Sistema de adquisición de datos portátil para la generación de nubes de puntos 3D georeferenciadas a partir de un sensor LIDAR 2D

(Parte I, Estudio y Comunicación)

GEEiA

Autor: Alejandro Solans Barón

Tutor: Alexandre Escolà – Jordi Palacín

Septiembre 2015

Índice general

Índice de figuras

Índice de tablas

Agradecimientos

Detrás de este trabajo hay personas sin las cuales la realización de este proyecto no hubiese sido posible. Por esto me gustaría dedicarles unas líneas con las cuales hacerles llegar mi más sincero agradecimiento.

En primer lugar agradecer a mis dos tutores de proyecto su confianza depositada en mí, en primer lugar a Jordi Palacín, ya que gracias a él conocí esta oportunidad y siempre me ha guiado y ha guiado mis pensamientos hacia la "cordura". Y en segundo lugar y no menos importante, a Àlex Escolà y todo su equipo de trabajo, en primer lugar por todo el soporte brindado y en segundo lugar e insistiendo en ello, por toda la confianza que he sentido hacia mi persona, sin ella este proyecto no hubiese llegado nunca a buen puerto.

Por otra parte, quisiera hacer una mención muy importante a las personas que sin ninguna obligación y sin pedir nada a cambio me han ofrecido su ayuda. A Isaac García por sus espléndidas explicaciones respecto a los ángulos de Euler y a Adolf por sus inacabables conocimientos de los diferentes dispositivos electrónicos.

También a Francisco Clariá por ser mi mentor durante todo el tiempo, por todo lo que me ha enseñado y por ponerse siempre en mi lugar y preocuparse por el estado del proyecto en todo momento, sus palabras siempre han sido una fuente de tranquilidad y serenidad para mí.

Y mi más especial y eterno agradecimiento, no solo por su ayuda en este proyecto, sino porque sin los conocimientos que me ha transmitido y la paciencia que ha tenido siempre no hubiese sido posible ni si quiera plantearse algo así, gracias por todo Marcel Tresanchez, ese todo engloba muchas cosas como para ser enumeradas, eternamente agradecido.

En último lugar, agradecer a mi familia y amigos el apoyo en los momentos de agobio y tensión, por haber entendido siempre mi situación y haberme empujado siempre para dar el siguiente paso.

1. Memoria

1.1. Introducción

El presente proyecto surge de la idea de poder mejorar un sistema para obtener los parámetros geométricos y estructurales (fenotipo) de plantas y árboles en campo, a demanda del Grup de Recerca en AgròTICa i Agricultura de Precisió (GRAP) de la Universitat de Lleida. El GRAP empezó sus trabajos en caracterización electrónica de la vegetación mediante sensores LIDAR (light detection and ranging) el año 2002 y actualmente dispone de un escáner láser terrestre móvil de diseño propio basado en un sensor LIDAR 2D, un sistema de posicionamiento Real-Time Kinematics (RTK) y un sistema de adquisición de datos basado en un programa desarrollado en LabVIEW (National Instruments) ejecutado en ordenador portátil de campo.

Con el sistema actual se consiguen nubes de puntos 3D, pero se tienen algunos inconvenientes, los cuales se quieren solventar en este nuevo proyecto, para lo cual se decide añadir una Unidad de Medida Inercial (IMU), que nos servirá para conocer en todo momento las inclinaciones en los 3 ejes del sensor. Además, se pretende cambiar el sistema de adquisición de datos, transportando este trabajo a un microprocesador de gama alta para desarrollar un escáner láser portátil.

La implementación de este proyecto en su totalidad se calcula para un plazo de un año y medio (comenzando en Marzo de 2015), tiempo en el cual se prevé que se tengan todas sus partes implementadas y funcionando conjuntamente. Una vez llegado a este punto, será motivo de estudio la posible mejora de la precisión en los datos y llevar a cabo ésta si procede.

De cara a la presentación de este proyecto como proyecto final de grado, se habrá invertido un tiempo total de 4 meses trabajando en dicho proyecto, y se espera haber conseguido la comunicación con los 3 sensores y el guardado de datos de estos en una memoria SD. Por lo tanto, lo presentado será un estudio del microprocesador y sensores utilizados, el estudio teórico de los algoritmos de corrección de posición, y las librerías utilizadas para la comunicación SSNG, IMU, LIDAR y memoria SD.

1.2. Antecedentes

El Grupo de Investigación en AgróTICa y Agricultura de Precisión (GRAP) nació en 2004 aglutinando investigadores de dos organismos: la Universitat de Lleida (UdL) y el Centro de Mecanización Agraria del Departamento de Agricultura, Ganadería, Pesca, Alimentación y Medio Natural de la Generalitat de Catalunya. Sin embargo, los investigadores integrantes llevan más de 15 años trabajando conjuntamente y constituyen uno de los grupos de referencia en Tecnología de Aplicaciones Fitosanitarias a nivel de todo el Estado español, con varias patentes y modelos de utilidad en este ámbito. El grupo también es pionero en el ámbito de los equipos robotizados e inteligentes para la ganadería de precisión, con una de las primeras patentes en este ámbito y equipos instalados en Canadá.

AgróTICa es un término que hace referencia a la aplicación de las Tecnologías de la Información y la Comunicación (TIC) en la agricultura, en sentido amplio. Las aplicaciones de las TIC pueden ser muchas y diversas. Así, las TIC pueden utilizarse para adquirir datos de sistemas agrícolas o ganaderos, convertir los datos en información útil para los agricultores y/o ganaderos y comunicar y/o compartir esta información mediante sistemas de comunicación.

La Agricultura de Precisión consiste en realizar las operaciones agrícolas adecuadas, de la manera adecuada, en el momento adecuado, en el lugar adecuado y de la manera adecuada (adaptado de las definiciones de Pierre Robert y Raj Khosla). Para poder llevar a cabo esta agricultura es necesario conocer el comportamiento de los campos y sus peculiaridades para obtener de él máximo provecho pero de forma eficiente y sostenible. Actualmente, los avances tecnológicos y las TIC permiten recopilar muchos datos sobre el cultivo y su medio con una altísima resolución espacial (muchos datos por metro cuadrado) y con un coste razonable. Visualizar estos datos, procesarlos y convertirlos en información útil, permite que los agricultores dispongan de un apoyo objetivo y fiable para poder tomar decisiones de manera coherente.

La práctica de la Agricultura de Precisión se puede llevar a cabo según **dos grandes metodologías**:

- **Agricultura de Precisión basada en mapas digitales de información**:

Antes de cualquier operación agrícola es necesario tomar datos de la parcela, analizar su variabilidad espacial y mapearlos (crear un mapa de la parcela con la distribución espacial de la

variable medida). Después de analizar los mapas obtenidos debe tomarse una decisión de manejo. El resultado de la toma de decisión será un nuevo mapa, denominado mapa de actuación o de prescripción, donde se muestra qué debe hacerse en cada punto de la parcela (intensidad de la operación o dosis de producto a aplicar). Para practicar este tipo de agricultura es indispensable disponer de un sistema de posicionamiento y navegación (vulgarmente llamado "un GPS") más o menos preciso tanto para la toma de datos como para la actuación.

- **Agricultura de Precisión basada en sensores y en tiempo real:**

Este tipo de agricultura estrictamente no requiere sistemas de posicionamiento y navegación puesto que la toma de datos, la decisión y la actuación se llevan a cabo en tiempo real mientras el tractor y el equipo se van desplazando por la parcela. Dado que la actuación va a seguir siendo variable, el equipo también debe ir equipado con tecnología de actuación variable. La diferencia radica en que la actuación no se basa en un mapa de prescripción sino en uno o varios sensores que van tomando datos "sobre la marcha".

Basándose en el primer tipo de Agricultura de Precisión, el GRAP, con la idea de obtener modelos 3D y mapas de la vegetación agrícola, se embarcó en un proyecto que consistía en desarrollar un escáner láser terrestre móvil (MTLS) basado en un sensor LIDAR 2D y un receptor RTK, adquiriendo estos datos a través de un ordenador. A este proyecto se le añadió una plataforma móvil basada en orugas y un sistema para mantener su horizontalidad basado en un sistema de realimentación de primer orden.

Éste sistema ya se ha puesto en marcha y con él se han conseguido construir diferentes modelos y mapas ( ) pero el objetivo actual es construir un sistema portátil a la vez que mejoramos la precisión y funcionalidad de éste.

Los aspectos que podrían ser mejorados, y por lo tanto, sometidos a estudio:

- El sistema para mantener la horizontalidad no funciona correctamente, se deduce que puede ser por la constante vibración de la plataforma en su desplazamiento por el terreno. A esto se le suma el ruido electromagnético producido por los variadores de frecuencia utilizados para controlar la velocidad de los motores que mueven la plataforma móvil.

Esto conlleva errores significativos en las lecturas, ya que hace perder mucha precisión angular.

- Para la toma de muestras se necesita llevar siempre el ordenador portátil ya que el sistema de adquisición de datos está implementado en Labview.
- Para realizar pruebas del sistema va bien tener esta plataforma de orugas, pero en un futuro será un inconveniente tener que depender de ella para las mediciones, lo ideal sería un sistema que se pueda colocar en la parte delantera de un tractor o que una persona lo pueda llevar sin dificultad en la mano.

Una vez estudiados, se barajan diferentes soluciones, y la elegida de cara a solventar estos problemas es una evolución para este proyecto basada en los conocimientos adquiridos sobre microprocesadores y programación en C a lo largo del Grado de Electrónica Industrial y Automática, más precisamente en las asignaturas optativas de Integración de Sistemas. Para el procesado posterior, nos basaremos en los conocimientos en Matlab adquiridos también en éste grado.

Esta evolución trata de:

- Añadir una IMU con la que conocer en todo momento la orientación e inclinación del sensor LIDAR y aplicar con estos datos un algoritmo de corrección en el procesado, con lo que evitaríamos los problemas de precisión causados por el sistema de estabilización de la horizontalidad.
- Realizar el software de adquisición de datos a través de un microprocesador de gama alta. Con esto conseguiríamos evitar la necesidad de llevar siempre el ordenador portátil para la adquisición y a la vez nos da la posibilidad de construir un sistema cómodo y manejable evitando así la plataforma de orugas. El uso energético de este sistema se verá reducido drásticamente.
- Implementar dicha base cómoda y manejable, la cual nos permita llevar el sistema en una mano o colocarla en la parte frontal de un tractor y así evitar la dependencia de la plataforma de orugas.

1.3. Objeto

El Grup de Recerca en AgròTICa i Agricultura de Precisió ha construido un sistema que consta de un escáner láser terrestre móvil (MTLS) basado en un sensor LIDAR 2D y un sistema satelital de navegación global (GNSS) Real-Time Kinematics (RTK), junto con un sistema de adquisición de datos que consiste en un ordenador portátil con un software propio en LabVIEW, para la obtención el fenotipo de la vegetación del campo estudiado. Todo esto está implementado sobre una plataforma de orugas móvil, la cual les permite desplazarse a través de las hileras de árboles a estudiar. Para mantener el sensor horizontal, han implementado un sistema dinámico de estabilización para la de corrección de posición.

El objeto de este proyecto es mejorar los puntos débiles de este sistema y conseguir un sistema portátil más compacto, manejable, ligero, preciso y económico, así como dar un paso más en su funcionalidad.

Por ello, necesitamos saber la viabilidad de solventar los problemas añadiendo al sistema actual un sensor inercial.

También se necesitará saber la viabilidad de transformar completamente el sistema y controlarlo todo a través de un microprocesador de gama alta, lo cual conlleva:

Estudio de la viabilidad de comunicación y adquisición de datos de los dos sensores ya utilizados, el GNSS y el sensor LIDAR, y en caso afirmativo, generar una librería para su utilización.

Estudio de la viabilidad de comunicación y adquisición de datos del nuevo sensor, el inercial, y en caso afirmativo, generar una librería pasa su utilización.

Guardado de los datos relevantes adquiridos de estos tres sensores en una tarjeta de memoria de forma que posteriormente sea cómodo el procesado de estos con Matlab.

1.4. Abasto

Inicialmente se realizará un estudio del microprocesador a utilizar, analizando las características y posibilidades de cada uno de los que se conoce que podrían servir para conseguir nuestra finalidad. Al mismo tiempo se estudiarán individualmente cada uno de los periféricos, sus características, y sus protocolos de comunicación para conocer la compatibilidad entre estos y el microprocesador.

Se partirá de conocimientos previos adquiridos durante el grado para elegir el microprocesador, y de las especificaciones técnicas de los periféricos para conocer cómo comunicarse y capturar la información deseada.

Con los protocolos de cada uno de ellos y las especificaciones del microprocesador elegido, se especificarán los pines utilizados por cada uno de los periféricos y se comenzará entonces a trabajar en las librerías de control para éstos.

Paralelamente, se trabajará en un sistema geométrico teórico en tres dimensiones, para una vez tener disponibles los datos de nuestros sensores, poder preparar a partir de esto los algoritmos pertinentes de corrección con Matlab.

Finalmente, se trabajará en una interface gráfica para visualizar el mapa y poder movernos por él y así estudiar la zona escaneada con sus diferentes finalidades.

El resultado final constará de dos partes diferenciadas. Por una parte el sistema de adquisición de datos de los sensores, incluyendo el guardado de éstos; y por otra una interface gráfica en el ordenador capaz de interpretar, tratar y mostrar visualmente los datos guardados por el microprocesador.

1.5. Requisitos del diseño

1.5.1. Requisitos generales

Por la parte mecánica, se requiere un diseño cómodo, ligero y manejable; a la vez tiene que ser resistente y seguro para nuestros sensores, los cuales tienen que estar bien anclados para evitar vibraciones.

También se necesitará que el acople y desacople a un tractor u otra máquina agraria sea rápido, y que tras el desacople el manejo manual sea instantáneo o lo más sencillo posible.

Por la parte electrónica, se requiere que el sistema sea capaz de captar, procesar y guardar los datos de los 3 sensores a una frecuencia máxima de 40 Hz, ya que es el límite de transmisión de datos impuesto por el sensor LIDAR y éste será el sensor de referencia del que no se quiera perder información.

Por otra parte, se requiere que el sistema sea entendible a la hora de trabajar con él, es decir, conocer en todo momento el estado del sistema; y robusto, es decir, que no pierda datos ni se quede bloqueado.

1.5.2. Sensores a utilizar

Para la realización de este proyecto se parte de 3 sensores ya adquiridos, el primero, el cual nos proporcionará los datos de distancia, será el LIDAR UTM-30LX-EW, de la marca Hokuyo; el segundo, un sistema de medida inercial, será el MTi de Xsens; y el tercero y último de nuestros sensores, el utilizado para la geolocalización, será el receptor Leica GPS 1200+.

La comunicación y características de estos tres sensores a utilizar está explicado más adelante en el apartado *1.7.5 Estudios de datos a captar*.

1.6. Análisis de soluciones

1.6.1. Ordenador

El primer estudio es sobre seguir el proyecto con la plataforma utilizada actualmente, es decir, un ordenador portátil con un software de adquisición y procesado de datos en Labview.

LabVIEW es un entorno de programación destinado al desarrollo de aplicaciones, similar a los sistemas de desarrollo comerciales que utilizan C o BASIC. Sin embargo, LabVIEW se diferencia de dichos programas estos lenguajes de programación se basan en líneas de texto para crear el código fuente del programa, mientras que LabVIEW emplea la programación gráfica o lenguaje G para crear programas basados en diagramas de bloques.

Labview tiene su mayor aplicación en sistemas de medición, como monitoreo de procesos y aplicaciones de control, un ejemplo de esto pueden ser sistemas de monitoreo en transportación, Laboratorios para clases en universidades, procesos de control industrial. Labview es muy utilizado en procesamiento digital de señales (wavelets, FFT, Total Distorsion Harmonic TDH), procesamiento en tiempo real de aplicaciones biomédicas, manipulación de imágenes y audio, automatización, diseño de filtros digitales, generación de señales, entre otras, etc.
El entorno grafico de LabView (Fig. 1), facilita el diseño y programación de la instrumentación virtual.

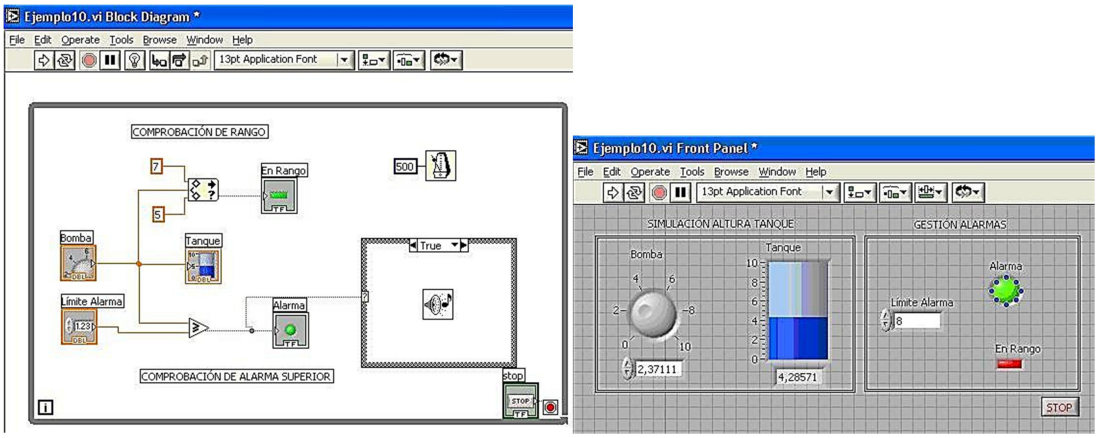


***Figura 1**: Entorno LabView*

El software sería muy cómodo para nuestro propósito a la hora de realizar el procesado en tiempo real, pero trabajar con el conlleva un problema que, a la larga, tendríamos que evitar, y es que éste tan solo puede trabajar con el apoyo de un ordenador.

Es por eso que se decide descartar esta solución y comenzar con el estudio de alternativas.

1.6.2. Microprocesador

En este capítulo se describen 2 tipos de plataformas que lleven a cabo la adquisición de todos los datos y realizar las operaciones pertinentes con éstos sin perder información por falta de velocidad de procesado. Además, tiene que cumplir los siguientes requisitos:

- Dimensiones pequeñas y ligero para poder implementarlo en una estructura ergonómica.
- Precio reducido tanto de la placa como de su herramienta de programación. De este modo si surge cualquier avería en ella, su reemplazo será de bajo coste.
- Lenguaje de programación sencillo y de fácil compilación.
- Cantidad de pines suficiente para poder trabajar sin limitaciones. Interesa poder usar elementos de entrada y salida (E/S), comunicación SPI, I2C, SDIO, ETHERNET o USB entre otros.
- Procesador de altas prestaciones que permita trabajar con elevadas cantidades de datos durante periodos de tiempo ininterrumpidos de entre 2 y 3 horas.

Las opciones planteadas son dos plataformas que ya se han usado previamente en distintas asignaturas del grado, por lo tanto ya se tiene conocimiento previo de su funcionamiento, de su modo de trabajo y de programación.

1.6.2.1. ARDUINO

La primera opción es la plataforma de hardware libre Arduino, se basa en un microprocesador y un entorno de desarrollo que varía según el modelo. Actualmente su uso está muy generalizado y por este motivo se dispone de mucha documentación y librerías gratuitas por la red.

El lenguaje de programación Arduino se basa en C/C++ por lo que es sencillo de programar, además su entorno de desarrollo integrado (IDE) es de código libre y se puede descargar de forma gratuita des de la página web de Arduino. Los proyectos realizados con esta plataforma pueden ejecutarse sin la necesidad de conexión a un ordenador y simplemente deben ser alimentados al voltaje especificado por cada placa.

Su hardware se puede montar a mano o adquirirse ya ensamblado, generalmente se basa en un microcontrolador Atmel AVR -como son el ATmega168 o ATmega328-, pero desde el año 2012 también se utiliza microcontroladores CortexM3 de ARM de 32 bits.

En la tabla 1 se observa una sencilla comparación de las características de algunas placas Arduino más usadas.

__Tabla 1__: Comparativa placas Arduino

Nombre	Uno	Due	Leonardo	Mega2560	Mega ADK
Procesador	ATmega328	**AT91SAM3X8E**	ATmega32u4	ATmega2560	ATmega2560
Voltaje de trabajo [V]	5	**3.3**	5	5	5
Velocidad CPU [MHz]	16	**84**	16	16	16
SRAM [KB]	2	**96**	2.5	8	8
Flash [KB]	32	**512**	32	256	256
USB	Regular	**2 Micro**	Micro	Regular	Regular
UART	1	**4**	1	4	4
E/S analógicas	6/0	**12/2**	12/0	16/0	16/0
E/S digitales	14/6	**54/12**	20/7	54/15	54/15
CAN	0	**1**	0	0	0
SPI	1	**1**	1	1	1
I2C	1	**1**	1	1	1

Debido a la gran diferencia de prestaciones entre la placa Arduino Due y los demás modelos, se analiza dicha opción con más profundidad para compararla más adelante:

ARDUINO DUE

El modelo Arduino Due (Fig. 2) es la primera placa de Arduino basada en un microcontrolador de 32 bits de ARM, el Atmel SAM3X8E ARM-Cortex M3 CPU. Está preparada para que funcione simplemente con conectarla al ordenador con un cable micro-USB o alimentándola con un adaptador de alterna a continua o una batería. A diferencia de otros modelos de placa, el Arduino Due trabaja a 3.3 V y por lo tanto, sus pines podrán tolerar como máximo dicho voltaje.

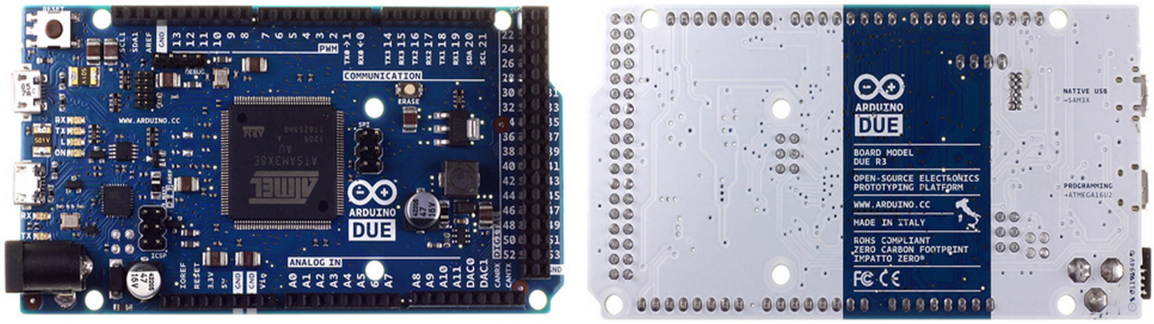


***Figura 2**: Placa Arduino DUE*

El hecho de llevar un núcleo de 32 bits de ARM mejora notablemente su rendimiento en comparación a las típicas placas con microcontroladores de 8 bits. Sus diferencias más significativas se pueden enumerar en 5 puntos:

- Un núcleo de 32 bits permite trabajar con mayores cantidades de bytes en un único ciclo de reloj, 4 bytes en total por ciclo.
- Su velocidad de reloj llega a los 84 MHz en comparación con los 16 y 8 MHz de los otros modelos.
- Su memoria SRAM es de 96 KBytes, lo que permite almacenar y manipular una mayor cantidad de variables de los programas de forma volátil. En los modelos de 8 bits dicha memoria es de 8 KByte como máximo.
- Dispone de 512 KBytes de memoria Flash, la memoria no volátil destinada al almacenamiento del código del programa que se debe ejecutar. De este modo permite usar códigos de programación mayores que en los modelos de 8 bits.
- Dispone de un controlador DMA, el cual permite que ciertos elementos de la placa accedan a la memoria del sistema sin la necesidad de la intervención de la CPU, por lo que ésta se libera de las tareas de escritura en la memoria.

1.6.2.2. Familia STM32

STM32 es una familia de microcontroladores de 32 bits diseñados por STMicroelectronics que se basan en un procesador ARM Cortex-M. Ofrecen un producto de 32 bits que combina alto rendimiento, trabajo en tiempo real, procesado digital de la señal, bajo consumo, bajo nivel de voltaje de trabajo y a su vez manteniendo facilidad en su desarrollo y programación.

Existe de una gran variedad de soluciones de entorno de desarrollo (IDE) para STM32, en general éstas suelen permitir programación tanto en lenguaje C/ C++ como ensamblador. Las herramientas más populares para poder programar, descargar el programa a la memoria Flash y depurar son Atollic TrueStudio, Keil MDK y IAR EWARM.

Los procesadores ARM disponen de distintas opciones configurables y STMicroelectronics es quien elige la configuración deseada para sus propios diseños. Interiormente, cada microcontrolador consiste en un procesador ARM, memoria RAM, memoria Flash e interface de depuración (debug), además ST añade sus propios periféricos al núcleo para así conseguir el producto final.

Existe una gran variedad de productos dentro de la familia STM32 y se clasifican en 4 grupos según el tipo de núcleo Cortex-M que utilizan, como podemos observar en la Figura 3, y su rendimiento también variará según éste.

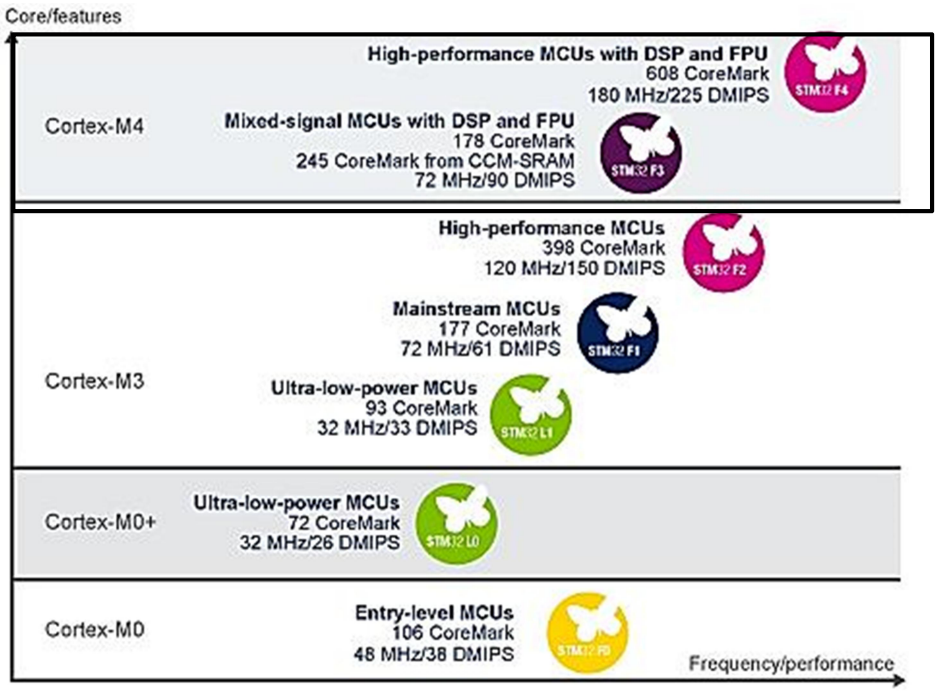


***Figura 3**: Clasificación de los microcontroladores STM32 según su núcleo*

De los mencionados en la figura anterior, la serie STM32F4 es el primer grupo de microcontroladores STM32 que se basan en un núcleo Cortex-M4 y es el que aporta el mayor rendimiento hasta la fecha de la gama de productos STM32. Además, también es la primera serie que dispone de Procesado Digital de la Señal (DSP) y Unidad de Punto Flotante (FPU) y sus características, las cuales podemos ver en la Figura 4, se pueden resumir en los siguientes puntos:

- **Núcleo**: ARM Cortex-M4 con ratios de reloj máximos de 84, 168 y 180 MHz.
- **Memoria:**
  - RAM: 192 KB de propósito general, 64 KB de memoria acoplada al núcleo (CCM) y 84 KB para el soporte de batería.
  - Flash: 512, 1024 o 2048 KB de propósito general, 30 KB del arranque del sistema, 512 bytes para una única programación (OTP) y 16 bytes de opciones.
  - Cada chip dispone de 96 bits para el número de identificación de cada dispositivo.
- **Periféricos**: Dentro de la serie STM32F4 hay un grupo común de periféricos y otro específico de cada modelo como se observa en la figura siguiente.
- **Reloj**: Para generar el reloj deseado, disponen de un oscilador interno (16 o 32 MHz) y otro externo (de 4 a 26 MHz, de 32.768 a 1000 KHz).
- **Voltaje de trabajo**: Trabaja en rangos de 1.8 a 3.6 voltios.

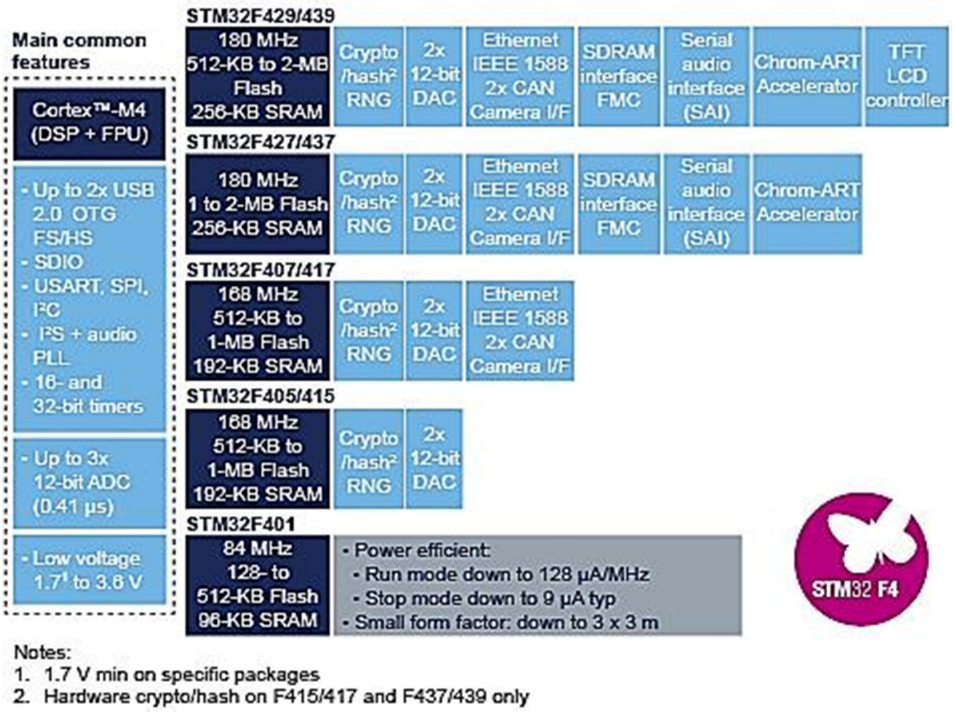


Figura 4: Especificaciones técnicas y periféricos de los modelos STM32F4

Dentro de los grupos de STM32F4 anteriores, se decide analizar el STM32F4-Discovery (STM32F407VZG) por su conocimiento previo y el STM32F429I-DISCO (STM32F429ZIT6) por ser uno de los microcontroladores con mayor rendimiento de la serie.

STM32F429I-DISCO

El STM32F429I-DISCO (Fig. 5) es una de las placas de desarrollo con mayor rendimiento de STMicroelectronics y consta de una amplia variedad de periféricos preparados para ser utilizados sin tener que añadir ninguna modificación de hardware. Se alimenta a través del bus USB o incluso con una alimentación externa de 3V o 5V.

Incluye la herramienta de depuración ST-LINK/V2 destinada a la programación de la placa. Consta de una memoria SDRAM de 64 bits, 2MB de flash i 256KB de RAM. En su diseño se añade una pantalla TFT LCD de 2.4''; leds y botones que se pueden usar para controlar el estado de la ejecución del programa; y un giróscopo de 3 ejes ya preparado para su funcionamiento.

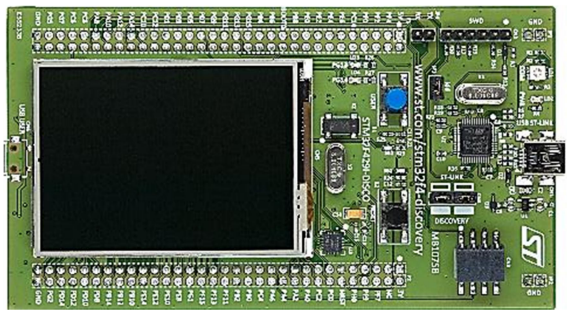


***Figura 5**: Placa STM32F429I-DISCO*

STM32F4Discovery

Por otro lado, la STM32F4Discovery (Fig. 6) tiene un rendimiento ligeramente inferior pero dispone de mayor soporte ya que por su configuración es la placa con mayores ventas de STM. Además, está diseñada para principiantes y se dispone de numerosos ejemplos en la red listos para su funcionamiento.

Igual que el modelo STM32F429, también consta de la herramienta de depuración ST-LINK/V2 para programar y se puede alimentar a través de USB o a partir de una alimentación de 3 o 5V. Su memoria es ligeramente inferior ya que su Flash es de 1MB y su RAM de 192KB. Viene con 4 leds, 2 botones, un acelerómetro o un micrófono entre otros.

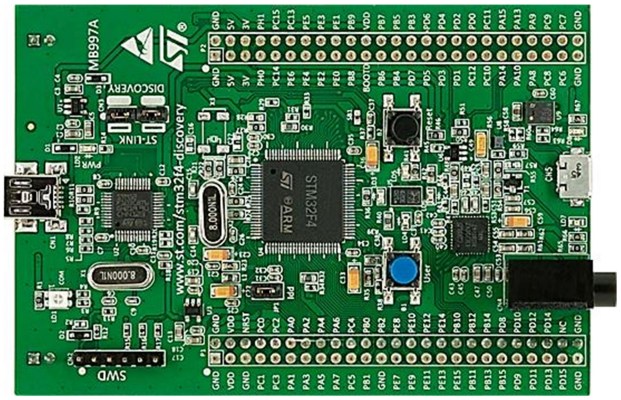


***Figura 6**: Placa STM32F4-DISCOVERY*

1.6.2.3. Elección de la placa

Para decidir qué placa se usará, se analizará el precio y sobretodo las prestaciones que aporta cada una en comparación con las demás en la Tabla 2.

Tabla 2: Comparativa entre placas STM32 y Arduino

	ARDUINO DUE	STM32F429I-DISCO	STM32F4DISCOVERY
Núcleo	Arm Cortex-M3 32bit	Arm Cortex-M4 32bit	ARM Cortex-M4 32bit
Máx. velocidad reloj	84MHz	180MHz	168MHz
Memoria Flash	512KBytes	2MBytes	1MByte
SRAM	96KB	256KB	192KB
E/S digitales	54	>100 GPIOs	82 GPIOs
Entradas analógicas	12	16 (3 ADC)	16 (3 ADC)
DAC	2	2	2
Alimentación	7-12V	3-5V	3-5V
Real Time Clock	No	Sí	Sí
UART	4	4 USART + 4UART	4 USART + 2UART
CAN	Sí	Sí	Sí
Consumo (Run mode)	130mA máximo	260µA/MHz= 46.8mA máximo	238µA/MHz= 40mA máximo
Corriente máxima de alimentación	800mA	100mA	100mA
Precio	43.32€	21,25€	14,63€

Se descarta inicialmente el Arduino Due por su precio mayor y sus peores prestaciones. Debido al gran conocimiento previo sobre la placa, a que tiene todo lo que necesitamos para la realización del proyecto, a que la tenemos disponible, a que las características son similares y por último a que su precio es menor, elegimos la placa **STM32F4DISCOVERY.**

1.6.3. Expansión

Una vez elegida la placa, necesitamos realizar todas las comunicaciones necesarias, esto es, comunicarnos con el GNSS, el sensor inercial y el sensor LIDAR. El primero y el segundo se comunican a través de RS-232, y el tercero a través de Ethernet. También necesitaremos, según la elección justificada más adelante, poder introducir una tarjeta Micro-SD para el guardado de datos.

En la placa base tenemos los pins necesarios para hacer estas conexiones, pero necesitaríamos un circuito adicional para cada una de ellas. Por este motivo, se decide utilizar una placa de expansión creada especialmente para nuestra placa discovery, la cual contiene la circuitería necesaria para cada una de las conexiones, y tan solo nos tenemos que preocupar de estudiar los pins utilizado por cada uno de ellos.

1.6.3.1. Placa de Expansión para STM32F4Discovery

Como hemos dicho, necesitamos un conector Ethernet, dos conectores RS-232 y una entrada para tarjeta Micro-SD. También podría ser interesante en un futuro interactuar con el sistema a través de una pantalla táctil LCD y tener otras comunicaciones disponibles por si se quiere añadir algún otro sensor (temperatura, humedad u otros).

Estas son las características que nos ofrece esta placa de expansión:

- Placa de expansión para STM32F4Discovery
- Interfaz para cámara
- Ranura para tarjeta Micro SD
- Pantalla táctil resistiva de 4 hilos
- 3.5 pulgadas LCD TFT a color (240 x 320 píxeles de resolución RGB, 262 mil colores, 16-bit 8080 interfaz paralela, control de brillo a través de PWM)
- 10/100 Ethernet con IEE 1588v2 (conector RJ45)
- Soporta uC / OS-II_v2.91
- Soporta Sistema FatFs_vR0.08a archivos (Utilizado para tarjeta del TF)
- Soporta LwIP_v1.3.2 Protocolo Stack

En la Figura 7 se puede observar más fácilmente todas las conexiones que nos facilita esta placa de expansión:

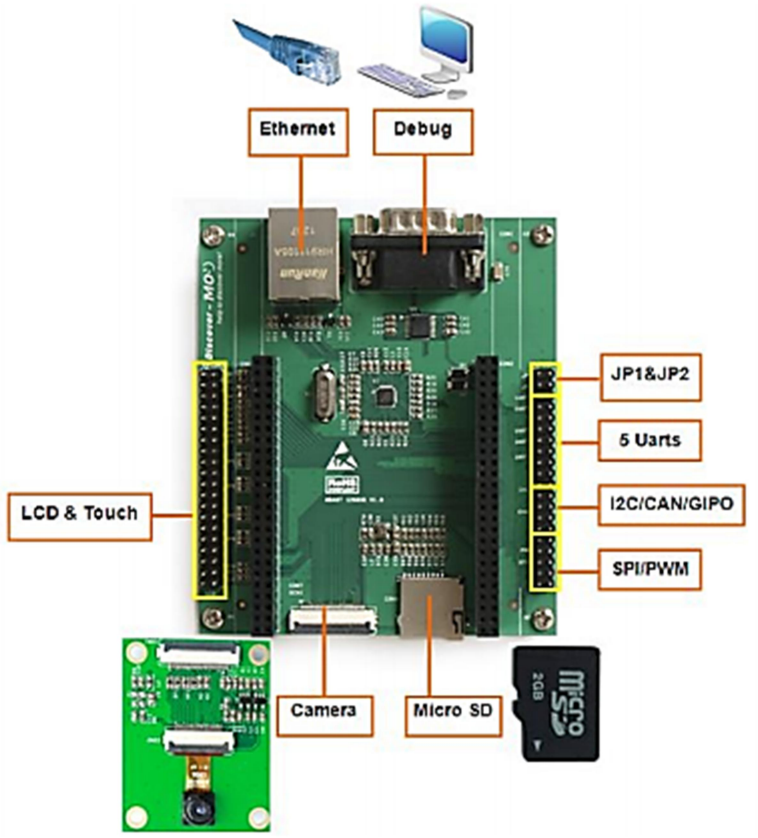


***Figura 7**: Comunicaciones de la placa de expansión de STMF4Discovery*

Podemos observar que tenemos disponible un conector para comunicación Ethernet, un conector para conectar un puerto RS-232 y un conector para introducir una tarjeta Micro-SD. Estos tres conectores tienen su correspondiente circuitería para adaptar la comunicación con lo que no tendremos que preocuparnos de realizarla nosotros, tan solo deberemos estudiar los puertos a los que están conectados estos dispositivos.

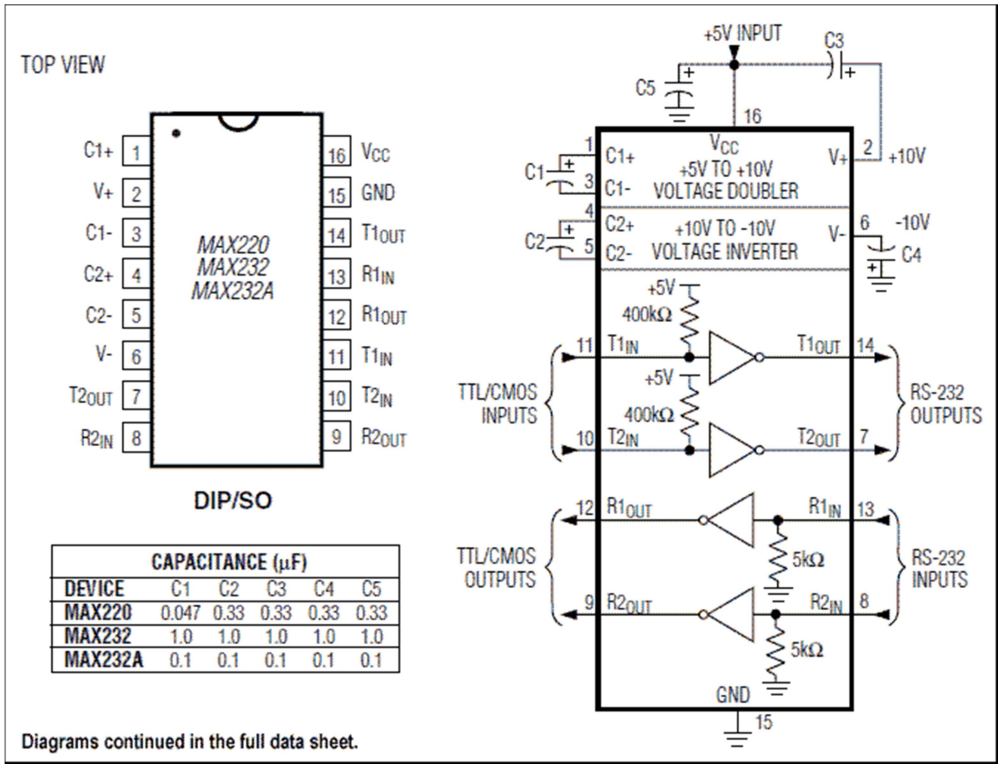
Tenemos el inconveniente de que esta placa de expansión tan solo cuenta con un puerto RS-232 de 9 PINs, pero cuenta con 5 hilos más para realizar la comunicación (5 Uarts), con lo que tan solo necesitaremos añadir uno externo y conectarlo correctamente a nuestra placa.

1.6.3.2. Puerto RS-232-Externo

Como hemos dicho, necesitaremos un conector externo para realizar la conexión del segundo dispositivo con el que nos comunicamos a través de este tipo de puerto. Para ello, podríamos realizar nosotros una circuitería, pero hemos visto que en el mercado hay placas sencillas y a buen precio que nos ahorran este trabajo y quizás incluso algo de dinero. Es por ello que se decide comprar un adaptador RS-232 Externo (Fig. 8).

***Figura 8**: Conector RS-232 Externo*

El funcionamiento de este dispositivo se base en un dispositivo MAX-232, al igual que en la placa de expansión, y su conexionado (Fig. 9) que éste necesita para trabajar correctamente.



CAPACITANCE (µF)					
DEVICE	C1	C2	C3	C4	C5
MAX220	0.047	0.33	0.33	0.33	0.33
MAX232	1.0	1.0	1.0	1.0	1.0
MAX232A	0.1	0.1	0.1	0.1	0.1

***Figura 9**: Esquemático MAX232*

Estos conexionados ya están hechos en la propia placa, y de lo único que nos tendremos que preocupar es de los pines de entrada y salida de datos hacia el mircroprocesador y los de alimentación (Fig. 10).

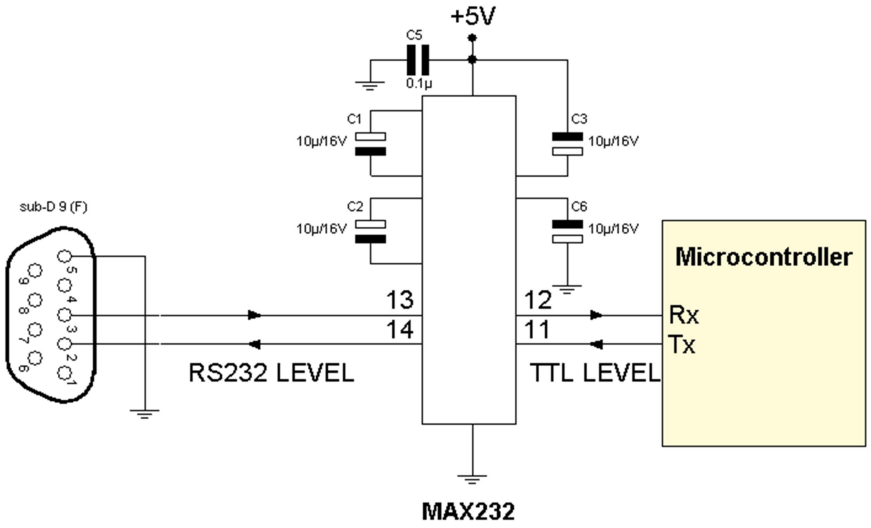


***Figura 10**: Conexionado Conector RS232-MAX232-Microprocesador*

Es decir, tan solo tendremos que conectar el pin 11 y 12 de nuestro adaptador a uno de nuestros cinco puertos Uarts disponibles (ver Fig. 11).

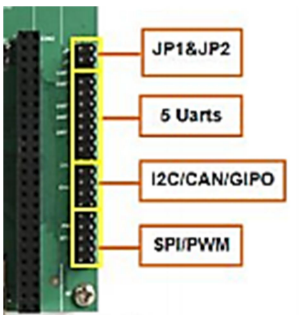


***Figura 11**: Uarts de la Placa de Extensión de STM32F4*

1.6.4. Entorno de programación

El microcontrolador con el que se trabaja dispone de un programador instalado, el ST-LINK/V2, fácilmente accesible mediante un cable USB type-A a Mini-B que se conectará al ordenador (véase Figura 12). Luego se necesita una herramienta de desarrollo compatible con la gama de productos STM32F4, concretamente para el STM32F4DISCOVERY. Atollic TrueStudio y Keil MDK son las dos más destacadas y con mayor soporte para el microcontrolador del proyecto.

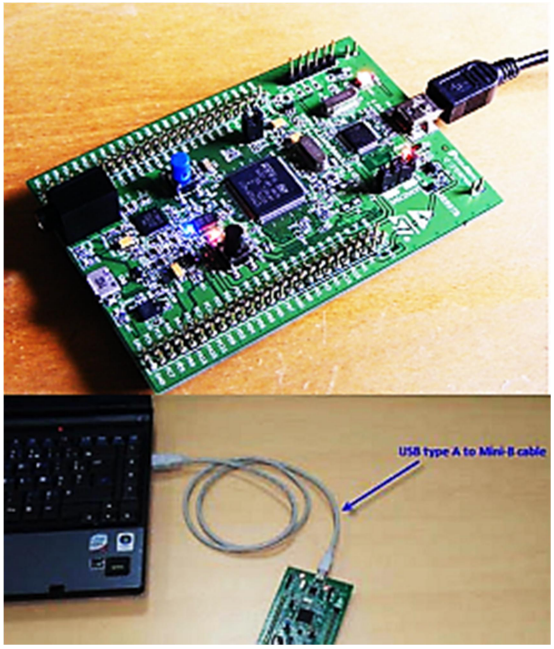


***Figura 12**: Esquema de conexión microcontrolador con ordenador para su programación*

1.6.4.1. Atollic TrueSTUDIO

Atollic es el entorno de desarrollo integrado (IDE) más potente disponible para microcontroladores ARM. Se trata de un IDE basado en uno de los compiladores C/C++ más usados en el mundo –el GCC creado por GNU Project– proporcionando así un generador de código fiable, compacto y de alto rendimiento para los Cortex-M4 como el utilizado en nuestro proyecto.

Con estas prestaciones, Atollic permite a los desarrolladores escribir y depurar rápidamente con facilidad. También facilita la creación de nuevos proyectos en pocos segundos gracias a un asistente integrado y además se disponen más de 1150 proyectos de ejemplo gratuitos para una gran variedad de placas de evaluación como la de este proyecto.

Su potente editor acepta lenguaje de programación C, C++ y ensamblador. Se destaca sus avanzadas prestaciones sobre la edición, navegación por el código y la estructura visual del código que aportan una eficiencia elevada en comparación con otras herramientas.

Su compilador C/C++ y el sistema de organización están altamente optimizados para crear un código eficiente y compacto. Junto a éste está el depurador o debugger que permite controlar no solo un procesador, sino que permite múltiples depuraciones simultáneas. Además, su sistema de análisis en tiempo real permite entender el comportamiento del sistema y programa para así identificar y corregir los errores.

Todas estas avanzadas prestaciones tienen el inconveniente de que se debe pagar un precio para poder usarlas. Por este motivo Atollic ofrece una versión gratuita que limita la extensión de código creado a 32KB, lo cual no es suficiente para generar nuestro código.

El programa se estructura en 3 zonas destacadas: el explorador de proyectos, el informe de errores o problemas y la zona de trabajo principal donde se edita el programa principal y las librerías (véase Figura 13).

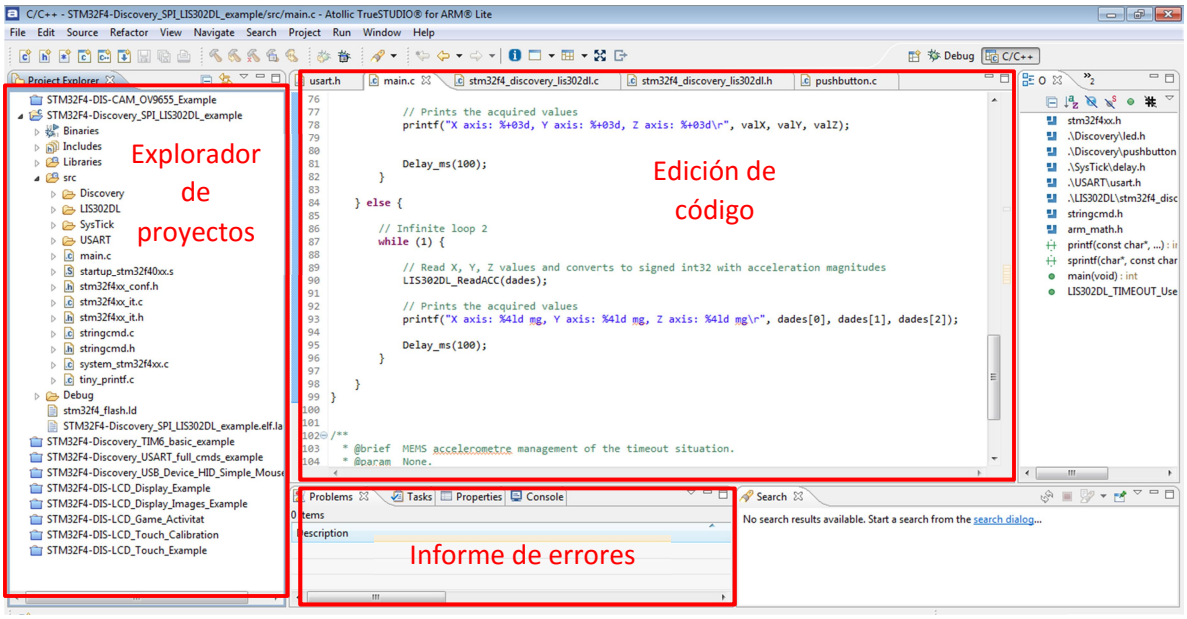


***Figura 13**: Entorno gráfico de AtollicTrueStudio*

1.6.4.2. KEIL MDK

El KEIL MDK-ARM es un entorno de desarrollo para procesadores basados en Cortex-M entre otros. Está especialmente diseñado para aplicaciones con microcontroladores, es fácil de aprender y de usarlo, y suficientemente potente para aplicaciones en sistemas encastados.

Usa un compilador distinto que el usado por Atollic pero no menos potente, el armcc creado por Arm. Es compatible con los lenguajes C/C++ y ensamblador. También permite depuración del programa a tiempo real para detectar errores en el código y ver su funcionamiento. Además, igual que Atollic, aporta una gran variedad de ejemplos para distintas placas de evaluación y da soporte para el STM32F4DISCOVERY.

Su editor es parecido al Atollic pero con menos prestaciones y peor navegación por el código. A pesar de esto, sigue siendo uno de los mejores por su buena estructura visual de código y por su extenso uso en el mundo de los microcontroladores. Consta de 3 zonas diferenciadas para editar el código, para gestionar las carpetas del proyecto y panel de información de la compilación (véase Figura 14).

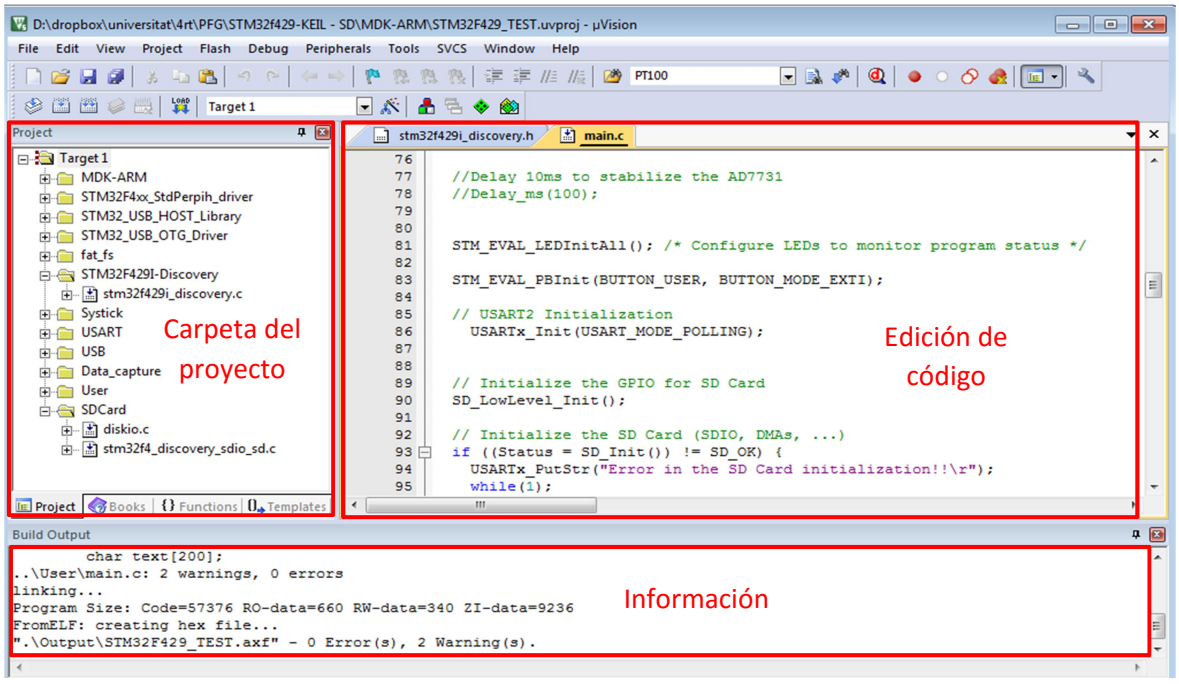


***Figura 14**: Entorno gráfico Keil MDK-ARM*

Dispone de una versión completa de su programa pero con un precio elevado. También ofrece una versión gratuita pero que limita el código a 32KB. En este punto ambas herramientas nos limitan el tamaño de código si no se desea pagar por una licencia pero se consigue una versión completa del KEIL MDK-ARM y se decide trabajar con él.

1.6.5. Guía de periféricos

En los siguientes puntos se procederá a describir cada uno de los periféricos usados para este proyecto. Se explicará de forma básica en que consiste cada uno y sus prestaciones en el microcontrolador STM32F4DISCOVERY.

1.6.5.1. ETHERNET

Ethernet, al que también se conoce como IEEE 802.3, es el estándar más popular para las LAN, usa el método de transmisión de datos llamado Acceso múltiple con detección de portadora y detección de colisiones (CSMA/CD) [4]. Antes de que un nodo envíe algún dato a través de una red Ethernet, primero escucha y se da cuenta si algún otro nodo está transfiriendo información; de no ser así, el nodo transferirá la información a través de la red. Todos los otros nodos escucharán y el nodo seleccionado recibirá la información. En caso de que dos nodos traten de enviar datos por la red al mismo tiempo, cada nodo se dará cuenta de la colisión y esperará una cantidad de tiempo aleatoria antes de volver a hacer el envío. Cada paquete enviado contiene la dirección de la estación destino, la dirección de la estación de envío y una secuencia variable de bits que representa el mensaje transmitido. El dato transmitido procede a 10 millones de bits por segundo y el paquete varia en una longitud de 64 a 1518 bytes, así el tiempo de transmisión de un paquete en la Ethernet está en un rango de 50 a 1200 microsegundos dependiendo de su longitud. La dirección de la estación de destino normalmente es referida por una única interfaz de red. Cada estación recibe una copia de cada paquete, pero ignora los paquetes que son dirigidos a otras computadoras y procesa solamente los que son dirigidos a ella. Las velocidades de envío de paquetes utilizando la tecnología Ethernet son de 10 Mbps (Ethernet estándar), 100 Mbps (Fast Ethernet – 100BASEX) y de 1000 Mbps utilizando el Gigabit Ethernet cuya especificación se encuentra respaldada por la IEEE con número 802.3z, el cual cumple los siguientes objetivos [38]:

- Permite realizar operaciones de envío y recepción de datos a una velocidad de 1000 Mbps.
- Usa el formato de frame Ethernet 802.3.
- Usa el método de acceso CSMA/CD con soporte para un repetidor por dominio de colisión.
- Las direcciones de retorno son compatibles con las tecnologías 10BASE-T y 100Base-T.

Las redes Ethernet tienen un esquema de direccionamiento de 48 bits. A cada computadora conectada a una red Ethernet se le asigna un número único de 48 bits conocido como dirección Ethernet. Para asignar una dirección, los fabricantes de hardware de Ethernet

adquieren bloques de direcciones Ethernet y las asignan en secuencia conforme fabrican el hardware de interfaz Ethernet, de esta manera no existen dos unidades de hardware de interfaz que tengan la misma dirección Ethernet. Por lo general, las direcciones Ethernet se colocan en el hardware de interfaz anfitrión de las máquinas de tal forma que se puedan leer. Debido a que el direccionamiento Ethernet se da entre dispositivos de hardware, a estos se les llama direccionamientos o direcciones físicas.

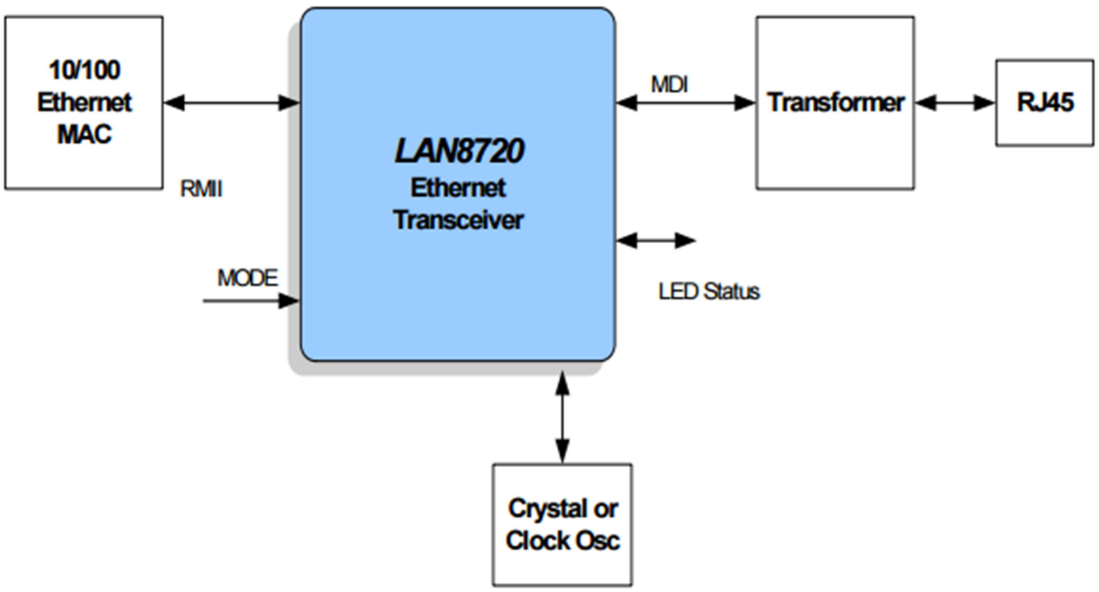


***Figura 15**: Esquema de funcionamiento LAN8720*

La trama de Ethernet es de una longitud variable pero no es menor a 64 bytes ni rebasa los 1518 bytes (encabezado, datos y CRC), cada trama contiene un campo con la información de la dirección de destino. En la figura 16 se muestra una trama Ethernet. Además de la información que identifica la fuente y el destino, cada trama transmitida contiene un preámbulo, un campo tipo, un campo de datos y un campo para verificación por redundancia cíclica (CRC- Cyclic Redundancy Check). El preámbulo consiste en 64 bits que alternan ceros y unos para ayudar a la sincronización de los nodos de recepción. El CRC de 32 bits ayuda a la interfaz a detectar los errores de transmisión: el emisor calcula el CRC como una función de los datos en la trama y el receptor calcula de nuevo el CRC para verificar que el paquete se reciba intacto [2].

Preámbulo	Dirección destino	Dirección fuente	Tipo	Datos	CRC
8 bytes	6 bytes	6 bytes	2 bytes	46-1500 bytes	4 bytes

***Figura 16**: Formato de una trama (paquete) que viaja a través de Ethernet*

El campo de tipo de trama contiene un entero de 16 bits que identifica el tipo de dato que se está transfiriendo en la trama. Desde el punto de vista de Internet, este campo es esencial porque significa que las tramas se autoidentifican. Cuando una trama llega a una máquina dada, el sistema operativo utiliza el tipo de trama para determinar qué módulo de software de protocolos se utilizará para procesar la trama. La mayor ventaja de que las tramas se autoidentifiquen es que éstas permiten que múltiples protocolos se utilicen juntos en una sola máquina y sea posible entremezclar diferentes protocolos en una sola red física sin interferencia. Los protocolos TCP/IP utilizan tramas Ethernet autoidentificables para hacer una selección entre varios protocolos. Cuando se transmite un datagrama IP versión 4 el campo tipo de trama contiene el valor hexadecimal 0800 [77] y al transmitir un datagrama IP versión 6 el campo tiene el valor hexadecimal 86DD [80].

- **Protocolos TCP/IP**

En forma general, el conjunto de protocolos TCP/IP tiene correspondencia con el modelo de comunicaciones de red definido por ISO (International Organization for Standardization), este modelo se denomina modelo de referencia de interconexión de sistemas abiertos (OSI). El modelo OSI describe un sistema ideal de redes que permite establecer una comunicación entre procesos de capas distintas y fáciles de identificar. En el host, las capas prestan servicios a capas superiores y reciben servicios de capas inferiores. La figura 17 muestra las siete capas del modelo de referencia OSI y su correspondencia general con las capas del conjunto de protocolos TCP/IP y en la tabla 3 se enumeran los protocolos más comunes del conjunto de protocolos TCP/IP y los servicios que proporcionan.

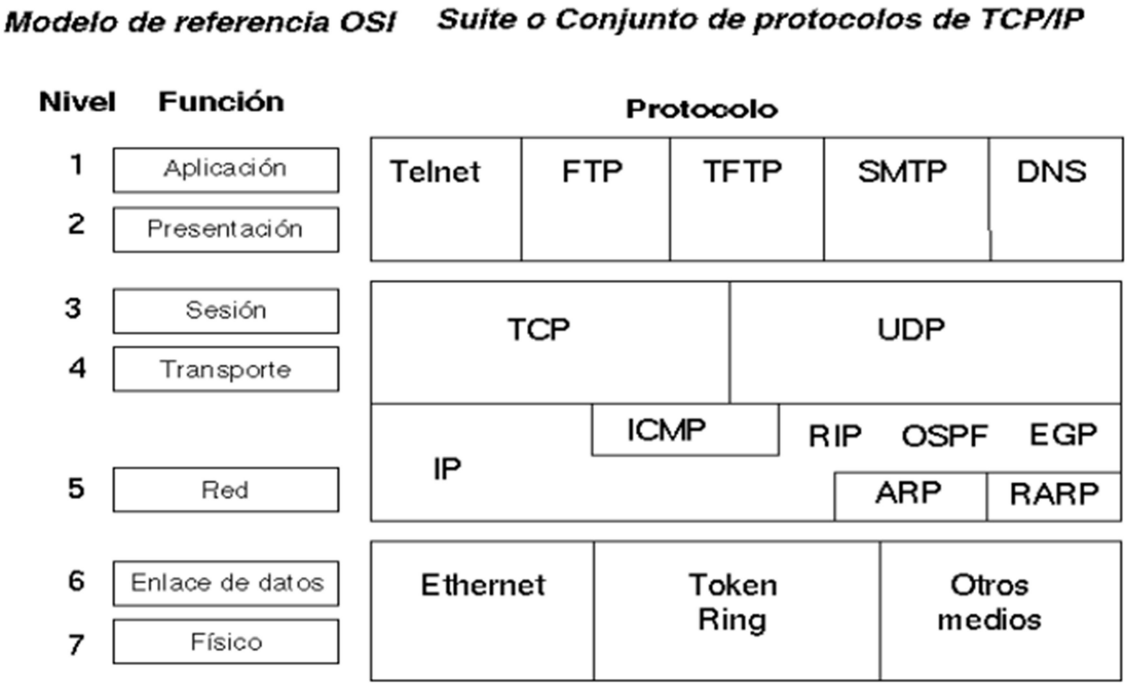


***Figura 17**: Modelo de referencia OSI y las capas de TCP/IP correspondientes*

***Tabla 3**: Protocolos más comunes de TCP/IP*

Protocolo	Servicio
Protocolo Internet (IP)	Proporciona servicios para la entrega de paquetes entre nodos.
Protocolo de control de mensajes de Internet (ICMP).	Regula la transmisión de mensajes de error y control entre los *hosts* y los *routers*.
Protocolo de resolución de direcciones (ARP).	Asigna direcciones Internet a direcciones físicas.
Protocolo de resolución de direcciones por réplica (RARP).	Asigna direcciones físicas a direcciones Internet.
Protocolo de control de transmisión (TCP).	Proporciona servicios de envío de flujos fiables entre los clientes.
Protocolo de *datagrama* de usuario (UDP).	Proporciona servicio de entrega de *datagramas* no fiable entre clientes.
Protocolo de transferencia de archivos (FTP).	Proporciona servicios de nivel de aplicación para la transferencia de archivos.
TELNET	Proporciona un método de emulación de terminal.
Protocolo de información de encaminamiento (RIP)	Permite el intercambio de información de rutas de vectores de distancia entre *routers*.
Protocolo abrir la vía más corta primero (OPSF)	Permite el intercambio de información de rutas de estado del enlace entre *routers*.
Protocolo Gateway Externo (EGP)	Permite el intercambio de información de rutas entre *routers* externos.

- **Protocolo de control de transmisión (TCP)**

Para las aplicaciones que deben enviar o recibir grandes volúmenes de datos, la entrega de datagramas no fiables puede convertirse en una carga. Del mismo modo que los datagramas UDP, los segmentos TCP se encapsulan en un datagrama IP. TCP guarda el flujo en el buffer y

espera a que un datagrama de tamaño grande se llene de datos antes de enviarlo, este flujo se caracteriza por carecer de estructura, de ahí que tanto la aplicación emisora como la receptora tengan que llegar a un acuerdo sobre el contenido del mismo antes de iniciar la transmisión. El protocolo TCP usa una transmisión dúplex integral (full-duplex), es decir, que pueden enviarse dos flujos de datos simultáneamente en direcciones opuestas. En consecuencia, la aplicación de destino puede enviar información de control o datos de vuelta a la aplicación emisora mientras ésta continúa enviando datos. El protocolo TCP asigna un número secuencial a cada segmento. La aplicación que se encuentra en el extremo receptor de la conexión, verifica los números de secuencia para asegurar que todos los segmentos se reciban y procesen en orden. El receptor envía un reconocimiento al emisor indicando los segmentos recibidos. TCP permite que el emisor tenga varios segmentos pendientes antes de que el receptor envíe un reconocimiento. Cuando el nodo emisor recibe el reconocimiento, indica a la aplicación que los últimos datos se enviaron satisfactoriamente, si el nodo emisor no recibe el reconocimiento de un segmento, en un período de tiempo determinado, volverá a retransmitir este segmento. Este esquema, llamado retransmisión con acuse de recibo, asegura que la entrega de flujo sea fiable. En la figura 1.9 se muestra el formato del segmento TCP, el cual tiene un encabezado mínimo de 20 bytes. Este encabezado está formado por los siguientes elementos: los campos PUERTO FUENTE y PUERTO DESTINO contienen los números de puertos TCP que identifican a los programas de aplicación en los extremos de la conexión. El campo NÚMERO DE SECUENCIA identifica la posición de los datos del segmento en el flujo de datos de transmisión. El campo NÚMERO DE ACUSE DE RECIBO identifica el número de bytes que la fuente espera recibir después. El campo HLEN contiene un número que especifica la longitud del encabezado del segmento, medida en múltiplos de 32 bits. El campo RESERVADO es de 6 bits que se reservan para ser usados en el futuro. El campo CODIGO de 6 bits es utilizado para determinar el propósito y contenido del segmento. Los seis bits indican cómo interpretar otros campos en el encabezado, de acuerdo con la tabla 4.

***Tabla 4**: Bits del campo CODIGO en el encabezado TCP*

Bit (de izquierda a derecha)	Significado si el bit está puesto a 1
URG	El campo de apuntador de urgente es válido
ACK	El campo de acuse de recibo es válido
PSH	Este segmento solicita una operación push
RST	Inicio de la conexión
SYN	Sincroniza el número de secuencia
FIN	Final de la conexión.

El campo VENTANA es utilizado para informar sobre cuántos datos está dispuesto a aceptar cada vez que se envía un segmento. El campo SUMA DE VERIFICACIÓN contiene una suma de verificación para comprobar la integridad de los datos así como el encabezado. El campo APUNTADOR DE URGENCIA solo es válido si la bandera URG está encendida, éste campo especifica la posición en la que terminan los datos urgentes dentro del segmento. El campo OPCIONES se utiliza comúnmente para especificar el tamaño máximo de segmento (MSS) que se está dispuesto a recibir (véase figura 18).

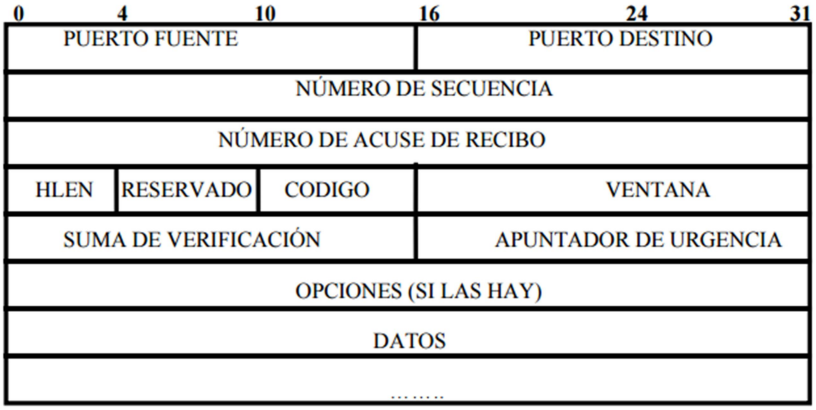


***Figura 18**: Formato de un segmento TCP*

Los pines Ethernet en el microprocesador se muestran en la figura 19

```
/*
Ethernet pins configuration
-----------------------------------------
ETH_RMII_TX_EN  -------> PB11
ETH_RMII_TXD0   -------> PB12
ETH_RMII_TXD1   -------> PB13

ETH_MII_RX_ER   -------> PB10    // NO ESTA CONECTAT FISICAMENT
ETH_RMII_RXD0   -------> PC4
ETH_RMII_RXD1   -------> PC5

ETH_RMII_CRS_DV -------> PA7
ETH_MDIO --------------> PA2
ETH_MDC ---------------> PC1

ETH_RMII_REF_CLK-------> PA1
ETH_RST_PIN     -------> PE2
-----------------------------------------
*/
```

***Figura 19**: Distribución de pines Ethernet en el microprocesador*

1.6.5.2. TIMER

Los temporizadores/timers son básicamente contadores que incrementan o disminuyen su valor automáticamente en función de un ratio marcado por su reloj. Cuando se alcanza el valor deseado en el contador, se produce un evento que reinicia el valor del contador a cero. Sus características principales se puedes resumir en los siguientes tres puntos:

- Permiten trabajar a altas frecuencias con suma precisión.
- No alteran el funcionamiento del procesador ni necesitan la intervención de la CPU ya que trabajan en segundo plano.
- Tienen una resolución de 8, 16 i 32 bits.

TIMERS en el microcontrolador

En el caso del STM32F4DISCOVERY, se dispone de un total de 14 temporizadores divididos en 3 grupos, cada uno de ellos con funcionalidades extra:

- **Temporizadores básicos**: Temporizadores de 16 bits sin entradas ni salidas que son usados como simples temporizadores o para activar conversiones DAC en la generación de señales (Tabla 5).

Tabla 5: Temporizadores básicos

Temporizador	Resolución contador	Tipo contador	Factor divisor	DMA	Captura/ Compara canales	Salida complementaria	Reloj máx del temporizador
TIM6, TIM7	16-bit	Ascendente	1 - 65536	Sí	0	No	90/180

- **Temporizadores de propósito general**: Temporizadores de 16 y 32 bits usados para medir la duración de pulso entrantes (input capture), comparar salidas o como generadores de señal PWM (Tabla 6).

Tabla 6: Temporizadores de propósito general

Temporizador	Resolución contador	Tipo contador	Factor divisor	DMA	Captura/ Compara canales	Salida complementaria	Reloj máx del temporizador
TIM2, TIM5	32-bit	Ascendente/ Descendente	1 - 65536	Sí	4	No	90/180
TIM3, TIM4	16-bit	Ascendente/ Descendente	1 - 65536	Sí	4	No	90/180
TIM9	16-bit	Ascendente	1 - 65536	No	2	No	180
TIM10, TIM11	16-bit	Ascendente	1 - 65536	No	1	No	180
TIM12	16-bit	Ascendente	1 - 65536	No	2	No	90/180
TIM13, TIM14	16-bit	Ascendente	1 - 65536	No	1	No	90/180

- **Temporizadores de control avanzado**: Temporizadores similares a los de propósito general pero con más funcionalidades. Permiten modular señales PWM entre 0 y 100% para usarlas en aplicaciones con motores (Tabla 7).

Tabla 7: Temporizadores de control avanzado

Temporizador	Resolución contador	Tipo contador	Factor divisor	DMA	Captura/ Compara canales	Salida complementaria	Reloj máx del temporizador
TIM1, TIM8	16-bit	Ascendente/ Descendente	1 - 65536	Sí	4	Sí	180

En los microcontroladores STM32F4 hay distintas fuentes de reloj, multiplexores y factores divisores que hay que tener en cuenta a la hora de configurar la frecuencia de trabajo de los temporizadores, por eso antes se debe observar el Clock Tree del Reference Manual (Fig 20) y así hallar la fuente de reloj correspondiente:

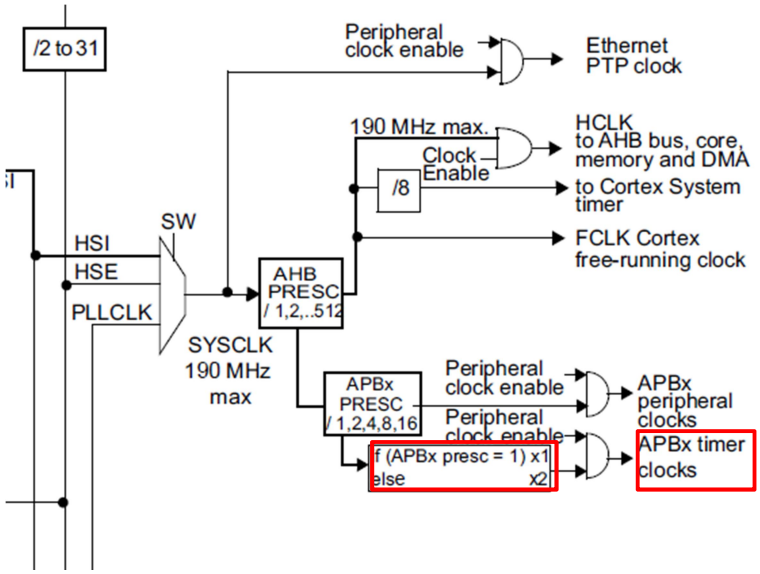


***Figura 20**: Clock Tree*

Cada temporizador tiene su propio reloj *TIMx_CLK* y este valor se determina en función de la frecuencia de reloj del bus (APBx) al que esté conectado cada uno (*PCLK1* en el bus *APB1* y *PCLK2* en el bus *APB2*). También se debe tener en cuenta los valores de los factores divisores (prescaler) del APB1 i APB2 ya que tal y como se muestra en la imagen anterior se puede dar dos casos:

$$APB1|APB2\ prescaler \neq 1 \qquad TIM_XCLK = (PCLK1|PCLK2) \cdot 2$$

$$APB1|APB2\ prescaler = 1 \qquad TIM_XCLK = (PCLK1|PCLK2)$$

Por defecto, en el STM32F4DISCOVERY estos valores son:

$$PCLK1\ (APB1) = 45MHz \qquad PCLK2\ (APB2) = 90MHz$$

$$APB1\ prescaler = 4 \qquad APB2\ prescaler = 2$$

Para conocer el bus al que está conectado cada temporizador se debe mirar el diagrama de bloques del microcontrolador (Fig. 21).

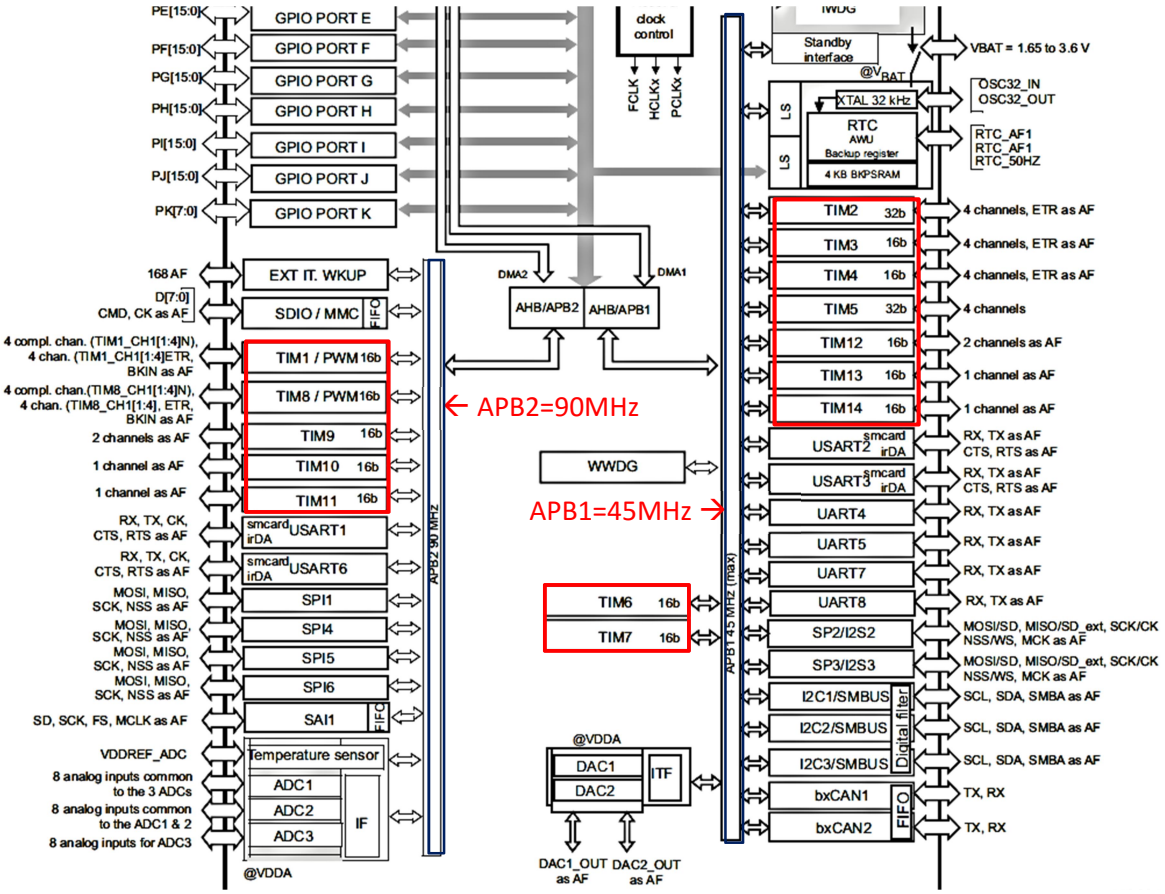


***Figura 21**: Diagrama de buses del microcontrolador*

El reloj *TIMx_CLK* del temporizador luego se divide por un factor llamado *prescaler* y un divisor del reloj llamado *Clock_division* que proporcionará la frecuencia de reloj *CLK_INT* que dirige el contador, por lo que cada $\frac{1}{CLK_{INT}}$ (s) el registro interno incrementará o disminuirá un bit:

$$CLK_{INT} = \frac{TIM_X CLK}{(Prescaler + 1) \cdot Clock\_division} \qquad Prescaler = \frac{TIM_X CLK}{CLK_{INT} \cdot Clock\_division} - 1$$

Una vez conocido su reloj interno, el temporizador se puede configurar para distintas funciones, entre ellas para activar un evento de forma periódica o para medir tiempo entre eventos:

Activar eventos periódicos

Después de configurar el reloj interno del temporizador, se determina el valor que debe tener el contador interno para que se produzca un evento o interrupción. El contador va aumentando con el reloj CLK_{INT} hasta que llega al valor predefinido *TIM_Period*, entonces se activa el evento o interrupción y el contador vuelve a cero para repetir el proceso (véase Figura 22).

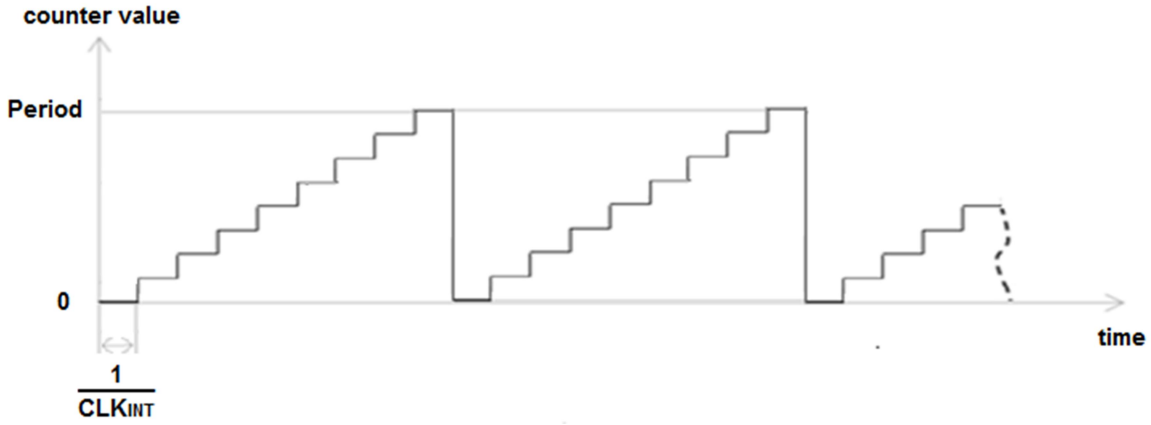


***Figura 22**: Gráfica (valor de contador-tiempo) para ver el funcionamiento básico de un temporizador*

La expresión del tiempo de actualización se calcula en función del reloj del temporizador configurado anteriormente y del valor de TIM_Period:

$$Tiempo_{actualización} = (TIM_{Period} + 1) \cdot \frac{1}{CLK_{INT}}$$

El valor máximo del TIM_Period dependerá de los bits de cada timer:

$$32\ bits \qquad TIM_{Period_{max}} = 2^{32} = 4.294.967.296$$
$$16\ bits \qquad TIM_{Period_{max}} = 2^{16} = 65.635$$

Así que la configuración del tiempo de cada temporizador en la placa STM32F4DISCOVERY se puede resumir en las dos expresiones siguientes:

$$Tiempo_{actualización} = (TIM_{Period} + 1) \cdot \frac{1}{CLK_{INT}} =$$
$$(TIM_{Period} + 1) \cdot \frac{(Prescaler + 1) \cdot Clock_{division}}{(PCLK1|PCLK2) \cdot 2}$$

$$Frecuencia_{actualización} = \frac{(PCLK1|PCLK2) \cdot 2}{(TIM_{Period} + 1)(Prescaler + 1) \cdot Clock_{division}}$$

Siempre teniendo en cuenta que TIM_Period no puede superar la capacidad del contador y que se usará PCLK1 (APB1) o PCLK2 (APB2) según el bus al que esté conectado cada temporizador.

Medir tiempo entre eventos

Se configura el temporizador en modo captura de canal o *input capture mode*. En este modo se prepara el temporizador para que detecte el nivel de subida o bajada de una señal y active una interrupción. Es muy útil cuando se quiere medir el tiempo entre dos pulsos de una señal, tanto periódica como no periódica.

El contador del temporizador irá aumentando de 0 hasta el valor de TIM_Period establecido (2^n como máximo) y repetirá la secuencia de forma indefinida (Véase Fig. 23). El comportamiento es el mismo que se ha explicado para activar eventos periódicos. La diferencia ahora es que cada vez que se active la interrupción de una señal entrante, se mide el valor del contador del timer en ese instante. Además, se contabiliza mediante interrupciones el número de veces que se ha sobrepasado el valor máximo del contador, es decir, el número de overflows generados.

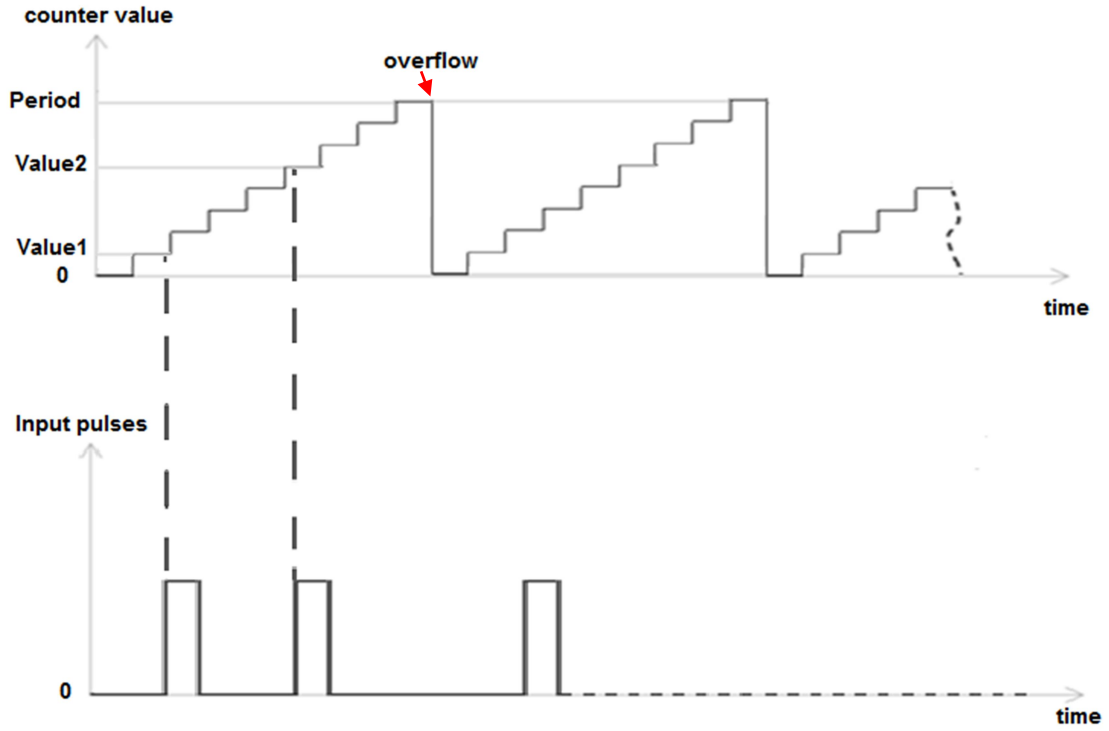


***Figura 23**: Gráficas temporizador y pulsos entrantes*

El tiempo entre eventos se calculará a partir de los valores de contador, del número de overflows y del reloj CLK_{INT} del temporizador:

$$Time = \frac{overflows \cdot period}{CLK_{INT}} + \frac{Value_2 - Value_1}{CLK_{INT}}$$

$$period = 2^n \ máximo$$

Donde n es el número de bits del temporizador usado y CLKINT es el clock interno del temporizador.

1.6.5.3. DMA

Una de las prestaciones más importantes en el microcontrolador es el Acceso Directo a Memoria (DMA). Este controlador permite la transferencia de datos a alta velocidad entre periféricos y memoria y entre memoria y memoria sin la acción de la CPU. Principalmente sirve para aplicaciones con alto nivel de carga de trabajo del procesador—como es el caso de este proyecto— ya que evita que la CPU esté constantemente ocupada en tareas de lectura y escritura y esté disponible para realizar otras tareas.

DMA en el microcontrolador

Concretamente, el microcontrolador STM32F4DISCOVERY dispone:

- 2 controladores DMA, cada uno con 8 corrientes o streams y luego cada stream tiene hasta 8 canales. Una corriente transmite la información de uno de sus canales y un elemento intermedio se encarga de arbitrar las prioridades entre todas las peticiones de cada stream para usar el DMA. Cada canal corresponde a un periférico específico de la placa y según la prioridad se enviará primero la información de una corriente u otra. En la Figura 24 se muestra el diagrama de bloques de un DMA con los 8 streams, los 8 canales para cada uno y el elemento arbitrario que gestionará qué corriente es la prioritaria, en este caso el Stream0.

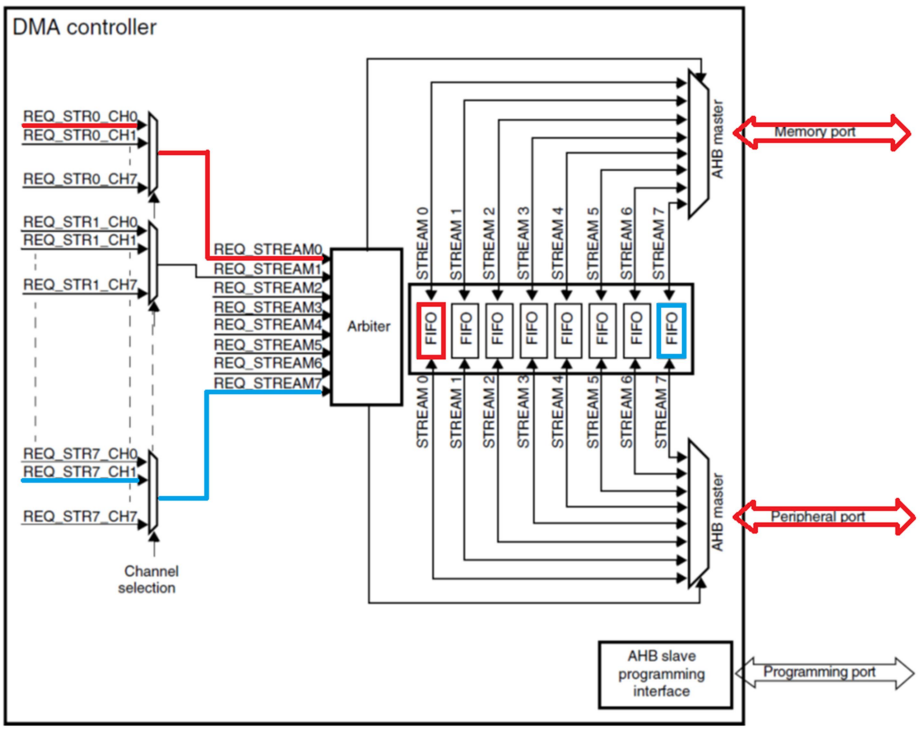


Figura 24: Diagrama de bloques del DMA donde el Stream0 tiene prioridad respecto el Stream7

- Consta de una arquitectura de doble bus de alto rendimiento AHB, uno para el acceso a memoria y el otro para el acceso a los periféricos. A pesar de disponer de 2 DMA, ambos no tienen las mismas prestaciones y solo el segundo controlador permite transferencias de memoria a memoria. A pesar de la restricción, el uso que se le dará en el presente proyecto será para transferencias de periféricos, los cuales leen los sensores, a memoria y no representa ningún problema.
- Buffers de memoria FIFO (First In, First Out) de 4 words (32 bits · 4).
- Por otro lado, cada controlador dispone de un conjunto de registros que pueden ser accedidos por la CPU y que se usan para su configuración. Consta de un registro de dirección de memoria, un registro de contador de bytes y registros de control. Éstos sirven para especificar los puertos de entrada y salida a usar, la dirección de la transferencia y las unidades de transferencia de datos —como bytes o words cada vez— entre otros. Las cantidad de datos a enviar y recibir son independientes en el STM32F4DISCOVERY por lo que se puede enviar 32 bits des del periférico y guardar solo 16 bits a través del DMA en memoria.
- Para todo los streams se puede configurar el buffer en modo circular, por lo que cuando se haya escrito a la última posición, vuelva a escribir en la primera automáticamente.
- Para gestionar los datos escritos en memoria y evitar perder información antes que el DMA vuelva a modificar los datos guardados, se dispone de 5 flags/banderas unificadas en una única interrupción para cada stream:
    - **DMA Half Transfer**: bandera para avisar que la transferencia de datos ha llenado la mitad del buffer. Útil en el modo circular para guardar en una memoria externa los datos de la primera mitad del buffer mientras se llena la otra mitad.
    - **DMA Transfer Complete**: bandera que avisa que se ha llenado el buffer. Igual que la de Half Transfer, es útil para guardar la segunda mitad del buffer mientras se llena la primera.
    - **DMA Transfer Error**, **DMA FIFO Error** y **DMA Direct Mode Error**: avisa de errores en la transferencia de datos.

Para la realización de cualquier operación a través del DMA, la CPU debe inicializar antes el proceso y configurar el sistema. Primero se envían los parámetros de la transferencia a

la interfaz (periférico o memoria) y luego al controlador DMA, donde se escribe a los registros recién mencionados para así definir los bytes a transferir, el tipo de transferencia —si es de lectura o escritura—, la dirección de memoria inicial o el canal del DMA a usar. A partir de este punto la CPU retoma sus tareas y se realiza la transferencia a través del DMA.

Principalmente la transferencia de periférico a memoria consiste en 3 operaciones consecutivas (Véase Fig. 25) que se pueden generalizar para cualquier otro modo de transferencia:

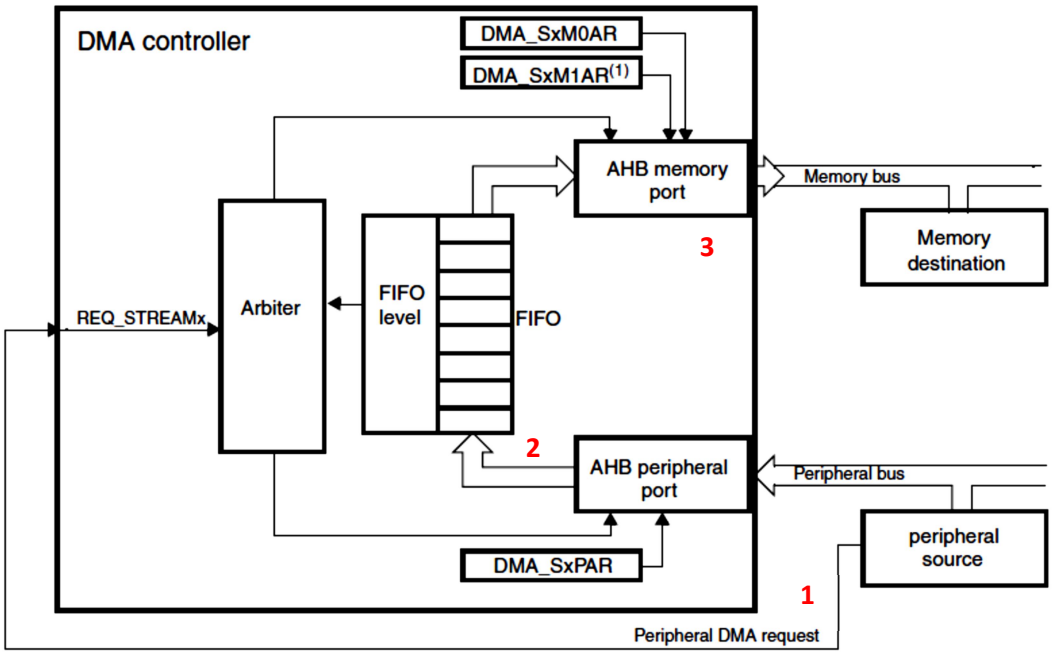


Figura 25: Diagrama de transferencia de periférico a memoria a través del DMA

1. Petición de envío de datos al elemento arbitrario del DMA des del periférico, luego éste prioriza las peticiones y activa la siguiente operación.
2. Carga de datos del registro de datos del periférico a un registro intermedio FIFO del DMA.
3. Los datos recién cargados al registro del DMA se envían a la localización de memoria configurada anteriormente.
4. Se decrece el registro que contiene el número de transacciones que todavía quedan por realizarse y se vuelve a enviar la petición.

1.6.5.4. USART/UART

El periférico USART/UART (Universal Synchronous/Asynchronous Receiver/Transmitter) es un elemento de hardware y que habilita la interface para comunicación serie usada en algunos dispositivos como el propio ordenador mediante el uso del protocolo RS-232.

A diferencia del UART, el USART permite dos tipos de modo de comunicación:

- Síncrona: disponible solo en USART. Se usa una línea extra para aportar una señal de reloj que rija el envío de datos en bloques.
- Asíncrona: El reloj para la transmisión de la información es generado de forma interna por cada dispositivo y se sincroniza mediante el uso de bits de inicio y fin de bloque de datos.

Las diferencias entre los dos modos se pueden resumir en los siguientes puntos:

- El modo síncrono necesita líneas para datos y para el reloj y en el asíncrono solo línea de datos.
- El modo síncrono la información se envía a una velocidad fija y en asíncrono no tiene por qué ser así.
- Los datos en síncrono se envían en forma de bloques, mientras que en asíncrono los datos se envían de un byte en un byte.
- El modo síncrono permite una mayor velocidad de transferencia de datos que en modo asíncrono.

La comunicación USART se caracteriza por una baja velocidad y por ser la más comúnmente usada en los años noventa, pero actualmente ésta se ha sido sustituida por interfaces más sofisticadas como SPI, USB o I2C ya que su velocidad es notablemente superior. A pesar de su antigüedad, todavía hoy es frecuente su uso en aplicaciones donde se necesite una comunicación simple y no muy veloz.

Protocolo de comunicación

La comunicación serie se caracteriza 5 pines pero se puede llegar a establecer una comunicación con un solo cable:

- RXD: Pin de entrada por el cual se lee los datos entrantes.
- TXD: Pin de salida del dispositivo que envía los datos al receptor.

- SCLK: Reloj serie en el caso de comunicación síncrona.
- CTS: Pin que determina el final de una transferencia de datos.
- RTS: Pin que avisa cuando el USART está preparado para recibir nuevos datos.

Existen distintos tipos de conexiones que el periférico USART permite, se resumen las principales tanto síncronas como asíncronas en los siguientes puntos:

- Conexión Half-Duplex asíncrona (Fig. 26): comunicación con un solo cable de forma asíncrona, se usa en aplicaciones donde solo existe comunicación en un sentido y un dispositivo es emisor y el otro receptor.

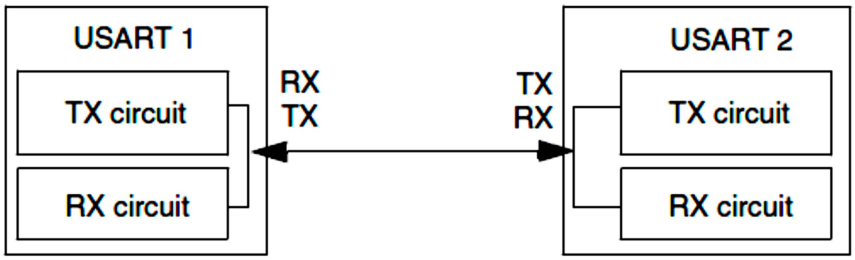


***Figura 26**: Conexión serie Half-Duplex asíncrona*

- Conexión Full-Duplex asíncrona (Fig. 27): comunicación con dos cables en la cual se envían datos entre los dos dispositivos.

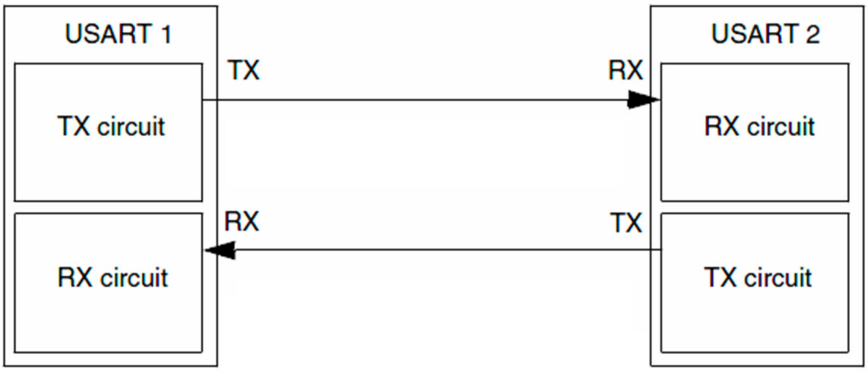


***Figura 27**: Conexión serie Full-Duplex asíncrona*

- Conexión Full-Duplex síncrona (Fig 28): comunicación con 3 cables, 2 para enviar y recibir la información y un tercero para la señal de reloj.

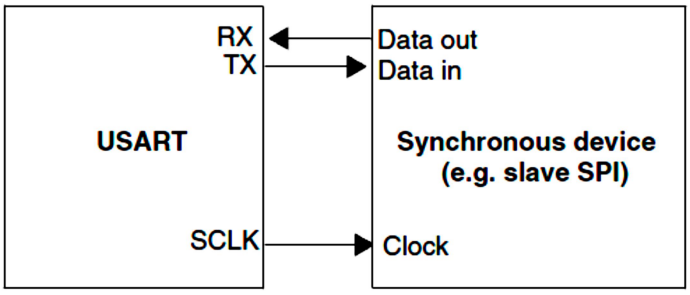


Figura 28: Conexión serie Full-Duplex síncrona

Para este proyecto se usará únicamente la comunicación síncrona Full-Duplex concretamente para la transmisión de caracteres (véase Figura 29 para ver un ejemplo de transmisión). Ésta se caracteriza por los siguientes puntos:

- Baud[1] rate totalmente programable de 4800, 9600, 19200 o 38400 entre otros. Será importante que cada dispositivo conozca el baudrate de la transmisión.
- Longitud de datos programable entre 5 y 9 bits.
- Bit de stop configurable para 0.5, 1, 1.5 o 2 bits de stop.
- Bit de control de paridad opcional.

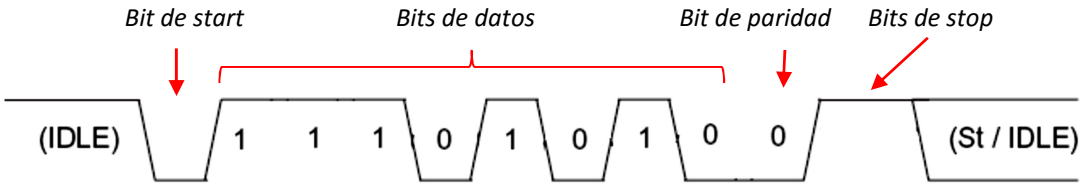


Figura 29: Ejemplo de transmisión del carácter W con 8 bits, un bit de paridad impar (0) y 1.5 bits de stop

USART/UART en el microcontrolador

El STM32F4DISCOVERY dispone de un total de 4 USART y 4 UART cada uno de ellos totalmente configurable e independiente del otro, sus características se resumen en los siguientes puntos:

- EL USART2, USART3, UART4, UART5, UART7 e UART8 están conectados al bus APB1 mientras que el USART1 e USART6 están conectados al bus APB2.
- Permite comunicación half-duplex con un solo cable y full-duplex.
- Permite tanto comunicación síncrona controlada por reloj serie como comunicación asíncrona.

[1] Baud: es la medida de velocidad de transmisión en comunicación asíncrona y es el número de señales por segundo, en el USART coincide con el número de bits por segundo.

- Número de bits de datos programable entre 8 y 9 bits y número de stop bits programables entre 0.5, 1, 1.5 y 2 bits.
- Baudrate totalmente programable.
    - SE alcanzan baud rates de 5250000 en los USART conectados al APB1.
    - Se alcanzan baud rates de 10500000 en los USART del APB2.
- DMA configurable con USART para almacenar datos recibidos sin la intervención de la CPU.

1.6.5.5. SDIO

Hoy en día es indispensable disponer de unidades de almacenamiento en todo tipo de dispositivos, tanto portables como no portables, y una forma de hacerlo es a través de discos duros de reducido tamaño. En este contexto aparece el Secure Digital (SD), un formato de tarjeta de memoria para dispositivos portátiles como teléfonos móviles, cámaras fotográficas o videoconsolas.

El estándar SD fue creado en 1999 como una mejora de las tarjetas multimedia (MMC). Dentro de este formato se incluye 4 versiones de tarjetas en distintos tamaños, son la Standard Capacity o SDSC, la High Capacity (SDHC), la Extended-Capacity (SDXC) y la Input/Ouput (SDIO). Además de estas 4, cualquier dispositivo con ranura SD es totalmente compatible con su antecesor, la MMC.

La aplicación del SDIO en este proyecto consiste en el uso de uno de los modelos de tarjetas de memoria, concretamente una microSD (la versión reducida de la SD como se puede ver en la Figura 30), para guardar los datos captados por los sensores y así poder transferirlos después al ordenador.

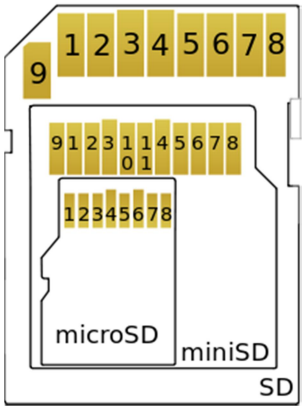


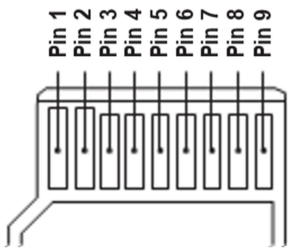
***Figura 30**: Comparativa de tamaños de SD, miniSD y microSD*

Protocolo de comunicación

A pesar de ser los modelos más recientes, todas las tarjetas de memoria SD y SDIO deben soportar los modos de transferencias siguientes:

- el antiguo modo MMC/SPI: 4 cables con salida y entrada serie.
- Modo SD de un bit: el cual separa comandos, canales de datos y un formato propietario de transferencia.
- Modo SD de 4 bits: el cual usa terminales extra para soportar transferencias de 4 bits en paralelo.

Por lo que respeta a la interfaz de hardware, SD y SDIO usan 9 pines que se diferencian de los 7 de la MMC (Véase Figura 31).



Pin	Nombr	Tipo	Descripción
1	DAT2	I/O/PP	Línea de datos (bit
2	DAT3	I/O/PP	Línea de datos (Bit
3	CMD	PP	Línea de comandas
4	VDD	S	Alimentación (+)
5	CLK	I	Reloj
6	VSS	S	Alimentación (-)
7	DAT0	I/O/PP	Línea de datos (bit
8	DAT1	I/O/PP	Línea de datos (bit
9	CD	I/O/PP	Detección tarjeta

***Figura 31**: Descripción de pines de tarjetas SD y SDIO*

La comunicación gira alrededor de comandas y transferencias de datos. En toda transacción deberá existir un servidor o host que la controle, en este caso será el microcontrolador. Éste enviará las comandas a través del terminal CMD y la tarjeta responderá a través del mismo, a continuación el host o tarjeta enviará bloques de datos a través de la líneas DAT según indica la comanda anterior. En la figura 32 podemos ver un ejemplo de esta comunicación.

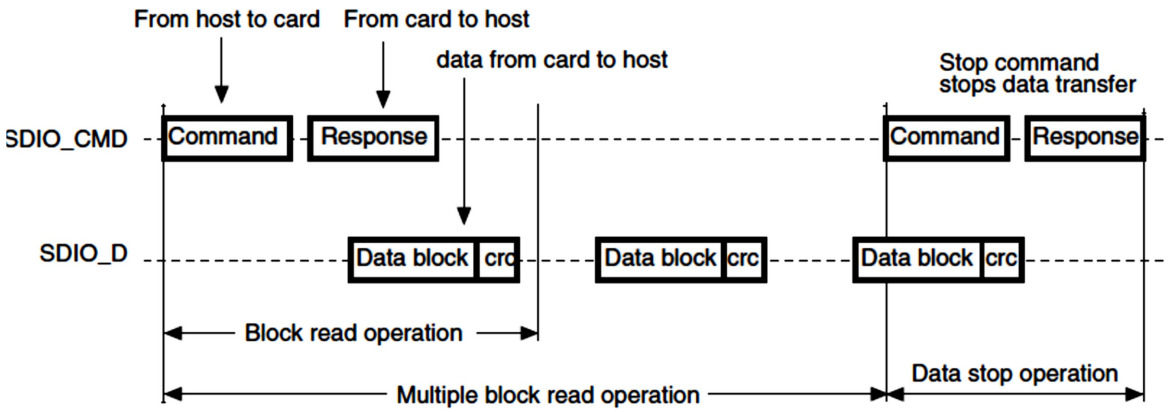


Figura 32: Diagrama de comunicación SDIO con envío de comanda inicial y lectura a través de las líneas de datos

Por otro lado, además de disponer de un sistema de escritura y lectura de datos como es el estándar SD/SDIO, se necesita un sistema de ficheros para la organización de los datos en la memoria de modo que puedan ser interpretados en otros dispositivos como un ordenador. Para este proyecto se elige el formato de ficheros FAT debido a la existencia de la librería FATFS diseñada específicamente para pequeños sistemas encastados como los microcontroladores. La FATFS está escrita en C y está completamente separada de la capa de entradas y salidas del disco que en este caso corresponde a las funciones del SDIO.

SDIO en el microcontrolador

- La interface SD/SDIO/MMC disponible en el STM32F4DISCOVERY consta:
- Compatibilidad total hasta la versión 4.2 de tarjeta multimedia MMC con soporte para 3 modos distintos de bus: 1 bit, 4 bits y 8 bits.
- Compatibilidad con las versiones 2.0 de tarjeta SD y SDIO con los modos bus de 1 y 4 bits.
- Transferencia máxima de 48 MHz para 8 bits.
- La versión actual solo permite la conexión de una única SD/SDIO/MMC a la vez.
- Señales de salida de comandas y datos activadas para controlar envío de información bidireccional.
- Los pines SDIO no son compatibles con la comunicación SPI, en caso que se desee usarla, se configurarán los pines específicos de uno de los SPI del microcontrolador.
- Reloj máximo de SDIO de 48MHz, en el caso del STM32F4DISCOVERY éste solo se podrá conseguir reduciendo el reloj máximo del microcontrolador a 168MHz sin alcanzar los 180MHz.

1.6.5.6. USB

El Universal Serial Bus o USB es un estándar de comunicación con origen en los años 90 que fue diseñado para unificar las conexiones de periféricos a los ordenadores personales. Actualmente su campo de aplicación está ampliamente y su uso está presenta tanto en los ordenador como en cualquier dispositivo electrónico portable.

La primera versión de USB fue creada en 1995 e implementada por primera vez por Intel. Fue nombrada USB 1.0 pero no fue hasta el año 1998 que las versiones 1.1 de USB fueron usadas de forma extendida. Inicialmente alcanzaban velocidades entre 1.5Mbit/s hasta 12Mbit/s.

La siguió la versión USB 2.0 creada en el año 2000 con un incremento de velocidad de 40 veces mayor, según las especificaciones se alcanzaban 480Mbit/s. 6 años más tarde salió el USB On-The-Go (OTG) como complemento al USB 2.0 ya que hacia posible que 2 dispositivos USB se comunicasen sin la necesidad de disponer de un USB host.

Ya en el año 2008 fue publicado el USB 3.0 el cual aportaba in incremento notable en el ratio de transferencia hasta 5Gbit/s. Además reducía el consumo y se hacía compatible con el USB 2.0. Se mejoró su versión en el año 2013 con el USB 3.1 para alcanzar velocidades de 10Gbit/s.

Dentro de sus especificaciones existe infinidad de clases de USB y de dispositivos. Son un ejemplo las webcams, los ratones, los teclados o los almacenamientos masivos (MSC) como las memorias USB (Fig. 33). Este proyecto se centrará en la aplicación de un MSC para guardar la información de las conversiones como alternativa a la memoria SD.

***Figura 33**: Memoria USB estándar*

Protocolo de comunicación

El protocolo USB se basa en un máster o host que controla el bus y múltiples esclavos pero mediante el USB OTG se extiende la comunicación entre 2 dispositivos ya que éste permite trabajar tanto como host como esclavo.

El USB 2.0 consta de 4 terminales mientras que el 3.0 añade 2 pares más:

- **VBUS:** terminal de alimentación entre 4.4 hasta 5.25V. Se necesita 100mA para el USB2.0 y 150mA para el USB3.0 y pueden soportar hasta 500mA y 900mA respectivamente.
- **GND:** terminal de suelo 0V.
- **D+/D-:** terminal de datos positivo y negativo, uno es la inversa del otro. Se usa para eliminar el ruido común en la comunicación (Véase figura 34).

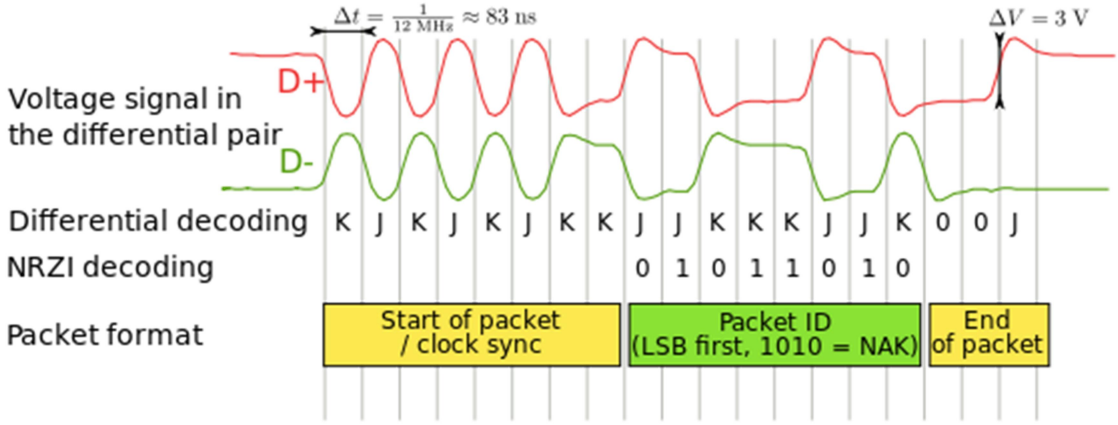


***Figura 34**: Comunicación USB a través de las señales de datos D+ y D-*

- **SSTX+/SSTX-:** par de terminales transmisores de SuperSpeed.
- **SSRX+/SSRX-:** par de terminales receptores de SuperSpeed.

La comunicación USB es compleja de estructurar, se basa en las llamadas tuberías virtuales o *pipes* por las cuales se envían distintos tipos de transferencia de datos correspondiente a un esclavo distinto por cada una: *Bulk Transfers*, *Control Transfers*, *Interrupt Transfers* o *Isochronous Transfers*. A vista más genérica, realmente una comunicación USB se basa en series de frames[2], cada una de ellas con un indicador de inicio del frame (SOF) seguido de una serie de transacciones que contienen los datos según el tipo de transferencia, tal y como se puede observar en la Figura 35.

[2] Unidad de medida de datos agrupados en un milisegundo (ms).

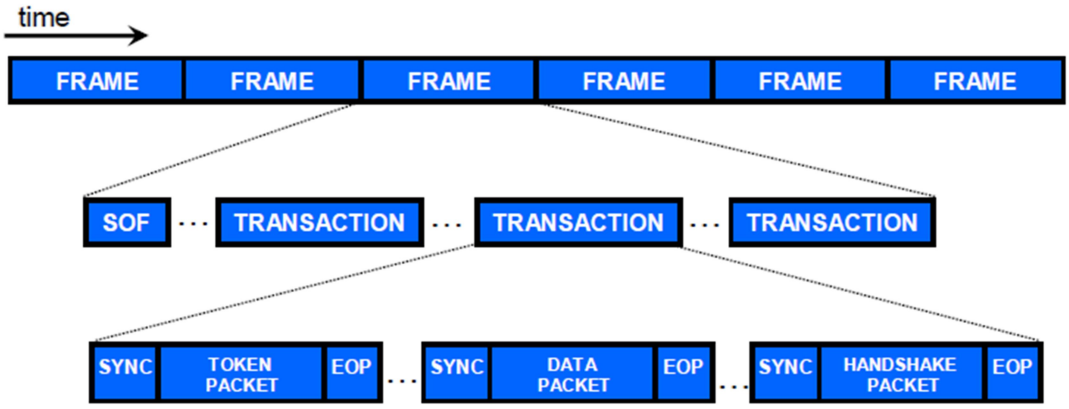


***Figura 35**: Diagrama general de comunicación USB con envío de frames*

Igual que ocurre con el SDIO, además de usar un sistema de escritura y lectura como es el USB, se necesita un sistema de ficheros como el FATFS para que los datos guardados puedan ser interpretados correctamente en el ordenador. La misma librería FATFS que se usará con el SDIO servirá para el USB.

USB OTG en el microcontrolador

El STM32F4DISCOVERY dispone de un USB OTG con las siguientes prestaciones:

- Sigue las especificaciones USB 2.0.
- Permite al host desactivar el VBUS en aplicaciones para conservar el consumo de batería.
- Permite control y monitoreo de VBUS con comparadores internos.
- Se dedica 1.25 Kbytes de RAM exclusivamente para el USB OTG.
- Garantiza el máximo ancho de banda del USB para la transferencia de un frame sin la intervención del sistema.
- Permite cambios dinámicos del rol host-periférico, de modo que el microcontrolador puede cambiar fácilmente entre uno y otro.

En la aplicación del proyecto se usará el microcontrolador en modo host para controlar la lectura y escritura en la memoria USB. En este modo el microcontrolador permite hasta 8 canales o pipes que cada uno puede ser totalmente reconfigurables para los 4 tipos de transferencias de datos, pero solo será necesario un canal con *Bulk Transfer* destinado exclusivamente a la memoria USB externa.

1.6.6. Adquisición y procesado de datos

Para conseguir el completo funcionamiento de este proyecto necesitamos la adquisición de datos de tres sensores distintos, uno que mide la distancia entre el objeto y el punto donde nos encontramos, el segundo que nos diga la inclinación a la que se encuentra nuestro sensor, y por último un sensor que nos diga nuestra posición exacta en el espacio. Por último, necesitaremos un dispositivo para realizar el guardado de todos estos datos para después poder utilizarlos.

Con los datos de estos tres sensores y su correcto guardado, posteriormente podremos trabajar estos datos en un entorno como Matlab y podremos dibujar un mapa en tres dimensiones del entorno escaneado.

1.6.6.1. Sensor LIDAR

La tecnología utilizada para la medida de distancias es la tecnología light detection and ranging (LIDAR), y más concretamente, la basada en el tiempo de vuelo (time of flight, ToF). El sensor utilizado es el LIDAR UTM-30LX-EW, de la marca Hokuyo. Sus características más importantes se muestran en la tabla 8.

__Tabla 8__: Características sensor LIDAR

Model	UTM-30LX-EW
Power Source	12V DC +/- 10% , Current Max 1A at start-up, Normal use 0.7A
Light Source	Pulsed laser diode (λ=905nm), Laser safety class 1
Detection Range	0.1m to 30m (500mm x 500mm or more, White Kent Sheet)
Accuracy	0.1m to 10m +/- 30mm, 10m to 30m +/- 50mm
Scan Area & Resolution	270º Resolution 0.25º
Scan speed	25ms/scan
Protocol	SCIP2.2 (Exclusive command)
Interface	Ethernet 100 Base-TX (Auto-negotiation) TCP/IP Synchronous output: NPN open collector
Connection	Power / synchronous output cable 2m Ethernet RJ-45 with male / female connector 30cm
Dimensions	62 x 62 x 87mm Weight 370g
Environment	-10 to +50°C @ 85% humidity (no condensing or icing)
Protection	IP67

Como podemos observar trabaja a través de Ethernet, con posibilidad de activar la auto-negociación de IP y el cable de conexión es el RJ-45. Por otro lado, nos está dando una resolución de 0,25º recorriendo un total de 270º, es decir, 1080 lecturas por escaneo (Fig. 36). También podemos ver que realiza un escaneo cada 25ms (40Hz), con lo que tendremos que ser capaces de recoger, decodificar y guardar esta información y la del resto de sensores en un tiempo menor que este para no perder ninguna lectura.

Para conseguir estos datos necesitaremos primero comunicarnos con el sensor y conocer las peticiones y respuestas con las que trabaja, así como sus diferentes configuraciones. Una vez se consigue recibir datos, tendremos que conseguir interpretarlos y decodificarlos.

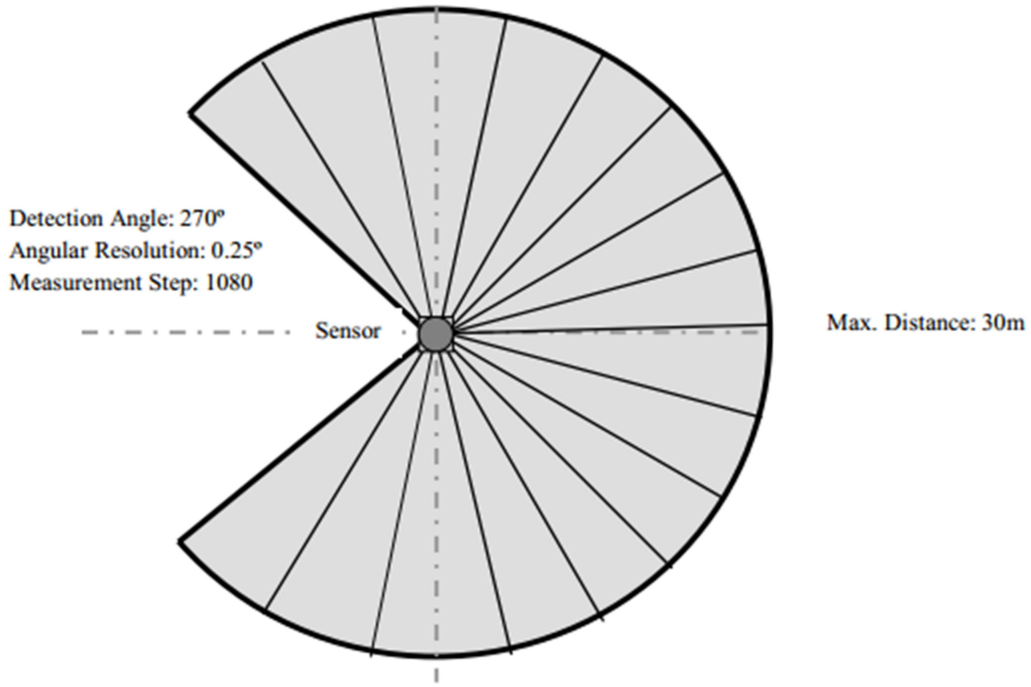


***Figura 36**: Area de detección del sensor LIDAR*

Comunicación con el LIDAR

En la comunicación básica, la petición de información se envía desde el equipo host hasta el sensor y el mensaje de respuesta se devuelve desde el sensor hasta el ordenador central (microprocesador en este caso). Cuando se solicitan datos para cada exploración, se envía desde el sensor hasta el sistema host de forma secuencial después del mensaje de respuesta correspondiente al mensaje de petición. Exploración se refiere a la medición de una exploración por el sensor. El mensaje enviado desde el sensor para cada exploración se denomina mensaje de respuesta de exploración. Un mensaje de respuesta de exploración se envía desde el sensor hasta el host hasta el final de un mensaje de solicitud de varias exploraciones o hasta que se envía un mensaje de petición de parada forzada.

Un mensaje de petición desde el host incluye un código de comando. Los mensajes de respuesta y respuesta de exploración se definen para cada código de comando. El grupo de mensaje de solicitud, mensaje de respuesta y el mensaje de respuesta de exploración como función de este código en su conjunto se llama comando, y su forma de utilización se define para cada sensor. El código de comando se expresa por dos caracteres del alfabeto en mayúsculas. Los comandos que empiezan con un carácter '%' son los comandos introducidos por Hokuyo automática Co. A continuación tenemos las tablas de comandos extendidas.

En la Tabla 9 se pueden encontrar los comandos para realizar peticiones de medida.

***Tabla 9**: Comandos de medicion (LIDAR)*

Command Code	Function	Request message parameters	Response message status and data	Scan response message status and data
GD GS	Distance acquisition	Start Step End Step Cluster Count	Status Time Distance	
GE	Distance and intensity acquisition	Start Step End Step Cluster Count	Status Time Distance, Intensity	
HD	Multiecho distance acquisition	Start Step End Step Cluster Count	Status Time Multiecho Distance	
HE	Multiecho distance and intensity acquisition	Start Step End Step Cluster Count	Status Time Multiecho Distance and Intensity	
MD MS	Distance acquisition with continuous scanning	Start Step End Step Cluster Count Scan Interval Number of Scans	Status	Status Time Distance
ME	Distance and intensity acquisition with continuous scanning	Start Step End Step Cluster Count Scan Interval Number of Scans	Status	Status Time Distance and Intensity
ND	Multiecho distance acquisition with continuous scanning	Start Step End Step Cluster Count Scan Interval Number of Scans	Status	Status Time Multiecho Distance
NE	Multiecho distance and intensity acquisition with continuous scanning	Start Step End Step Cluster Count Scan Interval Number of Scans	Status	Status Time Multiecho Distance and Intensity

En la Tabla 10 podemos ver los comandos para realizar cambios de configuración.

***Tabla 10**: Comandos de configuración (LIDAR)*

Command Code	Type	Function	Request message parameters	Response message status and data
%ST	State information	Obtain the current sensor condition		Status Current condition
BM	State transition	Transition to measurement state		Status
QT	State transition	Transition to standby state		Status
%SL	State transition	Transition to sleep state		Status
RS	Resetting	Reset		Status
RT	Resetting	Partial reset		Status
RB	Resetting	Reboot		Status
SS	Setup	Communication Speed Setup	Bitrate	Status
TM	Time synchronization	Time Setup	Control code	Status
VV	Information	Obtain Version		Status Version information
PP	Information	Obtain Sensor Parameters		Status Sensor parameters
II	Information	Obtain Sensor State		Status Sensor state

- **Request (Petición):**

Además del código de comando y los parámetros, un mensaje de solicitud puede incluir una cadena definida por el usuario, así como un terminador de petición. El código de comando se expresa por dos o tres caracteres del alfabeto en mayúsculas. De acuerdo con esa cadena de

código, se define los datos de estado del sensor y la respuesta. Los parámetros varían según el código de comando y son múltiples enteros a partir de cero. El número de dígitos de cada valor es fijo y se expresa en caracteres numéricos ASCII (en base 10). Cuando el número de dígitos del valor es menor que el número especificado de dígitos, deben colocarse ceros en los dígitos de orden superior.

Ejemplo	1 Carácter	1 2 3
	2 Caracteres	01 02 03 23 45
	3 Caracteres	001 003 023 045 002 678 789

La cadena definida por el usuario es una cadena de caracteres opcional comenzando con un punto y coma, que puede ser usado para identificar un mensaje. Los caracteres que se pueden utilizar para la cadena de caracteres después del punto y coma son alfabetos, números y seis tipos de caracteres '(espacio)', '.', '_', '+', '-' y '@'. El máximo de caracteres para esta cadena es de 15. El terminador de solicitud puede ser o bien el carácter de avance de línea (LF) o el de retorno de carro (CR), o sino ambos (LF y CR) escritos juntos como una cadena de dos caracteres. Las figuras 37 y 38 muestran un mensaje de solicitud y una cadena definida por el usuario, respectivamente.

Los elementos que pueden ser omitidos están marcados con un cuadro gris.

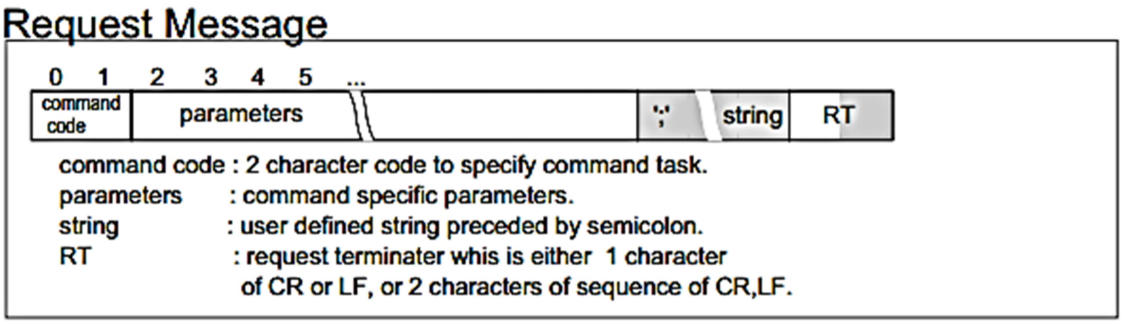


***Figura 37**: Estructura de mensaje de Petición LIDAR*

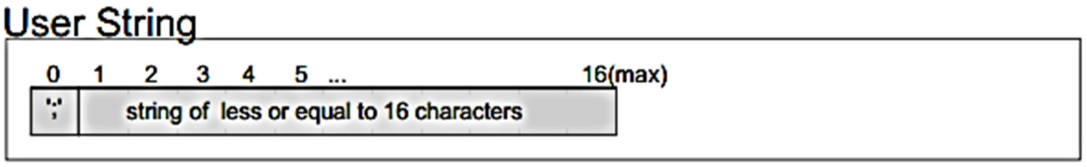


***Figura 38**: Frase de usuario para mensaje de petición LIDAR*

- **Respuesta**

El mensaje de respuesta se envía desde el sensor hasta el microprocesador tan pronto como le es posible tras recibir la solicitud. El mensaje de respuesta es una cadena de datos que se define por un código de eco, de estado y de comando de eco. Cada una de ellas está delimitada por un delimitador de respuesta. El eco es la retransmisión de la cadena de caracteres del mensaje de petición tal como lo hemos enviado, excluyendo sólo el finalizador petición (RT). El estado es una cadena de dos caracteres que se define de acuerdo con el

comando solicitado. Tras esto, hay un código de verificación para estos dos caracteres. A continuación, se añaden cadenas de datos opcionales en función del comando solicitado. Para la cadena de datos opcional, se añade un código de verificación de cada delimitador de respuesta. El final del mensaje de respuesta es una respuesta de dos delimitadores consecutivos. La Figura 39 muestra el formato básico de un mensaje de respuesta.

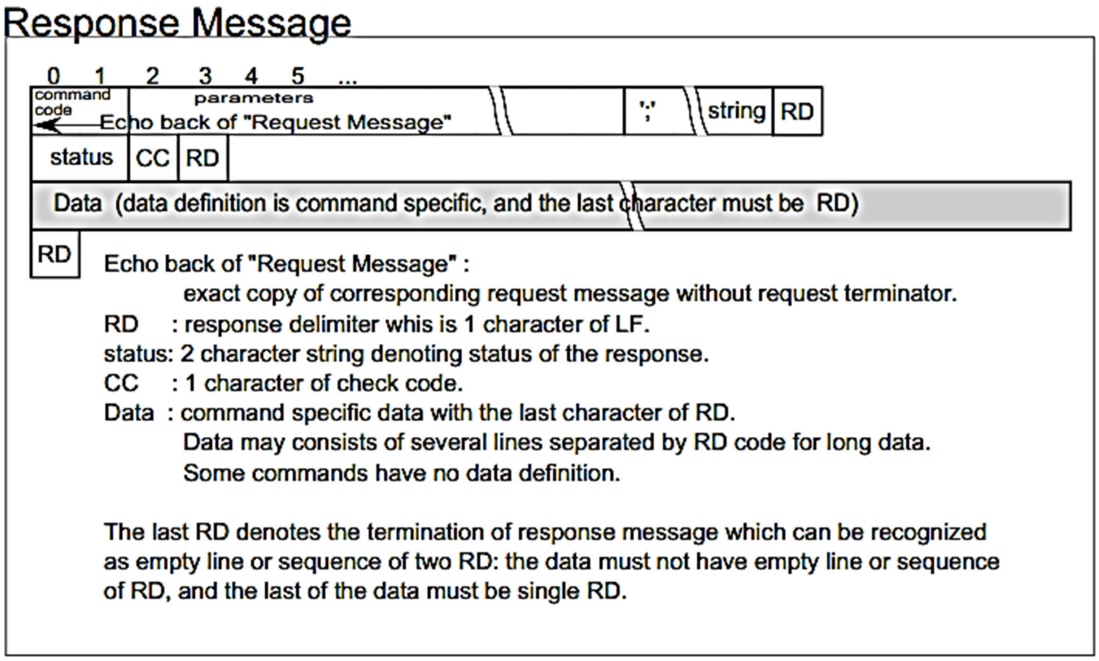


***Figura 39**: Estructura de mensaje de respuesta LIDAR*

Hay un comando de solicitud para obtener datos de varias exploraciones en modo continuo. Para ello, el sensor sólo envía un mensaje de respuesta sin datos de medición al host. Entonces, el sensor envía los datos de medición de cada exploración hacia el host como un mensaje de respuesta. El mensaje de respuesta tiene el mismo formato que el de un mensaje de respuesta normal. Hay que tener en cuenta que parte del eco de vuelta no es la cadena de caracteres del mensaje de petición como tal, sino que se cambió parcialmente. El estado consiste en un código de dos caracteres que muestra el estado de medición del sensor para cada exploración y un código de verificación. La Figura 40 muestra la forma básica del mensaje de respuesta.

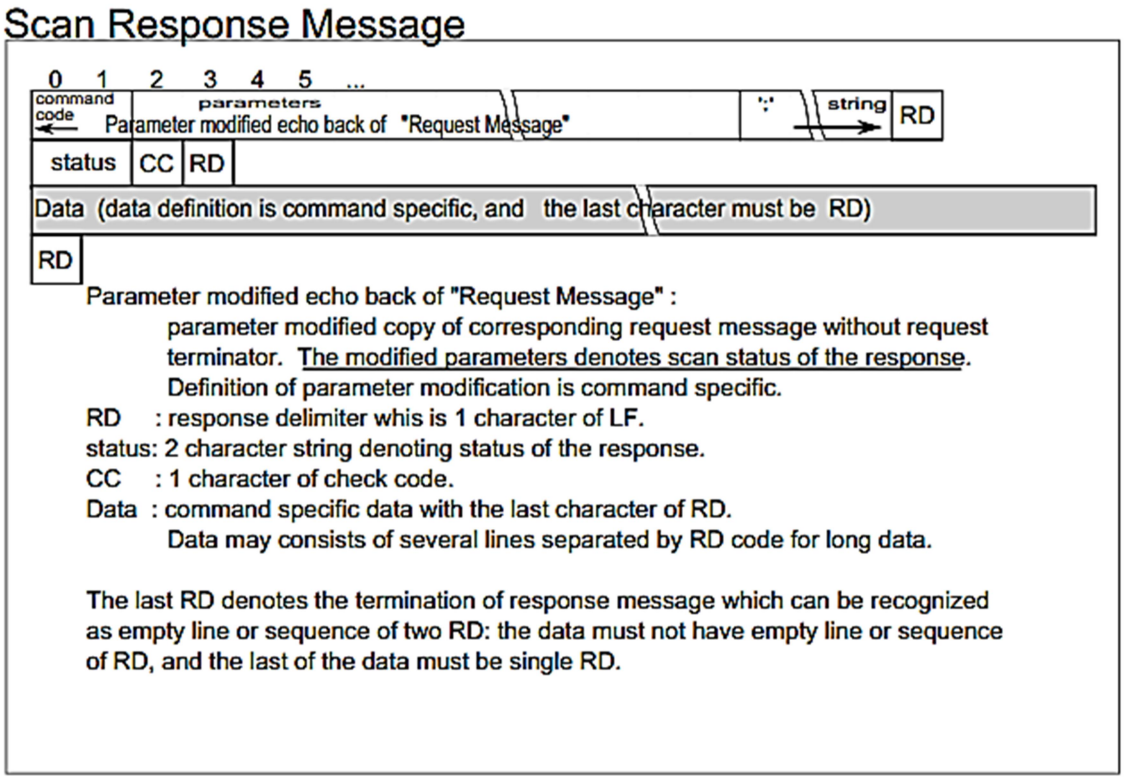


***Figura 40**: Estructura de mensaje de respuesta 2 LIDAR*

- **Descripción de la comunicación en este proyecto**

De cara a este proyecto, nos interesa que el microprocesador utilice todos sus recursos en captar, procesar y guardar los datos, evitando tener que estar enviando peticiones al sensor cada vez que queramos recibir datos.

Por ello, la opción elegida es la de recibir distancia multi-eco (hasta 3 distancia en una misma dirección) con adquisición continua. Esto es, el sensor, una vez le hagamos esta primera petición, no parará de enviarnos datos hasta que le digamos lo contrario.

Para ello, utilizaremos la sentencia ND, correspondiente a la acción anteriormente citada.

Cuando el sensor recibe un mensaje de solicitud de ND, envía un mensaje de respuesta que no tiene datos de medición. Si no hay error en el estado del mensaje de respuesta, el sensor envía la información de medición en los mensajes de respuesta hasta el final del proceso de escaneado. Sin embargo, si hay retrasos en las comunicaciones, algunos de los mensajes de respuesta de escaneo podrían no ser entregados. La información sobre el número de respuestas pendientes se incluye en cada mensaje de respuesta, por lo que es posible verificar si un mensaje no se entregó. En el último mensaje de respuesta enviado, el número de mensajes pendientes debe ser cero. La sintaxis básica del mensaje de respuesta no incluye datos de medición. El mensaje de respuesta incluye los datos de tiempo y datos de medición, sin embargo, si el estado corresponde a un error, se omiten tanto los datos de tiempo y como los datos de medición. El mensaje de respuesta incluye una porción de cadena de caracteres llamado echoback; en este echoback, el número de exploraciones se cambia todo el tiempo

con información sobre el número de escaneos pendientes (el echoback se modifica). El número de exploraciones pendiente indica el número de exploraciones que quedan por leer, sin incluir el mensaje actual. En el último mensaje de respuesta, el número de exploraciones pendientes se establece en 0, y cuando el número de exploraciones es ilimitado, el número de exploraciones pendientes en todos los mensajes de respuesta de exploración también es 0.

Cuando el sensor está en condición inestable, éste trata de recuperarse. Durante la recuperación, se devuelve el mensaje de respuesta con el estado que muestra la condición inestable. Cuando el sensor vuelve a la condición normal, el mensaje de respuesta se devuelve con el estado normal. Si el sensor conmuta a la condición anormal, el mensaje de respuesta muestra la condición anormal y la comunicación se para. Los datos de respuesta en el mensaje de respuesta para el comando ND incluye datos de la hora y datos de distancia se dividen en bloques. Sin embargo, si el estado en el mensaje de respuesta corresponde a un error, se omiten los datos de tiempo y datos de distancia. El comando ND utiliza codificación 3 caracteres (18 bits) para representar datos de distancia.

Las figuras 41 y 42 muestran la sintaxis básica de los parámetros de mensaje de respuesta y su orden. Esos campos que pueden ser opcionales (cadena definida por el usuario, datos en tiempo y bloques de datos a distancia) se presentan en color gris en la figura.

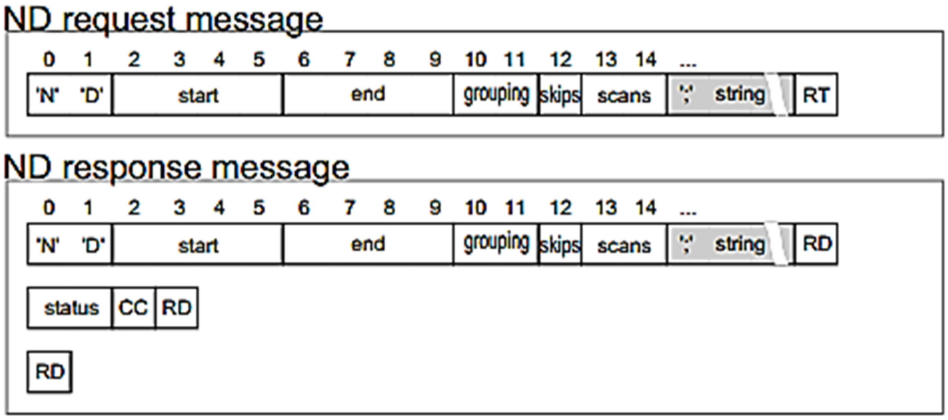


***Figura 41**: Petición y respuesta del comando ND (LIDAR)*

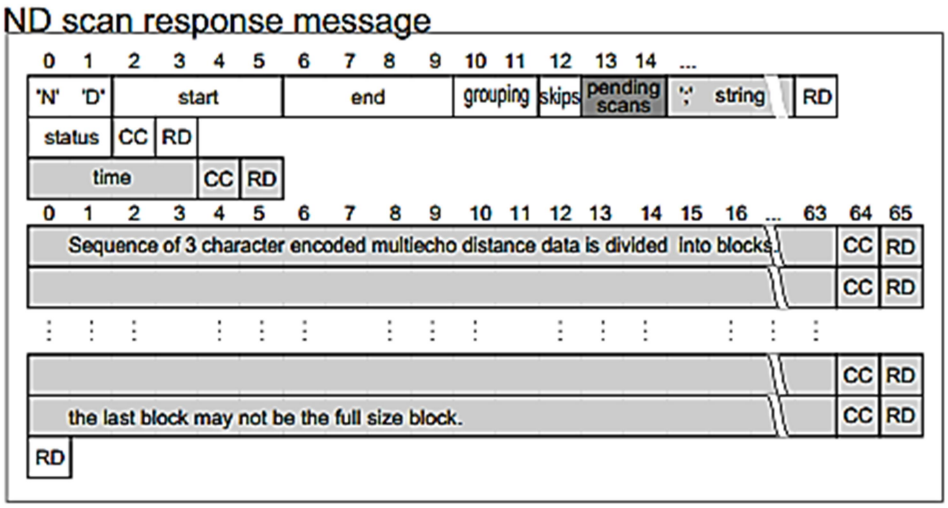


***Figura 42**: Respuesta con bloque de datos del comando ND (LIDAR)*

- **Check code**

El check code corresponde a la suma de los valores enteros de 8 bit de la cadena de caracteres a enviar, y después tomando los 6 bits de orden inferior y representando estos en codificación de 6 bits como un carácter simple. Aquí hay un ejemplo de esta fórmula:

```
'A'    'B'    'C'    '0'    '1'    '2'
0x41 + 0x42 + 0x43 + 0x30 + 0x31 + 0x32 = 0x159
0x19 + 0x30 = 0x49
              'I'
```

En cuanto al mensaje de respuesta, el check code es calculado utilizando la cadena de caracteres situada entre los delimitadores de respuesta. A continuación, el check code se pone entre la frase de caracteres y el delimitador de respuesta que le precede.

- **Codificación**

El sensor convierte la represenctación numérica a caracteres ASCII legibles a través de la codificación 6-bit para enviar el dato numérico con tráfico reducido en el canal de comunicación. Esta conversión se consigue añadiendo un offset de 0x30 (hexadecimal) al valor que se envía. Por ejemplo, el valor 26 (0x1a) es expresado en codificación de 6-bit como 0x4a, que corresponde al carácter alfabético "J".

Además de esto, cada paquete completo lo forman un total de 3 datos, es decir, el servidor (sensor) envía 6 bits hacia el cliente (microprocesador) en cada transferencia, pero un dato completo del sensor son 18 bits.

- **Distancia**

Los datos de distancia se envían como un número entero positivo expresado en milímetros, lo que puede representar distancias de 12 bits (hasta 4095 [mm]) y distancias de 18 bits (hasta 262143 [mm]). Éstos están representados por 2 codificación de caracteres y 3 de codificación de caracteres respectivamente. Como el valor máximo que puede devolver el sensor está definido, éste se devuelve si la distancia real supera el valor. La Figura 43 muestra el formato.

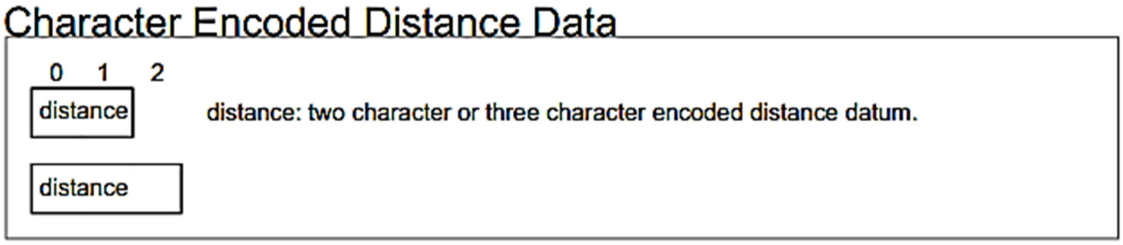


***Figura 43**: Formato del dato distancia (LIDAR)*

- **Multieco**

El sensor LIDAR es capaz de recibir múltiples reflejos para cada paso/ángulo (rayo láser), y obtener la información de la distancia para cada una de esas reflexiones. Recibir información múltiple de distancias se llama multiecho. En el modo agrupación de paso, se devuelve los datos multiecho para el paso con la distancia más pequeña en cada grupo. La forma de los datos multiecho incluye datos de distancia o datos par distancia intensidad pedidas a la menor distancia obtenida entre todos los ecos. Si hay datos para más de un eco, '&' se utiliza como

separador. El número máximo de datos que pueden ser devuelto por un paso depende de la especificación de sensor,en nuestro caso será de 3. La figura 44 muestra el formato de los datos multiecho.

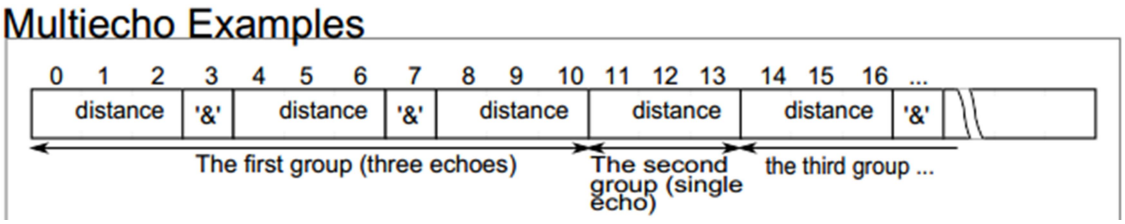


***Figura 44**: Estructura de una respuesta con multi-eco (LIDAR)*

Para las exploraciones que comprenden una única distancia, el número total de pasos en esa exploración fija la carga de datos y la longitud del mensaje para esa exploración. Sin embargo, para los datos multiecho, el número de datos varía según el paso para cada exploración. Por lo tanto, la longitud del mensaje cambiará entre exploraciones.

- **División de bloques**

Como la longitud del mensaje para cada exploración se hace muy largo, los datos de exploración se dividen en grupos de 64 y se insertan caracteres y delimitadores de respuesta. La cadena de caracteres dividido se llama bloque de datos. Para cada bloque, se añaden un código de verificación y un delimitador de respuesta. Esta operación se denomina división de bloques. La longitud de la cadena de caracteres de cada bloque es de 66 caracteres. Sin embargo, como la longitud de los datos de exploración no siempre es un múltiplo exacto de 64, sólo el último bloque podría tener menos de 64 caracteres de longitud. La figura 45 muestra el formato básico de un bloque.

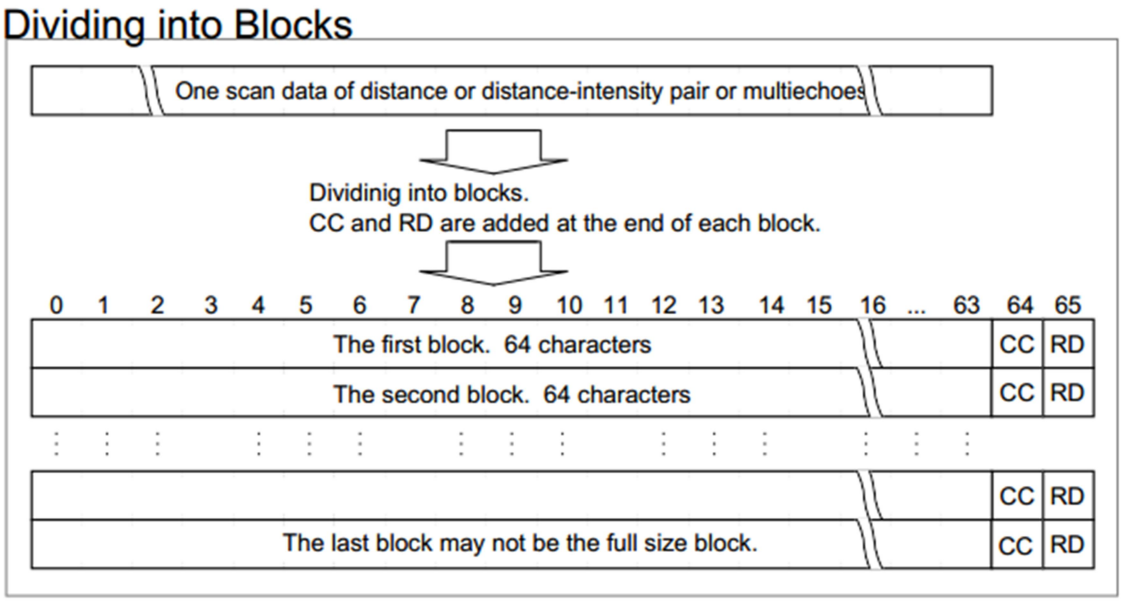


***Figura 45**: Estructura de bloques de datos (LIDAR)*

- **Código**

El código para la adquisición de datos de sensor LIDAR tiene la siguiente estructura (Fig 46).

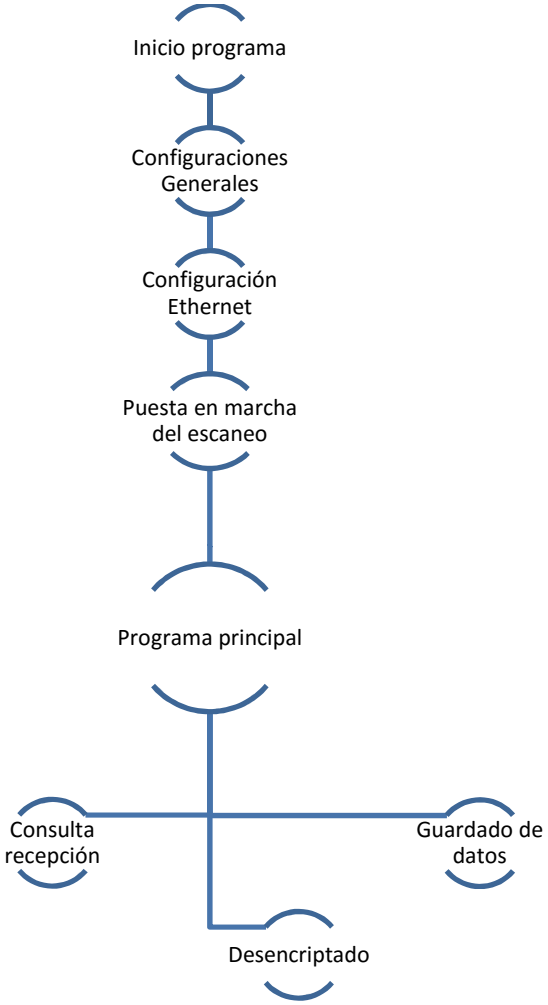


***Figura 46**: Esquema programa de control LIDAR*

Una vez iniciado el programa, el primer paso es la configuración e inicialización de todos los periféricos para su posible control y comunicación. Tras ello, nos comunicamos con el sensor LIDAR para que comience a hacer el escaneo. Por último, en el programa principal, iremos decodificando y guardando todos los datos que nos lleguen a través de la interrupción creada por la comunicación Ethernet con el LIDAR. A continuación se explica más detalladamente cada uno de estos pasos.

- **Configuraciones Generales**

- Configuración del SysTick para funciones de retardos.
- Configuración del Timer e inicialización de éste.
- Configuración de salidas para control de los LEDs.
- Configuración de entrada para recepción del Botón de usuario.
- Configuración e inicialización del USART.

- Configuración e inicialización de la ranura para la tarjeta SD.

- **Configuración Ethernet**

Configuración e inicialización de la comunicación Ethernet (MAC, PHY chip y LwIP stack):

Dirección IP: 192, 168, 0, 21
Máscara subred: 255, 255, 255, 0
Puerta de enlace: 192, 168, 0, 1
Velocidad de transmisión: Auto Negociación
DHCP: Desactivado

Una vez configurado, enviamos comando de Conexión a la dirección IP y puerto del sensor y esperamos confirmación de conexión, en caso contrario, tras un lapso de tiempo, volvemos a intentar una nueva conexión.

IP address: 192.168.0.11
Port number: 10940

- **Puesta en marcha del escaneo**

Para iniciar el escaneo del sensor, lo primero que haremos será encender el láser enviando el comando de encendido:

BM;\r

Tras ello, enviaremos el comando para iniciar el envío de datos por parte del sensor, en nuestro caso nos interesa multi-eco y envío de datos infinito, con lo que enviaremos la siguiente cadena:

ND0000108001000;\r

ND: Comando para activar el multi-eco (3 posibles distancias en la misma dirección) con escaneo contínuo.

0000: Punto de inicio del escaneo
1080: Punto final del escaneo

Es decir, analizamos los 270º que nos posibilita el sensor.

01: Cluster count (valor de la distancia más pequeña [entre ecos])

0: Intervalo de escaneo (escanea todos los puntos)

00: Número de escaneos a realizar (ilimitado [Hay que finalizar con QT])

- **Programa principal**

Una vez inicializado el escaneo, en el programa principal tan solo vamos a ir consultando si hemos recibido un dato (dato completo, la librería ethernet lo trabaja en "segundo plano"), y

una vez recibido, lo desencriptará y guardará en la memoria SD para después volver a esperar el siguiente dato.

Desencriptado

Una vez se tiene en un vector todos los bytes, hacemos los siguiente:

- Analizamos si el primer y segundo campo son “N” y “D” respectivamente.
- Saltamos a la siguiente linea de datos ya que el resto es la repetición del mensaje enviado por nosotros (buscamos un \n).
- Guardamos el Status para tenerlo en cuenta en el postprocesado.
- Saltamos a la siguiente línea.
- Desencriptamos y guardamos el tiempo ya que será útil a la hora de juntar los datos de los tres sensores. Para desencriptar el tiempo, restamos a sus 4 bytes un valor de 0x30 (hexadecimal) para después concatenarlos todos en una misma variable como se muestra en la figura 47.

***Figura 47**: Formato dato de tiempo (LIDAR)*

- Saltamos a la siguiente línea.
- Desencriptamos y guardamos los datos recibidos del sensor. Para realizar el desencriptado de datos hay varias cosas a tener en cuenta:
    - Hay que eliminar todos los \n y tener en cuenta que lo anterior al \n es un CheckSum. (Recordar que los datos vienen en bloques como muestra la Figura 45) .
    - Hay que detectar los “&” ya que quieren decir que el siguiente dato de distancia en un segundo o tercer dato en la misma posición angular.
    - Una vez hechos estos dos pasos, al resto de datos hay que restarles a todos 0x30 (hexadecimal) y después concatenarlos de 3 en 3, quedando como en la figura 48.

***Figura 48**: Formato de datos (LIDAR)*

1.6.6.2. Sensor Inercial

La MTI es una pequeñas y completa unidad de medida inercial con magnetómetros 3D integrados (brújula 3D), con un procesador integrado capaz de calcular alabeo, cabeceo y guiñada en tiempo real (Fig. 49), así como la salida calibrada de aceleración lineal 3D, velocidad de giro (giroscopio) y datos de campo magnético (tierra).

La MTI integra diversas opciones avanzadas IO como RS-422 y una salida de sincronización.

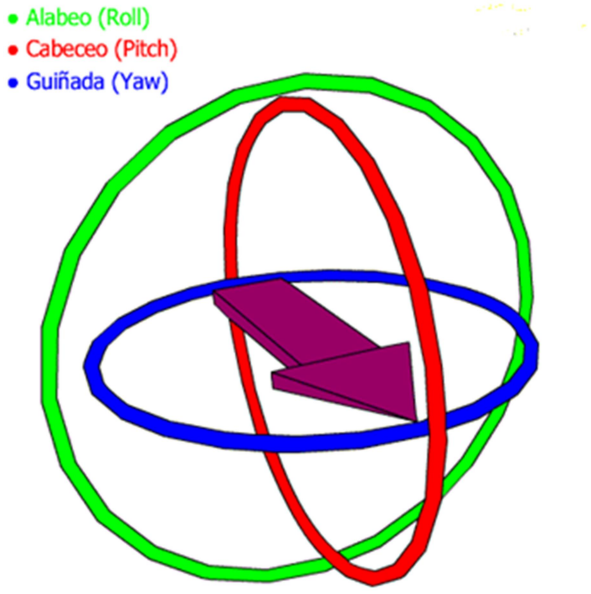


***Figura 49**: Esquema de giro en 3 Dimensiones*

Estados

La MT tiene dos estados, el estado de configuración y el estado de medición. En el Estado de Configuración se pueden leer y modificar distintos ajustes; y en el Estado de Medición el mensaje de salida de la MT depende de la configuración actual del dispositivo.

Hay dos formas diferentes de entrar en el Estado de Configuración o el Estado de medición (Fig. 50). Al encender el MT se inicia el procedimiento de reactivación y éste enviará el mensaje de WakeUp hacia el microprocesador. Si no se realiza ninguna acción, el dispositivo entrará en el estado de medición. Pero si se envía el mensaje WakeUpAck dentro de los siguientes 500ms después de la recepción del mensaje de WakeUp del MT, entonces éste entrará en el Estado de Configuración.

Antes de entrar en el Estado de medición, el mensaje de configuración siempre es enviado al host. Esta es la configuración que se leerá de la memoria interna no volátil y se utilizará en el Estado de medición. Los datos en el mensaje de configuración siempre se pueden utilizar para determinar el modo de salida y los ajustes.

Otra forma de entrar en la configuración o estado de medición es el uso de los mensajes GoToConfig o GoToMeasurement mientras el otro estado está activo.

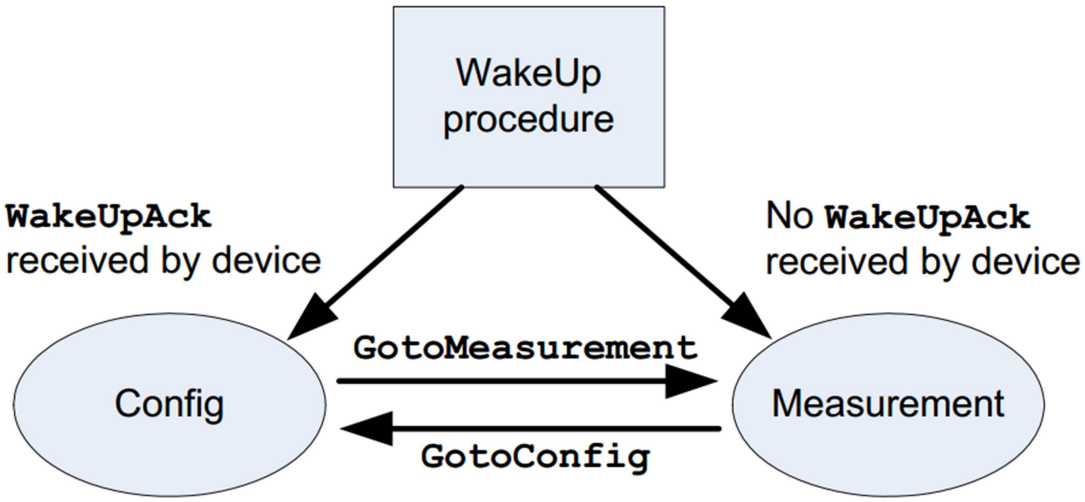


Figura 50: Esquema de funcionamiento interno IMU

Estado de Configuración

El Estado de Configuración se utiliza para obtener y/o establecer diferentes ajustes a la MT. La mayor parte de los ajustes cambiarán la configuración que define la funcionalidad del dispositivo en el estado de medición. Estos ajustes son por ejemplo la velocidad de comunicación en baudios, período de muestreo, el modo de salida, ajustes de salida o propiedades de sincronización.

A la hora de inicializar el dispositivo, todos los ajustes se leen de la memoria no volátil. Todos los ajustes se almacenan en un formato desarrollado por Xsens conocido como eMTS (extended Motion Tracker Specification), junto con otros datos específicos del dispositivo, como los parámetros de calibración. El formato está definido, pero todos los ajustes se pueden manipular mediante el uso de los mensajes de configuración apropiados.

La configuración modificada a través del Estado de Configuración se guarda inmediatamente en la memoria y conservará sus últimos valores, incluso si el dispositivo está desconectado de la alimentación. Algunos mensajes tienen un parámetro adicional que requiere que el usuario especifique expresamente si los nuevos valores deben ser almacenados en la memoria no volátil. De cualquier forma, los cambios de configuración son inmediatos.

Hay una excepción, que es el ajuste de velocidad de transmisión. El nuevo ajuste no será utilizador de inmediato, sino que se va a utilizar a partir del próximo encendido o después de un reinicio del software.

Estado de medición

En el Estado de Medición el MT envía los datos hacia el host en función de los valores de configuración definidos en el Estado de configuración. Un solo mensaje, MTData, se utiliza para todos los modos de salida. Por eso es importante que el anfitrión sepa cómo está configurado el dispositivo. La configuración actual determinará cómo deben interpretarse los

datos del mensaje. Un mensaje especial, *Configuration*, contiene la información relevante con la que con los datos recibidos por el anfitrión en el estado de medición se puede interpretar sin ambigüedades el mensaje. Al registrar los mensajes MTData, es aconsejable incluir el mensaje de configuración en la cabecera de información para el análisis futuro o post-procesamiento.

Si el host no responde al mensaje de WakeUp en el encendido (o después de la emisión de un mensaje de reinicio) el MT entrará automáticamente en el Estado de medición. Justo antes de entrar en el estado, se enviará el mensaje de configuración. Los valores de configuración son leídos de la memoria no volátil y son utilizados durante la medición. La configuración por defecto de la MT se muestra en la Tabla 11.

***Tabla 11**: Configuracion de serie del MTi*

Property	Value
Output mode	Orientation output
Output settings	Orientation in quaternion mode Sample counter enabled
Sample frequency	100 Hz
Baud rate	115k2 bps
Output skip factor	0
SyncIn	Disabled
SyncOut	Disabled

El Estado de medición normalmente no se utiliza para cambiar la configuración. Para cambiar la configuración del dispositivo se debe entrar en el Estado de Configuración, para lo que el usuario debe enviar primero el mensaje GoToConfig.

Comuniación

La comunicación con el MT se realiza mediante mensajes que se construyen de acuerdo a una estructura estándar. El mensaje tiene dos estructuras; una con una longitud estándar y otra con longitud extendida. El mensaje de longitud estándar tiene un máximo de 254 bytes de datos y es el utilizado con mayor frecuencia. En algunos casos, se necesitará utilizar el mensaje de longitud extendida si el número de bytes de datos supera 254 bytes.

Un mensaje MTComm (longitud estándar) contiene los siguientes campos (Fig. 51):

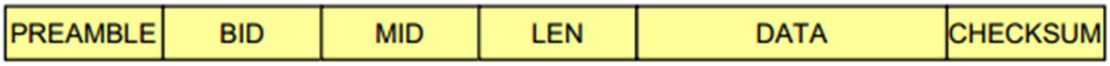


***Figura 51**: Esquema de mensaje con longitud estandar (IMU)*

Un mensaje MTComm (longitud extendida) contiene estos campos (Fig. 52):

***Figura 52**: Esquema de mensaje con longitud extendida (IMU)*

En la Tabla 12 se describe cada uno de los campos contenidos en el mensaje.

Tabla 12: Estructura del mensaje enviado por el MTi

Field	Field width	Description
Preamble	1 byte	Indicator of start of packet → 250 (0xFA)
BID	1 byte	Bus identifier or Address → 255 (0xFF)
MID	1 byte	Message identifier
LEN	1 byte	For standard length message: Value equals number of bytes in DATA field. Maximum value is 254 (0xFE) For extended length message: Field value is always 255 (0xFF)
EXT LEN	2 bytes	16 bit value representing the number of data bytes for extended length messages. Maximum value is 2048 (0x0800)
DATA (standard length)	0 – 254 bytes	Data bytes (optional)
DATA (extended length)	255 – 2048 bytes	Data bytes
Checksum	1 byte	Checksum of message

Preamble

Cada mensaje comienza con el Preamble. Este campo siempre contiene el valor 250 (= 0xFA).

BID

El campo BID (dirección ID del bus) está incluido en el formato del mensaje para que sea compatible con el *XbusMaster*, el cual conecta a múltiples *MotionTrackers*.

Una MT-independiente (sin conexión a un Xbus) tiene un valor BID de 1 (0x01) que indica "primer dispositivo". Un dispositivo MT independiente también es, sin embargo, el "Master" en su propio bus y también puede, por lo tanto, ser abordado mediante el valor BID 255 (0xFF) indicando que es un "Master".

Un MT solamente reconocerá un mensaje (respuesta) si es abordado con un BID válido. Un MT siempre reconocerá un mensaje con el mismo BID que se ha utilizado para abordarlo. Esto significa que un mismo dispositivo puede ser abordado utilizando tanto un BID de 255 (0xFF), como de 1 (0x01), y éste responderá con el BID correspondiente. Sin embargo, hay que tener en cuenta que los mensajes generados por el propio MT (no en reconocimiento de una petición) siempre tendrán un BID de 255 (0xFF). En la práctica, el único mensaje para los que esto ocurre es en el mensaje MTData.

MID

Este campo identifica el tipo de mensaje. Véase el Apéndice 1 Lista de mensajes de Referencia.

LEN

Especifica el número de bytes de datos en el campo de datos de mensaje de longitud estándar. Si en este campo se especifica el valor 255 (= 0xFF), el mensaje se interpreta como un mensaje

de longitud extendida y los siguientes dos bytes se utilizarán para especificar el número de bytes en el campo de datos. Si es cero, no existe campo de datos.

EXT LEN

Este campo tiene un tamaño de 16 bits, y representa el número de bytes en el campo de datos de un mensaje de longitud extendida.

DATA

Este campo contiene los datos y tiene una longitud variable especificada previamente en el campo LEN o EXT LEN. La interpretación de los datos está definida en el mensaje, es decir, en función del valor MID el significado de los datos es diferente. Los datos se transmiten siempre en formato big-endian.

Checksum

Este campo se utiliza para detectar errores en la comunicación. Si todos los bytes del mensaje excluyendo el preámbulo se suman y el valor de byte inferior del resultado es igual a cero, el mensaje es válido y puede ser procesado. El valor del checksum del mensaje debe ser incluido en la suma.

Forma de hacer la comprobación:

1. Pasamos a decimales los datos
2. Los sumamos
3. Calculamos el resto a 256
4. Restamos el resto a 256

Mensajes

En general, un mensaje con un cierto valor MID será contestada con un mensaje con un valor de MID que se incrementa en uno, es decir, en la confirmación de recepción. Dependiendo de lo que haya escrito en el mensaje confirmación, el mensaje puede tener un campo de datos (sin longitud fija) o no. Si no se especifica nada, no existe el campo de datos. En algunos casos se devolverá un mensaje de error (MID = 66 (0x42)). Esto ocurre en el caso el mensaje anterior tenga parámetros inválidos, no sea válido, o no se haya podido ejecutar correctamente. El mensaje de error contiene un código de error en el campo de datos.

Como ejemplo de mensaje, vamos a ver cómo se solicitaría el ID de dispositivo de un MT:

Mensaje enviado:

ReqDID = 0xFA 0xFF 0x00 0x00 0x01 (valores hexadecimales)

Mensaje recibido (Reconocimiento):

DeviceID = 0xFA 0xFF 0x01 0x04 HH HL LH LL CS (valores hexadecimales)

El ID de dispositivo solicitado se da en el mensaje de confirmación DeviceID (aquí se muestra como: HH HL LH LL, la suma de comprobación es CS). Como se puede ver el MID (ID del mensaje) del mensaje de confirmación se incrementa en uno en comparación con el mensaje enviado para hacer la petición, ReqDID.

Algunos mensajes tienen el mismo MID y, dependiendo de si el mensaje contiene o no campo de datos, el significado difiere. Este es el caso con todos los mensajes que se refieren a la configuración de variables. Por ejemplo, el MID del mensaje de solicitud del modo de salida (ReqOutputMode) es el mismo que el mensaje que establece el modo de salida (SetOutputMode). La diferencia entre los dos mensajes es que el campo de longitud del ReqOutputMode es cero y para SetOutputMode es distinto de cero.

Véase en el siguiente ejemplo la diferencia entre los mensajes de petición.

ReqOutputMode = 0xFA 0xFF 0xD0 0x00 0x31 (valores hexadecimales)

SetOutputMode = 0xFA 0xFF 0xD0 0x02 MH ML CS (valores hexadecimales)

MH y ML son datos que representan el modo actual. CS representa el valor de suma de comprobación (Checksum).

Ángulos de Navegación

Si se tiene un sistema de coordenadas móvil respecto de uno fijo, en tres dimensiones, y se desea dar la posición del sistema móvil en un momento dado, hay varias posibilidades de hacerlo. Una de ellas son los ángulos de navegación.

Los ángulos de navegación son un tipo de ángulos de Euler usados para describir la orientación de un objeto en tres dimensiones.

Los ángulos de navegación, llamados en matemáticas ángulos de Tait-Bryan (Fig. 53), son tres coordenadas angulares que definen un triedro rotado desde otro que se considera el sistema de referencia. Se definen matemáticamente de forma similar a los ángulos de Euler, pero en vez de usar como línea de nodos el corte entre dos planos homólogos (por ejemplo el XY es el homólogo del xy), se utilizan dos planos no homólogos (por ejemplo XY e yz).

Por ejemplo, en el dibujo adjunto que usa el convenio ZXY, se definen mediante los planos xy (plano perpendicular al eje "z" usado en la primera rotación) del sistema de referencia, y el plano ZX del sistema móvil (plano perpendicular al eje "Y" de la última rotación).

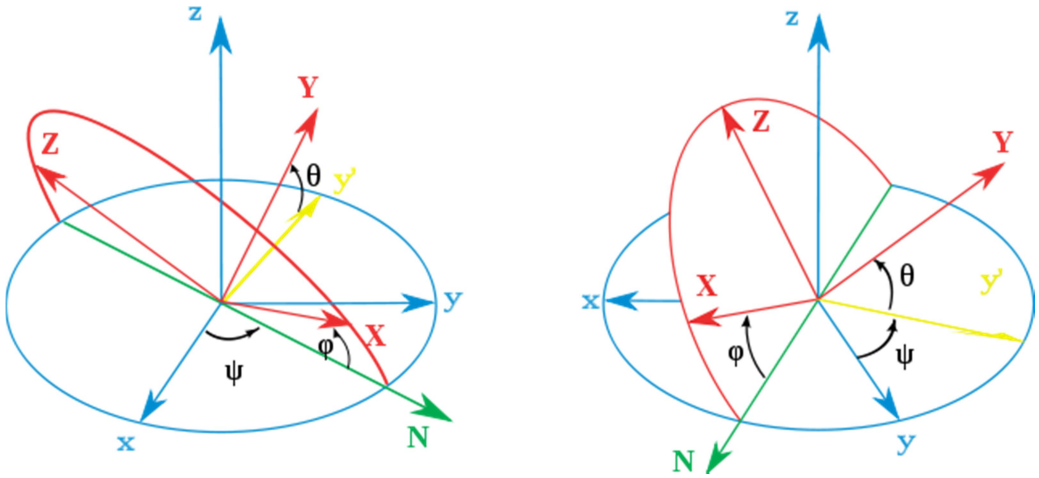


***Figura 53**: Esquema explicativo de los ángulos de Tait-Bryan*

Los tres ángulos son la dirección (yaw), elevación (pitch) y ángulo de alabeo (roll).

Estos tres ángulos son equivalentes a tres maniobras consecutivas. Dado un sistema de tres ejes fijos en el aeroplano, llamados eje de guiñada (yaw), de cabeceo (pitch) y de alabeo (roll), existen tres rotaciones principales, normalmente llamadas igual que el eje sobre el que se producen, que permiten alcanzar el sistema del aeroplano desde el sistema de referencia. Tienen que venir dadas en ese orden y ser realizadas en ese orden, ya que el resultado final depende del orden en que se apliquen.

Cabeceo: es una inclinación del morro del avión, o rotación respecto al eje ala-ala.

Alabeo: rotación respecto de un eje morro-cola del avión.

Guiñada: rotación intrínseca alrededor del eje vertical perpendicular al avión.

Son tres rotaciones intrínsecas, es decir, relativas al sistema móvil.

Los ángulos de navegación, llamados deriva (normalmente representado por la letra Ψ), inclinación (normalmente θ) y alabeo (ϕ), corresponden a los valores de estas tres rotaciones principales.

Cálculo de la orientación

Los datos del MTi están basados en dos sistemas de coordenadas. Primero, el sistema de coordinadas fijas con el MTi se representará aquí con el nombre S y viene representado en la figura 54. En segundo lugar, existe otro sistema de referencia que podemos denotar como G.

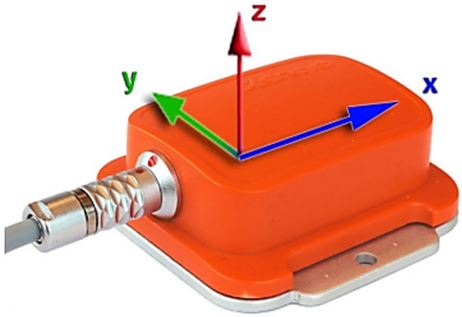


***Figura 54**: Dispositivo MTi de Xsens*

Hay tres maneras de obtener los datos de orientación de giro del MTi. Todas ellas describen la rotación, manteniendo el centro de coordenadas fijo, de S a G.

$$\vec{x}_G = R_{GS}\vec{x}_s \ (1)$$

- Dando los elementos de la matriz de rotación R_{GS}, lo que significa proporcionar 9 floats (números de punto flotante);

- Dando los ángulos de Tait-Bryan (también llamados de Euler o de Cardan): roll, pitch y yaw (cabeceo, alabeo y guiñada). Esto significa proporcionar 3 floats pero requiere cálculos

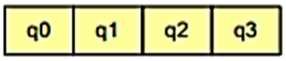
superiores posteriormente (hay que calcular senos y cosenos) por parte del procesador, es decir, requiere menos flujo de datos y menos memoria pero más procesado.

- Mediante el uso de cuaterniones.

En la siguiente Figura 55 podemos ver que el sensor nos posibilita las 4 salidas, y tan solo tendremos que escoger cual es la que queremos en la configuración de salida. También podemos ver que los datos son de 4 bytes cada uno.

Calibrated data output mode (36 bytes)
Contains the calibrated data output of the accelerations, rate of turn and magnetic field in X, Y and Z axes in floats.

Orientation data output mode – quaternion (16 bytes)
Contains the q0, q1, q2 and q3 quaternions, in floats, that represent the orientation of the MT.



Orientation data output mode – Euler angles (12 bytes)
Contains the three Euler angles, in floats, that represent the orientation of the MT

Figura 55: Posibles salidas del sensor inercial

La ecuación (1) nos permite pasar del sistema S al G. Podría ocurrir que conociésemos la matriz, sin saber cómo se calcula, y así podríamos hacer el cambio. Sin embargo, no toda matriz es una matriz de rotación: existe una relación algebraica entre sus componentes para que sea una rotación ($RR^T = R^TR = I$; det R = +1). Por eso, además de que esto significa que los grados de libertad son menos que el número de coordenadas, esto implica que podemos describir el giro con menos de 9 parámetros. En la práctica, además, estos parámetros condensan mejor el significado físico del giro. Así, tenemos que podemos definir cualquier rotación como la operación consecutiva de 3 rotaciones elementales:

- Una rotación alrededor del eje X_G, definido entre -180º y 180º , que llamamos roll o cabeceo ϕ,
- Una rotación alrededor del eje Y_G, definido entre -90º y 90º , que llamamos pitch o alabeo θ, y
- Una rotación alrededor del eje Z_G, definido entre -180º y 180º, que llamamos yaw o guiñada Ψ.

Esto se expresa matemáticamente mediante la siguiente relación

$$\begin{pmatrix} \cos\theta\cos\Psi & \sin\varphi\sin\theta\cos\Psi - \cos\varphi\sin\Psi & \cos\varphi\sin\theta\cos\Psi + \sin\varphi\sin\Psi \\ \cos\theta\sin\Psi & \sin\varphi\sin\theta\sin\Psi + \cos\varphi\cos\Psi & \cos\varphi\sin\theta\sin\Psi - \sin\varphi\cos\Psi \\ -\sin\theta & \sin\varphi\cos\theta & \cos\varphi\cos\theta \end{pmatrix} \quad (2)$$

Por tanto, si queremos describir una rotación, nos pueden dar la matriz R ya construida, es decir, a través de sus 9 componentes, y la operación a realizar será simplemente la indicada por la ecuación (1) si lo que queremos es hacer el cambio de coordenadas. Si queremos ser más económicos en nuestra transmisión de esa información, podemos sintetizarla en 3 parámetros, pero pagamos el precio de tener que construir la matriz a través de la ecuación (2) que implica el cálculo de senos y cosenos. Si tenemos que transmitir esta información entre nuestros subsistemas, lo haremos más rápidamente y menos uso de memoria con 3 parámetros que con 9, pero nuestro procesador tendrá que hacer más cálculos en el primer caso o podemos trabajar con cuaterniones.

Los cuaterniones son una generalización del número complejo j, los cuales nos permiten describir una rotación en 3D a través de 4 parámetros y el cálculo $\overrightarrow{x_G}$ a partir de $\overrightarrow{x_S}$ sin realizar tantas operaciones como las involucradas en (2). Así, tenemos que si el cuaternión viene dado por la "suma" de un escalar q_0 y un vector $q = q_0 + \vec{q}_i$. En este contexto, es posible representar la rotación como descrita por un cierto cuaternión.

$$\vec{x}_G = [q_0 + \vec{q}_i] \cdot \vec{x}_S [q_0 - \vec{q}_i] \text{ (3)}$$

Esta ecuación (3) sustituye a (1). Ahora, poniendo Q como función de un cierto giro α alrededor de un cierto eje dado por el versor $\hat{u}$ (de nuevo se tienen tres grados de libertad: dos para $\hat{u}$ y otro para α):

$$Q = \cos\frac{\alpha}{2} + \hat{u} \cdot \sin\frac{\alpha}{2}$$

Si trabajamos con cuaterniones estamos en una situación intermedia a la descrita en los casos anteriores: tenemos 4 parámetros y tenemos que realizar más cálculos que con la **R** ya conocida pero menos que si nos dan los ángulos de Tait-Bryan.

En conclusión, como para este proyecto necesitamos una velocidad rápida de envío y recepción de datos, y el procesado se realizará posteriormente con Matlab, la opción elegida será trabajar con los ángulos de Tait-Bryan.

- **Código**

El código para la adquisición de datos de sensor inercial tiene la siguiente estructura (Fig. 56).

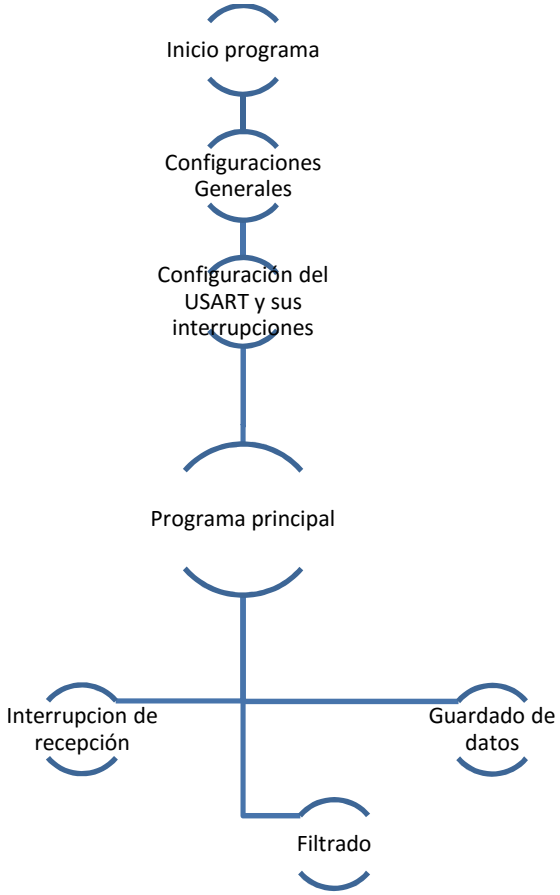


***Figura 56**: Esquema programa de control IMU*

- **Configuraciones Generales**

- Configuración del SysTick para funciones de retardos.

- Configuración del Timer e inicialización de éste.

- Configuración de salidas para control de los LEDs.

- Configuración de entrada para recepción del Botón de usuario.

- Configuración e inicialización de la ranura para la tarjeta SD.

- **Configuración del USART y sus interrupciones:** Se configura el periférico USART para que capte cada dato enviado por la IMU mediante una interrupción.
- **Interrupciones de captura de datos:** La interrupción del USART detecta cada carácter enviado por la IMU y se procesan hasta que se haya recibido un dato completo. En este punto se avisa al programa principal que se ha recibido el dato.

Dato completo: Se consulta el 4º byte de la frase para saber la longitud, con éste sabemos dónde se encontrará el CheckSum y el final de la frase. Entonces comprobamos el CheckSum y, si es correcto, le decimos al programa principal que ha recibido un dato completo.

- **Filtrado y Guardado:** El programa principal está en espera hasta que se reciba un dato completo, entonces es cuando se decodifican (se concatenan de 4 en 4 bytes [Fig. 47] y se convierten a float) y procesan los datos (tan solo campo de datos) y se guardan en la tarjeta microSD para su posterior utilización en Matlab.

1.6.6.3. Sistema satelital de navegación global

Un Sistema satelital de navegación global (su acrónimo en inglés: GNSS) consta de una constelación de satélites que transmite rangos de señales utilizados para el posicionamiento y localización en cualquier parte del globo terrestre, ya sea en tierra, mar o aire. Estos permiten determinar las coordenadas geográficas y la altitud de un punto dado como resultado de la recepción de señales provenientes de constelaciones de satélites artificiales de la Tierra para fines de navegación, transporte, geodésicos, hidrográficos, agrícolas, y otras actividades afines.

Un sistema de navegación basado en satélites artificiales puede proporcionar a los usuarios información sobre la posición y la hora (cuatro dimensiones) con una gran exactitud, en cualquier parte del mundo, las 24 horas del día y en todas las condiciones climatológicas.

Actualmente existen 4 sistemas de este tipo, pero en nuestro caso tan solo dos de estos sistemas son los aceptados por nuestro dispositivo, el Sistema de Posicionamiento Global (GPS) de los Estados Unidos de América y el Sistema Orbital Mundial de Navegación por Satélite (GLONASS) de la Federación Rusa son los únicos que forman parte del concepto GNSS.

GPS

Con el nombre inicial de NAVSTAR (Navigation System with Timing And Ranging), GPS (Global Positioning System - Sistema de Posicionamiento Global) fue desarrollado por el Departamento de Defensa de los Estados Unidos para proporcional de capacidades de navegación en cualquier punto del planeta para el ejército. Desde su implementación GPS se ha convertido en parte fundamental de muchas aplicaciones civiles en todo el mundo, incluido el uso recreativo.

GPS usa 24 satélites en una órbita circular de 20.200 Km de altitud. La constelación fue completada con éxito en Marzo de 1994.

GLONASS

El Sistema Mundial de Navegación por Satélites (GLONASS) proporciona determinaciones tridimensionales de posición y velocidad basadas en las mediciones del tiempo de tránsito y de desviación Doppler de las señales de radio frecuencia (RF) transmitidas por los satélites GLONASS. El sistema es operado por el Ministerio de Defensa de la Federación Rusa y ha sido utilizado como reserva por algunos receptores comerciales de GPS.Es comparable al GPS y emplea 21 satélites en una órbita circular a 19100 Km de altitud.

Dispositivo Utilizado

El uso del GNSS en este proyecto se centrará en la adquisición de la posición del sensor para determinar después las posiciones en el mapeo 3D de la nube de puntos. Los datos guardados serán posteriormente tratados en Matlab para su correcta visualización.

Se utilizará el receptor Leica GPS1200+, ya que se basa en los sistemas tanto GPS como GLONASS y es un sistema RTK, de precisión centimétrica, algo muy importante para este proyecto ya que estos datos serán la primera base a la hora de geolocalizar la nube de puntos.

Configuración y comunicación

El receptor Leica GPS1200+ funciona con alimentación externa proporcionada a través de una batería, y se conecta al microprocesador a través de un conector RS-232 que irá conectado a la placa de expansión.

Los pines RX i TX del GPS están conectados al USART6 del microcontrolador a través de la placa de expansión, el cual se configurará para la lectura de datos con un formato de 8 bits, sin paridad, con 1 bit de stop y a un baudrate inicial de 19200.

Cuando se haya conectado y configurado el puerto USART correctamente, automáticamente el GPS irá enviando datos en formato LLQ a través de su puerto serie TX con un periodo determinado, por defecto 20 veces por segundo, y entonces el microcontrolador leerá dicha información usando el USART previamente configurado.

Sentencias NMEA:

La National Marine Electronics Association (NMEA) es una asociación que ha desarrollado el estándar NMEA 0183, un estándar de mensajes que determina la estructura de la comunicación entre dispositivos electrónicos de la marina. Está definido y controlado por ellos y es comúnmente usado en los GPS. El estándar usa código ASCII enviado a través de protocolo de comunicación serie RS-232 y define como la información se transmite en sentencias.

Hay diferentes sentencias englobadas en este estándar, en nuestro caso se va a utilizar la sentencia LLQ (Tabla 13), propia de Leica, ya que nos proporciona las coordenadas en proyección UTM en metros y así no es necesario realizar la conversión desde coordenadas geográficas (en latitud y longitud) a coordenadas cartesianas.

***Tabla 13**: Comandos LLQ - Leica Local Position and Quality*

$–LLQ,hhmmss.ss,mmddyy,eeeeee.eee,M,nnnnnn.nnn,M,x,xx,x.x,x.x,M*hh<CR><LF>

Field	Description
$--LLQ	Header including talker ID
hhmmss.ss	UTC time of position
mmddyy	UTC date
eeeeee.eee	Grid Easting in metres
M	Units of grid Easting as fixed text M
nnnnnn.nnn	Grid Northing in metres
M	Units of grid Northing as fixed text M
x	Position quality 0 = Fix not available or invalid 1 = No real-time position, navigation fix 2 = Real-time position, ambiguities not fixed 3 = Real-time position, ambiguities fixed
xx	Number of satellites used in computation
x.x	Coordinate quality in metres
x.x	Altitude of position marker above/below mean sea level in metres. If no ortho-metric height is available the local ellipsoidal height will be exported.
M	Units of altitude as fixed text M
*hh	Checksum
<CR>	Carriage Return
<LF>	Line Feed

Standard Talker ID

$GNLLQ,113616.00,041006,764413.024,M,252946.774,M,3,12,0.010,1171.279,M*12
$GPLLQ,113616.00,041006,,,,,,08,,,*4D
$GLLLQ,113616.00,041006,,,,,,04,,,*5D

User defined Talker ID = GN

$GNLLQ,113806.00,041006,764413.021,M,252946.772,M,3,13,0.010,1171.283,M*1A

Código:

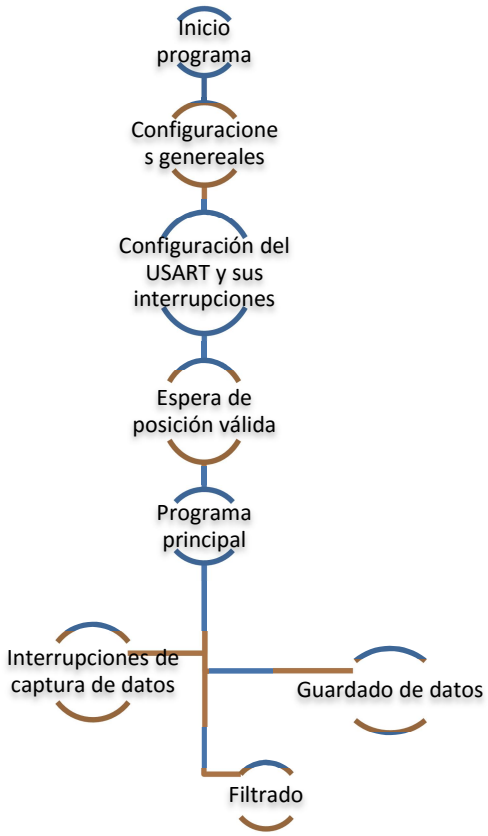


Figura 57: Esquema programa de controlGNSS

- **Configuraciones Generales:**

  - Configuración del SysTick para funciones de retardos.
  - Configuración del Timer e inicialización de éste.
  - Configuración de salidas para control de los LEDs.
  - Configuración de entrada para recepción del Botón de usuario.
  - Configuración e inicialización de la ranura para la tarjeta SD.

- **Configuración del USART y sus interrupciones:** Se configura el periférico USART para que capte cada dato enviado por el GPS mediante una interrupción.
- **Espera de posición válida:** Antes de empezar a guardar los datos captados, se debe esperar a que el GPS reciba señales de los satélites con una precisión alta. Se utiliza una función que realiza la función de espera hasta que detecta un valor del GPS correcto.
- **Interrupciones de captura de datos:** La interrupción del USART detecta cada carácter enviado por el GPS y se procesan hasta que se haya recibido una frase LLQ

completa. En este punto se avisa al programa principal que se ha recibido una frase completa.

- **Filtrado y Guardado:** El programa principal está en espera hasta que se reciba una frase LLQ, entonces es cuando se filtran los valores que nos interesan y se guardan en la tarjeta microSD para su procesado posterior en Matlab.

1.6.6.4. Almacenaje de datos

Para el almacenaje de los datos, se plantea la opción de usar una tarjeta microSD o una memoria USB. Se dispone de librerías para ambos y se configuran para que se pueda alternar entre uno u otro mediante una simple modificación del programa principal.

Para trabajar con la memoria USB se usará el puerto USB OTG del microcontrolador con su correspondiente cable (Fig. 58). Se configurarán los pines del microcontrolador pertinentes a la comunicación USB y se conectará la memoria al cable USB OTG:

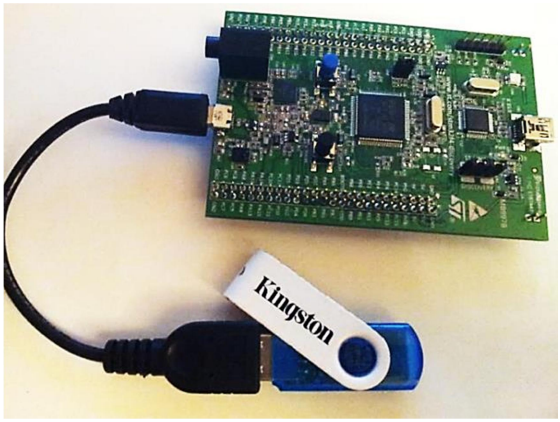


Figura 58: Conexión del microcontrolador con el USB mediante el cable USB OTG

Por otro lado, para trabajar con la tarjeta microSD se usará una la Expansion Board (Fig 59) mostrada al inicio y se comunicará con el protocolo de comunicación SDIO.

Figura 59: Placa de Expansion para STMF4

Se decide realizar un programa de testeo para comparar las velocidades de lectura y escritura de ficheros conjuntamente, y luego lectura y escritura por separado. Se usa un temporizador para controlar el tiempo, el programa funcionará durante un segundo y a partir de aquí se obtendrá las ratios de lectura y escritura del USB y la SD en el microcontrolador. Se realiza la misma prueba 5 veces para ambos y se compara los resultados (Fig. 60).

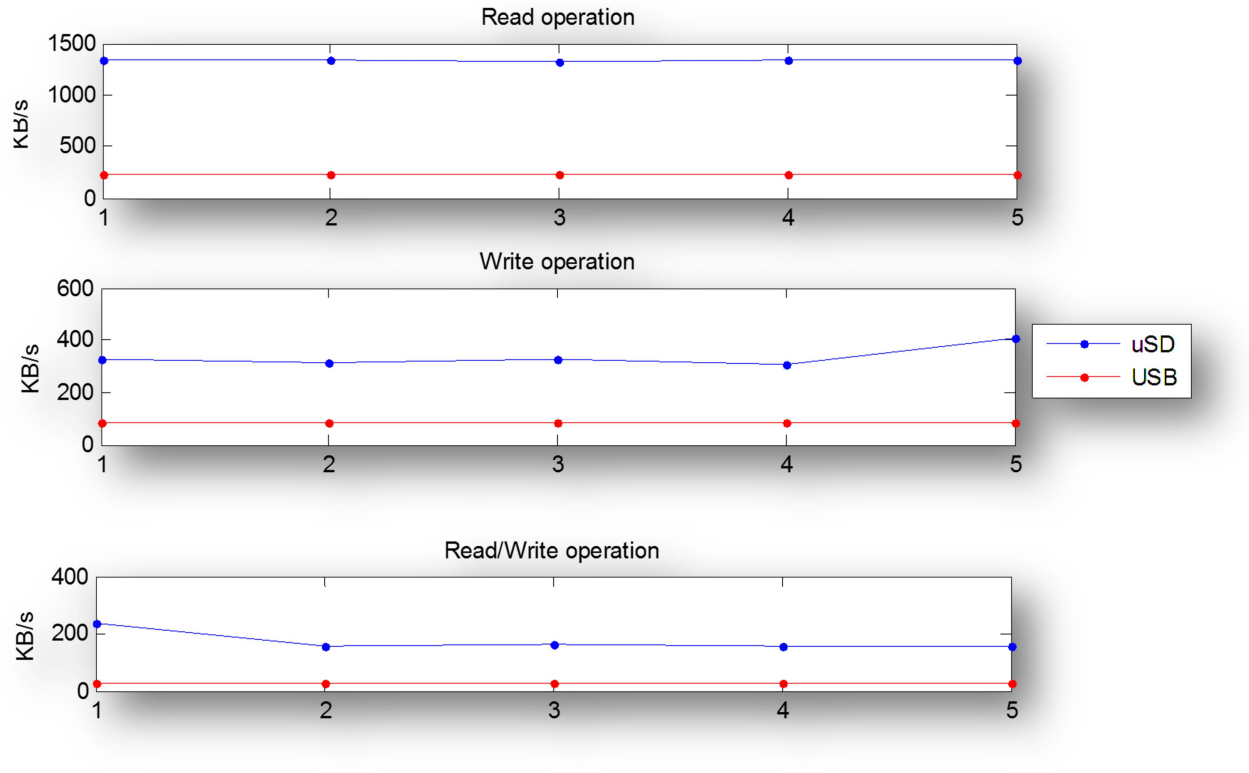


***Figura 60**: Gráficos comparativos de lectura y escritura de 5 pruebas entre SD y USB*

En la Tabla 14 se representa la media de las 5 pruebas realizadas para cada caso:

***Tabla 14**: Comparativa de velocidades entre SD y USB*

	microSD	**USB**
Lectura y escritura media	174.8 KB/s	28.6 KB/s
Lectura media	1330 KB/s	234.2 KB/s
Escritura media	336.6 KB/s	83 KB/s

Claramente se observan mejores resultados en la tarjeta microSD, una velocidad 6 veces superior al USB en lectura y escritura, 5.6 veces superior en lectura y 4 veces superior en escritura.

1.6.6.5. Interacción con el dispositivo

Todos los programas se han creado con la misma estructura de funcionamiento, ya que, en la evolución de este proyecto, es necesario que los 3 dispositivos recojan datos conjuntamente.

El funcionamiento es sencillo, una vez la placa tiene corriente, esta comienza la inicialización de todos los dispositivos y, si no encuentra ningún error, comenzará a captar datos de los periféricos.

Una vez queramos finalizar la recolección de datos, tan solo tenemos que presionar el botón de usuario, que es el pulsador azul que podemos ver en la Figura 61, y la tarjeta se desconectará de forma segura para no perder ningún dato al extraerla.

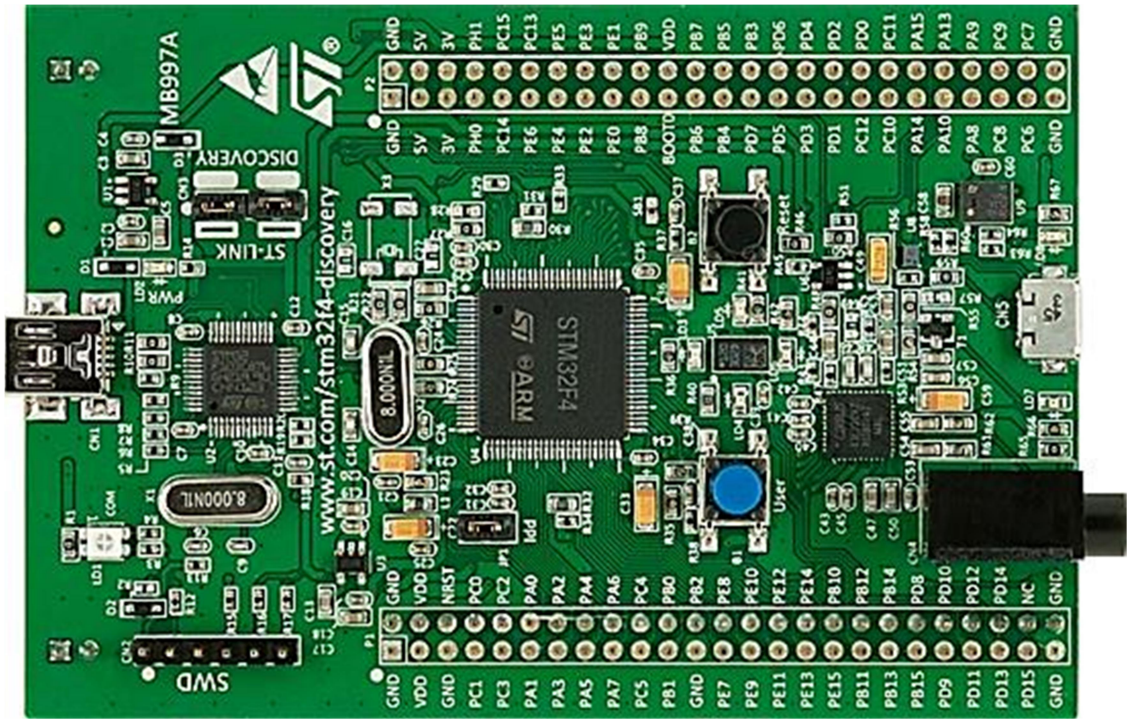


***Figura 61**: Placa STMF4-Discovery*

Todos estos pasos están indicados a través de la línea de comandos del puerto serie y podemos utilizar, por ejemplo, TeraTerm (véase Apéndice II), para conocer el estado del programa en cada momento. Pero ésta idea es para dentro de un laboratorio para realizar las pruebas, cuando estás en campo abierto y no puedes acceder a estos datos, necesitamos otra solución, y para ello están los diferentes 4 LEDs de los que disponemos en la placa.

Señalizaciones LED:

- **RED:**

El programa se ha quedado bloqueado en alguna parte, alguno de los dispositivos no está conectado correctamente, alguno de los periféricos no funciona correctamente o ha habido un error en alguna comunicación. Conviene comprobar todo y realizar un Reset (pulsador negro de la placa [véase Fig.60].

- **ORANGE:**

USART: Parpadea si el USART pierde información, es decir, recibimos los datos más rápido de lo que los podemos procesar. (Esta parte es solo para pruebas de velocidad, una vez acabado el prototipo, se eliminará o se pasará al parpadeo del LED Rojo [parpadearía con el LED Azul encendido).

ETHERNET: Si está encendido, la conexión está establecida.

GNSS: Si está encendido, está esperando señal del satélite correcta, cuando se apaga, es que ha encontrado una señal de buena calidad. (Se puede hacer parpadear el LED Rojo, ya que si parpadea aquí, será con el LED Azul apagado).

- **GREEN:**

Encendido: La placa está funcionando, este LED se enciende justo al comenzar el programa.

Parpadeando: Captura finalizada, se puede retirar la tarjeta SD.

- **BLUE:**

Ha entrado en el programa principal, esto quiere decir que todas las inicializaciones se han llevado a cabo correctamente y el microcontrolador está recibiendo datos del sensor. Se apaga al finalizar la captura.

Se puede observar que una vez se quieran unir los tres, habrá que buscar alguna forma de señalizar los diferentes eventos e incidencias para que no se solapen.

La función de los pulsadores es la siguiente:

- **RESET:** Reinicia el software forzosamente (es como quitar la corriente y volverla a poner).
- **PUSHBUTTON:** Finaliza la captura de datos.

1.7. Resultados finales

Los resultados finales son la adquisición de datos de los tres sensores en paralelo y posterior guardado de estos en un fichero .txt en la tarjeta SD. Todo esto con un formato correcto para facilitar el futuro trabajo con Matlab.

Datos obtenidos con el sensor GNSS:

En el fichero .txt que guarda los datos recogidos por el GPS, se guardarán estos 6 datos mostrados en la Tabla 15 con una tabulación entre ellos, para después introducir un salto de línea y volver a recoger el siguiente dato.

Tabla 15: Datos guardados del sensor GNSS

$--LLQ,hhmmss.ss,mmddyy,eeeeee.eee,M,nnnnnn.nnn,M,x,xx,x.x,x.x,M*hh<CR><LF>

Campo	Descripción
$--LLQ	Encabezado, incluyendo ID del transmisor
hhmmss.ss	Tiempo UTC de la posición
mmddyy	Fecha UTC
eeeeee.eee	Este de cuadrícula en metros
M	Unidades para el Este de cuadrícula, como texto fijo M
nnnnnn.nnn	Norte de cuadrícula en metros
M	Unidades para el Norte de cuadrícula, como texto fijo M
x	Calidad de posición 0 = Fija no disponible o no válida 1 = Sin posición en Tiempo Real, Fija de navegación 2 = Posición en Tiempo Real, ambigüedades sin fijar 3 = Posición en Tiempo Real, ambigüedades fijas
XX	Número de satélites empleados en el cálculo
x.x	Calidad de coordenadas en metros
x.x	Altitud de la posición del marcador sobre/debajo del nivel medio del mar en metros. En caso de no disponer de valores de altura ortométrica, se exportará la altura elipsoidal local.

Datos obtenidos con el sensor Inercial:

El sensor inercial está configurado para que nos devuelva los ángulos de Euler, con lo que en nuestro fichero .txt guardaremos estos datos en filas de 3 en 3 como se muestra en la Tabla 16. Si se guardasen de alguna de las otras dos formas explicadas en el apartado 1.7.6.2. Sensor Inercial, se guardarían o de 4 en 4 o de 9 en 9 y no habría que realizar ningún cambio en el programa, ya que el tamaño de los datos es el mismo (4 bytes).

Tabla 16: Orden de guardado del .txt del sensor inercial

Roll	Pitch	Yaw
4 bytes concatenados	4 bytes concatenados	4 bytes concatenados

Datos obtenidos con el sensor LIDAR:

Como se sabe, el sensor LIDAR es capaz de devolver hasta un máximo de 3 ecos en una misma posición angular. En la tabla 17 podemos ver las 4 posibles respuestas del sensor LIDAR en una misma posición angular, es decir, cada fila de nuestros datos tiene siempre estos tres campos. También se puede ver que, cuando no hay rebote, el valor devuelto es 60000. Esto es porque es el máximo valor posible, con lo que deberemos entender este como nulo.

__Tabla 17__: Respuestas del sensor LIDAR

Descripción	1er eco	2º eco	3er eco
Sin rebote	60000	-1	-1
Un solo rebote	4000	-1	-1
Dos rebotes	4000	5000	-1
Tres rebotes	4000	5000	6000

1.8. Conclusiones

Se concluye que no es viable solventar los problemas reutilizando el sistema actual y añadiendo a este el sensor inercial, ya que sí podría evitar los problemas de precisión actuales pero el sistema seguiría siendo voluminoso, portátil solo con algún tipo de ayuda externa (como puede ser ahora las orugas), poco económico e imposible, por ejemplo, de colocar en un tractor u otro tipo de maquinaria para que hiciese el escaneo automáticamente.

Por otro lado, respecto a cambiar todo el sistema y compactarlo en un microprocesador de gama alta, se concluye que es una solución viable y con grandes posibilidades de futuro. Tanto es así que se estudia conseguir una beca para llevar adelante el proyecto por este camino.

De los sensores ya utilizados en el proyecto anterior (LIDAR y GNSS), se consigue una comunicación precisa y sin pérdida de datos, así como se entiende, decodifica y filtra para el guardado éstos.

Del nuevo sensor, el sensor inercial, se ha generado una librería completa con todas sus posibles funciones, y como con los sensores anteriores, se consigue una comunicación sin pérdida de datos, y una decodificación y guardado sin errores.

La conclusión general es que se seguirá trabajando con el microprocesador, ya que la comunicación con todos los periféricos por separado es perfecta, y ahora se necesitará estudiar el comportamiento de éste trabajando con los tres sensores al mismo tiempo. Además, el sistema es ligero, manejable, preciso y económico (desconociendo los precios de los sensores ya que vienen “impuestos”). Con lo cual, el objetivo del proyecto está cumplido, ya que hemos conseguido un sistema que puedes llevar con una mano si se construye la plataforma adecuada, y hemos conseguido comunicar con los tres sensores recibiendo datos preciosos de estos.

1.9. Propuestas de mejora

Los objetivos a plantear a partir de ahora, formarán parte de la segunda parte del proyecto, el cual se basará en el procesado y tratamiento de los datos para poder generar la nube de puntos 3D.

- Unir en un solo código la obtención de datos de los tres sensores funcionando conjuntamente.

- Hacer un estudio de velocidad de adquisición de datos con los tres sensores funcionando al mismo tiempo.

- Buscar la precisión máxima en el tiempo de obtención de datos de los sensores para poder realizar una impresión de máxima calidad.

- Generar un entorno en Matlab agradable para poder trabajar y desplazarte por las nubes de puntos obtenidas.

1.10. Bibliografía

- Reference Manual STM32F4 (2013), STM. [RM0090]
- Application Note DMA STM32F4 (2015), STM. [AN4031]
- Application Note timer overview (2012), STM. [AN4013]
- Datasheet STM32F4 (2013), STM.
- User Manual STM32F4Discovery, STM. [UM1472]
- STM32F4DIS-BB Quick Start Manual (REV 1.0), Embest Technology Co., LTD
- STM32F4DIS-BB User Manual (REV 1.0), Embest Technology Co., LTD
- Documentación STM (2015), *www.stm.com* (en línea)
- Arduino (2015), *www.arduino.cc* (en línea)

- UTM-30LX-EW DATASHEET 2011 A5 small_2
- UTM-30LX-EW Protocol_en
- UTM-30LX-EW Communication Protocol Specification
  Hokuyo Automatic Co., Ltd. Rev 1.0 [UTM-30LX-EW Protocol_en]
- UTM-30LX-EW Specification (2012)
  Hokuyo Automatic Co., Ltd.
- http://www.hokuyo-aut.jp/

- MT Low-Level Communication Protocol Documentation, document id MT0101P, Revision L, 15 Oct 2010
- MT Software Development Kit Documentation, document id MT0200P, Revision J, 27 May 2009
- MT Manager User Manual, document id MT0216P, Revision L, 17 Enero 2014
- MTi and MTx User Manual and Technical Documentation, Document MT0100P, Revision O, 15 Oct 2010
- MTi-leaflet 2014
- MTi-WhitePaper (The Next Generation Xsens Motion Trackers for Industrial Applications (2014, Versión 1.0), Xsens

- Sistemas de Comunicaciones – Redes II
  ETHERNET Y PROTOCOLOS TCP/IPv4
  Capítulo 1 Curso 2005-1. M.C., Gabriel Gerónimo Castillo
- Puesta en marcha del Sensor Inercial MTi de Xsens (Noviembre 2011), Daniel Morales Tejero

2. Apéndices

2.1. Apéndice I: tabla MID del sensor inercial

En esta sección se da una referencia rápida de todos los mensajes válidos. Para obtener más información sobre el uso de los mensajes ver Documento MT Low-Level Communication.

***Tabla 18**: Mensajes de activación y estado*

Message	MID	Direction	Description
WakeUp	62 (0x3E)	To host	The device sends this message at power- up or reset
WakeUpAck	63 (0x3F)	To MT	If received within 500ms after WakeUp device enters Config State else Measurement State
GoToConfig	48 (0x30)	To MT	Device enters config state
GoToConfigAck	49 (0x31)	To host	Device acknowledges GoToConfig message
GoToMeasurement	16 (0x10)	To MT	Device enters measurement state
GoToMeasurementAck	17 (0x11)	To host	Device acknowledges GoToMeasurement message
Reset	64 (0x40)	To MT	Reset device
ResetAck	65 (0x41)	To host	Acknowledgement of Reset message

***Tabla 19**: Mensajes de información*

Message	MID	Direction	Description
ReqDID	0 (0x00)	To MT	Host request device ID of the device
DeviceID	1 (0x01)	To host	Device acknowledges request by sending its ID
InitMT	2 (0x02)	To MT	Request device ID of the device (for compatibility of Xbus Master users)
InitMTResults	3 (0x03)	To host	Same as DeviceID message
ReqProductCode	28 (0x1c)	To MT	Host request product code of the device
ProductCode	29 (0x1d)	To host	Device acknowledges request by sending its product code
ReqFWRev	18 (0x12)	To MT	Host requests firmware revision of device
FirmwareRev	19 (0x13)	To host	Device acknowledges request by sending its firmware revision
ReqDataLength	10 (0x0A)	To MT	Request the number of data bytes in MTData message
DataLength	11 (0x0B)	To host	Contains the number of data bytes of the MTData message
Error	66 (0x42)	To host	Error message
ReqGPSStatus	166 (0xA6)	To MT	Request GPS Status
GPSStatus	167 (0xA7)	To host	Device returns GPS Status

***Tabla 20**: Mensajes específicos del dispositivo*

Message	MID	Direction	Description
ReqBaudrate	24 (0x18)	To MT	Requests current baud rate of the serial communication
ReqBaudrateAck	25 (0x19)	To host	Device returns baud rate of serial communication

SetBaudrate	24 (0x18)	To MT	Host sets baud rate of serial communication
SetBaudrateAck	25 (0x19)	To host	Device acknowledges SetBaudrate message
ReqErrorMode	218 (0xDA)	To MT	Request error mode
ReqErrorModeAck	219 (0xDB)	To host	Device returns error mode
SetErrorMode	218 (0xDA)	To MT	Host sets error mode
SetErrorModeAck	219 (0xDB)	To host	Device acknowledges SetErrorMode message
ReqLocationID	132 (0x84)	To MT	Request location ID
ReqLocationIDAck	133 (0x85)	To host	Device returns location ID
SetLocationID	132 (0x84)	To MT	Host sets location ID
SetLocationIDAck	133 (0x85)	To host	Device acknowledges SetLocationID message
RestoreFactoryDef	14 (0x0E)	To MT	Restore factory defaults
RestoreFactoryDefAck	15 (0x0F)	To host	Device acknowledges RestoreFactoryDef message
ReqTransmitDelay	220 (0xDC)	To MT	Requests delay value which equals the minimum time between last byte reception and transmission start of acknowledge in RS485 mode
ReqTransmitDelayAck	221 (0xDD)	To host	Device returns delay value
SetTransmitDelay	220 (0xDC)	To MT	Host sets delay value
SetTransmitDelayAck	221 (0xDD)	To host	Device acknowledges SetTransmitDelay message
StoreXkfState	138 (0x8A)	To MT	Stores state of XKF in non-volatile memory of device

__Tabla 21__: Mensajes de sincronización

Message	**MID**	**Direction**	**Description**
ReqSyncInSettings	214 (0xD6)	To MT	Request a SyncIn setting of the device, i.e. SyncIn mode, skip factor or offset
ReqSyncInSettingsAck	215 (0xD7)	To host	Device returns SyncIn setting
SetSyncInSettings	214 (0xD6)	To MT	Host sets a SyncIn setting
SetSyncInSettingsAck	215 (0xD7)	To host	Device acknowledges SetSyncInSettings message
ReqSyncOutSettings	216 (0xD8)	To MTi	Request a SyncOut setting of the device, i.e. SyncOut mode, skip factor, offset or pulse width
ReqSyncOutSettingsAck	217 (0xD9)	To host	Device returns SyncOut setting
SetSyncOutSettings	216 (0xD8)	To MTi	Host sets a SyncOut setting
SetSyncOutSettingsAck	217 (0xD9)	To host	Device acknowledges SetSyncOutSettings message

__Tabla 22__: Mensajes de configuración

Message	**MID**	**Direction**	**Description**
ReqConfiguration	12 (0x0C)	To MT	Request the configuration of device. For logging/quick setup purposes
Configuration	13 (0x0D)	To host	Contains the configuration of device
ReqPeriod	4 (0x04)	To MT	Request current sampling period
ReqPeriodAck	5 (0x05)	To host	Device returns sampling period
SetPeriod	4 (0x04)	To MT	Host sets sampling period (10-500Hz)
SetPeriodAck	5 (0x05)	To host	Device acknowledges SetPeriod message
ReqOutputSkipFactor	212 (0xD4)	To MT	Request how many times the data output is skipped before sending a MTData message

ReqOutputSkipFactorAck	213 (0xD5)	To host	Device returns OutputSkipFactor
SetOutputSkipFactor	212 (0xD4)	To MT	Host sets OutputSkipFactor
SetOutputSkipFactorAck	213 (0xD5)	To host	Device acknowledges SetOutputSkipfactor message
ReqObjectAlignment	224 (0xE0)	To MT	Request object alignment matrix
ReqObjectAlignmentAck	225 (0xE1)	To host	Device returns object alignment matrix
SetObjectAlignment	224 (0xE0)	To MT	To set the object alignment matrix
SetObjectAlignmentAck	225 (0xE1)	To host	Device acknowledges SetObjectAlignment message
ReqOutputMode	208 (0xD0)	To MT	Request current output mode
ReqOutputModeAck	209 (0xD1)	To host	Device returns output mode
SetOutputMode	208 (0xD0)	To MT	Host sets output mode
SetOutputModeAck	209 (0xD1)	To host	Device acknowledges SetOutputMode message
ReqOutputSettings	210 (0xD2)	To MT	Request current output settings
ReqOutputSettingsAck	211 (0xD3)	To host	Device returns output settings
SetOutputSettings	210 (0xD2)	To MT	Host sets output settings
SetOutputSettingsAck	211 (0xD3)	To host	Device acknowledges SetOutputSettings message

***Tabla 23**: Mensajes de obtención de datos*

Message	MID	Direction	Description
ReqData	52 (0x34)	To MT	Host requests device to send MTData message
MTData	50 (0x32)	To host	Message with un-calibrated raw data, calibrated data,orientation data or GPS PVT data

***Tabla 24**: Mensajes de filtrado XKF*

Message	MID	Direction	Description
ReqHeading	130 (0x82)	To MT	Request heading settings
ReqHeadingAck	131 (0x83)	To host	Device returns heading
SetHeading	130 (0x82)	To MT	Host sets output mode
SetHeadingAck	131 (0x83)	To host	Device acknowledges SetHeading message
ResetOrientation	164 (0xA4)	To MT	Resets the orientation
ResetOrientationAck	165 (0xA5)	To host	Device acknowledges ResetOrientation message
ReqUTCTime	96 (0x60)	To MT	Request UTC Time
UTCTime	97 (0x61)	To host	Device return UTC Time
ReqAvailableScenarios	98 (0x62)	To MT	Request available scenarios
AvailableScenarios	99 (0x63)	To host	Device return available scenarios
ReqCurrentScenario	100 (0x64)	To MT	Request current used scenario
ReqCurrentScenarioAck	101 (0x65)	To host	Device return current scenario
SetCurrentScenario	100 (0x64)	To MT	Host set current scenario
SetCurrentScenarioAck	101 (0x65)	To host	Device acknowledges SetCurrentScenario
ReqGravityMagnitude	102 (0x66)	To MT	Request current used gravity magnitude
ReqGravityMagnitudeAck	103 (0x67)	To host	Device returns current gravity magnitude
SetGravityMagnitude	102 (0x66)	To MT	Host set gravity magnitude
SetGravityMagnitudeAck	103 (0x67)	To host	Device acknowledges SetGravityMagnitude
ReqleverArmGPS	104 (0x68)	To MT	Request current used lever arm GPS
ReqleverArmGPSAck	105 (0x69)	To host	Device returns current lever arm GPS

SetleverArmGPS	104 (0x68)	To MT	Host set current lever arm GPS
SetleverArmGPSAck	105 (0x69)	To host	Device acknowledges SetLeverArmGPS
ReqMagneticDeclination	106 (0x6A)	To MT	Request current used magnetic declination
ReqMagneticDeclinationAck	107 (0x6B)	To host	Device returns current magnetic declination
SetMagneticDeclination	106 (0x6A)	To MT	Host set current magnetic declination
SetMagneticDeclinationAck	107 (0x6B)	To host	Device acknowledges SetMagnetic Declination
ReqProcessingFlags	32 (0x20)	To MT	Request filter processing flags
ReqProcessingFlagsAck	33 (0x21)	To host	Device returns processing flags
SetProcessingFlags	32 (0x20)	To MT	Host sets processing flags
SetProcessingFlagsAck	33 (0x21)	To host	Device acknowledges SetProcessing Flags
SetNoRotation	34 (0x22)	To MT	Initiates XKF3 'no rotation' update procedure
SetNoRotationAck	35 (0x23)	To host	Device acknowledges SetNoRotation message

2.2. Apéndice II: Guía de configuración de la frecuencia de envío de datos a través de USART

En esta guía se explica el procedimiento a seguir para el envío de comandas y la configuración del entorno TeraTerm para poder comunicarnos tanto con el microprocesador, lo cual nos servirá para conocer errores, como con los dispositivos que se comunican a través de RS-232, lo que nos servirá para estudiar su protocolo de comunicación.

Para la comunicación entre el PC y los dispositivos se utilizará un cable conversor serie-USB

- Paso 1: Una vez alimentado el dispositivo a comunicar, conectar el pin RX de éste al pin TX del cable conectado al PC y el pin TX al pin RX del cable (Fig. 62).

***Figura 62**: Cable conversor USART/UART a USB (pin RX, TX y GND)*

- Paso 2: abrir el TeraTerm, iniciar una nueva conexión y seleccionar el puerto al que se ha conectado el USB (véase Fig. 63).

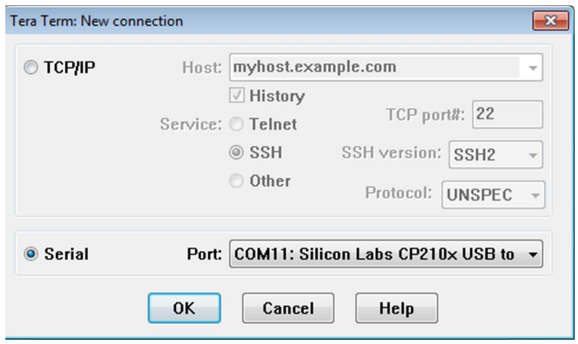


***Figura 63**: Configuración del puerto en el TeraTerm*

- Paso 3: Establecer el Baudrate del programa a la velocidad correcta para la comunicación con el dispositivo (véase datasheet). Ir a Setup-Serial Port (Fig. 64) para realizar la correcta configuración.

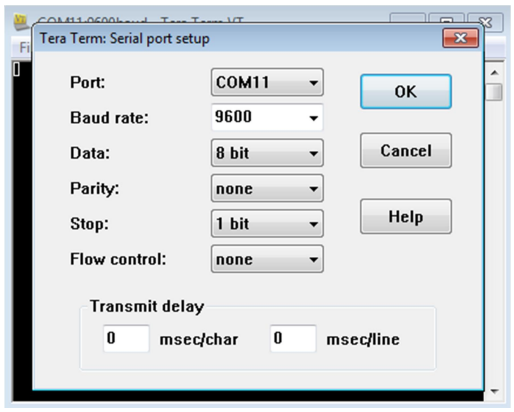


***Figura 64**: Configuración del Baudrate en el TeraTerm*

- Paso 4: Se modificarán los parámetros del terminal del TeraTerm en Setup-Terminal (Fig. 65).

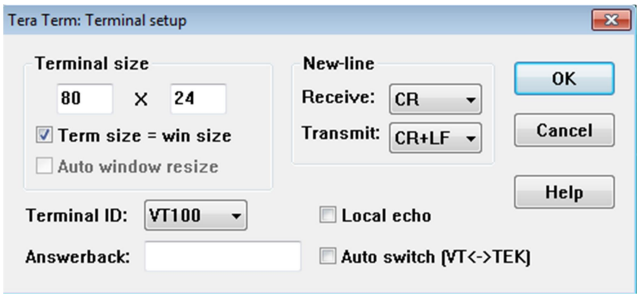


***Figura 65**: Configuración de parámetros del terminal*

- Paso 5: Se escribirá en la línea de comandos del TeraTerm el comando que se desee enviar y presionaremos ENTER (\n\r) [en caso de que queramos comunicar con uno de los sensores, sino, saltaremos este paso].
- Paso 6: Si la configuración es correcta, recibiremos la respuesta del dispositivo o los datos requeridos del microprocesador a medida que avanza su código.

2.3. Apéndice III: Hojas de especificaciones y otros materiales digitales (ver CD adjunto)