eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea

VITORIA-GASTEIZKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE VITORIA-GASTEIZ

GRADO EN INGENIERÍA
ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

TRABAJO DE FIN DE GRADO

DESARROLLO DE UN CLIENTE MQTT PARA COMUNICACIONES SEGURAS EN SISTEMAS EMBEBIDOS

Alumno : Jorge Botana Mtz. de Ibarreta
Director: José Miguel Gil-García Leiva

Curso: 2021 - 2022

Fecha: 24 de noviembre de 2021

LICENCIAS

RESUMEN

En el presente documento se expone el desarrollo de un cliente MQTT que permite realizar comunicaciones seguras mediante TLS, para ser usado en una aplicación que se ejecute sobre un sistema embebido.

La aplicación creada también hace uso de otros protocolos TCP/IP, motivo por el cual se han desarrollado clientes adicionales que también son expuestos en este documento.

El documento consta de dos partes. La primera parte es la memoria, y en ella se exponen aquellos aspectos teóricos y prácticos de los que es recomendable tener conocimiento para hacer uso del proyecto de programación creado, escrito en lenguaje C y que se encuentra en la segunda parte, de anexos.

Palabras clave

MQTT; TLS; PKI; STM32

CONTENIDO

Pág.

FIGURAS

Pág.

TABLAS

Pág.

NOMENCLATURA

AES - Advanced Encryption Standard
ANSI - American National Standards Institute
API - Application Programming Interface
ARM - Advanced RISC Machines
ASCII - American Standard Code for Information Interchange
ASN.1 - Abstract Syntax Notation 1
BSD - Berkeley Software Distribution
BSP - Board Support Package
C - Country
CA - Certification Authority
CBC - Cipher Block Chaining
CDT - C/C++ Development Tooling
CFG - ConFiGuration
CM7 - Cortex M7
CMSIS - Cortex Microcontroller Software Interface Standard
CN - Common Name
CR - Carriage Return
CRL - Certificate Revocation List
CSR - Certificate Signing Request
DER - Distinguished Encoding Rules
DES - Data Encryption Standard
DH - Diffie-Hellman
DHCP - Dynamic Host Configuration Protocol
DHE - Diffie-Hellman Ephemeral
DNS - Domain Name System
DSA - Digital Signature Algorithm
DTLS - Datagram Transport Layer Security
ECC - Elliptic Curve Cryptography
ECDH - Elliptic Curve Diffie-Hellman
ECDHE - Elliptic Curve Diffie-Hellman Ephemeral

ECDSA - Elliptic Curve Digital Signature Algorithm
ETH - ETHernet
EV - Extended Validation
FCS - Frame Check Sequence
FTP - File Transfer Protocol
GCC - GNU Compiler Collection
GCM - Galois/Counter Mode
GMT - Greenwich Mean Time
GPIO - General Purpose Input/Output
HAL - Hardware Abstraction Layer
HTTP - HyperText Transfer Protocol
HW - HardWare
ID - IDentifier
IDE - Integrated Development Environment
IETF - Internet Engineering Task Force
INC - INClude
IOT - Internet Of Things
IP - Internet Protocol
IRQ - Interrupt ReQuest
ISP - Internet Service Provider
IV - Initialization Vector
LAN - Local Area Network
LCD - Liquid Crystal Display
LED - Light Emitting Diode
LF - Line Feed
LTS - Long Time Support
LWIP - LightWeight IP
MAC (1) - Media Access Control
MAC (2) - Message Authentication Code
MD5 - Message-Digest algorithm 5
MISC - MISCellaneous
MIT - Massachusetts Institute of Technology
MITM - Man In The Middle
MPU - Memory Protection Unit
MQTT - Message Queuing Telemetry Transport
NAT - Network Address Translation
NTP - Network Time Protocol
O - Organization
OASIS - Organization for the Advancement of Structured Information Standards

PC - Personal Computer
PEM - Privacy Enhanced Mail
PHY - PHYsical Layer
PKI - Public Key Infrastructure
PRF - Pseudo-Random Function
PSK - Pre-Shared Key
QOS - Quality Of Service
RC4 - Rivest Cipher 4
RCC - Reset and Clock Controller
RFC - Request for Comments
RISC - Reduced Instruction Set Computing
RMII - Reduced Media-Independent Interface
RNG - Random Number Generator
RSA - Rivest-Shamir-Adleman
RTC - Real Time Clock
RTOS - Real Time Operative System
SANS - Simple Authoritative Name Server
SHA - Secure Hash Algorithm
SNTP - Simple Network Time Protocol
SRC - SouRCe
SSL - Secure Sockets Layer
STM32 - STMicroelectronics 32 bits
SW - SoftWare
SWD - Serial Wire Debug
TBS - To Be Signed
TCP - Transport Control Protocol
TDEA - Triple Data Encryption Algorithm
TFTPD64 - Trivial File Transfer Protocol Daemon 64 bits
TIM - TIMer
TLS - Transport Layer Security
UART - Universal Asynchronous Receiver/Transmitter
UDP - User Datagram Protocol
USB - Universal Serial Bus
UTF-8 - Unicode Transformation Format - 8 bits
VCOM - Virtual COMmunication port

Las siglas y abreviaturas anteriores son aquellas que se han utilizado a lo largo de este documento, sin tener en cuenta el contenido de las zonas con fondo gris (comandos, logs y código fuente) e imágenes no vectorizadas (fotos o capturas de pantalla).

MEMORIA

1

Introducción

1.1. Contexto

MQTT es un protocolo de mensajería basado en un modelo de publicador/suscriptor y que cuenta con las siguientes ventajas:

- Es ligero y sencillo, al mismo tiempo que dispone de múltiples opciones, varios modos de funcionamiento y características de seguridad.

- Se considera apto para ser usado en redes no fiables, con poco ancho de banda o con una alta latencia.

- Consume poca energía. Esto se debe al reducido tamaño de sus paquetes, a la minimización del número de ellos intercambiados y al hecho de que se permita dejar conexiones abiertas, evitando esfuerzos de reconexión TCP y TLS.

Estas características han favorecido que el protocolo MQTT haya experimentado una creciente popularidad en el ámbito del IoT, un concepto que hace referencia a la interconexión a través de Internet de todo tipo de objetos, donde un gran número de ellos corresponde a sistemas embebidos que a menudo disponen de bajas prestaciones de hardware (ya que usan microcontroladores), conexión a Internet poco fiable (a través de redes inalámbricas) y/o energía limitada (mediante baterías).

Las líneas anteriores son una breve introducción al tema central que ocupa este documento. Para conocer más acerca del funcionamiento del protocolo MQTT, se puede consultar la sección de este documento dedicada a ello.

1.2. Objetivos

El auge del protocolo MQTT ha motivado al autor a llevar a cabo el trabajo recogido en este documento, que tiene los siguientes objetivos:

- Desarrollar un cliente MQTT.
- Añadir TLS con autenticación del servidor y del cliente.
- Crear una aplicación para un sistema embebido.
- Comparar los tiempos de lectura y escritura de paquetes MQTT con y sin TLS.

El cumplimiento de estos objetivos queda demostrado con el proyecto de programación creado, que es configurable en tiempo de compilación y fácil de portar entre distintas plataformas. Su código fuente está escrito en lenguaje C y se encuentra en los anexos.

1.3. Tareas

Para alcanzar los objetivos anteriores se han llevado a cabo las siguientes tareas:

- Crear redes de comunicaciones con varios dispositivos: un sistema embebido, un PC, un router y equipos externos.

- Desarrollar clientes para los protocolos de capa de aplicación DHCP, DNS, NTP, TLS y MQTT, usando en los cuatro últimos la API de sockets BSD en la capa de transporte UDP/TCP para que sean portables entre distintos stacks de TCP/IP.

- Configurar en red local servidores para los protocolos anteriores.

- Capturar y analizar el tráfico de red, como parte del proceso de desarrollo de los clientes y de la aplicación para el sistema embebido que hace uso de ellos, y para comprobar el correcto funcionamiento de los servidores configurados en red local, situados en el PC y en el router.

- Construir infraestructuras de certificados usando herramientas destinadas a ello.

- Programar, configurar, ejecutar y depurar la aplicación, contando esta con un modo de funcionamiento normal (como ejemplo de uso de los clientes desarrollados) y otro de pruebas (para medir los tiempos de procesamiento con y sin TLS), permitiendo la interacción con ella usando un segundo cliente MQTT desde PC.

A lo largo de este documento se puede ver cómo se han llevado a cabo estas tareas.

2

Comunicaciones TCP/IP

2.1. Redes de dispositivos

Para poner en funcionamiento la aplicación creada, que es expuesta más adelante, se han dispuesto de las siguientes redes de comunicaciones:

- **Intranet**: Red interna de la que forma parte el primer dispositivo de la siguiente lista y al menos uno más:
  - **NUCLEO-H723ZG**: Placa de desarrollo que corresponde al sistema embebido. Contiene la aplicación, que hace uso de los clientes desarrollados por el autor de este documento.
  - **JORGE-PC**: Hostname del PC usado, que cuenta con servidores para todos los clientes usados por la aplicación.
  - **Router**: Dispone de servidores para algunos de los protocolos usados.
- **Internet**: Red externa que, en caso de hacer uso de ella, ha de contar con los siguientes dispositivos:
  - **Equipos externos**: Cuentan con servidores para poder usar la aplicación si no se utiliza un PC.
  - **Router**: Implementa la NAT necesaria para poder conectar la intranet con Internet.

En el siguiente diagrama se muestran las combinaciones de redes de dispositivos que se han creado.

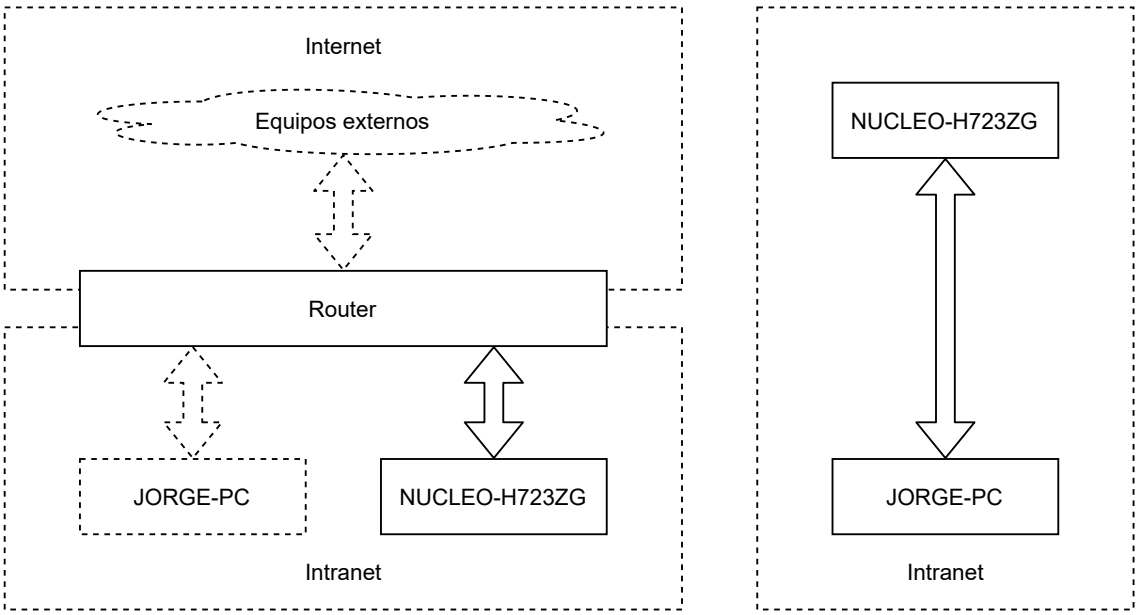

Figura I – Redes de dispositivos

Del diagrama anterior se puede deducir que:

- No hace falta conexión a Internet para poner en funcionamiento la aplicación, pero en ese caso la intranet ha de contar con todos los servidores necesarios.

- Si se dispone de conexión a Internet y todos los servidores externos empleados se encuentran disponibles, se puede prescindir completamente del PC.

Ha de tenerse en cuenta que, para poder hacer uso de los clientes y servidores, es necesario que estén configurados correctamente los cortafuegos del PC y del router para permitir todas las conexiones entrantes y salientes por los puertos utilizados por los distintos protocolos usados.

En la siguiente sección se expone una introducción teórica sobre los protocolos de capa de aplicación utilizados junto a instrucciones de uso de los clientes y servidores empleados. No es el caso de los clientes desarrollados por el autor de este documento, ya que las instrucciones de uso de estos se encuentran entre los comentarios del propio código fuente, situado en los anexos.

2.2. Clientes y servidores

2.2.1. DHCP

DHCP es un protocolo cuya función es obtener (cliente) o proveer (servidor) una **configuración de red** (dirección IP, máscara de subred y puerta de enlace). Pertenece a la capa de aplicación y opera sobre la capa de transporte **UDP** en los puertos **68** (cliente) y **67** (servidor).

A continuación se tienen todos los mensajes que se usan en el protocolo DHCP, de los cuales al menos se intercambian los cuatro primeros durante el proceso en el cual el servidor le proporciona una configuración de red al cliente:

- **DHCPDISCOVER**: El cliente lanza una difusión en busca de un servidor que le proporcione una configuración de red.
- **DHCPOFFER**: Un servidor ofrece al cliente una configuración de red que podrá usar durante un tiempo establecido, tras el cual tendrá que renovarla.
- **DHCPREQUEST**: El cliente solicita o renueva la configuración de red ofrecida por el servidor.
- **DHCPACK**: El servidor acepta una solicitud del cliente.
- **DHCPNACK**: El servidor rechaza una solicitud del cliente.
- **DHCPDECLINE**: El cliente informa al servidor de que la dirección IP asignada ya estaba en uso.

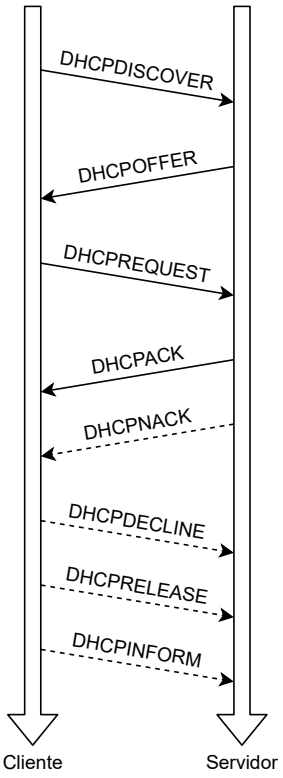

Figura II – Proceso DHCP

- **DHCPRELEASE**: El cliente informa al servidor de que libera la dirección IP asignada.
- **DHCPINFORM**: El cliente solicita al servidor parámetros de configuración de la red.

La especificación del protocolo se encuentra en el documento **IETF RFC 2131**, al que le complementan otros estándares con extensiones.

Se han hecho uso de las siguientes implementaciones del protocolo DHCP en las redes de dispositivos empleadas:

- **Clientes**: Se desconoce qué mecanismos siguen el router y los equipos externos para obtener sus direcciones IP públicas en Internet, mientras que dentro de la intranet se cuenta con un cliente DHCP para el sistema embebido y otro para el PC, que son los siguientes:

  - **DHCP_CLIENT**: Corresponde al cliente DHCP desarrollado por el autor de este documento, y permite establecer una configuración de red estática tras agotar un número de intentos establecido. Su código fuente se encuentra en los anexos y requiere de un stack de TCP/IP llamado **lwIP**. Es usado por la aplicación del sistema embebido.

  - **Servicio del OS del PC**: Se trata de un cliente DHCP que se ejecuta automáticamente al iniciar el sistema operativo del PC. En algunas ocasiones se ha establecido una configuración de red estática sin hacer uso de este cliente DHCP.

- **Servidores**: Dependiendo de la red de dispositivos empleada, el sistema embebido y el PC pueden obtener la configuración de red de un servidor u otro:

  - **Servicio del OS del router**: Corresponde al servidor DHCP del router que provee una configuración de red a dispositivos de la intranet que son conectados a él. Además, permite establecer en la configuración del router una dirección IP estática para la dirección MAC de un dispositivo.

  - **TFTPD64**: Se trata de una utilidad gratuita y de código abierto para PC que implementa servidores de varios protocolos, entre los que se incluye DHCP. De esta forma, si se conecta un dispositivo al PC, podrá obtenerse de él su configuración de red.

    A continuación se muestra un ejemplo de configuración de este servidor DHCP para poder proporcionar hasta 80 direcciones IP a partir de la `192.168.1.100`, y todas ellas con formato `192.168.1.XXX`, ya que la máscara de subred se establece como `255.255.255.0`. Se fija `192.168.1.1` como puerta de enlace y las direcciones IP han de ser renovadas tras 2880 minutos (48 horas).

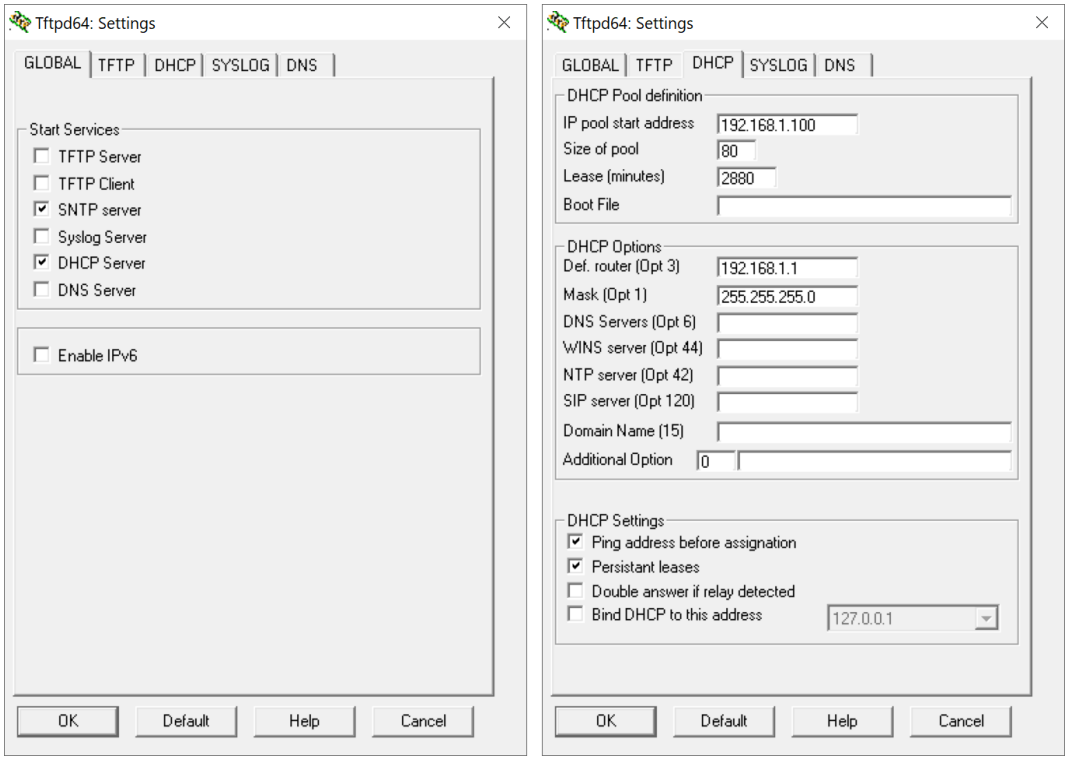

Figura III – Servidor DHCP de TFTPD64 (1)

Tras realizar esta configuración, en este caso el sistema embebido obtiene la dirección IP `192.168.1.101`, ya que el PC cuenta con la dirección IP estática `192.168.1.100`.

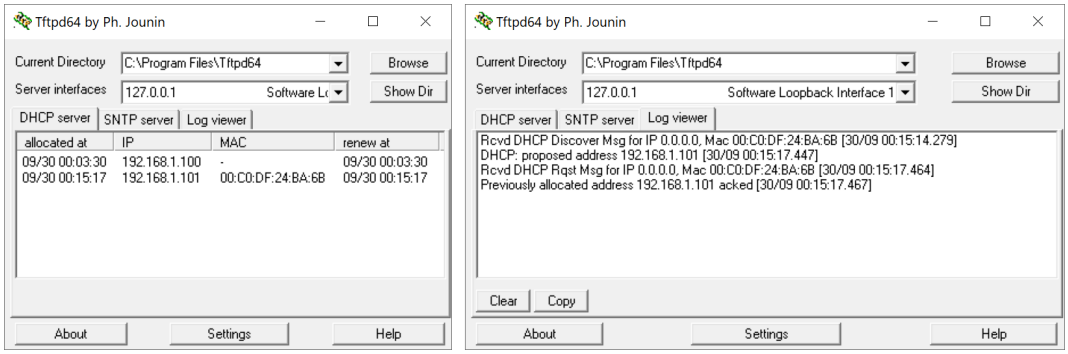

Figura IV – Servidor DHCP de TFTPD64 (2)

En las imágenes anteriores se puede observar que en TFTPD64 también se ha habilitado el servidor SNTP, un protocolo compatible con NTP, y la razón de ello es que también se ha hecho uso de estos protocolos, tal y como se expone más adelante.

2.2.2. DNS

DNS es un protocolo cuya función es **resolver hostnames**. Pertenece a la capa de aplicación y opera sobre la capa de transporte **UDP** (normalmente) en el puerto **53** (servidor).

El cliente resuelve un hostname intercambiando únicamente dos mensajes con el servidor:

- **DNS request**: El cliente envía una consulta al servidor con el hostname que quiere resolver.
- **DNS response**: El servidor responde al cliente con una o varias direcciones IP correspondientes al hostname consultado.

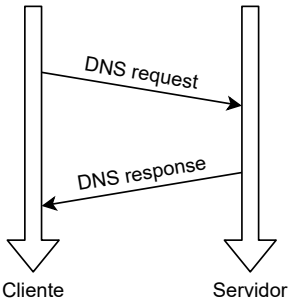

Figura V – Resolución DNS

La especificación del protocolo se encuentra en los documentos **IETF RFC 1034** e **IETF RFC 1035**, a los que complementan otros estándares con extensiones.

Se han hecho uso de las siguientes implementaciones del protocolo DNS en las redes de dispositivos empleadas:

- **Clientes**: Se cuenta con un cliente para el sistema embebido y otro para el PC, que son los siguientes:
  - **DNS_CLIENT**: Corresponde al cliente DNS desarrollado por el autor de este documento. Su código fuente se encuentra en los anexos y requiere de **UDP_CLIENT**, también desarrollado por el autor de este documento y que hace uso de la API de sockets BSD, pero sin hacer uso de funciones DNS que esta incluye como `gethostbyname()` o `getaddrinfo()`. Tampoco se utilizan librerías DNS de terceros. Es usado por la aplicación del sistema embebido.
  - **Eclipse Mosquitto**: Se trata de un broker/cliente MQTT que al mismo tiempo es un cliente DNS, ya que resuelve los hostnames de los sitios a los que conecta llamando a las rutinas DNS de la API de sockets BSD. Es usado por el PC.
- **Servidores**: Los clientes anteriores han conectado a los siguientes servidores:
  - **Google Public DNS**: Corresponde a dos servidores DNS de Internet con direcciones IP `8.8.8.8` y `8.8.4.4`. No pueden resolver hostnames de la intranet.

- **Cloudfare DNS**: Se trata de un servidor DNS de Internet con dirección IP `1.1.1.1`. No puede resolver hostnames de la intranet.

- **Servicio del OS del router**: Corresponde al servidor DNS del router que puede hacer de proxy redirigiendo consultas a servidores externos normalmente preestablecidos por el ISP, a menos que se consulte el hostname de un equipo de la intranet, en cuyo caso podría contestar con su dirección IP local.

- **SANS**: Se trata de un servidor DNS gratuito para PC que puede resolver aquellos hostnames que se registren en él. A continuación se muestra un ejemplo de configuración y funcionamiento en el que resuelven dos de los tres hostnames registrados.

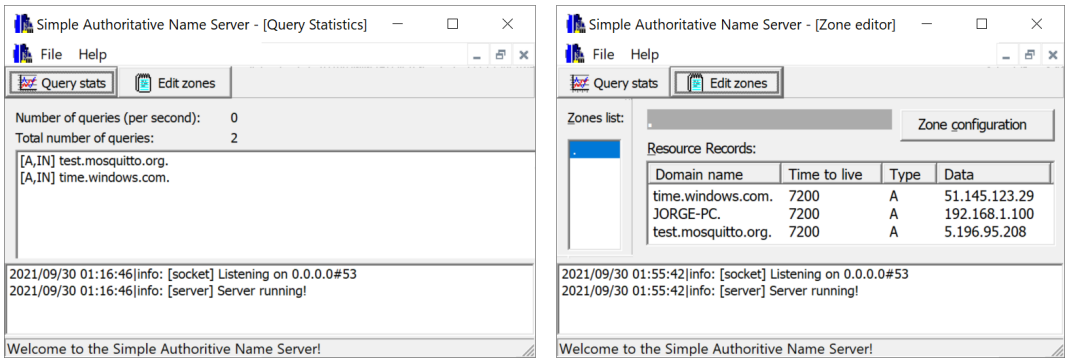

Figura VI – Servidor DNS SANS (1)

Tras añadir una zona DNS, han de introducirse todos los hostnames con sus direcciones IP.

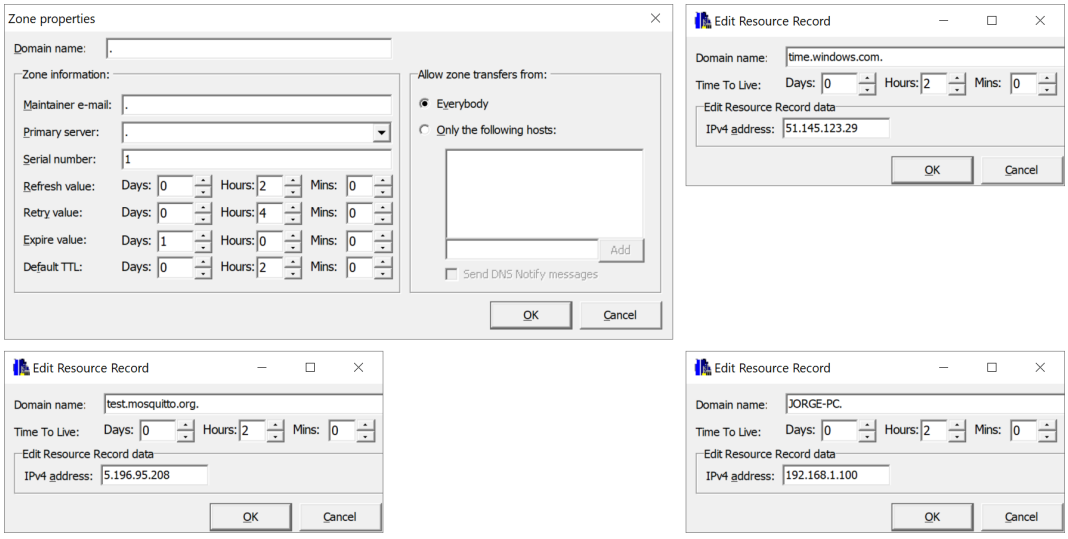

Figura VII – Servidor DNS SANS (2)

2.2.3. NTP

NTP es un protocolo cuya función es obtener (cliente) o proveer (servidor) la **fecha y hora**. Pertenece a la capa de aplicación y opera sobre la capa de transporte **UDP** (normalmente) en el puerto **123** (servidor).

El cliente obtiene la fecha y hora intercambiando únicamente dos mensajes con el servidor:

- **NTP request**: El cliente solicita al servidor la fecha y hora.
- **NTP response**: El servidor responde al cliente con el número de segundos transcurridos desde el 1 de enero de 1900.

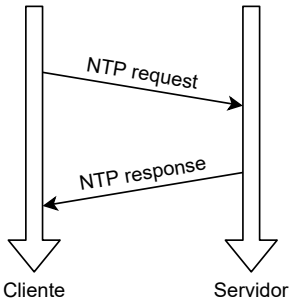

Figura VIII – Sincronización NTP

La especificación de la versión más reciente del protocolo en el momento de escritura de este documento (versión 4) se encuentra en el documento **IETF RFC 5905**.

Se han hecho uso de las siguientes implementaciones del protocolo NTP en las redes de dispositivos empleadas:

- **Cliente**: El único utilizado es el siguiente:
  - **NTP_CLIENT**: Corresponde al cliente NTP desarrollado por el autor de este documento. Su código fuente se encuentra en los anexos y requiere de **UDP_CLIENT**, también desarrollado por el autor de este documento y que hace uso de la API de sockets BSD. No se utilizan librerías NTP de terceros. Es usado por la aplicación del sistema embebido.
- **Servidores**: Algunos a los que el cliente puede conectar son:
  - **NTP Pool Project**: Corresponde a un grupo de servidores NTP en Internet. En este caso se ha usado `es.pool.ntp.org`.
  - **Windows Time Server**: Se trata del servidor NTP de Microsoft Windows. Su hostname es `time.windows.com`.
  - **TFTPD64**: Esta utilidad también cuenta con un servidor NTP, que ha sido activado a la vez que se ha configurado el servidor DHCP, haciendo clic en la pestaña "SNTP server". No se requieren más acciones y no genera ningún log.

2.2.4. TLS

TLS es un protocolo criptográfico cuya función es proporcionar **comunicaciones seguras**. Pertenece a la capa de aplicación y opera entre la capa de transporte **TCP** (para UDP el protocolo equivalente es DTLS) y otro protocolo de capa de aplicación, asegurando los datos intercambiados por este último entre el cliente y el servidor. Es el sucesor del protocolo SSL.

El puerto usado por el servidor viene determinado por el protocolo asegurado, y este puerto no suele coincidir con el utilizado en conexiones inseguras.

	HTTP	FTP	MQTT
Inseguro	80	20 - 21	1883
TLS	443	989 - 990	8883

Tabla I – Puertos usados por servidores de varios protocolos

Para que la comunicación entre el cliente y el servidor sea segura, esta ha de contar con las siguientes propiedades:

- **Confidencialidad**: Los datos intercambiados por el cliente y el servidor solo deben ser visibles por ellos, y no por un tercero que pueda capturar el tráfico mediante un sniffer de red.

- **Autenticación**: La identidad del servidor debe ser verificada, y la del cliente de manera opcional, evitando que un atacante suplante la identidad de uno de los dos, o la de ambos si actúa como un MiTM.

- **Integridad**: El cliente y el servidor deben comprobar que los datos intercambiados no han sido manipulados, falsificados o reproducidos.

Una conexión TLS consta de dos fases:

- Inicialmente, tras el handshake TCP, se lleva a cabo el handshake TLS, en el que el cliente establece una conexión segura con el servidor, normalmente mediante criptografía asimétrica en parte del proceso, la cual hace uso de dos claves para cifrar y descifrar mensajes:

  - **Clave pública**: Conocida por ambas partes de la comunicación y por terceros.

  - **Clave privada**: Conocida solo por solo una de las partes de la comunicación.

Y, de esta forma:

- Un mensaje encriptado con la clave pública solo puede ser desencriptado con la clave privada, lo que es útil para acordar entre el cliente y el servidor otras claves que se usarán para cifrar los datos intercambiados por el protocolo asegurado tras el handshake TLS.

- Un mensaje encriptado con la clave privada solo puede ser desencriptado con la clave pública, lo que es útil para autenticar el servidor y opcionalmente el cliente.

– Tras el handshake TLS, el cliente y el servidor ya están preparados para intercambiar datos sobre el protocolo asegurado, cifrados mediante criptografía simétrica con las claves acordadas durante el handshake TLS, donde una misma clave se usa tanto en encriptación como en desencriptación, y siendo más eficiente que la criptografía asimétrica.

Para satisfacer los requisitos de seguridad en la conexión TLS, las suites de cifrado proporcionan herramientas que permiten:

– Acordar entre el cliente y el servidor las claves simétricas usadas para cifrar el tráfico, que se obtienen a partir de tres secuencias aleatorias a las que se les aplica una función pseudoaleatoria en la que están implícitos algoritmos de hash:

  - **Client Random**: Generado por el cliente y enviado al servidor sin cifrar.

  - **Server Random**: Generado por el servidor y enviado al cliente sin cifrar.

  - **Pre-Master Secret**: Acordado entre el cliente y el servidor de manera que no pueda ser conocido por terceros.

  Para acordar el Pre-Master Secret entre el cliente y el servidor, estos recurren a un método de intercambio de clave, siendo algunos de ellos:

  - **RSA**: Haciendo uso de una infraestructura de clave pública, el cliente genera el Pre-Master Secret, lo encripta con la clave pública del servidor y es enviado a este, que es el único que podrá desencriptarlo con su clave privada.

  - **DH**: El cliente y el servidor aportan cada uno parámetros para crear conjuntamente el Pre-Master Secret usando algún algoritmo como DHE, ECDH o ECDHE. La seguridad de estos algoritmos radica en una conjetura matemática

basada en una función que es fácil de calcular en un sentido pero muy difícil en el inverso. Por esa razón, el Pre-Master Secret no puede ser interceptado por terceros. Los parámetros del servidor son firmados con su clave privada y verificados por el cliente usando la clave pública.

- **PSK**: El Pre-Master Secret es generado a partir de una clave previamente acordada entre el cliente y el servidor.

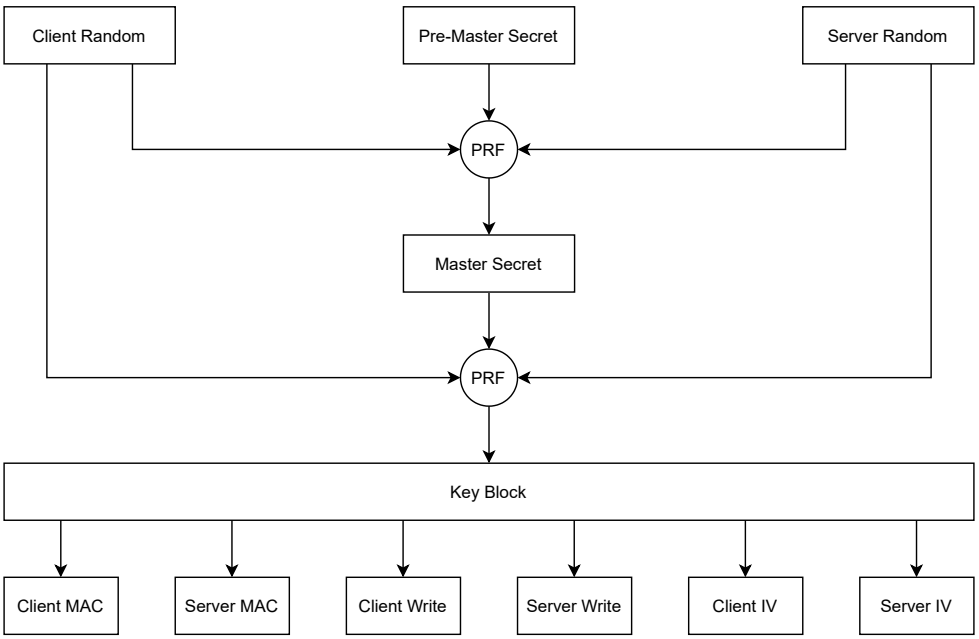

Figura IX – Generación de las claves de la sesión TLS

– Autenticar cada extremo de la comunicación, para lo cual existen distintos métodos:

- **RSA**: Haciendo uso de certificados de clave pública, la autenticación de un extremo se lleva a cabo al probarse que este cuenta con la clave privada correspondiente a su certificado. En el servidor esto ocurre cuando este requiere su clave privada durante el intercambio de clave para descifrar el Pre-Master Secret (RSA) o para firmar sus parámetros (DH), mientras que la autenticación del cliente se lleva a cabo al firmar con su clave privada los mensajes previos del handshake TLS, cuando dicha firma es verificada por el servidor haciendo uso de la clave pública del cliente.

- **DSA**: También mediante certificados, pero utilizando otro sistema de criptografía asimétrica menos extendido. Cuenta con la variante ECDSA.

- **PSK**: La autenticación se basa en una clave previamente acordada entre el cliente y el servidor.

- Establecer un algoritmo de criptografía simétrica. Algunos de ellos son AES, Camellia, TDEA, RC4 y DES. A estos algoritmos se les ha de especificar una longitud de la clave y, normalmente, también un modo de encriptación, por ejemplo, CBC o GCM.

- Comprobar la integridad de los datos intercambiados. Para ello, se adhiere a estos un código de autenticación MAC, que resulta de aplicar una función en la que intervienen los datos, unas de las claves simétricas intercambiadas y un algoritmo de hash. Este algoritmo de hash debería ser resistente a:

  - **Preimagen**: Dado un hash, es computacionalmente inviable construir un mensaje que lo genere.

  - **Segunda preimagen**: Dado un mensaje y su hash, es computacionalmente inviable construir otro mensaje diferente que genere el mismo hash.

  - **Colisiones**: Es computacionalmente inviable generar dos mensajes con el mismo hash.

  A continuación se tienen algunos de los algoritmos de hash más extendidos y sus vulnerabilidades encontradas hasta el momento de escritura de este documento. Estos algoritmos de hash también son usados en los procesos de firma y verificación de certificados digitales:

  - **MD5**: Hash de 128 bits que cuenta con colisiones, y también con un ataque que vulnera su resistencia a la preimagen. Además se computa muy rápido, lo que facilita los ataques por fuerza bruta. Estas características lo convierten en un hash inseguro, pero útil para comprobar la integridad de ficheros.

  - **SHA-1**: Hash de 160 bits que cuenta con al menos una colisión, lo que lo convierte en un hash potencialmente inseguro.

  - **SHA-2**: Hash de longitud variable (típicamente 256, 384 o 512 bits) seguro en el momento de escritura de este documento, al cumplir con los tres requisitos anteriores: resistencia a preimagen, segunda preimagen y colisiones.

  La existencia de vulnerabilidades en un algoritmo de hash lo convierte en inseguro ya que, aunque las colisiones puedan no comprometer realmente la seguridad del algoritmo de hash, estas pueden suponer un antecedente a otras vulnerabilidades que sí puedan hacerlo.

Habiéndose definido las herramientas de las que puede disponer una suite de cifrado, se tiene a continuación una a modo de ejemplo:

```
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384

- Protocol       : TLS                    - Cipher         : AES_256
- Key Exchange   : ECDHE                  - Mode           : GCM
- Authentication : RSA                    - Hash           : SHA384
```

Es importante tener en cuenta que algunas suites de cifrado usan el mismo sistema para el intercambio de clave y la autenticación, y en estos casos solo aparece una vez en el nombre de la suite de cifrado.

Quedando definidos los requisitos que plantean las comunicaciones seguras y las herramientas que existen para hacerlos frente, se establecen una serie de pasos para realizar el handshake TLS. Estos pasos dependen de muchos factores como la versión de TLS empleada y la suite de cifrado usada. En cualquier caso, durante el handshake TLS el cliente y el servidor:

- Establecen la versión de TLS y la suite de cifrado.
- Intercambian el Client Random y el Server Random.
- Opcionalmente solicitan, reciben y verifican el certificado del otro extremo.
- Acuerdan el Pre-Master Secret.
- Calculan las claves simétricas de la sesión.

A partir de este momento, el cliente y el servidor ya están preparados para intercambiar datos de manera segura, pero solo si se han tomado algunas precauciones, como las que se tienen a continuación (válidas en el momento de escritura de este documento), aunque esta lista no es exhaustiva:

- Usar TLS 1.2 o TLS 1.3.
- Verificar siempre los certificados.
- Si se usa RSA, utilizar claves de al menos 2048 bits.
- Emplear un algoritmo de hash SHA-2.
- Utilizar suites de cifrado que aseguren completamente la comunicación.

A continuación se tienen los esquemas del handshake de las versiones de TLS 1.2 y 1.3. Esta última, además de ser más moderna, cuenta con un handshake más rápido al ser realizado en un paso menos. Las flechas discontinuas indican mensajes que no siempre son enviados.

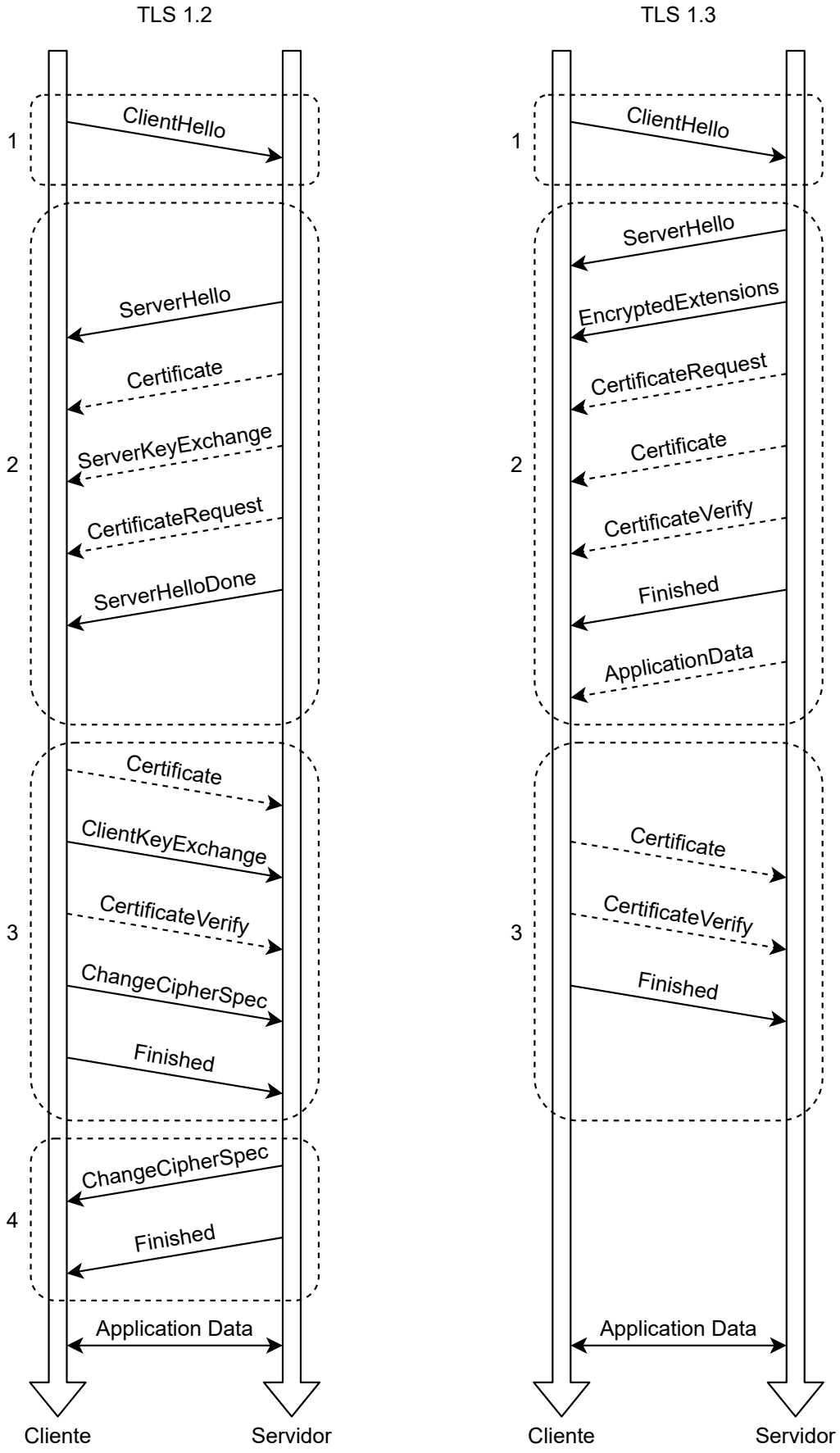

Figura X – Handshake TLS

La especificación del protocolo se encuentra en los siguientes documentos:

- TLS 1.2: **IETF RFC 5246**
- TLS 1.3: **IETF RFC 8446**

Se han hecho uso de las siguientes implementaciones del protocolo TLS en las redes de dispositivos empleadas para asegurar las comunicaciones por el protocolo MQTT, comentado más adelante:

- **Clientes**: Se cuenta con un cliente para el sistema embebido y otro para el PC, que son los siguientes:
  - **TLS_CLIENT**: Corresponde al cliente TLS 1.2 desarrollado por el autor de este documento. Su código fuente se encuentra en los anexos y requiere de un stack de TLS llamado **mbedTLS** y de **TCP_CLIENT**, este último también desarrollado por el autor de este documento y que hace uso de la API de sockets BSD. Es utilizado por la aplicación del sistema embebido.
  - **Eclipse Mosquitto**: Se trata de un broker/cliente MQTT que al mismo tiempo es un servidor/cliente TLS 1.2/1.3, ya que permite conexiones seguras haciendo uso de **OpenSSL**. Es usado por el PC.
- **Servidores**: Los clientes anteriores han conectado a los siguientes servidores:
  - **Mosquitto Test Server**: Corresponde a un broker MQTT de Internet que admite conexiones seguras mediante TLS.
  - **Eclipse Mosquitto**: Se trata de un broker/cliente MQTT que al mismo tiempo es un servidor/cliente TLS 1.2/1.3, ya que permite conexiones seguras haciendo uso de **OpenSSL**. Es usado por el PC.

Estos clientes y servidores TLS son configurados a la vez que el protocolo asegurado, que en este caso es MQTT.

2.2.5. MQTT

MQTT es un protocolo de **mensajería** basado en un modelo de publicador y suscriptor. Pertenece a la capa de aplicación y opera sobre la capa de transporte **TCP** en el puerto **1883** (servidor, conexiones inseguras sin TLS) o **8883** (servidor, conexiones seguras con TLS). En una red MQTT hay dos tipos de entidades: los clientes y el broker (servidor). Todas las conexiones son entre un cliente y el broker.

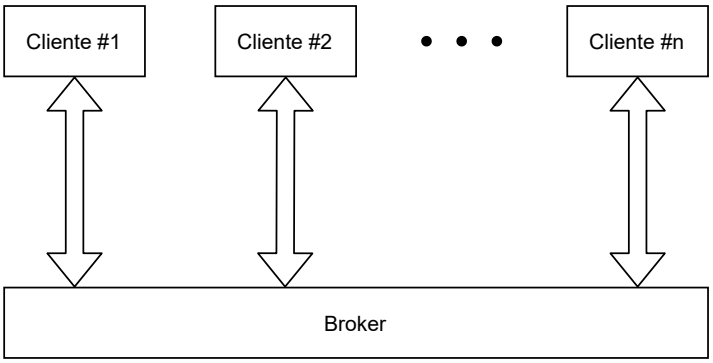

Figura XI – Red MQTT

Los clientes son aquellos que, tras conectar al broker, pueden publicar mensajes en temas y recibir los mensajes publicados (por cualquier cliente de la red MQTT) en los temas a los que se han suscrito.

El broker, único en la red MQTT, actúa de intermediario que mantiene una sesión con cada uno de los clientes y recibe de ellos los mensajes publicados en temas para reenviárselos a sus suscriptores. Todo el tráfico de la red MQTT pasa por él.

El protocolo MQTT funciona intercambiando diferentes tipos de paquetes de control que, en la versión 5.0 del mismo, son los que muestran a continuación.

#	Paquete	Dirección	
		Cliente	Broker
00	Reserved	—	
01	CONNECT	⟶	
02	CONNACK	⟵	
03	PUBLISH	⟷	
04	PUBACK	⟷	
05	PUBREC	⟷	
06	PUBREL	⟷	
07	PUBCOMP	⟷	

#	Paquete	Dirección	
		Cliente	Broker
08	SUBSCRIBE	⟶	
09	SUBACK	⟵	
10	UNSUBSCRIBE	⟶	
11	UNSUBACK	⟵	
12	PINGREQ	⟶	
13	PINGRESP	⟵	
14	DISCONNECT	⟷	
15	AUTH	⟷	

Tabla II – Paquetes MQTT v5.0

Los nombres de la mayoría de los paquetes son autoexplicativos, no obstante, a continuación se muestra la función de cada uno de ellos tras recogerlos en tres grupos:

– **Conexiones**: El cliente ha de solicitar la conexión con el broker enviándole un paquete CONNECT, incluyendo unas credenciales de acceso si este las requiriese. Como novedad en MQTT v5.0, el broker puede exigir una autenticación más compleja a través de un intercambio de paquetes AUTH. Tras este proceso, el broker ha de contestar con un paquete CONNACK indicando si la conexión del cliente ha sido exitosa o no. Para evitar que el broker expulse al cliente por inactividad, este debería enviar periódicamente un paquete PINGREQ, al que el broker contestaría con un paquete PINGRESP. Para finalizar la conexión, el cliente ha de notificárselo al broker enviándole un paquete DISCONNECT.

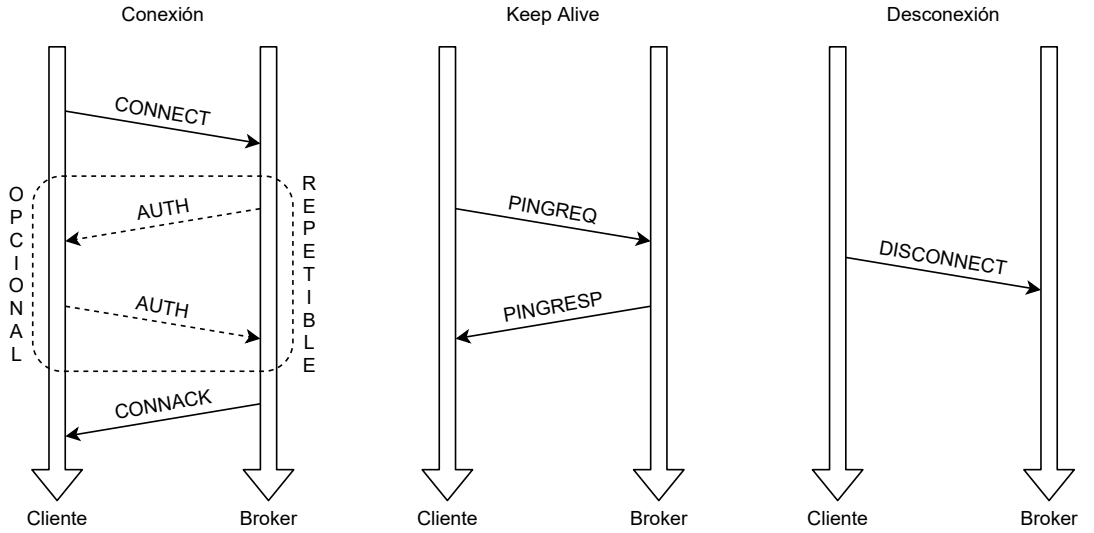

Figura XII – Conexiones MQTT

– **Publicaciones**: Cuando se publica un mensaje, este es enviado del cliente publicador al broker y luego del broker a los clientes suscriptores. Por ello, tanto los clientes como el broker envían y reciben mensajes, motivo por el cual se habla de emisores y receptores de mensajes. El protocolo permite establecer una calidad del servicio QoS, que determina el nivel de garantía de la entrega de los mensajes. Por esta razón, existen tres formas de publicarlos:

  ◦ **QoS 0**: El emisor envía una única vez (sin tener en cuenta retransmisiones TCP) un paquete PUBLISH que contiene el mensaje. De esta forma, el receptor recibe el mensaje como **máximo una vez**.

  ◦ **QoS 1**: El emisor envía un paquete PUBLISH que contiene el mensaje y espera a obtener un paquete PUBACK con el que ha de contestar el receptor. Si el

emisor no obtiene el paquete PUBACK en un tiempo establecido, este reenvía el paquete PUBLISH tantas veces como sea necesario hasta que sí lo obtenga. De esta forma, el receptor recibe el mensaje como **mínimo una vez**.

- **QoS 2**: El emisor envía un paquete PUBLISH que contiene el mensaje y espera a obtener un paquete PUBREC con el que ha de contestar el receptor. Si el emisor no obtiene el paquete PUBREC en un tiempo establecido, este reenvía el paquete PUBLISH tantas veces como sea necesario hasta que sí lo obtenga. Tras ello, el emisor sabe que ha conseguido entregar el mensaje al receptor, pero el receptor no sabe que el emisor tiene esta información, de forma que futuros mensajes podrían ser nuevos o duplicados del anterior. Por eso, el emisor, como respuesta al paquete PUBREC, envía un paquete PUBREL para que el receptor sepa que el emisor sabe que ha conseguido entregar el mensaje, al que el receptor ha de contestar con un paquete PUBCOMP cuando lo reciba. Si el receptor no obtiene el paquete PUBREL en un tiempo establecido, este vuelve a contestar con el paquete PUBREC al mensaje PUBLISH inicial, tantas veces como sea necesario hasta que sí lo obtenga. De esta forma, el receptor recibe el mensaje **exactamente una vez**, ya que con este mecanismo descarta mensajes duplicados.

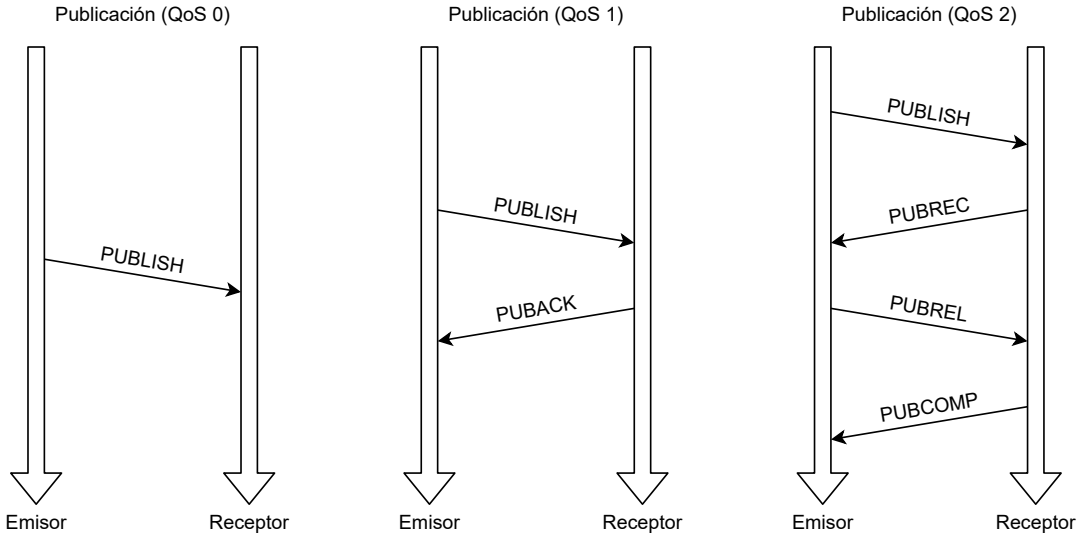

Figura XIII – Publicaciones MQTT

– **Suscripciones**: Un cliente envía un paquete SUBSCRIBE indicando el tema al que se quiere suscribir, al cual el broker ha de contestar con un paquete SUBACK aceptando la suscripción o indicando un código de error. Lo mismo ocurre con las desuscripciones, donde intervienen unos paquetes UNSUBSCRIBE y UNSUBACK respectivamente. Ha de tenerse en cuenta que, al suscribirse un tema, también se establece una QoS al igual que al publicar un mensaje, de manera que, si

el broker le envía un mensaje a un cliente y la QoS del paquete PUBLISH es superior al establecido en el paquete SUBSCRIBE, se usará la QoS establecida en la suscripción.

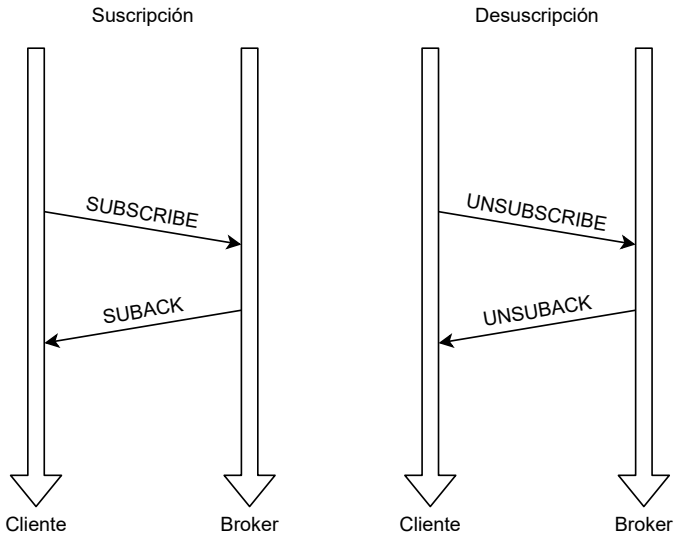

Figura XIV – Suscripciones MQTT

La especificación de la versión 5.0 del protocolo, la más reciente en el momento de de escritura de este documento, se encuentra en un estándar de **OASIS**.

Se han hecho uso de las siguientes implementaciones del protocolo MQTT en las redes de dispositivos empleadas:

- **Clientes**: Se cuenta con un cliente para el sistema embebido y otro para el PC, que son los siguientes:

  - **MQTT_CLIENT**: Corresponde al cliente MQTT v5.0 (QoS 0) desarrollado por el autor de este documento. Su código fuente se encuentra en los anexos y requiere de **TCP_CLIENT**, también desarrollado por el autor de este documento y que hace uso de la API de sockets BSD. No se utilizan librerías MQTT de terceros. Es usado por la aplicación del sistema embebido.

    Si se desea establecer una conexión segura, ha de situarse entre los dos clientes anteriores **TLS_CLIENT**, también desarrollado por el autor de este documento. Este último requiere de un stack de TLS llamado **mbedTLS**.

  - **Eclipse Mosquitto**: Se trata de un broker/cliente MQTT. Permite conexiones TLS gracias a **OpenSSL** y es usado por el PC para interactuar con la aplicación del sistema embebido. El cliente lo forman los siguientes archivos ejecutables, situados en el directorio de instalación:

◇ **mosquitto_pub**: Sirve para publicar mensajes en temas.

◇ **mosquitto_sub**: Permite suscribirse a un tema y recibir mensajes.

Se han ejecutado estos programas usando los siguientes argumentos:

◇ `-V` : Versión de MQTT
◇ `-h` : Hostname del broker
◇ `-p` : Puerto del broker
◇ `-t` : Tema
◇ `-m` : Mensaje
◇ `--cafile` : Certificado raíz e intermedios si los hay
◇ `--cert` : Certificado del cliente
◇ `--key` : Par de claves del cliente

Es recomendable "entrecomillar" todos los valores que contengan estos argumentos, y es obligatorio en aquellos que contengan espacios.

Se han detectado problemas en Windows al usar caracteres que no pertenecen al código ASCII original (7 bits), por ejemplo, "Ú" con tilde y sin comillas. Ocurre que, al escribir este caracter en la consola, aparentemente lo interpreta como ANSI (codificación predeterminada de Windows) y no UTF-8 (usado por MQTT), lo que genera un error. Para arreglarlo, ha de cambiarse la codificación del sistema operativo activando la siguiente casilla:

```
Control Panel -> Region -> Administrative -> Change system locale...  ->
    Beta: Use Unicode UTF-8 for worldwide language support
```

Para establecer conexiones seguras, es obligatorio usar el hostname del broker y no su dirección IP o "localhost" en caso de que el cliente y el broker estén en el mismo PC. Esto último haría que fallase la verificación del certificado del broker.

Como último apunte, es recomendable meter los certificados y pares de claves (todos ellos generados en el siguiente capítulo) en una nueva carpeta, pero en estos ejemplos se encuentran todos en el directorio de instalación, por simplificar y no tener que incluir la ruta completa o relativa de cada uno.

A continuación se tienen los comandos usados para interactuar con la aplicación creada para el sistema embebido. Se han dividido en varias líneas, pero han de ejecutarse en una única línea cada uno. En todo momento se utiliza QoS 0:

◇ Conexión insegura:

▵ ------------------------ JORGE-PC ------------------------

```
mosquitto_pub                    -V                      "5"
     -h              "JORGE-PC"  -p                    "1883"
     -t        "[TEMA_SECRETO]"  -m                 "Hola XD"
```

```
mosquitto_sub                    -V                      "5"
     -h              "JORGE-PC"  -p                    "1883"
     -t        "[TEMA_PÚBLICO]"
```

▵ -------------------- test.mosquitto.org --------------------

```
mosquitto_pub                    -V                      "5"
     -h    "test.mosquitto.org"  -p                    "1883"
     -t        "[TEMA_SECRETO]"  -m                 "Hola XD"
```

```
mosquitto_sub                    -V                      "5"
     -h    "test.mosquitto.org"  -p                    "1883"
     -t        "[TEMA_PÚBLICO]"
```

◇ Conexión TLS con autenticación del servidor:

▵ ------------------------ JORGE-PC ------------------------

```
mosquitto_pub                    -V                      "5"
     -h              "JORGE-PC"  -p                    "8883"
     -t        "[TEMA_SECRETO]" --cafile          "ca.chain"
     -m               "Hola XD"
```

```
mosquitto_sub                    -V                      "5"
     -h              "JORGE-PC"  -p                    "8883"
     -t        "[TEMA_PÚBLICO]" --cafile          "ca.chain"
```

▵ -------------------- test.mosquitto.org --------------------

```
mosquitto_pub                    -V                      "5"
     -h    "test.mosquitto.org"  -p                    "8883"
     -t        "[TEMA_SECRETO]" --cafile "mosquitto.org.crt"
     -m               "Hola XD"
```

```
mosquitto_sub                    -V                    "5"
     -h    "test.mosquitto.org"  -p                  "8883"
     -t        "[TEMA_PÚBLICO]" --cafile "mosquitto.org.crt"
```

◇ Conexión TLS con autenticación del servidor y del cliente:

▵ ------------------------ JORGE-PC ------------------------

```
mosquitto_pub                    -V                    "5"
     -h              "JORGE-PC"  -p                  "8884"
     -t        "[TEMA_SECRETO]" --cafile          "ca.chain"
    --cert         "client.crt" --key           "client.key"
     -m               "Hola XD"
```

```
mosquitto_sub                    -V                    "5"
     -h              "JORGE-PC"  -p                  "8884"
     -t        "[TEMA_PÚBLICO]" --cafile          "ca.chain"
    --cert         "client.crt" --key           "client.key"
```

▵ -------------------- test.mosquitto.org --------------------

```
mosquitto_pub                    -V                    "5"
     -h    "test.mosquitto.org"  -p                  "8884"
     -t        "[TEMA_SECRETO]" --cafile "mosquitto.org.crt"
    --cert           "test.crt" --key           "client.key"
     -m               "Hola XD"
```

```
mosquitto_sub                    -V                    "5"
     -h    "test.mosquitto.org"  -p                  "8884"
     -t        "[TEMA_PÚBLICO]" --cafile "mosquitto.org.crt"
    --cert           "test.crt" --key           "client.key"
```

– **Brokers**: Los clientes anteriores han conectado a los siguientes brokers:

◦ **Mosquitto Test Server**: Corresponde a un broker MQTT de Internet que admite conexiones seguras mediante TLS.

◦ **Eclipse Mosquitto**: Se trata del broker/cliente MQTT que se usa en el PC. Para poder tener control sobre cuando el broker está funcionando, es recomendable no instalar el servicio del OS. El broker ha sido configurado añadiendo las siguientes opciones a un fichero de configuración:

- `allow_anonymous` : Determina si los clientes pueden conectar al broker sin proporcionar un nombre de usuario y una contraseña. En las tres configuraciones que se muestran a continuación se permite que los clientes conecten al broker sin autenticación con un nombre de usuario y una contraseña ya que, en el único caso donde se requiere autenticación por parte del cliente, esta se lleva a cabo mediante su certificado digital.
- `listener` : Establece el puerto TCP usado por el broker.
- `cafile` : Establece el archivo que contiene los certificados requeridos para verificar el certificado del cliente y el del broker, que en este caso son un certificado intermedio y el certificado raíz de confianza.
- `certfile` : Establece el certificado del broker.
- `keyfile` : Establece el par de claves del broker.
- `require_certificate` : Determina si se requiere de autenticación del cliente mediante un certificado digital.
- `use_identity_as_username` : Determina si se usa el nombre del titular del certificado del cliente como su nombre de usuario en la conexión MQTT.

Se han creado tres ficheros de configuración (desde un archivo en blanco o modificando la plantilla **mosquitto.conf**, situada en el directorio de instalación de Eclipse Mosquitto), uno por cada tipo de conexión:

- Conexión insegura (**1883.conf**):

```
allow_anonymous                                    true
listener                                           1883
```

◇ Conexión TLS con autenticación del servidor (**8883.conf**):

```
allow_anonymous                                    true
listener                                           8883
cafile                                           ca.chain
certfile                                       broker.crt
keyfile                                        broker.key
```

◇ Conexión TLS con autenticación del servidor y del cliente (**8884.conf**):

```
allow_anonymous                                    true
listener                                           8884
cafile                                           ca.chain
certfile                                       broker.crt
keyfile                                        broker.key
require_certificate                                true
use_identity_as_username                           true
```

Por último, para iniciar el broker ha de ejecutarse el siguiente comando, indicando el archivo de configuración con el argumento `-c` y que se desea generar un log con el argumento `-v`:

```
mosquitto   -c "mosquitto.conf"   -v
```

La sesión que se muestra a continuación corresponde a una conexión de la aplicación creada por el autor de este documento para un sistema embebido.

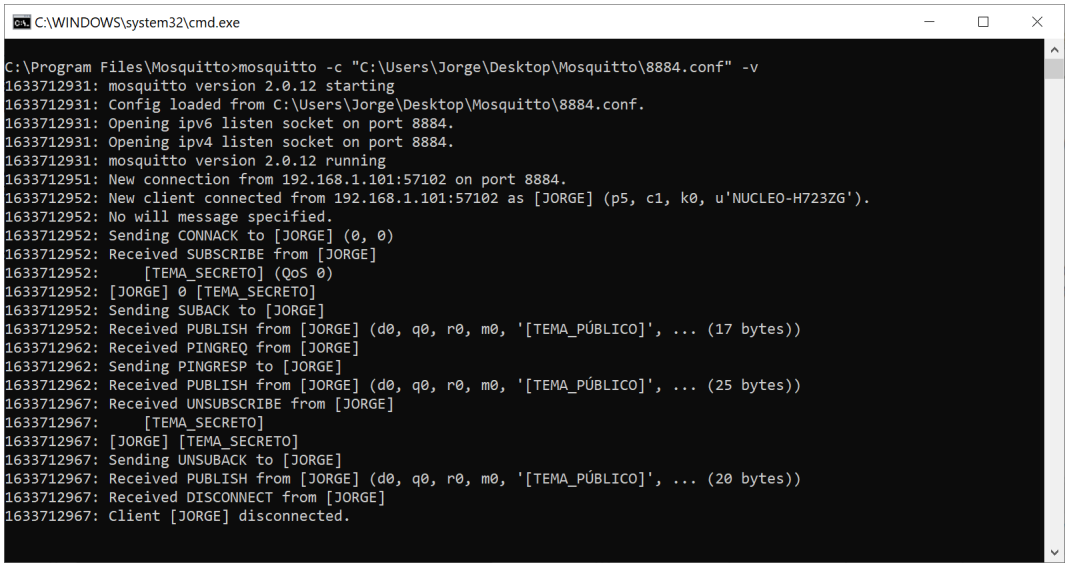

Figura XV – Broker MQTT Eclipse Mosquitto

2.3. Análisis de protocolos

2.3.1. Captura de tramas de red

Para estudiar los protocolos de comunicaciones es fundamental disponer de un analizador de red como **Wireshark**, el cual permite capturar tramas y desglosarlas en paquetes, segmentos y datos, siguiendo el esquema que se muestra a continuación y mostrando cada uno de los protocolos usados del modelo TCP/IP.

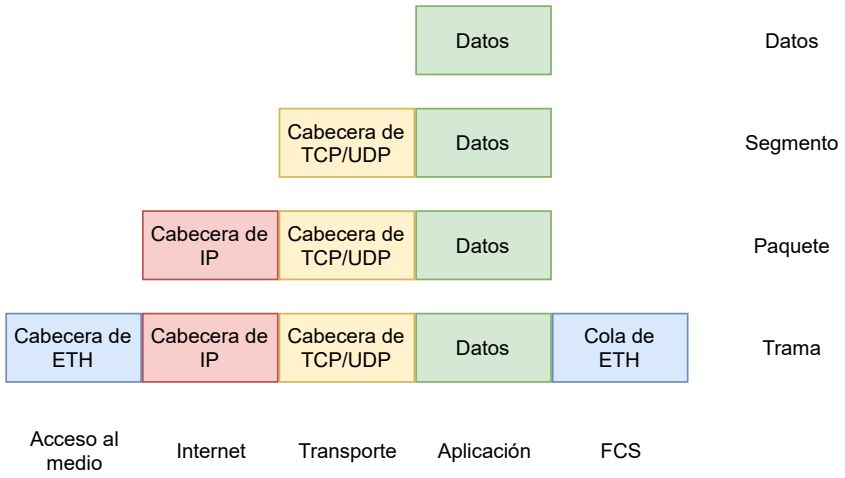

Figura XVI – Estructura de una trama de red

Su uso es tan sencillo como iniciar el programa, escoger una interfaz de red y comenzar a capturar tráfico. A continuación se pueden aplicar filtros, siendo algunos de ellos:

- Tramas que contengan una dirección IPv4 en el origen o destinatario:

  ```
  ip.addr  == 192.168.1.1
  ```

- Tramas que contengan un puerto UDP en el origen o destinatario:

  ```
  udp.port == 53
  ```

- Tramas que contengan un puerto TCP en el origen o destinatario:

  ```
  tcp.port == 8883
  ```

- Tramas que contengan una cadena de texto. Este filtro no aplica al tráfico desencriptado:

  ```
  frame contains "JORGE-PC"
  ```

Para combinar filtros, se pueden usar operaciones lógicas OR y AND, mediante `"||"` y `"&&"` respectivamente (sin comillas).

Tras capturar y filtrar las tramas deseadas, el programa permite exportarlas a un archivo.

```
File -> Export Specified Packets...
```

2.3.2. Descifrado de sesiones encriptadas

Wireshark también permite desencriptar el tráfico TLS cargando un archivo que contenga la siguiente información de la sesión:

- **Client Random**: Se encuentra sin cifrar en el Client Hello, por lo que es conocido por cualquiera que capture el paquete que lo contiene.

- **Master Secret**: Solamente es conocido por el cliente y el servidor, por lo que ha de obtenerse de uno de los dos. Se puede extraer de un log de mbedTLS si se habilita en él la depuración por `printf()`, tras el mensaje "dumping 'master secret' (48 bytes)", copiando y juntando sin espacios tres fragmentos de 16 bytes cada uno (2 números hexadecimales → 4 bits + 4 bits = 8 bits = 1 byte).

Conociendo estas dos secuencias, se debe crear un archivo de texto de una única línea con la siguiente estructura, y sin usar comillas:

```
CLIENT_RANDOM "Client Random" "Master Secret"
```

Resultando, a modo de ejemplo, y omitiendo en él muchos caracteres intermedios que sí han de aparecer en el archivo:

```
CLIENT_RANDOM 615fbb...e4e540 f87c86...409a4b
```

Este archivo ha de ser cargado en la sesión de Wireshark en:

```
Edit > Preferences > Protocols > TLS > (Pre)-Master-Secret log filename
```

Se puede ver el tráfico desencriptado tras cargar el archivo. En las siguientes páginas se muestra una sesión (de la aplicación creada para un sistema embebido) donde se ha hecho uso de los protocolos expuestos en la sección anterior, primero cifrada y luego descifrada. Se ha establecido la dirección IP `192.168.1.1` en el PC y `192.168.1.101` en el sistema embebido. La red creada la forman únicamente estos dos dispositivos.

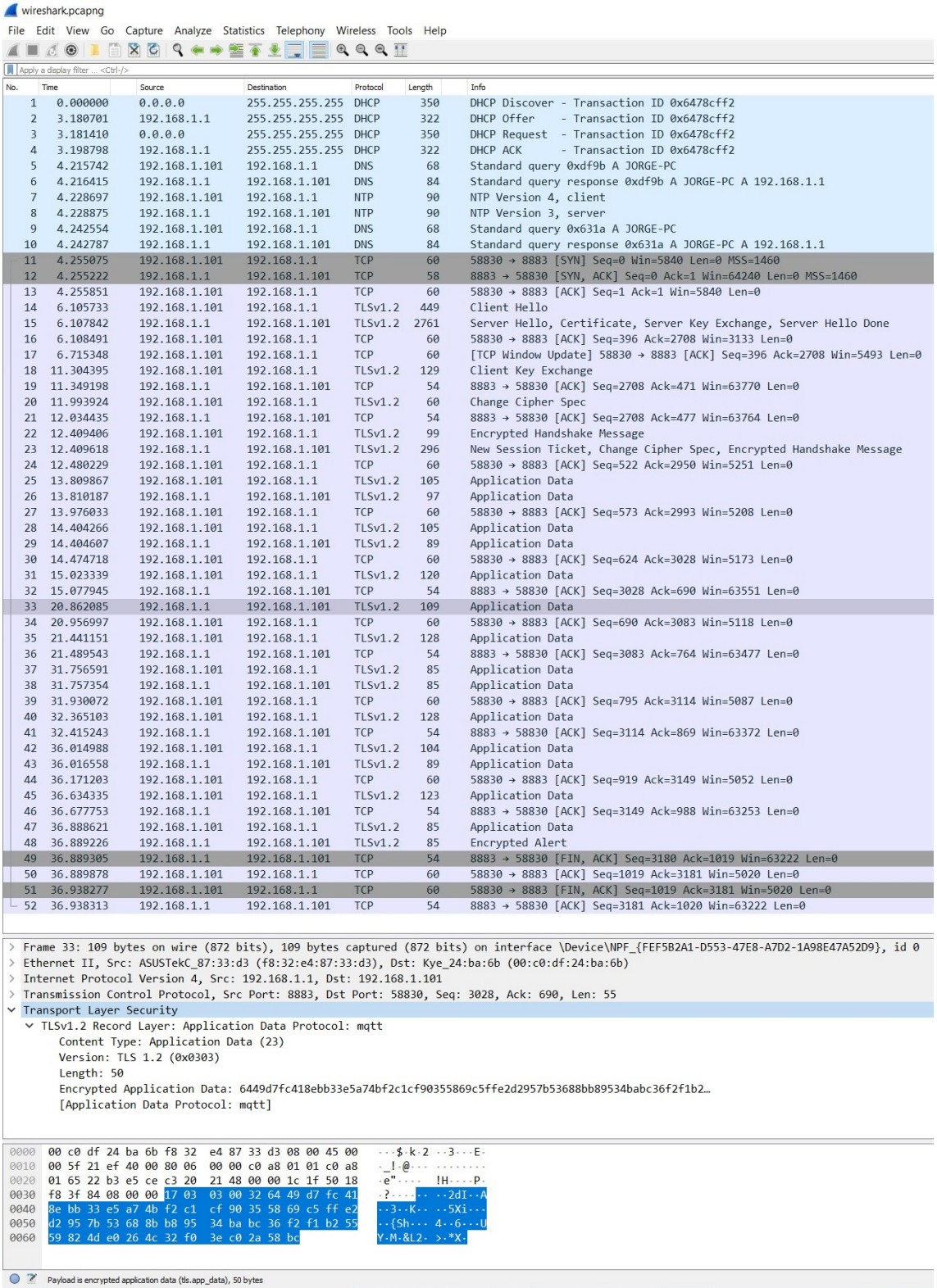

Figura XVII – Sesión cifrada

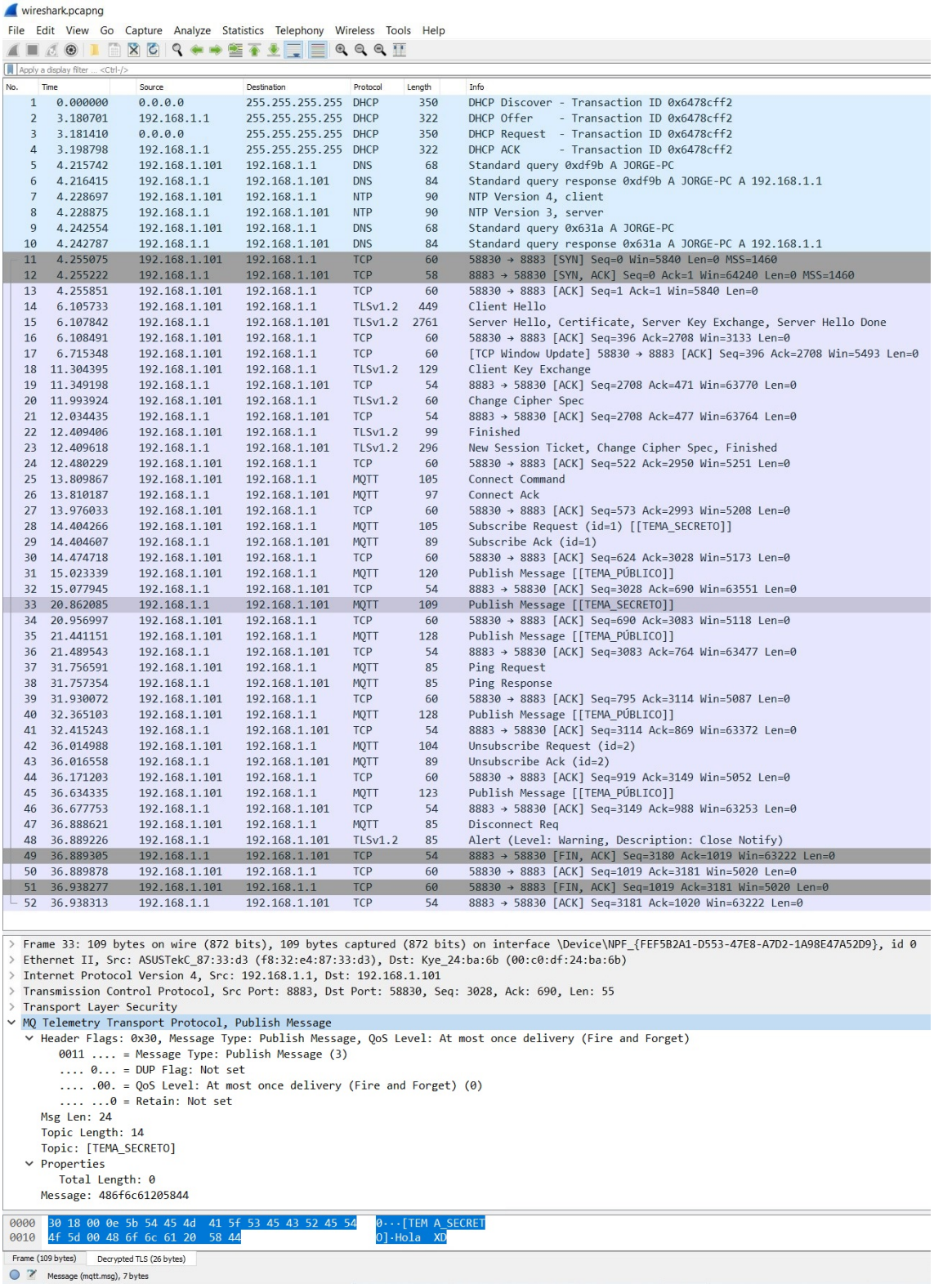

Figura XVIII – Sesión descifrada

3

Infraestructuras de clave pública

3.1. Aspectos teóricos

3.1.1. Certificados X.509

X.509 corresponde a un formato de certificados digitales ampliamente utilizado en infraestructuras de clave pública. Un certificado de este tipo es un archivo asociado a un titular en el que, tras una cabecera inicial, se distinguen los siguientes campos:

- **tbsCertificate**: Contiene unos datos entre los que se incluyen el nombre y la clave pública del titular del certificado, el nombre de la autoridad de certificación que lo ha emitido y el periodo de validez del mismo, entre otras cosas.

- **signatureAlgorithm**: Establece el algoritmo de firma digital con el que la autoridad de certificación que ha emitido el certificado ha cifrado un hash de los datos anteriores, haciendo uso de su clave privada.

- **signatureValue**: Guarda la firma digital que se obtiene tras aplicar el algoritmo de firma digital a los datos anteriores, protegiéndolos ante modificaciones, ya que estas invalidarían la firma digital que, para que pueda ser descifrada con la clave pública de la autoridad de certificación que lo ha emitido, ha de haber sido creada con su clave privada que únicamente ella ha de tener.

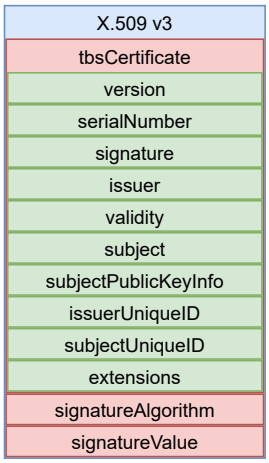

Figura XIX – Certificado X.509

Estos certificados desempeñan un rol en las siguientes tareas:

- **Firma de documentos**: Es el proceso que consiste en adjuntar a un documento su hash encriptado con la clave privada de un certificado. De esta manera, se puede comprobar si el documento ha sido modificado tras su firma, al comparar el hash del documento con la firma desencriptada con la clave pública del certificado.

- **Cifrado de mensajes**: Si se encripta un mensaje con la clave pública de un certificado, solo su titular podrá desencriptarlo, con su clave privada. Este mensaje puede ser una clave simétrica usada para cifrar el tráfico entre dos extremos.

- **Autenticación de usuarios**: El titular de un certificado puede autenticarse si de cualquier manera demuestra que posee la clave privada.

Para que una entidad obtenga un certificado del cual sea titular, ha de generar los siguientes archivos:

- **Par de claves (KEY)**: Contiene varios parámetros a partir de los cuales se obtienen su clave pública y clave privada.

- **Solicitud de firma del certificado (CSR)**: Recoge la información que contendrá su certificado, incluyendo el nombre y la clave pública de su titular y, opcionalmente, otros datos como su organización o país.

A continuación una autoridad de certificación podrá generar los siguientes archivos:

- **Certificado X.509 (CRT)**: Resulta de firmar con su clave privada la solicitud, información del certificado del emisor, un periodo de validez y permisos acerca de la emisión de otros certificados bajo el recién creado.

- **Lista de revocación (CRL)**: Permite revocar certificados que hayan sido comprometidos, por ejemplo, si la clave privada de su emisor ha sido robada.

Los cuatro archivos anteriores admiten varios esquemas de codificación, siendo los más usados los siguientes:

- **DER**: Se trata de una codificación binaria **ASN.1**.

- **PEM**: Corresponde a una codificación **Base64 ASCII** que resulta de sustituir cada 6 bits de un certificado DER por un caracter, y añadiendo dos etiquetas que marcan el comienzo y el final del certificado PEM.

3.1.2. Cadenas de confianza

El certificado de una entidad final ha de ser verificado para que este sea válido. Para ello, ha de obtenerse el certificado de la autoridad de certificación que lo ha emitido y extraerse de él su clave pública. Tras desencriptar con ella la firma del certificado de dicha entidad final, este puede ser verificado tras comparar el hash de los datos que contiene con la firma desencriptada.

El proceso de verificación de dicha entidad final no termina con el paso anterior, ya que también debe verificarse el certificado de la autoridad de certificación que lo ha emitido. Por ello, ha de repetirse el proceso anterior con los certificados superiores hasta el certificado raíz, en el que su titular y su emisor corresponden a la misma entidad, es decir, se encuentra autofirmado.

Para verificar el certificado raíz ha de llevarse a cabo un proceso que difiere del realizado durante la verificación del resto de certificados, ya que el certificado raíz no se puede comprobar contra el certificado de la autoridad de certificación que lo ha emitido, al ser el mismo. Por ello, la manera de verificarlo consiste en contar con él preinstalado en un almacén de certificados raíz de confianza.

Los certificados que han de ser verificados para autenticar una entidad forman su cadena de confianza. Se tiene a continuación un ejemplo de una con tres certificados.

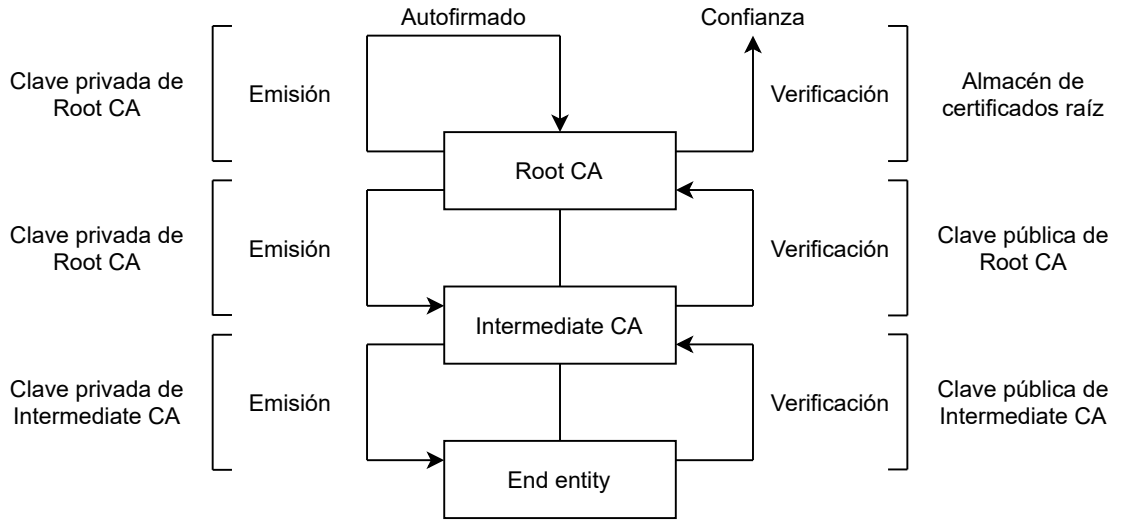

Figura XX – Cadena de confianza

Este ejemplo representa la cadena de confianza verificada al conectar de forma segura a un sitio como la web de GitHub, de donde se han recibido los certificados mostrados a continuación, exceptuando el certificado raíz que el cliente ya ha de tener.

```
DigiCert High Assurance EV Root CA
|_ DigiCert High Assurance TLS Hybrid ECC SHA256 2020 CA1
   |_ github.com
```

Antes de nada se ha de comprobar que en el certificado de la entidad final GitHub aparece su hostname. A continuación ha de verificarse su certificado contra el intermedio, luego este contra el raíz y finalmente se comprueba que este último es de confianza al estar previamente instalado en el sistema del cliente web.

Cabe preguntarse acerca de la procedencia de los certificados raíz. Los sistemas operativos suelen disponer de un almacén de certificados raíz con aquellos de importantes empresas de certificación (DigiCert, GlobalSign...), y este es generalmente usado al conectar a páginas web o al verificar archivos firmados digitalmente. Estos certificados raíz pueden ser añadidos o eliminados manualmente o a través de actualizaciones del sistema operativo. Sin embargo, este almacén del sistema operativo no siempre es usado para verificar certificados, ya que algunos programas (como Eclipse Mosquitto) o los sistemas embebidos tienen su propio almacén de certificados.

Otra importante cuestión viene acerca de la presencia de certificados intermedios en una cadena de confianza. Teóricamente, podría formarla un único certificado raíz correspondiente a la entidad final, pero esto no resultaría nada práctico, ya que para ello haría falta que todos los certificados que se quieren verificar estuviesen en el almacén de certificados de confianza. Pero esto no resuelve la cuestión anterior, ya que se podría poner solución a este problema con dos certificados, el raíz y el de la entidad final, sin uno intermedio. Es en este momento cuando hay que darle gran importancia a la protección de la clave privada de un certificado raíz ya que, si cayese en malas manos, un usuario malintencionado podría empezar a emitir certificados válidos bajo un certificado raíz de muy larga duración e instalado en muchísimos dispositivos. Para evitar esto, se minimiza el uso de la clave privada del certificado raíz emitiendo certificados intermedios de corta duración y son estos los que se encargan de emitir los certificados a las entidades finales. En resumen, los certificados intermedios constituyen una capa de seguridad adicional.

3.1.3. Información adicional

Habiéndose definido aquellos aspectos teóricos relativos a una infraestructura de clave pública formada por certificados X.509 cuyas cadenas de confianza son verificadas, puede encontrarse información adicional si se consulta la especificación **IETF RFC 5280**.

3.2. Creación de una PKI

3.2.1. Kits de herramientas

Para crear una infraestructura de clave pública es necesario contar con herramientas que puedan generar:

- Pares de claves
- Solicitudes de firma del certificado
- Certificados X.509

Existen diversos kits de herramientas que pueden generar los archivos anteriores, siendo OpenSSL el más conocido de todos. Sin embargo, para el proyecto de programación creado por el autor de este documento se ha utilizado **mbedTLS**, el cual no es tan completo como OpenSSL, pero es suficiente para la aplicación creada, y de esta manera se usa mbedTLS tanto como stack para el cliente TLS como para crear la infraestructura de clave pública.

La versión de mbedTLS usada es la 2.16.11, que corresponde a una versión LTS, es decir, una versión que, aunque no cuente con las últimas características, está muy mantenida con el objetivo de minimizar el número de errores que pueda tener.

mbedTLS incluye unos programas que permiten crear fácilmente una infraestructura de clave pública. Se han utilizado los siguientes:

```
/mbedtls-mbedtls-2.16.11.zip/mbedtls-mbedtls-2.16.11/programs/pkey/gen_key.c
/mbedtls-mbedtls-2.16.11.zip/mbedtls-mbedtls-2.16.11/programs/x509/cert_req.c
/mbedtls-mbedtls-2.16.11.zip/mbedtls-mbedtls-2.16.11/programs/x509/cert_write.c
```

Si se cuenta con Microsoft Visual Studio, se puede compilar un proyecto de programación ya creado situado en:

```
/mbedtls-mbedtls-2.16.11.zip/mbedtls-mbedtls-2.16.11/visualc/VS2010
```

De esta manera, ya se cuenta con los tres archivos ejecutables necesarios para crear la infraestructura de clave pública.

3.2.2. Intranet

Para establecer conexiones seguras mediante TLS con un broker MQTT instalado en un PC de la intranet, se ha creado la infraestructura de clave pública que se muestra a continuación.

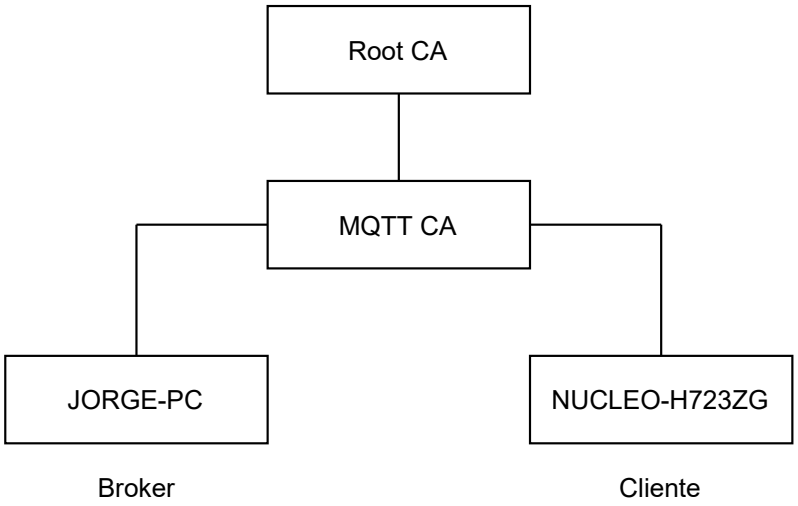

Figura XXI – Infraestructura de la intranet

Tanto el broker como el cliente tienen cada uno su par de claves, su propio certificado y el certificado raíz de confianza. Además, el broker cuenta con el certificado intermedio que ha de enviar al cliente durante el handshake TLS. De esta forma se puede establecer una conexión TLS con autenticación en ambos extremos: el cliente recibe y verifica la cadena de confianza del broker, y el broker opcionalmente recibe y verifica la cadena de confianza del cliente. Ha de tenerse en cuenta que, para que la verificación del broker sea posible, ha de aparecer su hostname en su certificado. Y si se trata de conectar al broker directamente escribiendo su dirección IP en lugar de su hostname, la verificación del certificado fallará, al no aparecer en él ninguna dirección IP.

A continuación se tienen los comandos utilizados que hacen uso de los archivos ejecutables compilados para crear la infraestructura de la intranet. Algunos de ellos se han dividido en varías líneas porque son demasiados largos, pero han de ejecutarse en una única línea cada uno:

- Generar los pares de claves con **`gen_key`** y los siguientes argumentos:
  - `rsa_keysize` : Longitud de la clave RSA, en bits.
  - `filename` : Nombre del archivo de salida con el par de claves.

Los comandos que se han ejecutado son los siguientes:

- Autoridad de certificación raíz:

```
gen_key          rsa_keysize=2048          filename="root_ca.key"
```

- Autoridad de certificación intermedia:

```
gen_key          rsa_keysize=2048          filename="mqtt_ca.key"
```

- Broker:

```
gen_key          rsa_keysize=2048           filename="broker.key"
```

- Cliente:

```
gen_key          rsa_keysize=2048           filename="client.key"
```

– Crear las solicitudes de firma del certificado con **`cert_req`** y los siguientes argumentos:

- `filename` : Nombre del archivo de entrada con el par de claves de la entidad que solicita el certificado. Su clave pública es requerida.
- `output_file` : Nombre del archivo de salida con la solicitud de firma del certificado.
- `subject_name` : Nombre de la entidad (CN) que solicita el certificado, y opcionalmente su organización (O) y país (C).

Los comandos que se han ejecutado son los siguientes:

- Autoridad de certificación raíz: no hace falta crear una solicitud de firma para un certificado que estará autofirmado.

- Autoridad de certificación intermedia:

```
cert_req                                    filename="mqtt_ca.key"
    output_file="mqtt_ca.csr"           subject_name="CN=MQTT CA"
```

- Broker:

```
cert_req                                     filename="broker.key"
    output_file="broker.csr"           subject_name="CN=JORGE-PC"
```

- Cliente:

```
cert_req                                    filename="client.key"
    output_file="client.csr"     subject_name="CN=NUCLEO-H723ZG"
```

– Escribir los certificados con **`cert_write`** y los siguientes argumentos:

- `request_file` : Nombre del archivo de entrada con la solicitud de firma del certificado.
- `issuer_key` : Nombre del archivo de entrada con el par de claves de la autoridad de certificación. Su clave privada es requerida.
- `output_file` : Nombre del archivo de salida con el certificado.
- `issuer_name` : Nombre de la entidad (CN) que emite el certificado, y opcionalmente su organización (O) y país (C).
- `not_before` : Periodo de validez del certificado (desde).
- `not_after` : Periodo de validez del certificado (hasta).
- `selfsign` : Determina si el certificado se encuentra autofirmado.
- `is_ca` : Determina si el titular del certificado es una autoridad de certificación, o si de lo contrario es una entidad final.
- `max_pathlen` : Máximo número de certificados intermedios que puede haber entre el certificado creado y el certificado de una entidad final para que su cadena de confianza sea válida.

Los comandos que se han ejecutado son los siguientes:

- Autoridad de certificación raíz:

```
cert_write  request_file=""             issuer_key="root_ca.key"
    output_file="root_ca.crt"           issuer_name="CN=Root CA"
    not_before="20000101000000"       not_after="20291231235959"
    selfsign=1              is_ca=1               max_pathlen=1
```

- Autoridad de certificación intermedia:

```
cert_write  request_file="mqtt_ca.csr"  issuer_key="root_ca.key"
    output_file="mqtt_ca.crt"           issuer_name="CN=Root CA"
    not_before="20200101000000"       not_after="20241231235959"
    selfsign=0              is_ca=1               max_pathlen=0
```

- Broker:

```
cert_write  request_file="broker.csr"   issuer_key="mqtt_ca.key"
    output_file="broker.crt"            issuer_name="CN=MQTT CA"
    not_before="20210101000000"       not_after="20231231235959"
    selfsign=0              is_ca=0             max_pathlen=-1
```

- Cliente:

```
cert_write  request_file="client.csr"   issuer_key="mqtt_ca.key"
    output_file="client.crt"            issuer_name="CN=MQTT CA"
    not_before="20210101000000"       not_after="20231231235959"
    selfsign=0              is_ca=0             max_pathlen=-1
```

Todos los archivos creados con estos programas cuentan con un esquema de codificación PEM.

En el archivo **certificates.h** del proyecto de programación del sistema embebido se encuentran el par de claves del cliente, el certificado del cliente y el certificado raíz usados por el cliente TLS, y estos mismos archivos también han sido utilizados por el cliente de Eclipse Mosquitto. Para configurar el broker de Eclipse Mosquitto surge un problema, y es que en la opción `cafile` de su fichero de configuración únicamente se puede cargar un archivo, pero en esta infraestructura de clave pública es necesario incluir dos: el certificado raíz y el intermedio. Para poner solución a este problema, ambos certificados han de unirse en único archivo **ca.chain** con la siguiente estructura:

```
-----BEGIN CERTIFICATE-----
...
certificado intermedio
...
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
...
certificado raíz
...
-----END CERTIFICATE-----
```

Esta unión se puede hacer con cualquier editor de texto plano.

3.2.3. Internet

La aplicación creada también puede formar parte de una infraestructura de clave pública que contiene certificados de dispositivos que se encuentran en redes externas. Esto ocurre cuando se establece una conexión segura con el broker MQTT de Internet `test.mosquitto.org`.

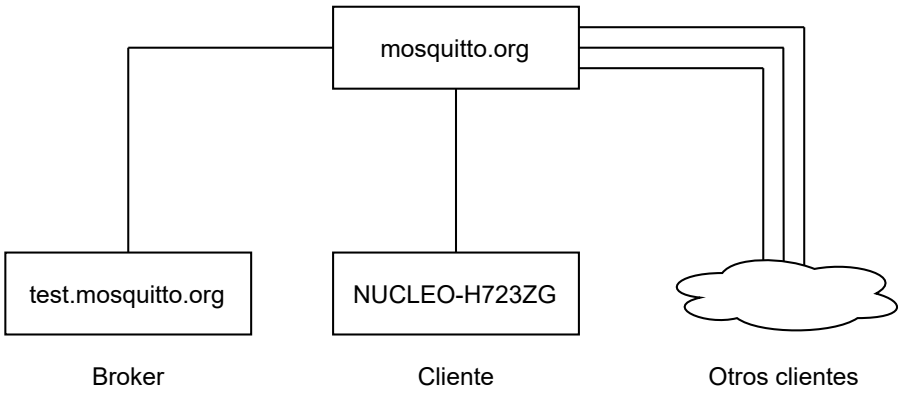

Figura XXII – Infraestructura de Internet

En este caso la infraestructura de certificados no cuenta con certificados intermedios, a diferencia de la de la intranet. Únicamente existe el certificado raíz, el del broker y los de los clientes. El certificado raíz ha de ser obtenido de la página web con el mismo hostname que el broker MQTT:

```
https://test.mosquitto.org
```

Para obtener el certificado del cliente, es necesario crear una nueva solicitud de firma del certificado debido a que la creada para la intranet solo incluye el **nombre común (CN)** entre la información del titular, y `test.mosquitto.org` requiere que también estén presentes la **organización (O)** y el **país (C)**. Pudiéndose usar el mismo par de claves que se ha creado para la intranet, ha de ejecutarse el siguiente comando:

```
cert_req        filename="client.key"        output_file="test.csr"
    subject_name="CN=NUCLEO-H723ZG,    O=TFG - Jorge Botana,    C=ES"
```

A continuación, se ha de enviar el contenido de la solicitud de firma del certificado desde el siguiente enlace:

```
https://test.mosquitto.org/ssl/
```

Obteniéndose el certificado del cliente, que ha sido renombrado como **test.crt**.

4

Aplicación

4.1. Funcionamiento

La aplicación creada hace uso de los clientes desarrollados por el autor de este documento.

UDP	DHCP	Establecer la dirección IP, máscara de subred y puerta de enlace
	DNS	Resolver los hostnames del servidor NTP y del broker MQTT
	NTP	Obtener la fecha y hora
TCP	TLS	Asegurar la conexión
	MQTT	Enviar y recibir mensajes

Tabla III – Usos de los protocolos de capa de aplicación implementados

La aplicación inicialmente establece la dirección IP, máscara de subred y puerta de enlace. Luego obtiene la fecha y hora. A continuación conecta con el broker MQTT y se suscribe a un tema secreto.

Tras esta inicialización, la aplicación publica periódicamente la fecha y hora en un tema público, mientras escucha los mensajes que puedan publicar otros clientes en el tema secreto. Al recibir un mensaje, la aplicación comprueba si es un comando para encender o apagar los LEDs de la placa de desarrollo y, de ser así, actúa en consecuencia y realiza una notificación publicando un mensaje en el tema público (por ejemplo, `"[LED 1 ENCENDIDO]"`). A continuación se muestran los comandos aceptados por la aplicación.

	LED 1	LED 2	LED 3
Encender	LD1ON	LD2ON	LD3ON
Apagar	LD1OFF	LD2OFF	LD3OFF

Tabla IV – Comandos aceptados por la aplicación

Se puede finalizar el programa accionando un pulsador (no el de reset) tras completar la inicialización de la aplicación, es decir, a partir del momento en el que la aplicación puede enviar y recibir mensajes MQTT.

La aplicación permite asegurar la conexión mediante TLS. Para ello ha de contarse con el certificado del CA raíz del broker y la fecha y hora correcta para poder verificar la cadena de confianza. Además, la aplicación también permite añadir un par de claves y certificado propios, para conexiones que requieran de autenticación del cliente por parte del broker. Esto último se encuentra deshabilitado por defecto.

Los detalles del funcionamiento de la aplicación se encuentran en los comentarios del archivo **main.c**.

4.2. Configuración

Para configurar la aplicación pueden modificarse los siguientes archivos:

- **settings.h**: Guarda los siguientes ajustes:
  - Nombre de la plataforma
  - Depuración por la UART
  - Intentos del cliente DHCP y configuración de red estática
  - Servidor DNS
  - Servidor NTP y zona horaria GMT
  - Uso de TLS, autenticación del cliente y depuración de TLS
  - Broker MQTT, ID del cliente, Keep Alive, tema público y tema secreto
  - Modo de pruebas, bytes por cada mensaje MQTT y ciclos totales
- **certificates.h**: Ha de contener lo siguiente:
  - Certificado raíz de confianza
  - Certificado del cliente
  - Par de claves del cliente

4.3. Depuración

La aplicación permite generar a través del puerto serie un log de funcionamiento. Para ello, ha de estar habilitada la depuración por la UART en el fichero de configuración general de la aplicación. Por defecto lo está. El otro extremo de la comunicación serial puede ser un PC, que usa un VCOM para realizar una conexión con un cable USB. En este PC ha de instalarse un terminal serie, por ejemplo **PuTTY** (versión usada: 0.73), en el cual se ha de establecer la siguiente configuración (y que es recomendable guardar para futuros usos):

```
- Session    -> Connection type : Serial
- Session    -> Logging -> All session output (e indicar la ruta)
- Terminal   -> Implicit CR in every LF
- Connection -> Serial  -> Serial line to connect to (la del PC)
- Connection -> Serial  -> Speed (baud) : 115200
- Connection -> Serial  -> Data bits    :      8
- Connection -> Serial  -> Stop bits    :      1
- Connection -> Serial  -> Parity       :   None
- Connection -> Serial  -> Flow control :   None
```

A continuación se muestra un log de uso de la aplicación, a modo de ejemplo de funcionamiento, y donde se requiere autenticación en ambos extremos de la comunicación.

```
------- MQTT/TLS demo v1.0 -------

- Autor        : Jorge Botana
- Plataforma   : NUCLEO-H723ZG
- Semilla      : 0286879759

- Inicializando RTOS .......... ok
-> FreeRTOS V10.4.4
-> CMSIS-RTOS2 wrapper

- Inicializando TCP/IP ........ ok
-> lwIP 2.1.2
-> MAC address : 00:C0:DF:24:BA:6B

- Esperando al cable LAN ...... ok

---------- Cliente DHCP ----------
```

```
- Buscando servidor DHCP ............. ok
- Configuración de red asignada por un servidor DHCP
-> IP address  : 192.168.1.101
-> Netmask     : 255.255.255.0
-> Gateway     : 192.168.1.1

---------- Cliente DNS -----------

- Preguntando a 8.8.8.8:53 ... ok
-> Hostname    : time.windows.com
-> IP address  : 20.101.57.9

---------- Cliente NTP -----------

- Preguntando a time.windows.com:123 ... ok
-> Date & Time : Fri 08-10-2021 - 14:26:13
-> Time zone   : GMT+2

---------- Cliente DNS -----------

- Preguntando a 8.8.8.8:53 ... ok
-> Hostname    : test.mosquitto.org
-> IP address  : 5.196.95.208

---------- Cliente MQTT ----------

- Conectando a test.mosquitto.org:8884 ... ok

---------- Cliente TLS -----------

- mbed TLS 2.16.11

- Inicializando los datos de la sesión TLS ...... ok
- Generando la semilla aleatoria ................ ok
- Cargando el certificado del CA de confianza ... ok
- Cargando el certificado del cliente ........... ok
- Cargando la clave privada del cliente ......... ok
- Creando la estructura TLS ..................... ok
- Realizando el handshake TLS ......................

- Cadena de confianza del servidor (depth = 1)
-> cert. version     : 3
-> serial number     : 05:8D:61:94:21:AF:76:3E:0D:84:15:E4:67:FB:8B:51:93:48:2C
   :0C
-> issuer name       : C=GB, ST=United Kingdom, L=Derby, O=Mosquitto, OU=CA, CN=
   mosquitto.org, emailAddress=roger@atchoo.org
-> subject name      : C=GB, ST=United Kingdom, L=Derby, O=Mosquitto, OU=CA, CN=
```

```
   mosquitto.org, emailAddress=roger@atchoo.org
-> issued  on        : 2020-06-09 11:06:39
-> expires on        : 2030-06-07 11:06:39
-> signed using      : RSA with SHA-256
-> RSA key size      : 2048 bits
-> basic constraints : CA=true
- Ok: verificación correcta

- Cadena de confianza del servidor (depth = 0)
-> cert. version     : 1
-> serial number     : 7D:D3:9B:4F:DC:5B:F7:2D:0F:0C:04:7E:B8:F3:23:9E:C1:9B:B7:
   B7
-> issuer name       : C=GB, ST=United Kingdom, L=Derby, O=Mosquitto, OU=CA, CN=
   mosquitto.org, emailAddress=roger@atchoo.org
-> subject name      : C=GB, ST=United Kingdom, L=Derby, O=Mosquitto, OU=Public
   server, CN=test.mosquitto.org
-> issued  on        : 2020-06-09 11:21:56
-> expires on        : 2030-06-06 11:21:56
-> signed using      : RSA with SHA-256
-> RSA key size      : 2048 bits
- Ok: verificación correcta

- Certificado del cliente
-> cert. version     : 3
-> serial number     : 00
-> issuer name       : C=GB, ST=United Kingdom, L=Derby, O=Mosquitto, OU=CA, CN=
   mosquitto.org, emailAddress=roger@atchoo.org
-> subject name      : CN=NUCLEO-H723ZG, O=TFG - Jorge Botana, C=ES
-> issued  on        : 2021-08-06 17:12:20
-> expires on        : 2021-11-04 17:12:20
-> signed using      : RSA with SHA-256
-> RSA key size      : 2048 bits
-> basic constraints : CA=false
-> key usage         : Digital Signature, Non Repudiation, Key Encipherment
- Ok: validado por el servidor

- Handshake TLS satisfactorio

- Enviando CONNECT ............ ok
- Esperando paquetes ............
- Recibido CONNACK
- Conexión aceptada

- Enviando SUBSCRIBE .......... ok
- Esperando paquetes ............
- Recibido SUBACK
- Suscripción aceptada : QoS 0
```

```
- Enviando PUBLISH ..... ok
-> [PLACA CONECTADA]

- Esperando paquetes .....
- Recibido PUBLISH
<- Hola XD

- Enviando PUBLISH ..... ok
-> Fri 08-10-2021 - 14:26:20

- Esperando paquetes .....
- Ningún paquete recibido
- Enviando PINGREQ ..... ok
- Esperando paquetes .....
- Recibido PINGRESP
- La conexión sigue activa

- Enviando PUBLISH ..... ok
-> Fri 08-10-2021 - 14:26:30

- Esperando paquetes .....
- Recibido PUBLISH
<- LD2OFF

- Enviando PUBLISH ..... ok
-> [LED 2 APAGADO]

- Esperando paquetes .....

- Botón pulsado !!!

- Enviando UNSUBSCRIBE ..... ok
- Esperando paquetes .....
- Recibido UNSUBACK
- Desuscripción aceptada

- Enviando PUBLISH ..... ok
-> [PLACA DESCONECTADA]

- Enviando DISCONNECT ..... ok
- Cerrando conexión ..... ok

------ Programa finalizado -------
```

En caso de error, el programa finaliza tras indicar qué ha fallado.

```
[...]

- Cable LAN desconectado !!!

------ Programa finalizado -------
```

La aplicación cuenta con una opción que permite incluir en el log todos los sucesos del stack de TLS usado. Esto es útil si se quiere desencriptar una sesión TLS, como se ha explicado en el segundo capítulo. Por defecto esta opción está deshabilitada.

4.4. Modo de pruebas

La aplicación cuenta con un modo de pruebas que permite medir tiempos de procesamiento de paquetes. Para acceder a él, ha de habilitarse en el fichero de configuración general de la aplicación y ha de estar presionado un pulsador (no el de reset) en el momento en el que se completa la inicialización de la aplicación, es decir, antes de que se publique el mensaje que indica que la placa ha sido conectada.

El modo de pruebas consiste en publicar en el tema secreto y recibir de él (al estar suscrito) un mensaje aleatorio de longitud predefinida, y repetir la operación un número de veces establecido. Se calculan y muestran los tiempos de procesamiento de cada iteración y los promedios. Tras terminar todos los ciclos, el programa finaliza mostrando los resultados. Por ello, es necesario que la depuración por el puerto serie esté habilitada en el fichero de configuración general de la aplicación, de lo contrario, no se podrán ver los resultados.

Es importante tomar en consideración que el modo de pruebas no tiene en cuenta el tiempo invertido durante el transporte de los paquetes. Es decir, se miden tiempos de lectura y escritura, y no de envío y recepción.

También ha de tenerse en cuenta que el uso de un temporizador añade una carga computacional importante que influye en los tiempos medidos. Es decir, si el modo de pruebas está deshabilitado, no se usará el temporizador y los tiempos serán menores.

El modo de pruebas es de especial interés para comparar los tiempos de procesamiento cuando se usa una conexión insegura y cuando se usa una conexión TLS, donde aumentan significativamente debido al coste computacional adicional. Esto se puede ver en el siguiente capítulo, donde se introduce el sistema embebido empleado (hasta ahora solo se ha mencionado) y se mide su rendimiento haciendo uso de dicho modo de pruebas.

5

Plataforma

5.1. Programación

La programación se ha realizado con **STM32CubeIDE**, un entorno de desarrollo integrado gratuito para microcontroladores STM32 basado en Eclipse CDT que cuenta con el toolchain de GNU. Integra el asistente de generación de código STM32CubeMX, aunque el proyecto de programación no ha sido creado sobre él.

5.2. Sistema embebido

La aplicación ha sido creada para una placa de desarrollo **NUCLEO-H723ZG**.

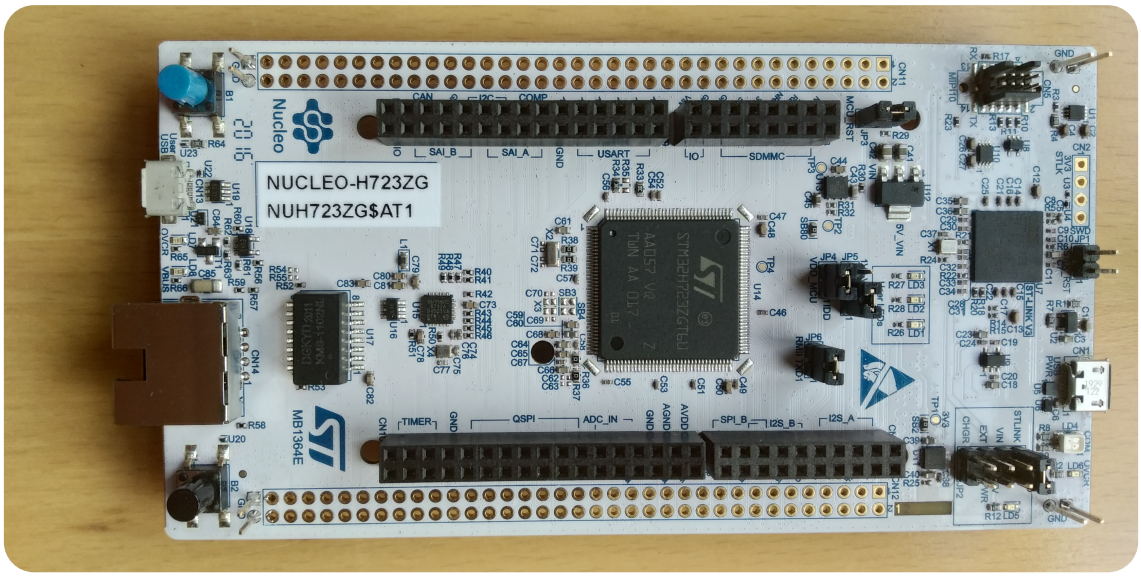

Figura XXIII – Placa de desarrollo NUCLEO-H723ZG

Esta placa de desarrollo cuenta con un microcontrolador **STM32H723ZGT6U**, cuyo core es un **ARM Cortex-M7**. Los siguientes periféricos están conectados a él:

- Un PHY de Ethernet LAN8742, conectado a través de una interfaz RMII
- Un módulo STLINK-V3E, conectado a través de un par de interfaces: SWD (programador y depurador) y UART (VCOM), con salida USB Micro-B
- Dos pulsadores: uno de reset (negro) y otro de usuario (azul)
- Tres LEDs de usuario
- Una interfaz con salida USB Micro-AB (no se usa en la aplicación)
- Dos conectores de expansión de GPIOs (no se usan en la aplicación)

Se pueden encontrar elementos adicionales en la placa de desarrollo, e información sobre ella en el manual de usuario **UM2407** y en los esquemáticos **MB1364** (diseñados con Altium Designer) de STMicroelectronics.

La aplicación requiere que el microcontrolador cuente con los siguientes periféricos:

- **ETH**: Ethernet, necesario al realizar comunicaciones TCP/IP mediante una conexión de red por cable LAN.
- **GPIOs**: Entradas y salidas de propósito general, necesarias para conectar al microcontrolador el PHY de Ethernet LAN8742, el módulo STLINK-V3E, los pulsadores y los LEDs de usuario.
- **RNG**: Generador de números aleatorios, requerido para que los valores secretos del Pre-Master Secret generados durante el handshake TLS sean completamente impredecibles, y que de esa forma la conexión pueda ser segura.
- **RTC**: Reloj de tiempo real en el que se ajusta la fecha y hora obtenida por NTP durante la inicialización de la aplicación.
- **TIMs**: Temporizadores. La aplicación requiere de uno para asignar un tiempo de ejecución entre tareas, otro para generar delays bloqueantes y uno adicional para medir tiempos en el modo de pruebas.
- **UART**: Receptor/transmisor asíncrono universal que permite visualizar en un terminal como PuTTY el log generado por la aplicación.

Para poder hacer uso de los periféricos anteriores, el microcontrolador cuenta con periféricos adicionales que no tienen relación directa con la aplicación y que algunos de ellos podrían no ser necesarios en otros microcontroladores. Se puede encontrar más información sobre los periféricos en la hoja de datos **DS13313** y en el manual de referencia **RM0468** de STMicroelectronics. Ha de puntualizarse que el microcontrolador usado no cuenta con aceleración de hardware para realizar operaciones criptográficas y de hash, sin embargo, no se ha echado en falta ya que en la parte de TLS estos cálculos se realizan por software.

En definitiva, esta placa de desarrollo de STMicroelectronics, al pertenecer a la serie NUCLEO, no ofrece grandes prestaciones, pero es de bajo coste (menos de 30 €) y lo que ofrece es suficiente para la aplicación creada. Las series DISCO y EVAL ofrecen placas de desarrollo de mayores prestaciones, pero con un coste más elevado.

5.3. Rendimiento

Para medir el rendimiento de la aplicación en esta plataforma, se ha usado el modo de pruebas expuesto en el capítulo anterior. A continuación se muestra un ejemplo muy sencillo con pocos mensajes MQTT y pocos bytes por cada uno de los mensajes.

```
------- MQTT/TLS demo v1.0 -------

- Autor        : Jorge Botana
- Plataforma   : NUCLEO-H723ZG
- Semilla      : 4061840850

- Inicializando RTOS .......... ok
-> FreeRTOS V10.4.4
-> CMSIS-RTOS2 wrapper

- Inicializando TCP/IP ........ ok
-> lwIP 2.1.2
-> MAC address : 00:C0:DF:24:BA:6B

- Esperando al cable LAN ...... ok

---------- Cliente DHCP ----------

- Buscando servidor DHCP ..... ok
- Configuración de red asignada por un servidor DHCP
-> IP address  : 192.168.1.101
-> Netmask     : 255.255.255.0
-> Gateway     : 192.168.1.1
```

```
---------- Cliente DNS -----------

- Preguntando a 192.168.1.100:53 ... ok
-> Hostname    : time.windows.com
-> IP address  : 51.145.123.29

---------- Cliente NTP -----------

- Preguntando a time.windows.com:123 ... ok
-> Date & Time : Sat 09-10-2021 - 00:08:49
-> Time zone   : GMT+2

---------- Cliente DNS -----------

- Preguntando a 192.168.1.100:53 ... ok
-> Hostname    : JORGE-PC
-> IP address  : 192.168.1.100

---------- Cliente MQTT ----------

- Conectando a JORGE-PC:8884 ... ok

---------- Cliente TLS -----------

- mbed TLS 2.16.11

- Inicializando los datos de la sesión TLS ..... ok
- Generando la semilla aleatoria ............... ok
- Cargando el certificado del CA de confianza ... ok
- Cargando el certificado del cliente .......... ok
- Cargando la clave privada del cliente ........ ok
- Creando la estructura TLS .................... ok
- Realizando el handshake TLS ..................

- Cadena de confianza del servidor (depth = 2)
-> cert. version     : 3
-> serial number     : 01
-> issuer name       : CN=Root CA
-> subject name      : CN=Root CA
-> issued  on        : 2000-01-01 00:00:00
-> expires on        : 2029-12-31 23:59:59
-> signed using      : RSA with SHA-256
-> RSA key size      : 2048 bits
-> basic constraints : CA=true, max_pathlen=1
- Ok: verificación correcta
```

```
- Cadena de confianza del servidor (depth = 1)
-> cert. version     : 3
-> serial number     : 01
-> issuer name       : CN=Root CA
-> subject name      : CN=MQTT CA
-> issued  on        : 2020-01-01 00:00:00
-> expires on        : 2024-12-31 23:59:59
-> signed using      : RSA with SHA-256
-> RSA key size      : 2048 bits
-> basic constraints : CA=true, max_pathlen=0
- Ok: verificación correcta

- Cadena de confianza del servidor (depth = 0)
-> cert. version     : 3
-> serial number     : 01
-> issuer name       : CN=MQTT CA
-> subject name      : CN=JORGE-PC
-> issued  on        : 2021-01-01 00:00:00
-> expires on        : 2023-12-31 23:59:59
-> signed using      : RSA with SHA-256
-> RSA key size      : 2048 bits
-> basic constraints : CA=false
- Ok: verificación correcta

- Certificado del cliente
-> cert. version     : 3
-> serial number     : 01
-> issuer name       : CN=MQTT CA
-> subject name      : CN=NUCLEO-H723ZG
-> issued  on        : 2021-01-01 00:00:00
-> expires on        : 2023-12-31 23:59:59
-> signed using      : RSA with SHA-256
-> RSA key size      : 2048 bits
-> basic constraints : CA=false
- Ok: validado por el servidor

- Handshake TLS satisfactorio

- Enviando CONNECT ............ ok
- Esperando paquetes ............
- Recibido CONNACK
- Conexión aceptada

- Enviando SUBSCRIBE .......... ok
- Esperando paquetes ............
- Recibido SUBACK
- Suscripción aceptada : QoS 0
```

```
- Enviando PUBLISH ............ ok
-> [PLACA CONECTADA]

-------- Modo de pruebas ---------

- Enviando PUBLISH ............ ok
-> H9"C/\}_-sZ%RdZ+Z"DO,,JIT{a-+9H:VQgTWyk1qf ,)Q^pdV@Ho+'J'\iAPJ]C

- Esperando paquetes ............
- Recibido PUBLISH
<- H9"C/\}_-sZ%RdZ+Z"DO,,JIT{a-+9H:VQgTWyk1qf ,)Q^pdV@Ho+'J'\iAPJ]C

- Ciclo 1/3
-> Escritura : 40 us
-> Lectura   : 55 us

- Promedios
-> Escritura : 40 us
-> Lectura   : 55 us

- Enviando PUBLISH ............ ok
-> @X~@*k6-+V?iZZQP)>e&.XtGX|A[rI[j{}f)J2BTY'gfEbuZ!,VN<(l[BwuS&eI'

- Esperando paquetes ............
- Recibido PUBLISH
<- @X~@*k6-+V?iZZQP)>e&.XtGX|A[rI[j{}f)J2BTY'gfEbuZ!,VN<(l[BwuS&eI'

- Ciclo 2/3
-> Escritura : 65 us
-> Lectura   : 50 us

- Promedios
-> Escritura : 52 us
-> Lectura   : 52 us

- Enviando PUBLISH ............ ok
-> hF59DCQ8M.1fx{X9;C|gB}{\>@6x%1Y,!pE>51(}}_pJo3;Cw/Oltj~o=j-1Td8Z

- Esperando paquetes ............
- Recibido PUBLISH
<- hF59DCQ8M.1fx{X9;C|gB}{\>@6x%1Y,!pE>51(}}_pJo3;Cw/Oltj~o=j-1Td8Z

- Ciclo 3/3
-> Escritura : 70 us
-> Lectura   : 50 us
```

```
- Promedios
-> Escritura : 58 us
-> Lectura   : 51 us

-------- Prueba terminada --------

- Conexión TLS
-> Broker    : JORGE-PC
-> Puerto    : 8884
-> bytes/msg : 64
-> Ciclos    : 3

- Resultados
-> Escritura : 58 us
-> Lectura   : 51 us

----------------------------------

- Enviando UNSUBSCRIBE ........ ok
- Esperando paquetes ............
- Recibido UNSUBACK
- Desuscripción aceptada

- Enviando PUBLISH ............ ok
-> [PLACA DESCONECTADA]

- Enviando DISCONNECT ......... ok
- Cerrando conexión ........... ok

------ Programa finalizado -------
```

Tras repetir este proceso para varios tamaños de mensaje MQTT con y sin TLS, se ha creado un gráfico con los resultados promedios de 100 iteraciones por cada prueba realizada. Los datos obtenidos siguen una relación lineal ($R^2 = 0,99$) ya que ningún paquete es dividido, al no superarse nunca el tamaño máximo de segmento TCP, de 1460 bytes en este caso. Se aprecia que:

- Añadir TLS incrementa los tiempos de forma significativa, debido al coste computacional adicional y a que el microcontrolador usado no cuenta con aceleración de hardware para operaciones criptográficas y de hash.

- La lectura es más rápida que la escritura (en este caso).

El gráfico creado se muestra en la siguiente página.

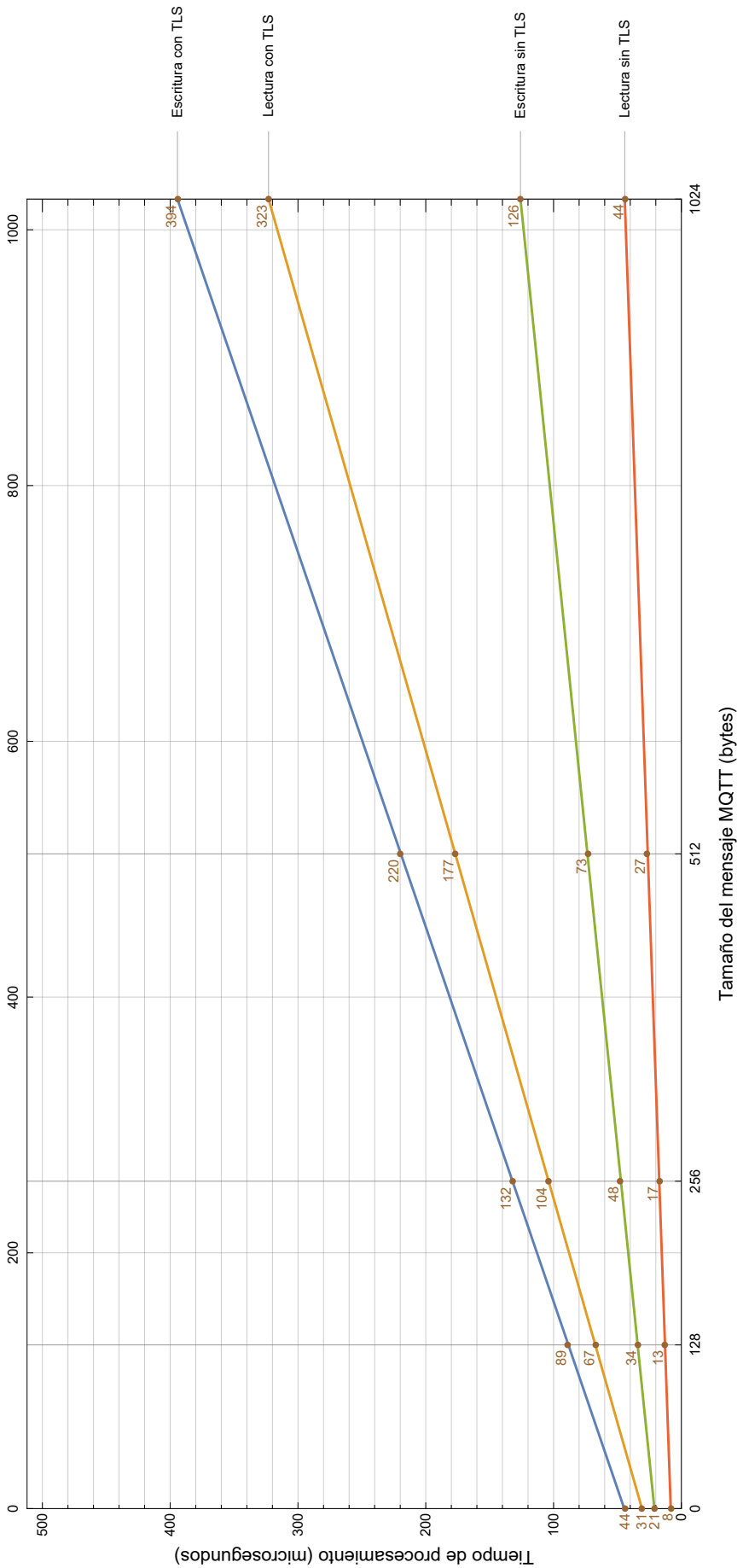

Figura XXIV – Comparativa del rendimiento con y sin TLS

6

Conclusiones y trabajo futuro

Del trabajo realizado se sacan algunas conclusiones:

- Los clientes desarrollados por el autor de este documento, al ser sencillos y fáciles de usar, constituyen una buena herramienta para el aprendizaje de protocolos.
- La capa segura TLS añade una carga computacional importante, no obstante, las prestaciones de los microcontroladores empleados permiten usarla sin problemas.
- Este documento puede ser útil para hacer trabajos relacionados con comunicaciones TCP/IP, infraestructuras de clave pública y/o microcontroladores STM32.

También existen una serie de tareas que han sido catalogadas como trabajo futuro:

- Desarrollar un cliente DHCP que no dependa de lwIP, para que pueda ser usado sobre cualquier stack de TCP/IP que disponga de la API de sockets BSD.
- Mejorar el cliente DNS para que pueda obtener varias direcciones IP por hostname.
- Actualizar el cliente TLS para que sea compatible con la versión 1.3 del protocolo. Para ello, se ha de esperar a que mbedTLS lo soporte.
- Añadir al cliente MQTT soporte para las calidades del servicio QoS 1 y QoS 2.
- Estudiar el uso de otros stacks de TCP/IP (por ejemplo, CycloneTCP) y de TLS (por ejemplo, WolfSSL) para sistemas embebidos.
- Crear un port para PC del proyecto de programación.

ANEXOS

A

Mapa de dependencias

A continuación se muestra el mapa de dependencias del proyecto de programación, en el que aparecen unos bloques con los nombres de los archivos de código fuente que el autor de este documento ha escrito o modificado, y unas flechas que indican las relaciones existentes entre dichos bloques.

El mapa de dependencias sigue un modelo jerárquico en el que se respeta una regla de diseño fundamental: si un bloque depende de otro, este último es independiente del primero.

Cada bloque representa un fichero fuente (.c), una cabecera (.h) o ambos. En este último caso, el fichero fuente siempre incluye a la cabecera. Cuando falta uno de los dos ficheros, es porque este no es necesario.

El origen de una flecha corresponde a un fichero fuente que incluye la cabecera del bloque hacia el que apunta la flecha. Si el bloque origen no cuenta con un fichero fuente, entonces el origen también es una cabecera.

En el mapa de dependencias también se indica qué bloques hacen uso de dependencias externas, es decir, librerías de terceros. No se han tenido en cuenta aquellas pertenecientes a la librería estándar.

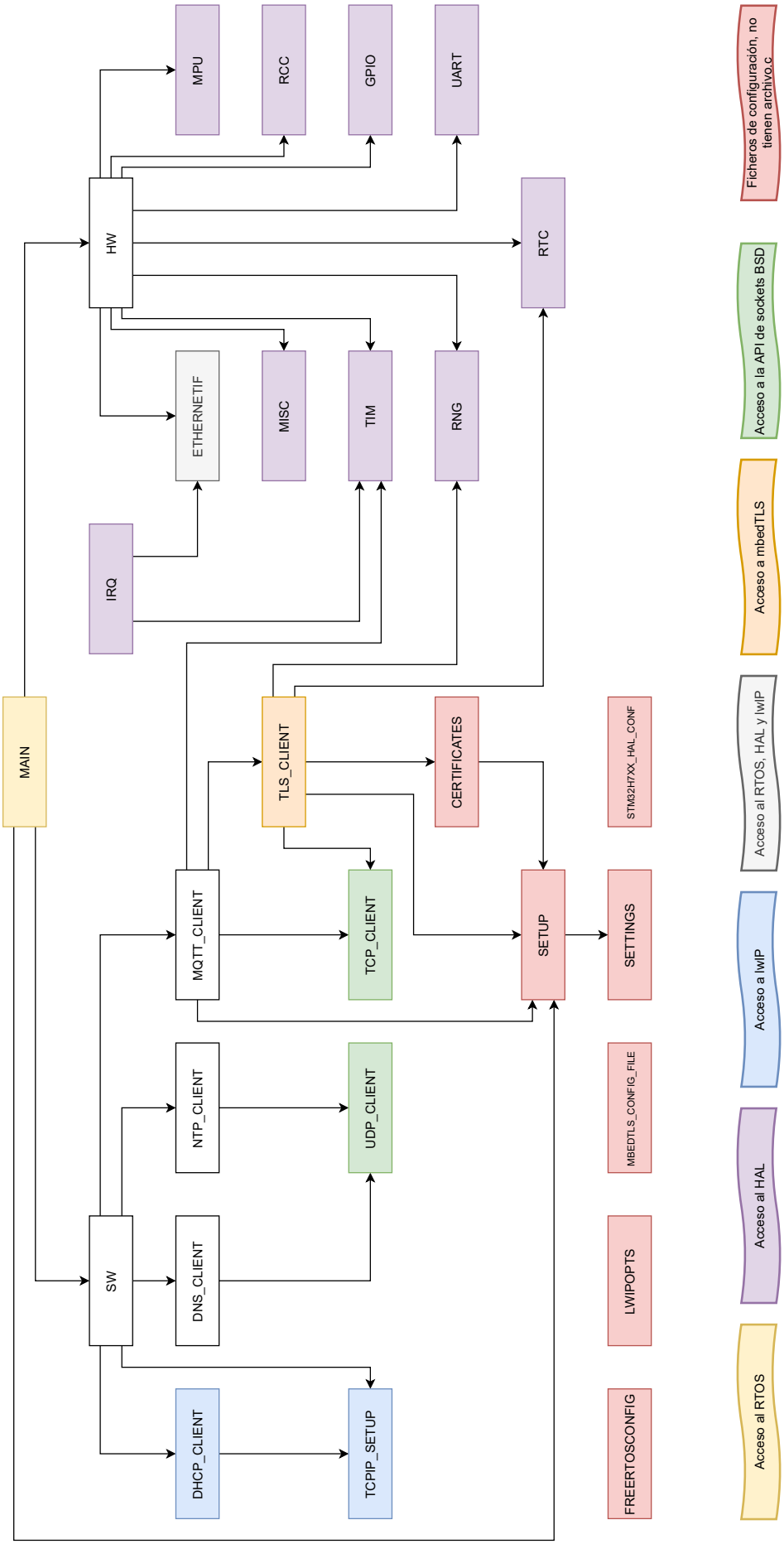

Figura XXV – Mapa de dependencias

B

Código fuente

B.1. Programa

B.1.1. MAIN

/NUCLEO_H723ZG_MQTT_TLS/Application/source/main.c

```
1  // ------------------------------------------------------------------------------
2  // MAIN
3  //
4  // - Programa que implementa los siguientes protocolos TCP/IP:
5  //
6  //   -> DHCP                          -> TLS
7  //   -> DNS                           -> MQTT
8  //   -> NTP
9  //
10 // - Los protocolos anteriores, al formar parte de la capa de aplicación,
11 //   funcionan sobre la capa de transporte, a la cual pertenecen los siguientes
12 //   protocolos:
13 //
14 //   -> UDP                           -> TCP
15 //
16 // - Se han creado clientes para los siete protocolos anteriores.
17 //
18 // - Los clientes de la capa de transporte se han programado con la API de
19 //   sockets BSD. Para hacer uso de ella con el stack de TCP/IP empleado, es
20 //   necesario un RTOS. Se tienen las siguientes tareas, ordenadas de arriba a
21 //   abajo de mayor a menor prioridad:
22 //
```

```
23 //   -> Tarea de Ethernet (en el archivo de usuario "eth.c").
24 //   -> Tarea de TCP/IP (en el archivo de lwIP "tcpip.c").
25 //   -> Tarea que mira a ver si se ha perdido la conexión LAN.
26 //   -> Tarea que mira a ver si el botón ha sido pulsado.
27 //   -> Tarea principal del programa.
28 //
29 // - Se proporciona un archivo de configuración del programa. Sus opciones más
30 //   significativas son aquellas que hacen referencia a la seguridad de la
31 //   comunicación MQTT, pudiéndose elegir entre:
32 //
33 //   -> Conexión insegura.
34 //   -> Conexión TLS con autenticación del servidor.
35 //   -> Conexión TLS con autenticación del servidor y del cliente.
36 //
37 // - El funcionamiento del programa se detalla a lo largo del mismo.
38 //
39 // ------------------------------------------------------------------------------
40 // Copyright (c) 2021 Jorge Botana Mtz. de Ibarreta
41 //
42 // Este archivo se encuentra bajo los términos de la Licencia MIT. Debería
43 // haberse proporcionado una copia de ella junto a este fichero. Si no es así, se
44 // puede encontrar en el siguiente enlace:
45 //
46 //                                          https://opensource.org/licenses/MIT
47 // ------------------------------------------------------------------------------
48
49
50 #include "hw.h"
51 #include "sw.h"
52 #include "setup.h"
53
54 #include "cmsis_os.h"
55
56 #include <stdio.h>
57 #include <stdlib.h>
58 #include <string.h>
59
60
61 #define   MAIN_THREAD_STACK_SIZE                  (6 * configMINIMAL_STACK_SIZE)
62 #define   LINK_THREAD_STACK_SIZE                  (1 * configMINIMAL_STACK_SIZE)
63 #define   BUTTON_THREAD_STACK_SIZE                (2 * configMINIMAL_STACK_SIZE)
64
65 #define   RECV_BUF_SIZE                                                    2048
66
67 #define   MSG_CONNECTED                                      "[PLACA CONECTADA]"
68 #define   MSG_DISCONNECTED                                "[PLACA DESCONECTADA]"
69
```

```
70  #define   MSG_LD1_ON                                          "[LED 1 ENCENDIDO]"
71  #define   MSG_LD1_OFF                                           "[LED 1 APAGADO]"
72  #define   MSG_LD2_ON                                          "[LED 2 ENCENDIDO]"
73  #define   MSG_LD2_OFF                                           "[LED 2 APAGADO]"
74  #define   MSG_LD3_ON                                          "[LED 3 ENCENDIDO]"
75  #define   MSG_LD3_OFF                                           "[LED 3 APAGADO]"
76
77  #define   FIRST_PRINTABLE_CHAR                                                 32
78  #define   LAST_PRINTABLE_CHAR                                                 126
79
80
81  osThreadAttr_t thread_attr  = {0};
82  osThreadId_t   main_handler = {0};
83
84  char mqtt_recv_buf[RECV_BUF_SIZE] = {0};
85
86  #ifdef    TEST_MODE
87  char mqtt_test_msg[TEST_MSG_BYTES + 1] = {0};
88  #endif // TEST_MODE
89
90
91  static void Main_Thread(void *arg);
92  static void Link_Thread(void *arg);
93  static void Button_Thread(void *arg);
94
95  #ifdef    TEST_MODE
96  static void Test_Mode(void);
97  #endif // TEST_MODE
98
99  static void Close_Program(void);
100 static void Error_Handler(void);
101
102
103 int main(void)
104 {
105     // Inicializar los periféricos.
106     Hw_Init();
107
108 #ifdef    UART_DEBUG
109     // Activar la UART.
110     Hw_UART_Start();
111 #endif // UART_DEBUG
112
113     // Mostrar el mensaje inicial.
114     printf("\n\n\n ------- MQTT/TLS demo v1.0 -------\n\n");
115
116     printf(" - Autor        : Jorge Botana\n");
```

```
117     printf(" - Plataforma   : %s\n", PLATFORM);
118
119     // Generar la semilla aleatoria para la función rand().
120     unsigned int seed = Hw_RNG_Get();
121     srand(seed);
122     printf(" - Semilla      : %010u\n\n", seed);
123
124     // Inicializar CMSIS-RTOS2, mostrando la versión de FreeRTOS.
125     printf(" - Inicializando RTOS ..........");
126     fflush(stdout);
127     osKernelInitialize();
128     printf(" ok\n");
129     printf(" -> FreeRTOS %s\n", tskKERNEL_VERSION_NUMBER);
130     printf(" -> CMSIS-RTOS2 wrapper\n\n");
131
132     // Crear la tarea principal.
133     thread_attr.stack_size = MAIN_THREAD_STACK_SIZE;
134     thread_attr.priority   = osPriorityBelowNormal;
135     main_handler = osThreadNew(Main_Thread, NULL, &thread_attr);
136
137     // Inicializar el scheduler.
138     osKernelStart();
139 }
140
141
142 static void Main_Thread(void *arg)
143 {
144     // Evitar "warnings" en el compilador debido a argumentos sin usar.
145     (void)arg;
146
147     // Encender el LED 1 tras inicializar el scheduler.
148     Hw_LD1_On();
149
150     // Inicializar el stack de TCP/IP.
151     Sw_TCPIP_Init();
152
153     // Bloquear el programa hasta que se detecte la conexión LAN.
154     printf(" - Esperando al cable LAN ......");
155     fflush(stdout);
156     while(Hw_ETH_Link_Status() < 0) {}
157     printf(" ok\n\n");
158
159     // Inicializar la tarea que comprueba periódicamente el estado de la conexión
160     // LAN, para terminar el programa si esta se pierde.
161     thread_attr.stack_size = LINK_THREAD_STACK_SIZE;
162     thread_attr.priority   = osPriorityAboveNormal;
163     osThreadNew(Link_Thread, NULL, &thread_attr);
```

```
164
165     // Intentar obtener la configuración de red por DHCP. Si se agotan los
166     // intentos, usar una configuración de red estática. La puerta de enlace será
167     // guardada en un buffer de 16 bytes (12 dígitos + 3 puntos + 1 NULL final),
168     // para poder usarla como servidor en las resoluciones DNS (si no se usa
169     // otro).
170     char gateway[16] = {0};
171     while(Sw_DHCP_Request(DHCP_ATTEMPTS, STATIC_IP, STATIC_NETMASK, STATIC_GW,
        gateway) < 0)
172     {
173         // Llamar a la función periódicamente cada 250 ms hasta que se establezca
174         // la configuración de red.
175         Hw_Tick_Delay(250);
176     }
177
178     // Encender el LED 2 tras establecer la configuración de red.
179     Hw_LD2_On();
180
181     // Obtener la dirección IPv4 del servidor NTP, que será guardada en un buffer
182     // de 16 bytes (12 dígitos + 3 puntos + 1 NULL final).
183     char ntp_ipaddr[16] = {0};
184     if(Sw_DNS_Request(DNS_SERVER, NTP_SERVER, sizeof(NTP_SERVER), ntp_ipaddr) <
        0)
185     {
186         Error_Handler();
187     }
188
189     // Obtener la fecha y hora del servidor NTP, guardando el timestamp.
190     time_t curtime = 0;
191     if(Sw_NTP_Request(NTP_SERVER, ntp_ipaddr, GMT_TIME, &curtime) < 0)
192     {
193         Error_Handler();
194     }
195
196     // Escribir la fecha y hora en el RTC.
197     Hw_RTC_Write(curtime);
198
199     // Obtener la dirección IPv4 del broker MQTT, que será guardada en un buffer
200     // de 16 bytes (12 dígitos + 3 puntos + 1 NULL final).
201     char mqtt_ipaddr[16] = {0};
202     if(Sw_DNS_Request(DNS_SERVER, MQTT_BROKER, sizeof(MQTT_BROKER), mqtt_ipaddr)
        < 0)
203     {
204         Error_Handler();
205     }
206
207     // Realizar el handshake de conexión TCP con el broker MQTT. Si la conexión
```

```
208     // es segura, también se realiza el handshake TLS.
209     if(Sw_MQTT_Init(MQTT_BROKER, mqtt_ipaddr, MQTT_PORT) < 0)
210     {
211         Error_Handler();
212     }
213 
214     // Conectarse al broker MQTT con el Client ID y los segundos de Keep Alive.
215     Sw_MQTT_Write_CONNECT(CLIENT_ID, sizeof(CLIENT_ID), KEEP_ALIVE);
216 
217     // Esperar a que el broker acepte la conexión.
218     if(Sw_MQTT_Read_Packet(mqtt_recv_buf, sizeof(mqtt_recv_buf)) < 0)
219     {
220         Error_Handler();
221     }
222 
223     // Suscribirse al tema secreto.
224     Sw_MQTT_Write_SUBSCRIBE(SECRET_TOPIC, sizeof(SECRET_TOPIC));
225 
226     // Esperar a que el broker acepte la suscripción.
227     if(Sw_MQTT_Read_Packet(mqtt_recv_buf, sizeof(mqtt_recv_buf)) < 0)
228     {
229         Error_Handler();
230     }
231 
232     // Publicar en el tema público que la placa ha sido conectada.
233     Sw_MQTT_Write_PUBLISH(PUBLIC_TOPIC, sizeof(PUBLIC_TOPIC), MSG_CONNECTED,
        sizeof(MSG_CONNECTED));
234 
235     // Encender el LED 3 tras finalizar la configuración inicial.
236     Hw_LD3_On();
237 
238 #ifdef    TEST_MODE
239     // Si el botón está pulsado ...
240     if(Hw_B1_Read() == 1)
241     {
242         // ... entrar en el modo de pruebas para medir tiempos de escritura y
243         // lectura de paquetes.
244         Test_Mode();
245     }
246 #endif // TEST_MODE
247 
248     // Inicializar la tarea que lee periódicamente el pulsador, para desconectar
249     // el cliente MQTT y terminar el programa cuando este sea pulsado.
250     thread_attr.stack_size = BUTTON_THREAD_STACK_SIZE;
251     thread_attr.priority   = osPriorityNormal;
252     osThreadNew(Button_Thread, NULL, &thread_attr);
253 
```

```
254     // Lazo sin fin. El programa publicará mensajes en el tema público y recibirá
255     // aquellos que otros clientes publiquen en el tema secreto. Usar una
256     // conexión TLS puede evitar que se filtre el tema secreto.
257     for( ;; )
258     {
259         // Esperar a paquetes entrantes. Si la operación de lectura devuelve un
260         // valor negativo, puede ser que:
261         // - No se ha recibido ningún paquete en 10 segundos.
262         // - Se ha perdido la conexión.
263         if(Sw_MQTT_Read_Packet(mqtt_recv_buf, sizeof(mqtt_recv_buf)) < 0)
264         {
265             // Enviar un PINGREQ al cual el broker tendrá que contestar con un
266             // PINGRESP, en caso de que la conexión siga activa.
267             Sw_MQTT_Write_PINGREQ();
268 
269             // Si la operación de lectura devuelve de nuevo un valor negativo ...
270             if(Sw_MQTT_Read_Packet(mqtt_recv_buf, sizeof(mqtt_recv_buf)) < 0)
271             {
272                 // ... se debe a que se ha perdido la conexión, y por ello se
273                 // finaliza el programa.
274                 printf("\n - Conexión perdida con el broker !!!\n\n");
275                 Error_Handler();
276             }
277         }
278 
279         // Comprobar si el paquete recibido contiene un comando secreto para
280         // encender o apagar los LEDs, y actuar en consecuencia. Una conexión TLS
281         // puede evitar que se filtren estos comandos secretos.
282         if      (strcmp(mqtt_recv_buf, "LD1ON" ) == 0)
283         {
284             Hw_LD1_On ();
285             Sw_MQTT_Write_PUBLISH(PUBLIC_TOPIC, sizeof(PUBLIC_TOPIC), MSG_LD1_ON
    , sizeof(MSG_LD1_ON ));
286         }
287         else if(strcmp(mqtt_recv_buf, "LD1OFF") == 0)
288         {
289             Hw_LD1_Off();
290             Sw_MQTT_Write_PUBLISH(PUBLIC_TOPIC, sizeof(PUBLIC_TOPIC), MSG_LD1_OFF
    , sizeof(MSG_LD1_OFF));
291         }
292         else if(strcmp(mqtt_recv_buf, "LD2ON" ) == 0)
293         {
294             Hw_LD2_On ();
295             Sw_MQTT_Write_PUBLISH(PUBLIC_TOPIC, sizeof(PUBLIC_TOPIC), MSG_LD2_ON
    , sizeof(MSG_LD2_ON ));
296         }
297         else if(strcmp(mqtt_recv_buf, "LD2OFF") == 0)
```

```
298         {
299             Hw_LD2_Off();
300             Sw_MQTT_Write_PUBLISH(PUBLIC_TOPIC, sizeof(PUBLIC_TOPIC), MSG_LD2_OFF
    , sizeof(MSG_LD2_OFF));
301         }
302         else if(strcmp(mqtt_recv_buf, "LD3ON" ) == 0)
303         {
304             Hw_LD3_On ();
305             Sw_MQTT_Write_PUBLISH(PUBLIC_TOPIC, sizeof(PUBLIC_TOPIC), MSG_LD3_ON
    , sizeof(MSG_LD3_ON ));
306         }
307         else if(strcmp(mqtt_recv_buf, "LD3OFF") == 0)
308         {
309             Hw_LD3_Off();
310             Sw_MQTT_Write_PUBLISH(PUBLIC_TOPIC, sizeof(PUBLIC_TOPIC), MSG_LD3_OFF
    , sizeof(MSG_LD3_OFF));
311         }
312         else
313         {
314             // Leer la fecha y hora del RTC.
315             curtime = Hw_RTC_Read();
316 
317             // Publicar en el tema público la fecha y hora si el paquete recibido
318             // no contenía un comando secreto.
319             char buf[30] = {0};
320             Sw_MQTT_Write_PUBLISH(PUBLIC_TOPIC, sizeof(PUBLIC_TOPIC), buf,
    strftime(buf, sizeof(buf), "%a %d-%m-%Y - %H:%M:%S", localtime(&curtime)) +
    1);
321         }
322     }
323 }
324 
325 
326 static void Link_Thread(void *arg)
327 {
328     // Evitar "warnings" en el compilador debido a argumentos sin usar.
329     (void)arg;
330 
331     for( ;; )
332     {
333         // Mirar a ver si se ha perdido la conexión LAN.
334         if(Hw_ETH_Link_Status() < 0)
335         {
336 #ifdef    UART_DEBUG
337             // Reiniciar la UART para evitar que deje de funcionar si se hubiese
338             // interrumpido un printf() al cambiar de tarea.
339             Hw_UART_Stop ();
```

```
340             Hw_UART_Start();
341 #endif // UART_DEBUG
342 
343             // Finalizar el programa porque se ha perdido la conexión LAN.
344             printf("\n - Cable LAN desconectado !!!\n\n");
345             Error_Handler();
346         }
347 
348         // Suspender la tarea durante 100 ms.
349         osDelay(100);
350     }
351 }
352 
353 
354 static void Button_Thread(void *arg)
355 {
356     // Evitar "warnings" en el compilador debido a argumentos sin usar.
357     (void)arg;
358 
359     for( ;; )
360     {
361         // Mirar a ver si el botón ha sido pulsado.
362         if(Hw_B1_Read() == 1)
363         {
364 #ifdef    UART_DEBUG
365             // Reiniciar la UART para evitar que deje de funcionar si se hubiese
366             // interrumpido un printf() al cambiar de tarea.
367             Hw_UART_Stop ();
368             Hw_UART_Start();
369 #endif // UART_DEBUG
370 
371             // Terminar la tarea principal ya que, de no hacerlo, seguiría
372             // ejecutándose cuando la tarea del botón se suspende durante una
373             // operación de lectura en el proceso de desconexión TCP.
374             osThreadTerminate(main_handler);
375 
376             // Desconectarse del broker y finalizar el programa.
377             printf("\n - Botón pulsado !!!\n\n");
378             Close_Program();
379         }
380 
381         // Suspender la tarea durante 50 ms.
382         osDelay(50);
383     }
384 }
385 
386 
```

```
387 #ifdef    TEST_MODE
388
389 static void Test_Mode(void)
390 {
391     // Modo de pruebas para medir tiempos de escritura y lectura de paquetes (con
392     // o sin TLS). No se mide el tiempo invertido durante el transporte de estos,
393     // ni lo que tarda el broker en responder.
394
395     // Se emplea un temporizador: el cliente MQTT lo inicia y lo para cuando
396     // corresponda, para ser leído desde esta función.
397
398     // Para medir los tiempos de escritura y lectura, se publica un mensaje
399     // aleatorio en el tema secreto, al cual se está suscrito, para recibir una
400     // respuesta inmediata del broker con ese mismo mensaje.
401
402     // Repetir el ciclo el número de veces establecido y obtener los tiempos
403     // promedios.
404
405     printf(" -------- Modo de pruebas ---------\n\n");
406
407     // -> n = Número de mediciones
408     int n       = 0;
409
410     // -> i = Medida actual
411     int i_write = 0;
412     int i_read  = 0;
413
414     // -> s = Valor acumulado
415     int s_write = 0;
416     int s_read  = 0;
417
418     // -> x = Media aritmética = s / n
419     int x_write = 0;
420     int x_read  = 0;
421
422     for( ;; )
423     {
424         // Incrementar el número de la medición actual.
425         n++;
426
427         // Generar un mensaje aleatorio con caracteres imprimibles del código
428         // ASCII original (7 bits). El último byte ha de ser un NULL (que luego
429         // el cliente MQTT eliminará).
430         int i = 0;
431         for(i = 0; i < (sizeof(mqtt_test_msg) - 1); i++)
432         {
433             do
```

```
434             {
435                 mqtt_test_msg[i] = rand();
436             }
437             while((mqtt_test_msg[i] < FIRST_PRINTABLE_CHAR) || \
438                   (mqtt_test_msg[i] >  LAST_PRINTABLE_CHAR));
439         }
440         mqtt_test_msg[i] = 0;
441 
442         // Medir el tiempo de escritura.
443         Sw_MQTT_Write_PUBLISH(SECRET_TOPIC, sizeof(SECRET_TOPIC), mqtt_test_msg,
    sizeof(mqtt_test_msg));
444 
445         // Actualizar las estadísticas de escritura.
446         i_write = Hw_TIM_Read();
447         s_write = s_write + i_write;
448         x_write = s_write / n;
449 
450         // Dar tiempo a que llegue la respuesta, ya que solo se está midiendo el
451         // tiempo invertido en escribir y leer los paquetes, y no el tiempo
452         // invertido durante el transporte de estos.
453         Hw_Tick_Delay(1000);
454 
455         // Medir el tiempo de lectura. Si la operación de lectura devuelve un
456         // valor negativo, es porque ha habido un error de conexión, ya que el
457         // broker debe devolver el mensaje que fue publicado en el mismo tema al
458         // que se está suscrito.
459         if(Sw_MQTT_Read_Packet(mqtt_recv_buf, sizeof(mqtt_recv_buf)) < 0)
460         {
461             // Interrumpir la prueba y finalizar el programa, ya que se ha
462             // perdido la conexión.
463             printf("\n - Conexión perdida con el broker !!!\n\n");
464             Error_Handler();
465         }
466 
467         // Actualizar las estadísticas de lectura.
468         i_read = Hw_TIM_Read();
469         s_read = s_read + i_read;
470         x_read = s_read / n;
471 
472         // Mostrar los tiempos de la medición actual.
473         printf(" - Ciclo %i/%i\n", n, TEST_N_CYCLES);
474         printf(" -> Escritura : %i us\n"  , i_write);
475         printf(" -> Lectura   : %i us\n\n", i_read );
476 
477         // Mostrar los tiempos promedios.
478         printf(" - Promedios\n");
479         printf(" -> Escritura : %i us\n"  , x_write);
```

```
480         printf(" -> Lectura   : %i us\n\n", x_read );
481
482         // Realizar el ciclo el número de veces especificado y luego terminar la
483         // prueba.
484         if(n < TEST_N_CYCLES)
485         {
486             // Esperar un segundo antes del siguiente ciclo.
487             Hw_Tick_Delay(1000);
488         }
489         else
490         {
491             // Mostrar los resultados de la prueba.
492             printf(" -------- Prueba terminada --------\n\n");
493
494 #ifdef    USE_TLS
495             printf(" - Conexión TLS\n"     );
496 #else  // USE_TLS
497             printf(" - Conexión insegura\n");
498 #endif // USE_TLS
499             printf(" -> Broker    : %s\n"  , MQTT_BROKER   );
500             printf(" -> Puerto    : %i\n"  , MQTT_PORT     );
501             printf(" -> bytes/msg : %i\n"  , TEST_MSG_BYTES);
502             printf(" -> Ciclos    : %i\n\n", TEST_N_CYCLES );
503
504             printf(" - Resultados\n");
505             printf(" -> Escritura : %i us\n"  , x_write);
506             printf(" -> Lectura   : %i us\n\n", x_read );
507
508             // Desconectarse del broker y finalizar el programa.
509             printf(" ----------------------------------\n\n");
510
511             Close_Program();
512         }
513     }
514 }
515
516 #endif // TEST_MODE
517
518
519 static void Close_Program(void)
520 {
521     // Desuscribirse del tema secreto.
522     Sw_MQTT_Write_UNSUBSCRIBE(SECRET_TOPIC, sizeof(SECRET_TOPIC));
523
524     // Esperar a que el broker acepte la desuscripción.
525     if(Sw_MQTT_Read_Packet(mqtt_recv_buf, sizeof(mqtt_recv_buf)) < 0)
526     {
```

```
527         Error_Handler();
528     }
529 
530     // Publicar en el tema público que la placa va a ser desconectada.
531     Sw_MQTT_Write_PUBLISH(PUBLIC_TOPIC, sizeof(PUBLIC_TOPIC), MSG_DISCONNECTED,
        sizeof(MSG_DISCONNECTED));
532 
533     // Notificar al broker que se va cerrar la conexión.
534     Sw_MQTT_Write_DISCONNECT();
535 
536     // Cerrar la conexión.
537     Sw_MQTT_Close(mqtt_recv_buf, sizeof(mqtt_recv_buf));
538 
539     // Finalizar el programa.
540     Error_Handler();
541 }
542 
543 
544 static void Error_Handler(void)
545 {
546     // Bloquear el RTOS.
547     osKernelLock();
548 
549     // Apagar los LEDs e indicar que el programa ha finalizado.
550     Hw_LD1_Off();
551     Hw_LD2_Off();
552     Hw_LD3_Off();
553     printf(" ------ Programa finalizado -------\n");
554 
555     // Bloquear aquí.
556     for( ;; );
557 }
```

B.1.2. SW

/NUCLEO_H723ZG_MQTT_TLS/Application/source/sw.c

```
1 // ------------------------------------------------------------------------------
2 // SW
3 //
4 // - API que contiene funciones que hacen uso de TCP/IP.
5 //
6 // - Las cadenas de texto han de incluir el NULL final.
7 //
8 // - Los tamaños han de contar el NULL final.
9 //
```

```
10 // ---------------------------------------------------------------------------
11 // Copyright (c) 2021 Jorge Botana Mtz. de Ibarreta
12 //
13 // Este archivo se encuentra bajo los términos de la Licencia MIT. Debería
14 // haberse proporcionado una copia de ella junto a este fichero. Si no es así, se
15 // puede encontrar en el siguiente enlace:
16 //
17 //                                          https://opensource.org/licenses/MIT
18 // ---------------------------------------------------------------------------
19
20
21 #include "sw.h"
22
23 #include "tcpip_setup.h"
24 #include "dhcp_client.h"
25 #include "dns_client.h"
26 #include "ntp_client.h"
27 #include "mqtt_client.h"
28
29
30 // ---------------------------------------------------------------------------
31 // Inicializa el stack de TCP/IP.
32 //
33 // -> (void    ) Nada.
34 //
35 // <- (void    ) Nada.
36 // ---------------------------------------------------------------------------
37 void Sw_TCPIP_Init(void)
38 {
39     TCPIP_Init();
40 }
41
42
43 // ---------------------------------------------------------------------------
44 // Establece la configuración de red por medio de un cliente DHCP. Se debe
45 // ejecutar periódicamente hasta que esto ocurra.
46 //
47 // -> (int     ) Número máximo de intentos del cliente DHCP hasta establecer una
48 //               configuración de red estática.
49 //
50 // -> (char   *) Dirección IPv4 estática usada en caso de que NO se obtenga la
51 //               configuración de red por DHCP.
52 //
53 // -> (char   *) Máscara de subred estática usada en caso de que NO se obtenga la
54 //               configuración de red por DHCP.
55 //
56 // -> (char   *) Puerta de enlace estática usada en caso de que NO se obtenga la
```

```
57  //               configuración de red por DHCP.
58  //
59  // -> (char   *) Buffer donde se guardará la puerta de enlace.
60  //
61  // <- (int     ) 0 en caso de que se haya establecido la configuración de red (ya
62  //               sea por DHCP o de forma estática tras agotar el número de
63  //               intentos), un valor negativo en caso de que aún se esté en
64  //               proceso de establecer la configuración de red.
65  // ----------------------------------------------------------------------------
66  int Sw_DHCP_Request(const int dhcp_attempts, const char *static_ip, const char *
        static_netmask, const char *static_gw, char *gateway)
67  {
68      return DHCP_Request(dhcp_attempts, static_ip, static_netmask, static_gw,
        gateway);
69  }
70
71
72  // ----------------------------------------------------------------------------
73  // Obtiene la dirección IPv4 correspondiente a un hostname mediante una
74  // resolución DNS.
75  //
76  // -> (char   *) Dirección IPv4 del servidor DNS.
77  //
78  // -> (char   *) Hostname que se quiere resolver.
79  //
80  // -> (size_t  ) Tamaño del hostname.
81  //
82  // -> (char   *) Buffer donde se guardará la dirección IPv4 del hostname.
83  //
84  // <- (int     ) 0 si NO hay error, un valor negativo en caso contrario.
85  // ----------------------------------------------------------------------------
86  int Sw_DNS_Request(const char *server, const char *hostname, size_t len, char *
        ipaddr)
87  {
88      TCPIP_Down_Up();
89
90      return DNS_Request(server, hostname, len, ipaddr);
91  }
92
93
94  // ----------------------------------------------------------------------------
95  // Obtiene la fecha y hora conectando a un servidor NTP.
96  //
97  // -> (char   *) Hostname del servidor NTP.
98  //
99  // -> (char   *) Dirección IPv4 del servidor NTP.
100 //
```

```
101 // -> (int     ) Zona horaria GMT.
102 //
103 // -> (time_t *) Buffer donde se guardará el timestamp.
104 //
105 // <- (int     ) 0 si NO hay error, un valor negativo en caso contrario.
106 // ------------------------------------------------------------------------
107 int Sw_NTP_Request(const char *hostname, const char *ipaddr, const int gmt_time,
        time_t *curtime)
108 {
109     TCPIP_Down_Up();
110 
111     return NTP_Request(hostname, ipaddr, gmt_time, curtime);
112 }
113 
114 
115 // ------------------------------------------------------------------------
116 // Conecta a un broker MQTT.
117 //
118 // -> (char   *) Hostname del broker MQTT.
119 //
120 // -> (char   *) Dirección IPv4 del broker MQTT.
121 //
122 // -> (int     ) Puerto usado por el broker MQTT.
123 //
124 // <- (int     ) 0 si NO hay error, un valor negativo en caso contrario.
125 // ------------------------------------------------------------------------
126 int Sw_MQTT_Init(const char *hostname, const char *ipaddr, const int port)
127 {
128     TCPIP_Down_Up();
129 
130     return MQTT_Init(hostname, ipaddr, port);
131 }
132 
133 
134 // ------------------------------------------------------------------------
135 // Desconecta de un broker MQTT.
136 //
137 // -> (char   *) Buffer de recepción.
138 //
139 // -> (size_t  ) Tamaño del buffer de recepción.
140 //
141 // <- (int     ) 0 si NO hay error, un valor negativo en caso contrario.
142 // ------------------------------------------------------------------------
143 int Sw_MQTT_Close(char *buf, size_t len)
144 {
145     return MQTT_Close(buf, len);
146 }
```

```
147
148
149 // ----------------------------------------------------------------------------
150 // Espera a paquetes MQTT entrantes. Bloquea hasta que se recibe un paquete o
151 // hasta el timeout. Procesa el paquete recibido, si lo hay.
152 //
153 // -> (char   *) Buffer de recepción.
154 //
155 // -> (size_t  ) Tamaño del buffer de recepción.
156 //
157 // <- (int     ) Número de bytes leídos o, en caso de error o timeout, un valor
158 //               negativo.
159 // ----------------------------------------------------------------------------
160 int Sw_MQTT_Read_Packet(char *buf, size_t len)
161 {
162     return MQTT_Read_Packet(buf, len);
163 }
164
165
166 // ----------------------------------------------------------------------------
167 // Envía un paquete CONNECT con el Client ID y los segundos de Keep Alive.
168 //
169 // -> (char   *) Client ID.
170 //
171 // -> (size_t  ) Tamaño del Client ID.
172 //
173 // -> (int     ) Segundos de Keep Alive.
174 //
175 // <- (int     ) Número de bytes escritos o, en caso de error, un valor negativo.
176 // ----------------------------------------------------------------------------
177 int Sw_MQTT_Write_CONNECT(const char *client, size_t c_len, const int keep_alive)
178 {
179     return MQTT_Write_CONNECT(client, c_len, keep_alive);
180 }
181
182
183 // ----------------------------------------------------------------------------
184 // Envía un paquete DISCONNECT.
185 //
186 // -> (void    ) Nada.
187 //
188 // <- (int     ) Número de bytes escritos o, en caso de error, un valor negativo.
189 // ----------------------------------------------------------------------------
190 int Sw_MQTT_Write_DISCONNECT(void)
191 {
192     return MQTT_Write_DISCONNECT();
193 }
```

```
194
195
196 // ----------------------------------------------------------------------------
197 // Envía un paquete PUBLISH con un tema y un mensaje.
198 //
199 // -> (char   *) Tema.
200 //
201 // -> (size_t  ) Tamaño del tema.
202 //
203 // -> (char   *) Mensaje.
204 //
205 // -> (size_t  ) Tamaño del mensaje.
206 //
207 // <- (int     ) Número de bytes escritos o, en caso de error, un valor negativo.
208 // ----------------------------------------------------------------------------
209 int Sw_MQTT_Write_PUBLISH(const char *topic, size_t t_len, const char *msg,
        size_t m_len)
210 {
211     return MQTT_Write_PUBLISH(topic, t_len, msg, m_len);
212 }
213
214
215 // ----------------------------------------------------------------------------
216 // Envía un paquete SUBSCRIBE con un tema.
217 //
218 // -> (char   *) Tema.
219 //
220 // -> (size_t  ) Tamaño del tema.
221 //
222 // <- (int     ) Número de bytes escritos o, en caso de error, un valor negativo.
223 // ----------------------------------------------------------------------------
224 int Sw_MQTT_Write_SUBSCRIBE(const char *topic, size_t t_len)
225 {
226     return MQTT_Write_SUBSCRIBE(topic, t_len);
227 }
228
229
230 // ----------------------------------------------------------------------------
231 // Envía un paquete UNSUBSCRIBE con un tema.
232 //
233 // -> (char   *) Tema.
234 //
235 // -> (size_t  ) Tamaño del tema.
236 //
237 // <- (int     ) Número de bytes escritos o, en caso de error, un valor negativo.
238 // ----------------------------------------------------------------------------
239 int Sw_MQTT_Write_UNSUBSCRIBE(const char *topic, size_t t_len)
```

```
240 {
241     return MQTT_Write_UNSUBSCRIBE(topic, t_len);
242 }
243 
244 
245 // ----------------------------------------------------------------------------
246 // Envía un paquete PINGREQ.
247 //
248 // -> (void    ) Nada.
249 //
250 // <- (int     ) Número de bytes escritos o, en caso de error, un valor negativo.
251 // ----------------------------------------------------------------------------
252 int Sw_MQTT_Write_PINGREQ(void)
253 {
254     return MQTT_Write_PINGREQ();
255 }
```

/NUCLEO_H723ZG_MQTT_TLS/Application/include/sw.h

```
1  // ----------------------------------------------------------------------------
2  // SW
3  //
4  // - Archivo de cabecera.
5  //
6  // ----------------------------------------------------------------------------
7  // Copyright (c) 2021 Jorge Botana Mtz. de Ibarreta
8  //
9  // Este archivo se encuentra bajo los términos de la Licencia MIT. Debería
10 // haberse proporcionado una copia de ella junto a este fichero. Si no es así, se
11 // puede encontrar en el siguiente enlace:
12 //
13 //                                          https://opensource.org/licenses/MIT
14 // ----------------------------------------------------------------------------
15 
16 #ifndef   SW_H
17 #define   SW_H
18 
19 #include <time.h>
20 
21 void Sw_TCPIP_Init(void);
22 
23 int Sw_DHCP_Request(const int dhcp_attempts, const char *static_ip, const char *
       static_netmask, const char *static_gw, char *gateway);
24 
25 int Sw_DNS_Request(const char *server, const char *hostname, size_t len, char *
       ipaddr);
26 
```

```
27 int Sw_NTP_Request(const char *hostname, const char *ipaddr, const int gmt_time,
       time_t *curtime);
28
29 int Sw_MQTT_Init(const char *hostname, const char *ipaddr, const int port);
30 int Sw_MQTT_Close(char *buf, size_t len);
31
32 int Sw_MQTT_Read_Packet(char *buf, size_t len);
33
34 int Sw_MQTT_Write_CONNECT(const char *client, size_t c_len, const int keep_alive)
       ;
35 int Sw_MQTT_Write_DISCONNECT(void);
36 int Sw_MQTT_Write_PUBLISH(const char *topic, size_t t_len, const char *msg,
       size_t m_len);
37 int Sw_MQTT_Write_SUBSCRIBE(const char *topic, size_t t_len);
38 int Sw_MQTT_Write_UNSUBSCRIBE(const char *topic, size_t t_len);
39 int Sw_MQTT_Write_PINGREQ(void);
40
41 #endif // SW_H
```

B.1.3. HW

/NUCLEO_H723ZG_MQTT_TLS/Application/source/hw.c

```
1  // ------------------------------------------------------------------------------
2  // HW
3  //
4  // - API que contiene funciones que hacen uso de periféricos.
5  //
6  // ------------------------------------------------------------------------------
7  // Copyright (c) 2021 Jorge Botana Mtz. de Ibarreta
8  //
9  // Este archivo se encuentra bajo los términos de la Licencia MIT. Debería
10 // haberse proporcionado una copia de ella junto a este fichero. Si no es así, se
11 // puede encontrar en el siguiente enlace:
12 //
13 //                                          https://opensource.org/licenses/MIT
14 // ------------------------------------------------------------------------------
15
16
17 #include "hw.h"
18
19 #include "mpu.h"
20 #include "misc.h"
21 #include "rcc.h"
22 #include "gpio.h"
23 #include "rng.h"
```

```
24 #include "rtc.h"
25 #include "uart.h"
26 #include "tim.h"
27 #include "ethernetif.h"
28
29
30 // ----------------------------------------------------------------------------
31 // Inicializa todos los periféricos usados excepto Ethernet, ya que este es
32 // inicializado por el stack de TCP/IP.
33 //
34 // -> (void    ) Nada.
35 //
36 // <- (void    ) Nada.
37 // ----------------------------------------------------------------------------
38 void Hw_Init(void)
39 {
40     MPU_Init  ();
41     CACHE_Init();
42     SYS_Init  ();
43     RCC_Init  ();
44     GPIO_Init ();
45     RNG_Init  ();
46     RTC_Init  ();
47     UART_Init ();
48     TIM_Init  ();
49 }
50
51
52 // ----------------------------------------------------------------------------
53 // Bloquea durante el tiempo especificado.
54 //
55 // -> (void    ) Número de milisegundos.
56 //
57 // <- (void    ) Nada.
58 // ----------------------------------------------------------------------------
59 void Hw_Tick_Delay(unsigned int delay_ms)
60 {
61     Tick_Delay(delay_ms);
62 }
63
64
65 // ----------------------------------------------------------------------------
66 // Lee el pulsador.
67 //
68 // -> (void    ) Nada.
69 //
70 // <- (int     ) 1 si está pulsado, 0 si no lo está.
```

```
71  // ------------------------------------------------------------------------
72  int Hw_B1_Read(void)
73  {
74      return B1_Read();
75  }
76
77
78  // ------------------------------------------------------------------------
79  // Enciende el LED 1.
80  //
81  // -> (void    ) Nada.
82  //
83  // <- (void    ) Nada.
84  // ------------------------------------------------------------------------
85  void Hw_LD1_On(void)
86  {
87      LD1_On();
88  }
89
90
91  // ------------------------------------------------------------------------
92  // Apaga el LED 1.
93  //
94  // -> (void    ) Nada.
95  //
96  // <- (void    ) Nada.
97  // ------------------------------------------------------------------------
98  void Hw_LD1_Off(void)
99  {
100     LD1_Off();
101 }
102
103
104 // ------------------------------------------------------------------------
105 // Enciende el LED 2.
106 //
107 // -> (void    ) Nada.
108 //
109 // <- (void    ) Nada.
110 // ------------------------------------------------------------------------
111 void Hw_LD2_On(void)
112 {
113     LD2_On();
114 }
115
116
117 // ------------------------------------------------------------------------
```

```
118 // Apaga el LED 2.
119 //
120 // -> (void    ) Nada.
121 //
122 // <- (void    ) Nada.
123 // ------------------------------------------------------------------------
124 void Hw_LD2_Off(void)
125 {
126     LD2_Off();
127 }
128 
129 
130 // ------------------------------------------------------------------------
131 // Enciende el LED 3.
132 //
133 // -> (void    ) Nada.
134 //
135 // <- (void    ) Nada.
136 // ------------------------------------------------------------------------
137 void Hw_LD3_On(void)
138 {
139     LD3_On();
140 }
141 
142 
143 // ------------------------------------------------------------------------
144 // Apaga el LED 3.
145 //
146 // -> (void    ) Nada.
147 //
148 // <- (void    ) Nada.
149 // ------------------------------------------------------------------------
150 void Hw_LD3_Off(void)
151 {
152     LD3_Off();
153 }
154 
155 
156 // ------------------------------------------------------------------------
157 // Obtiene un número aleatorio.
158 //
159 // -> (void    ) Nada.
160 //
161 // <- (uint    ) Número aleatorio.
162 // ------------------------------------------------------------------------
163 unsigned int Hw_RNG_Get(void)
164 {
```

```
165     return RNG_Get();
166 }
167 
168 
169 // ----------------------------------------------------------------------------
170 // Lee la fecha y hora del RTC.
171 //
172 // -> (void    ) Nada.
173 //
174 // <- (time_t  ) Timestamp.
175 // ----------------------------------------------------------------------------
176 time_t Hw_RTC_Read(void)
177 {
178     return RTC_Read();
179 }
180 
181 
182 // ----------------------------------------------------------------------------
183 // Escribe la fecha y hora en el RTC.
184 //
185 // -> (time_t  ) Timestamp.
186 //
187 // <- (void    ) Nada.
188 // ----------------------------------------------------------------------------
189 void Hw_RTC_Write(time_t curtime)
190 {
191     RTC_Write(curtime);
192 }
193 
194 
195 // ----------------------------------------------------------------------------
196 // Activa la UART.
197 //
198 // -> (void    ) Nada.
199 //
200 // <- (void    ) Nada.
201 // ----------------------------------------------------------------------------
202 void Hw_UART_Start(void)
203 {
204     UART_Start();
205 }
206 
207 
208 // ----------------------------------------------------------------------------
209 // Desactiva la UART.
210 //
211 // -> (void    ) Nada.
```

```
212 //
213 // <- (void    ) Nada.
214 // ----------------------------------------------------------------------------
215 void Hw_UART_Stop(void)
216 {
217     UART_Stop();
218 }
219 
220 
221 // ----------------------------------------------------------------------------
222 // Reinicia y arranca el temporizador.
223 //
224 // -> (void    ) Nada.
225 //
226 // <- (void    ) Nada.
227 // ----------------------------------------------------------------------------
228 void Hw_TIM_Start(void)
229 {
230     TIM_Start();
231 }
232 
233 
234 // ----------------------------------------------------------------------------
235 // Para el temporizador.
236 //
237 // -> (void    ) Nada.
238 //
239 // <- (void    ) Nada.
240 // ----------------------------------------------------------------------------
241 void Hw_TIM_Stop(void)
242 {
243     TIM_Stop();
244 }
245 
246 
247 // ----------------------------------------------------------------------------
248 // Lee la cuenta actual del temporizador.
249 //
250 // -> (void    ) Nada.
251 //
252 // <- (int     ) Número de microsegundos.
253 // ----------------------------------------------------------------------------
254 int Hw_TIM_Read(void)
255 {
256     return TIM_Read();
257 }
258 
```

```
259
260 // ------------------------------------------------------------------------
261 // Comprueba el estado de la conexión LAN.
262 //
263 // -> (void    ) Nada.
264 //
265 // <- (int     ) 0 si se detecta la conexión, un valor negativo en caso
266 //               contrario.
267 // ------------------------------------------------------------------------
268 int Hw_ETH_Link_Status(void)
269 {
270     return ETH_Link_Status();
271 }
```

/NUCLEO_H723ZG_MQTT_TLS/Application/include/hw.h

```
1  // ------------------------------------------------------------------------
2  // HW
3  //
4  // - Archivo de cabecera.
5  //
6  // ------------------------------------------------------------------------
7  // Copyright (c) 2021 Jorge Botana Mtz. de Ibarreta
8  //
9  // Este archivo se encuentra bajo los términos de la Licencia MIT. Debería
10 // haberse proporcionado una copia de ella junto a este fichero. Si no es así, se
11 // puede encontrar en el siguiente enlace:
12 //
13 //                                          https://opensource.org/licenses/MIT
14 // ------------------------------------------------------------------------
15
16 #ifndef   HW_H
17 #define   HW_H
18
19 #include <time.h>
20
21 void Hw_Init(void);
22
23 void Hw_Tick_Delay(unsigned int delay_ms);
24
25 int Hw_B1_Read(void);
26
27 void Hw_LD1_On(void);
28 void Hw_LD1_Off(void);
29
30 void Hw_LD2_On(void);
31 void Hw_LD2_Off(void);
```

```
32
33 void Hw_LD3_On(void);
34 void Hw_LD3_Off(void);
35
36 unsigned int Hw_RNG_Get(void);
37
38 time_t Hw_RTC_Read(void);
39 void Hw_RTC_Write(time_t curtime);
40
41 void Hw_UART_Start(void);
42 void Hw_UART_Stop(void);
43
44 void Hw_TIM_Start(void);
45 void Hw_TIM_Stop(void);
46 int Hw_TIM_Read(void);
47
48 int Hw_ETH_Link_Status(void);
49
50 #endif // HW_H
```

B.2. Ajustes

B.2.1. SETTINGS

/NUCLEO_H723ZG_MQTT_TLS/Application/include/cfg/settings.h

```
1  // ----------------------------------------------------------------------------
2  // SETTINGS
3  //
4  // - Archivo de configuración del programa.
5  //
6  // ----------------------------------------------------------------------------
7  // Copyright (c) 2021 Jorge Botana Mtz. de Ibarreta
8  //
9  // Este archivo se encuentra bajo los términos de la Licencia MIT. Debería
10 // haberse proporcionado una copia de ella junto a este fichero. Si no es así, se
11 // puede encontrar en el siguiente enlace:
12 //
13 //                                          https://opensource.org/licenses/MIT
14 // ----------------------------------------------------------------------------
15
16 #ifndef   SETTINGS_H
17 #define   SETTINGS_H
18
19 // ---------------------------------- General -----------------------------------
```

```
20
21 // ------------------------------------------------------------------------------
22 // Nombre de la plataforma sobre la que corre el programa.
23 //
24 // Valor predeterminado :                                        "[MY_PLATFORM]"
25 // ------------------------------------------------------------------------------
26 #define   PLATFORM                                               STM32_PLATFORM
27
28 // ------------------------------------------------------------------------------
29 // - Definido  : habilitar la depuración por la UART.
30 //
31 // - Comentado : deshabilitar la depuración por la UART.
32 // ------------------------------------------------------------------------------
33 #define   UART_DEBUG
34
35 // ------------------------------------ DHCP ------------------------------------
36
37 // ------------------------------------------------------------------------------
38 // Número máximo de intentos del cliente DHCP hasta establecer una configuración
39 // de red estática. Se puede asignar el valor 0 si se desea esto último.
40 //
41 // Advertencia: NO es el número máximo de paquetes "DHCP Request" enviados.
42 //
43 // Valor predeterminado :                                                    50
44 // ------------------------------------------------------------------------------
45 #define   DHCP_ATTEMPTS                                                   50
46
47 // ------------------------------------------------------------------------------
48 // IP estática.
49 //
50 // Valor predeterminado :                                      "192.168.1.10"
51 // ------------------------------------------------------------------------------
52 #define   STATIC_IP                                           "192.168.1.10"
53
54 // ------------------------------------------------------------------------------
55 // Máscara de subred estática.
56 //
57 // Valor predeterminado :                                      "255.255.255.0"
58 // ------------------------------------------------------------------------------
59 #define   STATIC_NETMASK                                      "255.255.255.0"
60
61 // ------------------------------------------------------------------------------
62 // Puerta de enlace estática.
63 //
64 // Valor predeterminado :                                        "192.168.1.1"
65 // ------------------------------------------------------------------------------
66 #define   STATIC_GW                                           "192.168.1.1"
```

```
67
68 // ------------------------------------ DNS -------------------------------------
69
70 // ------------------------------------------------------------------------------
71 // - Definido  : conectar al servidor DNS principal.
72 //
73 // - Comentado : conectar al servidor DNS alternativo.
74 //
75 // Nota: se puede asignar el valor "gateway" (sin comillas) para usar la puerta
76 // de enlace como servidor DNS.
77 //
78 // Advertencia: si se usa un servidor DNS externo, este no podrá resolver
79 // hostnames de dispositivos pertenecientes a la red local.
80 // ------------------------------------------------------------------------------
81 // #define   USE_MAIN_DNS_SERVER
82
83 // ------------------------------------------------------------------------------
84 // Dirección IPv4 del servidor DNS principal.
85 //
86 // Valor predeterminado :                                              gateway
87 // ------------------------------------------------------------------------------
88 #define   MAIN_DNS_SERVER                                             gateway
89
90 // ------------------------------------------------------------------------------
91 // Dirección IPv4 del servidor DNS alternativo.
92 //
93 // Valor predeterminado :                                            "1.1.1.1"
94 // ------------------------------------------------------------------------------
95 #define   ALT_DNS_SERVER                                            "8.8.8.8"
96
97 // ------------------------------------ NTP -------------------------------------
98
99 // ------------------------------------------------------------------------------
100 // - Definido  : conectar al servidor NTP principal.
101 //
102 // - Comentado : conectar al servidor NTP alternativo.
103 // ------------------------------------------------------------------------------
104 // #define   USE_MAIN_NTP_SERVER
105
106 // ------------------------------------------------------------------------------
107 // Hostname del servidor NTP principal.
108 //
109 // Valor predeterminado :                                 "MY-MAIN-NTP-SERVER"
110 // ------------------------------------------------------------------------------
111 #define   MAIN_NTP_SERVER                                           "JORGE-PC"
112
113 // ------------------------------------------------------------------------------
```

```
114 // Hostname del servidor NTP alternativo.
115 //
116 // Valor predeterminado :                                    "es.pool.ntp.org"
117 // ---------------------------------------------------------------------------
118 #define   ALT_NTP_SERVER                                   "time.windows.com"
119
120 // ---------------------------------------------------------------------------
121 // Zona horaria GMT.
122 //
123 // Valor predeterminado :                                                    0
124 // ---------------------------------------------------------------------------
125 #define   GMT_TIME                                                       +1
126
127 // ------------------------------------ TLS ------------------------------------
128
129 // ---------------------------------------------------------------------------
130 // - Definido  : usar una conexión TLS con autenticación del servidor.
131 //
132 // - Comentado : usar una conexión insegura.
133 // ---------------------------------------------------------------------------
134 #define   USE_TLS
135
136 // ---------------------------------------------------------------------------
137 // - Definido  : incorporar el certificado del cliente y conectar a un broker que
138 //               lo requiera.
139 //
140 // - Comentado : NO incorporar el certificado del cliente y conectar a un broker
141 //               que NO lo requiera.
142 //
143 // Nota: esta definición no tiene efecto en conexiones inseguras.
144 // ---------------------------------------------------------------------------
145 // #define   USE_CLIENT_AUTH
146
147 // ---------------------------------------------------------------------------
148 // Nivel de depuración por printf() de mbedTLS, desde 0 (desactivado) hasta 5
149 // (depuración máxima). No se consideran los mensajes principales del cliente TLS
150 // (esos salen siempre).
151 //
152 // Nota: es útil para obtener las claves intercambiadas durante el handshake TLS,
153 // con las que se podrá desencriptar la sesión TLS con un sniffer de red como
154 // Wireshark (siendo necesario como mínimo un nivel 3 de depuración).
155 //
156 // Valor predeterminado :                                                    0
157 // ---------------------------------------------------------------------------
158 #define   TLS_DEBUG_LEVEL                                                 0
159
160 // ------------------------------------ MQTT ------------------------------------
```

```
161
162 // ------------------------------------------------------------------------------
163 // - Definido  : conectar al broker MQTT principal.
164 //
165 // - Comentado : conectar al broker MQTT alternativo.
166 // ------------------------------------------------------------------------------
167 // #define   USE_MAIN_MQTT_BROKER
168
169 // ------------------------------------------------------------------------------
170 // Hostname del broker MQTT principal.
171 //
172 // Valor predeterminado :                                  "MY-MAIN-MQTT-BROKER"
173 // ------------------------------------------------------------------------------
174 #define   MAIN_MQTT_BROKER                                              "JORGE-PC"
175
176 // ------------------------------------------------------------------------------
177 // Hostname del broker MQTT alternativo.
178 //
179 // Valor predeterminado :                                   "test.mosquitto.org"
180 // ------------------------------------------------------------------------------
181 #define   ALT_MQTT_BROKER                                   "test.mosquitto.org"
182
183 // ------------------------------------------------------------------------------
184 // Identificador del cliente MQTT.
185 //
186 // Valor predeterminado :                                      "[MY_CLIENT_ID]"
187 // ------------------------------------------------------------------------------
188 #define   CLIENT_ID                                                     "[JORGE]"
189
190 // ------------------------------------------------------------------------------
191 // Segundos de Keep Alive.
192 //
193 // Nota: para que el broker no desconecte al cliente por inactividad, en esta
194 // aplicación el Keep Alive debe ser bastante superior a 10 segundos, a menos que
195 // se desactive asignándole el valor 0. Sin embargo, esto último podría dar
196 // problemas en algunos brokers.
197 //
198 // Valor predeterminado :                                                    0
199 // ------------------------------------------------------------------------------
200 #define   KEEP_ALIVE                                                      60
201
202 // ------------------------------------------------------------------------------
203 // Tema público MQTT.
204 //
205 // Valor predeterminado :                                    "[MY_PUBLIC_TOPIC]"
206 // ------------------------------------------------------------------------------
207 #define   PUBLIC_TOPIC                                          "[TEMA_PÚBLICO]"
```

```
208
209 // ------------------------------------------------------------------------
210 // Tema secreto MQTT.
211 //
212 // Valor predeterminado :                                  "[MY_SECRET_TOPIC]"
213 // ------------------------------------------------------------------------
214 #define   SECRET_TOPIC                                      "[TEMA_SECRETO]"
215
216 // ------------------------------------------------------------------------
217 // - Definido  : habilitar el modo de pruebas.
218 //
219 // - Comentado : deshabilitar el modo de pruebas.
220 // ------------------------------------------------------------------------
221 // #define   TEST_MODE
222
223 // ------------------------------------------------------------------------
224 // Tamaño de los mensajes MQTT enviados y recibidos en el modo de pruebas. Ha de
225 // ser bastante inferior al tamaño del buffer de recepción (2048 bytes) para
226 // poder recibir los mismos mensajes que se envían, ya que el tamaño de un
227 // paquete MQTT es superior al tamaño del mensaje que contiene, debido a que hay
228 // que sumarle el resto de bytes que forman dicho paquete. Y hay que tener en
229 // cuenta que, si se usa una conexión segura, un paquete MQTT pasará a ser un
230 // paquete TLS de mayor tamaño.
231 //
232 // Advertencia: si el tamaño de los paquetes supera el tamaño máximo de segmento
233 // TCP (1460 bytes), estos serán divididos, lo que va a influir en los resultados
234 // de la prueba.
235 //
236 // Valor predeterminado :                                              1024
237 // ------------------------------------------------------------------------
238 #define   TEST_MSG_BYTES                                            1024
239
240 // ------------------------------------------------------------------------
241 // Número de ciclos que se realizan en el modo de pruebas.
242 //
243 // Nota: 1 ciclo = 1 envío + 1 recepción.
244 //
245 // Advertencia: un número reducido de ciclos hará que los resultados de la prueba
246 // no sean fiables.
247 //
248 // Valor predeterminado :                                               100
249 // ------------------------------------------------------------------------
250 #define   TEST_N_CYCLES                                             100
251
252 // ------------------------------------------------------------------------
253
254 #endif // SETTINGS_H
```

B.2.2. CERTIFICATES

/NUCLEO_H723ZG_MQTT_TLS/Application/include/cfg/certificates.h

```
// ------------------------------------------------------------------------------
// CERTIFICATES
//
// - Certificados y par de claves para TLS_CLIENT:
//
//   -> Certificado del CA raíz de JORGE-PC
//   -> Certificado del cliente de JORGE-PC
//   -> Certificado del CA raíz de test.mosquitto.org
//   -> Certificado del cliente de test.mosquitto.org
//   -> Par de claves del cliente.
//
// ------------------------------------------------------------------------------
// Copyright (c) 2021 Jorge Botana Mtz. de Ibarreta
//
// Este archivo se encuentra bajo los términos de la Licencia MIT. Debería
// haberse proporcionado una copia de ella junto a este fichero. Si no es así, se
// puede encontrar en el siguiente enlace:
//
//                                          https://opensource.org/licenses/MIT
// ------------------------------------------------------------------------------

#ifndef   CERTIFICATES_H
#define   CERTIFICATES_H

// ------------------------------------------------------------------------------

#include "setup.h"

#ifdef    USE_MAIN_MQTT_BROKER

#define   CA_CRT                                                    \
"\
-----BEGIN CERTIFICATE-----\n\
MIIC8jCCAdqgAwIBAgIBATANBgkqhkiG9w0BAQsFADASMRAwDgYDVQQDDAdSb290\n\
IENBMB4XDTAwMDEwMTAwMDAwMFoXDTI5MTIzMTIzNTk1OVowEjEQMA4GA1UEAwwH\n\
Um9vdCBDQTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAL2Y/0Kqgrw+\n\
ypzuRCRn0GjPd1RPmDTVaw764zFjjAuCRSA2X5wzr27uIstRnOUbc2aLmcGH3cgk\n\
Hsn8yWUbTslhzK45SgejucJtnG8EuRm3LdLRvKh2cmPSHeCTX0uNk9sR0a6oy5S+\n\
F/Mv7yRfA2fiaizIajb6GpSeZ91sTshbNlLiql7vCUCJpJ1fJrUCHGM9NyOMe7Ra\n\
DgzJ2gbl7pDRIqzB/95wf4Zsfqz6P7avYHjzavOKZm0HGOO/drT5tBSWN5ovT/p4\n\
NgtuBfGjKXYoUeWXeq8T9zfSdpaezGcrEr0NJRTvauNrNZrfVAApzA2rtLsndynt\n\
xMakXOOUZWsCAwEAAaNTMFEwDwYDVR0TBAgwBgEB/wIBATAdBgNVHQ4EFgQU6snm\n\
9H6ertcqFio83qZ1PlhVeU0wHwYDVR0jBBgwFoAU6snm9H6ertcqFio83qZ1PlhV\n\
```

```
eU0wDQYJKoZIhvcNAQELBQADggEBAHaoVXSPOYB/j1WrjonMR192ZrKU44CsN7C2\n\
JArGiNhErOkRpf2t57UGTHF18gsrwBqI8cQ/hM6ij7Ewyvl9Ngsn+NvRy7iJIl3g\n\
fxzwH9bRiTCeo2vbqpIYYMYYfV++Od43xDEbOhU+zZQ1FZSRaC+kBszDQz7nmnQE\n\
luiDKbAThQipqQfTTIdswaZQCyOtczhoq7vuI117xvtSrjE2e0kkcOj9DdtaJJ8U\n\
DWODK6XGQJ4xlEScyhfgZjfjD232cHZV2sf5h6LDIpmBIEu/KlYoITuVz/yRubR/\n\
BtGp8avVp0obxQm2hJpW0IzdJBNt2VADL28qvHmlq/ehPtZYQ2g=\n\
-----END CERTIFICATE-----\n\
"

#define   CLIENT_CRT                                                \
"\
-----BEGIN CERTIFICATE-----\n\
MIIC8jCCAdqgAwIBAgIBATANBgkqhkiG9w0BAQsFADASMRAwDgYDVQQDDAdNUVRU\n\
IENBMB4XDTIxMDEwMTAwMDAwMFoXDTIzMTIzMTIzNTk1OVowGDEWMBQGA1UEAwwN\n\
TlVDTEVPLUg3MjNaRzCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBANlG\n\
LE9mF67UzARAgbRTMwqIONFlD5SmKJwJtcOeiMKEUoND41CPmd/jHp8PzbJKpCeD\n\
SA0fcr62CvkXMM/Hu2JvE+3CPHXjEJz5kPMk4nZ0v/vOr3aBj1FrMwQtw4vq8atv\n\
PH10pZ2g5/9fNHoEjO17RXyGUYGmXdA+6KFRNyB7AVqxyXmBbqfkaJ6YeY4pAANI\n\
3dARXtDXSK2e/Dts7Ra6X1em7gS8ObdQnoK1ipv9kjMLFbnKCjXV8seAYr7oUUdE\n\
QhdW21IWq3IanQP2vaA+KrgQqvQpJpUplHqIkzjxTOzsEN9DRn0sTlwli52IrWAJ\n\
73sKRrIYHJ0hlCvTAFkCAwEAAaNNMEswCQYDVR0TBAIwADAdBgNVHQ4EFgQU33lS\n\
Nf29lzhL+oldUG5Suj0kzXAwHwYDVR0jBBgwFoAUnQO2I0uPXS60oRliRvBD8vX4\n\
4RUwDQYJKoZIhvcNAQELBQADggEBAJbgc0x0/lCz6rwpTWi66jMV8aIL+iF9jHgn\n\
ycTZXuWuQjZ9DHoPTxIbPTrSgnsoffblKB0uPEzPfix23EIXTBsJ62a0lirZEiHm\n\
iiqdvu/M/+LuZr1dhUh0stxN7KzPLev0A3VM84bH/Lkfbvva9Qw8SZYH4r/AyoGD\n\
8m2y2zwBN3LmTKdVqddWDTEs1z7mJKJx3tg/oxqce/R0w7kZDvg5aIIrrWyaboUF\n\
vegOv2Qj7SE7GTYMyRaI+kDTcA+KCWNEIfnT2WRfrhwPEyB+kP7iVLC5V0Jb554L\n\
15udqxW134CDz9W95r/H1rR4q0sL4oBPnwMRvsf73WJgqChYydA=\n\
-----END CERTIFICATE-----\n\
"

#else  // USE_MAIN_MQTT_BROKER

#define   CA_CRT                                                    \
"\
-----BEGIN CERTIFICATE-----\n\
MIIEAzCCAuugAwIBAgIUBY1hlCGvdj4NhBXkZ/uLUZNILAwwDQYJKoZIhvcNAQEL\n\
BQAwgZAxCzAJBgNVBAYTAkdCMRcwFQYDVQQIDA5Vbml0ZWQgS2luZ2RvbTEOMAwG\n\
A1UEBwwFRGVyYnkxEjAQBgNVBAoMCU1vc3F1aXR0bzELMAkGA1UECwwCQ0ExFjAU\n\
BgNVBAMMDW1vc3F1aXR0by5vcmcxHzAdBgkqhkiG9w0BCQEWEHJvZ2VyQGF0Y2hv\n\
by5vcmcwHhcNMjAwNjA5MTEwNjM5WhcNMzAwNjA3MTEwNjM5WjCBkDELMAkGA1UE\n\
BhMCR0IxFzAVBgNVBAgMDlVuaXRlZCBLaW5nZG9tMQ4wDAYDVQQHDAVEZXJieTES\n\
MBAGA1UECgwJTW9zcXVpdHRvMQswCQYDVQQLDAJDQTEWMBQGA1UEAwwNbW9zcXVp\n\
dHRvLm9yZzEfMB0GCSqGSIb3DQEJARYQcm9nZXJAYXRjaG9vLm9yZzCCASIwDQYJ\n\
KoZIhvcNAQEBBQADggEPADCCAQoCggEBAME0HKmIzfTOwkKLT3THHe+ObdizamPg\n\
UZmD64Tf3zJdNeYGYn4CEXbyP6fy3tWc8S2boW6dzrH8SdFf9uo320GJA9B7U1FW\n\
Te3xda/Lm3JFfaHjkWw7jBwcauQZjpGINHapHRlpiCZsquAthOgxW9SgDgYlGzEA\n\
```

```
s06pkEFiMw+qDfLo/sxFKB6vQlFekMeCymjLCbNwPJyqyhFmPWwio/PDMruBTzPH\n\
3cioBnrJWKXc3OjXdLGFJOfj7pP0j/dr2LH72eSvv3PQQFl90CZPFhrCUcRHSSxo\n\
E6yjGOdnz7f6PveLIB574kQORwt8ePn0yidrTC1ictikED3nHYhMUOUCAwEAAaNT\n\
MFEwHQYDVR0OBBYEFPVV6xBUFPiGKDyo5V3+Hbh4N9YSMB8GA1UdIwQYMBaAFPVV\n\
6xBUFPiGKDyo5V3+Hbh4N9YSMA8GA1UdEwEB/wQFMAMBAf8wDQYJKoZIhvcNAQEL\n\
BQADggEBAGa9kS21N70ThM6/Hj9D7mbVxKLBjVWe2TPsGfbl3rEDfZ+OKRZ2j6AC\n\
6r7jb4TZO3dzF2p6dgbrlU71Y/4K0TdzIjRj3cQ3KSm41JvUQ0hZ/c04iGDg/xWf\n\
+pp58nfPAYwuerruPNWmlStWAXf0UTqRtg4hQDWBuUFDJTuWuuBvEXudz74eh/wK\n\
sMwfu1HFvjy5Z0iMDU8PUDepjVolOCue9ashlS4EB5IECdSR2TItnAIiIwimx839\n\
LdUdRudafMu5T5Xma182OC0/u/xRlEm+tvKGGmfFcN0piqVl8OrSPBgIlb+1IKJE\n\
m/XriWr/Cq4h/JfB7NTsezVslgkBaoU=\n\
-----END CERTIFICATE-----\n\
"

#define   CLIENT_CRT                                                \
"\
-----BEGIN CERTIFICATE-----\n\
MIIDaDCCAlCgAwIBAgIBADANBgkqhkiG9w0BAQsFADCBkDELMAkGA1UEBhMCR0Ix\n\
FzAVBgNVBAgMDlVuaXRlZCBLaW5nZG9tMQ4wDAYDVQQHDAVEZXJieTESMBAGA1UE\n\
CgwJTW9zcXVpdHRvMQswCQYDVQQLDAJDQTEWMBQGA1UEAwwNbW9zcXVpdHRvLm9y\n\
ZzEfMB0GCSqGSIb3DQEJARYQcm9nZXJAYXRjaG9vLm9yZzAeFw0yMTExMDIyMDE0\n\
NDlaFw0yMjAxMzEyMDE0NDlaMEIxFjAUBgNVBAMMDU5VQ0xFTy1INzIzWkcxGzAZ\n\
BgNVBAoMElRGRyAtIEpvcmdlIEJvdGFuYTELMAkGA1UEBhMCRVMwggEiMA0GCSqG\n\
SIb3DQEBAQUAA4IBDwAwggEKAoIBAQDZRixPZheu1MwEQIG0UzMKiDjRZQ+Upiic\n\
CbXDnojChFKDQ+NQj5nf4x6fD82ySqQng0gNH3K+tgr5FzDPx7tibxPtwjx14xCc\n\
+ZDzJOJ2dL/7zq92gY9RazMELcOL6vGrbzx9dKWdoOf/XzR6BIzte0V8hlGBpl3Q\n\
PuihUTcgewFascl5gW6n5GiemHmOKQADSN3QEV7Q10itnvw7bO0Wul9Xpu4EvDm3\n\
UJ6CtYqb/ZIzCxW5ygo11fLHgGK+6FFHREIXVttSFqtyGp0D9r2gPiq4EKr0KSaV\n\
KZR6iJM48Uzs7BDfQ0Z9LE5cJYudiK1gCe97CkayGBydIZQr0wBZAgMBAAGjGjAY\n\
MAkGA1UdEwQCMAAwCwYDVR0PBAQDAgXgMA0GCSqGSIb3DQEBCwUAA4IBAQBaDJjS\n\
/yGyypwb87wTp6QWivEl8FLrw3WdJokSXRxlxeXZqZfKeyat4G6EneQSRPsiAhQB\n\
POnt1/6IauYeoGg1AtnVDB8abSdEbuLEUcYMzrnGZJ2uYLHh3C+nUHptS12KYU7C\n\
yspMgp6LgnJBNyi8fD2GSSdhNCKHnYEKCKgJg6Ch4hLFgQCHLlHxwHktEW7zpmMK\n\
CO1iFTwVihv4Z89NXbXlr4cXCwcBuc/9tQNFdb/Wl1f9pX3zMFnkO53i/GT+1TFa\n\
sRYPuyZmSZsqI4kRQdSRcBcI6sWzjlkNMBw0i5vH2q9aQ2cH7AkeygEH1pLSsS/n\n\n\
bnHVWgpqe6KtORd7\n\
-----END CERTIFICATE-----\n\
"

#endif // USE_MAIN_MQTT_BROKER

#define   CLIENT_KEY                                                \
"\
-----BEGIN RSA PRIVATE KEY-----\n\
MIIEpAIBAAKCAQEA2UYsT2YXrtTMBECBtFMzCog40WUPlKYonAm1w56IwoRSg0Pj\n\
UI+Z3+Menw/NskqkJ4NIDR9yvrYK+Rcwz8e7Ym8T7cI8deMQnPmQ8yTidnS/+86v\n\
doGPUWszBC3Di+rxq288fXSlnaDn/180egSM7XtFfIZRgaZd0D7ooVE3IHsBWrHJ\n\
```

```
eYFup+Ronph5jikAA0jd0BFe0NdIrZ78O2ztFrpfV6buBLw5t1CegrWKm/2SMwsV\n\
ucoKNdXyx4BivuhRR0RCF1bbUharchqdA/a9oD4quBCq9CkmlSmUeoiTOPFM7OwQ\n\
3ONGfSxOXCWLnYitYAnvewpGshgcnSGUK9MAWQIDAQABAoIBAGE3REE8eXFJFoX6\n\
hO514ZsfS2jpsph85iO3NoRyLVf3k+HW4OFdd2iVO1DJZeyV+DATzgvNG+7YDj3Y\n\
0gbOI/pNoBmcdKf8sqTCB8i7IrcE0xfsqVa5brJr6DZA8cNrmrltup+ZrzbB1bxO\n\
UXEdEx95LiSjBtofYc95LmZrq+JUQ/DEGAQB0YDoS0pcs7IOA4TTxaMUSD5+sWVr\n\
2kg6jKCCLKzFlV9y3BGofhuvwV1MqT5fOZO8/CD27z1fiRvanotxT3Q+wz4g4lC0\n\
TwQ6yzverdux/Qv0ZrIoYWA9m6IDO8pWTp2ZpYuZG6ypZ+MwOzhgbbm3EGKw7Pio\n\
2lVV5RcCgYEA+PeBBar5DB8DEgKn8GaPuiHZvHJS81TuOdcdWL0qY382bvA/agBu\n\
2udfMSfSIVDfQvzt7m9nCJ3VgK3uqNGMayWqnugJg1LiiDBfSeSoggWFgr2ZOSTj\n\
mAuqP0D6fJzpXrEL9jVEnKOcaioYX/ECvq0aidXX1o2M5Iriyqp8t78CgYEA32l4\n\
uJin4QZKkySaKbxYrXYg8Wddfo6dic99UL9m2hwnsJhp10147Nf/A24R3CFHWGiK\n\
L3U/QvdHOVi2XfkTyLW+gYeE1lzOx17imZVZKLO6z38pnxf8afLFFAN3m/Hc9BvT\n\
dFrEFAUg+MmmkYl844O/Qx/1ZCml9U1NXvkrjecCgYEAihbNi1U/eYeh7lVmzKI2\n\
SekImctXUvviflZgQp35auJeUL4UjjjbQ7NdSyhpFm2Cx+b+GWfU4CSbDRpr61r+\n\
5VcpAaZ10E8isqMR1yxqgmOJn/CFkfpx+h1OOnNR+9gUYJ8WoWL+QVtGIGRkzKIj\n\
AZuFwrnHU4uDgCfAjuRs9IUCgYAUrpvpYuBvuXhf5MyMvo3rCPuFvY7vL16wKlAs\n\
kHuCB4c5FvL9ZoOiUWCfPapTmZUUPyFDwXPozCGA0pCXZH6LXx/St+v5OEPEzSjn\n\
FOdbGEBcAyakPsE0QbAy6RtCiYJt6sGJjPmH7O2fenN8F6FDIqgrG6NI8X4WlmBq\n\
QWDh0wKBgQCa/WZI3vXp5Pc6lux7ywc0jMd7S4sVI/LsnUXpOFv0/T1B0qR8mMiv\n\
1bjQ+U6yUKMYS6gkdhwos76gTS8SVtZvikTAL2ZxI6Pd04twoTwL9TFAUzyX5fhU\n\
ZkWDl/BfJcczve76qE0tJ1sqXRTnyfpLkDAKEp2kPDeYIqZ2Hfujnw==\n\
-----END RSA PRIVATE KEY-----\n\
"

// ---------------------------------------------------------------------------

#endif // CERTIFICATES_H
```

B.2.3. SETUP

/NUCLEO_H723ZG_MQTT_TLS/Application/include/cfg/setup.h

```
// ---------------------------------------------------------------------------
// SETUP
//
// - Carga del archivo de configuración del programa.
//
// - Para las definiciones que tengan que llevar asignado un valor, si alguna de
//   ellas no estuviera definida en el archivo de configuración del programa, se
//   cargará un valor predeterminado.
//
// - Carga de definiciones adicionales a partir de la configuración anterior.
//
// ---------------------------------------------------------------------------
// Copyright (c) 2021 Jorge Botana Mtz. de Ibarreta
```

```
14 //
15 // Este archivo se encuentra bajo los términos de la Licencia MIT. Debería
16 // haberse proporcionado una copia de ella junto a este fichero. Si no es así, se
17 // puede encontrar en el siguiente enlace:
18 //
19 //                                          https://opensource.org/licenses/MIT
20 // ------------------------------------------------------------------------------
21
22 #ifndef   SETUP_H
23 #define   SETUP_H
24
25 // ------------------- Archivo de configuración del programa --------------------
26
27 #include "settings.h"
28
29 // -------------------------- Valores predeterminados ---------------------------
30
31 #ifndef   PLATFORM
32 #define   PLATFORM                                                  "[MY_PLATFORM]"
33 #endif // PLATFORM
34
35 #ifndef   DHCP_ATTEMPTS
36 #define   DHCP_ATTEMPTS                                                         50
37 #endif // DHCP_ATTEMPTS
38
39 #ifndef   STATIC_IP
40 #define   STATIC_IP                                                  "192.168.1.10"
41 #endif // STATIC_IP
42
43 #ifndef   STATIC_NETMASK
44 #define   STATIC_NETMASK                                            "255.255.255.0"
45 #endif // STATIC_NETMASK
46
47 #ifndef   STATIC_GW
48 #define   STATIC_GW                                                   "192.168.1.1"
49 #endif // STATIC_GW
50
51 #ifndef   MAIN_DNS_SERVER
52 #define   MAIN_DNS_SERVER                                                   gateway
53 #endif // MAIN_DNS_SERVER
54
55 #ifndef   ALT_DNS_SERVER
56 #define   ALT_DNS_SERVER                                                  "1.1.1.1"
57 #endif // ALT_DNS_SERVER
58
59 #ifndef   MAIN_NTP_SERVER
60 #define   MAIN_NTP_SERVER                                       "MY-MAIN-NTP-SERVER"
```

```
61  #endif // MAIN_NTP_SERVER
62
63  #ifndef   ALT_NTP_SERVER
64  #define   ALT_NTP_SERVER                                    "es.pool.ntp.org"
65  #endif // ALT_NTP_SERVER
66
67  #ifndef   GMT_TIME
68  #define   GMT_TIME                                                          0
69  #endif // GMT_TIME
70
71  #ifndef   TLS_DEBUG_LEVEL
72  #define   TLS_DEBUG_LEVEL                                                   0
73  #endif // TLS_DEBUG_LEVEL
74
75  #ifndef   MAIN_MQTT_BROKER
76  #define   MAIN_MQTT_BROKER                                "MY-MAIN-MQTT-BROKER"
77  #endif // MAIN_MQTT_BROKER
78
79  #ifndef   ALT_MQTT_BROKER
80  #define   ALT_MQTT_BROKER                                  "test.mosquitto.org"
81  #endif // ALT_MQTT_BROKER
82
83  #ifndef   CLIENT_ID
84  #define   CLIENT_ID                                           "[MY_CLIENT_ID]"
85  #endif // CLIENT_ID
86
87  #ifndef   KEEP_ALIVE
88  #define   KEEP_ALIVE                                                        0
89  #endif // KEEP_ALIVE
90
91  #ifndef   PUBLIC_TOPIC
92  #define   PUBLIC_TOPIC                                     "[MY_PUBLIC_TOPIC]"
93  #endif // PUBLIC_TOPIC
94
95  #ifndef   SECRET_TOPIC
96  #define   SECRET_TOPIC                                     "[MY_SECRET_TOPIC]"
97  #endif // SECRET_TOPIC
98
99  #ifndef   TEST_MSG_BYTES
100 #define   TEST_MSG_BYTES                                                 1024
101 #endif // TEST_MSG_BYTES
102
103 #ifndef   TEST_N_CYCLES
104 #define   TEST_N_CYCLES                                                   100
105 #endif // TEST_N_CYCLES
106
107 // ------------------------- Definiciones adicionales -------------------------
```

```
108
109 #ifdef    USE_MAIN_DNS_SERVER
110 #define   DNS_SERVER                                          MAIN_DNS_SERVER
111 #else  // USE_MAIN_DNS_SERVER
112 #define   DNS_SERVER                                           ALT_DNS_SERVER
113 #endif // USE_MAIN_DNS_SERVER
114
115 #ifdef    USE_MAIN_NTP_SERVER
116 #define   NTP_SERVER                                          MAIN_NTP_SERVER
117 #else  // USE_MAIN_NTP_SERVER
118 #define   NTP_SERVER                                           ALT_NTP_SERVER
119 #endif // USE_MAIN_NTP_SERVER
120
121 #ifdef    USE_MAIN_MQTT_BROKER
122 #define   MQTT_BROKER                                        MAIN_MQTT_BROKER
123 #else  // USE_MAIN_MQTT_BROKER
124 #define   MQTT_BROKER                                         ALT_MQTT_BROKER
125 #endif // USE_MAIN_MQTT_BROKER
126
127 #ifdef    USE_TLS
128 #ifdef    USE_CLIENT_AUTH
129 #define   MQTT_PORT                                                    8884
130 #else  // USE_CLIENT_AUTH
131 #define   MQTT_PORT                                                    8883
132 #endif // USE_CLIENT_AUTH
133 #else  // USE_TLS
134 #define   MQTT_PORT                                                    1883
135 #endif // USE_TLS
136
137 // ----------------------------------------------------------------------------
138
139 #endif // SETUP_H
```

B.2.4. FREERTOSCONFIG

/NUCLEO_H723ZG_MQTT_TLS/Application/include/cfg/FreeRTOSConfig.h

```
1 // ----------------------------------------------------------------------------
2 // FREERTOSCONFIG
3 //
4 // - Archivo de configuración de FreeRTOS.
5 //
6 // ----------------------------------------------------------------------------
7 // FreeRTOS Kernel V10.4.4
8 // Copyright (C) 2021 Amazon.com, Inc. or its affiliates.
9 // All Rights Reserved.
```

```
10 //
11 // SPDX-License-Identifier: MIT
12 //
13 // Permission is hereby granted, free of charge, to any person obtaining a copy
14 // of this software and associated documentation files (the "Software"), to deal
15 // in the Software without restriction, including without limitation the rights
16 // to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
17 // copies of the Software, and to permit persons to whom the Software is
18 // furnished to do so, subject to the following conditions:
19 //
20 // The above copyright notice and this permission notice shall be included in all
21 // copies or substantial portions of the Software.
22 //
23 // THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
24 // IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
25 // FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
26 // AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
27 // LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
28 // OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
29 // SOFTWARE.
30 //
31 // https://www.FreeRTOS.org
32 // https://github.com/FreeRTOS
33 // -----------------------------------------------------------------------------
34
35 #ifndef   FREERTOSCONFIG_H
36 #define   FREERTOSCONFIG_H
37
38 // --------------------------- Dispositivo : STM32 ----------------------------
39
40 #include  STM32_DEVICE
41
42 // ----------------------- Port para GCC : ARM Cortex-M7 ------------------------
43
44 #define   configCPU_CLOCK_HZ                                    SystemCoreClock
45 #define   configTICK_RATE_HZ                                              1000
46
47 #define   configPRIO_BITS                                      __NVIC_PRIO_BITS
48 #define   configLIBRARY_LOWEST_INTERRUPT_PRIORITY                           15
49 #define   configLIBRARY_MAX_SYSCALL_INTERRUPT_PRIORITY                       5
50
51 #define   configKERNEL_INTERRUPT_PRIORITY                                    \
52 (                                                                            \
53     configLIBRARY_LOWEST_INTERRUPT_PRIORITY      <<                          \
54     (8 - configPRIO_BITS)                                                    \
55 )
56
```

```
57  #define   configMAX_SYSCALL_INTERRUPT_PRIORITY                                    \
58  (                                                                                  \
59      configLIBRARY_MAX_SYSCALL_INTERRUPT_PRIORITY <<                                \
60      (8 - configPRIO_BITS)                                                          \
61  )
62
63  #define   vPortSVCHandler                                              SVC_Handler
64  #define   xPortPendSVHandler                                        PendSV_Handler
65
66  // ------------------- Esquema de gestión de memoria : heap_4 -------------------
67
68  #define   configMINIMAL_STACK_SIZE                                            1024
69  #define   configTOTAL_HEAP_SIZE                    (25 * configMINIMAL_STACK_SIZE)
70
71  // --------------------------- Wrapper : CMSIS-RTOS2 ----------------------------
72
73  #define   INCLUDE_xSemaphoreGetMutexHolder                                       1
74  #define   INCLUDE_vTaskDelay                                                     1
75  #define   INCLUDE_vTaskDelayUntil                                                1
76  #define   INCLUDE_vTaskDelete                                                    1
77  #define   INCLUDE_xTaskGetCurrentTaskHandle                                      1
78  #define   INCLUDE_xTaskGetSchedulerState                                         1
79  #define   INCLUDE_uxTaskGetStackHighWaterMark                                    1
80  #define   INCLUDE_uxTaskPriorityGet                                              1
81  #define   INCLUDE_vTaskPrioritySet                                               1
82  #define   INCLUDE_eTaskGetState                                                  1
83  #define   INCLUDE_vTaskSuspend                                                   1
84  #define   INCLUDE_xTimerPendFunctionCall                                         1
85
86  #define   configUSE_TIMERS                                                       1
87  #define   configUSE_MUTEXES                                                      1
88  #define   configUSE_COUNTING_SEMAPHORES                                          1
89  #define   configUSE_TRACE_FACILITY                                               1
90  #define   configUSE_16_BIT_TICKS                                                 0
91  #define   configMAX_PRIORITIES                                                  56
92  #define   configUSE_PORT_OPTIMISED_TASK_SELECTION                                0
93
94  #define   configTIMER_TASK_PRIORITY                                              2
95  #define   configTIMER_QUEUE_LENGTH                                              10
96  #define   configTIMER_TASK_STACK_DEPTH                    configMINIMAL_STACK_SIZE
97
98  #define   configUSE_PREEMPTION                                                   1
99  #define   configUSE_IDLE_HOOK                                                    0
100 #define   configUSE_TICK_HOOK                                                    0
101
102 // ------------------------------------------------------------------------------
103
```

```
104 #endif // FREERTOSCONFIG_H
```

B.2.5. LWIPOPTS

/NUCLEO_H723ZG_MQTT_TLS/Application/include/cfg/lwipopts.h

```
1  // ------------------------------------------------------------------------
2  // LWIPOPTS
3  //
4  // - Archivo de configuración de lwIP.
5  //
6  // ------------------------------------------------------------------------
7  // Copyright (c) 2001-2004 Swedish Institute of Computer Science.
8  // All rights reserved.
9  //
10 // Redistribution and use in source and binary forms, with or without
11 // modification, are permitted provided that the following conditions are met:
12 //
13 // 1. Redistributions of source code must retain the above copyright notice, this
14 //    list of conditions and the following disclaimer.
15 //
16 // 2. Redistributions in binary form must reproduce the above copyright notice,
17 //    this list of conditions and the following disclaimer in the documentation
18 //    and/or other materials provided with the distribution.
19 //
20 // 3. The name of the author may not be used to endorse or promote products
21 //    derived from this software without specific prior written permission.
22 //
23 // THIS SOFTWARE IS PROVIDED BY THE AUTHOR ‘‘AS IS’’ AND ANY EXPRESS OR IMPLIED
24 // WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
25 // MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO
26 // EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
27 // SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
28 // PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
29 // BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER
30 // IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
31 // ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
32 // POSSIBILITY OF SUCH DAMAGE.
33 // ------------------------------------------------------------------------
34
35 #ifndef   LWIPOPTS_H
36 #define   LWIPOPTS_H
37
38 #ifdef    STM32H723xx
39 // ------------------------------------------------------------------------
40 // Realocar el puntero del heap de lwIP. Esto es necesario en los
```

```
41 // microcontroladores STM32H723, ya que sus descriptores DMA de Ethernet no
42 // pueden acceder a cualquier zona de la memoria RAM.
43 // ----------------------------------------------------------------------------
44 #define   LWIP_RAM_HEAP_POINTER                                    0x30004000
45 #endif // STM32H723xx
46
47 // ----------------------------------------------------------------------------
48 // Establecer la alineación de la memoria (CPU 32 bits = 4 bytes).
49 // ----------------------------------------------------------------------------
50 #define   MEM_ALIGNMENT                                                     4
51
52 // ----------------------------------------------------------------------------
53 // Aumentar el tamaño del heap de lwIP para poder enviar paquetes de gran tamaño
54 // (el predeterminado es de solo 1600 bytes).
55 // ----------------------------------------------------------------------------
56 #define   MEM_SIZE                                                (4 * 1024)
57
58 #ifdef    STM32H723xx
59 // ----------------------------------------------------------------------------
60 // Deshabilitar el "PBUF pool" para ahorrar mucha memoria RAM, ya que no se usa
61 // porque el driver de Ethernet empleado en un STM32H723 usa un "Zero-copy RX
62 // PBUF pool" en su lugar.
63 // ----------------------------------------------------------------------------
64 #define   PBUF_POOL_SIZE                                                    0
65 #endif // STM32H723xx
66
67 // ----------------------------------------------------------------------------
68 // Habilitar el módulo de DHCP integrado en lwIP.
69 // ----------------------------------------------------------------------------
70 #define   LWIP_DHCP                                                         1
71
72 // ----------------------------------------------------------------------------
73 // Aumentar el "TCP Maximum Segment Size" para ajustarlo al "Maximum Transmission
74 // Unit" (MTU) de Ethernet:
75 //
76 // -> 1500 bytes MTU - 20 bytes IP - 20 bytes TCP = 1460 bytes MSS
77 // ----------------------------------------------------------------------------
78 #define   TCP_MSS                                                        1460
79
80 // ----------------------------------------------------------------------------
81 // Configurar la tarea de TCP/IP.
82 // ----------------------------------------------------------------------------
83 #define   TCPIP_THREAD_STACKSIZE                  (2 * configMINIMAL_STACK_SIZE)
84 #define   TCPIP_THREAD_PRIO                                   osPriorityHigh
85 #define   TCPIP_MBOX_SIZE                                                   6
86 #define   DEFAULT_UDP_RECVMBOX_SIZE                                         6
87 #define   DEFAULT_TCP_RECVMBOX_SIZE                                         6
```

```
88
89 // ------------------------------------------------------------------------
90 // Habilitar un timeout para las operaciones de lectura en sockets.
91 // ------------------------------------------------------------------------
92 #define   LWIP_SO_RCVTIMEO                                              1
93
94 // ------------------------------------------------------------------------
95
96 #endif // LWIPOPTS_H
```

B.2.6. MBEDTLS_CONFIG_FILE

/NUCLEO_H723ZG_MQTT_TLS/Application/include/cfg/mbedTLS_config_file.h

```
1  // ------------------------------------------------------------------------
2  // MBEDTLS_CONFIG_FILE
3  //
4  // - Archivo de configuración de mbedTLS.
5  //
6  // ------------------------------------------------------------------------
7  // Copyright The Mbed TLS Contributors
8  // SPDX-License-Identifier: Apache-2.0
9  //
10 // Licensed under the Apache License, Version 2.0 (the "License"); you may not
11 // use this file except in compliance with the License. You may obtain a copy of
12 // the License at
13 //
14 // http://www.apache.org/licenses/LICENSE-2.0
15 //
16 // Unless required by applicable law or agreed to in writing, software
17 // distributed under the License is distributed on an "AS IS" BASIS, WITHOUT
18 // WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the
19 // License for the specific language governing permissions and limitations under
20 // the License.
21 // ------------------------------------------------------------------------
22
23 #ifndef   MBEDTLS_CONFIG_FILE_H
24 #define   MBEDTLS_CONFIG_FILE_H
25
26 // ------------------------------ mbedTLS 2.16.11 -------------------------------
27
28 // ------------------------------------------------------------------------
29 // Las opciones habilitadas son las mismas que están definidas en el archivo de
30 // configuración predeterminado que trae mbedTLS, ("config.h", que es sustituido
31 // por este fichero al definir el símbolo del proyecto MBEDTLS_CONFIG_FILE),
32 // realizando únicamente los siguientes cambios:
```

```
33 //
34 // - Habilitar    ->    #define MBEDTLS_PLATFORM_TIME_ALT
35 // - Habilitar    ->    #define MBEDTLS_ENTROPY_HARDWARE_ALT
36 // - Deshabilitar -> // #define MBEDTLS_FS_IO
37 // - Habilitar    ->    #define MBEDTLS_NO_PLATFORM_ENTROPY
38 // - Deshabilitar -> // #define MBEDTLS_NET_C
39 // - Deshabilitar -> // #define MBEDTLS_TIMING_C
40 //
41 // Estos cambios son necesarios para usar TLS_CLIENT y están pensados para usar
42 // mbedTLS en un sistema basado en microcontrolador, en lugar de un PC.
43 // ------------------------------------------------------------------------------
44
45 #define   MBEDTLS_HAVE_ASM
46 #define   MBEDTLS_HAVE_TIME
47 #define   MBEDTLS_HAVE_TIME_DATE
48 #define   MBEDTLS_PLATFORM_TIME_ALT
49 #define   MBEDTLS_ENTROPY_HARDWARE_ALT
50 #define   MBEDTLS_CIPHER_MODE_CBC
51 #define   MBEDTLS_CIPHER_MODE_CFB
52 #define   MBEDTLS_CIPHER_MODE_CTR
53 #define   MBEDTLS_CIPHER_MODE_OFB
54 #define   MBEDTLS_CIPHER_MODE_XTS
55 #define   MBEDTLS_CIPHER_PADDING_PKCS7
56 #define   MBEDTLS_CIPHER_PADDING_ONE_AND_ZEROS
57 #define   MBEDTLS_CIPHER_PADDING_ZEROS_AND_LEN
58 #define   MBEDTLS_CIPHER_PADDING_ZEROS
59 #define   MBEDTLS_REMOVE_ARC4_CIPHERSUITES
60 #define   MBEDTLS_REMOVE_3DES_CIPHERSUITES
61 #define   MBEDTLS_ECP_DP_SECP192R1_ENABLED
62 #define   MBEDTLS_ECP_DP_SECP224R1_ENABLED
63 #define   MBEDTLS_ECP_DP_SECP256R1_ENABLED
64 #define   MBEDTLS_ECP_DP_SECP384R1_ENABLED
65 #define   MBEDTLS_ECP_DP_SECP521R1_ENABLED
66 #define   MBEDTLS_ECP_DP_SECP192K1_ENABLED
67 #define   MBEDTLS_ECP_DP_SECP224K1_ENABLED
68 #define   MBEDTLS_ECP_DP_SECP256K1_ENABLED
69 #define   MBEDTLS_ECP_DP_BP256R1_ENABLED
70 #define   MBEDTLS_ECP_DP_BP384R1_ENABLED
71 #define   MBEDTLS_ECP_DP_BP512R1_ENABLED
72 #define   MBEDTLS_ECP_DP_CURVE25519_ENABLED
73 #define   MBEDTLS_ECP_DP_CURVE448_ENABLED
74 #define   MBEDTLS_ECP_NIST_OPTIM
75 #define   MBEDTLS_ECDSA_DETERMINISTIC
76 #define   MBEDTLS_KEY_EXCHANGE_PSK_ENABLED
77 #define   MBEDTLS_KEY_EXCHANGE_DHE_PSK_ENABLED
78 #define   MBEDTLS_KEY_EXCHANGE_ECDHE_PSK_ENABLED
79 #define   MBEDTLS_KEY_EXCHANGE_RSA_PSK_ENABLED
```

```
80  #define   MBEDTLS_KEY_EXCHANGE_RSA_ENABLED
81  #define   MBEDTLS_KEY_EXCHANGE_DHE_RSA_ENABLED
82  #define   MBEDTLS_KEY_EXCHANGE_ECDHE_RSA_ENABLED
83  #define   MBEDTLS_KEY_EXCHANGE_ECDHE_ECDSA_ENABLED
84  #define   MBEDTLS_KEY_EXCHANGE_ECDH_ECDSA_ENABLED
85  #define   MBEDTLS_KEY_EXCHANGE_ECDH_RSA_ENABLED
86  #define   MBEDTLS_PK_PARSE_EC_EXTENDED
87  #define   MBEDTLS_ERROR_STRERROR_DUMMY
88  #define   MBEDTLS_GENPRIME
89  #define   MBEDTLS_NO_PLATFORM_ENTROPY
90  #define   MBEDTLS_PK_RSA_ALT_SUPPORT
91  #define   MBEDTLS_PKCS1_V15
92  #define   MBEDTLS_PKCS1_V21
93  #define   MBEDTLS_SELF_TEST
94  #define   MBEDTLS_SSL_ALL_ALERT_MESSAGES
95  #define   MBEDTLS_SSL_ENCRYPT_THEN_MAC
96  #define   MBEDTLS_SSL_EXTENDED_MASTER_SECRET
97  #define   MBEDTLS_SSL_FALLBACK_SCSV
98  #define   MBEDTLS_SSL_CBC_RECORD_SPLITTING
99  #define   MBEDTLS_SSL_RENEGOTIATION
100 #define   MBEDTLS_SSL_MAX_FRAGMENT_LENGTH
101 #define   MBEDTLS_SSL_PROTO_TLS1
102 #define   MBEDTLS_SSL_PROTO_TLS1_1
103 #define   MBEDTLS_SSL_PROTO_TLS1_2
104 #define   MBEDTLS_SSL_PROTO_DTLS
105 #define   MBEDTLS_SSL_ALPN
106 #define   MBEDTLS_SSL_DTLS_ANTI_REPLAY
107 #define   MBEDTLS_SSL_DTLS_HELLO_VERIFY
108 #define   MBEDTLS_SSL_DTLS_CLIENT_PORT_REUSE
109 #define   MBEDTLS_SSL_DTLS_BADMAC_LIMIT
110 #define   MBEDTLS_SSL_SESSION_TICKETS
111 #define   MBEDTLS_SSL_EXPORT_KEYS
112 #define   MBEDTLS_SSL_SERVER_NAME_INDICATION
113 #define   MBEDTLS_SSL_TRUNCATED_HMAC
114 #define   MBEDTLS_VERSION_FEATURES
115 #define   MBEDTLS_X509_CHECK_KEY_USAGE
116 #define   MBEDTLS_X509_CHECK_EXTENDED_KEY_USAGE
117 #define   MBEDTLS_X509_RSASSA_PSS_SUPPORT
118 #define   MBEDTLS_AESNI_C
119 #define   MBEDTLS_AES_C
120 #define   MBEDTLS_ARC4_C
121 #define   MBEDTLS_ASN1_PARSE_C
122 #define   MBEDTLS_ASN1_WRITE_C
123 #define   MBEDTLS_BASE64_C
124 #define   MBEDTLS_BIGNUM_C
125 #define   MBEDTLS_BLOWFISH_C
126 #define   MBEDTLS_CAMELLIA_C
```

```
127 #define   MBEDTLS_CCM_C
128 #define   MBEDTLS_CERTS_C
129 #define   MBEDTLS_CHACHA20_C
130 #define   MBEDTLS_CHACHAPOLY_C
131 #define   MBEDTLS_CIPHER_C
132 #define   MBEDTLS_CTR_DRBG_C
133 #define   MBEDTLS_DEBUG_C
134 #define   MBEDTLS_DES_C
135 #define   MBEDTLS_DHM_C
136 #define   MBEDTLS_ECDH_C
137 #define   MBEDTLS_ECDSA_C
138 #define   MBEDTLS_ECP_C
139 #define   MBEDTLS_ENTROPY_C
140 #define   MBEDTLS_ERROR_C
141 #define   MBEDTLS_GCM_C
142 #define   MBEDTLS_HKDF_C
143 #define   MBEDTLS_HMAC_DRBG_C
144 #define   MBEDTLS_MD_C
145 #define   MBEDTLS_MD5_C
146 #define   MBEDTLS_OID_C
147 #define   MBEDTLS_PADLOCK_C
148 #define   MBEDTLS_PEM_PARSE_C
149 #define   MBEDTLS_PEM_WRITE_C
150 #define   MBEDTLS_PK_C
151 #define   MBEDTLS_PK_PARSE_C
152 #define   MBEDTLS_PK_WRITE_C
153 #define   MBEDTLS_PKCS5_C
154 #define   MBEDTLS_PKCS12_C
155 #define   MBEDTLS_PLATFORM_C
156 #define   MBEDTLS_POLY1305_C
157 #define   MBEDTLS_RIPEMD160_C
158 #define   MBEDTLS_RSA_C
159 #define   MBEDTLS_SHA1_C
160 #define   MBEDTLS_SHA256_C
161 #define   MBEDTLS_SHA512_C
162 #define   MBEDTLS_SSL_CACHE_C
163 #define   MBEDTLS_SSL_COOKIE_C
164 #define   MBEDTLS_SSL_TICKET_C
165 #define   MBEDTLS_SSL_CLI_C
166 #define   MBEDTLS_SSL_SRV_C
167 #define   MBEDTLS_SSL_TLS_C
168 #define   MBEDTLS_VERSION_C
169 #define   MBEDTLS_X509_USE_C
170 #define   MBEDTLS_X509_CRT_PARSE_C
171 #define   MBEDTLS_X509_CRL_PARSE_C
172 #define   MBEDTLS_X509_CSR_PARSE_C
173 #define   MBEDTLS_X509_CREATE_C
```

```
#define   MBEDTLS_X509_CRT_WRITE_C
#define   MBEDTLS_X509_CSR_WRITE_C
#define   MBEDTLS_XTEA_C
#define   MBEDTLS_TLS_DEFAULT_ALLOW_SHA1_IN_KEY_EXCHANGE

#include "mbedtls/check_config.h"

// ------------------------------------------------------------------------

#endif // MBEDTLS_CONFIG_FILE_H
```

B.2.7. STM32H7XX_HAL_CONF

/NUCLEO_H723ZG_MQTT_TLS/Application/include/cfg/stm32h7xx_hal_conf.h

```
// ------------------------------------------------------------------------
// STM32H7XX_HAL_CONF
//
// - Archivo de configuración de los drivers HAL y del BSP.
//
// ------------------------------------------------------------------------
// Copyright (c) 2017 STMicroelectronics.
// All rights reserved.
//
// This software component is licensed by ST under BSD 3-Clause license, the
// "License"; You may not use this file except in compliance with the License.
// You may obtain a copy of the License at:
//
//                                        opensource.org/licenses/BSD-3-Clause
// ------------------------------------------------------------------------

#ifndef   STM32H7XX_HAL_CONF_H
#define   STM32H7XX_HAL_CONF_H

// ------------------------------- HAL modules ------------------------------

#define   HAL_CORTEX_MODULE_ENABLED
#define   HAL_DMA_MODULE_ENABLED
#define   HAL_ETH_MODULE_ENABLED
#define   HAL_FLASH_MODULE_ENABLED
#define   HAL_GPIO_MODULE_ENABLED
#define   HAL_PWR_MODULE_ENABLED
#define   HAL_RCC_MODULE_ENABLED
#define   HAL_RNG_MODULE_ENABLED
#define   HAL_RTC_MODULE_ENABLED
#define   HAL_TIM_MODULE_ENABLED
```

```
32 #define   HAL_UART_MODULE_ENABLED
33
34 // ------------------------ Oscillator values adaptation ------------------------
35
36 #define   HSE_VALUE                                                    8000000
37 #define   HSE_STARTUP_TIMEOUT                                              100
38 #define   CSI_VALUE                                                    4000000
39 #define   HSI_VALUE                                                   64000000
40 #define   LSE_VALUE                                                      32768
41 #define   LSE_STARTUP_TIMEOUT                                             5000
42 #define   LSI_VALUE                                                      32000
43 #define   EXTERNAL_CLOCK_VALUE                                        12288000
44
45 // ---------------------------- System configuration ----------------------------
46
47 #define   VDD_VALUE                                                       3300
48 #define   TICK_INT_PRIORITY                                                 15
49
50 // ---------------------------- Ethernet configuration ----------------------------
51
52 #define   ETH_TX_DESC_CNT                                                    4
53 #define   ETH_RX_DESC_CNT                                                    4
54
55 #define   ETH_MAC_ADDR0                                                   0x00
56 #define   ETH_MAC_ADDR1                                                   0xC0
57 #define   ETH_MAC_ADDR2                                                   0xDF
58 #define   ETH_MAC_ADDR3                                                   0x24
59 #define   ETH_MAC_ADDR4                                                   0xBA
60 #define   ETH_MAC_ADDR5                                                   0x6B
61
62 // ---------------------------------- HAL includes ----------------------------------
63
64 #include "stm32h7xx_hal_rcc.h"
65 #include "stm32h7xx_hal_gpio.h"
66 #include "stm32h7xx_hal_dma.h"
67 #include "stm32h7xx_hal_eth.h"
68
69 #include "stm32h7xx_hal_cortex.h"
70 #include "stm32h7xx_hal_flash.h"
71 #include "stm32h7xx_hal_pwr.h"
72 #include "stm32h7xx_hal_rng.h"
73 #include "stm32h7xx_hal_rtc.h"
74 #include "stm32h7xx_hal_tim.h"
75 #include "stm32h7xx_hal_uart.h"
76
77 // ---------------------------------- No assert ----------------------------------
78
```

```
79 #define   assert_param(expr)                                          ((void)0)
80
81 // ----------------------------------------------------------------------------
82
83 #endif // STM32H7XX_HAL_CONF_H
```

B.3. Clientes

B.3.1. TCPIP_SETUP

/NUCLEO_H723ZG_MQTT_TLS/Application/source/sw/tcpip_setup.c

```
1  // ----------------------------------------------------------------------------
2  // TCPIP_SETUP
3  //
4  // - Inicialización del stack de TCP/IP lwIP.
5  //
6  // - Activación y desactivación de la interfaz de red.
7  //
8  // ----------------------------------------------------------------------------
9  // Copyright (c) 2021 Jorge Botana Mtz. de Ibarreta
10 //
11 // Este archivo se encuentra bajo los términos de la Licencia MIT. Debería
12 // haberse proporcionado una copia de ella junto a este fichero. Si no es así, se
13 // puede encontrar en el siguiente enlace:
14 //
15 //                                          https://opensource.org/licenses/MIT
16 // ----------------------------------------------------------------------------
17
18
19 #include "tcpip_setup.h"
20
21 #include "lwip/init.h"
22 #include "lwip/tcpip.h"
23
24
25 struct netif netif = {0};
26
27
28 void TCPIP_Init(void)
29 {
30     printf(" - Inicializando TCP/IP ........");
31     fflush(stdout);
32
33     // Crear la tarea de TCP/IP.
```

```
34      tcpip_init(NULL, NULL);
35
36      // Inicializar la interfaz de red.
37      ip4_addr_t ipaddr  = {0};
38      ip4_addr_t netmask = {0};
39      ip4_addr_t gw      = {0};
40      netif_add        (&netif, &ipaddr, &netmask, &gw, NULL, &ethernetif_init, &
        tcpip_input);
41      netif_set_default(&netif);
42      netif_set_link_up(&netif);
43      netif_set_up     (&netif);
44
45      printf(" ok\n");
46
47      // Mostrar la versión de lwIP.
48      printf(" -> lwIP %i.%i.%i\n",
49          LWIP_VERSION_MAJOR,
50          LWIP_VERSION_MINOR,
51          LWIP_VERSION_REVISION);
52
53      // Mostrar la dirección MAC.
54      printf(" -> MAC address : %02X:%02X:%02X:%02X:%02X:%02X\n\n", \
55          netif.hwaddr[0],                                          \
56          netif.hwaddr[1],                                          \
57          netif.hwaddr[2],                                          \
58          netif.hwaddr[3],                                          \
59          netif.hwaddr[4],                                          \
60          netif.hwaddr[5]);
61  }
62
63
64  void TCPIP_Down_Up(void)
65  {
66  #ifdef    STM32H723xx
67      // "Workaround" para hacer frente a un problema por el cual solo funciona el
68      // primer socket creado: tras cerrarlo y abrir un segundo socket, este último
69      // no funcionará a menos que se añadan las dos siguientes líneas antes de
70      // crear cada socket:
71
72      netif_set_down(&netif);
73      netif_set_up  (&netif);
74
75      // El problema aparece cuando se habilita la D-Cache + MPU en un STM32H723.
76  #endif // STM32H723xx
77  }
```

/NUCLEO_H723ZG_MQTT_TLS/Application/include/sw/tcpip_setup.h

```
// ------------------------------------------------------------------------------
// TCPIP_SETUP
//
// - Archivo de cabecera.
//
// ------------------------------------------------------------------------------
// Copyright (c) 2021 Jorge Botana Mtz. de Ibarreta
//
// Este archivo se encuentra bajo los términos de la Licencia MIT. Debería
// haberse proporcionado una copia de ella junto a este fichero. Si no es así, se
// puede encontrar en el siguiente enlace:
//
//                                          https://opensource.org/licenses/MIT
// ------------------------------------------------------------------------------

#ifndef   TCPIP_SETUP_H
#define   TCPIP_SETUP_H

#include "lwip/netif.h"

extern struct netif netif;

err_t ethernetif_init(struct netif *netif);

void TCPIP_Init(void);

void TCPIP_Down_Up(void);

#endif // TCPIP_SETUP_H
```

B.3.2. UDP_CLIENT

/NUCLEO_H723ZG_MQTT_TLS/Application/source/sw/udp_client.c

```
// ------------------------------------------------------------------------------
// UDP_CLIENT
//
// - Cliente UDP sobre IPv4.
//
// - Requiere de una implementación externa de la API de sockets BSD, que ha de
//   definirse en un símbolo de la configuración del proyecto:
//
//   -> Symbol : SOCKETS_IMPLEMENTATION
//   -> Value  : filename.h
//
```

```
12 // ---------------------------------------------------------------------------
13 // Copyright (c) 2021 Jorge Botana Mtz. de Ibarreta
14 //
15 // Este archivo se encuentra bajo los términos de la Licencia MIT. Debería
16 // haberse proporcionado una copia de ella junto a este fichero. Si no es así, se
17 // puede encontrar en el siguiente enlace:
18 //
19 //                                          https://opensource.org/licenses/MIT
20 // ---------------------------------------------------------------------------
21
22
23 #include "udp_client.h"
24
25 #ifndef   SOCKETS_IMPLEMENTATION
26 #error   "SOCKETS_IMPLEMENTATION                  no definido en el proyecto!!!"
27 #else  // SOCKETS_IMPLEMENTATION
28 #include  SOCKETS_IMPLEMENTATION
29
30
31 static void Error_Code(int ret);
32
33
34 int UDP_Request(const char *hostname, const char *ipaddr, const int port, const
       char *s_buf, size_t s_len, char *r_buf, size_t r_len)
35 {
36     printf(" - Preguntando a %s:%i ...", hostname, port);
37     fflush(stdout);
38
39     // Crear el socket UDP.
40     int s = socket(AF_INET, SOCK_DGRAM, 0);
41
42     // Establecer un timeout de recepción de 10 segundos.
43     struct timeval timeout = {0};
44     timeout.tv_sec         = 10;
45     timeout.tv_usec        =  0;
46     setsockopt(s, SOL_SOCKET, SO_RCVTIMEO, &timeout, sizeof(timeout));
47
48     // Establecer la dirección IP y el puerto del servidor.
49     struct sockaddr_in dst = {0};
50     dst.sin_len            = sizeof(dst);
51     dst.sin_family         = AF_INET;
52     dst.sin_port           = htons(port);
53     dst.sin_addr.s_addr    = inet_addr(ipaddr);
54
55     // Enviar el segmento UDP al servidor.
56     sendto(s, s_buf, s_len, 0, (struct sockaddr *)&dst, dst.sin_len);
57
```

```
58      // Obtener y guardar en un buffer la respuesta del servidor.
59      struct sockaddr src     = {0};
60      socklen_t       src_len = sizeof(src);
61      int ret = recvfrom(s, r_buf, r_len, 0, &src, &src_len);
62      Error_Code(ret);
63
64      // Cerrar el socket.
65      close(s);
66
67      return ret;
68  }
69
70
71  static void Error_Code(int ret)
72  {
73      // Indicar si se ha obtenido respuesta del servidor.
74      if(ret < 0)
75      {
76          printf(" error !!!\n");
77      }
78      else
79      {
80          printf(" ok\n");
81      }
82  }
83
84
85  #endif // SOCKETS_IMPLEMENTATION
```

/NUCLEO_H723ZG_MQTT_TLS/Application/include/sw/udp_client.h

```
1   // ------------------------------------------------------------------------
2   // UDP_CLIENT
3   //
4   // - Archivo de cabecera.
5   //
6   // ------------------------------------------------------------------------
7   // Copyright (c) 2021 Jorge Botana Mtz. de Ibarreta
8   //
9   // Este archivo se encuentra bajo los términos de la Licencia MIT. Debería
10  // haberse proporcionado una copia de ella junto a este fichero. Si no es así, se
11  // puede encontrar en el siguiente enlace:
12  //
13  //                                          https://opensource.org/licenses/MIT
14  // ------------------------------------------------------------------------
15
16  #ifndef   UDP_CLIENT_H
```

```
17 #define   UDP_CLIENT_H
18
19 #include <stddef.h>
20
21 int UDP_Request(const char *hostname, const char *ipaddr, const int port, const
       char *s_buf, size_t s_len, char *r_buf, size_t r_len);
22
23 #endif // UDP_CLIENT_H
```

B.3.3. TCP_CLIENT

/NUCLEO_H723ZG_MQTT_TLS/Application/source/sw/tcp_client.c

```
1  // ------------------------------------------------------------------------------
2  // TCP_CLIENT
3  //
4  // - Cliente TCP sobre IPv4.
5  //
6  // - Requiere de una implementación externa de la API de sockets BSD, que ha de
7  //   definirse en un símbolo de la configuración del proyecto:
8  //
9  //   -> Symbol : SOCKETS_IMPLEMENTATION
10 //   -> Value  : filename.h
11 //
12 // ------------------------------------------------------------------------------
13 // Copyright (c) 2021 Jorge Botana Mtz. de Ibarreta
14 //
15 // Este archivo se encuentra bajo los términos de la Licencia MIT. Debería
16 // haberse proporcionado una copia de ella junto a este fichero. Si no es así, se
17 // puede encontrar en el siguiente enlace:
18 //
19 //                                          https://opensource.org/licenses/MIT
20 // ------------------------------------------------------------------------------
21
22
23 #include "tcp_client.h"
24
25 #ifndef   SOCKETS_IMPLEMENTATION
26 #error   "SOCKETS_IMPLEMENTATION                  no definido en el proyecto!!!"
27 #else  // SOCKETS_IMPLEMENTATION
28 #include  SOCKETS_IMPLEMENTATION
29
30 #include <stdio.h>
31
32
33 int s = 0;
```

```
34
35
36 static void Error_Code(int ret);
37
38
39 int TCP_Init(const char *hostname, const char *ipaddr, const int port)
40 {
41     printf(" - Conectando a %s:%i ...", hostname, port);
42     fflush(stdout);
43
44     // Crear el socket TCP.
45     s = socket(AF_INET, SOCK_STREAM, 0);
46
47     // Establecer un timeout de recepción de 10 segundos.
48     struct timeval timeout = {0};
49     timeout.tv_sec         = 10;
50     timeout.tv_usec        =  0;
51     setsockopt(s, SOL_SOCKET, SO_RCVTIMEO, &timeout, sizeof(timeout));
52
53     // Establecer la dirección IP y el puerto del servidor.
54     struct sockaddr_in dst = {0};
55     dst.sin_len            = sizeof(dst);
56     dst.sin_family         = AF_INET;
57     dst.sin_port           = htons(port);
58     dst.sin_addr.s_addr    = inet_addr(ipaddr);
59
60     // Conectarse al servidor.
61     int ret = connect(s, (struct sockaddr *)&dst, dst.sin_len);
62     Error_Code(ret);
63
64     return ret;
65 }
66
67
68 int TCP_Close(char *buf, size_t len)
69 {
70     printf(" - Cerrando conexión ..........");
71     fflush(stdout);
72
73     // Procesar los datos aún no leídos para cerrar la conexión correctamente.
74     while(read(s, buf, len) > 0) {};
75
76     // Desconectarse del servidor y cerrar el socket.
77     int ret = close(s);
78     Error_Code(ret);
79
80     return ret;
```

```
81 }
82
83
84 int TCP_Read(char *buf, size_t len)
85 {
86     // Leer, bloqueando hasta que sea lea un dato o hasta el timeout.
87     return read(s, buf, len);
88 }
89
90
91 int TCP_Write(const char *buf, size_t len)
92 {
93     // Escribir, bloqueando hasta que se termine de escribir.
94     return write(s, buf, len);
95 }
96
97
98 static void Error_Code(int ret)
99 {
100     // Indicar si la conexión o desconexión se ha realizado correctamente.
101     if(ret < 0)
102     {
103         printf(" error !!!\n");
104     }
105     else
106     {
107         printf(" ok\n");
108     }
109     printf("\n");
110 }
111
112
113 #endif // SOCKETS_IMPLEMENTATION
```

/NUCLEO_H723ZG_MQTT_TLS/Application/include/sw/tcp_client.h

```
1  // ------------------------------------------------------------------------------
2  // TCP_CLIENT
3  //
4  // - Archivo de cabecera.
5  //
6  // ------------------------------------------------------------------------------
7  // Copyright (c) 2021 Jorge Botana Mtz. de Ibarreta
8  //
9  // Este archivo se encuentra bajo los términos de la Licencia MIT. Debería
10 // haberse proporcionado una copia de ella junto a este fichero. Si no es así, se
11 // puede encontrar en el siguiente enlace:
```

```
12 //
13 //                                               https://opensource.org/licenses/MIT
14 // ----------------------------------------------------------------------------
15
16 #ifndef   TCP_CLIENT_H
17 #define   TCP_CLIENT_H
18
19 #include <stddef.h>
20
21 extern int s;
22
23 int TCP_Init(const char *hostname, const char *ipaddr, const int port);
24 int TCP_Close(char *buf, size_t len);
25
26 int TCP_Read(char *buf, size_t len);
27 int TCP_Write(const char *buf, size_t len);
28
29 #endif // TCP_CLIENT_H
```

B.3.4. DHCP_CLIENT

/NUCLEO_H723ZG_MQTT_TLS/Application/source/sw/dhcp_client.c

```
1  // ----------------------------------------------------------------------------
2  // DHCP_CLIENT
3  //
4  // - Cliente DHCP IPv4.
5  //
6  // - Requiere del módulo de DHCP del stack de TCP/IP lwIP.
7  //
8  // ----------------------------------------------------------------------------
9  // Copyright (c) 2021 Jorge Botana Mtz. de Ibarreta
10 //
11 // Este archivo se encuentra bajo los términos de la Licencia MIT. Debería
12 // haberse proporcionado una copia de ella junto a este fichero. Si no es así, se
13 // puede encontrar en el siguiente enlace:
14 //
15 //                                               https://opensource.org/licenses/MIT
16 // ----------------------------------------------------------------------------
17
18
19 #include "dhcp_client.h"
20
21 #include "tcpip_setup.h"
22
23 #include "lwip/dhcp.h"
```

```
24 #include "lwip/inet.h"
25
26
27 #define   DHCP_START                                                    -2
28 #define   DHCP_WAIT                                                     -1
29 #define   DHCP_FINISH                                                    0
30
31
32 static void Netif_Info(struct netif netif, char *gateway);
33 static void Addr_Info(int addr);
34
35
36 int DHCP_Request(const int dhcp_attempts, const char *static_ip, const char *
       static_netmask, const char *static_gw, char *gateway)
37 {
38     static int DHCP_state = DHCP_START;
39     static int      count =          0;
40
41     switch(DHCP_state)
42     {
43         // Inicializar el módulo de DHCP.
44         case DHCP_START:
45             printf(" ---------- Cliente DHCP ----------\n\n");
46             if(dhcp_attempts > 0)
47             {
48                 printf(" - Buscando servidor DHCP ");
49                 fflush(stdout);
50                 dhcp_start(&netif);
51             }
52             else
53             {
54                 printf(" - DHCP omitido\n");
55             }
56             DHCP_state = DHCP_WAIT;
57             break;
58
59         // Esperar a que se obtenga la configuración de red.
60         case DHCP_WAIT:
61             if(dhcp_supplied_address(&netif))
62             {
63                 // Usar la configuración de red obtenida por DHCP.
64                 printf(" ok\n");
65                 printf(" - Configuración de red asignada por un servidor DHCP\n")
       ;
66                 Netif_Info(netif, gateway);
67                 DHCP_state = DHCP_FINISH;
68             }
```

```
69              else
70              {
71                  if(count < dhcp_attempts)
72                  {
73                      // Si el número de intentos es inferior al máximo permitido,
74                      // esperar a la vez que se van escribiendo puntos
75                      // suspensivos.
76                      count++;
77                      printf(".");
78                      fflush(stdout);
79                      DHCP_state = DHCP_WAIT;
80                  }
81                  else
82                  {
83                      // Parar el módulo de DHCP.
84                      if(dhcp_attempts > 0)
85                      {
86                          printf(" timeout !!!\n");
87                          dhcp_stop(&netif);
88                      }
89
90                      // Usar una configuración de red estática.
91                      const ip4_addr_t ip      = {
92                          .addr = inet_addr(static_ip     )
93                      };
94
95                      const ip4_addr_t netmask = {
96                          .addr = inet_addr(static_netmask)
97                      };
98
99                      const ip4_addr_t gw      = {
100                         .addr = inet_addr(static_gw     )
101                     };
102
103                     netif_set_addr(&netif, &ip, &netmask, &gw);
104
105                     printf(" - Configuración de red estática\n");
106                     Netif_Info(netif, gateway);
107                     DHCP_state = DHCP_FINISH;
108                 }
109             }
110             break;
111
112         default:
113             break;
114     }
115
```

```
116     return DHCP_state;
117 }
118 
119 
120 static void Netif_Info(struct netif netif, char *gateway)
121 {
122     // Mostrar la configuración de red.
123     printf(" -> IP address  : ");
124     Addr_Info(netif.ip_addr.addr);
125     printf(" -> Netmask     : ");
126     Addr_Info(netif.netmask.addr);
127     printf(" -> Gateway     : ");
128     Addr_Info(netif.gw.addr     );
129     printf("\n");
130 
131     // Guardar la puerta de enlace en un buffer.
132     sprintf(gateway, "%s", inet_ntoa(netif.gw));
133 }
134 
135 
136 static void Addr_Info(int addr)
137 {
138     // Mostrar una dirección IP, máscara de subred o puerta de enlace.
139     printf("%i.%i.%i.%i\n",        \
140         (addr & 0x000000FF) >>  0, \
141         (addr & 0x0000FF00) >>  8, \
142         (addr & 0x00FF0000) >> 16, \
143         (addr & 0xFF000000) >> 24);
144 }
```

/NUCLEO_H723ZG_MQTT_TLS/Application/include/sw/dhcp_client.h

```
1  // ------------------------------------------------------------------------------
2  // DHCP_CLIENT
3  //
4  // - Archivo de cabecera.
5  //
6  // ------------------------------------------------------------------------------
7  // Copyright (c) 2021 Jorge Botana Mtz. de Ibarreta
8  //
9  // Este archivo se encuentra bajo los términos de la Licencia MIT. Debería
10 // haberse proporcionado una copia de ella junto a este fichero. Si no es así, se
11 // puede encontrar en el siguiente enlace:
12 //
13 //                                          https://opensource.org/licenses/MIT
14 // ------------------------------------------------------------------------------
15 
```

```
16 #ifndef   DHCP_CLIENT_H
17 #define   DHCP_CLIENT_H
18
19 int DHCP_Request(const int dhcp_attempts, const char *static_ip, const char *
       static_netmask, const char *static_gw, char *gateway);
20
21 #endif // DHCP_CLIENT_H
```

B.3.5. DNS_CLIENT

/NUCLEO_H723ZG_MQTT_TLS/Application/source/sw/dns_client.c

```
1  // ------------------------------------------------------------------------------
2  // DNS_CLIENT
3  //
4  // - Cliente DNS IPv4.
5  //
6  // - Requiere de UDP_CLIENT.
7  //
8  // - No emplea librerías DNS de terceros, y supone una alternativa a funciones
9  //   como gethostbyname() o getaddrinfo().
10 //
11 // ------------------------------------------------------------------------------
12 // Copyright (c) 2021 Jorge Botana Mtz. de Ibarreta
13 //
14 // Este archivo se encuentra bajo los términos de la Licencia MIT. Debería
15 // haberse proporcionado una copia de ella junto a este fichero. Si no es así, se
16 // puede encontrar en el siguiente enlace:
17 //
18 //                                          https://opensource.org/licenses/MIT
19 // ------------------------------------------------------------------------------
20
21
22 #include "dns_client.h"
23
24 #include "udp_client.h"
25
26 #include <stdio.h>
27 #include <stdlib.h>
28
29
30 #define   DNS_PORT                                                        53
31 #define   HOST_START                                                      12
32 #define   DNS_PACKET_MIN_SIZE                                             17
33 #define   DNS_PACKET_MAX_SIZE                                            512
34
```

```
35
36 int DNS_Request(const char *server, const char *hostname, size_t len, char *
       ipaddr)
37 {
38     printf(" ---------- Cliente DNS ----------\n\n");
39
40     // Omitir la resolución DNS en caso de que se haya introducido una dirección
41     // IPv4 por hostname. Se presupone en un principio que esto es así, y se
42     // comprueba caracter por caracter en busca de caracteres distintos a los que
43     // aparecerían en una dirección IPv4.
44     int hostname_is_ipaddr = 1;
45     int i = 0;
46     for(i = 0; i < len; i++)
47     {
48         if(hostname[i] != '0' && hostname[i] != '1' && \
49            hostname[i] != '2' && hostname[i] != '3' && \
50            hostname[i] != '4' && hostname[i] != '5' && \
51            hostname[i] != '6' && hostname[i] != '7' && \
52            hostname[i] != '8' && hostname[i] != '9' && \
53            hostname[i] != '.' && hostname[i] != 0)
54         {
55             hostname_is_ipaddr = 0;
56         }
57     }
58
59     if(hostname_is_ipaddr == 0)
60     {
61         // Calcular la longitud total de este paquete DNS.
62         size_t dns_request_len = DNS_PACKET_MIN_SIZE + len;
63
64         // Alocar e inicializar a cero memoria para el paquete DNS.
65         char *DNS_REQUEST = calloc(dns_request_len, sizeof(char));
66
67         // - Transaction ID
68         DNS_REQUEST[ 0] = rand();
69         DNS_REQUEST[ 1] = rand();
70
71         // - Flags
72         DNS_REQUEST[ 2] = 0b00000001;
73                         //  7|||||||  - Response: Message is a query
74                         //   6543|||  - Opcode: Standard query (0)
75                         //       2||
76                         //        1|  - Truncated: Message is not truncated
77                         //         0  - Recursion desired: Do query recursively
78
79         DNS_REQUEST[ 3] = 0b00000000;
80                         //  7|||||||
```

```
81                      //   6|||||| - Z: reserved (0)
82                      //    5|||||
83                      //     4|||| - Non-authenticated data: Unacceptable
84                      //      3210
85
86          // - Questions      : 1
87          DNS_REQUEST[ 4] = 0x00;
88          DNS_REQUEST[ 5] = 0x01;
89
90          // - Answer RRs     : 0
91          DNS_REQUEST[ 6] = 0x00;
92          DNS_REQUEST[ 7] = 0x00;
93
94          // - Authority RRs  : 0
95          DNS_REQUEST[ 8] = 0x00;
96          DNS_REQUEST[ 9] = 0x00;
97
98          // - Additional RRs : 0
99          DNS_REQUEST[10] = 0x00;
100         DNS_REQUEST[11] = 0x00;
101
102         // - Queries
103
104         int i = 0;          // Posición actual en la cadena de texto del
105                             // hostname.
106
107         int j = 0;          // Contador de caracteres en una palabra (p.e. "time"
108                             // o "com").
109
110         int k = HOST_START; // Posición a partir de la cual se comienza a
111                             // escribir el hostname en el paquete.
112
113         // -> Name
114
115         // Mientras no se haya llegado al final del hostname ...
116         do
117         {
118             // ... se reinicia el número de caracteres hasta el siguiente punto o
119             // hasta el final del hostname.
120             j = 0;
121
122             // Mientras no se haya llegado al siguiente punto o al final del
123             // hostname ...
124             while((hostname[i + j] != '.') && \
125                   (hostname[i + j] != 0  ))
126             {
127                 // ... se calcula el número de caracteres hasta el siguiente
```

```
128                 // punto o hasta el final del hostname.
129                 j++;
130             }
131
132             // Se copia al paquete el número de caracteres hasta el siguiente
133             // punto o hasta el final del hostname.
134             DNS_REQUEST[k] = j;
135
136             // Se determina dónde dejar de escribir, cuando se haya llegado al
137             // siguiente punto o al final del hostname.
138             j = j + k;
139             for(k = k + 1; k <= j; k++)
140             {
141                 // Caracter actual del hostname copiado al paquete.
142                 DNS_REQUEST[k] = hostname[i];
143
144                 // Se avanza al siguiente caracter del hostname.
145                 i++;
146             }
147
148             // Se salta el punto para la siguiente iteración (los puntos no se
149             // copian al paquete). Si se tratase del NULL final, también hay que
150             // incrementar i para indicar que se ha llegado al final.
151             i++;
152
153             // Si se ha llegado al final, añadir el NULL final antes de salir del
154             // bucle.
155             if(i == len)
156             {
157                 DNS_REQUEST[k] = 0;
158             }
159         }
160         while(i < len);
161
162         // -> Type: A (Host Address) (1)
163         DNS_REQUEST[k + 1] = 0;
164         DNS_REQUEST[k + 2] = 1;
165
166         // -> Class: IN
167         DNS_REQUEST[k + 3] = 0;
168         DNS_REQUEST[k + 4] = 1;
169
170         // Realizar la comunicación y luego liberar la memoria alocada para el
171         // paquete DNS enviado.
172         char recv_buf[DNS_PACKET_MAX_SIZE] = {0};
173         int ret = UDP_Request(server, server, DNS_PORT, DNS_REQUEST,
    dns_request_len, recv_buf, sizeof(recv_buf));
```

```
174         free(DNS_REQUEST);
175         if(ret < 0)
176         {
177             printf(" - No se pudo resolver %s\n\n", hostname);
178             return ret;
179         }
180 
181         // Obtener la última dirección IP de la respuesta DNS (se suelen recibir
182         // varias, pero resulta sencillo coger la última porque son los últimos 4
183         // bytes del paquete recibido).
184         char get_IP[4] = {0};
185         for(i = 0; i < 4; i++)
186         {
187             get_IP[i] = recv_buf[ret - 4 + i];
188         }
189         sprintf(ipaddr, "%i.%i.%i.%i", get_IP[0], get_IP[1], get_IP[2], get_IP
    [3]);
190     }
191     else
192     {
193         // Si el hostname introducido es la dirección IP, entonces copiar el
194         // hostname a ipaddr, sin hacer la resolución DNS.
195         for(i = 0; i < len; i++)
196         {
197             ipaddr[i] = hostname[i];
198         }
199         printf(" - DNS omitido\n");
200     }
201 
202     // Mostrar la dirección IP del hostname tras haberla guardado en un buffer.
203     printf(" -> Hostname    : %s\n"  , hostname);
204     printf(" -> IP address  : %s\n\n", ipaddr  );
205 
206     return 0;
207 }
```

/NUCLEO_H723ZG_MQTT_TLS/Application/include/sw/dns_client.h

```
1  // ----------------------------------------------------------------------------
2  // DNS_CLIENT
3  //
4  // - Archivo de cabecera.
5  //
6  // ----------------------------------------------------------------------------
7  // Copyright (c) 2021 Jorge Botana Mtz. de Ibarreta
8  //
9  // Este archivo se encuentra bajo los términos de la Licencia MIT. Debería
```

```
10 // haberse proporcionado una copia de ella junto a este fichero. Si no es así, se
11 // puede encontrar en el siguiente enlace:
12 //
13 //                                              https://opensource.org/licenses/MIT
14 // ----------------------------------------------------------------------------
15
16 #ifndef   DNS_CLIENT_H
17 #define   DNS_CLIENT_H
18
19 #include <stddef.h>
20
21 int DNS_Request(const char *server, const char *hostname, size_t len, char *
       ipaddr);
22
23 #endif // DNS_CLIENT_H
```

B.3.6. NTP_CLIENT

/NUCLEO_H723ZG_MQTT_TLS/Application/source/sw/ntp_client.c

```
1  // ----------------------------------------------------------------------------
2  // NTP_CLIENT
3  //
4  // - Cliente NTP v4.
5  //
6  // - Requiere de UDP_CLIENT.
7  //
8  // - No emplea librerías NTP de terceros.
9  //
10 // ----------------------------------------------------------------------------
11 // Copyright (c) 2021 Jorge Botana Mtz. de Ibarreta
12 //
13 // Este archivo se encuentra bajo los términos de la Licencia MIT. Debería
14 // haberse proporcionado una copia de ella junto a este fichero. Si no es así, se
15 // puede encontrar en el siguiente enlace:
16 //
17 //                                              https://opensource.org/licenses/MIT
18 // ----------------------------------------------------------------------------
19
20
21 #include "ntp_client.h"
22
23 #include "udp_client.h"
24
25 #include <stdio.h>
26
```

```
27
28 #define   NTP_PORT                                                    123
29 #define   NTP_PACKET_SIZE                                              48
30 #define   SECS_FROM_1900_TO_1970                                 2208988800
31
32
33 int NTP_Request(const char *hostname, const char *ipaddr, const int gmt_time,
       time_t *curtime)
34 {
35     printf(" ---------- Cliente NTP ----------\n\n");
36
37     // Generar el paquete NTP.
38     char NTP_REQUEST[NTP_PACKET_SIZE] = {0};
39
40     // Modificar únicamente el primer byte del paquete NTP. Los otros bytes
41     // pueden valer 0.
42     NTP_REQUEST[0] = 0b11100011;
43                    //  76|||||| - Leap Indicator: unknown (clock unsynchronized)
44                    //    ||||||   (3)
45                    //    543||| - Version number: NTP Version 4 (4)
46                    //       210 - Mode: client (3)
47
48     // Realizar la comunicación.
49     char recv_buf[NTP_PACKET_SIZE] = {0};
50     int ret = UDP_Request(hostname, ipaddr, NTP_PORT, NTP_REQUEST,
       NTP_PACKET_SIZE, recv_buf, sizeof(recv_buf));
51     if(ret < 0)
52     {
53         printf(" - No se pudo obtener la fecha y hora\n\n");
54         return ret;
55     }
56
57     // Obtener los segundos desde el 01-01-1900 (contenidos en los bytes [40, 41,
58     // 42, 43] del paquete recibido), el origen de tiempos que usa NTP (UTC).
59     unsigned long secs_since_1900 = \
60         (recv_buf[40] << 24) +      \
61         (recv_buf[41] << 16) +      \
62         (recv_buf[42] <<  8) +      \
63         (recv_buf[43] <<  0);
64
65     // Obtener los segundos desde el 01-01-1970, el origen de tiempos que usan
66     // las funciones de "time.h" (UTC).
67     *curtime = (time_t)(secs_since_1900 - SECS_FROM_1900_TO_1970);
68
69     // Actualizar a la fecha y hora local (GMT).
70     *curtime = *curtime + (gmt_time * 60 * 60);
71
```

```
72      // Mostrar la fecha y hora.
73      char buf[30] = {0};
74      strftime(buf, sizeof(buf), "%a %d-%m-%Y - %H:%M:%S", localtime(curtime));
75      printf(" -> Date & Time : %s\n", buf);
76      printf(" -> Time zone   : GMT%+i\n\n", gmt_time);
77
78      return 0;
79  }
```

/NUCLEO_H723ZG_MQTT_TLS/Application/include/sw/ntp_client.h

```
1   // ------------------------------------------------------------------------------
2   // NTP_CLIENT
3   //
4   // - Archivo de cabecera.
5   //
6   // ------------------------------------------------------------------------------
7   // Copyright (c) 2021 Jorge Botana Mtz. de Ibarreta
8   //
9   // Este archivo se encuentra bajo los términos de la Licencia MIT. Debería
10  // haberse proporcionado una copia de ella junto a este fichero. Si no es así, se
11  // puede encontrar en el siguiente enlace:
12  //
13  //                                          https://opensource.org/licenses/MIT
14  // ------------------------------------------------------------------------------
15
16  #ifndef   NTP_CLIENT_H
17  #define   NTP_CLIENT_H
18
19  #include <time.h>
20
21  int NTP_Request(const char *hostname, const char *ipaddr, const int gmt_time,
        time_t *curtime);
22
23  #endif // NTP_CLIENT_H
```

B.3.7. TLS_CLIENT

/NUCLEO_H723ZG_MQTT_TLS/Application/source/sw/tls_client.c

```
1   // ------------------------------------------------------------------------------
2   // TLS_CLIENT
3   //
4   // - Cliente TLS 1.2.
5   //
6   // - Comprueba si el certificado del servidor es válido y, en caso de que falle
```

```
7  //   la verificación, el handshake TLS es abortado al no poder autenticarse el
8  //   servidor.
9  //
10 // - Permite añadir su propio certificado y clave privada, por si el servidor
11 //   requiriese de la autenticación del cliente.
12 //
13 // - Requiere de:
14 //
15 //   -> Un archivo de configuración.
16 //   -> Un archivo del que cargar certificados y claves privadas.
17 //   -> Una implementación externa de un RNG.
18 //   -> Una implementación externa de un RTC.
19 //   -> TCP_CLIENT.
20 //   -> El stack de TLS mbedTLS.
21 //
22 // - Algunas dependencias han de definirse en símbolos de la configuración del
23 //   proyecto:
24 //
25 //   -> Symbol : X_FILE_OR_IMPLEMENTATION
26 //   -> Value  : filename.h
27 //
28 // ----------------------------------------------------------------------------
29 // Copyright (c) 2021 Jorge Botana Mtz. de Ibarreta
30 //
31 // Este archivo se encuentra bajo los términos de la Licencia MIT. Debería
32 // haberse proporcionado una copia de ella junto a este fichero. Si no es así, se
33 // puede encontrar en el siguiente enlace:
34 //
35 //                                          https://opensource.org/licenses/MIT
36 // ----------------------------------------------------------------------------
37
38
39 #include "tls_client.h"
40
41 #ifndef   TLS_CLIENT_CONFIG_FILE
42 #error   "TLS_CLIENT_CONFIG_FILE                no definido en el proyecto !!!"
43 #else  // TLS_CLIENT_CONFIG_FILE
44 #include  TLS_CLIENT_CONFIG_FILE
45 // Comprobar:
46 // - Si está definido   USE_CLIENT_AUTH
47 // - Valor de           TLS_DEBUG_LEVEL
48 #endif // TLS_CLIENT_CONFIG_FILE
49
50 #ifndef   CERTIFICATES_FILE
51 #error   "CERTIFICATES_FILE                     no definido en el proyecto !!!"
52 #else  // CERTIFICATES_FILE
53 #include  CERTIFICATES_FILE
```

```
54 // Definir y establecer el certificado del CA de confianza usado para autenticar
55 // el servidor:
56 // - CA_CRT
57 // Si se habilita la autenticación del cliente, también se ha de definir y
58 // establecer su propio certificado y clave privada:
59 // - CLIENT_CRT
60 // - CLIENT_KEY
61 #endif // CERTIFICATES_FILE
62
63 #ifndef   RNG_IMPLEMENTATION
64 #error   "RNG_IMPLEMENTATION                    no definido en el proyecto !!!"
65 #else  // RNG_IMPLEMENTATION
66 #include  RNG_IMPLEMENTATION
67 // Función requerida: unsigned int RNG_Get(void);
68 #endif // RNG_IMPLEMENTATION
69
70 #ifndef   RTC_IMPLEMENTATION
71 #error   "RTC_IMPLEMENTATION                    no definido en el proyecto !!!"
72 #else  // RTC_IMPLEMENTATION
73 #include  RTC_IMPLEMENTATION
74 // Función requerida: time_t RTC_Read(void);
75 #endif // RTC_IMPLEMENTATION
76
77 #ifndef   MBEDTLS_CONFIG_FILE
78 #error   "MBEDTLS_CONFIG_FILE                    no definido en el proyecto !!!"
79 #else  // MBEDTLS_CONFIG_FILE
80 // Fichero personalizado de configuración de mbedTLS que se adjunta con este
81 // cliente. Aunque no sea una dependencia directa del cliente, igualmente es
82 // requerido por él.
83 #endif // MBEDTLS_CONFIG_FILE
84
85 #include "tcp_client.h"
86
87 #include "mbedtls/ctr_drbg.h"
88 #if       TLS_DEBUG_LEVEL > 0
89 #include "mbedtls/debug.h"
90 #endif // TLS_DEBUG_LEVEL > 0
91 #include "mbedtls/entropy.h"
92 #include "mbedtls/error.h"
93 #include "mbedtls/net_sockets.h"
94 #include "mbedtls/version.h"
95
96 #include <stdio.h>
97 #include <string.h>
98
99
100 mbedtls_ssl_context      tls      = {0};
```

```
101 mbedtls_ssl_config       conf     = {0};
102 mbedtls_x509_crt         ca_crt   = {0};
103 #ifdef    USE_CLIENT_AUTH
104 mbedtls_x509_crt         cli_crt  = {0};
105 mbedtls_pk_context       cli_key  = {0};
106 #endif // USE_CLIENT_AUTH
107 mbedtls_ctr_drbg_context ctr_drbg = {0};
108 mbedtls_entropy_context  entropy  = {0};
109
110
111 static mbedtls_time_t RTC_Time(mbedtls_time_t *time);
112 static int CRT_Verify_Result(void *data, mbedtls_x509_crt *crt, int depth,
        uint32_t *flags);
113 static void CRT_Info(mbedtls_x509_crt *crt);
114 #if       TLS_DEBUG_LEVEL > 0
115 static void mbedTLS_Debug(void *ctx, int level, const char *file, int line, const
         char *str);
116 #endif // TLS_DEBUG_LEVEL > 0
117 static int Error_Code(int ret);
118
119
120 int TLS_Init(const char *hostname, const char *ipaddr, const int port)
121 {
122     // Establecer una conexión TCP.
123     int ret = -1;
124     if((ret = TCP_Init(hostname, ipaddr, port)) < 0)
125     {
126         return ret;
127     }
128
129     printf(" ---------- Cliente TLS ----------\n\n");
130
131     // Mostrar la versión de mbedTLS.
132     printf(" - %s\n\n", MBEDTLS_VERSION_STRING_FULL);
133
134     // Inicializar las variables de mbedTLS.
135     printf(" - Inicializando los datos de la sesión TLS ......");
136     fflush(stdout);
137     mbedtls_ssl_init       (&tls     );
138     mbedtls_ssl_config_init(&conf    );
139     mbedtls_x509_crt_init  (&ca_crt  );
140 #ifdef    USE_CLIENT_AUTH
141     mbedtls_x509_crt_init  (&cli_crt );
142     mbedtls_pk_init        (&cli_key );
143 #endif // USE_CLIENT_AUTH
144     mbedtls_ctr_drbg_init  (&ctr_drbg);
145     mbedtls_entropy_init   (&entropy );
```

```
146     printf(" ok\n");
147 
148     // Establecer el callback de lectura del RTC.
149     mbedtls_platform_set_time(RTC_Time);
150 
151 #if       TLS_DEBUG_LEVEL > 0
152     // Establecer el nivel de depuración de mbedTLS.
153     mbedtls_debug_set_threshold(TLS_DEBUG_LEVEL);
154 #endif // TLS_DEBUG_LEVEL > 0
155 
156     // Generar la semilla aleatoria con el RNG.
157     printf(" - Generando la semilla aleatoria ................");
158     fflush(stdout);
159     if((ret = mbedtls_ctr_drbg_seed(&ctr_drbg, mbedtls_entropy_func, &entropy,
    NULL, 0)) < 0)
160     {
161         printf(" error !!!\n\n");
162         printf(" - mbedtls_ctr_drbg_seed() returned -0x%04X\n", -ret);
163         return Error_Code(ret);
164     }
165     printf(" ok\n");
166 
167     // Cargar el certificado del CA de confianza usado para autenticar el
168     // servidor. Normalmente es un certificado raíz.
169     printf(" - Cargando el certificado del CA de confianza ...");
170     fflush(stdout);
171     if((ret = mbedtls_x509_crt_parse(&ca_crt , (const unsigned char *)CA_CRT,
    sizeof(CA_CRT))) < 0)
172     {
173         printf(" error !!!\n\n");
174         printf(" - mbedtls_x509_crt_parse() returned -0x%04X\n", -ret);
175         return Error_Code(ret);
176     }
177     printf(" ok\n");
178 
179 #ifdef    USE_CLIENT_AUTH
180     // Cargar el certificado del cliente.
181     printf(" - Cargando el certificado del cliente ...........");
182     fflush(stdout);
183     if((ret = mbedtls_x509_crt_parse(&cli_crt, (const unsigned char *)CLIENT_CRT,
     sizeof(CLIENT_CRT))) < 0)
184     {
185         printf(" error !!!\n\n");
186         printf(" - mbedtls_x509_crt_parse() returned -0x%04X\n", -ret);
187         return Error_Code(ret);
188     }
189     printf(" ok\n");
```

```
190
191     // Cargar la clave privada del cliente.
192     printf(" - Cargando la clave privada del cliente .........");
193     fflush(stdout);
194     if((ret = mbedtls_pk_parse_key(&cli_key, (const unsigned char *)CLIENT_KEY,
        sizeof(CLIENT_KEY), NULL, 0)) < 0)
195     {
196         printf(" error !!!\n\n");
197         printf(" - mbedtls_pk_parse_key() returned -0x%04X\n", -ret);
198         return Error_Code(ret);
199     }
200     printf(" ok\n");
201 #endif // USE_CLIENT_AUTH
202
203     // Comenzar a crear la estructura TLS.
204     printf(" - Creando la estructura TLS ......................");
205     fflush( stdout );
206
207     // Inicializar una configuración TLS predeterminada para un cliente TLS.
208     if((ret = mbedtls_ssl_config_defaults(&conf, MBEDTLS_SSL_IS_CLIENT,
        MBEDTLS_SSL_TRANSPORT_STREAM, MBEDTLS_SSL_PRESET_DEFAULT)) < 0)
209     {
210         printf(" error !!!\n\n");
211         printf(" - mbedtls_ssl_config_defaults() returned -0x%04X\n", -ret);
212         return Error_Code(ret);
213     }
214
215     // En la configuración TLS, establecer el callback que muestra durante el
216     // handshake TLS el resultado de la verificación de la cadena de confianza
217     // del servidor.
218     mbedtls_ssl_conf_verify(&conf, CRT_Verify_Result, NULL);
219
220     // En la configuración TLS, requerir de la autenticación del servidor para
221     // que el handshake TLS sea satisfactorio.
222     mbedtls_ssl_conf_authmode(&conf, MBEDTLS_SSL_VERIFY_REQUIRED);
223
224     // En la configuración TLS, establecer el callback de generación de números
225     // aleatorios a partir de la semilla aleatoria ya creada.
226     mbedtls_ssl_conf_rng(&conf, mbedtls_ctr_drbg_random, &ctr_drbg);
227
228 #if       TLS_DEBUG_LEVEL > 0
229     // En la configuración TLS, establecer el callback de depuración de mbedTLS.
230     mbedtls_ssl_conf_dbg(&conf, mbedTLS_Debug, NULL);
231 #endif // TLS_DEBUG_LEVEL > 0
232
233     // En la configuración TLS, establecer el certificado del CA de confianza
234     // usado para autenticar el servidor.
```

```
235     mbedtls_ssl_conf_ca_chain(&conf, &ca_crt, NULL);
236
237 #ifdef    USE_CLIENT_AUTH
238     // En la configuración TLS, asociar el certificado del cliente con su clave
239     // privada.
240     if((ret = mbedtls_ssl_conf_own_cert(&conf, &cli_crt, &cli_key)) < 0)
241     {
242         printf(" error !!!\n\n");
243         printf(" - mbedtls_ssl_conf_own_cert() returned -0x%04X\n", -ret);
244         return Error_Code(ret);
245     }
246 #endif // USE_CLIENT_AUTH
247
248     // En el contexto TLS, cargar la configuración TLS.
249     if((ret = mbedtls_ssl_setup(&tls, &conf)) < 0)
250     {
251         printf(" error !!!\n\n");
252         printf(" - mbedtls_ssl_setup() returned -0x%04X\n", -ret);
253         return Error_Code(ret);
254     }
255
256     // En el contexto TLS, establecer el hostname del servidor. Debe aparecer en
257     // su certificado.
258     if((ret = mbedtls_ssl_set_hostname(&tls, hostname)) < 0)
259     {
260         printf(" error !!!\n\n");
261         printf(" - mbedtls_ssl_set_hostname() returned -0x%04X\n", -ret);
262         return Error_Code(ret);
263     }
264
265     // En el contexto TLS, establecer el socket TCP y los callbacks de envío y
266     // recepción.
267     mbedtls_ssl_set_bio(&tls, &s, mbedtls_net_send, mbedtls_net_recv, NULL);
268
269     // Terminar de crear la estructura TLS.
270     printf(" ok\n");
271
272     // Inicializar el handshake TLS.
273     printf(" - Realizando el handshake TLS .....................\n\n");
274     if((ret = mbedtls_ssl_handshake(&tls)) < 0)
275     {
276         printf(" - mbedtls_ssl_handshake() returned -0x%04X\n", -ret);
277         return Error_Code(ret);
278     }
279
280 #ifdef    USE_CLIENT_AUTH
281     // Mostrar el certificado del cliente.
```

```
282     printf(" - Certificado del cliente\n");
283     CRT_Info(&cli_crt);
284     printf(" - Ok: validado por el servidor\n\n");
285 #endif // USE_CLIENT_AUTH
286 
287     // Salir tras completar el handshake TLS.
288     printf(" - Handshake TLS satisfactorio\n\n");
289 
290     return 0;
291 }
292 
293 
294 int TLS_Close(char *buf, size_t len)
295 {
296     // Liberar los recursos empleados por mbedTLS.
297     mbedtls_ssl_free       (&tls     );
298     mbedtls_ssl_config_free(&conf    );
299     mbedtls_x509_crt_free  (&ca_crt  );
300 #ifdef    USE_CLIENT_AUTH
301     mbedtls_x509_crt_free  (&cli_crt );
302     mbedtls_pk_free        (&cli_key );
303 #endif // USE_CLIENT_AUTH
304     mbedtls_ctr_drbg_free  (&ctr_drbg);
305     mbedtls_entropy_free   (&entropy );
306 
307     // Realizar el handshake de desconexión TCP.
308     return TCP_Close(buf, len);
309 }
310 
311 
312 int TLS_Read(char *buf, size_t len)
313 {
314     // Leer datos ya desencriptados por mbedTLS.
315     return mbedtls_ssl_read(&tls, (unsigned char *)buf, len);
316 }
317 
318 
319 int TLS_Write(const char *buf, size_t len)
320 {
321     // Escribir datos que serán encriptados por mbedTLS.
322     return mbedtls_ssl_write(&tls, (const unsigned char *)buf, len);
323 }
324 
325 
326 int mbedtls_hardware_poll(void *data, unsigned char *output, size_t len, size_t *
    olen)
327 {
```

```
328     // Evitar "warnings" en el compilador debido a argumentos sin usar.
329     (void)data;
330 
331     // Generar la semilla aleatoria con el RNG. Se generan y guardan 4 bytes por
332     // iteración, hasta completar la semilla.
333     int i = 0;
334     for(i = 0; i < len / 4; i++)
335     {
336         unsigned int random = RNG_Get();
337         *olen = *olen + 4;
338         memset(&(output[4 * i]), (int)random, 4);
339     }
340 
341     return 0;
342 }
343 
344 
345 int mbedtls_net_recv(void *ctx, unsigned char *buf, size_t len)
346 {
347     // Evitar "warnings" en el compilador debido a argumentos sin usar.
348     (void)ctx;
349 
350     // Leer datos encriptados.
351     return TCP_Read((char *)buf, len);
352 }
353 
354 
355 int mbedtls_net_send(void *ctx, const unsigned char *buf, size_t len)
356 {
357     // Evitar "warnings" en el compilador debido a argumentos sin usar.
358     (void)ctx;
359 
360     // Escribir datos encriptados.
361     return TCP_Write((const char *)buf, len);
362 }
363 
364 
365 static mbedtls_time_t RTC_Time(mbedtls_time_t *time)
366 {
367     // Evitar "warnings" en el compilador debido a argumentos sin usar.
368     (void)time;
369 
370     // Para verificar que funciona la comprobación de la fecha y hora de emisión
371     // y expiración de los certificados, o para establecerla manualmente, se
372     // puede sustituir RTC_Read() por el timestamp correspondiente a una fecha y
373     // hora concreta (GMT 0).
374 
```

```
375     // return 1577836800; // Comienzo del año 2020.
376
377     // return 1735689600; // Comienzo del año 2025.
378
379     // return 2208988800; // Comienzo del año 2040.
380
381     // Obtener el timestamp.
382     return RTC_Read();
383 }
384
385
386 static int CRT_Verify_Result(void *data, mbedtls_x509_crt *crt, int depth,
        uint32_t *flags)
387 {
388     // Evitar "warnings" en el compilador debido a argumentos sin usar.
389     (void)data;
390     (void)depth;
391     (void)flags;
392
393     // Construir la cadena de confianza del servidor. Antes del handshake, el
394     // certificado del CA de confianza es añadido con mbedtls_x509_crt_parse() y
395     // es establecido como certificado de confianza con
396     // mbedtls_ssl_conf_ca_chain().
397
398     // Este certificado del CA de confianza normalmente es un certificado raíz.
399     // El resto de certificados de la cadena de confianza deberían ser
400     // proporcionados por el servidor durante el handshake TLS. Es posible que el
401     // servidor también proporcione el certificado raíz, aunque lo mismo da
402     // porque, para que sea de confianza para el cliente, este ya debe contar con
403     // él antes del handshake TLS.
404
405     // Tras construir la cadena de confianza del servidor, la verificación se
406     // realiza desde el certificado del servidor hasta el certificado del CA de
407     // confianza. Luego se muestra la cadena de confianza del servidor con el
408     // resultado de la verificación.
409
410     // Nota: mbedTLS permite instalar un certificado de un CA intermedio como
411     // certificado de confianza sin el certificado raíz. De todas formas, esto no
412     // es muy lógico y en otras implementaciones de TLS podría no estar
413     // permitido, al intentarse seguir la cadena hacia el certificado raíz.
414
415     printf(" - Cadena de confianza del servidor (depth = %i)\n", depth);
416
417     // Mostrar la información del certificado.
418     CRT_Info(crt);
419
420     // Mostrar el resultado de la verificación.
```

```
421     if(*flags == 0)
422     {
423         printf(" - Ok: verificación correcta\n\n");
424     }
425     else
426     {
427         char buf[2048] = {0};
428         mbedtls_x509_crt_verify_info(buf, sizeof(buf), " - Error: ", *flags);
429         printf("%s\n", buf);
430         fflush(stdout);
431     }
432 
433     return 0;
434 }
435 
436 
437 static void CRT_Info(mbedtls_x509_crt *crt)
438 {
439     // Mostrar la información del certificado.
440     char buf[2048] = {0};
441     mbedtls_x509_crt_info(buf, sizeof(buf), " -> ", crt);
442     printf("%s", buf);
443 }
444 
445 
446 #if       TLS_DEBUG_LEVEL > 0
447 
448 static void mbedTLS_Debug(void *ctx, int level, const char *file, int line, const
        char *str)
449 {
450     // Evitar "warnings" en el compilador debido a argumentos sin usar.
451     (void)(ctx  );
452     (void)(level);
453 
454     // Mostrar el mensaje de depuración (archivo, línea, mensaje).
455     printf(" - File: %s\n", file);
456     printf(" - Line: %i\n", line);
457     printf(" - Msg : %s\n", str );
458 }
459 
460 #endif // TLS_DEBUG_LEVEL > 0
461 
462 
463 static int Error_Code(int ret)
464 {
465     // Mostrar el mensaje de error correspondiente a un código de error.
466     char buf[2048] = {0};
```

```
467     mbedtls_strerror(ret, buf, sizeof(buf));
468     printf(" -> %s\n\n", buf);
469 
470     return ret;
471 }
```

/NUCLEO_H723ZG_MQTT_TLS/Application/include/sw/tls_client.h

```
1  // ------------------------------------------------------------------------------
2  // TLS_CLIENT
3  //
4  // - Archivo de cabecera.
5  //
6  // ------------------------------------------------------------------------------
7  // Copyright (c) 2021 Jorge Botana Mtz. de Ibarreta
8  //
9  // Este archivo se encuentra bajo los términos de la Licencia MIT. Debería
10 // haberse proporcionado una copia de ella junto a este fichero. Si no es así, se
11 // puede encontrar en el siguiente enlace:
12 //
13 //                                          https://opensource.org/licenses/MIT
14 // ------------------------------------------------------------------------------
15 
16 #ifndef   TLS_CLIENT_H
17 #define   TLS_CLIENT_H
18 
19 #include <stddef.h>
20 
21 int TLS_Init(const char *hostname, const char *ipaddr, const int port);
22 int TLS_Close(char *buf, size_t len);
23 
24 int TLS_Read(char *buf, size_t len);
25 int TLS_Write(const char *buf, size_t len);
26 
27 #endif // TLS_CLIENT_H
```

B.3.8. MQTT_CLIENT

/NUCLEO_H723ZG_MQTT_TLS/Application/source/sw/mqtt_client.c

```
1  // ------------------------------------------------------------------------------
2  // MQTT_CLIENT
3  //
4  // - Cliente MQTT v5.0 (QoS 0).
5  //
6  // - Permite conexiones con y sin TLS.
```

```
7  //
8  // - Paquetes MQTT implementados:
9  //
10 //   -> 00 - Reserved   - No          -> 08 - SUBSCRIBE   - Sí ( W)
11 //   -> 01 - CONNECT    - Sí ( W)     -> 09 - SUBACK      - Sí (R )
12 //   -> 02 - CONNACK    - Sí (R )     -> 10 - UNSUBSCRIBE - Sí ( W)
13 //   -> 03 - PUBLISH    - Sí (RW)     -> 11 - UNSUBACK    - Sí (R )
14 //   -> 04 - PUBACK     - No          -> 12 - PINGREQ     - Sí ( W)
15 //   -> 05 - PUBREC     - No          -> 13 - PINGRESP    - Sí (R )
16 //   -> 06 - PUBREL     - No          -> 14 - DISCONNECT  - Sí ( W)
17 //   -> 07 - PUBCOMP    - No          -> 15 - AUTH        - No
18 //
19 // - Requiere de:
20 //
21 //   -> Un archivo de configuración.
22 //   -> Una implementación externa de un temporizador para hacer tests midiendo
23 //      tiempos de escritura y lectura de paquetes MQTT. Si no se van a hacer
24 //      estos tests, entonces no es necesario el temporizador.
25 //   -> TLS_CLIENT (si se usa una conexión TLS).
26 //   -> TCP_CLIENT (si se usa una conexión insegura).
27 //
28 // - Algunas dependencias han de definirse en símbolos de la configuración del
29 //   proyecto:
30 //
31 //   -> Symbol : X_FILE_OR_IMPLEMENTATION
32 //   -> Value  : filename.h
33 //
34 // - No emplea librerías MQTT de terceros.
35 //
36 // ------------------------------------------------------------------------------
37 // Copyright (c) 2021 Jorge Botana Mtz. de Ibarreta
38 //
39 // Este archivo se encuentra bajo los términos de la Licencia MIT. Debería
40 // haberse proporcionado una copia de ella junto a este fichero. Si no es así, se
41 // puede encontrar en el siguiente enlace:
42 //
43 //                                          https://opensource.org/licenses/MIT
44 // ------------------------------------------------------------------------------
45
46
47 #include "mqtt_client.h"
48
49 #ifndef   MQTT_CLIENT_CONFIG_FILE
50 #error   "MQTT_CLIENT_CONFIG_FILE               no definido en el proyecto !!!"
51 #else  // MQTT_CLIENT_CONFIG_FILE
52 #include  MQTT_CLIENT_CONFIG_FILE
53 // Comprobar:
```

```
54 // - Si está definido   TEST_MODE
55 // - Si está definido   USE_TLS
56 #endif // MQTT_CLIENT_CONFIG_FILE
57
58 #ifdef    TEST_MODE
59 #ifndef   TIM_IMPLEMENTATION
60 #error   "TIM_IMPLEMENTATION                    no definido en el proyecto !!!"
61 #else  // TIM_IMPLEMENTATION
62 #include  TIM_IMPLEMENTATION
63 // Funciones requeridas: void TIM_Start(void);
64 //                       void TIM_Stop(void);
65 #endif // TIM_IMPLEMENTATION
66 #endif // TEST_MODE
67
68 #ifdef    USE_TLS
69 #include "tls_client.h"
70 #else  // USE_TLS
71 #include "tcp_client.h"
72 #endif // USE_TLS
73
74 #include <math.h>
75 #include <stdio.h>
76 #include <stdlib.h>
77
78
79 #define   FIXED_HEADER_MIN_SIZE                                          2
80
81 #define   CONNECT_MIN_SIZE                                              15
82 #define   DISCONNECT_MIN_SIZE                                            2
83 #define   PUBLISH_MIN_SIZE                                               5
84 #define   SUBSCRIBE_MIN_SIZE                                             8
85 #define   UNSUBSCRIBE_MIN_SIZE                                           7
86 #define   PINGREQ_MIN_SIZE                                               2
87
88
89 static int MQTT_Read_CONNACK(char *buf, size_t ret);
90 static int MQTT_Read_PUBLISH(char *buf, size_t ret);
91 static int MQTT_Read_SUBACK(char *buf, size_t ret);
92 static int MQTT_Read_UNSUBACK(char *buf, size_t ret);
93 static int MQTT_Read_PINGRESP(char *buf, size_t ret);
94
95 static int MQTT_Write_Packet(const char *buf, size_t len);
96
97 static int MQTT_Generate_Packet_ID(void);
98
99
100 int MQTT_Init(const char *hostname, const char *ipaddr, const int port)
```

```
101 {
102     printf(" ---------- Cliente MQTT ----------\n\n");
103 
104     // Realizar el handshake de conexión.
105 #ifdef    USE_TLS
106     return TLS_Init(hostname, ipaddr, port);
107 #else  // USE_TLS
108     return TCP_Init(hostname, ipaddr, port);
109 #endif // USE_TLS
110 }
111 
112 
113 int MQTT_Close(char *buf, size_t len)
114 {
115     // Realizar el handshake de desconexión.
116 #ifdef    USE_TLS
117     return TLS_Close(buf, len);
118 #else  // USE_TLS
119     return TCP_Close(buf, len);
120 #endif // USE_TLS
121 }
122 
123 
124 int MQTT_Read_Packet(char *buf, size_t len)
125 {
126     printf(" - Esperando paquetes .............\n");
127 
128 #ifdef    TEST_MODE
129     // Iniciar el temporizador.
130     TIM_Start();
131 #endif // TEST_MODE
132 
133     // Leer paquetes entrantes.
134 #ifdef    USE_TLS
135     int ret = TLS_Read(buf, len);
136 #else
137     int ret = TCP_Read(buf, len);
138 #endif // USE_TLS
139 
140 #ifdef    TEST_MODE
141     // Detener el temporizador.
142     TIM_Stop();
143 #endif // TEST_MODE
144 
145     // Procesar el paquete leído, si lo hay.
146     if(ret < 0)
147     {
```

```
148         printf(" - Ningún paquete recibido\n");
149     }
150     else
151     {
152         // Determinar el tipo de paquete leído, que viene definido por el nibble
153         // superior del primer byte. A continuación, llamar a la función
154         // correspondiente.
155 
156         char packet_type = buf[0] & 0b11110000;
157 
158         printf(" - Recibido ");
159 
160         if      (packet_type == 0b00100000)
161         {
162             printf("CONNACK\n" );
163             ret = MQTT_Read_CONNACK (buf, ret);
164         }
165         else if(packet_type == 0b00110000)
166         {
167             printf("PUBLISH\n" );
168             ret = MQTT_Read_PUBLISH (buf, ret);
169         }
170         else if(packet_type == 0b10010000)
171         {
172             printf("SUBACK\n"  );
173             ret = MQTT_Read_SUBACK  (buf, ret);
174         }
175         else if(packet_type == 0b10110000)
176         {
177             printf("UNSUBACK\n");
178             ret = MQTT_Read_UNSUBACK(buf, ret);
179         }
180         else if(packet_type == 0b11010000)
181         {
182             printf("PINGRESP\n");
183             ret = MQTT_Read_PINGRESP(buf, ret);
184         }
185         else
186         {
187             printf("paquete desconocido !!!\n\n");
188             ret = -1;
189         }
190     }
191 
192     return ret;
193 }
194 
```

```
195
196 static int MQTT_Read_CONNACK(char *buf, size_t ret)
197 {
198     // Leer el "Reason Code". Es el cuarto byte del paquete CONNACK. Si vale 0,
199     // es que la conexión ha sido aceptada.
200     if(buf[3] == 0)
201     {
202         printf(" - Conexión aceptada\n\n");
203
204         return ret;
205     }
206     else
207     {
208         printf(" - Error de conexión !!!\n\n");
209
210         return -1;
211     }
212 }
213
214
215 static int MQTT_Read_PUBLISH(char *buf, size_t ret)
216 {
217     // Calcular la longitud del mensaje recibido. Explicación:
218
219     // - ret es el tamaño total del paquete PUBLISH.
220     size_t m_len = ret;
221
222     // - Restar 1 byte del campo "Header Flags".
223     m_len--;
224
225     // - Restar n bytes del campo "Remaining Length". Deja de haber bytes cuando
226     //   se encuentra un 1 en el MSB de un byte.
227     int n = 0;
228     do
229     {
230         n++;
231     }
232     while((buf[n] & 0b10000000) != 0);
233     m_len = m_len - n;
234
235     // - Restar 2 bytes del campo "Topic Length".
236     m_len = m_len - 2;
237
238     // - Restar el número de bytes del tema.
239     size_t t_len = 128 * buf[n + 1] + buf[n + 2];
240     m_len = m_len - t_len;
241
```

```
242     // - Restar 1 byte del campo "Properties" -> "Total Length".
243     m_len = m_len - 1;
244 
245     // - Restar p bytes de propiedades.
246     size_t p = buf[n + 2 + t_len + 1];
247     m_len = m_len - p;
248 
249     // Desplazar el mensaje (contenido al final del paquete) al comienzo del
250     // buffer.
251     int i = 0;
252     for(i = 0; i < m_len; i++)
253     {
254         buf[i] = buf[i + (ret - m_len)];
255     }
256 
257     // Marcar el final del mensaje recibido con un NULL.
258     buf[i] = 0;
259 
260     // Mostrar el mensaje recibido.
261     printf(" <- %s\n\n", buf);
262 
263     return ret;
264 }
265 
266 
267 static int MQTT_Read_SUBACK(char *buf, size_t ret)
268 {
269     // Comprobar si la suscripción ha sido aceptada. Para ello hay que examinar
270     // el último byte recibido del paquete SUBACK (para "ret" bytes recibidos, se
271     // construye el buffer desde la posición 0 hasta la posición ret - 1).
272     if(buf[ret - 1] == 0)
273     {
274         printf(" - Suscripción aceptada : QoS 0\n\n");
275 
276         return ret;
277     }
278     else
279     {
280         printf(" - Error de conexión !!!\n\n");
281 
282         return -1;
283     }
284 }
285 
286 
287 static int MQTT_Read_UNSUBACK(char *buf, size_t ret)
288 {
```

```
289     // Comprobar si la desuscripción ha sido aceptada. Para ello hay que examinar
290     // el último byte recibido del paquete UNSUBACK (para "ret" bytes recibidos,
291     // se construye el buffer desde la posición 0 hasta la posición ret - 1).
292     if(buf[ret - 1] == 0)
293     {
294         printf(" - Desuscripción aceptada\n\n");
295 
296         return ret;
297     }
298     else
299     {
300         printf(" - Error de conexión !!!\n\n");
301 
302         return -1;
303     }
304 }
305 
306 
307 static int MQTT_Read_PINGRESP(char *buf, size_t ret)
308 {
309     // Evitar "warnings" en el compilador debido a argumentos sin usar.
310     (void)buf;
311 
312     printf(" - La conexión sigue activa\n\n");
313 
314     return ret;
315 }
316 
317 
318 int MQTT_Write_CONNECT(const char *client, size_t c_len, const int keep_alive)
319 {
320     printf(" - Enviando CONNECT ............");
321     fflush(stdout);
322 
323     // Descontar el NULL de la longitud.
324     c_len--;
325 
326     // Calcular el número de bytes necesarios para codificar "Remaining Length".
327     size_t n     = 0;
328     size_t power = 0;
329     do
330     {
331         n++;
332         power = pow(128, n);
333     }
334     while((CONNECT_MIN_SIZE - FIXED_HEADER_MIN_SIZE + c_len) / power > 0);
335 
```

```
336     // Calcular la longitud total de este paquete CONNECT.
337     size_t connect_len = CONNECT_MIN_SIZE + (n - 1) + c_len;
338
339     // Alocar memoria para el paquete CONNECT.
340     char *CONNECT = calloc(connect_len, sizeof(char));
341
342     // - Header Flags
343     CONNECT[0] = 0b00010000;
344                //  7654|||| - Message Type: Connect Command (1)
345                //      3210 - Reserved: 0
346
347     // - Remaining Length
348     size_t rem_len = connect_len - FIXED_HEADER_MIN_SIZE - (n - 1);
349     int i = 0;
350     do
351     {
352         CONNECT[1 + i] = rem_len % 128;
353         rem_len        = rem_len / 128;
354         if(rem_len > 0)
355         {
356             CONNECT[1 + i] = CONNECT[1 + i] | 128;
357             i++;
358         }
359     }
360     while(rem_len > 0);
361
362     // - Protocol Name Length
363     CONNECT[ 2 + i] = 0; // MSB
364     CONNECT[ 3 + i] = 4; // LSB
365
366     // - Protocol Name
367     CONNECT[ 4 + i] = 'M';
368     CONNECT[ 5 + i] = 'Q';
369     CONNECT[ 6 + i] = 'T';
370     CONNECT[ 7 + i] = 'T';
371
372     // - Version
373     CONNECT[ 8 + i] = 5;
374
375     // - Connect Flags
376     CONNECT[ 9 + i] = 0b00000010;
377                     //  7|||||||  - User Name Flag: Not set
378                     //   6||||||  - Password Flag: Not set
379                     //    5|||||  - Will Retain: Not set
380                     //     43|||  - QoS Level: At most once delivery (Fire and
381                     //       |||    Forget) (0)
382                     //       2||  - Will Flag: Not set
```

```
383                     //        1| - Clean Session Flag: Set
384                     //         0 - (Reserved): Not set
385
386     // - Keep Alive
387     CONNECT[10 + i] = keep_alive / 256; // MSB
388     CONNECT[11 + i] = keep_alive % 256; // LSB
389
390     // - Properties
391     CONNECT[12 + i] = 0; // - Total Length
392         // La longitud es 0 porque no hay propiedades. En caso de haber
393         // propiedades, estas irían a continuación.
394
395     // - Client ID Length
396     CONNECT[13 + i] = c_len / 256; // MSB
397     CONNECT[14 + i] = c_len % 256; // LSB
398
399     // - Client ID
400     int j = 0;
401     for(j = 0; j < c_len; j++)
402     {
403         CONNECT[15 + i + j] = client[j];
404     }
405
406     // Enviar el paquete CONNECT y liberar la memoria alocada.
407     int ret = MQTT_Write_Packet(CONNECT, connect_len);
408     free(CONNECT);
409
410     return ret;
411 }
412
413
414 int MQTT_Write_PUBLISH(const char *topic, size_t t_len, const char *msg, size_t
        m_len)
415 {
416     printf(" - Enviando PUBLISH ............");
417     fflush(stdout);
418
419     // Descontar el NULL de las longitudes.
420     t_len--;
421     m_len--;
422
423     // Calcular el número de bytes necesarios para codificar "Remaining Length".
424     size_t n     = 0;
425     size_t power = 0;
426     do
427     {
428         n++;
```

```
429         power = pow(128, n);
430     }
431     while((PUBLISH_MIN_SIZE - FIXED_HEADER_MIN_SIZE + t_len + m_len) / power > 0)
        ;
432
433     // Calcular la longitud total de este paquete PUBLISH.
434     size_t publish_len = PUBLISH_MIN_SIZE + (n - 1) + t_len + m_len;
435
436     // Alocar memoria para el paquete PUBLISH.
437     char *PUBLISH = calloc(publish_len, sizeof(char));
438
439     // - Header Flags
440     PUBLISH[0] = 0b00110000;
441                //  7654|||| - Message Type: Publish Command (3)
442                //      3||| - DUP Flag: Not set
443                //       21| - QoS Level: At most once delivery (Fire and Forget)
444                //         |   (0)
445                //         0 - Retain: Not set
446
447     // - Remaining Length
448     size_t rem_len = publish_len - FIXED_HEADER_MIN_SIZE - (n - 1);
449     int i = 0;
450     do
451     {
452         PUBLISH[1 + i] = rem_len % 128;
453         rem_len        = rem_len / 128;
454         if(rem_len > 0)
455         {
456             PUBLISH[1 + i] = PUBLISH[1 + i] | 128;
457             i++;
458         }
459     }
460     while(rem_len > 0);
461
462     // - Topic Length
463     PUBLISH[2 + i] = t_len / 256; // MSB
464     PUBLISH[3 + i] = t_len % 256; // LSB
465
466     // - Topic
467     int j = 0;
468     for(j = 0; j < t_len; j++)
469     {
470         PUBLISH[4 + i + j] = topic[j];
471     }
472
473     // - Properties
474     PUBLISH[4 + i + j] = 0; // - Total Length
```

```
475         // La longitud es 0 porque no hay propiedades. En caso de haber
476         // propiedades, estas irían a continuación.
477 
478     // - Message
479     int k = 0;
480     for(k = 0; k < m_len; k++)
481     {
482         PUBLISH[5 + i + j + k] = msg[k];
483     }
484 
485     // Enviar el paquete PUBLISH y liberar la memoria alocada.
486     int ret = MQTT_Write_Packet(PUBLISH, publish_len);
487     free(PUBLISH);
488 
489     // Mostrar el mensaje enviado.
490     printf(" -> %s\n\n", msg);
491 
492     return ret;
493 }
494 
495 
496 int MQTT_Write_SUBSCRIBE(const char *topic, size_t t_len)
497 {
498     printf(" - Enviando SUBSCRIBE ..........");
499     fflush(stdout);
500 
501     // Descontar el NULL de la longitud.
502     t_len--;
503 
504     // Calcular el número de bytes necesarios para codificar "Remaining Length".
505     size_t n     = 0;
506     size_t power = 0;
507     do
508     {
509         n++;
510         power = pow(128, n);
511     }
512     while((SUBSCRIBE_MIN_SIZE - FIXED_HEADER_MIN_SIZE + t_len) / power > 0);
513 
514     // Calcular la longitud total de este paquete SUBSCRIBE.
515     size_t subscribe_len = SUBSCRIBE_MIN_SIZE + (n - 1) + t_len;
516 
517     // Alocar memoria para el paquete SUBSCRIBE.
518     char *SUBSCRIBE = calloc(subscribe_len, sizeof(char));
519 
520     // - Header Flags
521     SUBSCRIBE[0] = 0b10000010;
```

```
522                 //  7654|||| - Message Type: Subscribe Request (8)
523                 //      3210 - Reserved: 2
524
525     // - Remaining Length
526     size_t rem_len = subscribe_len - FIXED_HEADER_MIN_SIZE - (n - 1);
527     int i = 0;
528     do
529     {
530         SUBSCRIBE[1 + i] = rem_len % 128;
531         rem_len          = rem_len / 128;
532         if(rem_len > 0)
533         {
534             SUBSCRIBE[1 + i] = SUBSCRIBE[1 + i] | 128;
535             i++;
536         }
537     }
538     while(rem_len > 0);
539
540     // - Message Identifier
541     int packet_identifier = MQTT_Generate_Packet_ID();
542     SUBSCRIBE[2 + i] = packet_identifier / 128; // MSB
543     SUBSCRIBE[3 + i] = packet_identifier % 128; // LSB
544
545     // - Properties
546     SUBSCRIBE[4 + i] = 0; // - Total Length
547         // La longitud es 0 porque no hay propiedades. En caso de haber
548         // propiedades, estas irían a continuación.
549
550     // - Topic Length
551     SUBSCRIBE[5 + i] = t_len / 256; // MSB
552     SUBSCRIBE[6 + i] = t_len % 256; // LSB
553
554     // - Topic
555     int j = 0;
556     for(j = 0; j < t_len; j++)
557     {
558         SUBSCRIBE[7 + i + j] = topic[j];
559     }
560
561     // - Subscription Options
562     SUBSCRIBE[7 + i + j] = 0b00000000;
563                         //  76|||||| - Reserved: 0x0
564                         //    54|||| - Retain Handling: Send msgs at
565                         //      |||   subscription time (0)
566                         //      3||| - Retain as Published: Not set
567                         //       2|| - No Local: Not set
568                         //        10 - QoS: At most once delivery (Fire and
```

```
569                         //            Forget) (0)
570
571     // Enviar el paquete SUBSCRIBE y liberar la memoria alocada.
572     int ret = MQTT_Write_Packet(SUBSCRIBE, subscribe_len);
573     free(SUBSCRIBE);
574
575     return ret;
576 }
577
578
579 int MQTT_Write_UNSUBSCRIBE(const char *topic, size_t t_len)
580 {
581     printf(" - Enviando UNSUBSCRIBE ........");
582     fflush(stdout);
583
584     // Descontar el NULL de la longitud.
585     t_len--;
586
587     // Calcular el número de bytes necesarios para codificar "Remaining Length".
588     size_t n     = 0;
589     size_t power = 0;
590     do
591     {
592         n++;
593         power = pow(128, n);
594     }
595     while((UNSUBSCRIBE_MIN_SIZE - FIXED_HEADER_MIN_SIZE + t_len) / power > 0);
596
597     // Calcular la longitud total de este paquete UNSUBSCRIBE.
598     size_t unsubscribe_len = UNSUBSCRIBE_MIN_SIZE + (n - 1) + t_len;
599
600     // Alocar memoria para el paquete UNSUBSCRIBE.
601     char *UNSUBSCRIBE = calloc(unsubscribe_len, sizeof(char));
602
603     // - Header Flags
604     UNSUBSCRIBE[0] = 0b10100010;
605                     //  7654|||| - Message Type: Unsubscribe Request (10)
606                     //      3210 - Reserved: 2
607
608     // - Remaining Length
609     size_t rem_len = unsubscribe_len - FIXED_HEADER_MIN_SIZE - (n - 1);
610     int i = 0;
611     do
612     {
613         UNSUBSCRIBE[1 + i] = rem_len % 128;
614         rem_len            = rem_len / 128;
615         if(rem_len > 0)
```

```
616         {
617             UNSUBSCRIBE[1 + i] = UNSUBSCRIBE[1 + i] | 128;
618             i++;
619         }
620     }
621     while(rem_len > 0);
622 
623     // - Message Identifier
624     int packet_identifier = MQTT_Generate_Packet_ID();
625     UNSUBSCRIBE[2 + i] = packet_identifier / 128; // MSB
626     UNSUBSCRIBE[3 + i] = packet_identifier % 128; // LSB
627 
628     // - Properties
629     UNSUBSCRIBE[4 + i] = 0; // - Total Length
630         // La longitud es 0 porque no hay propiedades. En caso de haber
631         // propiedades, estas irían a continuación.
632 
633     // - Topic Length
634     UNSUBSCRIBE[5 + i] = t_len / 256; // MSB
635     UNSUBSCRIBE[6 + i] = t_len % 256; // LSB
636 
637     // - Topic
638     int j = 0;
639     for(j = 0; j < t_len; j++)
640     {
641         UNSUBSCRIBE[7 + i + j] = topic[j];
642     }
643 
644     // Enviar el paquete UNSUBSCRIBE y liberar la memoria alocada.
645     int ret = MQTT_Write_Packet(UNSUBSCRIBE, unsubscribe_len);
646     free(UNSUBSCRIBE);
647 
648     return ret;
649 }
650 
651 
652 int MQTT_Write_PINGREQ(void)
653 {
654     printf(" - Enviando PINGREQ .....");
655     fflush(stdout);
656 
657     // Inicializar memoria para el paquete PINGREQ.
658     char PINGREQ[PINGREQ_MIN_SIZE] = {0};
659 
660     // - Header Flags
661     PINGREQ[0] = 0b11000000;
662                 //  7654|||| - Message Type: Ping Request (12)
```

```
663                 //      3210 - Reserved: 0
664
665     // - Remaining Length
666     PINGREQ[1] = 0;
667
668     // Enviar el paquete PINGREQ.
669     return MQTT_Write_Packet(PINGREQ, PINGREQ_MIN_SIZE);
670 }
671
672
673 int MQTT_Write_DISCONNECT(void)
674 {
675     printf(" - Enviando DISCONNECT .........");
676     fflush(stdout);
677
678     // Inicializar memoria para el paquete DISCONNECT.
679     char DISCONNECT[DISCONNECT_MIN_SIZE] = {0};
680
681     // - Header Flags
682     DISCONNECT[0] = 0b11100000;
683                   //  7654|||| - Message Type: Disconnect Req (14)
684                   //      3210 - Reserved: 0
685
686     // - Remaining Length
687     DISCONNECT[1] = 0;
688
689     // Enviar el paquete DISCONNECT.
690     return MQTT_Write_Packet(DISCONNECT, DISCONNECT_MIN_SIZE);
691 }
692
693
694 static int MQTT_Write_Packet(const char *buf, size_t len)
695 {
696 #ifdef    TEST_MODE
697     // Iniciar el temporizador.
698     TIM_Start();
699 #endif // TEST_MODE
700
701     // Escribir el paquete.
702 #ifdef    USE_TLS
703     int ret = TLS_Write(buf, len);
704 #else  // USE_TLS
705     int ret = TCP_Write(buf, len);
706 #endif // USE_TLS
707
708 #ifdef    TEST_MODE
709     // Detener el temporizador.
```

```
710     TIM_Stop();
711 #endif // TEST_MODE
712 
713     // Indicar si el paquete se ha enviado correctamente.
714     if(ret < 0)
715     {
716         printf(" error !!!\n");
717     }
718     else
719     {
720         printf(" ok\n");
721     }
722 
723     return ret;
724 }
725 
726 
727 static int MQTT_Generate_Packet_ID(void)
728 {
729     // Generar un nuevo identificador del paquete.
730     static int packet_identifier = 0;
731     if(++packet_identifier == 0)
732     {
733         // El 0 está prohibido por el protocolo, por lo que si hay un overflow
734         // del 65535 al 0, hay que saltar al 1.
735         packet_identifier++;
736     }
737 
738     // Devolver el identificador del paquete generado.
739     return packet_identifier;
740 }
```

/NUCLEO_H723ZG_MQTT_TLS/Application/include/sw/mqtt_client.h

```
1  // ----------------------------------------------------------------------------
2  // MQTT_CLIENT
3  //
4  // - Archivo de cabecera.
5  //
6  // ----------------------------------------------------------------------------
7  // Copyright (c) 2021 Jorge Botana Mtz. de Ibarreta
8  //
9  // Este archivo se encuentra bajo los términos de la Licencia MIT. Debería
10 // haberse proporcionado una copia de ella junto a este fichero. Si no es así, se
11 // puede encontrar en el siguiente enlace:
12 //
13 //                                          https://opensource.org/licenses/MIT
```

```
14 // ---------------------------------------------------------------------------
15
16 #ifndef   MQTT_CLIENT_H
17 #define   MQTT_CLIENT_H
18
19 #include <stddef.h>
20
21 int MQTT_Init(const char *hostname, const char *ipaddr, const int port);
22 int MQTT_Close(char *buf, size_t len);
23
24 int MQTT_Read_Packet(char *buf, size_t len);
25
26 int MQTT_Write_CONNECT(const char *client, size_t c_len, const int keep_alive);
27 int MQTT_Write_PUBLISH(const char *topic, size_t t_len, const char *msg, size_t
       m_len);
28 int MQTT_Write_SUBSCRIBE(const char *topic, size_t t_len);
29 int MQTT_Write_UNSUBSCRIBE(const char *topic, size_t t_len);
30 int MQTT_Write_PINGREQ(void);
31 int MQTT_Write_DISCONNECT(void);
32
33 #endif // MQTT_CLIENT_H
```

B.4. Periféricos

B.4.1. IRQ

/NUCLEO_H723ZG_MQTT_TLS/Application/source/hw/irq.c

```
1  // ---------------------------------------------------------------------------
2  // IRQ
3  //
4  // - Interrupciones y excepciones.
5  //
6  // ---------------------------------------------------------------------------
7  // Copyright (c) 2021 Jorge Botana Mtz. de Ibarreta
8  //
9  // Este archivo se encuentra bajo los términos de la Licencia MIT. Debería
10 // haberse proporcionado una copia de ella junto a este fichero. Si no es así, se
11 // puede encontrar en el siguiente enlace:
12 //
13 //                                          https://opensource.org/licenses/MIT
14 // ---------------------------------------------------------------------------
15
16
17 #include "tim.h"
```

```
18 #include "ethernetif.h"
19
20
21 // --------------- Interrupciones y excepciones del ARM Cortex-M7 ---------------
22
23
24 void NMI_Handler(void)
25 {
26     for( ;; );
27 }
28
29
30 void HardFault_Handler(void)
31 {
32     for( ;; );
33 }
34
35
36 void MemManage_Handler(void)
37 {
38     for( ;; );
39 }
40
41
42 void BusFault_Handler(void)
43 {
44     for( ;; );
45 }
46
47
48 void UsageFault_Handler(void)
49 {
50     for( ;; );
51 }
52
53
54 // void SVC_Handler(void)
55 // {
56 //  for( ;; ); // Ya implementado por FreeRTOS.
57 // }
58
59
60 void DebugMon_Handler(void)
61 {
62     for( ;; );
63 }
64
```

```
65
66 // void PendSV_Handler(void)
67 // {
68 //  for( ;; ); // Ya implementado por FreeRTOS.
69 // }
70
71
72 // void SysTick_Handler(void)
73 // {
74 //  for( ;; ); // Ya implementado por CMSIS-RTOS2.
75 // }
76
77
78 // ---------------------- Interrupciones de periféricos -----------------------
79
80
81 void TIM6_DAC_IRQHandler(void)
82 {
83     HAL_TIM_IRQHandler(&htim6);
84 }
85
86
87 void TIM7_IRQHandler(void)
88 {
89     HAL_TIM_IRQHandler(&htim7);
90 }
91
92
93 void ETH_IRQHandler(void)
94 {
95     HAL_ETH_IRQHandler(&heth);
96 }
```

B.4.2. MPU

/NUCLEO_H723ZG_MQTT_TLS/Application/source/hw/mpu.c

```
1 // ------------------------------------------------------------------------------
2 // MPU
3 //
4 // - Inicialización de la unidad de protección de memoria en RAM_D2:
5 //
6 //   -> SRAM1: 0x30000000 - 0x30003FFF : ETH DMA desc. + Rx buffers
7 //   -> SRAM2: 0x30004000 - 0x30007FFF : Tx buffers (lwIP RAM heap)
8 //
9 //   RAM_D2 = SRAM1 + SRAM2 = 16 KB + 16 KB = 32 KB
```

```
10 //
11 //   Esto es necesario en los microcontroladores STM32H723 para que la D-Cache no
12 //   haga uso de esta zona de memoria, ya que es necesario reservarla para los
13 //   descriptores DMA de Ethernet y sus buffers.
14 //
15 // ----------------------------------------------------------------------------
16 // Copyright (c) 2021 Jorge Botana Mtz. de Ibarreta
17 //
18 // Este archivo se encuentra bajo los términos de la Licencia MIT. Debería
19 // haberse proporcionado una copia de ella junto a este fichero. Si no es así, se
20 // puede encontrar en el siguiente enlace:
21 //
22 //                                          https://opensource.org/licenses/MIT
23 // ----------------------------------------------------------------------------
24
25
26 #include "mpu.h"
27
28
29 void MPU_Init(void)
30 {
31 #ifdef    STM32H723xx
32     MPU_Region_InitTypeDef MPU_InitStruct = {0};
33
34     MPU_InitStruct.Enable           = MPU_REGION_ENABLE;
35     MPU_InitStruct.Number           = MPU_REGION_NUMBER0;
36     MPU_InitStruct.BaseAddress      = 0x30000000;
37     MPU_InitStruct.Size             = MPU_REGION_SIZE_32KB;
38     MPU_InitStruct.TypeExtField     = MPU_TEX_LEVEL1;
39     MPU_InitStruct.AccessPermission = MPU_REGION_FULL_ACCESS;
40     HAL_MPU_ConfigRegion(&MPU_InitStruct);
41
42     HAL_MPU_Enable(MPU_PRIVILEGED_DEFAULT);
43 #endif // STM32H723xx
44 }
```

/NUCLEO_H723ZG_MQTT_TLS/Application/source/hw/mpu.h

```
1 // ----------------------------------------------------------------------------
2 // MPU
3 //
4 // - Archivo de cabecera.
5 //
6 // ----------------------------------------------------------------------------
7 // Copyright (c) 2021 Jorge Botana Mtz. de Ibarreta
8 //
9 // Este archivo se encuentra bajo los términos de la Licencia MIT. Debería
```

```
10 // haberse proporcionado una copia de ella junto a este fichero. Si no es así, se
11 // puede encontrar en el siguiente enlace:
12 //
13 //                                              https://opensource.org/licenses/MIT
14 // ----------------------------------------------------------------------------
15
16 #ifndef   MPU_H
17 #define   MPU_H
18
19 #include  HAL_DRIVER
20
21 void MPU_Init(void);
22
23 #endif // MPU_H
```

B.4.3. MISC

/NUCLEO_H723ZG_MQTT_TLS/Application/source/hw/misc.c

```
1  // ----------------------------------------------------------------------------
2  // MISC
3  //
4  // - Inicialización de la caché y del driver HAL.
5  //
6  // ----------------------------------------------------------------------------
7  // Copyright (c) 2021 Jorge Botana Mtz. de Ibarreta
8  //
9  // Este archivo se encuentra bajo los términos de la Licencia MIT. Debería
10 // haberse proporcionado una copia de ella junto a este fichero. Si no es así, se
11 // puede encontrar en el siguiente enlace:
12 //
13 //                                              https://opensource.org/licenses/MIT
14 // ----------------------------------------------------------------------------
15
16
17 #include "misc.h"
18
19
20 void CACHE_Init(void)
21 {
22     // Habilitar la caché mejora el rendimiento del programa, y eso se puede ver
23     // en los tiempos de escritura y lectura de paquetes en el modo de pruebas,
24     // que pasan a ser mucho más pequeños.
25
26     SCB_EnableICache(); // Caché de instrucciones
27     SCB_EnableDCache(); // Caché de datos
```

```
28 }
29
30
31 void SYS_Init(void)
32 {
33     HAL_Init();
34 }
35
36
37 void HAL_MspInit(void)
38 {
39     __HAL_RCC_SYSCFG_CLK_ENABLE();
40 }
```

/NUCLEO_H723ZG_MQTT_TLS/Application/include/hw/misc.h

```
1  // ------------------------------------------------------------------------
2  // MISC
3  //
4  // - Archivo de cabecera.
5  //
6  // ------------------------------------------------------------------------
7  // Copyright (c) 2021 Jorge Botana Mtz. de Ibarreta
8  //
9  // Este archivo se encuentra bajo los términos de la Licencia MIT. Debería
10 // haberse proporcionado una copia de ella junto a este fichero. Si no es así, se
11 // puede encontrar en el siguiente enlace:
12 //
13 //                                          https://opensource.org/licenses/MIT
14 // ------------------------------------------------------------------------
15
16 #ifndef   MISC_H
17 #define   MISC_H
18
19 #include  HAL_DRIVER
20
21 void CACHE_Init(void);
22
23 void SYS_Init(void);
24
25 #endif // MISC_H
```

B.4.4. RCC

/NUCLEO_H723ZG_MQTT_TLS/Application/source/hw/rcc.c

```
1  // ----------------------------------------------------------------------------
2  // RCC
3  //
4  // - Inicialización de los osciladores y relojes del sistema.
5  //
6  // ----------------------------------------------------------------------------
7  // Copyright (c) 2021 Jorge Botana Mtz. de Ibarreta
8  //
9  // Este archivo se encuentra bajo los términos de la Licencia MIT. Debería
10 // haberse proporcionado una copia de ella junto a este fichero. Si no es así, se
11 // puede encontrar en el siguiente enlace:
12 //
13 //                                          https://opensource.org/licenses/MIT
14 // ----------------------------------------------------------------------------
15
16
17 #include "rcc.h"
18
19
20 void RCC_Init(void)
21 {
22     RCC_OscInitTypeDef RCC_OscInitStruct = {0};
23     RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
24
25     // Configurar la fuente de alimentación del sistema.
26     HAL_PWREx_ConfigSupply(PWR_LDO_SUPPLY);
27
28     // Configurar la tensión de salida del regulador interno principal.
29     __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE0);
30     while(!__HAL_PWR_GET_FLAG(PWR_FLAG_VOSRDY)) {}
31
32     // Configurar el LSE.
33     HAL_PWR_EnableBkUpAccess();
34     __HAL_RCC_LSEDRIVE_CONFIG(RCC_LSEDRIVE_LOW);
35
36     // Habilitar los osciladores.
37     RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE | \
38                                        RCC_OSCILLATORTYPE_LSE | \
39                                        RCC_OSCILLATORTYPE_HSI48;
40
41     // System oscillator:
42     // -> HSE   =  8     MHz (bypass clock source)
43     RCC_OscInitStruct.HSEState       = RCC_HSE_BYPASS;
44
45     // RTC oscillator:
46     // -> LSE   = 32.768 kHz (crystal/ceramic resonator)
47     RCC_OscInitStruct.LSEState       = RCC_LSE_ON;
```

```
48
49      // RNG oscillator:
50      // -> HSI48 = 48     MHz
51      RCC_OscInitStruct.HSI48State     = RCC_HSI48_ON;
52
53      // PLL setup.
54      RCC_OscInitStruct.PLL.PLLState   = RCC_PLL_ON;
55
56      // PLL source MUX = HSE
57      RCC_OscInitStruct.PLL.PLLSource  = RCC_PLLSOURCE_HSE;
58
59      // DIVM - Division factor for PLL VCO input clock.
60      RCC_OscInitStruct.PLL.PLLM       =   4;
61
62      // DIVN - Multiplication factor for PLL VCO output clock.
63      RCC_OscInitStruct.PLL.PLLN       = 275;
64
65      // DIVP - Division factor for system clock.
66      RCC_OscInitStruct.PLL.PLLP       =   1;
67
68      // DIVQ - Division factor for peripheral clocks.
69      RCC_OscInitStruct.PLL.PLLQ       =   2;
70
71      // DIVR - Division factor for peripheral clocks.
72      RCC_OscInitStruct.PLL.PLLR       =   2;
73
74      // PLL1 clock input range frequency between 2 and 4 MHz:
75      // -> HSE * DIVM = 8 / 4 = 2 MHz
76      RCC_OscInitStruct.PLL.PLLRGE     = RCC_PLL1VCIRANGE_1;
77
78      // PLL1 clock output range.
79      RCC_OscInitStruct.PLL.PLLVCOSEL  = RCC_PLL1VCOWIDE;
80
81      // Fractional part of the multiplication factor.
82      RCC_OscInitStruct.PLL.PLLFRACN   =   0;
83
84      // Inicializar los osciladores.
85      HAL_RCC_OscConfig(&RCC_OscInitStruct);
86
87      // Habilitar los relojes.
88      RCC_ClkInitStruct.ClockType      = RCC_CLOCKTYPE_SYSCLK  | \
89                                         RCC_CLOCKTYPE_HCLK    | \
90                                         RCC_CLOCKTYPE_D1PCLK1 | \
91                                         RCC_CLOCKTYPE_PCLK1   | \
92                                         RCC_CLOCKTYPE_PCLK2   | \
93                                         RCC_CLOCKTYPE_D3PCLK1;
94
```

```
95      // System clock MUX = PLLCLK
96      // -> SYSCLK = HSE *   DIVM1   * DIVN1 *   DIVP1   =
97      //           =   8 * ( 1 / 4 ) *   275 * ( 1 / 1 ) = 550   MHz
98      RCC_ClkInitStruct.SYSCLKSource   = RCC_SYSCLKSOURCE_PLLCLK;
99
100     // System clock (SYSCLK ) prescaler:
101     // -> D1CPRE         = ( 1 / 1 ) ---> D1CPRE clock = 550   MHz
102     RCC_ClkInitStruct.SYSCLKDivider  = RCC_SYSCLK_DIV1;
103
104     // AHB    clock (HCLK   ) prescaler:
105     // -> HPRE           = ( 1 / 2 ) --->   HCLK clock = 275   MHz
106     RCC_ClkInitStruct.AHBCLKDivider  = RCC_HCLK_DIV2;
107
108     // APB3   clock (D1PCLK1) prescaler:
109     // -> D1PPRE         = ( 1 / 2 ) --->   APB3 clock = 137.5 MHz
110     RCC_ClkInitStruct.APB3CLKDivider = RCC_APB3_DIV2;
111
112     // APB1   clock (PCLK1  ) prescaler:
113     // -> D2PPRE1        = ( 1 / 2 ) --->   APB1 clock = 137.5 MHz
114     RCC_ClkInitStruct.APB1CLKDivider = RCC_APB1_DIV2;
115
116     // APB2   clock (PCLK2  ) prescaler:
117     // -> D2PPRE2        = ( 1 / 2 ) --->   APB2 clock = 137.5 MHz
118     RCC_ClkInitStruct.APB2CLKDivider = RCC_APB2_DIV2;
119
120     // APB4   clock (D3PCLK1) prescaler:
121     // -> D3PPRE         = ( 1 / 2 ) --->   APB4 clock = 137.5 MHz
122     RCC_ClkInitStruct.APB4CLKDivider = RCC_APB4_DIV2;
123
124     // Inicializar los relojes.
125     HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_3);
126 }
```

/NUCLEO_H723ZG_MQTT_TLS/Application/include/hw/rcc.h

```
1  // ----------------------------------------------------------------------------
2  // RCC
3  //
4  // - Archivo de cabecera.
5  //
6  // ----------------------------------------------------------------------------
7  // Copyright (c) 2021 Jorge Botana Mtz. de Ibarreta
8  //
9  // Este archivo se encuentra bajo los términos de la Licencia MIT. Debería
10 // haberse proporcionado una copia de ella junto a este fichero. Si no es así, se
11 // puede encontrar en el siguiente enlace:
12 //
```

```
13 //                                                https://opensource.org/licenses/MIT
14 // ----------------------------------------------------------------------------
15
16 #ifndef   RCC_H
17 #define   RCC_H
18
19 #include  HAL_DRIVER
20
21 void RCC_Init(void);
22
23 #endif // RCC_H
```

B.4.5. GPIO

/NUCLEO_H723ZG_MQTT_TLS/Application/source/hw/gpio.c

```
1  // ----------------------------------------------------------------------------
2  // GPIO
3  //
4  // - Lectura del pulsador y encendido/apagado de los LEDs.
5  //
6  // ----------------------------------------------------------------------------
7  // Copyright (c) 2021 Jorge Botana Mtz. de Ibarreta
8  //
9  // Este archivo se encuentra bajo los términos de la Licencia MIT. Debería
10 // haberse proporcionado una copia de ella junto a este fichero. Si no es así, se
11 // puede encontrar en el siguiente enlace:
12 //
13 //                                            https://opensource.org/licenses/MIT
14 // ----------------------------------------------------------------------------
15
16
17 #include "gpio.h"
18
19
20 #define   B1_CLK_EN()                          __HAL_RCC_GPIOC_CLK_ENABLE()
21 #define   B1_PORT                                                    GPIOC
22 #define   B1_PIN                                                GPIO_PIN_13
23
24 #define   LD1_CLK_EN()                         __HAL_RCC_GPIOB_CLK_ENABLE()
25 #define   LD1_PORT                                                   GPIOB
26 #define   LD1_PIN                                                GPIO_PIN_0
27
28 #define   LD2_CLK_EN()                         __HAL_RCC_GPIOE_CLK_ENABLE()
29 #define   LD2_PORT                                                   GPIOE
30 #define   LD2_PIN                                                GPIO_PIN_1
```

```
31
32 #define   LD3_CLK_EN()                                __HAL_RCC_GPIOB_CLK_ENABLE()
33 #define   LD3_PORT                                                          GPIOB
34 #define   LD3_PIN                                                     GPIO_PIN_14
35
36
37 void GPIO_Init(void)
38 {
39     GPIO_InitTypeDef GPIO_InitStruct = {0};
40
41     B1_CLK_EN();
42     GPIO_InitStruct.Pin   = B1_PIN;
43     GPIO_InitStruct.Mode  = GPIO_MODE_INPUT;
44     GPIO_InitStruct.Pull  = GPIO_NOPULL;
45     HAL_GPIO_Init(B1_PORT, &GPIO_InitStruct);
46
47     LD1_CLK_EN();
48     GPIO_InitStruct.Pin   = LD1_PIN;
49     GPIO_InitStruct.Mode  = GPIO_MODE_OUTPUT_PP;
50     GPIO_InitStruct.Pull  = GPIO_NOPULL;
51     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
52     HAL_GPIO_Init(LD1_PORT, &GPIO_InitStruct);
53
54     LD2_CLK_EN();
55     GPIO_InitStruct.Pin   = LD2_PIN;
56     GPIO_InitStruct.Mode  = GPIO_MODE_OUTPUT_PP;
57     GPIO_InitStruct.Pull  = GPIO_NOPULL;
58     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
59     HAL_GPIO_Init(LD2_PORT, &GPIO_InitStruct);
60
61     LD3_CLK_EN();
62     GPIO_InitStruct.Pin   = LD3_PIN;
63     GPIO_InitStruct.Mode  = GPIO_MODE_OUTPUT_PP;
64     GPIO_InitStruct.Pull  = GPIO_NOPULL;
65     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
66     HAL_GPIO_Init(LD3_PORT, &GPIO_InitStruct);
67 }
68
69
70 int B1_Read(void)
71 {
72     return HAL_GPIO_ReadPin(B1_PORT, B1_PIN);
73 }
74
75
76 void LD1_On(void)
77 {
```

```
78      HAL_GPIO_WritePin(LD1_PORT, LD1_PIN, GPIO_PIN_SET);
79  }
80
81
82  void LD1_Off(void)
83  {
84      HAL_GPIO_WritePin(LD1_PORT, LD1_PIN, GPIO_PIN_RESET);
85  }
86
87
88  void LD2_On(void)
89  {
90      HAL_GPIO_WritePin(LD2_PORT, LD2_PIN, GPIO_PIN_SET);
91  }
92
93
94  void LD2_Off(void)
95  {
96      HAL_GPIO_WritePin(LD2_PORT, LD2_PIN, GPIO_PIN_RESET);
97  }
98
99
100 void LD3_On(void)
101 {
102     HAL_GPIO_WritePin(LD3_PORT, LD3_PIN, GPIO_PIN_SET);
103 }
104
105
106 void LD3_Off(void)
107 {
108     HAL_GPIO_WritePin(LD3_PORT, LD3_PIN, GPIO_PIN_RESET);
109 }
```

/NUCLEO_H723ZG_MQTT_TLS/Application/include/hw/gpio.h

```
1  // ------------------------------------------------------------------------
2  // GPIO
3  //
4  // - Archivo de cabecera.
5  //
6  // ------------------------------------------------------------------------
7  // Copyright (c) 2021 Jorge Botana Mtz. de Ibarreta
8  //
9  // Este archivo se encuentra bajo los términos de la Licencia MIT. Debería
10 // haberse proporcionado una copia de ella junto a este fichero. Si no es así, se
11 // puede encontrar en el siguiente enlace:
12 //
```

```
13 //                                                https://opensource.org/licenses/MIT
14 // ----------------------------------------------------------------------------
15
16 #ifndef   GPIO_H
17 #define   GPIO_H
18
19 #include  HAL_DRIVER
20
21 void GPIO_Init(void);
22
23 int B1_Read(void);
24
25 void LD1_On(void);
26 void LD1_Off(void);
27
28 void LD2_On(void);
29 void LD2_Off(void);
30
31 void LD3_On(void);
32 void LD3_Off(void);
33
34 #endif // GPIO_H
```

B.4.6. RNG

/NUCLEO_H723ZG_MQTT_TLS/Application/source/hw/rng.c

```
1  // ----------------------------------------------------------------------------
2  // RNG
3  //
4  // - Generación de números aleatorios.
5  //
6  // ----------------------------------------------------------------------------
7  // Copyright (c) 2021 Jorge Botana Mtz. de Ibarreta
8  //
9  // Este archivo se encuentra bajo los términos de la Licencia MIT. Debería
10 // haberse proporcionado una copia de ella junto a este fichero. Si no es así, se
11 // puede encontrar en el siguiente enlace:
12 //
13 //                                                https://opensource.org/licenses/MIT
14 // ----------------------------------------------------------------------------
15
16
17 #include "rng.h"
18
19
```

```
20 RNG_HandleTypeDef hrng = {0};
21
22
23 void RNG_Init(void)
24 {
25     hrng.Instance = RNG;
26     HAL_RNG_Init(&hrng);
27 }
28
29
30 unsigned int RNG_Get(void)
31 {
32     unsigned int random = 0;
33     HAL_RNG_GenerateRandomNumber(&hrng, (uint32_t *)&random);
34
35     return random;
36 }
37
38
39 void HAL_RNG_MspInit(RNG_HandleTypeDef *hrng)
40 {
41     RCC_PeriphCLKInitTypeDef PeriphClkInitStruct = {0};
42
43     if(hrng->Instance == RNG)
44     {
45         PeriphClkInitStruct.PeriphClockSelection = RCC_PERIPHCLK_RNG;
46         PeriphClkInitStruct.RngClockSelection    = RCC_RNGCLKSOURCE_HSI48;
47         HAL_RCCEx_PeriphCLKConfig(&PeriphClkInitStruct);
48         __HAL_RCC_RNG_CLK_ENABLE();
49     }
50 }
```

/NUCLEO_H723ZG_MQTT_TLS/Application/include/hw/rng.h

```
1  // ------------------------------------------------------------------------
2  // RNG
3  //
4  // - Archivo de cabecera.
5  //
6  // ------------------------------------------------------------------------
7  // Copyright (c) 2021 Jorge Botana Mtz. de Ibarreta
8  //
9  // Este archivo se encuentra bajo los términos de la Licencia MIT. Debería
10 // haberse proporcionado una copia de ella junto a este fichero. Si no es así, se
11 // puede encontrar en el siguiente enlace:
12 //
13 //                                          https://opensource.org/licenses/MIT
```

```
14 // ---------------------------------------------------------------------------
15
16 #ifndef   RNG_H
17 #define   RNG_H
18
19 #include  HAL_DRIVER
20
21 void RNG_Init(void);
22
23 unsigned int RNG_Get(void);
24
25 #endif // RNG_H
```

B.4.7. RTC

/NUCLEO_H723ZG_MQTT_TLS/Application/source/hw/rtc.c

```
1  // ---------------------------------------------------------------------------
2  // RTC
3  //
4  // - Lectura y escritura del reloj de tiempo real.
5  //
6  // ---------------------------------------------------------------------------
7  // Copyright (c) 2021 Jorge Botana Mtz. de Ibarreta
8  //
9  // Este archivo se encuentra bajo los términos de la Licencia MIT. Debería
10 // haberse proporcionado una copia de ella junto a este fichero. Si no es así, se
11 // puede encontrar en el siguiente enlace:
12 //
13 //                                          https://opensource.org/licenses/MIT
14 // ---------------------------------------------------------------------------
15
16
17 #include "rtc.h"
18
19
20 RTC_HandleTypeDef hrtc = {0};
21
22
23 void RTC_Init(void)
24 {
25     hrtc.Instance          = RTC;
26     hrtc.Init.AsynchPrediv = 127;
27     hrtc.Init.SynchPrediv  = 255;
28     HAL_RTC_Init(&hrtc);
29 }
```

```
30
31
32 time_t RTC_Read(void)
33 {
34     RTC_TimeTypeDef sTime = {0};
35     RTC_DateTypeDef sDate = {0};
36     struct tm s_time_date = {0};
37
38     // Advertencia: las lecturas deben hacerse en el siguiente orden: primero
39     // HAL_RTC_GetTime() y luego HAL_RTC_GetDate(). De lo contrario, no se va a
40     // leer correctamente el RTC.
41     HAL_RTC_GetTime(&hrtc, &sTime, RTC_FORMAT_BIN);
42     HAL_RTC_GetDate(&hrtc, &sDate, RTC_FORMAT_BIN);
43
44     s_time_date.tm_hour = sTime.Hours;
45     s_time_date.tm_min  = sTime.Minutes;
46     s_time_date.tm_sec  = sTime.Seconds;
47
48     s_time_date.tm_wday = sDate.WeekDay;
49     s_time_date.tm_mon  = sDate.Month;
50     s_time_date.tm_mday = sDate.Date;
51     s_time_date.tm_year = sDate.Year;
52
53     return mktime(&s_time_date);
54 }
55
56
57 void RTC_Write(time_t curtime)
58 {
59     RTC_TimeTypeDef sTime = {0};
60     RTC_DateTypeDef sDate = {0};
61     struct tm s_time_date = {0};
62
63     localtime_r(&curtime, &s_time_date);
64
65     sTime.Hours   = s_time_date.tm_hour;
66     sTime.Minutes = s_time_date.tm_min;
67     sTime.Seconds = s_time_date.tm_sec;
68
69     sDate.WeekDay = s_time_date.tm_wday;
70     sDate.Month   = s_time_date.tm_mon;
71     sDate.Date    = s_time_date.tm_mday;
72     sDate.Year    = s_time_date.tm_year;
73
74     HAL_RTC_SetTime(&hrtc, &sTime, RTC_FORMAT_BIN);
75     HAL_RTC_SetDate(&hrtc, &sDate, RTC_FORMAT_BIN);
76 }
```

```
77
78
79 void HAL_RTC_MspInit(RTC_HandleTypeDef *hrtc)
80 {
81     RCC_PeriphCLKInitTypeDef PeriphClkInitStruct = {0};
82
83     if(hrtc->Instance == RTC)
84     {
85         PeriphClkInitStruct.PeriphClockSelection = RCC_PERIPHCLK_RTC;
86         PeriphClkInitStruct.RTCClockSelection    = RCC_RTCCLKSOURCE_LSE;
87         HAL_RCCEx_PeriphCLKConfig(&PeriphClkInitStruct);
88         __HAL_RCC_RTC_ENABLE();
89     }
90 }
```

/NUCLEO_H723ZG_MQTT_TLS/Application/include/hw/rtc.h

```
1  // ----------------------------------------------------------------------------
2  // RTC
3  //
4  // - Archivo de cabecera.
5  //
6  // ----------------------------------------------------------------------------
7  // Copyright (c) 2021 Jorge Botana Mtz. de Ibarreta
8  //
9  // Este archivo se encuentra bajo los términos de la Licencia MIT. Debería
10 // haberse proporcionado una copia de ella junto a este fichero. Si no es así, se
11 // puede encontrar en el siguiente enlace:
12 //
13 //                                          https://opensource.org/licenses/MIT
14 // ----------------------------------------------------------------------------
15
16 #ifndef   RTC_H
17 #define   RTC_H
18
19 #include  HAL_DRIVER
20
21 #include <time.h>
22
23 void RTC_Init(void);
24
25 time_t RTC_Read(void);
26 void RTC_Write(time_t curtime);
27
28 #endif // RTC_H
```

B.4.8. UART

/NUCLEO_H723ZG_MQTT_TLS/Application/source/hw/uart.c

```
1  // ----------------------------------------------------------------------------
2  // UART
3  //
4  // - Retarget de printf() a puerto serie.
5  //
6  // ----------------------------------------------------------------------------
7  // Copyright (c) 2021 Jorge Botana Mtz. de Ibarreta
8  //
9  // Este archivo se encuentra bajo los términos de la Licencia MIT. Debería
10 // haberse proporcionado una copia de ella junto a este fichero. Si no es así, se
11 // puede encontrar en el siguiente enlace:
12 //
13 //                                          https://opensource.org/licenses/MIT
14 // ----------------------------------------------------------------------------
15
16
17 #include "uart.h"
18
19
20 #define   UART_INSTANCE                                                USART3
21 #define   UART_PORT                                                     GPIOD
22 #define   TX_PIN                                                    GPIO_PIN_8
23 #define   RX_PIN                                                    GPIO_PIN_9
24
25 #define   UART_CLK_EN()                              __HAL_RCC_USART3_CLK_ENABLE()
26 #define   GPIO_CLK_EN()                               __HAL_RCC_GPIOD_CLK_ENABLE()
27 #define   PERIPHERAL_CLK                                      RCC_PERIPHCLK_USART3
28 #define   UART_CLK                                 RCC_USART234578CLKSOURCE_D2PCLK1
29 #define   UART_ALTERNATE                                        GPIO_AF7_USART3
30
31
32 UART_HandleTypeDef huart = {0};
33
34
35 void __io_putchar(int ch)
36 {
37     HAL_UART_Transmit(&huart, (uint8_t *)&ch, 1, 0xFFFF);
38 }
39
40
41 void UART_Init(void)
42 {
43     huart.Instance        = UART_INSTANCE;
```

```
44      huart.Init.BaudRate    = 115200;
45      huart.Init.WordLength = UART_WORDLENGTH_8B;
46      huart.Init.StopBits    = UART_STOPBITS_1;
47      huart.Init.Parity      = UART_PARITY_NONE;
48      huart.Init.Mode        = UART_MODE_TX_RX;
49      huart.Init.HwFlowCtl   = UART_HWCONTROL_NONE;
50  }
51
52
53  void UART_Start(void)
54  {
55      HAL_UART_Init(&huart);
56  }
57
58
59  void UART_Stop(void)
60  {
61      HAL_UART_DeInit(&huart);
62  }
63
64
65  void HAL_UART_MspInit(UART_HandleTypeDef *huart)
66  {
67      RCC_PeriphCLKInitTypeDef PeriphClkInitStruct = {0};
68      GPIO_InitTypeDef         GPIO_InitStruct     = {0};
69
70      if(huart->Instance == UART_INSTANCE)
71      {
72          PeriphClkInitStruct.PeriphClockSelection      = PERIPHERAL_CLK;
73          PeriphClkInitStruct.Usart234578ClockSelection = UART_CLK;
74          HAL_RCCEx_PeriphCLKConfig(&PeriphClkInitStruct);
75          UART_CLK_EN();
76
77          GPIO_CLK_EN();
78          GPIO_InitStruct.Pin       = TX_PIN | RX_PIN;
79          GPIO_InitStruct.Mode      = GPIO_MODE_AF_PP;
80          GPIO_InitStruct.Pull      = GPIO_NOPULL;
81          GPIO_InitStruct.Speed     = GPIO_SPEED_FREQ_VERY_HIGH;
82          GPIO_InitStruct.Alternate = UART_ALTERNATE;
83          HAL_GPIO_Init(UART_PORT, &GPIO_InitStruct);
84      }
85  }
```

/NUCLEO_H723ZG_MQTT_TLS/Application/include/hw/uart.h

```
1  // -----------------------------------------------------------------------------
2  // UART
```

```
3  //
4  // - Archivo de cabecera.
5  //
6  // ---------------------------------------------------------------------------
7  // Copyright (c) 2021 Jorge Botana Mtz. de Ibarreta
8  //
9  // Este archivo se encuentra bajo los términos de la Licencia MIT. Debería
10 // haberse proporcionado una copia de ella junto a este fichero. Si no es así, se
11 // puede encontrar en el siguiente enlace:
12 //
13 //                                          https://opensource.org/licenses/MIT
14 // ---------------------------------------------------------------------------
15
16 #ifndef   UART_H
17 #define   UART_H
18
19 #include  HAL_DRIVER
20
21 void UART_Init(void);
22
23 void UART_Start(void);
24 void UART_Stop(void);
25
26 #endif // UART_H
```

B.4.9. TIM

/NUCLEO_H723ZG_MQTT_TLS/Application/source/hw/tim.c

```
1  // ---------------------------------------------------------------------------
2  // TIM
3  //
4  // - Temporizador (TIM6) usado por el driver HAL y que reemplaza al SysTick, ya
5  //   que este último lo utiliza el RTOS (el driver HAL y el RTOS no deberían
6  //   compartir fuente de reloj). Este temporizador también provee delays
7  //   bloqueantes. Un segundo temporizador (TIM7) es usado para medir tiempos en
8  //   microsegundos.
9  //
10 // ---------------------------------------------------------------------------
11 // Copyright (c) 2021 Jorge Botana Mtz. de Ibarreta
12 //
13 // Este archivo se encuentra bajo los términos de la Licencia MIT. Debería
14 // haberse proporcionado una copia de ella junto a este fichero. Si no es así, se
15 // puede encontrar en el siguiente enlace:
16 //
17 //                                          https://opensource.org/licenses/MIT
```

```
18 // ----------------------------------------------------------------------------
19
20
21 #include "tim.h"
22
23
24 #define   TIM7_FREQUENCY_MHZ                                              275
25 #define   TIM7_PERIOD_US                                                    5
26 #define   TIM7_PSC                                                          0
27 #define   TIM7_ARR                      (TIM7_FREQUENCY_MHZ - 1) * TIM7_PERIOD_US
28
29
30 TIM_HandleTypeDef htim6 = {0};
31 TIM_HandleTypeDef htim7 = {0};
32
33 volatile int us_counter = 0;
34
35
36 HAL_StatusTypeDef HAL_InitTick(uint32_t TickPriority)
37 {
38     RCC_ClkInitTypeDef clkconfig       = {0};
39     uint32_t           pFLatency       = 0;
40     uint32_t           uwAPB1Prescaler = 0;
41     uint32_t           uwTimclock      = 0;
42
43     HAL_NVIC_SetPriority(TIM6_DAC_IRQn, TickPriority, 0);
44     HAL_NVIC_EnableIRQ  (TIM6_DAC_IRQn);
45     uwTickPrio = TickPriority;
46
47     __HAL_RCC_TIM6_CLK_ENABLE();
48
49     HAL_RCC_GetClockConfig(&clkconfig, &pFLatency);
50
51     uwAPB1Prescaler = clkconfig.APB1CLKDivider;
52
53     if(uwAPB1Prescaler == RCC_HCLK_DIV1)
54     {
55         uwTimclock  =     HAL_RCC_GetPCLK1Freq();
56     }
57     else
58     {
59         uwTimclock  = 2 * HAL_RCC_GetPCLK1Freq();
60     }
61
62     htim6.Instance         = TIM6;
63     htim6.Init.Prescaler   = (uwTimclock / 1000000) - 1;
64     htim6.Init.CounterMode = TIM_COUNTERMODE_UP;
```

```
65      htim6.Init.Period      = (   1000000 /    1000) - 1;
66      HAL_TIM_Base_Init(&htim6);
67
68      return HAL_TIM_Base_Start_IT(&htim6);
69  }
70
71
72  void Tick_Delay(unsigned int delay_ms)
73  {
74      HAL_Delay(delay_ms);
75  }
76
77
78  void TIM_Init(void)
79  {
80      htim7.Instance               = TIM7;
81      htim7.Init.Prescaler         = TIM7_PSC;
82      htim7.Init.CounterMode       = TIM_COUNTERMODE_UP;
83      htim7.Init.Period            = TIM7_ARR;
84      htim7.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_ENABLE;
85      HAL_TIM_Base_Init(&htim7);
86  }
87
88
89  void TIM_Start(void)
90  {
91      us_counter = 0;
92      HAL_TIM_Base_Start_IT(&htim7);
93  }
94
95
96  void TIM_Stop(void)
97  {
98      HAL_TIM_Base_Stop_IT(&htim7);
99  }
100
101
102 int TIM_Read(void)
103 {
104     return us_counter;
105 }
106
107
108 void HAL_TIM_Base_MspInit(TIM_HandleTypeDef *htim)
109 {
110     if(htim->Instance == TIM7)
111     {
```

```
112         __HAL_RCC_TIM7_CLK_ENABLE();
113
114         HAL_NVIC_SetPriority(TIM7_IRQn, 0, 0);
115         HAL_NVIC_EnableIRQ  (TIM7_IRQn);
116     }
117 }
118
119
120 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
121 {
122     if(htim->Instance == TIM6)
123     {
124         HAL_IncTick();
125     }
126
127     if(htim->Instance == TIM7)
128     {
129         us_counter = us_counter + TIM7_PERIOD_US;
130     }
131 }
132
133
134 // ------------------------------------------------------------------------------
135 // Fórmula empleada para calcular la frecuencia de un temporizador:
136 //
137 // -> f (Hz) = Timer clock    Hz / [(TIM_PSC + 1) * (TIM_ARR + 1)]
138 //
139 // Ejemplo para un temporizador de 1 Hz (periodo de 1 segundo):
140 //
141 // -> 1 (Hz) = 275 * (10 ^ 6) Hz / [(  9 999 + 1) * ( 27 499 + 1)]
142 //
143 // En este caso el temporizador interrumpiría cada segundo.
144 // ------------------------------------------------------------------------------
```

/NUCLEO_H723ZG_MQTT_TLS/Application/include/hw/tim.h

```
1  // ------------------------------------------------------------------------------
2  // TIM
3  //
4  // - Archivo de cabecera.
5  //
6  // ------------------------------------------------------------------------------
7  // Copyright (c) 2021 Jorge Botana Mtz. de Ibarreta
8  //
9  // Este archivo se encuentra bajo los términos de la Licencia MIT. Debería
10 // haberse proporcionado una copia de ella junto a este fichero. Si no es así, se
11 // puede encontrar en el siguiente enlace:
```

```
12 //
13 //                                                https://opensource.org/licenses/MIT
14 // ----------------------------------------------------------------------------
15
16 #ifndef   TIM_H
17 #define   TIM_H
18
19 #include  HAL_DRIVER
20
21 extern TIM_HandleTypeDef htim6;
22 extern TIM_HandleTypeDef htim7;
23
24 void Tick_Delay(unsigned int delay_ms);
25
26 void TIM_Init(void);
27
28 void TIM_Start(void);
29 void TIM_Stop(void);
30 int TIM_Read(void);
31
32 #endif // TIM_H
```

B.4.10. ETHERNETIF

El siguiente archivo fue creado por STMicroelectronics y cuenta con modificaciones del autor de este documento.

/NUCLEO_H723ZG_MQTT_TLS/Application/source/hw/ethernetif.c

```
1  // ----------------------------------------------------------------------------
2  // ETHERNETIF
3  //
4  // - Interfaz de red para lwIP en modo multitarea con CMSIS-RTOS2, haciendo uso
5  //   del driver de Ethernet proporcionado por STMicroelectronics para un LAN8742.
6  //
7  // ----------------------------------------------------------------------------
8  // Copyright (c) 2017 STMicroelectronics.
9  // All rights reserved.
10 //
11 // This software component is licensed by ST under BSD 3-Clause license, the
12 // "License"; You may not use this file except in compliance with the License.
13 // You may obtain a copy of the License at:
14 //
15 //                                           opensource.org/licenses/BSD-3-Clause
16 // ----------------------------------------------------------------------------
17
```

```
18
19 #include "ethernetif.h"
20
21 #include "cmsis_os.h"
22
23 #include "lwip/etharp.h"
24
25 #include "lan8742.h"
26
27
28 #define   IFNAME0                                                    's'
29 #define   IFNAME1                                                    't'
30
31 #define   INTERFACE_THREAD_STACK_SIZE                  configMINIMAL_STACK_SIZE
32 #define   TIME_WAITING_FOR_INPUT                                osWaitForever
33
34 #define   ETH_RX_BUFFER_SIZE                                             1524
35 #define   ETH_DMA_TRANSMIT_TIMEOUT                                         20
36
37
38 LWIP_MEMPOOL_DECLARE(RX_POOL, 10, sizeof(struct pbuf_custom), "Zero-copy RX PBUF
       pool");
39
40
41 ETH_DMADescTypeDef DMARxDscrTab[ETH_RX_DESC_CNT]     __attribute__((section(".
       RxDecripSection")));
42 ETH_DMADescTypeDef DMATxDscrTab[ETH_TX_DESC_CNT]     __attribute__((section(".
       TxDecripSection")));
43 uint8_t Rx_Buff[ETH_RX_DESC_CNT][ETH_RX_BUFFER_SIZE] __attribute__((section(".
       RxArraySection" )));
44
45 ETH_HandleTypeDef  heth     = {0};
46 ETH_TxPacketConfig TxConfig = {0};
47
48 osSemaphoreId RxPktSemaphore = NULL;
49
50
51 static void low_level_init(struct netif *netif);
52 static void ethernetif_input(void *arg);
53 static struct pbuf *low_level_input(struct netif *netif);
54 static err_t low_level_output(struct netif *netif, struct pbuf *p);
55
56 static void pbuf_free_custom(struct pbuf *p);
57
58 static int32_t ETH_PHY_IO_Init(void);
59 static int32_t ETH_PHY_IO_DeInit(void);
60 static int32_t ETH_PHY_IO_ReadReg(uint32_t DevAddr, uint32_t RegAddr, uint32_t *
```

```
    pRegVal);
61 static int32_t ETH_PHY_IO_WriteReg(uint32_t DevAddr, uint32_t RegAddr, uint32_t
    RegVal);
62 static int32_t ETH_PHY_IO_GetTick(void);
63
64
65 err_t ethernetif_init(struct netif *netif)
66 {
67     // Set the interface name.
68     netif->name[0]    = IFNAME0;
69     netif->name[1]    = IFNAME1;
70
71     // Set the interface output functions.
72     netif->output     = etharp_output;
73     netif->linkoutput = low_level_output;
74
75     // Initialize the hardware.
76     low_level_init(netif);
77
78     return ERR_OK;
79 }
80
81
82 static void low_level_init(struct netif *netif)
83 {
84     // Initialize Ethernet peripheral.
85     heth.Instance            = ETH;
86     heth.Init.MACAddr[0]     = ETH_MAC_ADDR0;
87     heth.Init.MACAddr[1]     = ETH_MAC_ADDR1;
88     heth.Init.MACAddr[2]     = ETH_MAC_ADDR2;
89     heth.Init.MACAddr[3]     = ETH_MAC_ADDR3;
90     heth.Init.MACAddr[4]     = ETH_MAC_ADDR4;
91     heth.Init.MACAddr[5]     = ETH_MAC_ADDR5;
92     heth.Init.MediaInterface = HAL_ETH_RMII_MODE;
93     heth.Init.TxDesc         = DMATxDscrTab;
94     heth.Init.RxDesc         = DMARxDscrTab;
95     heth.Init.RxBuffLen      = ETH_RX_BUFFER_SIZE;
96     HAL_ETH_Init(&heth);
97
98     // Set Tx packet config common parameters.
99     TxConfig.Attributes   = ETH_TX_PACKETS_FEATURES_CSUM | \
100                             ETH_TX_PACKETS_FEATURES_CRCPAD;
101    TxConfig.ChecksumCtrl = ETH_CHECKSUM_IPHDR_PAYLOAD_INSERT_PHDR_CALC;
102
103    // Initialize the RX POOL.
104    LWIP_MEMPOOL_INIT(RX_POOL);
105
```

```
106     // Set the Maximum Transmission Unit (MTU).
107     netif->mtu        = ETH_MAX_PAYLOAD;
108
109     // Set MAC hardware address.
110     netif->hwaddr[0]  = ETH_MAC_ADDR0;
111     netif->hwaddr[1]  = ETH_MAC_ADDR1;
112     netif->hwaddr[2]  = ETH_MAC_ADDR2;
113     netif->hwaddr[3]  = ETH_MAC_ADDR3;
114     netif->hwaddr[4]  = ETH_MAC_ADDR4;
115     netif->hwaddr[5]  = ETH_MAC_ADDR5;
116     netif->hwaddr_len = ETH_HWADDR_LEN;
117
118     // Set device capabilities.
119     netif->flags     |= NETIF_FLAG_BROADCAST | \
120                         NETIF_FLAG_ETHARP;
121
122     // Assign memory buffers to DMA Rx descriptors.
123     int i = 0;
124     for(i = 0; i < ETH_RX_DESC_CNT; i++)
125     {
126         HAL_ETH_DescAssignMemory(&heth, i, Rx_Buff[i], NULL);
127     }
128
129     // Create a binary semaphore used for informing ethernetif of frame
130     // reception.
131     RxPktSemaphore = osSemaphoreNew(1, 1, NULL);
132
133     // Create the task that handles the ETH_MAC.
134     const osThreadAttr_t attributes =
135     {
136         .stack_size = INTERFACE_THREAD_STACK_SIZE,
137         .priority   = osPriorityRealtime
138     };
139     osThreadNew(ethernetif_input, netif, &attributes);
140
141     // Set PHY IO functions.
142     lan8742_Object_t LAN8742       = {0};
143     lan8742_IOCtx_t  LAN8742_IOCtx =
144     {
145         ETH_PHY_IO_Init,
146         ETH_PHY_IO_DeInit,
147         ETH_PHY_IO_WriteReg,
148         ETH_PHY_IO_ReadReg,
149         ETH_PHY_IO_GetTick
150     };
151     LAN8742_RegisterBusIO(&LAN8742, &LAN8742_IOCtx);
152
```

```
153     // Initialize the LAN8742 ETH PHY.
154     LAN8742_Init(&LAN8742);
155 
156     // Get link state.
157     int32_t  PHYLinkState = LAN8742_GetLinkState(&LAN8742);
158     uint32_t speed        = 0;
159     uint32_t duplex       = 0;
160     switch(PHYLinkState)
161     {
162         case LAN8742_STATUS_100MBITS_FULLDUPLEX:
163             speed  = ETH_SPEED_100M;
164             duplex = ETH_FULLDUPLEX_MODE;
165             break;
166         case LAN8742_STATUS_100MBITS_HALFDUPLEX:
167             speed  = ETH_SPEED_100M;
168             duplex = ETH_HALFDUPLEX_MODE;
169             break;
170         case LAN8742_STATUS_10MBITS_FULLDUPLEX:
171             speed  = ETH_SPEED_10M;
172             duplex = ETH_FULLDUPLEX_MODE;
173             break;
174         case LAN8742_STATUS_10MBITS_HALFDUPLEX:
175             speed  = ETH_SPEED_10M;
176             duplex = ETH_HALFDUPLEX_MODE;
177             break;
178         default:
179             speed  = ETH_SPEED_100M;
180             duplex = ETH_FULLDUPLEX_MODE;
181             break;
182     }
183 
184     // Set MAC config.
185     ETH_MACConfigTypeDef MACConf = {0};
186     HAL_ETH_GetMACConfig(&heth, &MACConf);
187     MACConf.Speed      = speed;
188     MACConf.DuplexMode = duplex;
189     HAL_ETH_SetMACConfig(&heth, &MACConf);
190 
191     // Start Ethernet in interrupt mode.
192     HAL_ETH_Start_IT(&heth);
193 }
194 
195 
196 static void ethernetif_input(void *arg)
197 {
198     struct pbuf  *p     = NULL;
199     struct netif *netif = arg;
```

```
200
201     for( ;; )
202     {
203         if(osSemaphoreAcquire(RxPktSemaphore, TIME_WAITING_FOR_INPUT) == osOK)
204         {
205             do
206             {
207                 p = low_level_input(netif);
208                 if(p != NULL)
209                 {
210                     if(netif->input(p, netif) != ERR_OK)
211                     {
212                         pbuf_free(p);
213                     }
214                 }
215             }
216             while(p != NULL);
217         }
218     }
219 }
220
221
222 static struct pbuf *low_level_input(struct netif *netif)
223 {
224     int i = 0;
225     struct pbuf *p = NULL;
226     ETH_BufferTypeDef RxBuff[ETH_RX_DESC_CNT] = {0};
227
228     for(i = 0; i < ETH_RX_DESC_CNT - 1; i++)
229     {
230         RxBuff[i].next = &RxBuff[i + 1];
231     }
232
233     if(HAL_ETH_GetRxDataBuffer(&heth, RxBuff) == HAL_OK)
234     {
235         uint32_t framelength = 0;
236
237         HAL_ETH_GetRxDataLength(&heth, &framelength);
238
239         // Build Rx descriptor to be ready for next data reception.
240         HAL_ETH_BuildRxDescriptors(&heth);
241
242         // Invalidate data cache for ETH Rx buffers.
243         SCB_InvalidateDCache_by_Addr(RxBuff->buffer, framelength);
244
245         struct pbuf_custom *custom_pbuf = LWIP_MEMPOOL_ALLOC(RX_POOL);
246         if(custom_pbuf != NULL)
```

```
247         {
248             custom_pbuf->custom_free_function = pbuf_free_custom;
249             p = pbuf_alloced_custom(PBUF_RAW, framelength, PBUF_REF, custom_pbuf,
     RxBuff->buffer, framelength);
250         }
251     }
252 
253     return p;
254 }
255 
256 
257 static err_t low_level_output(struct netif *netif, struct pbuf *p)
258 {
259     int i = 0;
260     struct pbuf *q = NULL;
261     ETH_BufferTypeDef Txbuffer[ETH_TX_DESC_CNT] = {0};
262 
263     for(q = p; q != NULL; q = q->next)
264     {
265         if(i >= ETH_TX_DESC_CNT)
266         {
267             return ERR_IF;
268         }
269 
270         Txbuffer[i].buffer = q->payload;
271         Txbuffer[i].len    = q->len;
272 
273         if(i > 0)
274         {
275             Txbuffer[i - 1].next = &Txbuffer[i];
276         }
277 
278         if(q->next == NULL)
279         {
280             Txbuffer[i    ].next = NULL;
281         }
282 
283         i++;
284     }
285 
286     TxConfig.Length   = p->tot_len;
287     TxConfig.TxBuffer = Txbuffer;
288 
289     HAL_ETH_Transmit(&heth, &TxConfig, ETH_DMA_TRANSMIT_TIMEOUT);
290 
291     return ERR_OK;
292 }
```

```
293
294
295 u32_t sys_now(void)
296 {
297     return HAL_GetTick();
298 }
299
300
301 int ETH_Link_Status(void)
302 {
303     uint32_t regvalue = 0;
304     HAL_ETH_ReadPHYRegister(&heth, 0, LAN8742_BSR, &regvalue);
305     if((regvalue & LAN8742_BSR_LINK_STATUS) == RESET)
306     {
307         return -1;
308     }
309
310     return 0;
311 }
312
313
314 void HAL_ETH_MspInit(ETH_HandleTypeDef *heth)
315 {
316     GPIO_InitTypeDef GPIO_InitStruct = {0};
317
318     if(heth->Instance == ETH)
319     {
320         // Enable the Ethernet clocks.
321         __HAL_RCC_ETH1MAC_CLK_ENABLE();
322         __HAL_RCC_ETH1TX_CLK_ENABLE ();
323         __HAL_RCC_ETH1RX_CLK_ENABLE ();
324
325         // Enable the GPIOs clocks.
326         __HAL_RCC_GPIOA_CLK_ENABLE  ();
327         __HAL_RCC_GPIOB_CLK_ENABLE  ();
328         __HAL_RCC_GPIOC_CLK_ENABLE  ();
329         __HAL_RCC_GPIOG_CLK_ENABLE  ();
330
331         // Ethernet pins configuration (RMII mode).
332         GPIO_InitStruct.Mode      = GPIO_MODE_AF_PP;
333         GPIO_InitStruct.Pull      = GPIO_NOPULL;
334         GPIO_InitStruct.Speed     = GPIO_SPEED_FREQ_VERY_HIGH;
335         GPIO_InitStruct.Alternate = GPIO_AF11_ETH;
336
337         // - REF_CLK -> PA1
338         // - MDIO    -> PA2
339         // - CRS_DV  -> PA7
```

```
340         GPIO_InitStruct.Pin       = GPIO_PIN_1  | \
341                                     GPIO_PIN_2  | \
342                                     GPIO_PIN_7 ;
343         HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
344
345         // - TXD1    -> PB13
346         GPIO_InitStruct.Pin       = GPIO_PIN_13;
347         HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
348
349         // - MDC     -> PC1
350         // - RXD0    -> PC4
351         // - RXD1    -> PC5
352         GPIO_InitStruct.Pin       = GPIO_PIN_1  | \
353                                     GPIO_PIN_4  | \
354                                     GPIO_PIN_5 ;
355         HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
356
357         // - RX_ER   -> PG2
358         // - TX_EN   -> PG11
359         // - TXD0    -> PG13
360         GPIO_InitStruct.Pin       = GPIO_PIN_2  | \
361                                     GPIO_PIN_11 | \
362                                     GPIO_PIN_13;
363         HAL_GPIO_Init(GPIOG, &GPIO_InitStruct);
364
365         // Enable the Ethernet global interrupt.
366         HAL_NVIC_SetPriority(ETH_IRQn, 7, 0);
367         HAL_NVIC_EnableIRQ  (ETH_IRQn);
368     }
369 }
370
371
372 void HAL_ETH_RxCpltCallback(ETH_HandleTypeDef *heth)
373 {
374     osSemaphoreRelease(RxPktSemaphore);
375 }
376
377
378 static void pbuf_free_custom(struct pbuf *p)
379 {
380     struct pbuf_custom *custom_pbuf = (struct pbuf_custom *)p;
381     LWIP_MEMPOOL_FREE(RX_POOL, custom_pbuf);
382 }
383
384
385 static int32_t ETH_PHY_IO_Init(void)
386 {
```

```
387     HAL_ETH_SetMDIOClockRange(&heth);
388 
389     return 0;
390 }
391 
392 
393 static int32_t ETH_PHY_IO_DeInit(void)
394 {
395     return 0;
396 }
397 
398 
399 static int32_t ETH_PHY_IO_ReadReg(uint32_t DevAddr, uint32_t RegAddr, uint32_t *
        pRegVal)
400 {
401     if(HAL_ETH_ReadPHYRegister(&heth, DevAddr, RegAddr, pRegVal) != HAL_OK)
402     {
403         return -1;
404     }
405 
406     return 0;
407 }
408 
409 
410 static int32_t ETH_PHY_IO_WriteReg(uint32_t DevAddr, uint32_t RegAddr, uint32_t
        RegVal)
411 {
412     if(HAL_ETH_WritePHYRegister(&heth, DevAddr, RegAddr, RegVal) != HAL_OK)
413     {
414         return -1;
415     }
416 
417     return 0;
418 }
419 
420 
421 static int32_t ETH_PHY_IO_GetTick(void)
422 {
423     return HAL_GetTick();
424 }
```

/NUCLEO_H723ZG_MQTT_TLS/Application/include/hw/ethernetif.h

```
1 // ----------------------------------------------------------------------------
2 // ETHERNETIF
3 //
4 // - Archivo de cabecera.
```

```
5  //
6  // ------------------------------------------------------------------------------
7  // Copyright (c) 2017 STMicroelectronics.
8  // All rights reserved.
9  //
10 // This software component is licensed by ST under BSD 3-Clause license, the
11 // "License"; You may not use this file except in compliance with the License.
12 // You may obtain a copy of the License at:
13 //
14 //                                        opensource.org/licenses/BSD-3-Clause
15 // ------------------------------------------------------------------------------
16
17 #ifndef   ETHERNETIF_H
18 #define   ETHERNETIF_H
19
20 #include  HAL_DRIVER
21
22 extern ETH_HandleTypeDef heth;
23
24 int ETH_Link_Status(void);
25
26 #endif // ETHERNETIF_H
```

B.5. Otros

B.5.1. SYSCALLS

El siguiente archivo fue generado por el IDE al crear el proyecto de programación.

/NUCLEO_H723ZG_MQTT_TLS/Application/source/sys/syscalls.c

```
1  /**
2   ******************************************************************************
3   * @file      syscalls.c
4   * @author    Auto-generated by STM32CubeIDE
5   * @brief     STM32CubeIDE Minimal System calls file
6   *
7   *            For more information about which c-functions
8   *            need which of these lowlevel functions
9   *            please consult the Newlib libc-manual
10  ******************************************************************************
11  * @attention
12  *
13  * <h2><center>&copy; Copyright (c) 2020 STMicroelectronics.
14  * All rights reserved.</center></h2>
```

```
15  *
16  * This software component is licensed by ST under BSD 3-Clause license,
17  * the "License"; You may not use this file except in compliance with the
18  * License. You may obtain a copy of the License at:
19  *                        opensource.org/licenses/BSD-3-Clause
20  *
21  ******************************************************************************
22  */
23
24 /* Includes */
25 #include <sys/stat.h>
26 #include <stdlib.h>
27 #include <errno.h>
28 #include <stdio.h>
29 #include <signal.h>
30 #include <time.h>
31 #include <sys/time.h>
32 #include <sys/times.h>
33
34
35 /* Variables */
36 extern int __io_putchar(int ch) __attribute__((weak));
37 extern int __io_getchar(void) __attribute__((weak));
38
39
40 char *__env[1] = { 0 };
41 char **environ = __env;
42
43
44 /* Functions */
45 void initialise_monitor_handles()
46 {
47 }
48
49 int _getpid(void)
50 {
51     return 1;
52 }
53
54 int _kill(int pid, int sig)
55 {
56     errno = EINVAL;
57     return -1;
58 }
59
60 void _exit (int status)
61 {
```

```
62      _kill(status, -1);
63      while (1) {}        /* Make sure we hang here */
64  }
65
66  __attribute__((weak)) int _read(int file, char *ptr, int len)
67  {
68      int DataIdx;
69
70      for (DataIdx = 0; DataIdx < len; DataIdx++)
71      {
72          *ptr++ = __io_getchar();
73      }
74
75  return len;
76  }
77
78  __attribute__((weak)) int _write(int file, char *ptr, int len)
79  {
80      int DataIdx;
81
82      for (DataIdx = 0; DataIdx < len; DataIdx++)
83      {
84          __io_putchar(*ptr++);
85      }
86      return len;
87  }
88
89  int _close(int file)
90  {
91      return -1;
92  }
93
94
95  int _fstat(int file, struct stat *st)
96  {
97      st->st_mode = S_IFCHR;
98      return 0;
99  }
100
101 int _isatty(int file)
102 {
103     return 1;
104 }
105
106 int _lseek(int file, int ptr, int dir)
107 {
108     return 0;
```

```
109 }
110
111 int _open(char *path, int flags, ...)
112 {
113     /* Pretend like we always fail */
114     return -1;
115 }
116
117 int _wait(int *status)
118 {
119     errno = ECHILD;
120     return -1;
121 }
122
123 int _unlink(char *name)
124 {
125     errno = ENOENT;
126     return -1;
127 }
128
129 int _times(struct tms *buf)
130 {
131     return -1;
132 }
133
134 int _stat(char *file, struct stat *st)
135 {
136     st->st_mode = S_IFCHR;
137     return 0;
138 }
139
140 int _link(char *old, char *new)
141 {
142     errno = EMLINK;
143     return -1;
144 }
145
146 int _fork(void)
147 {
148     errno = EAGAIN;
149     return -1;
150 }
151
152 int _execve(char *name, char **argv, char **env)
153 {
154     errno = ENOMEM;
155     return -1;
```

```
156 }
```

B.5.2. SYSMEM

El siguiente archivo fue generado por el IDE al crear el proyecto de programación.

/NUCLEO_H723ZG_MQTT_TLS/Application/source/sys/sysmem.c

```
1  /**
2   ******************************************************************************
3   * @file      sysmem.c
4   * @author    Generated by STM32CubeIDE
5   * @brief     STM32CubeIDE System Memory calls file
6   *
7   *            For more information about which C functions
8   *            need which of these lowlevel functions
9   *            please consult the newlib libc manual
10  ******************************************************************************
11  * @attention
12  *
13  * <h2><center>&copy; Copyright (c) 2020 STMicroelectronics.
14  * All rights reserved.</center></h2>
15  *
16  * This software component is licensed by ST under BSD 3-Clause license,
17  * the "License"; You may not use this file except in compliance with the
18  * License. You may obtain a copy of the License at:
19  *                        opensource.org/licenses/BSD-3-Clause
20  *
21  ******************************************************************************
22  */
23
24 /* Includes */
25 #include <errno.h>
26 #include <stdint.h>
27
28 /**
29  * Pointer to the current high watermark of the heap usage
30  */
31 static uint8_t *__sbrk_heap_end = NULL;
32
33 /**
34  * @brief _sbrk() allocates memory to the newlib heap and is used by malloc
35  *        and others from the C library
36  *
37  * @verbatim
38  * ############################################################################
```

```
 * #  .data  #  .bss  #       newlib heap       #          MSP stack          #
 * #         #        #                         # Reserved by _Min_Stack_Size #
 * ###############################################################################
 * ^-- RAM start      ^-- _end                             _estack, RAM end --^
 * @endverbatim
 *
 * This implementation starts allocating at the '_end' linker symbol
 * The '_Min_Stack_Size' linker symbol reserves a memory for the MSP stack
 * The implementation considers '_estack' linker symbol to be RAM end
 * NOTE: If the MSP stack, at any point during execution, grows larger than the
 * reserved size, please increase the '_Min_Stack_Size'.
 *
 * @param incr Memory size
 * @return Pointer to allocated memory
 */
void *_sbrk(ptrdiff_t incr)
{
  extern uint8_t _end; /* Symbol defined in the linker script */
  extern uint8_t _estack; /* Symbol defined in the linker script */
  extern uint32_t _Min_Stack_Size; /* Symbol defined in the linker script */
  const uint32_t stack_limit = (uint32_t)&_estack - (uint32_t)&_Min_Stack_Size;
  const uint8_t *max_heap = (uint8_t *)stack_limit;
  uint8_t *prev_heap_end;

  /* Initialize heap end at first call */
  if (NULL == __sbrk_heap_end)
  {
    __sbrk_heap_end = &_end;
  }

  /* Protect heap from growing into the reserved MSP stack */
  if (__sbrk_heap_end + incr > max_heap)
  {
    errno = ENOMEM;
    return (void *)-1;
  }

  prev_heap_end = __sbrk_heap_end;
  __sbrk_heap_end += incr;

  return (void *)prev_heap_end;
}
```

B.5.3. STARTUP

El siguiente archivo fue generado por el IDE al crear el proyecto de programación.

/NUCLEO_H723ZG_MQTT_TLS/Application/startup/startup_stm32h723zgtx.s

```
/**
 ******************************************************************************
 * @file      startup_stm32h723zgtx.s
 * @author    Auto-generated by STM32CubeIDE
 * @brief     STM32H723ZGTx device vector table for GCC toolchain.
 *            This module performs:
 *                - Set the initial SP
 *                - Set the initial PC == Reset_Handler,
 *                - Set the vector table entries with the exceptions ISR address
 *                - Branches to main in the C library (which eventually
 *                  calls main()).
 ******************************************************************************
 * @attention
 *
 * <h2><center>&copy; Copyright (c) 2020 STMicroelectronics.
 * All rights reserved.</center></h2>
 *
 * This software component is licensed by ST under BSD 3-Clause license,
 * the "License"; You may not use this file except in compliance with the
 * License. You may obtain a copy of the License at:
 *                        opensource.org/licenses/BSD-3-Clause
 *
 ******************************************************************************
 */

.syntax unified
.cpu cortex-m7
.fpu softvfp
.thumb

.global g_pfnVectors
.global Default_Handler

/* start address for the initialization values of the .data section.
defined in linker script */
.word _sidata
/* start address for the .data section. defined in linker script */
.word _sdata
/* end address for the .data section. defined in linker script */
.word _edata
/* start address for the .bss section. defined in linker script */
.word _sbss
/* end address for the .bss section. defined in linker script */
.word _ebss

/**
 * @brief  This is the code that gets called when the processor first
 *          starts execution following a reset event. Only the absolutely
 *          necessary set is performed, after which the application
 *          supplied main() routine is called.
 * @param  None
```

```
52  * @retval : None
53 */
54
55   .section .text.Reset_Handler
56   .weak Reset_Handler
57   .type Reset_Handler, %function
58 Reset_Handler:
59   ldr   r0, =_estack
60   mov   sp, r0          /* set stack pointer */
61 /* Call the clock system intitialization function.*/
62   bl  SystemInit
63
64 /* Copy the data segment initializers from flash to SRAM */
65   ldr r0, =_sdata
66   ldr r1, =_edata
67   ldr r2, =_sidata
68   movs r3, #0
69   b LoopCopyDataInit
70
71 CopyDataInit:
72   ldr r4, [r2, r3]
73   str r4, [r0, r3]
74   adds r3, r3, #4
75
76 LoopCopyDataInit:
77   adds r4, r0, r3
78   cmp r4, r1
79   bcc CopyDataInit
80
81 /* Zero fill the bss segment. */
82   ldr r2, =_sbss
83   ldr r4, =_ebss
84   movs r3, #0
85   b LoopFillZerobss
86
87 FillZerobss:
88   str  r3, [r2]
89   adds r2, r2, #4
90
91 LoopFillZerobss:
92   cmp r2, r4
93   bcc FillZerobss
94
95 /* Call static constructors */
96   bl __libc_init_array
97 /* Call the application's entry point.*/
98   bl main
99
100 LoopForever:
101     b LoopForever
102
103   .size Reset_Handler, .-Reset_Handler
104
105 /**
106  * @brief  This is the code that gets called when the processor receives an
107  *         unexpected interrupt.  This simply enters an infinite loop, preserving
108  *         the system state for examination by a debugger.
109  *
110  * @param  None
```

```
111  * @retval : None
112 */
113   .section .text.Default_Handler,"ax",%progbits
114 Default_Handler:
115 Infinite_Loop:
116   b Infinite_Loop
117   .size Default_Handler, .-Default_Handler
118
119 /******************************************************************************
120 *
121 * The STM32H723ZGTx vector table.  Note that the proper constructs
122 * must be placed on this to ensure that it ends up at physical address
123 * 0x0000.0000.
124 *
125 ******************************************************************************/
126   .section .isr_vector,"a",%progbits
127   .type g_pfnVectors, %object
128   .size g_pfnVectors, .-g_pfnVectors
129
130 g_pfnVectors:
131   .word _estack
132   .word Reset_Handler
133   .word NMI_Handler
134   .word HardFault_Handler
135   .word MemManage_Handler
136   .word BusFault_Handler
137   .word UsageFault_Handler
138   .word 0
139   .word 0
140   .word 0
141   .word 0
142   .word SVC_Handler
143   .word DebugMon_Handler
144   .word 0
145   .word PendSV_Handler
146   .word SysTick_Handler
147   .word WWDG1_IRQHandler                          /* Window Watchdog interrupt                */
148   .word PVD_PVM_IRQHandler                        /* PVD through EXTI line                    */
149   .word RTC_TAMP_STAMP_CSS_LSE_IRQHandler         /* RTC tamper, timestamp                    */
150   .word RTC_WKUP_IRQHandler                       /* RTC Wakeup interrupt                     */
151   .word FLASH_IRQHandler                          /* Flash memory                             */
152   .word RCC_IRQHandler                            /* RCC global interrupt                     */
153   .word EXTI0_IRQHandler                          /* EXTI Line 0 interrupt                    */
154   .word EXTI1_IRQHandler                          /* EXTI Line 1 interrupt                    */
155   .word EXTI2_IRQHandler                          /* EXTI Line 2 interrupt                    */
156   .word EXTI3_IRQHandler                          /* EXTI Line 3interrupt                     */
157   .word EXTI4_IRQHandler                          /* EXTI Line 4interrupt                     */
158   .word DMA_STR0_IRQHandler                       /* DMA1 Stream0                             */
159   .word DMA_STR1_IRQHandler                       /* DMA1 Stream1                             */
160   .word DMA_STR2_IRQHandler                       /* DMA1 Stream2                             */
161   .word DMA_STR3_IRQHandler                       /* DMA1 Stream3                             */
162   .word DMA_STR4_IRQHandler                       /* DMA1 Stream4                             */
163   .word DMA_STR5_IRQHandler                       /* DMA1 Stream5                             */
164   .word DMA_STR6_IRQHandler                       /* DMA1 Stream6                             */
165   .word ADC1_2_IRQHandler                         /* ADC1 and ADC2                            */
166   .word FDCAN1_IT0_IRQHandler                     /* FDCAN1 Interrupt 0                       */
167   .word FDCAN2_IT0_IRQHandler                     /* FDCAN2 Interrupt 0                       */
168   .word FDCAN1_IT1_IRQHandler                     /* FDCAN1 Interrupt 1                       */
169   .word FDCAN2_IT1_IRQHandler                     /* FDCAN2 Interrupt 1                       */
```

```
170   .word EXTI9_5_IRQHandler                  /* EXTI Line[9:5] interrupts                  */
171   .word TIM1_BRK_IRQHandler                  /* TIM1 break interrupt                       */
172   .word TIM1_UP_IRQHandler                   /* TIM1 update interrupt                      */
173   .word TIM1_TRG_COM_IRQHandler              /* TIM1 trigger and commutation               */
174   .word TIM_CC_IRQHandler                    /* TIM1 capture / compare                     */
175   .word TIM2_IRQHandler                      /* TIM2 global interrupt                      */
176   .word TIM3_IRQHandler                      /* TIM3 global interrupt                      */
177   .word TIM4_IRQHandler                      /* TIM4 global interrupt                      */
178   .word I2C1_EV_IRQHandler                   /* I2C1 event interrupt                       */
179   .word I2C1_ER_IRQHandler                   /* I2C1 global error interrupt                */
180   .word I2C2_EV_IRQHandler                   /* I2C2 event interrupt                       */
181   .word I2C2_ER_IRQHandler                   /* I2C2 global error interrupt                */
182   .word SPI1_IRQHandler                      /* SPI1 global interrupt                      */
183   .word SPI2_IRQHandler                      /* SPI2 global interrupt                      */
184   .word USART1_IRQHandler                    /* USART1 global interrupt                    */
185   .word USART2_IRQHandler                    /* USART2 global interrupt                    */
186   .word USART3_IRQHandler                    /* USART3 global interrupt                    */
187   .word EXTI15_10_IRQHandler                 /* EXTI Line[15:10] interrupts                */
188   .word RTC_ALARM_IRQHandler                 /* RTC alarms (A and B)                       */
189   .word 0                                    /* Reserved                                   */
190   .word TIM8_BRK_TIM12_IRQHandler            /* TIM8 and 12 break global                   */
191   .word TIM8_UP_TIM13_IRQHandler             /* TIM8 and 13 update global                  */
192   .word TIM8_TRG_COM_TIM14_IRQHandler        /* TIM8 and 14 trigger /commutation and global */
193   .word TIM8_CC_IRQHandler                   /* TIM8 capture / compare                     */
194   .word DMA1_STR7_IRQHandler                 /* DMA1 Stream7                               */
195   .word FMC_IRQHandler                       /* FMC global interrupt                       */
196   .word SDMMC1_IRQHandler                    /* SDMMC1 global interrupt                    */
197   .word TIM5_IRQHandler                      /* TIM5 global interrupt                      */
198   .word SPI3_IRQHandler                      /* SPI3 global interrupt                      */
199   .word UART4_IRQHandler                     /* UART4 global interrupt                     */
200   .word UART5_IRQHandler                     /* UART5 global interrupt                     */
201   .word TIM6_DAC_IRQHandler                  /* TIM6 global interrupt                      */
202   .word TIM7_IRQHandler                      /* TIM7 global interrupt                      */
203   .word DMA2_STR0_IRQHandler                 /* DMA2 Stream0 interrupt                     */
204   .word DMA2_STR1_IRQHandler                 /* DMA2 Stream1 interrupt                     */
205   .word DMA2_STR2_IRQHandler                 /* DMA2 Stream2 interrupt                     */
206   .word DMA2_STR3_IRQHandler                 /* DMA2 Stream3 interrupt                     */
207   .word DMA2_STR4_IRQHandler                 /* DMA2 Stream4 interrupt                     */
208   .word ETH_IRQHandler                       /* Ethernet global interrupt                  */
209   .word ETH_WKUP_IRQHandler                  /* Ethernet wakeup through EXTI               */
210   .word FDCAN_CAL_IRQHandler                 /* CAN2TX interrupts                          */
211   .word 0                                    /* Reserved                                   */
212   .word 0                                    /* Reserved                                   */
213   .word 0                                    /* Reserved                                   */
214   .word 0                                    /* Reserved                                   */
215   .word DMA2_STR5_IRQHandler                 /* DMA2 Stream5 interrupt                     */
216   .word DMA2_STR6_IRQHandler                 /* DMA2 Stream6 interrupt                     */
217   .word DMA2_STR7_IRQHandler                 /* DMA2 Stream7 interrupt                     */
218   .word USART6_IRQHandler                    /* USART6 global interrupt                    */
219   .word I2C3_EV_IRQHandler                   /* I2C3 event interrupt                       */
220   .word I2C3_ER_IRQHandler                   /* I2C3 error interrupt                       */
221   .word 0                                    /* Reserved                                   */
222   .word 0                                    /* Reserved                                   */
223   .word 0                                    /* Reserved                                   */
224   .word 0                                    /* Reserved                                   */
225   .word DCMI_IRQHandler                      /* DCMI global interrupt                      */
226   .word 0                                    /* Reserved                                   */
227   .word 0                                    /* Reserved                                   */
228   .word FPU_IRQHandler                       /* Floating point unit interrupt              */
```

```
229   .word UART7_IRQHandler                  /* UART7 global interrupt                    */
230   .word UART8_IRQHandler                  /* UART8 global interrupt                    */
231   .word SPI4_IRQHandler                   /* SPI4 global interrupt                     */
232   .word SPI5_IRQHandler                   /* SPI5 global interrupt                     */
233   .word SPI6_IRQHandler                   /* SPI6 global interrupt                     */
234   .word SAI1_IRQHandler                   /* SAI1 global interrupt                     */
235   .word LTDC_IRQHandler                   /* LCD-TFT global interrupt                  */
236   .word LTDC_ER_IRQHandler                /* LCD-TFT error interrupt                   */
237   .word DMA2D_IRQHandler                  /* DMA2D global interrupt                    */
238   .word 0                                 /* Reserved                                  */
239   .word OCTOSPI1_IRQHandler               /* OCTOSPI1 global interrupt                 */
240   .word LPTIM1_IRQHandler                 /* LPTIM1 global interrupt                   */
241   .word CEC_IRQHandler                    /* HDMI-CEC global interrupt                 */
242   .word I2C4_EV_IRQHandler                /* I2C4 event interrupt                      */
243   .word I2C4_ER_IRQHandler                /* I2C4 error interrupt                      */
244   .word SPDIF_IRQHandler                  /* SPDIFRX global interrupt                  */
245   .word 0                                 /* Reserved                                  */
246   .word 0                                 /* Reserved                                  */
247   .word 0                                 /* Reserved                                  */
248   .word 0                                 /* Reserved                                  */
249   .word DMAMUX1_OV_IRQHandler             /* DMAMUX1 overrun interrupt                 */
250   .word 0                                 /* Reserved                                  */
251   .word 0                                 /* Reserved                                  */
252   .word 0                                 /* Reserved                                  */
253   .word 0                                 /* Reserved                                  */
254   .word 0                                 /* Reserved                                  */
255   .word 0                                 /* Reserved                                  */
256   .word 0                                 /* Reserved                                  */
257   .word DFSDM1_FLT0_IRQHandler            /* DFSDM1 filter 0 interrupt                 */
258   .word DFSDM1_FLT1_IRQHandler            /* DFSDM1 filter 1 interrupt                 */
259   .word DFSDM1_FLT2_IRQHandler            /* DFSDM1 filter 2 interrupt                 */
260   .word DFSDM1_FLT3_IRQHandler            /* DFSDM1 filter 3 interrupt                 */
261   .word 0                                 /* Reserved                                  */
262   .word SWPMI1_IRQHandler                 /* SWPMI global interrupt                    */
263   .word TIM15_IRQHandler                  /* TIM15 global interrupt                    */
264   .word TIM16_IRQHandler                  /* TIM16 global interrupt                    */
265   .word TIM17_IRQHandler                  /* TIM17 global interrupt                    */
266   .word MDIOS_WKUP_IRQHandler             /* MDIOS wakeup                              */
267   .word MDIOS_IRQHandler                  /* MDIOS global interrupt                    */
268   .word 0                                 /* Reserved                                  */
269   .word MDMA_IRQHandler                   /* MDMA Global interrupt                     */
270   .word 0                                 /* Reserved                                  */
271   .word SDMMC2_IRQHandler                 /* SDMMC2 global interrupt                   */
272   .word HSEM0_IRQHandler                  /* HSEM global interrupt 1                   */
273   .word 0                                 /* Reserved                                  */
274   .word ADC3_IRQHandler                   /* ADC3 global interrupt                     */
275   .word DMAMUX2_OVR_IRQHandler            /* DMAMUX D3 overrun interrupt               */
276   .word 0                                 /* Reserved                                  */
277   .word 0                                 /* Reserved                                  */
278   .word 0                                 /* Reserved                                  */
279   .word 0                                 /* Reserved                                  */
280   .word 0                                 /* Reserved                                  */
281   .word 0                                 /* Reserved                                  */
282   .word 0                                 /* Reserved                                  */
283   .word 0                                 /* Reserved                                  */
284   .word COMP_IRQHandler                   /* COMP1 and COMP2 global interrupt          */
285   .word LPTIM2_IRQHandler                 /* LPTIM2 timer interrupt                    */
286   .word LPTIM3_IRQHandler                 /* LPTIM2 timer interrupt                    */
287   .word LPTIM4_IRQHandler                 /* LPTIM2 timer interrupt                    */
```

```
  .word LPTIM5_IRQHandler                   /* LPTIM2 timer interrupt                    */
  .word LPUART_IRQHandler                   /* LPUART global interrupt                   */
  .word 0                                   /* Reserved                                  */
  .word CRS_IRQHandler                      /* Clock Recovery System global interrupt    */
  .word 0                                   /* Reserved                                  */
  .word SAI4_IRQHandler                     /* SAI4 global interrupt                     */
  .word 0                                   /* Reserved                                  */
  .word 0                                   /* Reserved                                  */
  .word WKUP_IRQHandler                     /* WKUP1 to WKUP6 pins                       */
  .word OCTOSPI2_IRQHandler                 /* OCTOSPI2 global interrupt                 */
  .word 0                                   /* Reserved                                  */
  .word 0                                   /* Reserved                                  */
  .word FMAC_IRQHandler                     /* FMAC interrupt                            */
  .word CORDIC_IT_IRQHandler                /* CORDIC interrupt                          */
  .word 0                                   /* Reserved                                  */
  .word USART10_IRQHandler                  /* USART10 interrupt                         */
  .word I2C5_EV_IRQHandler                  /* I2C5 event interrupt                      */
  .word I2C5_ER_IRQHandler                  /* I2C5 error interrupt                      */
  .word FDCAN3_IT0_IRQHandler               /* FDCAN3 Interrupt 0                        */
  .word FDCAN3_IT1_IRQHandler               /* FDCAN3 Interrupt 1                        */
  .word TIM23_IRQHandler                    /* TIM23 global interrupt                    */
  .word TIM24_IRQHandler                    /* TIM24 global interrupt                    */

/*******************************************************************************
*
* Provide weak aliases for each Exception handler to the Default_Handler.
* As they are weak aliases, any function with the same name will override
* this definition.
*
*******************************************************************************/

    .weak   NMI_Handler
    .thumb_set NMI_Handler,Default_Handler

    .weak   HardFault_Handler
    .thumb_set HardFault_Handler,Default_Handler

    .weak   MemManage_Handler
    .thumb_set MemManage_Handler,Default_Handler

    .weak   BusFault_Handler
    .thumb_set BusFault_Handler,Default_Handler

    .weak   UsageFault_Handler
    .thumb_set UsageFault_Handler,Default_Handler

    .weak   SVC_Handler
    .thumb_set SVC_Handler,Default_Handler

    .weak   DebugMon_Handler
    .thumb_set DebugMon_Handler,Default_Handler

    .weak   PendSV_Handler
    .thumb_set PendSV_Handler,Default_Handler

    .weak   SysTick_Handler
    .thumb_set SysTick_Handler,Default_Handler

    .weak   WWDG1_IRQHandler
```

```
347     .thumb_set WWDG1_IRQHandler,Default_Handler
348
349     .weak   PVD_PVM_IRQHandler
350     .thumb_set PVD_PVM_IRQHandler,Default_Handler
351
352     .weak   RTC_TAMP_STAMP_CSS_LSE_IRQHandler
353     .thumb_set RTC_TAMP_STAMP_CSS_LSE_IRQHandler,Default_Handler
354
355     .weak   RTC_WKUP_IRQHandler
356     .thumb_set RTC_WKUP_IRQHandler,Default_Handler
357
358     .weak   FLASH_IRQHandler
359     .thumb_set FLASH_IRQHandler,Default_Handler
360
361     .weak   RCC_IRQHandler
362     .thumb_set RCC_IRQHandler,Default_Handler
363
364     .weak   EXTI0_IRQHandler
365     .thumb_set EXTI0_IRQHandler,Default_Handler
366
367     .weak   EXTI1_IRQHandler
368     .thumb_set EXTI1_IRQHandler,Default_Handler
369
370     .weak   EXTI2_IRQHandler
371     .thumb_set EXTI2_IRQHandler,Default_Handler
372
373     .weak   EXTI3_IRQHandler
374     .thumb_set EXTI3_IRQHandler,Default_Handler
375
376     .weak   EXTI4_IRQHandler
377     .thumb_set EXTI4_IRQHandler,Default_Handler
378
379     .weak   DMA_STR0_IRQHandler
380     .thumb_set DMA_STR0_IRQHandler,Default_Handler
381
382     .weak   DMA_STR1_IRQHandler
383     .thumb_set DMA_STR1_IRQHandler,Default_Handler
384
385     .weak   DMA_STR2_IRQHandler
386     .thumb_set DMA_STR2_IRQHandler,Default_Handler
387
388     .weak   DMA_STR3_IRQHandler
389     .thumb_set DMA_STR3_IRQHandler,Default_Handler
390
391     .weak   DMA_STR4_IRQHandler
392     .thumb_set DMA_STR4_IRQHandler,Default_Handler
393
394     .weak   DMA_STR5_IRQHandler
395     .thumb_set DMA_STR5_IRQHandler,Default_Handler
396
397     .weak   DMA_STR6_IRQHandler
398     .thumb_set DMA_STR6_IRQHandler,Default_Handler
399
400     .weak   ADC1_2_IRQHandler
401     .thumb_set ADC1_2_IRQHandler,Default_Handler
402
403     .weak   FDCAN1_IT0_IRQHandler
404     .thumb_set FDCAN1_IT0_IRQHandler,Default_Handler
405
```

```
406     .weak   FDCAN2_IT0_IRQHandler
407     .thumb_set FDCAN2_IT0_IRQHandler,Default_Handler
408
409     .weak   FDCAN1_IT1_IRQHandler
410     .thumb_set FDCAN1_IT1_IRQHandler,Default_Handler
411
412     .weak   FDCAN2_IT1_IRQHandler
413     .thumb_set FDCAN2_IT1_IRQHandler,Default_Handler
414
415     .weak   EXTI9_5_IRQHandler
416     .thumb_set EXTI9_5_IRQHandler,Default_Handler
417
418     .weak   TIM1_BRK_IRQHandler
419     .thumb_set TIM1_BRK_IRQHandler,Default_Handler
420
421     .weak   TIM1_UP_IRQHandler
422     .thumb_set TIM1_UP_IRQHandler,Default_Handler
423
424     .weak   TIM1_TRG_COM_IRQHandler
425     .thumb_set TIM1_TRG_COM_IRQHandler,Default_Handler
426
427     .weak   TIM_CC_IRQHandler
428     .thumb_set TIM_CC_IRQHandler,Default_Handler
429
430     .weak   TIM2_IRQHandler
431     .thumb_set TIM2_IRQHandler,Default_Handler
432
433     .weak   TIM3_IRQHandler
434     .thumb_set TIM3_IRQHandler,Default_Handler
435
436     .weak   TIM4_IRQHandler
437     .thumb_set TIM4_IRQHandler,Default_Handler
438
439     .weak   I2C1_EV_IRQHandler
440     .thumb_set I2C1_EV_IRQHandler,Default_Handler
441
442     .weak   I2C1_ER_IRQHandler
443     .thumb_set I2C1_ER_IRQHandler,Default_Handler
444
445     .weak   I2C2_EV_IRQHandler
446     .thumb_set I2C2_EV_IRQHandler,Default_Handler
447
448     .weak   I2C2_ER_IRQHandler
449     .thumb_set I2C2_ER_IRQHandler,Default_Handler
450
451     .weak   SPI1_IRQHandler
452     .thumb_set SPI1_IRQHandler,Default_Handler
453
454     .weak   SPI2_IRQHandler
455     .thumb_set SPI2_IRQHandler,Default_Handler
456
457     .weak   USART1_IRQHandler
458     .thumb_set USART1_IRQHandler,Default_Handler
459
460     .weak   USART2_IRQHandler
461     .thumb_set USART2_IRQHandler,Default_Handler
462
463     .weak   USART3_IRQHandler
464     .thumb_set USART3_IRQHandler,Default_Handler
```

```
465
466     .weak   EXTI15_10_IRQHandler
467     .thumb_set EXTI15_10_IRQHandler,Default_Handler
468
469     .weak   RTC_ALARM_IRQHandler
470     .thumb_set RTC_ALARM_IRQHandler,Default_Handler
471
472     .weak   TIM8_BRK_TIM12_IRQHandler
473     .thumb_set TIM8_BRK_TIM12_IRQHandler,Default_Handler
474
475     .weak   TIM8_UP_TIM13_IRQHandler
476     .thumb_set TIM8_UP_TIM13_IRQHandler,Default_Handler
477
478     .weak   TIM8_TRG_COM_TIM14_IRQHandler
479     .thumb_set TIM8_TRG_COM_TIM14_IRQHandler,Default_Handler
480
481     .weak   TIM8_CC_IRQHandler
482     .thumb_set TIM8_CC_IRQHandler,Default_Handler
483
484     .weak   DMA1_STR7_IRQHandler
485     .thumb_set DMA1_STR7_IRQHandler,Default_Handler
486
487     .weak   FMC_IRQHandler
488     .thumb_set FMC_IRQHandler,Default_Handler
489
490     .weak   SDMMC1_IRQHandler
491     .thumb_set SDMMC1_IRQHandler,Default_Handler
492
493     .weak   TIM5_IRQHandler
494     .thumb_set TIM5_IRQHandler,Default_Handler
495
496     .weak   SPI3_IRQHandler
497     .thumb_set SPI3_IRQHandler,Default_Handler
498
499     .weak   UART4_IRQHandler
500     .thumb_set UART4_IRQHandler,Default_Handler
501
502     .weak   UART5_IRQHandler
503     .thumb_set UART5_IRQHandler,Default_Handler
504
505     .weak   TIM6_DAC_IRQHandler
506     .thumb_set TIM6_DAC_IRQHandler,Default_Handler
507
508     .weak   TIM7_IRQHandler
509     .thumb_set TIM7_IRQHandler,Default_Handler
510
511     .weak   DMA2_STR0_IRQHandler
512     .thumb_set DMA2_STR0_IRQHandler,Default_Handler
513
514     .weak   DMA2_STR1_IRQHandler
515     .thumb_set DMA2_STR1_IRQHandler,Default_Handler
516
517     .weak   DMA2_STR2_IRQHandler
518     .thumb_set DMA2_STR2_IRQHandler,Default_Handler
519
520     .weak   DMA2_STR3_IRQHandler
521     .thumb_set DMA2_STR3_IRQHandler,Default_Handler
522
523     .weak   DMA2_STR4_IRQHandler
```

```
524     .thumb_set DMA2_STR4_IRQHandler,Default_Handler
525
526     .weak   ETH_IRQHandler
527     .thumb_set ETH_IRQHandler,Default_Handler
528
529     .weak   ETH_WKUP_IRQHandler
530     .thumb_set ETH_WKUP_IRQHandler,Default_Handler
531
532     .weak   FDCAN_CAL_IRQHandler
533     .thumb_set FDCAN_CAL_IRQHandler,Default_Handler
534
535     .weak   DMA2_STR5_IRQHandler
536     .thumb_set DMA2_STR5_IRQHandler,Default_Handler
537
538     .weak   DMA2_STR6_IRQHandler
539     .thumb_set DMA2_STR6_IRQHandler,Default_Handler
540
541     .weak   DMA2_STR7_IRQHandler
542     .thumb_set DMA2_STR7_IRQHandler,Default_Handler
543
544     .weak   USART6_IRQHandler
545     .thumb_set USART6_IRQHandler,Default_Handler
546
547     .weak   I2C3_EV_IRQHandler
548     .thumb_set I2C3_EV_IRQHandler,Default_Handler
549
550     .weak   I2C3_ER_IRQHandler
551     .thumb_set I2C3_ER_IRQHandler,Default_Handler
552
553     .weak   DCMI_IRQHandler
554     .thumb_set DCMI_IRQHandler,Default_Handler
555
556     .weak   FPU_IRQHandler
557     .thumb_set FPU_IRQHandler,Default_Handler
558
559     .weak   UART7_IRQHandler
560     .thumb_set UART7_IRQHandler,Default_Handler
561
562     .weak   UART8_IRQHandler
563     .thumb_set UART8_IRQHandler,Default_Handler
564
565     .weak   SPI4_IRQHandler
566     .thumb_set SPI4_IRQHandler,Default_Handler
567
568     .weak   SPI5_IRQHandler
569     .thumb_set SPI5_IRQHandler,Default_Handler
570
571     .weak   SPI6_IRQHandler
572     .thumb_set SPI6_IRQHandler,Default_Handler
573
574     .weak   SAI1_IRQHandler
575     .thumb_set SAI1_IRQHandler,Default_Handler
576
577     .weak   LTDC_IRQHandler
578     .thumb_set LTDC_IRQHandler,Default_Handler
579
580     .weak   LTDC_ER_IRQHandler
581     .thumb_set LTDC_ER_IRQHandler,Default_Handler
582
```

```
583     .weak   DMA2D_IRQHandler
584     .thumb_set DMA2D_IRQHandler,Default_Handler
585
586     .weak   OCTOSPI1_IRQHandler
587     .thumb_set OCTOSPI1_IRQHandler,Default_Handler
588
589     .weak   LPTIM1_IRQHandler
590     .thumb_set LPTIM1_IRQHandler,Default_Handler
591
592     .weak   CEC_IRQHandler
593     .thumb_set CEC_IRQHandler,Default_Handler
594
595     .weak   I2C4_EV_IRQHandler
596     .thumb_set I2C4_EV_IRQHandler,Default_Handler
597
598     .weak   I2C4_ER_IRQHandler
599     .thumb_set I2C4_ER_IRQHandler,Default_Handler
600
601     .weak   SPDIF_IRQHandler
602     .thumb_set SPDIF_IRQHandler,Default_Handler
603
604     .weak   DMAMUX1_OV_IRQHandler
605     .thumb_set DMAMUX1_OV_IRQHandler,Default_Handler
606
607     .weak   DFSDM1_FLT0_IRQHandler
608     .thumb_set DFSDM1_FLT0_IRQHandler,Default_Handler
609
610     .weak   DFSDM1_FLT1_IRQHandler
611     .thumb_set DFSDM1_FLT1_IRQHandler,Default_Handler
612
613     .weak   DFSDM1_FLT2_IRQHandler
614     .thumb_set DFSDM1_FLT2_IRQHandler,Default_Handler
615
616     .weak   DFSDM1_FLT3_IRQHandler
617     .thumb_set DFSDM1_FLT3_IRQHandler,Default_Handler
618
619     .weak   SWPMI1_IRQHandler
620     .thumb_set SWPMI1_IRQHandler,Default_Handler
621
622     .weak   TIM15_IRQHandler
623     .thumb_set TIM15_IRQHandler,Default_Handler
624
625     .weak   TIM16_IRQHandler
626     .thumb_set TIM16_IRQHandler,Default_Handler
627
628     .weak   TIM17_IRQHandler
629     .thumb_set TIM17_IRQHandler,Default_Handler
630
631     .weak   MDIOS_WKUP_IRQHandler
632     .thumb_set MDIOS_WKUP_IRQHandler,Default_Handler
633
634     .weak   MDIOS_IRQHandler
635     .thumb_set MDIOS_IRQHandler,Default_Handler
636
637     .weak   MDMA_IRQHandler
638     .thumb_set MDMA_IRQHandler,Default_Handler
639
640     .weak   SDMMC2_IRQHandler
641     .thumb_set SDMMC2_IRQHandler,Default_Handler
```

```
642
643     .weak   HSEM0_IRQHandler
644     .thumb_set HSEM0_IRQHandler,Default_Handler
645
646     .weak   ADC3_IRQHandler
647     .thumb_set ADC3_IRQHandler,Default_Handler
648
649     .weak   DMAMUX2_OVR_IRQHandler
650     .thumb_set DMAMUX2_OVR_IRQHandler,Default_Handler
651
652     .weak   COMP_IRQHandler
653     .thumb_set COMP_IRQHandler,Default_Handler
654
655     .weak   LPTIM2_IRQHandler
656     .thumb_set LPTIM2_IRQHandler,Default_Handler
657
658     .weak   LPTIM3_IRQHandler
659     .thumb_set LPTIM3_IRQHandler,Default_Handler
660
661     .weak   LPTIM4_IRQHandler
662     .thumb_set LPTIM4_IRQHandler,Default_Handler
663
664     .weak   LPTIM5_IRQHandler
665     .thumb_set LPTIM5_IRQHandler,Default_Handler
666
667     .weak   LPUART_IRQHandler
668     .thumb_set LPUART_IRQHandler,Default_Handler
669
670     .weak   CRS_IRQHandler
671     .thumb_set CRS_IRQHandler,Default_Handler
672
673     .weak   SAI4_IRQHandler
674     .thumb_set SAI4_IRQHandler,Default_Handler
675
676     .weak   WKUP_IRQHandler
677     .thumb_set WKUP_IRQHandler,Default_Handler
678
679     .weak   OCTOSPI2_IRQHandler
680     .thumb_set OCTOSPI2_IRQHandler,Default_Handler
681
682     .weak   FMAC_IRQHandler
683     .thumb_set FMAC_IRQHandler,Default_Handler
684
685     .weak   CORDIC_IT_IRQHandler
686     .thumb_set CORDIC_IT_IRQHandler,Default_Handler
687
688     .weak   USART10_IRQHandler
689     .thumb_set USART10_IRQHandler,Default_Handler
690
691     .weak   I2C5_EV_IRQHandler
692     .thumb_set I2C5_EV_IRQHandler,Default_Handler
693
694     .weak   I2C5_ER_IRQHandler
695     .thumb_set I2C5_ER_IRQHandler,Default_Handler
696
697     .weak   FDCAN3_IT0_IRQHandler
698     .thumb_set FDCAN3_IT0_IRQHandler,Default_Handler
699
700     .weak   FDCAN3_IT1_IRQHandler
```

```
701     .thumb_set FDCAN3_IT1_IRQHandler,Default_Handler
702
703     .weak   TIM23_IRQHandler
704     .thumb_set TIM23_IRQHandler,Default_Handler
705
706     .weak   TIM24_IRQHandler
707     .thumb_set TIM24_IRQHandler,Default_Handler
708
709     .weak   SystemInit
710
711 /*********************** (C) COPYRIGHT STMicroelectonics *****END OF FILE****/
```

B.5.4. LINKER SCRIPT

El siguiente archivo fue generado por el IDE al crear el proyecto de programación y cuenta con modificaciones del autor de este documento.

/NUCLEO_H723ZG_MQTT_TLS/STM32H723ZGTX_FLASH.ld

```
1  /*
2  ******************************************************************************
3  **
4  **  File        : LinkerScript.ld
5  **
6  **  Author      : STM32CubeIDE
7  **
8  **  Abstract    : Linker script for STM32H7 series
9  **                1024Kbytes FLASH and 560Kbytes RAM
10 **
11 **                Set heap size, stack size and stack location according
12 **                to application requirements.
13 **
14 **                Set memory bank area and size if external memory is used.
15 **
16 **  Target      : STMicroelectronics STM32
17 **
18 **  Distribution: The file is distributed as is, without any warranty
19 **                of any kind.
20 **
21 ******************************************************************************
22 ** @attention
23 **
24 ** <h2><center>&copy; Copyright (c) 2021 STMicroelectronics.
25 ** All rights reserved.</center></h2>
26 **
27 ** This software component is licensed by ST under BSD 3-Clause license,
28 ** the "License"; You may not use this file except in compliance with the
29 ** License. You may obtain a copy of the License at:
```

```
30 **                        opensource.org/licenses/BSD-3-Clause
31 **
32 *****************************************************************************
33 */
34
35 /* Entry Point */
36 ENTRY(Reset_Handler)
37
38 /* Highest address of the user mode stack */
39 _estack = ORIGIN(RAM_D1) + LENGTH(RAM_D1);    /* end of RAM */
40 /* Generate a link error if heap and stack don't fit into RAM */
41 _Min_Heap_Size = 0x200 ;      /* required amount of heap  */
42 _Min_Stack_Size = 0x400 ; /* required amount of stack */
43
44 /* Specify the memory areas */
45 MEMORY
46 {
47   ITCMRAM (xrw)    : ORIGIN = 0x00000000,   LENGTH = 64K
48   DTCMRAM (xrw)    : ORIGIN = 0x20000000,   LENGTH = 128K
49   FLASH    (rx)    : ORIGIN = 0x08000000,   LENGTH = 1024K
50   RAM_D1   (xrw)   : ORIGIN = 0x24000000,   LENGTH = 320K
51   RAM_D2   (xrw)   : ORIGIN = 0x30000000,   LENGTH = 32K
52   RAM_D3   (xrw)   : ORIGIN = 0x38000000,   LENGTH = 16K
53 }
54
55 /* Define output sections */
56 SECTIONS
57 {
58   /* The startup code goes first into FLASH */
59   .isr_vector :
60   {
61     . = ALIGN(4);
62     KEEP(*(.isr_vector)) /* Startup code */
63     . = ALIGN(4);
64   } >FLASH
65
66   /* The program code and other data goes into FLASH */
67   .text :
68   {
69     . = ALIGN(4);
70     *(.text)           /* .text sections (code) */
71     *(.text*)          /* .text* sections (code) */
72     *(.glue_7)         /* glue arm to thumb code */
73     *(.glue_7t)        /* glue thumb to arm code */
74     *(.eh_frame)
75
76     KEEP (*(.init))
```

```
77    KEEP (*(.fini))
78
79    . = ALIGN(4);
80    _etext = .;        /* define a global symbols at end of code */
81  } >FLASH
82
83  /* Constant data goes into FLASH */
84  .rodata :
85  {
86    . = ALIGN(4);
87    *(.rodata)         /* .rodata sections (constants, strings, etc.) */
88    *(.rodata*)        /* .rodata* sections (constants, strings, etc.) */
89    . = ALIGN(4);
90  } >FLASH
91
92  .ARM.extab   : { *(.ARM.extab* .gnu.linkonce.armextab.*) } >FLASH
93  .ARM : {
94    __exidx_start = .;
95    *(.ARM.exidx*)
96    __exidx_end = .;
97  } >FLASH
98
99  .preinit_array     :
100 {
101   PROVIDE_HIDDEN (__preinit_array_start = .);
102   KEEP (*(.preinit_array*))
103   PROVIDE_HIDDEN (__preinit_array_end = .);
104 } >FLASH
105
106 .init_array :
107 {
108   PROVIDE_HIDDEN (__init_array_start = .);
109   KEEP (*(SORT(.init_array.*)))
110   KEEP (*(.init_array*))
111   PROVIDE_HIDDEN (__init_array_end = .);
112 } >FLASH
113
114 .fini_array :
115 {
116   PROVIDE_HIDDEN (__fini_array_start = .);
117   KEEP (*(SORT(.fini_array.*)))
118   KEEP (*(.fini_array*))
119   PROVIDE_HIDDEN (__fini_array_end = .);
120 } >FLASH
121
122 /* used by the startup to initialize data */
123 _sidata = LOADADDR(.data);
```

```
124
125   /* Initialized data sections goes into RAM, load LMA copy after code */
126   .data :
127   {
128     . = ALIGN(4);
129     _sdata = .;        /* create a global symbol at data start */
130     *(.data)           /* .data sections */
131     *(.data*)          /* .data* sections */
132     *(.RamFunc)        /* .RamFunc sections */
133     *(.RamFunc*)       /* .RamFunc* sections */
134
135     . = ALIGN(4);
136     _edata = .;        /* define a global symbol at data end */
137   } >RAM_D1 AT> FLASH
138
139   /* Uninitialized data section */
140   . = ALIGN(4);
141   .bss :
142   {
143     /* This is used by the startup in order to initialize the .bss secion */
144     _sbss = .;         /* define a global symbol at bss start */
145     __bss_start__ = _sbss;
146     *(.bss)
147     *(.bss*)
148     *(COMMON)
149
150     . = ALIGN(4);
151     _ebss = .;         /* define a global symbol at bss end */
152     __bss_end__ = _ebss;
153   } >RAM_D1
154
155   /* User_heap_stack section, used to check that there is enough RAM left */
156   ._user_heap_stack :
157   {
158     . = ALIGN(8);
159     PROVIDE ( end = . );
160     PROVIDE ( _end = . );
161     . = . + _Min_Heap_Size;
162     . = . + _Min_Stack_Size;
163     . = ALIGN(8);
164   } >RAM_D1
165
166 /*****************************************************************************/
167
168     /* Modificación para que funcione lwIP */
169     .lwip_sec (NOLOAD) :
170     {
```

```
171         . = ABSOLUTE(0x30000000);
172         *(.RxDecripSection)
173         . = ABSOLUTE(0x30000060);
174         *(.TxDecripSection)
175         . = ABSOLUTE(0x30000200);
176         *(.RxArraySection)
177     } >RAM_D2 AT> FLASH
178 
179 /*****************************************************************************/
180 
181   /* Remove information from the standard libraries */
182   /DISCARD/ :
183   {
184     libc.a ( * )
185     libm.a ( * )
186     libgcc.a ( * )
187   }
188 
189   .ARM.attributes 0 : { *(.ARM.attributes) }
190 }
```

C

Librerías

C.1. Repositorios

El proceso de añadir las librerías usadas en el proyecto de programación es algo complejo, motivo por el cual se adjunta este anexo, donde se detallan los pasos a seguir.

Al usar un microcontrolador de la familia STM32, existen dos formas de añadir las librerías al proyecto de programación:

- De forma manual, siguiendo el procedimiento general.
- A través del asistente de generación de código STM32CubeMX.

Antes de escoger un método, es conveniente evaluar las ventajas y desventajas de cada uno.

Asistente STM32CubeMX	Proceso manual
+ Se automatiza el proceso de añadir las librerías al proyecto de programación. Desde el asistente se añaden y configuran la librerías.	- Hay que descargar y añadir manualmente las librerías, además de establecer las rutas de inclusión y símbolos de la configuración del proyecto de programación.

- Las librerías provienen del repositorio de STMicroelectronics, donde suelen estar desactualizadas y, a veces, también modificadas respecto a las originales.	+ Las librerías se pueden obtener de sus repositorios oficiales, en sus últimas versiones y sin modificaciones.
- Los directorios donde se guardan las librerías están predefinidos, así como los de sus archivos de configuración e inicialización.	+ El usuario puede elegir dónde guarda las librerías y sus archivos de configuración e inicialización.
- El asistente obliga al usuario a escribir el código en zonas concretas, entre unas etiquetas `/* USER CODE BEGIN N */` y `/* USER CODE END N */`. Si el usuario escribe algo que no esté entre estas etiquetas, será eliminado al regenerar el código. Además, el asistente no permite al usuario crear sus propias zonas de código.	+ Al no haber un asistente, existe total libertad para editar los archivos de código fuente.
- Al usar un asistente, se generan archivos adicionales (no de código fuente) en el proyecto de programación.	+ No se generan archivos adicionales en el proyecto de programación.
- Puede no entenderse lo que se está haciendo, al dejar que el asistente lo haga todo.	+ El usuario, al tener que realizar todo el proceso, tiene control sobre lo que hace.

Tabla V – Comparativa entre dos formas de añadir librerías

Tras evaluar la tabla anterior, se pueden obtener las siguientes conclusiones:

– Si no se tienen conocimientos sobre una librería, es conveniente utilizar el asistente para aprender a configurarla y usarla.

– Una vez se tienen conocimientos sobre las librerías usadas, es mejor crear proyectos de programación sin utilizar el asistente.

– Una opción es generar un proyecto de programación con el asistente y, a partir de ese momento, continuar sin regenerar código con él. Así se eliminan algunas de las desventajas.

– Otra opción es usar dos proyectos de programación paralelos: uno "bueno", sin usar el asistente, y otro con el asistente del que tomar algunos fragmentos de código. Esta es la metodología seguida por el autor de este documento.

Para llevar a cabo el proceso manual de adición de las librerías usadas en el proyecto de programación creado por el autor de este documento, se han de descargar los repositorios que las contienen. Uno de ellos es el de STMicroelectronics para el microcontrolador usado, y este es el único repositorio empleado por el asistente, al contener todas las librerías utilizadas. No obstante, y como se ha explicado antes, estas librerías suelen están desactualizadas y, a veces, también modificadas. Por tanto, lo mejor es obtener las librerías de sus repositorios oficiales, mientras que del repositorio de STMicroelectronics solamente se han de extraer los archivos necesarios que no se encuentren en los otros repositorios.

Repositorio	Versión	Uso en el proyecto de programación
lwIP	2.1.2	Stack de TCP/IP
mbedTLS	2.16.11	Stack de TLS Infraestructura de certificados
FreeRTOS	10.4.4	Sistema operativo de tiempo real
CMSIS	5.8.0	Parte 1 de la capa de abstracción de hardware
STM32CubeH7	1.9.0	Parte 2 de la capa de abstracción de hardware Parte 1 del wrapper sobre FreeRTOS
CMSIS-FreeRTOS	10.3.1	Parte 2 del wrapper sobre FreeRTOS

Tabla VI – Repositorios de librerías

A continuación se detalla paso a paso cómo se añaden las librerías al proyecto de programación, omitiendo la configuración de las rutas de inclusión y los símbolos, ya que esto último se detalla en el siguiente anexo:

– Crear el siguiente directorio:

```
/NUCLEO_H723ZG_MQTT_TLS/Libraries
```

Los siguientes pasos son específicos de cada librería usada y se encuentran en las siguientes secciones, donde se mencionan algunos archivos de configuración de librerías que no se sitúan en la carpeta que contiene todas ellas, y en cambio se encuentran en la carpeta de la aplicación. Lo mismo ocurre con un archivo (no de configuración) usado por lwIP pero no perteneciente a este stack de TCP/IP.

C.2. Instalación

C.2.1. CMSIS

CMSIS es una capa de abstracción de hardware para microcontroladores con cores ARM-Cortex. Consiste en una serie de componentes que son independientes del microcontrolador usado, no obstante, pueden ser necesarios componentes adicionales que sí dependen de él, que han de ser proporcionados por su fabricante. En el proyecto de programación se han hecho uso de los siguientes componentes de CMSIS:

- **CMSIS-Core (Cortex-M)**: API que contiene definiciones para el core y los periféricos del ARM Cortex-M. En este proyecto de programación, en el que se usa un microcontrolador con un core ARM Cortex-M7, se toman únicamente los archivos requeridos por él, algunos del repositorio de CMSIS y otros del repositorio de STMicroelectronics.

- **CMSIS-RTOS v2**: Wrapper para varios sistemas operativos de tiempo real. Para este proyecto de programación es necesario obtener archivos del repositorio de CMSIS y otros específicos de FreeRTOS de otro repositorio.

Para añadir CMSIS al proyecto de programación han de seguirse los siguientes pasos:

- Crear el siguiente directorio:

  `/NUCLEO_H723ZG_MQTT_TLS/Libraries/CMSIS`

- Crear el siguiente directorio:

  `/NUCLEO_H723ZG_MQTT_TLS/Libraries/CMSIS/Core`

- Crear el siguiente directorio:

  `/NUCLEO_H723ZG_MQTT_TLS/Libraries/CMSIS/Core/Device`

- Copiar al directorio anterior los siguientes archivos:

  `/STM32CubeH7-1.9.0.zip/STM32CubeH7-1.9.0/Drivers/CMSIS/Device/ST/STM32H7xx/Include/stm32h7xx.h`

  `/STM32CubeH7-1.9.0.zip/STM32CubeH7-1.9.0/Drivers/CMSIS/Device/ST/STM32H7xx/Include/stm32h723xx.h`

  `/STM32CubeH7-1.9.0.zip/STM32CubeH7-1.9.0/Drivers/CMSIS/Device/ST/STM32H7xx/Include/system_stm32h7xx.h`

  `/STM32CubeH7-1.9.0.zip/STM32CubeH7-1.9.0/Projects/NUCLEO-H723ZG/Templates/Src/system_stm32h7xx.c`

– Crear el siguiente directorio:

`/NUCLEO_H723ZG_MQTT_TLS/Libraries/CMSIS/Core/Include`

– Copiar al directorio anterior los siguientes archivos:

`/ARM.CMSIS.5.8.0.pack/CMSIS/Core/Include/cachel1_armv7.h`
`/ARM.CMSIS.5.8.0.pack/CMSIS/Core/Include/cmsis_compiler.h`
`/ARM.CMSIS.5.8.0.pack/CMSIS/Core/Include/cmsis_gcc.h`
`/ARM.CMSIS.5.8.0.pack/CMSIS/Core/Include/cmsis_version.h`
`/ARM.CMSIS.5.8.0.pack/CMSIS/Core/Include/core_cm7.h`
`/ARM.CMSIS.5.8.0.pack/CMSIS/Core/Include/mpu_armv7.h`

– Crear el siguiente directorio:

`/NUCLEO_H723ZG_MQTT_TLS/Libraries/CMSIS/RTOS2`

– Copiar al directorio anterior los siguientes archivos:

`/ARM.CMSIS.5.8.0.pack/CMSIS/RTOS2/Include/cmsis_os2.h`
`/ARM.CMSIS.5.8.0.pack/CMSIS/RTOS2/Include/os_tick.h`
`/ARM.CMSIS-FreeRTOS.10.3.1.pack/CMSIS/RTOS2/FreeRTOS/Include/freertos_mpool.h`
`/ARM.CMSIS-FreeRTOS.10.3.1.pack/CMSIS/RTOS2/FreeRTOS/Include/freertos_os2.h`
`/ARM.CMSIS-FreeRTOS.10.3.1.pack/CMSIS/RTOS2/FreeRTOS/Include1/cmsis_os.h`
`/ARM.CMSIS-FreeRTOS.10.3.1.pack/CMSIS/RTOS2/FreeRTOS/Source/cmsis_os2.c`

CMSIS no requiere de archivos de configuración de usuario.

C.2.2. FreeRTOS

FreeRTOS es un sistema operativo de tiempo real para microcontroladores. Para añadirlo al proyecto de programación han de seguirse los siguientes pasos:

– Crear el siguiente directorio:

`/NUCLEO_H723ZG_MQTT_TLS/Libraries/FreeRTOS`

– Copiar al directorio anterior el contenido del siguiente directorio:

`/FreeRTOSv202107.00.zip/FreeRTOSv202107.00/FreeRTOS/Source`

- Acceder al siguiente directorio:

  `/NUCLEO_H723ZG_MQTT_TLS/Libraries/FreeRTOS/portable`

- Dentro del directorio anterior, borrar todo excepto los siguientes directorios:

  `/NUCLEO_H723ZG_MQTT_TLS/Libraries/FreeRTOS/portable/GCC`
  `/NUCLEO_H723ZG_MQTT_TLS/Libraries/FreeRTOS/portable/MemMang`

- Acceder al siguiente directorio:

  `/NUCLEO_H723ZG_MQTT_TLS/Libraries/FreeRTOS/portable/GCC`

- Dentro del directorio anterior, borrar todo excepto el siguiente directorio:

  `/NUCLEO_H723ZG_MQTT_TLS/Libraries/FreeRTOS/portable/GCC/ARM_CM7`

- Acceder al siguiente directorio:

  `/NUCLEO_H723ZG_MQTT_TLS/Libraries/FreeRTOS/portable/MemMang`

- Dentro del directorio anterior, borrar todo excepto el siguiente archivo:

  `/NUCLEO_H723ZG_MQTT_TLS/Libraries/FreeRTOS/portable/MemMang/heap_4.c`

FreeRTOS requiere del archivo de configuración de usuario **FreeRTOSConfig.h**.

C.2.3. lwIP

lwIP es un stack de TCP/IP de bajos requisitos de hardware. Dispone de tres APIs:

- **Raw/native API**: La de más bajo nivel, propia de lwIP. Es la única API disponible cuando se cuenta con una única tarea en el programa (es decir, sin sistema operativo, bare-metal). Es la API que menos recursos consume.

- **Netconn/sequential API**: Funciona sobre la Raw API y también es propia de lwIP. Requiere de un sistema multitarea (normalmente con un sistema operativo). Consume más recursos que la Raw API.

- **BSD-style socket API**: Opera sobre la Netconn API y corresponde a la API de sockets BSD. También requiere de un sistema multitarea. Consume más recursos que la Netconn/Sequential API.

De las tres APIs anteriores, en el proyecto de programación se ha hecho uso de la última, la que emplea la API de sockets BSD, debido a las siguientes razones:

- La API de sockets BSD está muy extendida, lo que permite que el código sea portable a otros sistemas que hagan uso de otros stacks de TCP/IP.

- Aunque la API de sockets BSD sea la que más recursos consume, eso no es un problema en microcontroladores modernos como los STM32.

- La Raw API tiene la ventaja de no requerir un sistema multitarea, pero los programas que hacen uso de ella no son portables a otros stacks de TCP/IP.

Este stack de TCP/IP, al interactuar directamente con el hardware y con otros componentes de software, requiere algunos archivos adicionales que han de ser proporcionados por el usuario:

- Una interfaz de red entre lwIP y el PHY de Ethernet, en este caso un LAN8742. Esta interfaz de red se encuentra en el fichero **ethernetif.c**, que se encuentra en la carpeta de la aplicación.

- Una capa de abstracción que haga de interfaz entre lwIP y el sistema operativo empleado (si se usa). El stack de TCP/IP hace uso de unas funciones del sistema operativo que han de ser implementadas por el usuario, normalmente en un archivo **sys_arch.c**, que en este proyecto de programación se ha obtenido del repositorio de STMicroelectronics y copiado a la carpeta que contiene las librerías. Los prototipos de estas funciones se encuentran en el archivo **sys.h**, incluido en lwIP.

- Unas cabeceras con definiciones específicas de la arquitectura de hardware empleada. Se encuentran en la carpeta de lwIP.

Para añadir lwIP al proyecto de programación han de seguirse los siguientes pasos:

- Crear el siguiente directorio:

  `/NUCLEO_H723ZG_MQTT_TLS/Libraries/lwIP`

- Copiar al directorio anterior los siguientes directorios:

  `/lwip-2.1.2.zip/lwip-2.1.2/src`
  `/STM32CubeH7-1.9.0.zip/STM32CubeH7-1.9.0/Middlewares/Third_Party/LwIP/system`

– Borrar el siguiente directorio:

  `/NUCLEO_H723ZG_MQTT_TLS/Libraries/lwIP/src/apps`

lwIP requiere del archivo de configuración de usuario **lwipopts.h**, en el que han de definirse las configuraciones diferentes a las predeterminadas, situadas en el fichero de la librería **opt.h**.

C.2.4. mbedTLS

mbedTLS es un kit de herramientas criptográficas que incluye programas que permiten crear infraestructuras de certificados digitales de clave pública y un stack de TLS. Los programas son usados en un PC, mientras que el stack de TLS es utilizado por el proyecto de programación, que para añadírselo han de seguirse los siguientes pasos:

– Crear el siguiente directorio:

  `/NUCLEO_H723ZG_MQTT_TLS/Libraries/mbedTLS`

– Copiar al directorio anterior los siguientes directorios:

  `/mbedtls-mbedtls-2.16.11.zip/mbedtls-mbedtls-2.16.11/library`
  `/mbedtls-mbedtls-2.16.11.zip/mbedtls-mbedtls-2.16.11/include`

mbedTLS cuenta con el fichero de configuración predeterminado **config.h**, que es válido para que el stack de TLS funcione en un PC (Windows / UNIX like OS), pero no en otros sistemas como el utilizado para el proyecto de programación. Por tanto, es necesario usar un archivo personalizado de configuración de usuario **mbedTLS_config_file.h** cuyo nombre es definido en un símbolo de la configuración del proyecto de programación.

C.2.5. STM32H7

STM32H7 es la familia del microcontrolador usado, para la cual existen drivers proporcionados por STMicroelectronics para configurar y usar sus periféricos. Para añadir estos drivers al proyecto de programación han de seguirse los siguientes pasos:

– Crear el siguiente directorio:

  `/NUCLEO_H723ZG_MQTT_TLS/Libraries/STM32H7`

– Crear el siguiente directorio:

`/NUCLEO_H723ZG_MQTT_TLS/Libraries/STM32H7/HAL`

– Crear el siguiente directorio:

`/NUCLEO_H723ZG_MQTT_TLS/Libraries/STM32H7/HAL/Inc`

– Copiar al directorio anterior los siguientes archivos:

`/STM32CubeH7-1.9.0.zip/STM32CubeH7-1.9.0/Drivers/STM32H7xx_HAL_Driver/Inc/stm32h7xx_hal.h`

`/STM32CubeH7-1.9.0.zip/STM32CubeH7-1.9.0/Drivers/STM32H7xx_HAL_Driver/Inc/stm32h7xx_hal_cortex.h`

`/STM32CubeH7-1.9.0.zip/STM32CubeH7-1.9.0/Drivers/STM32H7xx_HAL_Driver/Inc/stm32h7xx_hal_def.h`

`/STM32CubeH7-1.9.0.zip/STM32CubeH7-1.9.0/Drivers/STM32H7xx_HAL_Driver/Inc/stm32h7xx_hal_dma.h`

`/STM32CubeH7-1.9.0.zip/STM32CubeH7-1.9.0/Drivers/STM32H7xx_HAL_Driver/Inc/stm32h7xx_hal_dma_ex.h`

`/STM32CubeH7-1.9.0.zip/STM32CubeH7-1.9.0/Drivers/STM32H7xx_HAL_Driver/Inc/stm32h7xx_hal_eth.h`

`/STM32CubeH7-1.9.0.zip/STM32CubeH7-1.9.0/Drivers/STM32H7xx_HAL_Driver/Inc/stm32h7xx_hal_eth_ex.h`

`/STM32CubeH7-1.9.0.zip/STM32CubeH7-1.9.0/Drivers/STM32H7xx_HAL_Driver/Inc/stm32h7xx_hal_flash.h`

`/STM32CubeH7-1.9.0.zip/STM32CubeH7-1.9.0/Drivers/STM32H7xx_HAL_Driver/Inc/stm32h7xx_hal_flash_ex.h`

`/STM32CubeH7-1.9.0.zip/STM32CubeH7-1.9.0/Drivers/STM32H7xx_HAL_Driver/Inc/stm32h7xx_hal_gpio.h`

`/STM32CubeH7-1.9.0.zip/STM32CubeH7-1.9.0/Drivers/STM32H7xx_HAL_Driver/Inc/stm32h7xx_hal_gpio_ex.h`

`/STM32CubeH7-1.9.0.zip/STM32CubeH7-1.9.0/Drivers/STM32H7xx_HAL_Driver/Inc/stm32h7xx_hal_pwr.h`

`/STM32CubeH7-1.9.0.zip/STM32CubeH7-1.9.0/Drivers/STM32H7xx_HAL_Driver/Inc/stm32h7xx_hal_pwr_ex.h`

`/STM32CubeH7-1.9.0.zip/STM32CubeH7-1.9.0/Drivers/STM32H7xx_HAL_Driver/Inc/stm32h7xx_hal_rcc.h`

`/STM32CubeH7-1.9.0.zip/STM32CubeH7-1.9.0/Drivers/STM32H7xx_HAL_Driver/Inc/stm32h7xx_hal_rcc_ex.h`

`/STM32CubeH7-1.9.0.zip/STM32CubeH7-1.9.0/Drivers/STM32H7xx_HAL_Driver/Inc/stm32h7xx_hal_rng.h`

`/STM32CubeH7-1.9.0.zip/STM32CubeH7-1.9.0/Drivers/STM32H7xx_HAL_Driver/Inc/stm32h7xx_hal_rng_ex.h`

`/STM32CubeH7-1.9.0.zip/STM32CubeH7-1.9.0/Drivers/STM32H7xx_HAL_Driver/Inc/stm32h7xx_hal_rtc.h`

`/STM32CubeH7-1.9.0.zip/STM32CubeH7-1.9.0/Drivers/STM32H7xx_HAL_Driver/Inc/stm32h7xx_hal_rtc_ex.h`

`/STM32CubeH7-1.9.0.zip/STM32CubeH7-1.9.0/Drivers/STM32H7xx_HAL_Driver/Inc/stm32h7xx_hal_tim.h`

`/STM32CubeH7-1.9.0.zip/STM32CubeH7-1.9.0/Drivers/STM32H7xx_HAL_Driver/Inc/stm32h7xx_hal_tim_ex.h`

`/STM32CubeH7-1.9.0.zip/STM32CubeH7-1.9.0/Drivers/STM32H7xx_HAL_Driver/Inc/stm32h7xx_hal_uart.h`

`/STM32CubeH7-1.9.0.zip/STM32CubeH7-1.9.0/Drivers/STM32H7xx_HAL_Driver/Inc/stm32h7xx_hal_uart_ex.h`

– Crear el siguiente directorio:

`/NUCLEO_H723ZG_MQTT_TLS/Libraries/STM32H7/HAL/Inc/Legacy`

– Copiar al directorio anterior el siguiente archivo:

`/STM32CubeH7-1.9.0.zip/STM32CubeH7-1.9.0/Drivers/STM32H7xx_HAL_Driver/Inc/Legacy/stm32_hal_legacy.h`

– Crear el siguiente directorio:

`/NUCLEO_H723ZG_MQTT_TLS/Libraries/STM32H7/HAL/Src`

– Copiar al directorio anterior los siguientes archivos:

`/STM32CubeH7-1.9.0.zip/STM32CubeH7-1.9.0/Drivers/STM32H7xx_HAL_Driver/Src/stm32h7xx_hal.c`

`/STM32CubeH7-1.9.0.zip/STM32CubeH7-1.9.0/Drivers/STM32H7xx_HAL_Driver/Src/stm32h7xx_hal_cortex.c`

`/STM32CubeH7-1.9.0.zip/STM32CubeH7-1.9.0/Drivers/STM32H7xx_HAL_Driver/Src/stm32h7xx_hal_dma.c`

`/STM32CubeH7-1.9.0.zip/STM32CubeH7-1.9.0/Drivers/STM32H7xx_HAL_Driver/Src/stm32h7xx_hal_dma_ex.c`

`/STM32CubeH7-1.9.0.zip/STM32CubeH7-1.9.0/Drivers/STM32H7xx_HAL_Driver/Src/stm32h7xx_hal_eth.c`

`/STM32CubeH7-1.9.0.zip/STM32CubeH7-1.9.0/Drivers/STM32H7xx_HAL_Driver/Src/stm32h7xx_hal_eth_ex.c`

`/STM32CubeH7-1.9.0.zip/STM32CubeH7-1.9.0/Drivers/STM32H7xx_HAL_Driver/Src/stm32h7xx_hal_flash.c`

`/STM32CubeH7-1.9.0.zip/STM32CubeH7-1.9.0/Drivers/STM32H7xx_HAL_Driver/Src/stm32h7xx_hal_flash_ex.c`

`/STM32CubeH7-1.9.0.zip/STM32CubeH7-1.9.0/Drivers/STM32H7xx_HAL_Driver/Src/stm32h7xx_hal_gpio.c`

`/STM32CubeH7-1.9.0.zip/STM32CubeH7-1.9.0/Drivers/STM32H7xx_HAL_Driver/Src/stm32h7xx_hal_pwr.c`

`/STM32CubeH7-1.9.0.zip/STM32CubeH7-1.9.0/Drivers/STM32H7xx_HAL_Driver/Src/stm32h7xx_hal_pwr_ex.c`

`/STM32CubeH7-1.9.0.zip/STM32CubeH7-1.9.0/Drivers/STM32H7xx_HAL_Driver/Src/stm32h7xx_hal_rcc.c`

`/STM32CubeH7-1.9.0.zip/STM32CubeH7-1.9.0/Drivers/STM32H7xx_HAL_Driver/Src/stm32h7xx_hal_rcc_ex.c`

`/STM32CubeH7-1.9.0.zip/STM32CubeH7-1.9.0/Drivers/STM32H7xx_HAL_Driver/Src/stm32h7xx_hal_rng.c`

`/STM32CubeH7-1.9.0.zip/STM32CubeH7-1.9.0/Drivers/STM32H7xx_HAL_Driver/Src/stm32h7xx_hal_rng_ex.c`

`/STM32CubeH7-1.9.0.zip/STM32CubeH7-1.9.0/Drivers/STM32H7xx_HAL_Driver/Src/stm32h7xx_hal_rtc.c`

`/STM32CubeH7-1.9.0.zip/STM32CubeH7-1.9.0/Drivers/STM32H7xx_HAL_Driver/Src/stm32h7xx_hal_rtc_ex.c`

`/STM32CubeH7-1.9.0.zip/STM32CubeH7-1.9.0/Drivers/STM32H7xx_HAL_Driver/Src/stm32h7xx_hal_tim.c`

`/STM32CubeH7-1.9.0.zip/STM32CubeH7-1.9.0/Drivers/STM32H7xx_HAL_Driver/Src/stm32h7xx_hal_tim_ex.c`

`/STM32CubeH7-1.9.0.zip/STM32CubeH7-1.9.0/Drivers/STM32H7xx_HAL_Driver/Src/stm32h7xx_hal_uart.c`

`/STM32CubeH7-1.9.0.zip/STM32CubeH7-1.9.0/Drivers/STM32H7xx_HAL_Driver/Src/stm32h7xx_hal_uart_ex.c`

– Crear el siguiente directorio:

`/NUCLEO_H723ZG_MQTT_TLS/Libraries/STM32H7/LAN8742`

– Copiar al directorio anterior los siguientes archivos:

`/STM32CubeH7-1.9.0.zip/STM32CubeH7-1.9.0/Drivers/BSP/Components/lan8742/lan8742.c`

`/STM32CubeH7-1.9.0.zip/STM32CubeH7-1.9.0/Drivers/BSP/Components/lan8742/lan8742.h`

Los drivers del STM32H7 requieren del archivo de configuración de usuario **stm32h7xx_hal_conf.h**.

D

Desarrollo

D.1. Proyecto del STM32

Para crear el proyecto de programación ha de abrirse STM32CubeIDE y realizar los siguientes pasos:

`File > New > STM32 Project`

Se escoge el microcontrolador STM32H723ZG o la placa NUCLEO-H723ZG, y se elige el lenguaje C, binario ejecutable y tipo de proyecto que no use STM32CubeMX.

D.2. Rutas de inclusión

Dentro de STM32CubeIDE, las rutas de inclusión han de añadirse en la siguiente ubicación:

`Project > Properties > C/C++ General > Paths and Symbols > Includes > GNU C`

A continuación, han de añadirse los siguientes directorios como pertenecientes al espacio de trabajo:

```
/NUCLEO_H723ZG_MQTT_TLS/Application/include
/NUCLEO_H723ZG_MQTT_TLS/Application/include/cfg
/NUCLEO_H723ZG_MQTT_TLS/Application/include/hw
/NUCLEO_H723ZG_MQTT_TLS/Application/include/sw
```

/NUCLEO_H723ZG_MQTT_TLS/Libraries/CMSIS/Core/Device
/NUCLEO_H723ZG_MQTT_TLS/Libraries/CMSIS/Core/Include
/NUCLEO_H723ZG_MQTT_TLS/Libraries/CMSIS/RTOS2
/NUCLEO_H723ZG_MQTT_TLS/Libraries/FreeRTOS/include
/NUCLEO_H723ZG_MQTT_TLS/Libraries/FreeRTOS/portable/GCC/ARM_CM7/r0p1
/NUCLEO_H723ZG_MQTT_TLS/Libraries/lwIP/src/include
/NUCLEO_H723ZG_MQTT_TLS/Libraries/lwIP/system
/NUCLEO_H723ZG_MQTT_TLS/Libraries/mbedTLS/include
/NUCLEO_H723ZG_MQTT_TLS/Libraries/STM32H7/HAL/Inc
/NUCLEO_H723ZG_MQTT_TLS/Libraries/STM32H7/LAN8742

D.3. Símbolos

Los símbolos se pueden añadir dentro de STM32CubeIDE, en la siguiente ubicación:

`Project > Properties > C/C++ General > Paths and Symbols > Symbols > GNU C`

En el proyecto de programación se han usado los siguientes símbolos con los siguientes valores:

Symbol	Value
CERTIFICATES_FILE	"certificates.h"
HAL_DRIVER	"stm32h7xx_hal.h"
MBEDTLS_CONFIG_FILE	"mbedTLS_config_file.h"
MQTT_CLIENT_CONFIG_FILE	"setup.h"
RNG_IMPLEMENTATION	"rng.h"
RTC_IMPLEMENTATION	"rtc.h"
SOCKETS_IMPLEMENTATION	"lwip/sockets.h"
STM32_DEVICE	"stm32h723xx.h"
STM32_PLATFORM	"NUCLEO-H723ZG"
STM32H723xx	
TIM_IMPLEMENTATION	"tim.h"
TLS_CLIENT_CONFIG_FILE	"setup.h"
USE_HAL_DRIVER	

Tabla VII – Símbolos del proyecto de programación

E

Portabilidad

El proyecto ha sido programado con el objetivo de que pueda ser portado a otros sistemas sin tener que hacer demasiadas modificaciones. Por ello, a modo de ejemplo, se proporcionan algunas directrices para crear un port para la placa de desarrollo **DISCO-F769NI**, que cuenta con un microcontrolador **STM32F769NIH6**, cuyo core es un **ARM Cortex-M7**.

Para crear el port se ha de obtener el repositorio STM32CubeF7 en su versión 1.16.1, del que se han de extraer los archivos de librerías que reemplazan a algunos de los existentes en el proyecto de programación original.

NUCLEO-H723ZG	DISCO-F769NI
stm32h7xx.h	stm32f7xx.h
stm32h723xx.h	stm32f769xx.h
system_stm32h7xx.h	system_stm32f7xx.h
system_stm32h7xx.c	system_stm32f7xx.c
stm32h7xx_hal.h	stm32f7xx_hal.h
stm32h7xx_hal_cortex.h	stm32f7xx_hal_cortex.h
stm32h7xx_hal_def.h	stm32f7xx_hal_def.h
stm32h7xx_hal_dma.h	stm32f7xx_hal_dma.h
stm32h7xx_hal_dma_ex.h	stm32f7xx_hal_dma_ex.h
stm32h7xx_hal_eth.h	stm32f7xx_hal_eth.h
stm32h7xx_hal_eth_ex.h	
stm32h7xx_hal_flash.h	stm32f7xx_hal_flash.h
stm32h7xx_hal_flash_ex.h	stm32f7xx_hal_flash_ex.h
stm32h7xx_hal_gpio.h	stm32f7xx_hal_gpio.h
stm32h7xx_hal_gpio_ex.h	stm32f7xx_hal_gpio_ex.h

stm32h7xx_hal_pwr.h	stm32f7xx_hal_pwr.h
stm32h7xx_hal_pwr_ex.h	stm32f7xx_hal_pwr_ex.h
stm32h7xx_hal_rcc.h	stm32f7xx_hal_rcc.h
stm32h7xx_hal_rcc_ex.h	stm32f7xx_hal_rcc_ex.h
stm32h7xx_hal_rng.h	stm32f7xx_hal_rng.h
stm32h7xx_hal_rng_ex.h	
stm32h7xx_hal_rtc.h	stm32f7xx_hal_rtc.h
stm32h7xx_hal_rtc_ex.h	stm32f7xx_hal_rtc_ex.h
stm32h7xx_hal_tim.h	stm32f7xx_hal_tim.h
stm32h7xx_hal_tim_ex.h	stm32f7xx_hal_tim_ex.h
stm32h7xx_hal_uart.h	stm32f7xx_hal_uart.h
stm32h7xx_hal_uart_ex.h	stm32f7xx_hal_uart_ex.h
stm32h7xx_hal.c	stm32f7xx_hal.c
stm32h7xx_hal_cortex.c	stm32f7xx_hal_cortex.c
stm32h7xx_hal_dma.c	stm32f7xx_hal_dma.c
stm32h7xx_hal_dma_ex.c	stm32f7xx_hal_dma_ex.c
stm32h7xx_hal_eth.c	stm32f7xx_hal_eth.c
stm32h7xx_hal_eth_ex.c	
stm32h7xx_hal_flash.c	stm32f7xx_hal_flash.c
stm32h7xx_hal_flash_ex.c	stm32f7xx_hal_flash_ex.c
stm32h7xx_hal_gpio.c	stm32f7xx_hal_gpio.c
stm32h7xx_hal_pwr.c	stm32f7xx_hal_pwr.c
stm32h7xx_hal_pwr_ex.c	stm32f7xx_hal_pwr_ex.c
stm32h7xx_hal_rcc.c	stm32f7xx_hal_rcc.c
stm32h7xx_hal_rcc_ex.c	stm32f7xx_hal_rcc_ex.c
stm32h7xx_hal_rng.c	stm32f7xx_hal_rng.c
stm32h7xx_hal_rng_ex.c	
stm32h7xx_hal_rtc.c	stm32f7xx_hal_rtc.c
stm32h7xx_hal_rtc_ex.c	stm32f7xx_hal_rtc_ex.c
stm32h7xx_hal_tim.c	stm32f7xx_hal_tim.c
stm32h7xx_hal_tim_ex.c	stm32f7xx_hal_tim_ex.c
stm32h7xx_hal_uart.c	stm32f7xx_hal_uart.c
stm32h7xx_hal_uart_ex.c	stm32f7xx_hal_uart_ex.c
lan8742.c lan8742.h	

Tabla VIII – Reemplazo de archivos de librerías

Los archivos de librerías que no aparecen en la tabla anterior no cambian entre el proyecto de programación original y el port.

Luego se han de hacer cambios en la configuración del proyecto de programación.

NUCLEO-H723ZG		DISCO-F769NI	
Symbol	Value	Symbol	Value
HAL_DRIVER	"stm32h7xx_hal.h"	HAL_DRIVER	"stm32f7xx_hal.h"
STM32_DEVICE	"stm32h723xx.h"	STM32_DEVICE	"stm32f769xx.h"
STM32_PLATFORM	"NUCLEO-H723ZG"	STM32_PLATFORM	"DISCO-F769NI"
STM32H723xx		STM32F769xx	

Tabla IX – Cambios en la configuración del proyecto de programación

Además, es conveniente modificar los nombres de algunos directorios, sin olvidar modificar también sus rutas de inclusión.

NUCLEO-H723ZG	DISCO-F769NI
/NUCLEO_H723ZG_MQTT_TLS	/DISCO_F769NI_MQTT_TLS
/../../STM32H7	/../../STM32F7
/../../../LAN8742	

Tabla X – Renombrado de directorios

A continuación se han de sustituir algunos archivos de usuario. De la siguiente tabla, el primero de ellos requiere de modificaciones por parte del usuario, mientras que los otros dos son generados automáticamente por STM32CubeIDE.

NUCLEO-H723ZG	DISCO-F769NI
stm32h7xx_hal_conf.h	stm32f7xx_hal_conf.h
startup_stm32h723zgtx.s	startup_stm32f769nihx.s
STM32H723ZGTX_FLASH.ld	STM32F769NIHX_FLASH.ld

Tabla XI – Reemplazo de archivos de usuario

Por último, se han de realizar modificaciones en algunos archivos de usuario.

Archivo	Cambios
ethernetif.c	Gran parte del archivo
gpio.c	Posición del pulsador y de los LEDs
rcc.c	Gran parte del archivo
rng.c	Estructura de inicialización del RNG
tim.c	Frecuencia que le llega a los temporizadores
uart.c	Estructura de inicialización de la UART y posición de sus pines

Tabla XII – Modificaciones de los archivos de los periféricos

Un nuevo proyecto de programación que introduzca los cambios anteriores debería funcionar en esta placa de desarrollo, no obstante, para aprovechar el potencial de esta se puede hacer uso del display LCD que lleva incorporado para mostrar en ella el mismo log que es enviado por el puerto serie. Para lograr esto es conveniente hacer uso de algún stack gráfico, por ejemplo, el que acompaña a los drivers HAL en el repositorio de STMicroelectronics, añadiendo algunas modificaciones (por ejemplo, para interpretar las vocales con tilde como si no la tuvieran, ya que no se pueden representar con este stack gráfico).

Figura XXVI – Port para la placa de desarrollo DISCO-F769NI

Ha de tenerse en cuenta que en este ejemplo no son necesarios muchos cambios porque se está usando un microcontrolador parecido. Si el microcontrolador contase con otro core, aunque también fuese un ARM Cortex-M, también habría que cambiar algunos archivos del CMSIS y de FreeRTOS. Si se utilizase uno de otro fabricante, toda la parte de los periféricos cambiaría completamente, y no se podría hacer uso de drivers HAL, al ser específicos de los microcontroladores STM32.

Este proyecto de programación también puede ser portado para PC, aunque esto requeriría de cambios mayores. Por ejemplo, FreeRTOS sería sustituido por un sistema operativo Windows o de tipo UNIX, que incluye middlewares que implementan las funciones de TCP/IP y del cliente DHCP, estableciendo la configuración de red automáticamente al iniciar el sistema operativo. En cuanto a las entradas y salidas, los pulsadores y LEDs serían sustituidos por teclas y elementos en la pantalla. También ha de tenerse en cuenta que la programación del hardware cambiaría completamente, probablemente simplificándola mucho. En cualquier caso, la creación de un port para PC no está dentro del alcance del trabajo expuesto en este documento.

BIBLIOGRAFÍA

Protocolos

[1] OASIS MQTT 5 Specification
https://mqtt.org/mqtt-specification/

[2] IETF RFC 1034 -- Domain names - concepts and facilities
https://datatracker.ietf.org/doc/rfc1034/

[3] IETF RFC 1035 -- Domain names - implementation and specification
https://datatracker.ietf.org/doc/rfc1035/

[4] IETF RFC 2131 -- Dynamic Host Configuration Protocol
https://datatracker.ietf.org/doc/rfc2131/

[5] IETF RFC 5246 -- The Transport Layer Security (TLS) Protocol Version 1.2
https://datatracker.ietf.org/doc/rfc5246/

[6] IETF RFC 5280 -- Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile
https://datatracker.ietf.org/doc/rfc5280/

[7] IETF RFC 5905 -- Network Time Protocol Version 4: Protocol and Algorithms Specification
https://datatracker.ietf.org/doc/rfc5905/

[8] IETF RFC 8446 -- The Transport Layer Security (TLS) Protocol Version 1.3
https://datatracker.ietf.org/doc/rfc8446/

Plataformas

[9] NUCLEO-H723ZG
https://www.st.com/en/evaluation-tools/nucleo-h723zg.html

[10] DISCO-F769NI
https://www.st.com/en/evaluation-tools/32f769idiscovery.html

Programas

[11] TFTPD64 - Servidores DHCP y SNTP, entre otros
https://pjo2.github.io/tftpd64/

[12] SANS - Servidor DNS
http://posadis.sourceforge.net/sans

[13] Eclipse Mosquitto - Broker y cliente MQTT + TLS
https://mosquitto.org/

[14] Wireshark - Analizador de red
https://www.wireshark.org/

[15] PuTTY - Terminal de comunicación serial y otras funcionalidades
https://www.putty.org/

[16] STM32CubeIDE - Entorno de desarrollo integrado para STM32
https://www.st.com/en/development-tools/stm32cubeide.html

Repositorios

[17] lwIP
https://download.savannah.nongnu.org/releases/lwip/

[18] mbedTLS
https://github.com/ARMmbed/mbedtls/tags

[19] FreeRTOS
https://github.com/FreeRTOS/FreeRTOS/tags

[20] CMSIS Version 5
https://github.com/ARM-software/CMSIS_5/tags

[21] STM32CubeH7 MCU Firmware Package
https://github.com/STMicroelectronics/STM32CubeH7/tags

[22] CMSIS-FreeRTOS
https://github.com/ARM-software/CMSIS-FreeRTOS/tags

[23] STM32CubeF7 MCU Firmware Package
https://github.com/STMicroelectronics/STM32CubeF7/tags