Facultad de Ciencias

MARCO PARA EL DESARROLLO DE APLICACIONES ADA SOBRE MICROCONTROLADORES STM32

(FRAMEWORK FOR THE DEVELOPMENT OF ADA APPLICATIONS ON MICROCONTROLLERS STM32)

Trabajo de Fin de Grado

para acceder al

GRADO EN INGENIERÍA INFORMÁTICA

Autor: Daniel Arranz Ortega

Director: Mario Aldea Rivas

Codirector: Héctor Pérez Tijero

Santander, septiembre 2019

RESUMEN

El presente Trabajo de Fin de Grado realiza un estudio sobre los microcontroladores de la familia STM32FX de la marca STMicroelectronics. Este estudio se centra en investigar sobre los distintos drivers que puedan existir en la actualidad y asentar unas bases para el desarrollo y la depuración de programas en el lenguaje Ada bajo el sistema operativo Linux.

Inicialmente el proyecto comenzó con el desarrollo de pequeños programas de pruebas sobre la placa STM32F407 debido a sus funcionalidades más simples. Posteriormente centramos nuestros objetivos en la placa STM32F769. Esta última tiene más potencia y entre sus características cuenta con una pantalla LCD táctil de 4 pulgadas y conexiones rápidas compatibles con las del Arduino Uno. Esta combinación hace que dicho microcontrolador sea una buena alternativa en proyectos de robótica u otros proyectos llevados a cabo con tarjetas de la familia Arduino o incluso Raspberry Pi. Por todo ello, se propone su incorporación en los laboratorios de algunas asignaturas del Grado en Ingeniería Informática relacionadas con proyectos software/hardware en tiempo real.

Para culminar el trabajo se ha desarrollado un proyecto de coche dirigido desde otra placa STM32F769 en el que se integra todo el bagaje aprendido sobre la familia de microcontroladores STM32: comunicación con otras placas, entradas y salida digital, I^2C, entrada analógica o el uso de la pantalla táctil, entre otros.

Palabras clave: microcontrolador STM32, robótica, hardware/software, Ada.

ABSTRACT

The present Final Bachelor's Dissertation is based on a study concerning the STM32Fx family microcontrollers launched by STMicroelecctronics. This study focuses on investigating the different drivers that may exist today and laying a foundation for the development and debugging of programs in the Ada language under the Linux operating system.

Initially the project began with the development of small test programs on the STM32F407 eval board due to its simpler functionalities. Then we focused our objectives on the STM32F769 board. The latter has more power and among its features has a 4-inch LCD touch screen and quick connections compatible with those of the Arduino Uno. Thus, this combination makes this microcontroller a good candidate in robotics projects or other projects carried out with boards from the Arduino family or even Raspberry Pi. Therefore, its incorporation is proposed into the laboratories of some subjects related to software / hardware projects in real time within our B.Sc. in Computer Engineering.

In order to complete this project, a car project directed from another STM32F769 board has been developed in which all the baggage learnt about the STM32 family of microcontrollers is integrated on it; i.e. cross-compilation, communication with other boards, digital inputs and outputs, I2C, analogue input or the use of the touch screen.

Keywords: microcontroller, STM32 eval board, robotics, hardware / software, Ada.

ÍNDICE

ÍNDICE DE FIGURAS

PRESENTACIÓN

El presente documento titulado *Marco para el Desarrollo de Aplicaciones ADA sobre Microcontroladores STM32*, corresponde a la asignatura Trabajo Fin de Grado con código G692, de carácter obligatorio. Además, supone la finalización de los estudios de Grado en Ingeniería Informática. Ha sido realizado por D. Daniel Arranz Ortega y tutelado por D. Mario Aldea Rivas y D. Héctor Pérez Tijero, ambos profesores del departamento de Ingeniería Informática y Electrónica en el área de Lenguajes y Sistemas Informáticos.

La realización de este Grado pone a disposición del alumnado la adquisición de competencias generales y especificas desarrolladas en las treinta y ocho asignaturas que componen su plan de estudios. Pienso que la gratitud en silencio no sirve a nadie. Por ello, es a todos los profesores de ese contexto formativo universitario, en concreto a los profesores que han formado parte de este grado, a quienes me gustaría mostrar un gran agradecimiento por sus enseñanzas e implicación en mi proceso formativo. Especialmente, me gustaría dar las gracias a D. Héctor Pérez Tijero y a D. Mario Aldea Rivas, por sus consejos tan útiles en la elaboración de este Trabajo. Asimismo, agradezco a la Universidad de Cantabria por darme la oportunidad de realizar este trabajo en el Departamento de Ingeniería Software y Tiempo Real de la Facultad de Ciencias que me ha dado a conocer el gran camino en la investigación.

Somos lo que somos por quienes nos han acompañado y nos acompañan en el camino. Por ello, mi agradecimiento más sentido va para mi familia. Especialmente, a mi abuela Sofía, por habernos dedicado todo su tiempo y un cariño incalculable. A mis padres y a mi hermana, por haberme servido de ejemplo de esfuerzo e inspiración. Y a mis amigos, por apoyarme en todo momento.

Antes de comenzar con el grueso de este TFG me gustaría resaltar una cita de una de las celebridades en nuestro ámbito, John Von Neumann, que plasmó a mitad del siglo la realidad tecnológica de nuestros días.

"Podría parecer que hemos llegado a los límites alcanzables por la tecnología informática, aunque uno debe ser prudente con estas afirmaciones, pues tienden a sonar bastante tontas en cinco años".

John Von Neumann (1949)

1. INTRODUCCIÓN

El uso de los microcontroladores desde su descubrimiento en 1971 hasta nuestros días se ha visto incrementado exponencialmente debido a las múltiples ventajas que presenta su aplicación. No solo es el caso de aparatos electrónicos pequeños sino de infinidad de productos donde su presencia incrementa la calidad del producto en gran medida, por ejemplo, en automóviles. Su popularidad se debe a su adaptabilidad en programación, capacidad, tamaño y, sobre todo, a un consumo muy reducido.

Los microprocesadores – por su naturaleza tecnológica-informática – están en continua evolución y renovación, ya que el mercado es cada vez más competitivo y novedoso con los productos que se lanzan cada año. En la actualidad podemos encontrar gran variedad de microcontroladores de placa única (*Single Board Computer* / SBC), de los cuales destacan en popularidad aquellos de la familia Arduino y Raspberry Pi. Éstos ofrecen multitud de opciones con las que poderlos configurar, en función del proyecto al que lo queramos dedicar.

A lo largo del Grado de Ingeniería Informática hemos tenido la posibilidad de entrar en contacto con algunos de estos microcontroladores. Por ejemplo, *Raspberry Pi 3* en las asignaturas "Introducción a los computadores" o "Estructura de Computadores", así como en distintos talleres o el *Hackton HackToProgress*. Sin embargo, en las asignaturas del Grado relacionadas con los sistemas de tiempo real se utilizan computadores empotrados industriales con circuitería externa, cuyo coste es alto y no está al alcance de los alumnos. Por otro lado, el software actualmente disponible para estos equipos permite la codificación de aplicaciones en lenguaje Ada - uno de los lenguajes más habituales en sistemas de tiempo real-. Dentro de los microcontroladores que proporcionan soporte inicial para desarrollar en lenguaje Ada se encuentran los microcontroladores de la empresa *STMicroelectronics* que dispone de un amplio catálogo de tarjetas. De hecho, la investigación de dos modelos de las placas STM32 y su posterior utilización son los pilares de este Trabajo de Fin de Grado. Hablamos de los microcontroladores STM32F407 y STM32F769, seleccionados por su adaptabilidad a diversos proyectos y sus procesadores ARM-Córtex de 32 bits, que permiten no solo multitud de funcionalidades en aplicaciones embebidas de tiempo real, sino también la conectividad con multitud de periféricos externos.

Este Trabajo de Fin de Grado presenta un proyecto en el que se ha desarrollado un software que facilita la ejecución de aplicaciones Ada en los microcontroladores STM32 anteriormente mencionados. Por tanto, este proyecto establece las bases para en un futuro poder integrar este tipo de microcontroladores en el laboratorio docente de sistemas de tiempo real.

Dadas estas nuevas posibilidades, y gracias a las constantes y versátiles renovaciones tecnológicas de las que nuestro ámbito de estudio es pionero y partícipe, creemos en la necesidad de incorporar estas nuevas tecnologías en las partes prácticas de los estudios de Grado. Mediante el uso de los microcontroladores SBC no sólo se llevaría a cabo una mejora en la calidad de la enseñanza acorde con la actualidad del mercado, sino que también se podría habilitar la docencia semipresencial o a distancia en este tipo de asignaturas especializadas y se sustituirían los computadores más caros, ya obsoletos.

2. OBJETIVOS

Considerando lo previamente expuesto y dadas las características de un proyecto científico-tecnológico, formulamos en este apartado los objetivos de este Trabajo de Fin de Grado.

El objetivo fundamental de este proyecto reside en desarrollar un soporte inicial para la futura incorporación de SBC de bajo coste en el laboratorio docente de sistemas de tiempo real, así como proporcionar un entorno de desarrollo apto para el uso de alumnos de últimos cursos de Grado o Máster. Dentro de los objetivos secundarios encontramos:

- Conocer la potencialidad de la familia de placas STM32 y desarrollar un análisis exhaustivo de las características y el funcionamiento de los modelos de placas STM32F407 Y STM32F769.
- Diseñar una puesta en marcha de aplicaciones en lenguaje Ada utilizando las placas STM32F407 y STM32F769.
- Facilitar el manejo de dispositivos conectados en estas placas. Desarrollando una interfaz intuitiva que permita hacer uso de estos con una configuración sencilla.
- Fijar unas bases que sirvan para el desarrollo de proyectos software/hardware con el lenguaje de programación Ada y el IDE GPS junto con un cargador y debugger estable.
- Habilitar el uso de la red de comunicaciones para poder desarrollar sistemas distribuidos de tiempo real y que sirva de base para aplicaciones del Grupo de Ingeniería Software y Tiempo Real a nivel Ethernet.
- Desarrollar un demostrador que muestre cómo podría incorporarse los SBC en los laboratorios de algunas asignaturas impartidas en grado en Ingeniería Informática, cuyas guías docentes estén relacionadas con “Tiempo Real”, “Programación Paralela Concurrente y de Tiempo Real” o similares.

3. HERRAMIENTAS Y TECNOLOGÍAS

En este capítulo se abordarán las herramientas y tecnologías utilizadas para los microprocesadores de nuestro proyecto: STM32F407 y STM32F769. Para ello, se explicará en qué consiste la arquitectura ARM, el compilador GNAT, el proceso de desarrollo cruzado, el IDE de desarrollo, los debugger OpenOCD y GDB, el lenguaje Ada, el perfil Ravenscar SFP y el STMCubeMx. Además, también se darán a conocer las distintas alternativas que se han valorado, en cuanto a librerías (ADL y Ada-ENET) y la forma en la que se ha procedido durante todo el bagaje. Sin embargo, antes de avanzar en estos temas, consideramos esencial tratar algunos aspectos básicos como el concepto de microcontrolador.

3.1. MICROCONTROLADOR

Esta invención del americano Gary Boone fue desarrollada en 1971 se acuñó en primer lugar con el nombre de TMS1802NC [31]. La idea detrás del concepto de microcontrolador[1] se corresponde con un ordenador pequeño en un solo circuito integrado que contiene un núcleo de procesador, memoria y periféricos con entradas y salidas programables.

Los microcontroladores hoy en día están diseñados para aplicaciones integradas y la mayoría de ellos se integran en otras máquinas, como automóviles, teléfonos o, sistemas informáticos. Esto es lo que se conoce como sistema embebido, donde un microcontrolador realiza pocas funciones muy concretas en un sistema de computación de tiempo real.

El diccionario Merriam-Webster ofrece la siguiente definición de microcontrolador [30]:

> "Microcontroller: an integrated circuit that contains a microprocessor along with memory and associated circuits and that controls some or all of the functions of an electronic device (such as a home appliance) or system."

Un circuito integrado que contiene un microprocesador junto con la memoria y los circuitos asociados y que controla algunas o todas las funciones de un dispositivo electrónico (como un electrodoméstico) o sistema.

Si bien esto es sólo una definición. En este Trabajo, vamos a trabajar en el microcontrolador, programándolo y usándolo para crear algunas aplicaciones orientadas a las asignaturas del Grado.

3.2. ARM

ARM Holdings es una multinacional británica que desarrolla el conjunto de instrucciones y la arquitectura para productos basados en ARM [7]. Las siglas ARM corresponden a

[1] "A microcontroller is a small computer (SoC) on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals." https://www.scitechnol.com/embedded-system/microprocessor-microcontroller.php

Advanced RISC Machine. Y, a su vez, RISC significa *Reduced Intruction Set Computer*; es decir, ordenador con un conjunto reducido de instrucciones.

Entre las compañías que hacen chips que implementen una arquitectura ARM se incluyen empresas de gran renombre mundial como Apple, Nvidia, Samsung Electronics o ST Microelectronics.

A nivel mundial, ARM es la arquitectura de conjuntos de instrucciones más utilizada. Esta popularidad se debe mayoritariamente al bajo consumo de energía que consumen los procesadores ARM.

Actualmente, hay tres perfiles de arquitecturas para el procesador RISC ARM de 32 bits que destacan en popularidad y en los cuales varía su núcleo [8][4][5][6]: La familia de núcleos ARM Cortex-R están optimizados para implementar el perfil de tiempo real o ARM Real-time (R). Del mismo modo, la familia Cortex-A implementan el perfil de Aplicación (A) y, por último, la familia Cortex-M implementa el perfil de Microcontrolador (M).

Como cada familia consta con unas características propias y está optimizada para un tipo de perfil específico, la elección de un tipo de familia dependerá de la aplicación que se pretenda construir. Teniendo en cuenta la gran oferta que presenta esta arquitectura, este trabajo está orientado dentro de la arquitectura Cortex-M. Más específicamente, Cortex-M4 y Cortex-M7.

3.3. GNAT

El compilador es el software que se encarga de la compilación [37] - en nuestro caso compilación cruzada - y, además, es uno de los componentes necesarios para implementar el entorno de compilación.

En nuestro proyecto, hemos optado por el compilador GNAT ya que es el único compilador gratuito para Ada, el lenguaje de programación elegido, y está basado en la infraestructura de compilación de GCC de software libre. [20][21]

El compilador GNAT fue desarrollado por AdaCore [34]. AdaCore nació fruto de un proyecto que tuvo lugar en la universidad de Nueva York (NYU). El objeto de ese proyecto era la creación del compilador GNAT para Ada 95, sin embargo, una vez finalizado dicho proyecto, la universidad no contaba con los recursos suficientes para validar y certificar el software. Ante esta situación, se decidió crear una organización privada que se encargase del mantenimiento a largo plazo, la cual se llamaría AdaCore Technologies.

En la actualidad, AdaCore [1] continúa siendo un proveedor de software comercial y de código abierto para el lenguaje Ada enfocado a grandes aplicaciones y de alta criticidad; se encarga de su mantenimiento y periódicamente publica versiones públicas bajo licencia *General Public License* (GPL)[2]. Además, es el proveedor del IDE de desarrollo utilizado

[2] "GNU General Public License, también conocida como General Public License (GPL), es una licencia de software libre ampliamente utilizada, originalmente escrita por Richard Stallman para el Proyecto GNU. El Proyecto GNU se lanzó en 1984 para desarrollar un sistema operativo completo de estilo UNIX que es software libre: el sistema GNU.

en este proyecto y en la gran mayoría de proyectos desarrollados en Ada: GPS. Su página web oficial es https://www.adacore.com/

3.4. PLACAS STM32FX

Los microcontroladores de la familia STM32 [40] cuentan con multitud de versiones, lo que hace posible encontrar uno que se adapte bien a las características del proyecto a desarrollar. Están basados en procesadores ARM-Córtex-M de 32 bits, permitiendo una multitud de funcionalidades en aplicaciones embebidas de tiempo real combinado con un bajo consumo.

3.4.1. STM32F407 Discovery

La placa STM32F407 [43] cuenta con un microcontrolador que tiene la arquitectura ARM Cortex-M4, e incluye todo lo necesario para que tanto los usuarios principiantes como los usuarios con experiencia en este tipo de placas comiencen rápidamente.

Por otro lado, las características de las placas de esta serie son:

Procesador:	ARM Cortex-M4 a una frecuencia de reloj entre 84 y 180MHz
Memoria:	- 192 KB de memoria RAM - 1 MB de memoria ROM
Otros:	- Dos pulsadores Botón de usuario y Botón de Reset - Acelerómetro de 3 ejes - Fuente de alimentación de aplicación externa: 3 V y 5 V - Sensor de temperatura
Imagen:	 
Información en la web de la empresa:	https://www.st.com/en/evaluation-tools/stm32f4discovery.html

Esta placa la hemos utilizado a modo de introducción en la programación embebida, dado que es más simple y dispone de menos conectores que la STM32F769.

3.4.2. STM32F769iDiscovery

Esta placa STM32F769 [44] tiene un microcontrolador con arquitectura ARM Cortex-M7. Y sus características principales se exponen en la siguiente tabla:

Procesador:	ARM Cortex-M7 con una frecuencia de reloj máxima de 216 MHz
Memoria:	- Mb de memoria ROM - 512+16+4 Kb de memoria RAM
Otros:	- Pantalla táctil LCD de 4 pulgadas - Conector ethernet compatible con IEEE-802.3-2002 - Dos botones integrados sobre la propia tarjeta: botón de usuario y botón Reset. - Slot para tarjetas microSD - Dos jacks de 3.5mm para la conexión de periféricos de entrada y salida respectivamente - Conectores Arduino™ Uno V3
Imagen:	
Información en la web de la empresa:	https://www.st.com/en/evaluation-tools/32f769idiscovery.html

Desde nuestro punto de vista, consideramos que ambas placas son muy interesantes además de lo suficientemente potentes para proyectos electrónicos de cualquier tipo.

No obstante, para este proyecto en concreto hemos utilizado un microcontrolador STM32F769 por su capacidad de procesamiento, por la disposición de conectores Arduino Uno V3 y por disponer de pantalla táctil integrada. El hecho de que el microcontrolador sea compatible con Arduino nos permite aprovechar la gran cantidad de dispositivos que existen en el mercado para las placas Arduino. Además, estos conectores también nos permiten que el microcontrolador pueda comunicarse con otros dispositivos externos mediante distintos protocolos, lectura y escritura de valores analógicas y digitales, tomas de tensión a distintos voltajes y tomas de tierra necesarias para periféricos externos, entre otros.

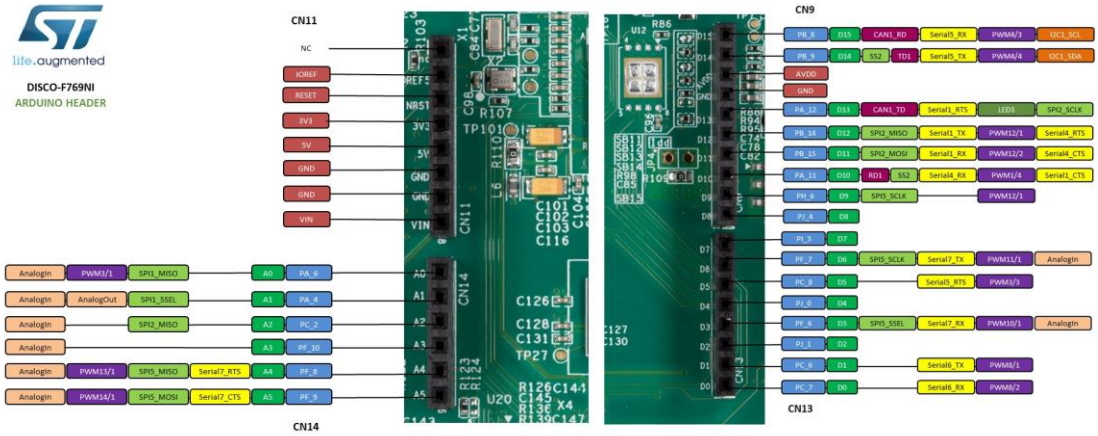

Figura 1 - Conexiones STM32F769

En esta figura se pueden ver los distintos tipos de pines disponibles en el conector Arduino Uno V3 disponibles en la tarjeta STM32F769.

3.5. OPENOCD

OpenOCD, cuyas siglas hacen referencia a *On-Chip Debugger*, es un proyecto de código abierto que ofrece soporte para depurar y cargar programas en dispositivos integrados [33]. Además, permite escribir en la memoria ROM de todos los dispositivos de la familia STM32; por ello, nos permite cargar el programa directamente en la memoria ROM del microcontrolador que hemos seleccionado y con la dirección de memoria inicial que se encuentra en el script de la placa correspondiente.

OpenOCD también puede comunicarse con el debugger GDB y así se conectarse con el dispositivo externo para poder depurar los programas cargados en él, mediante una conexión de socket. Para cargar y depurar programas hemos utilizado OpenOCD en su versión 0.10.0.

3.6. GDB

En este proyecto hemos utilizado GNU Debugger (GDB), que permiten depurar programas en entornos cruzados. Depurar (debug) es el proceso encontrar y corregir los errores o *bugs* que contenga un programa. Más específicamente, cuando se ejecuta la aplicación paso a paso, se puede observar tanto el flujo seguido durante su ejecución como los resultados intermedios. Finalmente, se detectan las anomalías.

GDB es el depurador de proyectos por excelencia de GNU. Es de gran utilidad en la práctica puesto que permite conocer que es lo que está sucediendo dentro de un programa en ejecución, el estado de las variables, en tiempo de ejecución, el estado de las tareas, la memoria o la siguiente instrucción que se va a ejecutar. GDB soporta multitud de leguajes y entre ellos Ada.

Existen distintas formas de realizar la depuración de programas, sin embargo, la utilización de *breakpoints* o puntos de interrupción son muy utilizados en estas situaciones. Consisten en puntos de parada de la CPU, de tal forma que al llegar a la

instrucción donde se encuentra este *breakpoint*, la CPU se para y no continúa ejecutando el código hasta que reciba la orden de continuar.

3.7. LENGUAJE ADA

Ada es un lenguaje de desarrollado en 1974 por el departamento de defensa de los Estados Unidos con el propósito de utilizarse en sistemas embebidos.

Si hacemos una comparación con otros lenguajes, Ada es un lenguaje de gran extensión, ya que aborda muchos temas importantes relevantes en la actualidad para la programación de sistemas prácticos del mundo real.

Las características principales del lenguaje son: tipificación muy fuerte, la mayoría de los errores son detectados en tiempo de compilación o por las restricciones en tiempo de ejecución. Un programa desarrollado en Ada se puede comprender con una sola lectura debido a su gran legibilidad, permite la programación orientada a objetos, gestión de excepciones, comunicación con sistemas de alta integridad y criticidad y además incorpora la gestión de tareas de forma nativa [10].

Por todo ello, es el lenguaje utilizado por el grupo ISTR de la Universidad de Cantabria para desarrollar proyectos en sistemas embebidos en los que es habitual que sea necesario crear varios hilos de ejecución. Esta planificación de tareas permite realizar el trabajo de forma más eficiente aprovechando al máximo todos los recursos que pueda ofrecer el procesador y minimizando el consumo de energía.

3.8. PERFIL RAVENSCAR SFP

El perfil de Ravenscar es un subconjunto del lenguaje de programación Ada que se empezó a desarrollar en 1997 para los sistemas de tiempo real de alto rendimiento e integridad [36]. Alan Burns, profesor del departamento de informática en la universidad de York en su libro *The Ravenscar Profile*, afirma que está diseñado para ser lo suficientemente pequeño como para permitir un soporte de tiempo de ejecución eficiente y certificable, pero lo suficientemente grande como para permitir los estilos de programación necesarios en la práctica para sistemas en tiempo real [11][12]. Está destinado a aplicaciones que tienen un número fijo de tareas donde cada cuerpo de tarea es un bucle. Entre otras de sus características detacamos:

- No hay declaraciones relativas de espera.
- No hay referencias al paquete Ada.Calendar.
- Cada expresión de barrera de entrada debe ser una sola variable booleana.
- Tiene una política de planificación de tareas FIFO con prioridades.
- Se permiten referencias al paquete Ada.Real_Time.

Para los proyectos realizados hemos utilizado el perfil Ravenscar *Small Footprint Profile* (SFP) puesto que se ajusta a las caracteristicas de rendimiento que se necesitan en este proyecto.

3.9. STMCUBEMX

STM32CubeMx[3] es una herramienta gráfica desarrollada por la propia marca de los microprocesadores STM32 para facilitar la preconfiguración de todos los drivers necesarios con el fin de crear proyectos en C [42] proporcionando librerías para toda su gama. Genera el código necesario para crear un proyecto basado en las especificaciones definidas. Su utilización es bastante sencilla: tras seleccionar el microcontrolador con el que se va a trabajar, la aplicación permite definir multitud de parámetros del proyecto como, por ejemplo: activación del DMA, características del ADC, configuración de ciertos periféricos e incluso calcular el consumo estimado de energía.

Otra herramienta muy interesante es el IDE de STM32, que consiste en el IDE Eclipse, con un *pluggin* para poder integrar un proyecto hecho en STM32CubeMX y continuar su desarrollo. Desde este IDE podemos compilar el proyecto y depurarlo con GCC y GDB respectivamente. Esto nos permite generar un código de inicialización y configurarlo a nuestras necesidades en el lenguaje C o C++ y, posteriormente, poder continuar su desarrollo en otros lenguajes como Ada. Todo este proceso es posible gracias a que Ada permite realizar llamadas a funciones de un programa escrito en C mediante la creación de enlaces. Es importante destacar que ambas herramientas están disponibles tanto para Windows como para Linux.

3.10. ADA DRIVERS LIBRARY (ADL)

A pesar de que en la web de STM no se habla de la compatibilidad de sus dispositivos con Ada, la compañía AdaCore junto con la ayuda de distintos desarrolladores ha creado drivers para dichos microcontroladores. Los drivers están disponibles en GitHub[4] bajo en nombre Ada_Drivers_Library y son de gran utilidad para tomarlos como base en el desarrollo de proyectos. Además, y como algo complementario, existen proyectos realizados por desarrolladores independientes que pueden servir como ejemplos de uso y utilización.

Ada_Drivers_Library incluye drivers para el uso y control de "GPIOs" (*general-purpose input/output*), "DAC/ADC" (*Analog to Digital Converter (ADC) and Digital to Analog Converter (DAC)*), "PWM" (*Pulse Width Modulation*), protocolos de comunicación "I^2S" (*Inter-IC Sound*), "I^2C" (*Inter-Integrated Circuit*), "SPI" (*Serial Peripheral Interface*), gestión de timers, control de interrupciones, audio de entrada y salida, o la pantalla táctil, entre otros.

3.11. ADA – ENET

AdaENET es una biblioteca desarrollada por el ingeniero de software francés Stéphane Carrez y presentada en la Conferencia Internacional sobre Tecnologías de Software Seguro: Ada-Europe 2017.

[3] STM32CubeMx https://www.st.com/en/development-tools/stm32cubemx.html
[4] URL del repositorio GitHub https://github.com/AdaCore/Ada_Drivers_Library

Esta biblioteca implementa los protocolos "ARP"[5] (*Address Resolution Protocol*), "IPv4"[6] (*Internet Protocol version 4*), "UDP"[7] (*User Datagram Protocol*), "DNS"[8] (*Domain Name System protocol*) y "DHCP"[9] (*Dynamic Host Configuration Protocol*) sobre un controlador Ethernet. La biblioteca puede compilarse para las placas STM32F746 o STM32F769 proporcionando al proyecto acceso a la red sobre el protocolo elegido.

Esta biblioteca se puede descargar de GitHub: https://github.com/stcarrez/ada-enet, donde se encuentra la información sobre cómo configurarla en nuestra placa. Además, incluye distintos proyectos de ejemplo de uso de la biblioteca.

3.12. PROCESO DE DESARROLLO CRUZADO

La compilación es un proceso de conversión que pertenece a la fase de codificación para el desarrollo de un programa y se encarga de la traducción del código fuente al código objeto (código máquina) utilizando un compilador. [14]

Sin embargo, para crear un código fuente en un entorno A que vaya a ser traducido a un código objeto en un entorno B – donde A y B tienen distintas arquitecturas – necesitamos utilizar el tipo de compilación denominada compilación cruzada. Por su función, ésta es la compilación que se usa en el desarrollo de programas para sistemas embebidos [13][15][16].

No obstante, para realizar la compilación cruzada es esencial contar con una serie de programas y librerías que establezcan un entorno de compilación cruzada; es decir, el ambiente adecuado para llevar a cabo este proceso de conversión.

En nuestro proyecto - como está representado en la siguiente figura - el entorno A es un ordenador que desempeña la función de *host* (sistema huésped) con arquitectura x86 (Intel i7) y sistema operativo Linux; mientras que el entorno B es el microcontrolador STM32F769iDiscovery que actuará de *target* (sistema objetivo), con una arquitectura ARM.

[5] ARP es un protocolo de comunicaciones de la capa de red, responsable de encontrar la dirección de hardware (Ethernet MAC) que corresponde a una determinada dirección IP. [9]
[6] El Protocolo de Internet versión 4 (IPv4) es la cuarta versión del Protocolo de Internet (IP). Es uno de los protocolos centrales de los métodos de interconexión de redes basados en estándares en Internet y otras redes de conmutación de paquetes. [28]
[7] UDP es un protocolo de comunicaciones alternativo al Protocolo de Control de Transmisión (TCP) utilizado principalmente para establecer conexiones de baja latencia y tolerancia a pérdidas entre aplicaciones en Internet. [45]
[8] DNS traduce los nombres de dominio a direcciones IP para que los navegadores puedan cargar recursos de Internet. [19]
[9] DHCP es un protocolo de administración de red utilizado en redes UDP / IP mediante el cual un servidor DHCP asigna dinámicamente una dirección IP y otros parámetros de configuración de red a cada dispositivo en una red para que puedan comunicarse con otras redes IP. [18]

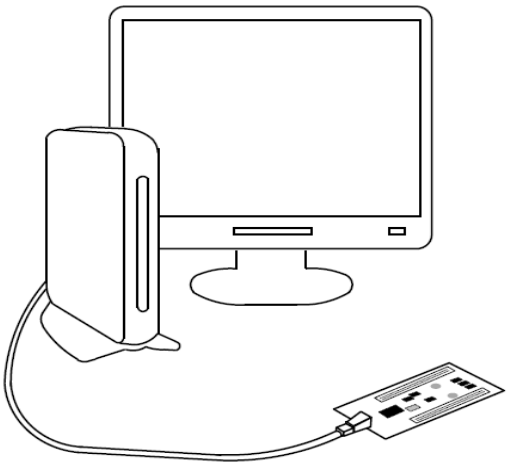

Figura 2 - Representación de la compilación cruzada, entornos A y B

La creación, depuración y edición de programas que se cargarán sobre el microcontrolador tienen lugar en el sistema huésped, donde está instalado el entorno de desarrollo, pero no se ejecutarán allí. Posteriormente, mediante el proceso de compilación y linkado se traduce el programa creado en un código máquina ejecutable y compatible con la arquitectura del microcontrolador, pero no del ordenador. Finalmente, el programa creado queda cargado sobre el microcontrolador.

Hay que tener en cuenta que los microcontroladores de placa única (SBC) no tienen ningún entorno de desarrollo para poder crear los programas que posteriormente ejecutará, por lo que es necesario un software de compilación cruzada instalado en un PC. Básicamente, los ordenadores son mucho más potentes que los microcontroladores y se utilizan para agilizar el proceso de desarrollo. Por ello, y debido a esta desventaja, las características del SBC utilizado deben estar presentes durante todo el proceso.

3.13. IDE DE DESARROLLO: GPS

Integrated Development Environment (IDE), en español Entorno de Desarrollo Integrado, es un software que proporciona y consolida las herramientas básicas necesarias para escribir y probar software [27]. Además, el IDE puede identificar y minimizar errores de codificación, así como tipográficos.

Para el desarrollo de los programas que serán cargados en la placa hemos utilizado el IDE proporcionado por AdaCore cuyas siglas se corresponden con GPS [24] (GNAT Programming Studio).

Este IDE proporciona un entorno versátil donde poder gestionar todos los ficheros que componen el proyecto, el compilador a utilizar e incluso distintas formas de depurar el programa en cuestión.

4. DESARROLLO DE LIBRERÍA

Desde un primer momento, dispusimos de dos alternativas: las librerías en C proporcionada por STM junto con la herramienta STM32CubeMx y las librerías Ada Drivers Library (ADL) en Ada. Puesto que la primera opción requeriría crear una capa de adaptación para poder ser llamada desde el lenguaje Ada, nos pareció más directo utilizar la librería escrita en Ada (ADL).

En la primera toma de contacto con ADL nos dimos cuenta de que interactuar con ella directamente también podría resultar en ocasiones algo tedioso. Principalmente porque ADL es una librería muy completa que permite controlar hasta el mínimo detalle los dispositivos disponibles. Pero, a su vez, es de bajo nivel, por lo que requiere que el usuario deba gestionar todos los parámetros de configuración, lo que dificulta mucho su uso para usuarios inexpertos. Por ello y con la finalidad de facilitar el trabajo a futuros usuarios, decidimos realizar una capa de abstracción que permitiese utilizar las funciones y procedimientos que realmente son necesarios.

La interfaz de la capa de abstracción simplifica la utilización de los drivers proporcionados por ADL, puesto que evita al usuario tener que conocer todos los detalles de su funcionamiento y ahorrarse tiempo en configuraciones que pueden ser desconocidas y en muchos casos innecesarias. Esta capa de abstracción está disponible en el repositorio de GitHub: https://github.com/dao703/Robot_Car_UC/tree/master/Drivers_UC

La siguiente figura representa la arquitectura para la aplicación Ada:

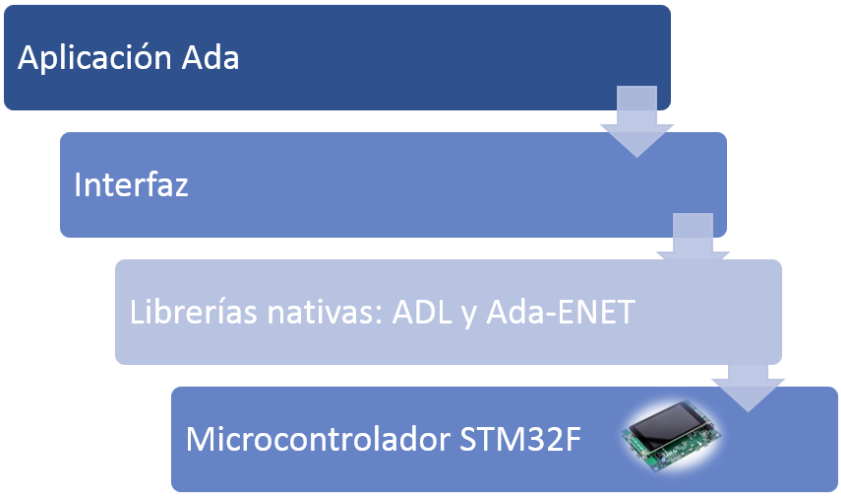

Figura 3 - Arquitectura de una aplicación Ada

Esta estructura de capas representa la arquitectura de una aplicación Ada que hace uso de las interfaces desarrolladas; a su vez, facilita el uso de las librerías nativas ayudando así en la configuración y utilización del software necesario para los dispositivos utilizados. Tiene como objetivo proporcionar una capa sencilla e intuitiva para usuarios inexperto que se están introduciendo en la materia.

A continuación, en la figura 4, se detalla la estructura interna de la capa "interfaz" de la figura anterior. Aquí se pueden ver las interfaces desarrolladas que interactúan directamente con las librerías Ada Drivers Library (ADL) y Ada-Enet.

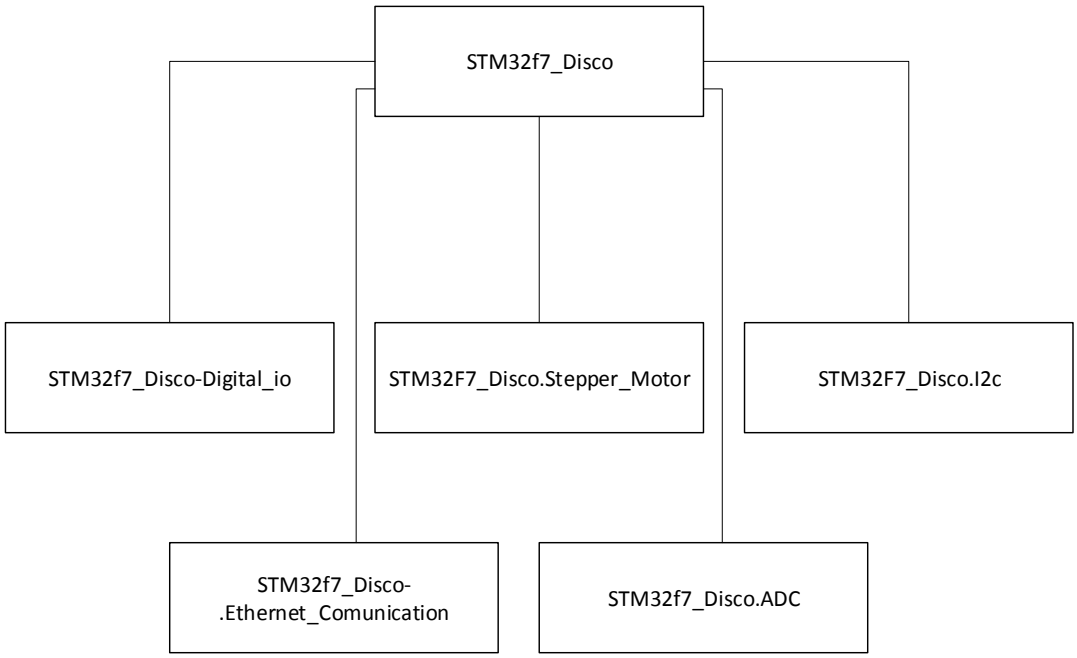

Figura 4 - Interfaces desarrolladas que interactúan con ADL y Ada-ENET

4.1. CONFIGURACIÓN DEL ENTORNO DE DESARROLLO CRUZADO

Para configurar el entorno de desarrollo con los microcontroladores STM32 debemos seleccionar el compilador para la arquitectura ARM junto con el Ada Runtime según corresponda, en la sección Toolchains de las propiedades del proyecto (véase figura 5):

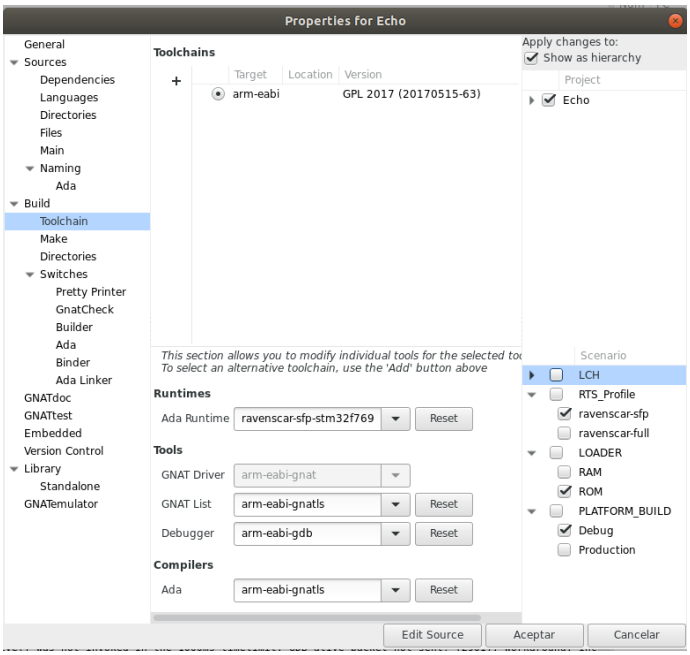

Figura 5 - Vista de las propiedades de un proyecto

En nuestro caso hemos utilizado el IDE GPS que integra el compilador GNAT en su version de 2017 por contar con un debugger estable y funcional en linux. Esta disponible en la pagina web de AdaCore en la sección descargas bajo el nombre GNAT-GPL-2017-arm.

Para configurar dicho cargador es necesario seleccionar en las opciones del proyecto, en la sección “embedded” el script de dispositivo donde se va a transferir el fichero binario, el protocolo de comunicación con el mismo y el puerto. Para nuestro caso será:

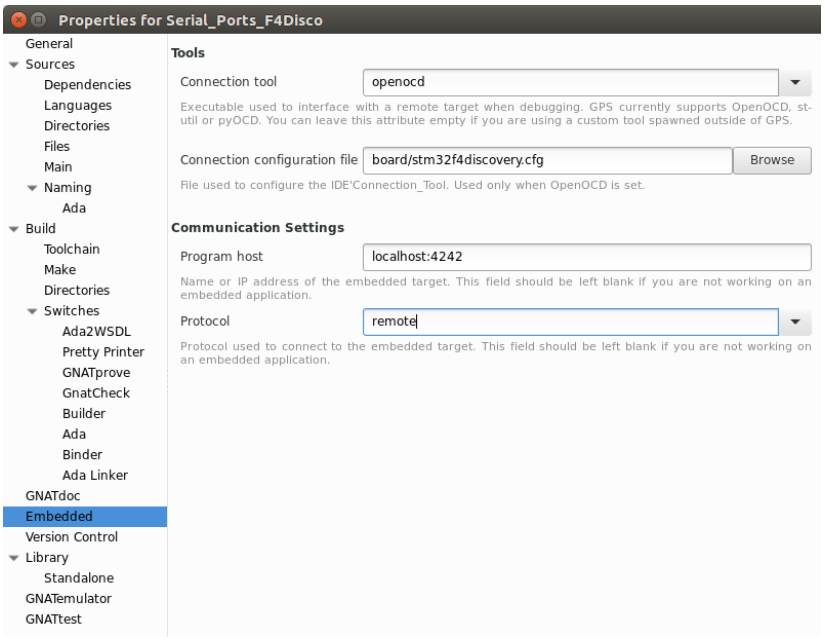

Figura 6 - Configuración de la conexión con el dispositivo STM32

En la figura 6 se muestra la configuración necesaria para la conexión a GDB, concretamente lo que hace referencia al protocolo y al socket.

4.2. INTERFAZ CON ADA DRIVERS LIBRARY

4.2.1. Entrada / salida digital

Multitud de placas (SBC) tienen integradas de forma más o menos accesible distintos pines que pueden ser configurados por software para que se comporten como una entrada o salida de digital. Estos pines reciben el nombre de GPIO cuyas siglas hacen referencia a *General-Purpose Input/Output* [22]. Su utilización permite desarrollar multitud de aplicaciones como encender o apagar un led, consultar el estado de un interruptor o de cualquier otro periférico digital. Para facilitar su utilización hemos desarrollado una capa software que hace su uso más intuitivo:

```
40      procedure Configure_Pin (Pin : GPIO_Point; Mode : Pin_Mode);
41
42      function Digital_Read (Pin : GPIO_Point) return Boolean;
43
44      procedure Digital_Write (Pin : GPIO_Point; Value : Boolean);
45
46      procedure Toogle_Pin (Pin : GPIO_Point);
```

Figura 7- Drivers para la entrada/salida digital

Como se puede observar en la figura anterior, la interfaz proporciona procedimientos y funciones para configurar un pin como entrada o salida digital, y para leer y cambiar su estado. En funcion de su configuracion se utilizan los métodos de lectura o escritura para ver el estado del pin o establecerle un valor digital.

El procedimiento `Configure_Pin` se encarga de llamar al procedimiento de configuración de los GPIO con el pin a habilitar junto con su configuración. Gracias a este procedimiento se consigue ocultar detalles de configuración, con lo que se facilítala la tarea al usuario. Esta configuración incluye parámetros como su Modo(entrada/salida), frecuencia de refresco (varía entre 2MHz y 100MHz) y el tipo de resistencia que asociada (Pull-Up, Open-Drain o Floating). En nuestro caso se ha utilizado una frecuencia de refresco de 100Mhz y resistencias Floating puesto que son los más apropiados para nuestros proyectos.

Los métodos `Digital_Read` y `Digital_Write` hacen llamadas a procedimientos llamados set del paquete STM32.GPIO de la librería ADL para leer o escribir un valor. El nombre de estos métodos en la librería ADL puede llevar a equívoco puesto que lo que hacen es completamente opuesto y pueden llevar a confusión a un inexperto.

4.2.2. Entrada analógica: ADC

Un conversor analógico digital (ADC) es un dispositivo utilizado para que un sistema digital sea capaz de leer un valor analógico [2][3]. Cuando un dispositivo ADC detecta una tensión analógica, realiza una conversión a un valor binario en un periodo de tiempo. Es decir, el dispositivo toma muestras de la tensión leída como valor de entrada, y produce un valor binario como salida. En el caso de nuestro microcontrolador, disponemos de tres ADC para poderlos utilizar de forma independiente o combinándolos entre sí.

La tasa de resolución de un ADC viene definida por el número de bits que tenga la misma. Esto será lo que determine el número de escalones que existirán a la hora de leer el valor analógico en Voltios. Esto viene dado por la ecuación 2^n donde n es el número de bits. La figura 8 muestra la función de transferencia de un ADC con 4bits y 5 V. Dado que hay 4 bits, el número total de escalones es $2^4 = 16$. En las STM32F4 y F7 existen distintas configuraciones: 12-bit, 10-bit, 8-bit o 6-bit configurables por software.

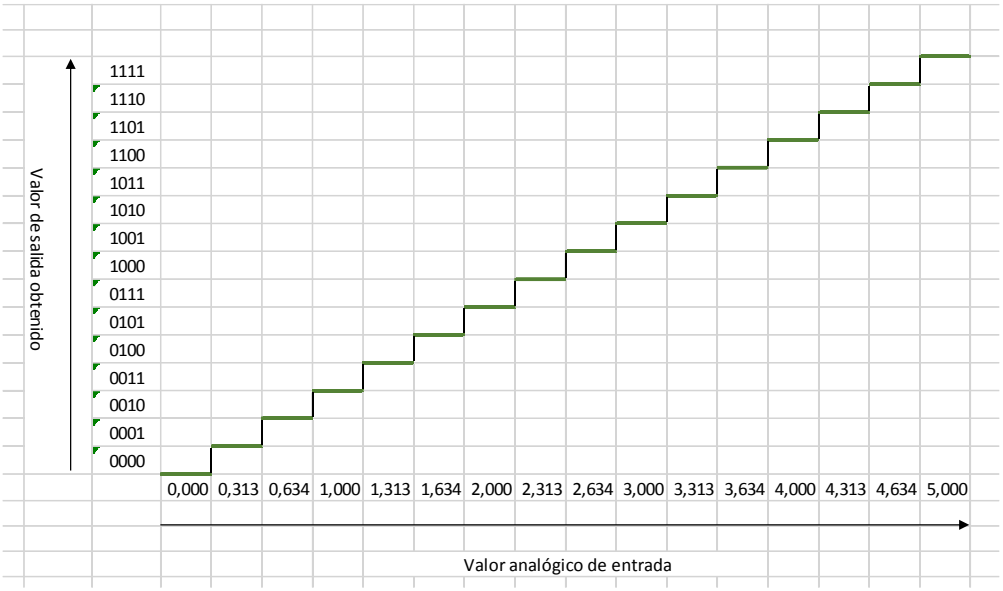

Figura 8 - Función de transferencia de un ADC con 4bits y 5V

La frecuencia de muestreo viene determinada por las características propias del dispositivo, en nuestro caso las muestras se pueden tomar con un tiempo de espera mínimo de 9 ciclos de reloj para una resolución de 6 bits. Cuantas más muestras se tomen en un segundo, más precisa será la lectura de la tensión analógica, lo que hace posible que tener una tasa de muestro lo suficientemente grande como para que no se produzca un solapamiento en las tensiones leídas donde dos señales pasarían a ser indistinguibles. Para evitar esto, se recomienda que la tasa de muestreo sea, al menos, dos veces superior a la frecuencia más alta a transmitir, lo que se conoce como tasa de Nyquist.

```
51      procedure Configure_ADC_1_Channel_6;
52
53      procedure Start_Conversion_ADC_1_Channel_6;
54
55      function Get_Value_ADC_1_Channel_6 return UInt16;
```

Figura 9 - Drivers para el uso del ADC

Como muestra la figura 9, para poder hacer uso de un ADC se disponen de 3 métodos principales: la configuración, el comienzo de la lectura y finalmente la obtención del valor obtenido. Este paquete permite hacer uso del ADC_1 en el canal 6, siendo el canal el pin en el que se tiene que conectar el dispositivo analógico.

Mientras que en la librería ADL los procedimientos de comienzo de la lectura y de la obtención de su valor requieren llamar a un solo procedimiento de la librería ADL; para inicializar u obtener su valor, el procedimiento de configuración es mucho más complejo. Este procedimiento en primer lugar se encarga de habilitar el reloj asociado al conversor analógico digital. En segundo lugar, borra cualquier tipo de configuración que tuviera asociada. A continuación, configura las propiedades comunes a todos los ADC del microcontrolador. Estas propiedades van desde indicar si la lectura del ADC es simple, doble o triple en función de si se van a utilizar uno o varios ADC, especificar si se quieren transferir los datos a memoria y como con DMA o la velocidad de operación de los ADC. Para finalizar con la configuración, se pasa a definir las propias características del ADC:

la resolución (define la precisión de la conversión. Se puede elegir un valor entre 6,8,10 o 12 bits), la organización del dato obtenido en el registro (alineados a la derecha o a la izquierda) si se desea habilitar un disparador asociado al convertidor, o activar un aviso cuando se haya realizado un valor y los ciclos de reloj a esperar entre distintas muestras. En nuestro caso hemos elegido utilizar el conversor de manera simple, deshabilitar el DMA, una resolución de 12 bits alineados a la derecha, con una frecuencia de muestreo de cada 15 ciclos y deshabilitando el resto de las opciones. Se ha elegido esta configuración puesto que es lo suficientemente versátil y completa como para poder realizar el tipo de prácticas del laboratorio.

4.2.3. Motor paso a paso

Los motores paso a paso [32][41] son un tipo de motores que convierten pulsos electicos en movimientos discretos del rotor. Es decir, el eje de este tipo de motores gira paso a paso a medida que recibe impulsos electricos que excitan sus bobinas de forma consecutiva. La secuencia de activacion de cada bobina esta relacionada directamente con el setido de giro del motor. Estos motores normalmente tienen acoplado una reductora para incrementar la precisión de sus movimientos. Su comportamiento en lazo abierto – sin feedback que determine su posición - es muy preciso, puesto que el ángulo de giro por pulso es fijo.

El sentido de giro viene definido por la secuencia de pulsos y su velocidad, con la frecuencia de los pulsos recibidos en la entrada. Lo habitual es que este tipo de motores se utilicen con un controlador que facilita su utilización. Este controlador tiene 4 pines que hacen referencia a cada una de las 4 bobinas y los conectores de alimentación que en nuestro caso utilizaremos el de 5V.

Encendiendo y apagando las bobinas, siguiendo una secuencia ordenada, se consigue el movimiento del rotor. A continuación, la siguiente figura muestra un ejemplo de cómo excitar las bobinas para que se produzca el movimiento de una fase al mismo tiempo y conseguir que el motor gire:

Paso	Bobina 1 – In1	Bobina 2 – In2	Bobina 3 – In3	Bobina 4 – In4
1	On	Off	Off	Off
2	Off	On	Off	Off
3	Off	Off	On	Off
4	Off	Off	Off	On

Figura 10 - Ejemplo excitación de bobinas

También existe la posibilidad de excitar varias bobinas correlativas a la vez para conseguir más par o duplicar el número de pasos según se desee.

```
16
17      procedure Initialize_Motor(M : Motor );
18
19      procedure Change_Rotation(M :in out Motor);
20
21      procedure Move_One_Step(M :in Out Motor; Lag : Duration);
22
23      procedure Move_N_Steps(M :in out Motor;Num_Steps : Natural; Lag : Duration);
24
25      --this procedure prevents the coils from burning
26      procedure Power_Off (M :in out Motor);
```

Figura 11 - Drivers para el uso de motores paso a paso

Con esta librería se consigue configurar un motor de forma sencilla estableciendo los pines donde se conecta las cuatro entradas del controlador del motor paso a paso y su sentido de giro. El prcedimiento `Initialize_Motor` se encarga de configurar la estructura de datos Motor asignado los pines que alimentaran al motor como salidas digitales y el sentido inicial de giro.

El procedimiento `Change_Rotation`, como bien indica su nombre, invierte el sentido de giro de forma que la siguiente vez que se mueva el motor, las bobinas se excitaran en sentido contrario. `Move_One_Step` y `Move_N_Step` se encargan de mover el motor activand y desactivando los pines correspondientes para continuar el giro del motor, según corresponda.

Finalmente el procedimiento `Power_Off` apaga el motor, desactivando todas las salidas digitanes asignadas al mismo, es decir sus pines. Previene que el motor se pueda quemar, si se quedo activada alguna de las salidas. Esta librería se ha realizado sin apoyarse en otra capa softwara por debajo, que facilite la configuracion de este tipo de motores. El motivo de esto es que ADL no implementa nada parecido al respecto

4.2.4. Protocolo I^2C

El I^2C es uno de los protocolos de comunicación síncrona más utilizados en cualquier tipo de dispositivo electrónico, o equipos electrónicos industriales. Sus siglas provienen del inglés *Inter-Integrated Circuit* desarrollado por Phillips en 1980 [35]. Está formado por un bus de dos cables: SDA - Cable de datos serie - y SCL - Cable de reloj serie -. Cada dispositivo conectado al bus es direccionarle a través de una dirección lo que permite que existan relaciones maestro esclavo en todo momento, donde los maestros pueden hacer las funciones de transmisores o receptores. Incluye detección de colisiones y arbitrio con el fin de prevenir que dos o más maestros realicen una transmisión de datos al mismo tiempo. La comunicación entre dispositivos está orientada a 8 bits con diferentes modos de velocidad entre los 100 kbit/s y 3.4 Mbit/s

```
28      function Is_Port_Enabled return Boolean;
29
30      function Is_Configured return Boolean;
31
32      procedure Begin_Transmision (Clock_Speed : UInt32 := 100_000);
33
34      procedure Write (Direction : HAL.I2C.I2C_Address; Intput_Data : HAL.I2C.I2C_Data);
35
36      procedure Read (Direction : HAL.I2C.I2C_Address; Output_Data : out HAL.I2C.I2C_Data);
37
38      function  Get_Status return I2C_Status;
```

Figura 12 - Drivers para el uso del I2C

Para comunicarse con un dispositivo a traves del protocolo I^2C es necesario conocer su dirección, para asi poder enviarle el mensaje de referencia que deseemos. Con estos métodos su utilización es más intuitiva y facilita en gran medida su uso.

A diferencia del ADC, este protocolo no requeiere de una configuración muy compleja con demasiados parametros. El metodo para su configuracion es `Begin_Tansmition`. Este metodo se encarga de configurar el I^2C, para ello neceista configurar los pines asociados a la señal de reloj y datos junto con la frecuencia de reloj asociado al bus. Esta frecuencia define la velocidad de comunicación en el protocolo, para nuestros proyectos hemos considerando suficiente una frecuencia de 100KHz. Los metodos `Write` y `Read` envían y reciben informacion a los dispositivos localizados en las direcciones que reciben como parametros. Para concluir con esta interfaz, `Get_Status`, `Is_Configured` y `Is_Port_Enable` son 3 métodos que dan información acerca del estado del protocolo.

4.2.5. LCD

Las siglas LCD provienen del inglés *Liquid Crystal Display* o pantalla de cristal líquido [29]. Es la tecnología utilizada para pantallas en computadores portátiles y otros computadores más pequeños. Al igual que las tecnologías de diodo emisor de luz (LED) y plasma de gas, las pantallas LCD permiten que las pantallas sean mucho más delgadas en comparación con otras como las de tubo de rayos catódicos (CRT). Las pantallas LCD consumen mucha menos energía que las pantallas LED y de gas porque funcionan bloqueando la luz en lugar de emitirla.

En nuestro proyecto, la pantalla LCD es un componente que juega un papel muy importante a la hora de interactuar con cualquier dispositivo externo. Es un elemento muy útil tanto como para mostrar el resultado final como para utilizarla como complemento de depuración.

La utilización de la pantalla del microcontrolador se ha realizado directamente a través de la interfaz proporcionada por Ada_Drivers_Library realizando llamadas a los métodos del paquete HAL.Bitmap. De la misma manera, para obtener la posición sobre la que presiona un usuario, se hace uso del paquete HAL.Touch_Panel. Estos paquetes contienen métodos funciones y procedimientos suficientemente claros como para que alguien su mucha experiencia los pueda utilizar: Para inicializar la pantalla simplemente tenemos que llamar a los métodos `Display.Initialize` y `Display.Initialize_Layer`.

El primero de ellos inicializa la pantalla mientras que el segundo inicializa una capa. Las capas son entidades virtuales en las que se puede escribir y hacer intercambio entre ellas para mostrar unas u otras según se desee. Para escribir un mensaje simplemente tenemos que llamar al procedimiento `LCD_Std_Out.Put_Line` pasándole como parámetro el mensaje que queremos que se muestre y este aparecerá en la pantalla. Por el contrario, si lo que queremos es limpiar la pantalla el procedimiento `LCD_Std_Out.Clear_Screen`, es el que debe ejecutarse. Por otra parte, también podemos obtener el valor en coordenadas X, Y del punto que ha pulsado un usuario. Para ello debemos inicializar previamente el panel táctil llamando al procedimiento `Touch_Panel.Initialize`, y después podremos obtener dichos datos con el método `Touch_Panel.Get_Touch_Point`.

Con el fin de introducir a los usuarios innexpertos en la utilización de este dispositivo se han desarrollado una serie de programas de prueba que hacen uso del LCD.

4.3. INTERFAZ CON ADA-ENET

Ada-Enet es la librería elegida pues implementa los protocolos “ARP”, “IPv4”, “UDP”, “DNS”, “DHCP” sobre un controlador Ethernet. Gracias a esta librería es posible la comunicación entre dos o más microcontroladores por medio de un cable ethernet.

La placa STM32F769iDiscovery utilizada cuenta con un conector Ethernet hembra RJ45. Esto nos permite conectar la placa a internet o comunicarla con otro dispositivo a través de un cable ethernet.

```
22      procedure Initialize (Ifnet : in out Net.Interfaces.STM32.STM32_Ifnet;
23                            Mac_Src : Ether_Addr);
24
25
26
27      function Send_Message (Buffer : in out Net.Buffers.Buffer_Type;
28                             Ifnet : in out Net.Interfaces.STM32.STM32_Ifnet;
29                             Ether : in out Net.Headers.Ether_Header_Access;
30                             Mac_Src : Ether_Addr; Mac_Dst : Ether_Addr;
31                             Protocol : Uint16; Message :  Message_Type) return Boolean;
32
33
34      function Get_Message (Ifnet : in out Net.Interfaces.STM32.STM32_Ifnet;
35                            Packet : in out Net.Buffers.Buffer_Type) return Message_Type;
```

Figura 13 - Drivers para el uso de las comunicaciones por Ethernet

Se ha desarrollado la interfaz mostrada por los subprogramas que aparecen en la figura 13. Estos tres métodos permiten enviar paquetes Ethernet a través del conector de Ethernet y poder conectar la placa a la red de internet o a otra placa de forma aislada.

Se ha hecho una interfaz solamente a nivel de ethernet puesto que es lo que necesitan las asignaturas del grupo para trabajar en los laboratorios tanto a nivel local como a través de redes. En primer lugar, tenemos el método `Inicialize`. Este método simplifica en gran medida la configuración del interfaz Ethernet: tras inicializar la memoria RAM del dispositivo y hacer una reserva de espacio para los paquetes, inicializa la interfaz de Ethernet que se pasa como parámetro de entrada-salida con la dirección MAC que le hayamos indicado como parámetro para poder comenzar la comunicación. La interfaz

Ifnet es un parámetro de entrada-salida en este método debido a que pertenece al objeto que invoca este método y su estado cambia al salir de `Inicialize`, pues ahora tiene asignada la dirección MAC del dispositivo.

En el método `Send_Message` lo que hacemos es crear un paquete con dirección MAC de origen, destino y su protocolo utilizado. Una vez hecho esto, se asigna este paquete al buffer indicándole el tamaño de la trama –en nuestro caso es de 64 bytes ya que es un tamaño óptimo para nuestros proyectos pues permite enviar y recibir mensajes de 50 caracteres además de las cabeceras ethernet-y se envía a través de la interfaz de red. El tamaño de la trama está definido en la propia librería, por lo que se asigna directamente sin necesidad de que se reciba como parámetro. Los paquetes se encapsulan en un buffer para que así puedan ser serializables y enviarlos por la red.

En el método complementario, `Receive_Message`, se produce la secuencia opuesta; con la peculiaridad de que tiene una llamada bloqueante a la espera de una recepción. Una vez ha recibido el buffer que contiene el paquete con un mensaje que la fuente nos quiso hacer llegar, lo retorna como valor de salida. Estos 3 métodos simplifican en gran medida el uso de las comunicaciones de estos microcontroladores y sus drivers. Evitando así tener que hacer llamadas a varios métodos que a priori pueden resultar enrevesados y sin la necesidad de disponer en los métodos de tareas internas que se encarguen del envío o recepción de paquetes.

Por ejemplo, en el caso del envió de un mensaje, si se usara directamente la librería Ada-ENET, habría que realizar la llamada a los métodos: `Buffer.Allocate` para obtener un buffer en el que encapsular nuestro paquete, `Buffer.Set_Type (Kind => Net.Buffers.ETHER_PACKET)` para establecer el tipo de paquete que contendrá el buffer. A continuación, se asignarían las cabeceras del paquete Ethernet mediante los métodos: `Ether.Ether_Shost` –establece la MAC de origen-, `Ether.Ether_Dhost` –establece la MAC de destino- y `Ether.Ether_Type` –establece el tipo de protocolo Ethernet que se utiliza-. Después se asigna el mensaje que quiere transportarse por la red empaquetado en una trama - `Buffer.Put_String` y el tamaño total del buffer `Buffer.Set_Length (Packet_Size).` Finalmente se envía el buffer, que es la estructura serializable que se puede enviar por la red, a través de la interfaz `Ifnet.Send (Buf => Buffer);`

5. DEMOSTRADORES

En este apartado se pretende dar a conocer una serie de ejemplos sencillos que pongan en práctica las interfaces desarrolladas para distintos componentes o protocolos descritos en el punto anterior. Estos ejemplos se encuentran disponibles en el repositorio de GitHub: https://github.com/dao703/Robot_Car_UC/tree/master/Examples_UC. Para el desarrollo de estos ejemplos se han consultado numerosas fuentes apoyándonos sobre todo en el manual de referencia del microcontrolador, manuales de referencia de los dispositivos que intervienen, ejemplos de utilización de dichos dispositivos en otros lenguajes y plataformas, así como webs con proyectos Ada para otros microcontroladores como https://www.hackster.io

5.1. PRUEBAS SIMPLES DE LAS INTERFACES DESARROLLADAS

Interfaz ADC: main_adc_example
Este sencillo ejemplo buscar comprobar el correcto funcionamiento del convertidor analógico digital. Para ello, se hace uso de uno de los ADC disponibles en la placa con una configuración predeterminada según se describió en las características de la interfaz en el anterior apartado. Consta de una configuración previa del ADC 1 canal 6 – pin A0 de la figura 1- y un bucle infinito de lectura de la tensión de entrada.

Muestra en pantalla de forma periódica un valor decimal referente a la tensión leída por el ADC en el pin PA6. Este valor se refresca cada 0.3 segundos y varía entre 0 para una tensión de 0V y 4095 para una tensión de 3.3V.

Interfaz de entrada/salida digital: Main_Digital_IO_Example
El objetivo de este programa es verificar que los pines GPIO esta configurados correctamente como entrada o salida digital y utilizarse según corresponda. Para ello, se utiliza la interfaz STM32f7_Disco-Digital_io que permite configurar un pin como entrada o salida digital con el procedimiento *configure_pin.*

De manera preliminar, configura un pin como entrada digital en el pin D0 –ver figura 1- y dos pines como salida digital en PC6 y PJ1. Los pines de salida alimentan de forma independiente dos leds, mientras que el pin configurado como entrada tiene conectado un pulsador. Los leds cambian de estado - encendido/apagado - en función de si el pulsador esta pulsado o no.

Interfaz de Ethernet: Main_ethernet_[transmitter/receiver] example
En este apartado ha sido necesario desarrollar dos ejemplos con el objetivo de poder verificar con una placa, que los paquetes se están enviado y recibiendo en el otro extremo. Una peculiaridad de estos microcontroladores es que las direcciones MAC pueden ser configurables por el usuario y asignarlas libremente e incluso modificarse en tiempo de ejecución. Esto es debido a que se trata de dispositivos de desarrollo y no de productos finales. Para realizar esta comunicación lo primero que hemos hecho ha sido definir el tipo de tramas que se van a transferir, en total 64 bytes, pues es el tamaño mínimo de un paquete ethernet, repartidos de la siguiente manera:

- 6 bytes para la MAC de origen.
- 6 bytes para la MAC de destino.
- 2 bytes para el protocolo.
- 50 bytes para el mensaje que se desean transmitir.

Por una parte, tenemos el microcontrolador encargado de la transmisión de paquetes, estos paquetes contienen la dirección MAC de origen, la dirección MAC de destino, el tipo de protocolo ethernet utilizado y un mensaje de 50 caracteres que va variando en el tiempo. Por otra parte, en el otro microcontrolador tenemos la aplicación encargada de recibir los paquetes. Cada paquete que recibe incrementa su contador de paquetes recibidos y muestra en pantalla el contenido del mensaje. En ambos programas se inicializa el controlador de ethernet indicando cual es la MAC de origen.

Interfaz de I2C: Main_I2C_Example

Este apartado tiene como objetivo proporcionar un ejemplo de utilización del protocolo I^2C para la lectura de un valor proporcionado por una brújula digital. Esta brújula se comunica con el microcontrolador a través de dicho protocolo.

Gracias a la interfaz I^2C desarrollada, utilizar un dispositivo externo que se comunica a con el microcontrolador través de este protocolo, es muy sencillo. Tras la configuración, se procede a realizar la primera lectura de información. Para solicitar un valor a la brújula es necesario enviarla un valor que represente el tipo de acción que se espera de ella. Los dispositivos I^2C puede recibir distintos mensajes en función de si se quiere cambiar si dirección de acceso, solicitar una lectura, u otras configuraciones. Para la comunicación de mensajes con la brújula, se hace uso de dos variables en las que se deposita el valor a enviar o recibir según corresponda.

En este caso los datos que recibimos de la brújula y requieren de un procesamiento previo para que un usuario sepa lo que quiere decir esta información. Este programa cuenta con una fase de inicialización de la brújula y un bucle infinito en el que se hacen peticiones constantes de la orientación cada medio segundo, mostrando en la pantalla la información obtenida. Esta información varía entre los 0º y los 359º, que representa el ángulo a girar la brújula hasta orientarla al norte.

Interfaz para motores Paso a Paso: Main_Stepper_Motor_Example.adb

El objetivo de esta sección es comprobar el funcionamiento de un motor paso a paso haciendo uso de la interfaz desarrollada para este tipo de motores. El motor que hemos utilizado cuenta con una pequeña placa controladora ULN2003 basada en un conjunto de 7 transistores en configuración Darlintong. Esta configuración se basa en conectar dos transistores bipolares en cascada obteniendo una ganancia muy grande de cada uno de los transistores. Con ello se consigue controlar cargas de cierta potencia con tensiones de alimentación muy pequeñas.

Consta de una asignación previa de los pines de los que hace uso el motor y el sentido de giro inicial. A continuación, en un bucle infinito se hace uso de los métodos de la interfaz:

Change_Rotation, Move_One_Step y Move_N_Step para mover el motor una serie de pasos en el sentido indicado.

5.2. ROBOT CONTROLADO DE FORMA REMOTA

Para culminar este trabajo e integrar cada interfaz generada en un único demostrador se ha desarrollado un robot controlado de forma remota por un usuario. Este robot consta de dos microcontroladores STM32F769, uno localizado encima del robot y otro en el lado del usuario, ambos unidos por un cable ethernet. En el lado del usuario se disponen los controles para el movimiento del robot junto con una brújula que indica la orientación del norte. En el lado del robot, está el microcontrolador encargado de recibir los movimientos a realizar ordenados por el usuario. Además, cuenta con un sensor de obstáculos del que recive información y tras procesarla se envía al usuario para que pueda visualizarla en su pantalla. Con el fin de facilitar al lector la organización de todos estos elementos, la siguiente figura muestra el sistema en funcionamiento.

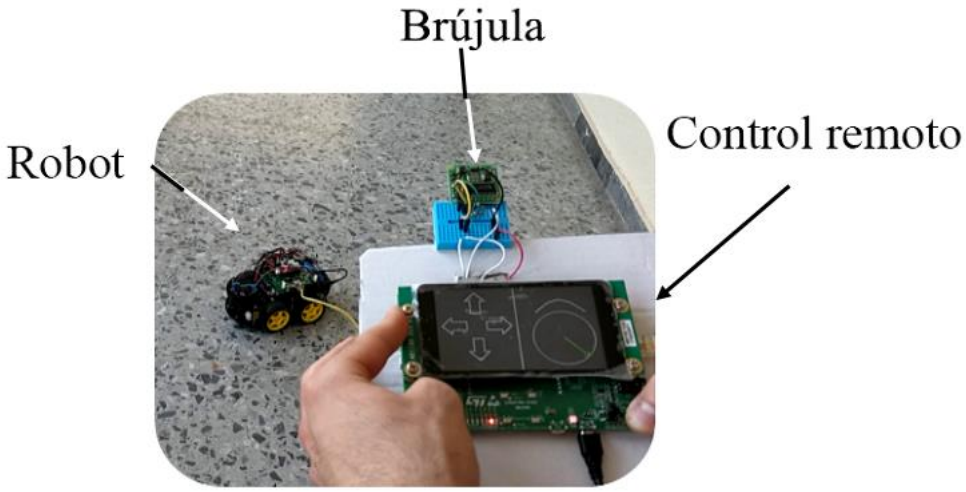

Figura 14 - Robot controlado de forma remota

5.2.1. Características del lenguaje Ada utilizadas en el demostrador

Para este proyecto se ha utilizado el perfil de Ada Ravenscar SFP puesto que, como se explicó en la sección 3.8., se ajusta muy bien a las características necesarias. Este perfil solo permite establecer una política de planificación de tareas FIFO – *first in, first out* – con prioridades, es decir, la primera que llega es la primera que puede ejecutarse, mientras que no llegue ninguna tarea con más prioridad. En cuanto a los periodos de las tareas, el perfil Ravenscar permite definir el momento de la siguiente ejecución tomando como referencia absoluta el reloj del sistema.

Otra característica muy interesante es que las tareas deben tener una prioridad fija, lo cual evita que podamos cometer fallos en el código permitiendo que las tareas puedan cambiar de prioridad si se producen *deadlocks*[10]. Además, establece que las tareas y los objetos protegidos que intervienen se definan en paquetes independientes.

[10] Deadlock es una situación en la que dos programas de ordenador que comparten el mismo recurso impiden de manera efectiva el acceso al recurso, lo que hace que ambos programas dejen de funcionar. También conocido como bloqueo mutuo. [17]

Un objeto protegido consiste en una entidad de datos encapsulados del que se proporciona acceso a ellos solo a través de subprogramas protegidos. Ada garantiza que estas entradas y subprogramas se ejecutaran bajo exclusión mutua, garantizando así que sólo una tarea acceda al recurso en un momento dado. En el demostrador se ha utilizado un objeto protegido llamado Movimiento para leer y escribir el siguiente movimiento a ejecutar por el robot desde distintas tareas.

El sistema de planificación de tareas de Ada para este proyecto es expulsor de prioridades fijas. A medida que las tareas van llegando se encolan en una pila FIFO. El planificador es la parte del software que se encarga de organizar las tareas y decide cual se ejecuta en un momento determinado. Como criterio de elección de la tarea elige la más prioritaria de todas las activas. Una tarea puede ser expulsada si mientras se está ejecutando, otra tarea se activa y tiene prioridad más alta.

5.2.2. Estructura de comunicación de las partes del robot:

En ambos nodos se dispone de un microcontrolador STM32f769 conectados entre sí mediante un cable Ethernet

En cuanto a la comunicación, se han definidos dos estructuras de paquetes que se utilizaran en la comunicación entre microcontroladores. El microcontrolador con el control remoto tiene la dirección MAC: 0:51:255:5:5:0 mientras que el control de abordo tiene la 0:51:255:5:5:1, ambos con protocolo de comunicación: 1010. La asignación de estas direcciones no sigue ningún estándar global puesto que la aplicación se va a probar en una red local controlada. Es decir, este proyecto ha sido probado de manera local en una red privada, y por ello tanto el protocolo de comunicación como las direcciones MAC de los dispositivos no hacen referencia a un estándar global para redes públicas.

Las siguientes figuras muestran el contenido de los paquetes de 64 bytes intercambiados por los dos microcontroladores, junto con la distribución de la información. En primer lugar, está el paquete enviado desde el nodo 2 al nodo 1 y en segundo lugar la respuesta enviada al nodo 2.

MAC Destino	MAC Origen	Protocolo	Obstáculos por la izq.	Obstáculos por la der.	Padding
0 ... 5	6 ... 11	12… 13	14 … 28	29 … 43	44… 63

Esta trama incluye un mensaje de 50 caracteres con la codificación de la distancia a posibles obstáculos. Para su codificación se ha asignado un valor que va desde 1 hasta 4 en función de si el objeto está cerca o lejos, respectivamente. Además, se añade un relleno con 0 hasta completar los 64 bytes.

Esta segunda trama se envía desde el nodo 1 al 2 e incluye un mensaje con el tipo de acción a realizar por el robot. Se incluirá en el mensaje un número diferente en función de si se desea que el robot avance hacia adelante, derecha, atrás o izquierda respectivamente.

MAC Destino	MAC Origen	Protocolo	Movimiento por realizar
0 … 5	6 … 11	12 … 13	14 … 63

5.2.3. Nodo1: Control remoto

Este nodo es el encargado de generar y enviar movimientos, recibir información del entorno del robot - distancia a los objetos próximos- y presentar los resultados en pantalla. Entre los dispositivos que utiliza el nodo 1, se encuentran la pantalla LCD para representar la información de interés para el usuario, el conector RJ45 para la comunicación por Ethernet con el otro microcontrolador y los pines referentes a la comunicación I^2C para comunicarnos con una brújula.

El arranque de este nodo debe de ser controlado de forma que, por ejemplo, no se intente escribir en la pantalla antes de que se inicialice. Para conseguir esto las tareas no se activan hasta que no se han configurado todos los dispositivos.

En el nodo del control remoto encontramos cuatro tareas cuya prioridad se ha definido utilizando el algoritmo RMA -Rate Monotonic Algoritm- para las tareas periódicas. Este algoritmo asigna prioridades de manera inversamente proporcional a su periodo.

A continuación, se exponen las cuatro posibles tareas acompañadas de unos resúmenes esquemáticos con la funcionalidad de cada tarea.

- Transmitter_Task: Tarea encargada de transmitir los paquetes Ethernet con el siguiente movimiento a realizar por el robot. Esta tarea hace uso del objeto protegido Movimiento. En la figura 15 se detalla su funcionalidad.

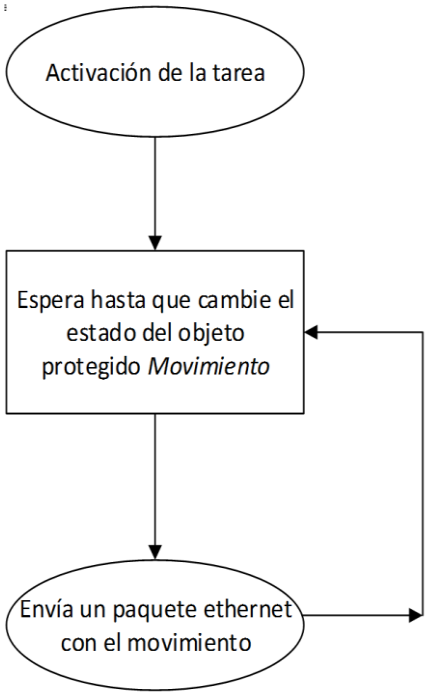

Figura 15 - Trasmitter_Task

- Receiver_Task: Tarea encargada de recoger y procesar los paquetes enviados desde el robot al control remoto. La información recibida del nodo robot consta de la distancia a los obstáculos tanto en el lado izquierdo como en el derecho. En la figura 16 se detalla su funcionalidad.

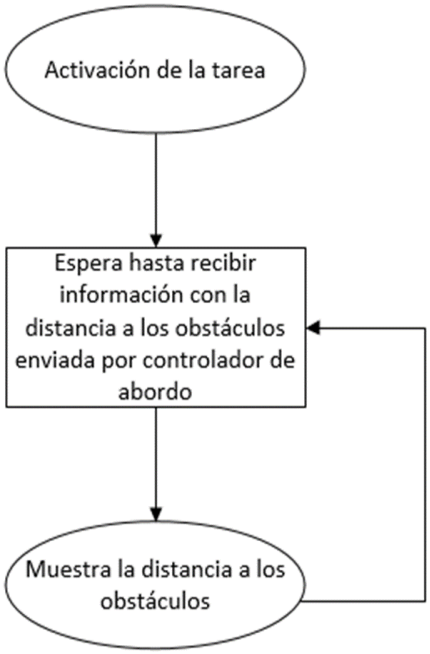

Figura 16 - Receiver_Task

- Check_Display: Tarea periódica encargada de refrescar la pantalla mostrando la información más actual enviada por el propio robot. Esta tarea se ejecuta con un periodo de 0.1 segundos. En la figura 17 se detalla su funcionalidad.

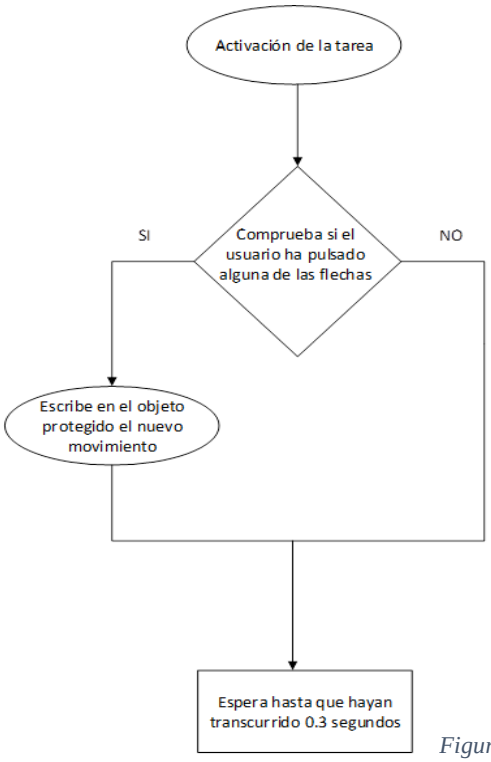

Figura 17 - Check_Display

- Orientation_Task: Tarea periódica encargada de comprobar la dirección del norte consultando la información proporcionada por la brújula y mostrarlo por pantalla. Tiene un periodo de 0.3 segundos. En la figura 18 se detalla su funcionalidad.

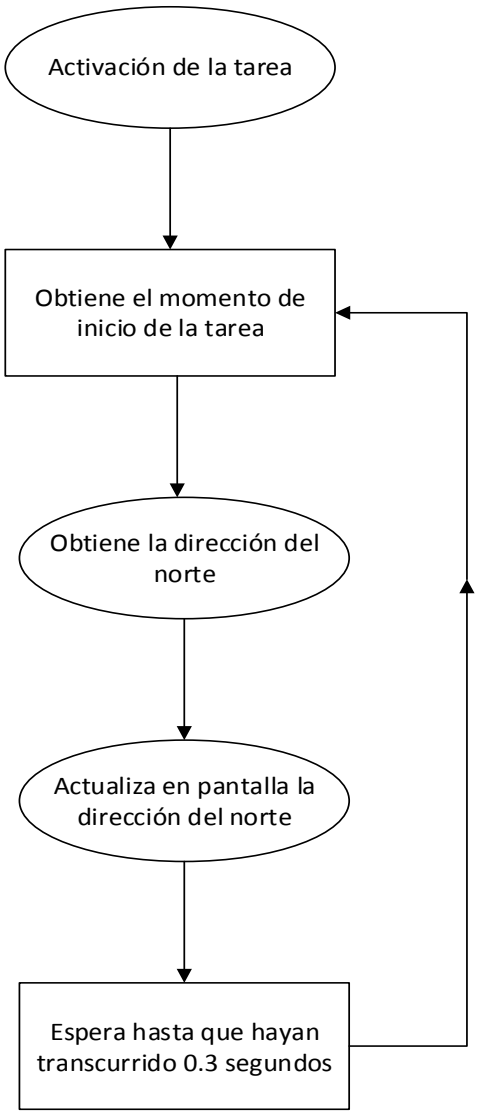

Figura 18 - Orientation_Task

Todas estas tareas interaccionan de forma directa o indirecta con el Human-Machine Interface (HMI)[11] mostrado en la pantalla del microcontrolador:

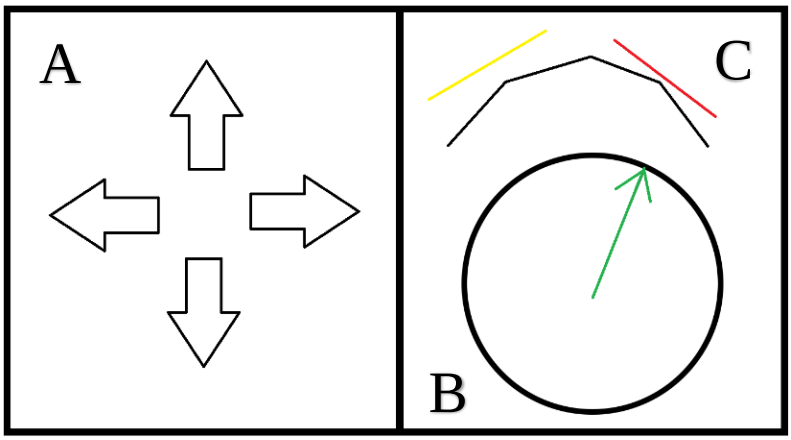

Figura 19 - Interacción global en la pantalla del STM32

Las flechas de la figura 19-A hacen referencia a los posibles movimientos del robot: adelante, atrás, izquierda, derecha. Esta es la parte de la pantalla que revive información del movimiento a realizar por parte del robot.

La parte B de la figura 19 representa la orientación del control remoto hacia el norte. Esta parte del HMI proporciona al usuario de hacia dónde dirigir el robot. La flecha verde se actualiza de forma automática pudiendo cambiar de posición en función de la orientación: Norte, Noreste, Este, Sureste, Sur, suroeste, oeste y noroeste.

La parte C de la figura 19 muestra la distancia que hay por la izquierda y por la derecha hacia posibles objetos, de forma independiente. Para dar una idea al usuario de lo lejos o cerca que están los obstáculos se ha utilizado una codificación de colores: Sin obstáculos a la vista (sin línea de información), obstáculos entre 12cm y 9cm (línea amarilla), obstáculos entre 9cm y 5cm (línea naranja), obstáculos a menos de 5 cm (línea roja).

5.2.4. Nodo 2: Control de abordo

Este nodo contiene la parte que se encarga de la recepción de movimientos, captación de información del entorno y envió de la información. Se hace uso de distintos pines digitales y analógicos para activar los motores de los que dispone el robot en función del movimiento, así como la lectura analógica del sensor de proximidad y el servomotor orienta al sensor.

Este nodo contiene solamente una tarea, onboard_task. Esta tarea es la encargada de realizar las funciones de recepción de paquetes de la parte del control remoto. Dependiendo de la información recibida, esta tarea puede activar los encargados del movimiento, mover el servomotor para orientar el sensor encargado de determina la distancia a los objetos de alrededor y envío de paquetes con toda la información recogida.

[11] Human Machine Interface (HMI) o Interfaz hombre-máquina es la interfaz de usuario en un sistema de fabricación o control de procesos. Proporciona una visualización basada en gráficos de un sistema de control y monitoreo industrial. [25]

Antes de que la tarea comience se llevan a cabo las configuraciones de todos los dispositivos que se conectan: motores de corriente continua con control de velocidad por ancho de pulsos, servomotor para orientar el señor de distancia, sensor analógico para medida de la distancia y el controlador de ethernet para el envío de paquetes.

Tras realizar estas configuraciones previas se da paso a la tarea cíclica que se repite constantemente.

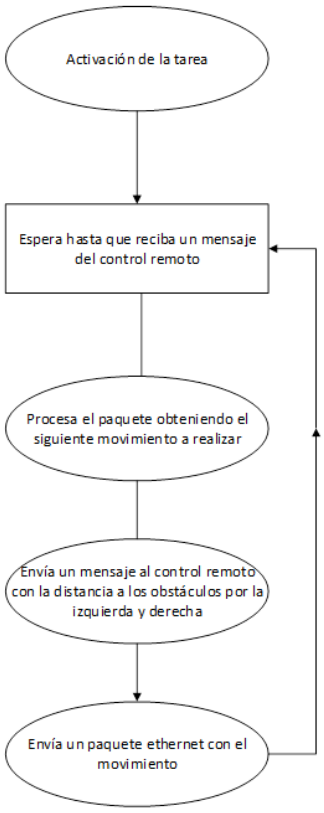

Figura 20 - Onboard_Task

En la figura 20 se muestra la funcionalidad del algoritmo que realiza en cada iteración la tarea OnBoard_Task, la única presente en el nodo de control de abordo.

Para controlar el servomotor por PWM (*Pulse Width Modulation*) es necesario crear una secuencia de pulsos eléctricos de duración variable que determinan el ángulo en el que se posicionara el servo. El motor que hemos utilizado necesita ciclos de 20 ms a una frecuencia de 50 Hz. En función del tiempo que este activo el flanco en este periodo se obtiene la posición del eje del motor, como se muestra en la figura 21.

Para limitar la velocidad de giro de los motores encargados del movimiento se ha desarrollado una estructura que activa y desactiva la entrada del motor simulando un controlador PWM.

```
151          for I  in 1 .. 50 loop
152             STM32F7_Disco.Digital_IO.Digital_Write (Pin   => STM32F7_Disco.Digital_IO.Pin_A1, Value => True);
153             delay until Ada.Real_Time.Clock + To_Time_Span (0.005_0);
154             STM32F7_Disco.Digital_IO.Digital_Write (Pin   => STM32F7_Disco.Digital_IO.Pin_A1, Value => False);
155             delay until Ada.Real_Time.Clock + To_Time_Span (0.015_0);
156          end loop;
```

Figura 21- Control de velocidad de los motores de avance

6. CONCLUSIONES Y TRABAJO FUTURO

6.1. CONCLUSIONES

Este proyecto ha tenido como objetivo asentar unas bases que sirvan para el desarrollo de proyectos software/hardware con el lenguaje de programación Ada y el IDE GPS junto con un cargador y debugger estable.

Este tipo de microcontroladores son cada vez más utilizados tanto en empresas como en multitud de universidades. Por un lado, las universidades los utilizan para introducir a los alumnos en el mundo de la programación. Mientras que en las empresas son muy utilizados como elementos de prototipo y pruebas de proyectos de gran envergadura.

A lo largo de todo este tiempo nos hemos encontrado con multitud de situaciones difíciles, puesto que la interacción con el hardware siempre añade complejidad a los desarrollos. En estos casos, el uso del depurador cruzado ha ayudado a acotar los errores y conseguir solucionarlo.

Otro de los mayores retos de esta propuesta ha sido la escasez de ejemplos que existen sobre estas placas usando el lenguaje Ada y sus correspondientes librerías. Mediante horas de pruebas y trabajo, hemos conseguido finalmente alcanzar los resultados esperados.

La principal contribución de este proyecto reside en el desarrollo de una librería que permite el manejo de distintos dispositivos hardware en microcontroladores STM32. Entre otros, se han integrado dispositivos de entrada y salida digital, entrada analógica, motores paso a paso, servomotores, pantalla táctil y dispositivos que hacen uso del protocolo de comunicación I^2C.

Por otro lado, también se han conseguido integrar el uso de la red de comunicaciones que ha permitido que varios microcontroladores puedan intercambiar mensajes a nivel Ethernet.

Finalmente se ha desarrollado un proyecto de robot compuesto por dos microcontroladores donde se ha puesto en práctica todo lo aprendido.

6.2. OBSERVACIONES Y LÍNEAS DE TRABAJO FUTURO

Exponemos, asimismo, algunas propuestas para la mejora y expansión de este Trabajo, con el objetivo de que futuras investigaciones, relacionadas con este campo de estudio, puedan continuarlo.

Existen multitud de posibilidades muy interesantes bajo nuestro punto de vista:

- La comunicación mediante un cable ethernet brinda la posibilidad de proyectos conectados con el mundo real, es decir, IoT. Mejorar las librerías de red e integrar los microcontroladores en un sistema de adquisición de información del entorno con multitud de sensores y todo ello con un coste muy bajo.

- La investigación del resto de características de la placa STM32F769 como el micrófono, librerías para el control de dispositivos mediante el ancho de pulsos o las entradas y salidas de audio. El desarrollo de videojuegos utilizando la pantalla táctil (LCD) es también un gran reto.

- El desarrollo de sistemas SCADAS de utilidad en procesos de fabricación con el fin de poder utilizar este tipo de placas para sustituir a computadores más caros y programas más costosos. Para ello sería muy interesante continuar con el desarrollo de elementos visuales. Estos podrían ser utilizados dentro de un mapa del proceso de producción de una industria. Su integración en la industria 4.0 podría ser un de gran utilidad aprovechando su bajo coste y su gran potencialidad.

7. REFERENCIAS

[1] AdaCore info: https://learn.adacore.com/about.html

[2] ADC info: https://www.digikey.es/es/articles/techzone/2017/sep/adc-dac-tutorial

[3] Analog to Digital Converter (ADC) and Digital to Analog Converter (DAC), definición.http://macao.communications.museum/eng/exhibition/secondfloor/MoreInfo/ADConverter.html

[4] ARM Cortex-A, official website https://www.arm.com/products/silicon-ip-cpu

[5] ARM Cortex-M, official website. https://developer.arm.com/ip-products/processors/cortex-m

[6] ARM Cortex-R, official website. https://developer.arm.com/ip-products/processors/cortex-r

[7] ARM Holdings. Official website. https://www.arm.com/

[8] ARM processors, official website. https://developer.arm.com/ip-products/processors

[9] ARP (Address resolution protocol) Universidad de Aberdeen. Archivado desde el original el 20 de junio de 1997. https://web.archive.org/web/19970620095137/https://erg.abdn.ac.uk/users/gorry/course/inet-pages/arp.html

[10] Barnes, John. *Programming in Ada 2012*. Cambridge University Press. ISBN 978-1-107-42481-4 (p.11-12)

[11] Burns, A. *The Ravenscar Profile for High Integrity RealTime Programs,* Reliable Software Technologies - Ada Europe '98, Springer-Verlag Lecture Notes in Computer Science, Número 1411

[12] Burns, A. *The Ravenscar Profile*. ACM SIGAda Ada Letters. XIX (4): 49–52. doi:10.1145/340396.340450

[13] Compilación cruzada, definición http://linuxemb.wikidot.com/tesis-c3

[14] Compilation, definición https://www.computerhope.com/jargon/c/compilat.htm

[15] Cross- compilation in an embedded system https://www.quora.com/What-is-meant-by-cross-compilation-in-an-embedded-system

[16] Cross-compilation, definición. GNU official website. https://www.gnu.org/savannah-checkouts/gnu/automake/manual/html_node/Cross_002dCompilation.html

[17] Deadlock, definición. Coulouris, George (2012). Distributed Systems Concepts and Design. Pearson. p. 716. ISBN 978-0-273-76059-7.

[18] DHCP (Dynamic Host Configuration Protocol), definición. TechTarget Network: https://searchnetworking.techtarget.com/definition/DHCP

[19] DNS (Domain Name Systems), definición. *What is DNS?* https://www.cloudflare.com/learning/dns/what-is-dns/

[20] GNAT (GNU NYU Ada Translator), definición. Dong, Jielin. *Network Dictionary (2007).*, Javvin Technologies Inc. ISBN 1602670005 9781602670006

[21] GNAT Pro User's Guide Supplement for High-Integrity Edition Platforms *The GNAT Ada Compiler GNAT GPL* Edition, Version 2012 Document revision level 246224.

[22] GPIOs (General Purpose Input/Output). Oracle® Java ME Embedded Developer's Guide (8 ed.). Oracle Corporation. 2014.

[23] GPL General Public License, definición. Dong, Jielin. *Network Dictionary (2007).*, Javvin Technologies Inc. ISBN 1602670005 9781602670006

[24] GPS (GNAT Programming Studio), definición. Dong, Jielin. *Network Dictionary (2007).*, Javvin Technologies Inc. ISBN 1602670005 9781602670006

[25] HMI (Human-Machine Interface), definición https://www.pcmag.com/encyclopedia/term/44300/hmi

[26] I2S (Inter-IC Sound). Arduino official website https://www.arduino.cc/en/Reference/I2S

[27] IDE, Entorno de Desarrollo: http://www.fdi.ucm.es/profesor/mendias/PSyD/docs/PSyDtema3.pdf

[28] IPv4 (Internet Protocol version 4) https://www.iana.org/numbers

[29] LCD (Liquid Crystal Display) https://whatis.techtarget.com/definition/LCD-liquid-crystal-display

[30] Microcontrolador, definición. https://www.merriam-webster.com/dictionary/microcontroller#h1

[31] Microcontroller, invention history. http://www.circuitstoday.com/microcontroller-invention-history

[32] Motor paso a paso: https://www.luisllamas.es/motor-paso-paso-28byj-48-arduino-driver-uln2003/

[33] OpenOCD info: http://openocd.org/doc/html/index.html

[34] Orígenes e historia del compilador GNAT, artículo de Novática nº 126. https://web.archive.org/web/20051028191254/http://webs.uvigo.es/h06/weba573/persoal/jmv/material/LRL.pdf

[35] Paret, Dominique. *The I2C Bus: From Theory to Practice*. (1997) ISBN 978-0-471-96268-7.

[36] Perfil Ravenscar, Uppsala Universitet http://www.it.uu.se/edu/course/homepage/realtid/2017-368/labs/lab2/ravenscar

[37] Programa compilador: http://www.ictea.com/cs/index.php?rp=/knowledgebase/8817/iQue-es-un-programa-compilador.html

[38] PWM (Pulse Width Modulation). Arduino official website. https://www.arduino.cc/en/tutorial/PWM

[39] SPI (Serial Peripheral Interface) https://www.mct.net/faq/spi.html

[40] ST Microelectronics. Página Web oficial: https://www.st.com/content/st_com/en.html

[41] Stepper motor: https://dronebotworkshop.com/stepper-motors-with-arduino/

[42] STM32CubeMX, definición. https://www.st.com/en/development-tools/stm32cubemx.html

[43] STM32F407, key features official web ST https://www.st.com/en/evaluation-tools/stm32f4discovery.html

[44] STM32F769, key features official web ST https://www.st.com/en/evaluation-tools/32f769idiscovery.html

[45] UDP (User Datagram Protocol), definición. Kurose, J. F.; Ross, K. W. (2010). *Computer Networking: A Top-Down Approach* (5th ed.). Boston, MA: Pearson Education. ISBN 978-0-13-136548-3.