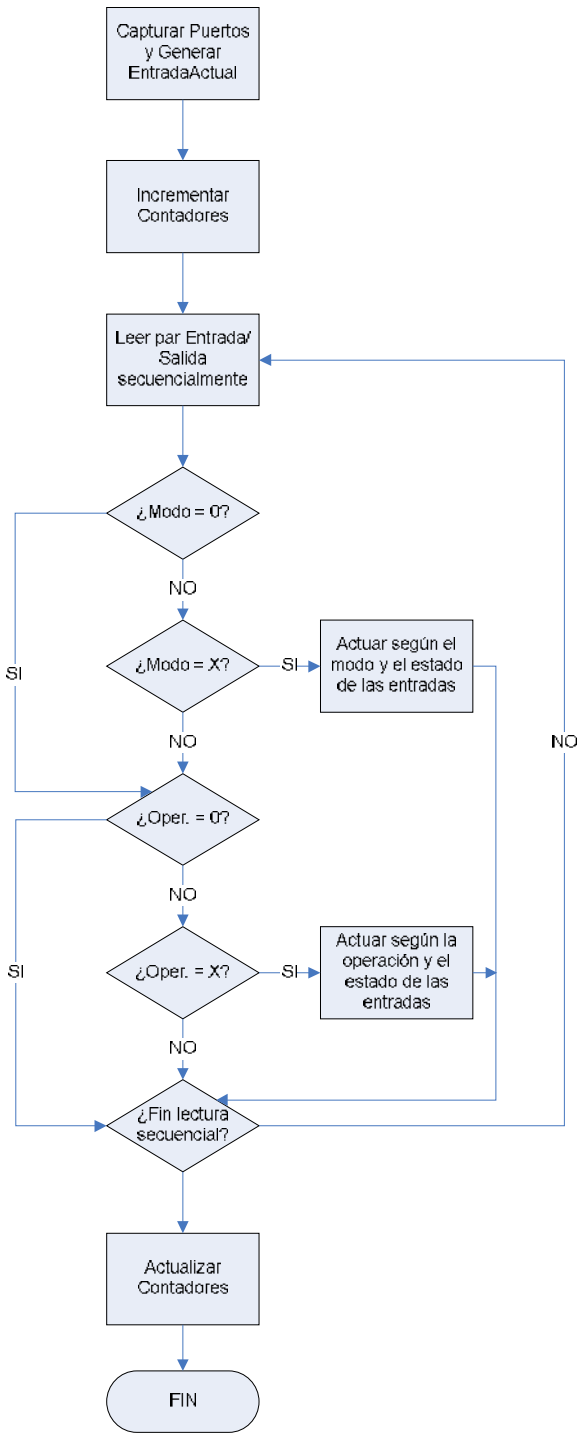

Diagrama 4.3 Diagrama de flujo de CapturaEntradas

4.6.2 Lectura de entradas no digitales

Las entradas no digitales consideradas son las tramas de datos provenientes de otros nodos del sistema domótico. Su captura y procesado se verán en detalle en el apartado 4.8.3.

4.7 Activación de salidas

El número de tipos de salidas es más limitado que el de entradas. En apartados anteriores se han descrito salidas digitales a través de relé, salidas reguladas mediante TRIAC y salidas a través de IR o de dispositivos externos.

En el presente proyecto se han considerados las salidas digitales a través de relé (denominadas en este apartado salidas binarias) y las salidas reguladas mediante TRIAC (denominadas salidas reguladas).

Al igual que en el caso de las entradas, de manera más genérica pueden considerarse salidas del sistema a las tramas de datos transmitidas a través del bus de datos.

4.7.1 Activación de salidas binarias

La activación y desactivación de las salidas binarias se lleva a cabo mediante la tarea periódica **ActivaSalidas**. Esta tarea es la encargada de llevar a cabo las operaciones determinadas por la tarea de captura de entradas (descrita en el apartado anterior).

Es importante destacar que las salidas reguladas en intensidad también se pueden controlar de manera binaria (ON/OFF) si consideramos el estado apagado como aquel con intensidad nula y el estado encendido aquel con una intensidad diferente de cero. Para poder controlar este tipo de salidas de manera análoga al resto, la tarea ActivaSalidas las trata de manera similar a las salidas puramente binarias.

La diferencia reside por tanto en el mecanismo usado (de manera externa a la tarea ActivaSalidas) para la regulación en intensidad de dicha salida una vez que esta está activa. Este mecanismo se explicará en el siguiente apartado.

Lo primero que hace la tarea de activación de salidas cada vez que se ejecuta es comprobar si se debe llevar a cabo alguna operación remota, es decir, alguna operación que requiera de la transmisión de tramas a algún nodo remoto. En caso de ser así, se recoge la información contenida en las variables que usa CapturaEntradas para almacenar la información de la comunicación que debe establecerse y la usa para actualizar la tabla de direcciones con la dirección de la trama destino, y el NT y NR adecuados.

De este modo, en caso de ser la primera vez que se transmite una trama a dicho nodo remoto, la tabla de direcciones ya contiene la información del NT enviado y el NR esperado, permitiendo el correcto funcionamiento del mecanismo de asentimiento en el resto de transmisiones que tengan lugar entre ambos nodos.

A continuación se recorre la tabla *Operacion[]* y se lleva a cabo la operación asignada a cada salida del nodo local. Cada una de las filas de dicha tabla se corresponde con una salida local o remota.

Debido al formato de la configuración descrito en apartados anteriores, si en la tabla de asignaciones de entradas y salidas existiesen dos asignaciones a una misma salida una local y otra remota, en caso de darse simultáneamente el evento que las active, prevalecería la que apareciera más tarde en la tabla *Sal[]*.

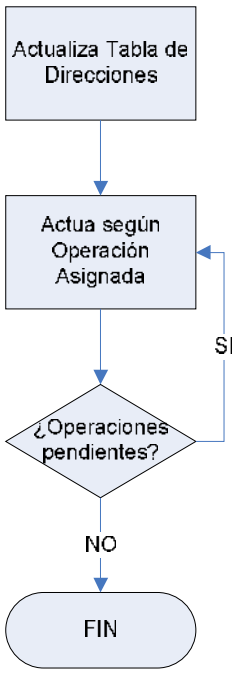

Diagrama 4.4 Diagrama de flujo de ActivaSalidas

4.7.2 Activación de salidas reguladas

La regulación de salidas en intensidad se lleva a cabo mediante el uso de TRIAC's. De forma coloquial podría decirse que un TRIAC es un interruptor capaz de conmutar la corriente alterna. Su estructura interna se asemeja en cierto modo a la disposición que formarían dos tiristores en antiparalelo.

Su versatilidad lo hace ideal para el control de corrientes alternas y muchas otras aplicaciones. Una de ellas es su utilización como interruptor estático ofreciendo muchas ventajas sobre los interruptores mecánicos convencionales y los relés.

En la práctica, un TRIAC en un interruptor capaz de conmutar de manera mucho más rápida a como lo haría un relé. Por este motivo se usa para "recortar" la señal eléctrica de la red de alimentación para la regulación en intensidad (o en potencia) de elementos de iluminación (generalmente).

Para llevar a cabo esta labor, no solo es necesario el uso de un TRIAC. Igual de importante es el elemento que lo controla. En este caso, el control del TRIAC lo hace de manera software el microcontrolador de cada nodo remoto del sistema domótico.

Para el control de los TRIAC's se han definido e implementado las interrupciones externa y de desbordamiento de temporizadores y el resto de funciones que gestionan el estado de ambos TRIAC's.

Como ya se ha dicho, el control de potencia o de intensidad, se basa en el "recortado" de la señal eléctrica de la red de alimentación. Cuando no se realiza regulación y se deja pasar el 100% de la potencia de la señal eléctrica, la forma de onda que llega a la salida es una senoide completa con lóbulos positivos y negativos (Figura 4.5).

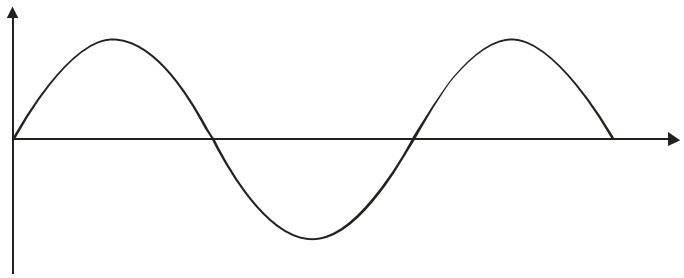

Figura 4.5 Señal eléctrica completa

La regulación se hace mediante la sincronización con los pasos por cero de la señal eléctrica. En la Figura 4.6 se puede ver una señal regulada al 50%. Como se puede observar, de cada lóbulo de la forma de onda senoidal se deja pasar la primera mitad. La regulación es igualmente válida si se deja pasar la segunda mitad de cada lóbulo (método usado en el presente proyecto).

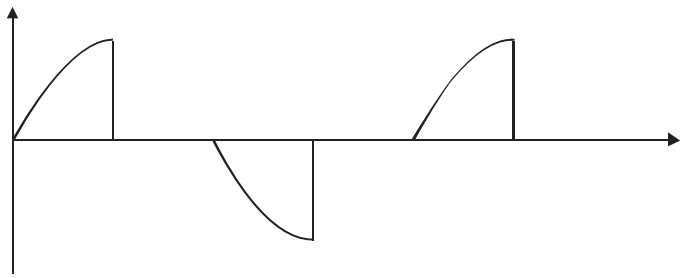

Figura 4.6 Señal eléctrica al 50 %

Se han definido 210 niveles de potencia posibles para las salidas reguladas. De todos estos niveles son accesibles 190, puesto que se ha definido un margen de seguridad de 10 niveles en torno a los extremos del rango posible. El objetivo de estos

márgenes es evitar los efectos indeseados que tienen lugar cuando los TRIAC's son conmutados en las proximidades de los pasos por cero de la señal eléctrica de la red.

Sabiendo que cada lóbulo de la onda senoidal dura 10 milisegundos, se puede sincronizar el inicio de cuenta del temporizador con el paso por cero de la señal de la red, de modo que pasado un cierto tiempo se apague el TRIAC. De este modo, el TRIAC se enciende en el paso por cero y se apaga X milisegundos después. En el caso del presente proyecto el procedimiento es inverso. El TRIAC se apaga en el paso por cero y se enciende al cabo de X milisegundos.

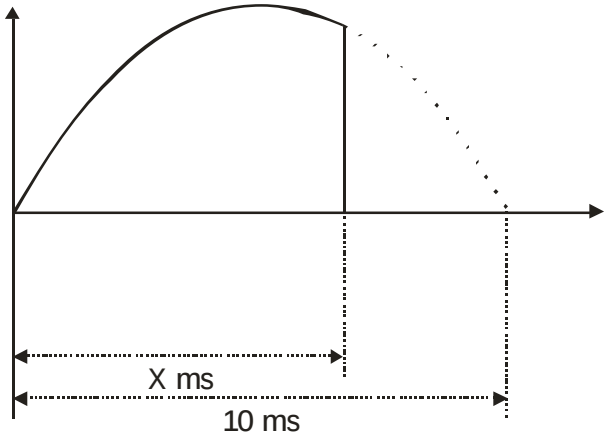

Figura 4.7 Recortado de la señal eléctrica

Los temporizadores usados para tal fin son el timer 1 y el timer 3, ambos de 16 bits de longitud. Con este dato y sabiendo que la frecuencia de incremento interno de los contadores es de una cuarta parte de la frecuencia de reloj, es posible calcular el valor que debe escribirse en el registro interno de los temporizadores para sincronizar el desbordamiento con el momento en que se desea activar (o desactivar) el TRIAC.

Para ello es usa la siguiente instrucción:

```
set_timerX(OVERFLOW-(_mul(PotDimmer[i],DIMMER_SCALE)));
```

OVERFLOW es el valor de desbordamiento de los temporizadores. DIMMER_SCALE es el factor por el que se debe multiplicar al valor de potencia deseado para obtener el valor que debe restarse a OVERFLOW para esperar un tiempo determinado.

De este modo, si el valor de potencia deseado es un número elevado (próximo a 200), se le resta a OVERFLOW un valor grande, por lo que el TRIAC tarda más en encenderse y la bombilla brilla con menos intensidad.

El código correspondiente a las rutinas de interrupción externa y de desbordamiento puede consultarse en el Anexo de la presente memoria.

De manera menos directa con este mecanismo se encuentra relacionada la función de cambio de potencia de dimmer, que se encarga de cambiar de manera cíclica el valor de potencia de los dimmers.

Librerías usadas: rtos_yield(), write_eeprom(dir, dato), bit_test(byte, bit), bit_clear(byte, bit), output_high(pin), output_low(pin), enable_interrupts(int), disable_interrupts(int), clear_interrupt(int), set_timer1(valor), set_timer3(valor), _mul(op1, op2).

4.7.3 Activación de salidas no binarias

Las salidas no binarias consideradas son las tramas de datos enviadas a otros nodos del sistema domótico. Su generación y envío se verán en detalle en el apartado 4.8.2.

4.8 Transmisión por puerto serie

Cuando hablamos de transmisión por puerto serie nos referimos a los datos que se transmiten al transceiver de RS485 a través de la UART interna del microcontrolador para que este los transmita a su vez a través del bus de datos.

De manera genérica se han considerado a las tramas recibidas como entradas al sistema y a las tramas enviadas como salidas. Los elementos de software que entran en juego en un caso o en otro son diferentes, pudiendo distinguir para el caso de las tramas enviadas las funciones de generación de tramas y de envío y para el caso de las tramas recibidas las de recepción de bytes, recogida de tramas e interpretación de tramas.

4.8.1 Nivel físico: Parámetros no fijados por RS-485

Existen muchos parámetros de las transmisiones de datos que no están determinados por el estándar RS485. Estos parámetros no sólo se refieren a aspectos de niveles lógicos por encima del nivel físico. Son muchos los parámetros del nivel físico que tampoco están definidos en el estándar mencionado.

Algunos de estos parámetros son estos:

- Velocidad de transmisión: En el presente proyecto se transmite a una velocidad de 19200 bits por segundo. Esta velocidad es suficiente para el volumen de tráfico generado en una aplicación domótico convencional.
- Bits de datos: Se usan 8 bits de datos por cada transmisión. Es decir, se transmiten bytes.
- Paridad: No se usa bit de paridad. Como mecanismo de detección de errores se usa un código checksum para cada trama.
- Bits de stop: No se definen bits de stop. Este aspecto de las transmisiones depende de las características del tranceiver usado.

4.8.2 Transmisión de datos

El proceso completo por el cual se transmiten datos al exterior puede variar según el caso. Lo más habitual es que se trate de la transmisión de un comando remoto como respuesta a un estímulo externo o interno que, mediante la configuración del nodo, esté asociado a una salida remota, es decir, a una salida que pertenece a otro nodo de la red.

Otra posibilidad es la transmisión de tramas de asentimiento, que no requieren como condición previa la ocurrencia de ningún evento en las entradas locales del sistema. En este caso el estímulo sería de carácter remoto, el deberse esta transmisión a la recepción previa de una trama que requiere asentimiento.

De igual modo, en el caso de las tramas de datos de configuración, el estímulo será un comando de alto nivel generado por el usuario del sistema.

El caso más habitual es aquel en el que las tramas se generan como respuesta a un evento externo local. En este caso, el proceso de generación y envío de tramas se inicia en la tarea periódica de activación de salidas, la cual, en caso de tener que procesar una salida remota, hace una llamada a la función de creación de tramas.

En paralelo con este proceso, la tarea de envío de tramas comprueba constantemente (con la frecuencia indicada por el diseñador mediante la directiva de preprocesado #task) la existencia de tramas en el búfer de salida del nodo, enviándolas en caso de existir y de estar en condiciones de hacerlo.

La función de creación de tramas (CreaTrama) recibe como parámetros los campos de la trama necesarios, es decir, la dirección de destino, el campo de control, la longitud y un puntero a los datos. Con esta información se introduce una nueva trama en el búfer de salida del nodo, añadiendo a todo ello la bandera inicial, la dirección origen y el código checksum que se calcula conforme se introducen los datos en dicho búfer.

La tarea de envío de tramas se ejecuta cada 100 milisegundos y no sólo se encarga de enviar las tramas que haya en el búfer de salida. También se encarga de indicar el estado de la ventana de transmisión en función del tipo de trama enviado.

Esta tarea no transmite ninguna trama mientras el búfer de salida esté vacío o la ventana de transmisión esté llena, a menos que se trate de una retransmisión. Cuando estas condiciones se cumplen, se transmite la trama que haya en el búfer (en caso de tratarse de una retransmisión, la trama a retransmitir habrá sido previamente copiada al búfer de salida para simplificar la labor de esta tarea) y se activan los mecanismos de expiración de asentimiento si procede.

El Diagrama 4.8 se corresponde con el diagrama de flujo de esta tarea. La espera se realiza devolviendo el control al despachador de tareas cada vez que esta tarea sea invocada y no se cumplan las condiciones de operación.

Librerías usadas: rtos_yield(), _mul(op1, op2), fputc(byte).

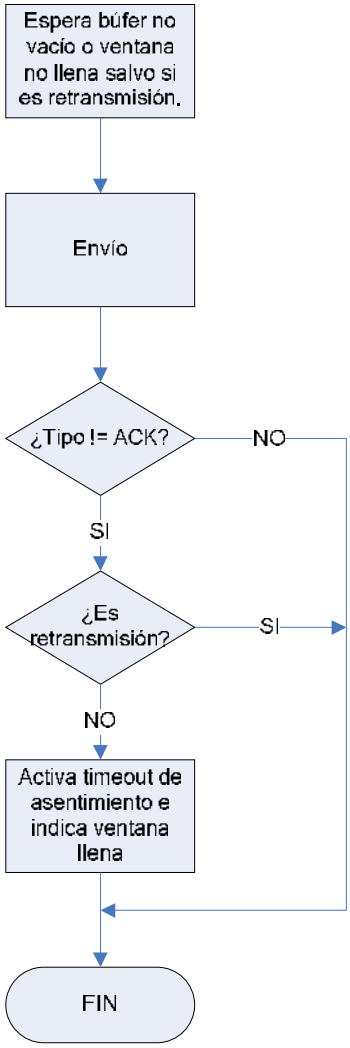

Diagrama 4.8 Diagrama de flujo de EnvíaTramas

4.8.3 Recepción de datos

En la recepción e interpretación de tramas intervienen más elementos de software que en la transmisión. Concretamente, para la recepción de tramas entran en juego la rutina de interrupción de byte recibido y la tarea de recogida de tramas. Por otro lado, su interpretación es llevada a cabo por la tarea periódica de interpretación de tramas.

La interrupción de byte recibido se ejecuta cada vez que se recibe un byte de datos en el búfer hardware del microcontrolador. El nodo por tanto, recibirá las tramas byte a byte, debiendo saber en todo momento qué byte se espera recibir según el formato de las tramas y los bytes recibidos previamente.

Para realizar esta labor se ha definido una variable que almacena es estado del nodo en relación a las recepciones de datos: **EstadoBus**. Los posibles valores de esta variable son los siguientes:

- **BUS_OCIOSO**: Es el estado habitual del nodo, el estado de reposo. Cuando el nodo se encuentre en este estado, se esperará recibir la bandera inicial de una trama. Una vez que eso ocurra, el estado pasará a ser BUS_ESP_DEST y se iniciará la temporización de expiración de recepción.
- **BUS_ESP_DEST**: Una vez recibida la bandera inicial, el nodo esperará recibir la dirección destino de la trama. Si esta dirección es la propia o si se trata de una trama de difusión, el estado pasa a ser el de espera de dirección origen. Es caso negativo, se captura aún así la dirección origen para actualizar la tabla de direcciones con las direcciones de los nodos que se vayan capturando. Con este byte comienza el cálculo de código checksum.
- **BUS_ESP_ORIG**: La dirección de origen es el primer byte que se almacena en el prebúfer de entrada del nodo. A continuación es pasa a esperar ambos bytes del campo control.
- **BUS_ESP_CTRL1**: Cuando el nodo se encuentra en este estado, el byte recibido será interpretado como el primer byte del campo de control y se pasará a esperar el segundo.
- **BUS_ESP_CTRL2**: De manera análoga al caso anterior, se recibe el segundo byte y se pasa a esperar la longitud de los datos de la trama.
- **BUS_ESP_LONG**: Cuando el estado es este, el dato recibido es la longitud del campo de información. Si este valor es nulo, se pasa directamente a esperar el código checksum de la trama, que será comparado con el calculado. Si la longitud no es nula, se usa la propia variable EstadoBus para conocer la cantidad de bytes de datos recibidos hasta el momento. Cuando se reciban todos, se pasa al estado siguiente.
- **BUS_ESP_CKSM**: El código checksum recibido debe coincidir con el calculado durante la recepción de los bytes de la trama. Sólo en ese caso se notificará a la tarea de recogida de tramas de la existencia de una trama válida lista para ser movida al búfer de entrada del nodo. Igualmente, en este punto se desactiva la temporización de expiración de recepción.

La rutina de interrupción de byte recibido no es más que la implementación de la secuencia de estados antes citados.

Del mismo modo, la tarea de recogida de tramas comprueba periódicamente la existencia de alguna trama lista para ser transferida al búfer de entrada. Esto lo hace

la rutina de interrupción a través del flag **TramaRecibida**, el cual activa tras la recepción de un código checksum correcto.

Una vez que la trama se ha transferido al búfer de entrada, la tarea **InterpretaTramas** es la encargada de procesar los datos de dicha trama así como de llevar a cabo el mecanismo de asentimiento. Esta tarea, al igual que muchas otras, espera a que haya alguna trama en el búfer para operar (en caso de no haber ninguna, devuelve el control al RTOS).

El diagrama de flujo de esta tarea se encuentra representado en el Diagrama 4.9.

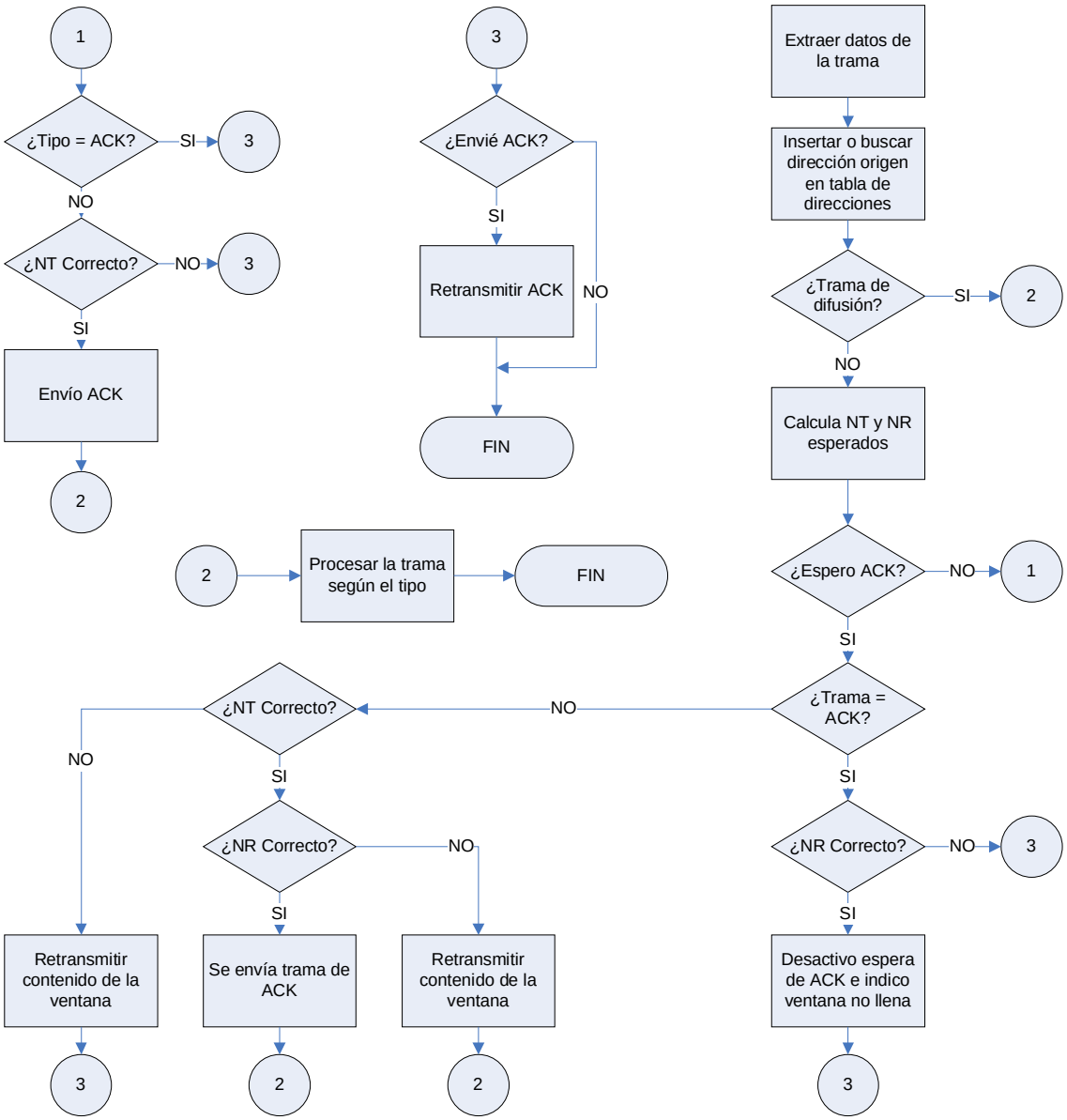

Diagrama 4.9 Diagrama de flujo de InterpretaTramas.