



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería
Informática**

**Generación de rutas
utilizando sistemas
empotrados**



Presentado por Alejandro Fernández
Lampreave
en Universidad de Burgos — 13 de febrero
de 2019
Tutor: Dr. Alejandro Merino Gómez



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



D. Alejandro Merino Gómez, profesor del departamento de Ingeniería Electromecánica, área de Ingeniería de Sistemas y Automática.

Expone:

Que el alumno D. Alejandro Fernández Lampreave, con DNI 71290315K, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado Generación de rutas utilizando sistemas empotrados.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 13 de febrero de 2019

Vº. Bº. del Tutor:

D. Alejandro Merino Gómez

Resumen

Este proyecto se centra en utilizar la placa de desarrollo FRDM K64F, con un módulo GPS incorporado, que sea capaz de registrar en una tarjeta microSD la ruta seguida por el dispositivo. La información recogida por el dispositivo será mostrada en una interfaz web.

Para ello se debe desentramar la cadena GPS y almacenarla. La activación/desactivación de la captura GPS se hará de acuerdo a la información recogida por el acelerómetro.

Descriptores

Generación de rutas, módulo GPS, acelerómetro, sistema empujado, registro microSD, estadísticas diarias, aplicación web.

Abstract

This project focuses on using the FRDM K64F development board, with a built-in GPS module, which is capable of recording the path followed by the device on a microSD card. The information collected by the device will be displayed in a web interface.

To do this, the GPS chain must be unlocked and stored. The activation / deactivation of the GPS capture will be done according to the information collected by the accelerometer.

Keywords

Routes generation, GPS module, accelerometer, embedded system, microSD log, daily statistics, web application.

Índice general

Índice general	III
Índice de figuras	V
Introducción	1
1.1. Estructura de la memoria	1
1.2. Estructura de los anexos	2
Objetivos del proyecto	3
2.1. Objetivos generales	3
2.2. Objetivos técnicos	4
Conceptos teóricos	5
3.1. Sistema embebido	5
3.2. Sistema de Posicionamiento Global GPS	6
3.3. Estándar NMEA	6
3.4. Sistema operativo de tiempo real para sistemas empotrados	7
3.5. CRS	8
3.6. EPSG	8
3.7. UART	9
3.8. I2C	9
3.9. SPI	9
3.10. Tarjeta microSD	9
3.11. Acelerómetro	10
Técnicas y herramientas	11
4.1. Kinetis	11

4.2. Lenguaje C	11
4.3. Processor Expert	12
4.4. Termite	13
4.5. Mapbox	14
4.6. MyGeodata	14
4.7. XAMPP	15
4.8. PHP	15
4.9. Python	15
4.10. GitHub	16
4.11. ZenHub	16
4.12. LaTeX	17
4.13. Sonarcloud	17
4.14. Selenium	17
Aspectos relevantes del desarrollo del proyecto	19
5.1. Ciclo de vida del desarrollo	19
5.2. Resoluciones técnicas	22
Trabajos relacionados	41
6.1. Cronología en Google Maps	41
6.2. Endomondo	43
6.3. Strava	43
6.4. Relive	43
6.5. Conclusión	45
Conclusiones y Líneas de trabajo futuras	47
7.1. Conclusiones	47
7.2. Líneas de trabajo futuras	48
Bibliografía	49

Índice de figuras

4.1. Processor Expert, vista de los diferentes componentes instalados.	13
4.2. Terminal termite, ventana en blanco donde imprime la información que recibe.	14
5.3. Conexiones hechas entre los pines Rx y Tx hacia Digital I/O 1 y 0 respectivamente.	20
5.4. Placa de desarrollo FRDMK64F con el módulo GPS integrado y registrando datos.	21
5.5. Acelerómetro de 6 ejes.[1]	23
5.6. Componentes necesarios para permitir el almacenamiento en la tarjeta microSD.	24
5.7. Componentes para poder obtener los datos del acelerómetro. . .	25
5.8. Componentes necesarios del módulo GPS.	25
5.9. Función Imprime encargada de pasar los datos al pc por puerto serie.	26
5.10. Tarea en tiempo real dedicada a recibir e introducir los caracteres en la cadena.	27
5.11. Código necesario para el correcto envío de datos a la tarjeta microSD.	28
5.12. Inicialización de tareas de tiempo real.	29
5.13. Mapa creado con la API de Google.	30
5.14. Empresas que ya trabajan con Mapbox.	30
5.15. Mapa creado con la API de Mapbox.	31
5.16. Información básica que necesita para poder convertir el archivo.	32
5.17. Formulario que habilita la subida de archivos.	32
5.18. Interfaz de subida de archivos familiar para el usuario.	32
5.19. Llamada al script que convierte las coordenadas.	33
5.20. Descompresión del <i>zip</i> generado por MyGeodata.	33

5.21. Documentos con el día y la fecha en el título.	34
5.22. Código que habilita los botones de control, la descarga del mapa y el centrarlo en tu posición.	35
5.23. Página de inicio con los botones de subida del archivo.	36
5.24. Código encargado que calibrar el acelerómetro y capturar los valores de los ejes.	37
5.25. Excepción en PHP que comprueba que el archivo subido sea de texto.	38
6.26. Se muestran todos los trayectos que se realicen en el día.	42
6.27. Se puede seleccionar cualquier día, desde que tenga datos.	42
6.28. Ruta completa tradicional realizada en 2D.	44
6.29. Vídeo en 3D recorriendo todos los puntos por los que se ha pasado.	44

Introducción

El proyecto se centra en la generación de rutas a través de un sistema embebido mediante la recolección de datos que aporta el módulo GPS adherido a la placa. Este desarrollo busca crear una herramienta que recolecte y almacene las coordenadas GPS y represente en un mapa la ruta generada. De esta manera se permite una visualización completa de todos los puntos almacenados y un histórico de los datos recogidos en diferentes días.

1.1. Estructura de la memoria

- **Objetivos del proyecto:** plantea los objetivos que se pretenden solucionar.
- **Conceptos Teóricos:** exposición de las diferentes cuestiones teóricas vistas durante el proyecto.
- **Técnicas y herramientas:** listado de las diferentes técnicas y herramientas utilizadas a lo largo del proyecto.
- **Aspectos relevantes del desarrollo del proyecto:** muestra los aspectos con más relevancia en la realización del proyecto.
- **Trabajos relacionados:** muestra algunos trabajos con cierto grado de similitud a la generación de rutas.
- **Conclusiones y líneas de trabajo futuras:** conclusiones obtenidas en la realización del proyecto y algunas posibles ideas futuras de mejora.

1.2. Estructura de los anexos

- **Plan de Proyecto Software:** planificación temporal y estudio de la viabilidad legal y económica.
- **Especificación de requisitos:** expone los requisitos y objetivos establecidos al principio del desarrollo.
- **Especificación de diseño:** recoge la información del diseño arquitectónico y el procedimental.
- **Manual del programador:** incluye los aspectos más importantes para el programador: cómo instalar las herramientas, compilar, instalar y ejecutar.
- **Manual de usuario:** guía de información básica para el usuario, para un uso correcto de la aplicación.

Objetivos del proyecto

En este capítulo trataremos de abordar cuáles han sido los objetivos estipulados en el proyecto que denominaremos objetivos generales y también nombraremos como objetivos técnicos aquellos que vienen dados a la hora de llevar a cabo los anteriores.

2.1. Objetivos generales

- Almacenar las coordenadas suministradas por el GPS en la tarjeta microSD.
- Desarrollar una web que muestre la ruta almacenada.
- Poder almacenar los datos en archivos de texto a modo de histórico.
- Poder descargar una imagen del mapa con la ruta.

2.2. Objetivos técnicos

- Cablear los pines Tx y Rx tanto de envío como de recepción de datos del módulo GPS con los pines digitales I/O, 0 y 1 del mismo.
- Desentramar la cadena GPS transmitida por el módulo GPS, adherido a la placa de desarrollo.
- Enviar los datos del módulo GPS en modo soft serial a través del puerto (UART3) y por puerto serie (UART0) al ordenador.
- Guardar los mensajes NMEA recibidos en la tarjeta microSD.
- Incluir un sistema operativo de tiempo real (RTOS) para la gestión de los procesos.
- Desarrollar una interfaz web intuitiva y clara para el usuario que permita la subida de ficheros.
- Los archivos deberán ser convertidos de NMEA a GeoJSON.
- Los ficheros se guardarán en el servidor acompañados de la fecha y hora.
- La ventana de la interfaz debe adaptarse a diferentes tamaños y resoluciones de pantalla.
- La aplicación podrá utilizarse en cualquier dispositivo con un navegador instalado.
- La aplicación deberá representar las coordenadas recogidas en el menor tiempo posible.
- Emplear Git como herramienta de control de versiones incluido en la plataforma GitHub.

Conceptos teóricos

Inicialmente, uno de los apartados que más dificultad presentaba el proyecto era que tendría que manejarlo con un estándar de posicionamiento y se tendría que realizar sobre un sistema embebido, nunca antes había trabajado con uno. A continuación se detallan algunos de los conceptos más importantes que ayudarán a la comprensión del proyecto.

3.1. Sistema embebido

Se denomina *Sistema embebido* [36] a aquel sistema electrónico que está diseñado para un propósito específico. Suelen ser compactos, realizan tareas sencillas y tienen todos o la mayoría de componentes que necesitan para desarrollar su función incorporados.

Las características de un *Sistema embebido* son [6]:

- Deben ser sistemas robustos.
- Deben ser sistemas confiables, es decir que trabaje correctamente (reliability).
- Después de un fallo deben volver a funcionar en un periodo corto de tiempo (maintainability).
- Deben estar disponibles en cualquier momento (availability).
- La comunicación ha de ser confidencial y autenticada.
- Son eficientes desde un punto de vista energético, de peso, de coste y de cantidad de código.
- Habitualmente nos los encontramos sin teclado, ratón ni pantalla.

3.2. Sistema de Posicionamiento Global GPS

Entendemos por *GPS* [7], como un sistema de posicionamiento global de radionavegación, con origen en Los Estados Unidos de América. Ofrece gratuitamente posicionamiento, navegación y cronometría de forma ininterrumpida a toda la tierra y sin número límite de usuarios. Solo es necesario contar con un receptor GPS, el sistema te proporcionará tu localización y hora local, independientemente de las condiciones climáticas y tu posición.

A día de hoy el *GPS* se ha convertido en una herramienta imprescindible en todos los medios de transporte, ya sea tierra mar o aire y es utilizado a diario por los servicios de emergencia a la hora de localizar, coordinar y socorrer. Además, se ha convertido en los últimos años, en un elemento de vital importancia en operaciones bancarias o en las redes energéticas, entre otras, ya que se gana precisión cronométrica.

3.3. Estándar NMEA

NMEA [18] corresponde a la abreviatura National Marine Electronics Association es una asociación creada por un grupo de marcas relacionadas con la electrónica naval, a la que de forma progresiva se fueron adhiriendo más grupos y colectivos oficiales. [11]

En la versión de NMEA0183 ya existían GPS enviando información a través de este protocolo, sin embargo, solo estaba diseñado para enviar o recibir una señal y en ningún caso era capaz de recibir dos señales distintas. Para tal efecto se ayudaba de lo que se conoce como un distribuidor NMEA, cuya función es la de recibir las señales de cada dispositivo, procesarlas, y emitir una nueva señal con la información en una misma cadena.

Con la nueva versión NMEA 2000, podemos prescindir de este distribuidor ya que soporta la conexión de diferentes equipos enviando y recibiendo información simultáneamente.

El propósito del estándar NMEA a día de hoy principalmente reside en dar un mismo soporte común a una gran variedad de GPS, en lugar de que cada uno tenga que desarrollar el suyo propio. Lo que incrementa la dificultad del estándar es que no hay solo una cadena con información, hay varias con distintos propósitos que, aunque no siempre se aprovechan todas.

Vamos a poner un ejemplo de una de las cadenas más frecuentes en

3.4. SISTEMA OPERATIVO DE TIEMPO REAL PARA SISTEMAS EMPOTRADOS

7

NMEA: [5]. \$GPGGA,172009.00,4205.7041778,N,09054.4977280,W,4,14,1.00,394.123,M,28.199,M,0.10,0000*39

Todas las cadenas empiezan por \$ y los campos van separados por comas.

- GP: representa que se trata de una posición GPS.
- 172009.00: hora, 17:20:09 UTC.
- 4205.7041778: latitud.
- N: punto cardinal de la latitud, Norte.
- 09054.4977280: longitud.
- W: punto cardinal de la longitud, Oeste.
- 4: precisión de la coordenada (mín 1-4 max).
- 14: número de satélites usados.
- 1.00: Dispersión de la precisión horizontal o incertidumbre (HDOP).
- 394.123: altura de la antena.
- M: unidad de medida de la altura de la antena.
- 28.199: altura geoidal (restando la altura de la antena obtenemos la altura elipsoidal HAE).
- M: unidad usada en la altura geoidal.
- 0.10: segundos desde la última actualización (opcional).
- 0000: ID de la estación (opcional).
- *39: checksum.

Aparte del mensaje NMEA \$GPGGA hay otros frecuentemente utilizados como \$GPGLL y \$GPRMC que son muy similares y también incluyen las coordenadas. Pero hay otros menos utilizados capaces de aportar diferente información que no detallaremos por no sernos de utilidad. Como por ejemplo \$GPGSA, \$GPGSV, \$GPVTG o \$GPGST.

3.4. Sistema operativo de tiempo real para sistemas empotrados

Hablamos de sistema operativo de tiempo real [9] o RTOS cuando la importancia no recae sobre los usuarios si no sobre los procesos, de forma que se le exige dar una respuesta en un tiempo previamente determinado, si no, diremos que ha fallado. Los RTOS disponen de mecanismos de comunicación, sincronización, acceso a recursos compartidos y sincronización. Destacan: FreeRTOS, MQx y VxWorks.

Algunas de sus características son [43]:

- El uso de memoria es más bien escaso.
- Un evento en el hardware puede ejecutar una tarea.
- El código sirve para múltiples arquitecturas de CPU.
- Usualmente poseen tiempos de respuesta predecibles.

FreeRTOS

FreeRTOS [14] es uno de los sistemas operativos de tiempo real existentes, apto para al menos 35 plataformas de microcontrolador diferentes. Una de sus principales ventajas reside en que se encuentra distribuida bajo la licencia gratuita y de libre uso MIT [42].

Está escrito en C, para facilitar su legibilidad, portabilidad y mantenibilidad. Diseñado para ser simple y pequeño, lo que facilita su utilización en sistemas empujados, consta de métodos para la creación de hilos (con prioridades), semáforos, temporizadores etc. Las tareas se ejecutan en función de la prioridad y se van rotando por medio de una interrupción.

3.5. CRS

La función de un sistema de coordenadas de referencia[32] es definir, junto con la ayuda de las coordenadas, la forma en la que un mapa bidimensional se relaciona con lugares reales de la tierra. Este mapa bidimensional será proyectado en un GIS. (programa informático que se utiliza para relacionar datos geográficos). Los CRS pueden dividirse en sistemas de referencia de coordenadas geográficas o proyectados. El sistema de coordenadas geográficas utiliza los grados de longitud, latitud y en ocasiones altitud para representar la situación. La latitud se representa tomando como referencia la línea del ecuador y se divide a cada hemisferio en 90 secciones. La longitud se basa en el meridiano de Greenwich y son líneas perpendiculares al ecuador. El sistema de referencia de coordenadas proyectado normalmente se define en un plano bidimensional denominado plano XY. X será el eje horizontal e Y el eje vertical. En ocasiones se añade un tercer eje, el Z, por lo que el sistema se definirá en un plano tridimensional.

3.6. EPSG

Acrónimo de European Petroleum Survey Group,[8] es una organización que se relaciona con la industria del petróleo en Europa. Este organismo desarrolló un archivo de parámetros geodésicos que contiene información

sobre las proyecciones cartográficas de todo el mundo. En la actualidad, las tareas del EPSG las lleva a cabo el Subcomité de Geodesia del Comité de Geomática de la OGP. Este archivo es de gran importancia ya que es utilizado para la definición de datos de posición en los Sistemas de Información Geográfica. Será de gran utilidad para las actividades que requieran trabajar con datos espaciales en un ambiente digital.

3.7. UART

Universal Asynchronous Receiver-Transmitter.[22] Se trata de un bus serie muy usado en microcontroladores, encargado de recibir y transmitir información de forma asíncrona entre ordenadores y otros microcontroladores. Es posible su utilización a través de software en caso necesario. Es en el caso concreto de los ordenadores donde se utilizan los “puerto serie” o COM. La velocidad de transmisión con la que se envía y recibe información, es expresada en baudios y mide la cantidad de datos (bits) que se envían en un segundo. Actualmente está siendo sustituido por otras comunicaciones como la SATA o USB, sin embargo, persiste en la industria.

3.8. I2C

Acrónimo de Inter-Integrated Circuit.[23] Estándar de comunicación creado en un principio por Philips para las televisiones, ahora destaca por su implementación en la conexión entre microcontroladores o con periféricos.

3.9. SPI

Serial Peripheral Interface. [24] Otro estándar de comunicación creado por Motorola, en este caso periféricos con comunicación en serie y síncronos. Tiene la peculiaridad de que permite la conexión uno a varios (maestro-esclavos) a través de Full Duplex.

3.10. Tarjeta microSD

También conocida como TransFlash,[20] fue creada por SanDisk en el 2005 con el objetivo de almacenar información digital incorporándolas en diferentes sistemas. Trabajan a 3.3V, apenas llegan a los 15 milímetros en su parte más alargada, pero sin embargo, pueden llegar alcanzar capacidades

de 2TB. Actualmente se pueden adquirir tarjetas de diferentes tamaños velocidades y capacidades. Como desventaja remarcar que no disponen de ningún sistema de protección contra escritura.

3.11. Acelerómetro

Un acelerómetro[29] es una herramienta encargada de medir vibraciones o bien la aceleración que experimenta un cuerpo. La fuerza que se genera con el movimiento ejerce una presión sobre un material piezoeléctrico, que a su vez produce una diferencia de potencial que puede ser expresada en fuerzas G. La carga es directamente proporcional a la aceleración debido a que la masa no varía y depende directamente de la fuerza experimentada($F=m.a$).

Técnicas y herramientas

4.1. Kinetis

Para conseguir almacenar los mensajes NMEA en la tarjeta microSD, se ha utilizado Kinetis Design Studio 3 IDE [15]. Se trata de un software de desarrollo basado en GNU/Eclipse antes utilizado en exclusiva para los servicios de los dispositivos de Freescale Kinetis. Desde la adquisición por parte de NXP ahora presta servicio a multitud de sistemas empujados.

Soporta dispositivos Kinetis basados en Cortex-M, integra Processor Expert y un kit de desarrollo de software. Además soporta diversas herramientas de debugueo, entre ellas: SEGGER J-Link/J-Trace, P&E USB Multilink Universal/USB Multilink Universal FX y CMSIS-DAP. Utiliza la nueva librería nano C que ayuda a reducir la memoria utilizada por el sistema embebido.

4.2. Lenguaje C

El lenguaje C [41] es un lenguaje de programación poco tipificado, de tipo de datos estático y de medio nivel porque tiene las estructuras comunes de los de alto nivel, pero además también tiene estructuras que le habilitan un control a bajo nivel.

Cuenta como principal virtud la eficiencia que su código produce. Normalmente utilizado en la creación de software de sistemas y ciertas aplicaciones. La programación a bajo nivel permite la modificación de registros del procesador, lo que lo convierte en ideal para la programación de sistemas empujados.

4.3. Processor Expert

Processor Expert [16] es un sistema de desarrollo para configurar, crear, optimizar y migrar componentes software para las placas de desarrollo. Los componentes embebidos propios de las placas de desarrollo, son módulos que puedes instalar con Processor Expert y que vienen con una serie de funcionalidades incorporadas especialmente diseñadas para sacar el máximo partido a dicho componente. Cobra un papel muy importante en nuestro proyecto dado que la programación directa de sistemas empotrados es compleja. Se trata de una programación a muy bajo nivel que requiere el acceso y la modificación de registros o incluso llegando a la programación en ensamblador. Con la intención de evitar esto, nace esta herramienta que proporciona una serie de librerías que permiten la programación a más alto nivel.

Cuando lo usamos genera dependiendo del componente, cabeceras, ciertos archivos de configuración o código fuente.

Un componente representa un elemento hardware, un periférico, un algoritmo o cualquier colección lógica de funciones.

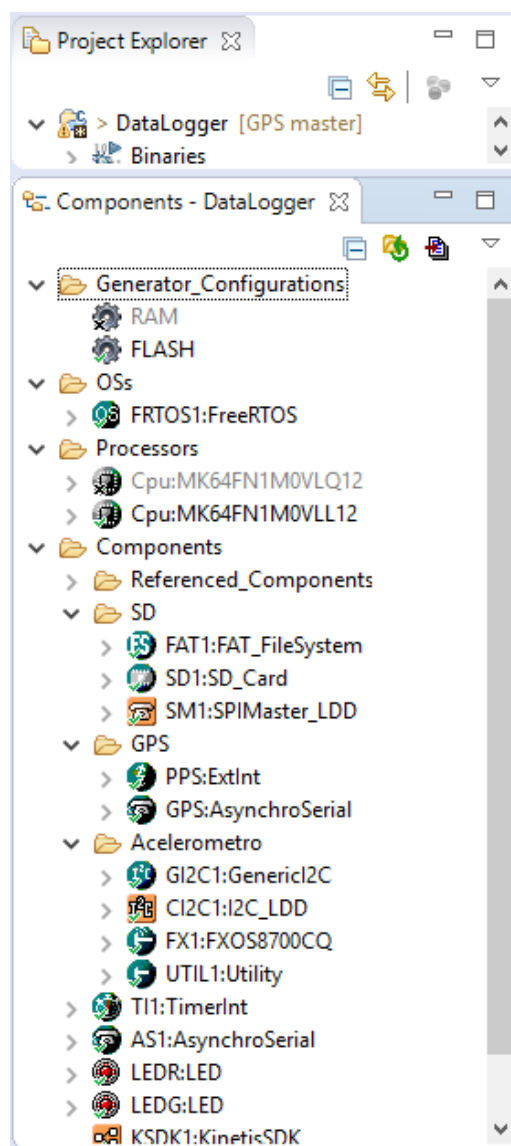


Figura 4.1: Processor Expert, vista de los diferentes componentes instalados.

4.4. Termite

Termite [3] es una terminal que permite la comunicación serie entre pc y placa, haciendo uso de los puertos COM del ordenador. Cuenta con una interfaz similar a la de un chat, en la que va mostrando toda la información que recibe y además cuenta con una línea para poder introducir texto y poder enviarlo.

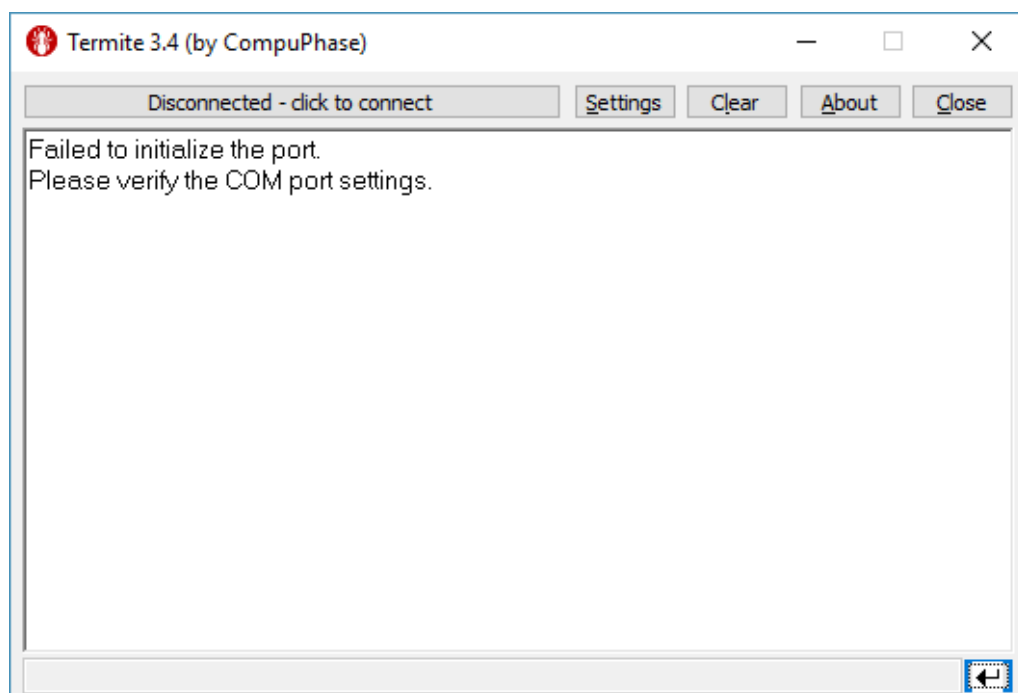


Figura 4.2: Terminal termite, ventana en blanco donde imprime la información que recibe.

4.5. Mapbox

Mapbox [30] es un proveedor de mapas online para páginas web que nació en 2010. Se utiliza para integrar mapas en aplicaciones móviles y web. Es una de las plataformas de datos más elegidas por los clientes por diferentes aspectos. En Mapbox todo es personalizable, puedes escoger cualquier color, ocultar la capa que no quieras del mapa, etc. Tiene una serie de estilos predefinidos, pero también puedes crear el tuyo a través de la herramienta MapboxStudio. Todo su código es open source y puedes encontrarlo en GitHub, lo que te permite ver las funcionalidades que están en desarrollo, notificar cualquier problema, hasta puedes contribuir mandando mejoras. Se puede utilizar en cualquier plataforma, ya que hay SDKs disponibles para la mayoría, Android, iOS, web, etc.

4.6. MyGeodata

Mygeodata [2] es un conversor de datos geográficos en línea que consta de diferentes formatos, CAD, GIS y Raster.

Se divide en tres apartados principales aunque en el proyecto solo se ha aprovechado el conversor:

- *Drive*: en el cual puedes guardar tu conjunto de datos y gestionarlos o ver archivos que hayan compartido otros usuarios.
- *Converter*: es el apartado online donde puedes traducir tus coordenadas al formato de datos que elijas.
- *Map*: donde se muestran nuestros datos en un mapa de forma online.

4.7. XAMPP

XAMPP [4] es una herramienta que permite realizar pruebas de nuestro trabajo en el ordenador sin la necesidad de tener acceso a internet. Es una forma sencilla de instalar Apache. Cuenta con un intérprete del lenguaje PHP, servidores de bases de datos como MySQL, servidores FTP, etc. Su instalación es muy sencilla, basta con descargar y ejecutar un archivo comprimido, con unas pequeñas configuraciones. Además se actualiza de forma regular para que el usuario tenga las últimas versiones de los diferentes servidores que tiene.

4.8. PHP

PHP [25] es un lenguaje *open source* y puede ser incrustado en HTML. Se ejecuta en el servidor web, por lo que puede tener acceso a bases de datos, conexiones de red y otra serie de tareas para crear la página resultado que será la que vea el cliente. Esta página final contendrá únicamente código HTML, por lo que será compatible con cualquier navegador. Que el lenguaje se combine con HTML hace que sea fácil de utilizar, una de sus grandes ventajas, además de ser gratuito, multiplataforma, tenga una gran seguridad y rapidez. Fue creado por Ramus Lerdorf en 1994, pero al ser de código abierto ha tenido varias contribuciones a lo largo de estos años.

4.9. Python

Python [26] es un lenguaje orientado a objetos, que puede utilizarse para distintos objetivos, desde aplicaciones Windows a páginas web. Es muy conocido por varias razones, tiene una gran cantidad de librerías que cuentan con funciones muy útiles, es sencillo y rápido en comparación a otros lenguajes, se puede desarrollar en varias plataformas y es gratuito. Fue

creado por Guido Van Rossum, cuyo objetivo era un lenguaje que hiciese más fácil la orientación a objetos. Las características más destacables del lenguaje son:

- Es de propósito general, puede utilizarse para la creación de varios programas.
- Es multiplataforma, existen diferentes versiones para sistemas informáticos distintos.
- Interpretado, no hay que compilar el código para realizar la ejecución. Es verdad que sí hay una compilación, pero es transparente al programador.
- Es interactivo, podemos ejecutar cada sentencia para ver qué nos muestra.
- Es un lenguaje orientado a objetos.
- Cuenta con una gran cantidad de funciones y librerías que hacen más sencillo el trabajo, evitando tener que programar desde cero.
- La sintaxis utilizada es muy sencilla y clara, se utiliza la tabulación para separar el código y distinguir los bucles o funciones.

Todo esto hace que Python actualmente esté siendo utilizado en empresas como Google, Yahoo, la NASA, etc

4.10. GitHub

GitHub [17] es una plataforma online de desarrollo colaborativo de software para almacenar en la nube proyectos, haciendo uso del sistema de control de versiones Git. Git es un sistema de control de versiones de código abierto, gratuito y distribuido. Ha sido creado para poder manejar cualquier proyecto con eficiencia y velocidad. Github ha sido utilizado en este proyecto como piedra angular, para tener en todo momento el control sobre las distintas versiones.

4.11. ZenHub

ZenHub [44] es una extensión que puede integrarse en GitHub, pero contiene interfaz propia. Con ella nos ayudamos a tener un mayor control sobre nuestros proyectos, con un sistema de paneles, donde podemos ir catalogando las diferentes tareas según la fase de desarrollo en la que se encuentren.

Los diferentes segmentos o *pipelines* en los que se puede encontrar una tarea o *issue* son: *New issues*, *Icebox*, *Backlog*, *In progress Review/QA*, *Done* y *Closed* pero se pueden añadir más.

Además genera estadísticas, tiene un sistema de votación y permite obtener retroalimentación. Su gran baza al igual que GitHub, es que es gratuito para proyectos públicos.

4.12. LaTeX

LaTeX [19] es un sistema utilizado para componer textos, principalmente indicado para aquellos que necesiten de una alta calidad de acabado tipográfico. Para la creación de esta memoria se ha utilizado Texmaker [40].

MiKTeX

Además, junto con Texmaker hemos necesitado MiKTeX [27] de código abierto, tiene compiladores LaTeX y TeX y tiene la posibilidad de instalar unos 800 paquetes diferentes con macros, tipografías etc.

4.13. Sonarcloud

Sonarcloud[37] consiste en una plataforma *online* que nos permite evaluar la calidad del código de nuestro proyecto de forma gratuita. Utiliza herramientas avanzadas que le permiten detectar posibles fallos o vulnerabilidades y que nos pueden ayudar a mejorar nuestro código. De forma general te presentan un panel informativo tras analizar el código con los siguientes aspectos: *Bugs*, *vulnerabilities*, *code smells*, *coverage* y *duplications*. Se aporta el link del proyecto para poder revisarlo.

4.14. Selenium

Selenium[35] es una herramienta que permite realizar pruebas de código sobre proyectos web. En nuestro caso se han llevado a cabo con Selenium IDE que ofrece una interfaz que permite grabar nuestros movimientos dentro de la web y provee una serie de comandos para hacer *asserts* de una forma mas intuitiva que con Selenium WebDriver. De forma adicional permite exportar los test creados.

Aspectos relevantes del desarrollo del proyecto

Este capítulo trata de recoger los aspectos más interesantes del desarrollo del proyecto. Las diferentes decisiones que se tomaron, pasando por los múltiples problemas que se sucedieron, propios de un proyecto creado desde cero y cómo se fueron solventando.

5.1. Ciclo de vida del desarrollo

El proyecto ha ido desarrollándose y se ha distribuido de forma semanal, basándonos en la metodología ágil Scrum. De forma que cada semana había una reunión con el tutor, en las que se veía el estado del proyecto y se establecían las nuevas tareas.

Lo primero fue instalar el entorno de desarrollo, una vez configurado y con el sistema embebido en mi poder, me puse manos a la obra y fui viendo cómo funcionaba la placa, el módulo gps y el IDE. Una vez hechas las conexiones del módulo gps, para que pudieran pasar los datos a la placa, se pudo ver por primera vez gracias a un pequeño código, cómo se mostraban los mensajes NMEA por pantalla, en la terminal de Termite.

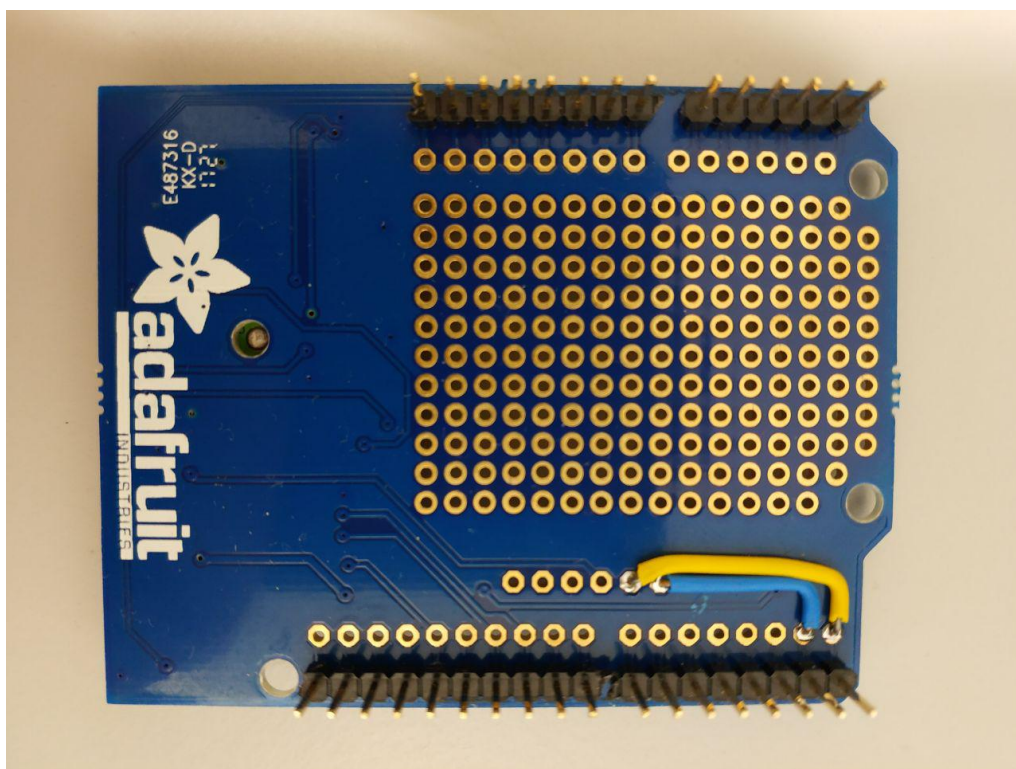


Figura 5.3: Conexiones hechas entre los pines Rx y Tx hacia Digital I/O 1 y 0 respectivamente.

El siguiente paso fue ver cómo funcionaba el acelerómetro que incorporaba la placa, para ello se decidió hacer un programa que almacenase los datos recogidos por él mismo en la tarjeta microSD.

Para ello, se necesitaban crear y configurar diferentes módulos para que pudiesen funcionar, nos hemos basado en unos tutoriales que, a pesar de estar en algunos casos desactualizados, nos han servido de base [12]. Algunos de estos módulos fueron:

- El sistema de almacenamiento.
- El componente del acelerómetro.
- El componente de la unidad de tiempo.
- El componente de la fecha.
- El componente de secciones críticas.
- El componente de la tarjeta microSD.
- El componente de la comunicación síncrona.

Cabe destacar que la mayoría de estas interfaces necesitan una pequeña configuración en sus parámetros para que sean funcionales y se terminen de acoplar con el resto del sistema, no entraremos en detalle ya que considero que se escapa del objetivo de esta memoria, sin embargo queda citado por si fuera de interés[12]. Finalmente, tras esto y una pequeña función, teníamos los datos del acelerómetro en la tarjeta microSD.

Era el momento hacer lo mismo con los mensajes NMEA proporcionados por el módulo GPS, pero antes había que instalar dos módulos más[13], el de una interrupción externa y el de una señal asíncrona que sería la encargada de pasar los datos del módulo GPS a la placa. De esta forma ya podíamos almacenar las coordenadas en un archivo de texto.



Figura 5.4: Placa de desarrollo FRDMK64F con el módulo GPS integrado y registrando datos.

A continuación vendría la creación de la interfaz para poder mostrar las coordenadas, previamente almacenadas. Se hizo una primera versión de una interfaz que simplemente mostrara un mapa donde poder representar coordenadas, para ello nos ayudamos de una API como es Mapbox. En este

punto del proyecto se consiguió representar por primera vez una ruta. Sin embargo debía automatizar el proceso ya que la conversión de los datos arrojados por la placa (NMEA) y los leídos por la API de Mapbox había sido realizada de forma manual, mediante una herramienta de internet.

El primer paso consistía en transformar los datos de NMEA a GeoJSON de forma automática, para ello, fue imprescindible la ayuda de la API de MyGeodata, realizando un script en Python que se encargaba de ello. A partir de ahí vino la implementación de la subida de archivos al servidor, para así poder pasárselo al convertidor. Una vez que el archivo estaba almacenado en la carpeta *subidas*, ya se podía convertir, descomprimir los archivos generados y representarlos en el mapa.

Mas tarde se añadieron algunas funcionalidades al mapa como los controles de navegación, edificios tridimensionales, la opción de descargar una imagen del mapa, centrarlo basándose en tu localización o la de regresar a la página de inicio. Además se añadieron las excepciones al código.

Por último se contó con herramientas como Selenium o Sonarcloud para hacer las pruebas del sistema y comprobar la calidad del código respectivamente.

El resultado del proyecto se compone de todas estas fases de desarrollo por las que ha ido pasando. A continuación abordaremos de una forma más detallada todos los problemas técnicos que surgieron, cómo los tratamos y sobre todo cómo fueron resueltos, con mayor o menor acierto.

5.2. Resoluciones técnicas

Arreglar Kinetis Studio

Tras apenas una semana de trabajo sobre el proyecto, Kinetis Design Studio dejó de funcionar, para ser más exactos, la carga de Processor Expert al ser integrado en un nuevo proyecto bloqueaba el programa quedando cargando en el 4 %.

Lo primero que se hizo fue probar con otros proyectos, pero no importaba si era nuevo o ya estaba creado, si contenía Processor Expert no cargaba correctamente. Se comprobó el log de errores que arrojaba y tras mucho buscar por internet no se encontró una solución válida, se optó por borrar registros, parámetros, re-instalaciones parciales del Processor Expert e incluso reinstalar el programa entero en varias ocasiones.

Tras ver que nada de esto funcionaba se optó por limpiar el registro de Windows, lo cual tampoco lo solventó. Por último solo se pudo pensar en una restauración del sistema operativo, que finalmente sí tuvo éxito.

Almacenar los valores del acelerómetro en la tarjeta microSD

Como un primer acercamiento a los componentes de la placa tras haber realizado la Práctica 1 de Sistemas Empotrados en el que se creaba un *Hola mundo*, se opta por capturar los datos aportados por los ejes del acelerómetro y almacenarlos en la tarjeta microSD. Se ha llevado a cabo con unas líneas de código que luego podremos tomar como referencia para enviar las coordenadas del GPS por puerto serie.

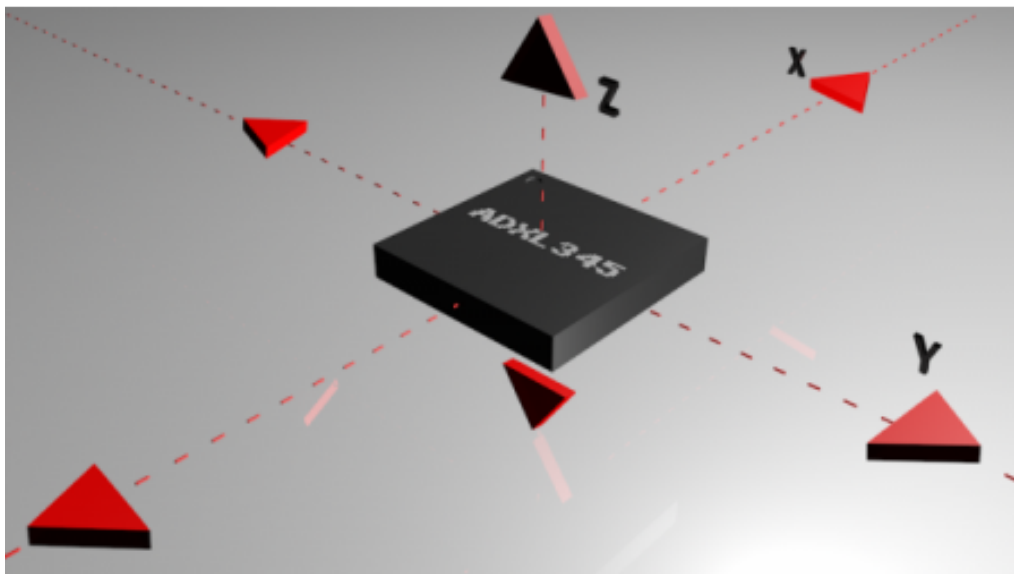


Figura 5.5: Acelerómetro de 6 ejes.[1]

Añadir componentes básicos

Una vez creado el proyecto y con la configuración básica debemos añadir algunos componentes. En este caso, para poder almacenar la información en la tarjeta utilizaremos el componente *Fat_FileSystem* que permite almacenar los datos gracias a la función `FAT1_Write`. Pero este componente requiere otro que aporte la fecha, para ello elegimos *GenericTimeDate* y por último, uno que permita proteger las secciones críticas (*CriticalSection*)), necesario para evitar que una interrupción pueda hacer que la fecha que se proporcione sea inconsistente. Las secciones críticas evitan que una interrupción o tarea de más prioridad haga un cambio de contexto dentro de la misma.

Además se requiere implementar una comunicación serie síncrona de tipo maestro esclavo, de la cual se encarga el componente *SPIMaster_LDD*, donde solo dos dispositivos se pueden comunicar entre si al mismo tiempo.

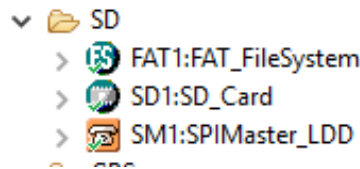


Figura 5.6: Componentes necesarios para permitir el almacenamiento en la tarjeta microSD.

Hay que relacionar y configurar todos los componentes (como por ejemplo *SPIMaster_LDD* estableciendo cuales serán los pines de entrada y salida de datos de la placa con el pc).

Para poder trabajar con el acelerómetro es necesario añadir también *FXOS8700CQ* que contiene varias funciones entre las que destacan, `FX1_GetX()`, `FX1_GetY()` y `FX1_GetZ()`. Se encargan de obtener los valores de cada eje en ese preciso momento. Automáticamente se añadirá el componente *GenericI2C* que implementa un *driver* genérico para el bus de comunicaciones en serie I2C (InterIntegrated Circuit), de esta forma funciona tanto para controladores de dispositivos lógicos *LDD* (Logical Device Driver) como para los clásicos que no lo son. En nuestro caso particular nos interesa activar la comunicación LDD I2C y esto obliga a añadir un componente más *I2C_LDD* que encapsula la interfaz de la comunicación I2C.

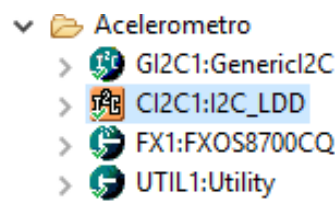


Figura 5.7: Componentes para poder obtener los datos del acelerómetro.

Por último tuve que comprobar que detectaba la tarjeta microSD y que se encontraba montado y funcionando el archivo del sistema (*fileSystemObject*).

Mostrar las sentencias NMEA por pantalla

Para recoger la información del módulo GPS debemos saber que una vez que el módulo conecta con los satélites, produce un *PPS* (*Pulso por Segundo*) cada 100 milisegundos. Por lo tanto, lo primero que debemos hacer es añadir los módulos necesarios y configurar los parámetros según el tutorial que me sirvió de guía [13]. Primero el módulo *ExtInt*, que no es más que una señal externa que se sincroniza con los pulsos del GPS, de forma que así podemos contar los pulsos o simplemente sincronizarlos con otras señales propias.

Segundo, necesitamos *AsynchroSerial* una señal asíncrona que implementa el protocolo serie UART y que interactuará con las líneas Rx y Tx del GPS. Como el receptor está constantemente enviando la información con 9600 baudios, debemos seleccionar la misma tasa.

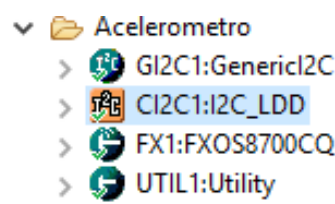


Figura 5.8: Componentes necesarios del módulo GPS.

Ahora con los módulos añadidos y configurados quedaba añadir el código que permitiera la visualización de las sentencias NMEA a través de la terminal Termite. Con esto era suficiente, pero como veremos en el siguiente punto, al añadir *FreeRTOS* se incorpora una cadena de caracteres donde almacenar las sentencia NMEA y un manejador de la misma (*xQueueHandle*).

```

static void Imprime (void) {
    char ch;
    int i;

    StorageOn();
    for(;;) {
        LEDR_Neg(); LEDG_Off();//Led rojo
        if(FRTOS1_xQueueReceive(caracteres, &ch ,10000) == pdTRUE){
            if (ch != '\n'){
                cadena[i++] = ch;
                cadena[i]=0;
            }else{
                cadena[i++] = '\n';
                if ((i>4 && cadena[3]=='R')){
                    /* Se ha recibido un dato. Se escribe por el puerto serie */
                    for(int j = 0; j < i; j++)
                        while(AS1_SendChar(cadena[j]) != ERR_OK) {}
                    EscribeSD(cadena);
                }
                i=0;
            }
        }
    }
}

```

Figura 5.9: Función Imprime encargada de pasar los datos al pc por puerto serie.

Almacenar las sentencias NMEA en la tarjeta microSD

Ahora que ya tenemos la configuración hecha (de las coordenadas del acelerómetro), que permite el almacenamiento de datos en la tarjeta SD, parece obvio que tan solo fuera necesario reutilizar la cadena ya rellena con nuestra información, pero aquí vino una de las mayores dificultades, se perdía información muy probablemente porque antes de que acabara de escribir en la cadena se empezaba a escribir en la tarjeta microSD y viceversa, de forma que se optó por incluir un sistema operativo en tiempo real como *FreeRTOS*. Para ello tuve que documentarme, añadir dos módulos necesarios al proyecto para que funcionara, *FreeRTOS* y *KinetisSDK*, y crear las tareas pertinentes con su nivel de prioridad asociado. De esta forma, ahora la información que se muestra en la pantalla se pasa directamente a la función *EscribeSD()* encargada de escribir una cadena dada, en la tarjeta microSD.

Como medida de eficiencia se optó por únicamente almacenar los mensajes NMEA de tipo GPRMC, que aportan toda la información que nosotros

necesitamos reduciendo en un 75 % la cantidad de datos almacenados.

Los datos se introducen en la cadena con el método *xQueueSendToBack* (dado que es una cola FIFO) y se extraen con *xQueueReceive* por el final de la cola.

```
static void CharGPS(void) {  
    byte err;  
    char ch;  
    GPS_ClearRxBuf(); //limpiamos el buffer del gps  
    for(;;) {  
        LEDR_Off(); LEDG_Neg(); //led verde  
        do {err = GPS_RecvChar(&ch);  
            } while((err != ERR_OK));  
        FRTOS1_xQueueSendToBack(caracteres, &ch , (portTickType) 0xFFFFFFFF);  
    }  
}
```

Figura 5.10: Tarea en tiempo real dedicada a recibir e introducir los caracteres en la cadena.

De manera adicional se decidió dar utilidad al diodo led rgb que incorpora la placa y se configuró de forma que emitiera luz verde (*LEDR_Off()* y *LEDG_Neg()*) cuando estuviera recolectando datos del GPS y luz roja (*LEDR_Neg()* y *LEDG_Off()*) cuando estuviera ocupado escribiendo en la tarjeta microSD.

```

void EscribeSD(char *cadena){
    UINT bandwidth;

    /* Abrir fichero */
    if (FAT1_open(&file, "./log_gps.txt", FA_OPEN_ALWAYS|FA_WRITE)!=FR_OK) {
        Err();
    }
    /* Nos posicionamos en el final del archivo */
    if (FAT1_lseek(&file, f_size(&file)) != FR_OK || file.fptr != f_size(&file)) {
        Err();
    }

    /* Se escribe la cadena en la microSD */
    if (FAT1_write(&file, cadena, UTIL1_strlen(cadena), &bandwidth)!=FR_OK) {
        (void)FAT1_close(&file);
        Err();
    }

    /* Cerrar el fichero */
    (void)FAT1_close(&file);
}

```

Figura 5.11: Código necesario para el correcto envío de datos a la tarjeta microSD.

Añadir un sistema en tiempo real

En un principio cabe destacar que no incluimos un sistema operativo de tiempo real, lo cual generaba diferentes problemas, el principal es que era necesario programar la gestión y planificación de las tareas, utilizando hilos, interrupciones periódicas etc. Esto cuando el problema es sencillo se puede hacer, pero cuando el problema empieza a complicarse es mejor delegarlo en un RTOS.

En consecuencia, la información almacenada no correspondía en muchas ocasiones con la mostrada por pantalla. Más adelante descubrimos que esto se debía a que muy probablemente estuviéramos almacenando información en nuestra cadena antes si quiera de llegar a escribir por completo la que ya teníamos en el buffer. Utilizamos FreeRTOS por ser uno de los más extendidos para pequeños sistemas embebidos. Añadimos el módulo correspondiente en Kinetis *FreeRTOS* y lo configuramos.

```

if (xTaskCreate(
    CharGPS, /* Introduce en la cadena de caracteres lo que recibe del módulo GPS */
    "gps", /* nombre de la tarea para el kernel */
    configMINIMAL_STACK_SIZE, /* tamaño pila asociada a la tarea */
    (void*)NULL, /* puntero a los parámetros iniciales de la tarea */
    5, /* prioridad de la tarea, cuanto más bajo es el número menor es la prioridad */
    NULL /* manejo de la tarea, NULL si ni se va a crear o destruir */
) != pdPASS) { /* devuelve pdPASS si se ha creado la tarea */
    for(;;) {} /* error! Probablemente sin memoria */
}

if (xTaskCreate(
    Imprime, /* Se encarga de sacar por pantalla los mensajes NMEA */
    "print", /* nombre de la tarea para el kernel */
    configMINIMAL_STACK_SIZE, /* tamaño pila asociada a la tarea */
    (void*)NULL, /* puntero a los parámetros iniciales de la tarea */
    6, /* prioridad de la tarea, cuanto más bajo es el número menor es la prioridad */
    NULL /* manejo de la tarea, NULL si ni se va a crear o destruir */
) != pdPASS) { /* devuelve pdPASS si se ha creado la tarea */
    for(;;) {} /* error! Probablemente sin memoria */
}

if (xTaskCreate(
    Acce, /* Recoge periodicamente los valores de las coordenadas del acelerómetro */
    "Acc", /* nombre de la tarea para el kernel */
    configMINIMAL_STACK_SIZE, /* tamaño pila asociada a la tarea */
    (void*)NULL, /* puntero a los parámetros iniciales de la tarea */
    1, /* prioridad de la tarea, cuanto más bajo es el número menor es la prioridad */
    NULL /* manejo de la tarea, NULL si ni se va a crear o destruir */
) != pdPASS) { /* devuelve pdPASS si se ha creado la tarea */
    for(;;) {} /* error! Probablemente sin memoria */
}

```

Figura 5.12: Inicialización de tareas de tiempo real.

Elección de la API para representar las rutas

Puede parecer un tema trivial, pero se le dio bastante importancia. En un primer momento siempre se consideró la opción de utilizar la de Google Maps, de hecho se llegó a realizar el código para que funcionara (y así lo hizo), pero desde junio de 2018 se convirtió en una API de pago, si bien te otorgan en tu cuenta de Google Maps, 300\$ al mes de crédito para gastar en su API en función del tráfico que tengan tus mapas creados, había que introducir una forma de pago por si excedías el límite, por lo que no me pareció la mejor de las opciones.

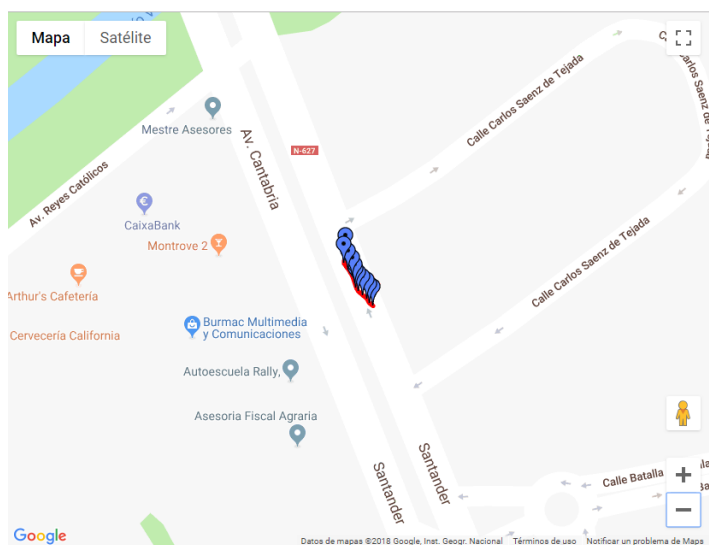


Figura 5.13: Mapa creado con la API de Google.

De forma que busqué una alternativa gratuita, es ahí donde descubrí Mapbox, una API que poco tiene que envidiar de la de Google. Con un funcionamiento similar multitud de opciones y gratuita. Ya cuenta con numerosas empresas que usan sus servicios como: Tinder, Bosch, National Geographic, Mastercard, CNN o Snapchat entre las más importantes.



Figura 5.14: Empresas que ya trabajan con Mapbox.

Creación de una interfaz básica HTML

Con la ayuda de la documentación de Mapbox^[21] se creó una primera página HTML sencilla que mostraba unas coordenadas pasadas directamente en un array. Pensé que tendría que leer las que yo había capturado de forma aislada y pasárselas. Por fortuna descubrí que se podía pasar el fichero entero en formato GeoJSON. Convertí mi archivo NMEA a GeoJSON con una herramienta web y pasándoselo o bien de manera local o un enlace raw de

GitHub funcionaba perfectamente. Pero el objetivo era automatizar este proceso y que la mayoría de acciones fueran transparentes al usuario.

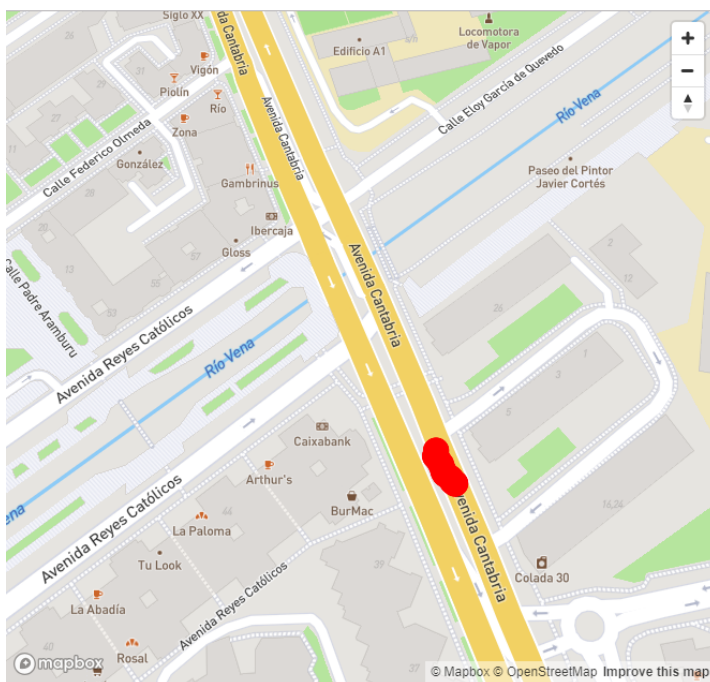


Figura 5.15: Mapa creado con la API de Mapbox.

Conversión automática de NMEA a GeoJSON

Tras mucho investigar descubrí que la conversión de NMEA a GeoJSON es una de las menos frecuentes, son más típicas otras como GPX (para el caso del receptor Garmin) o KML en el caso de querer representar en Google Maps. Estuve barajando hacer un parser manualmente, pero tendría que dedicar muchos recursos en aprender las especificaciones del estándar GeoJSON, así que por temas de eficiencia opté por buscar un recurso que lo agilizará. Descubrí MyGeodata, que es sin ninguna duda el mejor sitio web para convertir entre cientos de estándares. Solo había un problema y era que de forma gratuita solo ofrece 3 conversiones mensuales, pero tras hablar con el soporte y explicar mi situación de estudiante, me ofrecieron una *key* gratuita.

Con la ayuda de su documentación,[28] desarrollé el código, pero entre los parámetros que había que poner para poder hacer uso de su herramienta se encontraba uno que me estaba dando problemas. Se trata de *outcrs*,

(sistema de referencias de coordenadas de salida) que no sabía qué había que indicar, más allá del formato de salida que se deseaba. Por defecto se realiza mediante *'EPSG:3857'*, necesario para conversiones para la API de Google en KML, en nuestro caso Mapbox necesitaba GeoJSON de forma que nos correspondía *'EPSG:4326'*.

```
data = {'format': 'geojson', 'outcrs': 'EPSG:4326', 'outform': 'binary', 'key': '5PsoGucmRRD2h0c'}
```

Figura 5.16: Información básica que necesita para poder convertir el archivo.

Habilitar la subida de archivos al servidor

La opción más cómoda y amigable que se me ocurrió para el usuario, con la que estuviera familiarizado, fue que realizara la subida del archivo tal cual lo encontraba en la microSD. La parte del *backend* se encargaba de convertirlo y cargarlo, con la opción que brinda en un futuro tener los archivos en el servidor para poder mostrar el mapa deseado por el usuario.

La subida se hace mediante un formulario que llama al archivo *upload_file.php* que es el encargado de casi todo. En un principio se representaba el mapa en la página principal, pero al no tener todavía las coordenadas y estar vacío se decidió mover a una página secundaria con el fin de optimizar la página de inicio.

```
<div id='upload'>
  <form action="upload_file.php" method="post" enctype="multipart/form-data">
    <input type="file" name="archivo" id="archivo"></input>
    <input type="submit" value="Subir archivo"></input>
  </form>
</div>
```

Figura 5.17: Formulario que habilita la subida de archivos.

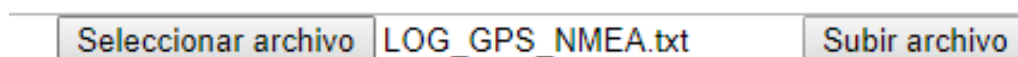


Figura 5.18: Interfaz de subida de archivos familiar para el usuario.

Configuración de XAMPP y Python

En el anterior punto para la correcta ejecución del código PHP, instalamos XAMPP en nuestro ordenador para trabajar de manera local y que luego fuera fácilmente exportable a un servidor si así se requiere en un futuro. De manera nativa XAMPP no permite la ejecución de Python, pero se puede conseguir añadiendo la extensión *.py* a la *Common Gateway Interface* (interfaz de entrada común) como se detallará en el manual del programador. De esta forma utilizo la función *shell_exec* para que ejecute el script encargado de la conversión de NMEA a GeoJSON.

```
/*
*Segundo, vamos a llamar a un script hecho en Python que se encarga de
*transformar los mensajes NMEA a GeoJSON.
*De no tener python configurado en el path, indicar donde esta instalado,
*ej: C:/Users/Alejandro/Anaconda3/python
*shell_exec('/path/to/python /path/to/your/script.py ' . $nombreArchivo);*/
shell_exec('python ./subidas/convert_NMEA-GeoJSON.py ./subidas/' . $_FILES['archivo']['name'] );
```

Figura 5.19: Llamada al script que convierte las coordenadas.

Descomprimir el archivo generado por MyGeodata

La conversión de coordenadas genera un archivo comprimido, por lo tanto, para que pueda tener acceso, debemos primero descomprimirlo. Esto se realiza mediante la creación de una variable de tipo *ZipArchive* de PHP, la cual tiene un método dedicado a la extracción del contenido (*extractTo()*) y lo almacena en otra carpeta.

```
$compressed = new ZipArchive;
$flag = $compressed->open('mygeodata.zip');
if ($flag === TRUE) {
    $compressed->extractTo('./mygeodata');
    $compressed->close();
    // Archivo descomprimido.
} else {
    // Error en la descompresión...
}
```

Figura 5.20: Descompresión del *zip* generado por MyGeodata.

Añadir fecha y hora a los documentos

Se planteaba un problema, al subir archivos con el mismo nombre se sobrescribían, para solventarlo se pensó en acompañarlos de la fecha y la hora de forma que cada uno tuviese un nombre único sin tener que forzar al usuarios a renombrarlos ni que se sobrescribieran perdiendo los datos anteriores. Utilizamos la función *date* de PHP y un formato del tipo *dd-mm-yyyy_h-m-s_* delante del nombre del fichero.


















Nombre	Fecha de modifica...	Tipo	Tamaño
 26-01-2019_14-18-41_LOG_GPS_NMEA.txt	26/01/2019 14:18	Documento de tex...	3 KB
 26-01-2019_14-18-58_LOG_GPS_NMEA.txt	26/01/2019 14:18	Documento de tex...	3 KB
 26-01-2019_14-25-49_LOG_GPS_NMEA.txt	26/01/2019 14:25	Documento de tex...	3 KB
 26-01-2019_14-26-42_LOG_GPS_NMEA.txt	26/01/2019 14:26	Documento de tex...	3 KB
 26-01-2019_14-27-01_LOG_GPS_NMEA.txt	26/01/2019 14:27	Documento de tex...	3 KB
 26-01-2019_14-28-01_LOG_GPS_NMEA.txt	26/01/2019 14:28	Documento de tex...	3 KB
 26-01-2019_14-28-25_LOG_GPS_NMEA.txt	26/01/2019 14:28	Documento de tex...	3 KB
 26-01-2019_14-28-41_LOG_GPS_NMEA.txt	26/01/2019 14:28	Documento de tex...	3 KB
 27-01-2019_20-00-44_LOG_GPS_NMEA.txt	27/01/2019 20:00	Documento de tex...	3 KB
 27-01-2019_20-01-19_LOG_GPS_NMEA.txt	27/01/2019 20:01	Documento de tex...	3 KB
 27-01-2019_20-01-31_LOG_GPS_NMEA.txt	27/01/2019 20:01	Documento de tex...	3 KB
 27-01-2019_20-02-22_LOG_GPS_NMEA.txt	27/01/2019 20:02	Documento de tex...	3 KB
 27-01-2019_20-03-08_LOG_GPS_NMEA.txt	27/01/2019 20:03	Documento de tex...	3 KB
 28-01-2019_10-51-20_LOG_GPS_NMEA.txt	28/01/2019 10:51	Documento de tex...	3 KB
 28-01-2019_10-51-32_LOG_GPS_NMEA.txt	28/01/2019 10:51	Documento de tex...	3 KB
 28-01-2019_10-53-52_LOG_GPS_NMEA.txt	28/01/2019 10:53	Documento de tex...	3 KB
 28-01-2019_10-55-34_LOG_GPS_NMEA.txt	28/01/2019 10:55	Documento de tex...	3 KB

Figura 5.21: Documentos con el día y la fecha en el título.

Exportar imagen del mapa mostrado y Geolocalización

Para que el usuario se pueda llevar una instantánea del mapa generado con la ruta, se ha incluido un botón que habilita la descarga de una imagen. Se ha conseguido habilitando el *buffer* de lo dibujado *preserveDrawingBuffer:true* y recuperándolo más tarde cuando demos al botón llamando a *getCanvas()* [34]

De igual forma se añadió otro botón para centrar el mapa basándose en tu posición actual. Para ello, se requiere el permiso por parte del navegador y en última estancia del usuario, para poder acceder a tu posición.

```
// Añadir botonera de zoom y rotación del mapa.
map.addControl(new mapboxgl.NavigationControl());

//Obtener la imagen del mapa mostrado en pantalla.
$('#enlaceDescarga').click(function() {
    var img = map.getCanvas().toDataURL('image/png')
    this.href = img
})

//Geolocalizar tu posición y centrar el mapa en ella.
document.getElementById('volar').addEventListener('click', function () {
    if ('geolocation' in navigator) {
        navigator.geolocation.getCurrentPosition(function(position) {
            'use strict';
            map.flyTo({center: [position.coords.longitude,
                position.coords.latitude], zoom: 14});
        });
    }
},
```

Figura 5.22: Código que habilita los botones de control, la descarga del mapa y el centrarlo en tu posición.

Añadir una plantilla HTML

Una vez que la aplicación era funcional, era hora de dotarla de una interfaz agradable al usuario, se utilizó una plantilla basada en *Bootstrap* [38] para la página de inicio, que tras una serie de cambios como: imagen de fondo, textos, imágenes y crear los botones de subida de los archivos, quedó finalizada. Cabe destacar que botones como: “Iniciar sesión”, “About”, “Contact”, “Terms of Use”, “Privacy Policy”, los de las redes sociales o el formulario del *email* no se han implementado, pero se han dejado por estética y como línea de mejora de futuro. La página a la que da acceso tras la subida de los ficheros carece de plantilla pues se muestra el mapa a lo largo de la pantalla y contiene 3 botones.

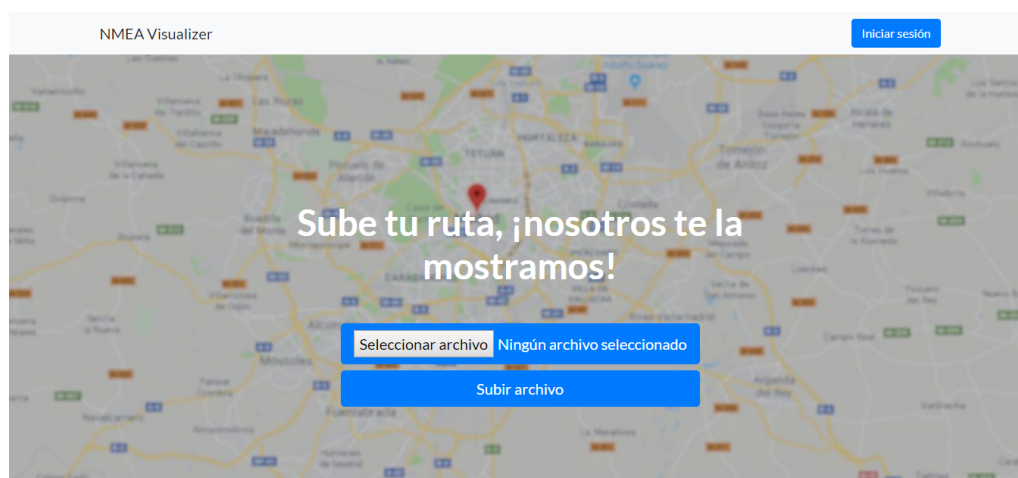


Figura 5.23: Página de inicio con los botones de subida del archivo.

Despliegue de la aplicación

Para que se pudiera probar la herramienta se necesitaba una placa de desarrollo correctamente programada, o una ruta previamente registrada y una página web. Pensamos que *hostear* la página sería una buena opción, sin embargo finalmente no fue la medida adoptada por diversos motivos. En dominios como *x10hosting*, *5gbfree*, *000webhost* que presentaban una buena oferta gratuita de navegación y número de páginas máximas alojadas, tenían la limitación de no ofrecer Python de forma gratuita, con lo cual no se podía ejecutar la conversión. También se probó en “GitHub Pages” arrojando el mensaje “405 not allowed” al tratar de ejecutar el código en Python y finalmente tampoco fue posible en *OpenShift* ni *Heroku*.

Optamos por realizar nuestro despliegue sobre una máquina virtual de Windows 7 limpia que se incluirá en la entrega. Se instaló XAMPP y Python y se configuró correctamente para el correcto funcionamiento.

Activación/desactivación por acelerómetro

Más adelante retomamos una actividad pendiente, que el registro de los datos en la tarjeta microSD dependiera del movimiento con el fin de no almacenar coordenadas de forma repetitiva, por ejemplo en un semáforo. Lo primero fue calibrar el acelerómetro, suponiendo una fuerza de 0G en los ejes 'x' e 'y' y 1G en el eje 'z' proveniente de la gravedad de la tierra, mediante los métodos *FX1_CalibrateX1g()*, *FX1_CalibrateY1g()* y *FX1_CalibrateZ1g()* respectivamente. Más tarde, pusimos como condición que si no se sobrepasaba

una fuerza, que establecimos como la mínima para evitar errores producidos por microvibraciones, no se escribiera en la tarjeta microSD.

```
static void Acce(void) {
    FX1_Enable(); //Activa el acelerometro
    /* Calibra los diferentes ejes, suponiendo 0G en 'x' e 'y' y la fuerza
     * normal de la gravedad 1G en el 'z' */
    FX1_CalibrateX1g();
    FX1_CalibrateY1g();
    FX1_CalibrateZ1g();
    for(;;) {
        x = FX1_GetX();
        y = FX1_GetY();
        z = FX1_GetZ();
        vTaskDelay(500/portTICK_RATE_MS);
    }
}
```

Figura 5.24: Código encargado que calibrar el acelerómetro y capturar los valores de los ejes.

Excepciones PHP

Por último decidimos crear excepciones para tratar de capturar cualquier error que pudiera surgir en la interfaz. Comprobamos que solo se pudieran subir archivos con extensión txt, que son los que genera la placa, que se moviera a la carpeta “subidas” correctamente, que contuviera mensajes NMEA en su interior y que el archivo se convirtiese a GeoJSON sin errores.

```
function estxt() {
    if ($_FILES['archivo']['type'] != 'text/plain'){ //diferente a un txt
        throw new Exception();
    }
    return true;
}

if ($_FILES['archivo']['error'] > 0){
    echo "Error: " . $_FILES['archivo']['error'] . "<br>";
}
else{
    try{
        estxt();
        //echo 'Si ves esto el archivo es un txt';
    }catch(Exception $e){
        echo<script type="text/javascript">
            alert("Error en la subida del fichero. Por favor suba un fichero .txt");
            window.location.href="index.html";
        </script>;
    }
}
```

Figura 5.25: Excepción en PHP que comprueba que el archivo subido sea de texto.

Problemas encontrados en el desarrollo

En este apartado explicaremos algunas cuestiones que nos han provocado retrasos y que detallaremos a continuación por si pudiera servir de ayuda en un futuro.

La placa deja de funcionar

En algunas ocasiones es probable que la placa deje de funcionar en el modo de *debugueo* OpenSDA y solo se encienda la luz naranja de corriente. Para solventar esto es necesario enchufar la placa al ordenador mientras se aprieta el botón de *reset* de forma que podamos acceder al *bootloader* y allí pegar el archivo OpenSDA_V2.bin que se proporciona con el proyecto.

Comunicación del módulo GPS con la placa

El módulo GPS tiene un interruptor, para que el envío de datos se produzca de forma satisfactoria debe estar posicionado en “Soft Serial” y no en la posición “Direct”.

Kinetis no ejecuta los cambios que hemos hecho

En ciertos momentos nos hemos encontrado con que Kinetis no acometía los cambios que se le había hecho al proyecto, para este problema no hemos

encontrado una solución al respecto y la única ha sido reiniciar el programa.

Cómo copiar y pegar proyectos en Kinetis

Para poder hacer una copia de tu proyecto y así poder hacer modificaciones sin alterar el original debes hacer antes unos pasos previos:

- Al crear el proyecto evita la opción de que haya archivos enlazados y créalo *standalone* para que los proyectos no compartan archivos y sean independientes.
- Dentro de Project>Properties>C/C++ Build>Settings>Build Artifact>Artifact name asegúrate de poner \$ProjectName de forma que no referencie al nombre del primer proyecto cuando vaya a ser copiado.
- Evita rutas absolutas dentro de Project>Properties>C/C++ Build>Settings>Tool Settings>Include paths y usa relativas.
- Es una buena idea deshacernos de todo el residuo que pueda haber dejado el proyecto padre, para ello Project>Clean

Guía extraída de: <https://mcuoneclipse.com/2017/02/18/tips-for-making...>

Aumentar velocidad de compilación

Es común que tengas que compilar cada poco tiempo, con el objetivo de reducir el tiempo de espera, es buena idea activar la construcción de código de forma paralela, para ello ve a Project>Properties>C/C++ Build>Behavior>Enable Parallel Build y selecciona el máximo que permita tu ordenador.

Trabajos relacionados

Vamos a hacer un pequeño resumen de las mejores herramientas ya existentes que realizan algo muy similar al proyecto aquí expuesto.

6.1. Cronología en Google Maps

Sin duda para mí, la cronología de Google [39] es la reina indiscutible en este apartado. Se trata de un recolector profesional de ubicaciones, que incluso pausándolo, recolecta tu posición si por ejemplo tienes activado la *Actividad en la Web y en Aplicaciones* (algo que viene por defecto) para personalizar los resultados de tu búsqueda, en función de tu posición o si haces una foto, para añadir la posición donde fue tomada. Además añade en tu ruta establecimientos visitados, tiempo de visita y no tiene miedo a pronosticar dónde vives o dónde está tu trabajo, gracias a sus algoritmos.

Cuenta con una interfaz muy simple, accesible tanto desde el ordenador como a través de la aplicación móvil de Google Maps, que te permite ver tu recorrido a lo largo de un día cualquiera, gracias a la antena GPS que incorpora tu dispositivo. Con unos conocimientos avanzados sí se podría desactivar por completo, se puede editar alguna entrada específica o incluso borrarla y por supuesto, es privada. Descompresión del *zip* generado por MyGeodata.

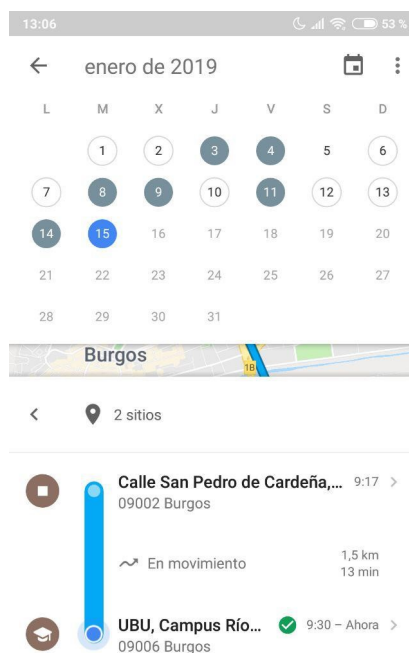


Figura 6.26: Se muestran todos los trayectos que se realicen en el día.

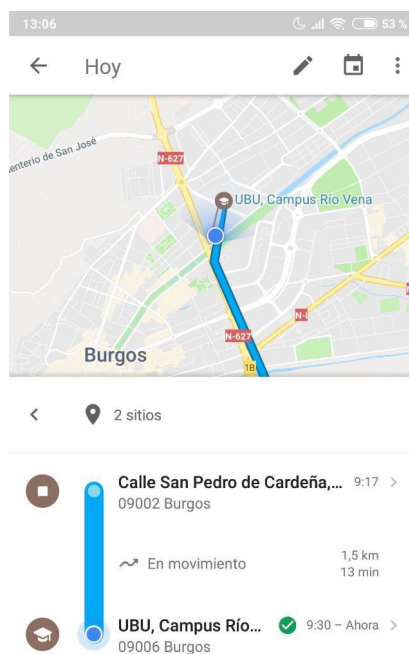


Figura 6.27: Se puede seleccionar cualquier día, desde que tenga datos.

6.2. Endomondo

Endomondo [10] es una aplicación móvil diseñada específicamente para deportistas. Si bien, aparte de recoger la ruta aporta otras muchas opciones y características, realiza la captura de datos de forma correcta gracias al GPS integrado del móvil, pero a diferencia de la anterior, en este caso, tú marcas cuando empieza a registrar tu ruta. Al final de la misma te dice kilómetros recorridos, tiempo invertido, velocidad media, etc.

Cuenta con la posibilidad de crear grupos entre amigos para poder retarles en una misma ruta y así poder comparar tiempos de forma directa a través de la aplicación. Destaca su amplia compatibilidad con otros dispositivos y cuentas (Garmin Connect, Polar Flow, TomTom MySports o Fitbit) para sumar funcionalidades a la aplicación.

6.3. Strava

Otra aplicación diseñada directamente para el público más deportista, Strava [31] es considerada la red social de los *runners*, altamente compatible con los pulsómetros más avanzados del mercado, permite una alta interacción entre los usuarios pudiendo ver las rutas de otros y sus estadísticas.

6.4. Relive

La última a comparar Relive [33], a pesar de recoger la ruta como los dos anteriores, de forma tradicional y manual, cuenta con algo diferencial, y es la forma en la que es capaz de representar los mapas. Se muestran en forma de vídeo 3D, recreando la ruta por donde has ido pasando e incluyendo fotos y vídeos que hayas hecho por el camino si así lo deseas.



Figura 6.28: Ruta completa tradicional realizada en 2D.



Figura 6.29: Vídeo en 3D recorriendo todos los puntos por los que se ha pasado.

6.5. Conclusión

Como comentario adicional a todas estas aplicaciones móviles cabe destacar que en mi caso no dispongo de un smartphone, se trata de un registrador de bajo costo, no asociado a ningún terminal móvil y que además almacena las posiciones en una tarjeta microSD. Las aplicaciones finales de este dispositivo podrían ser otras muy diferentes como poder acoplarlo en un coche o en una bici y tener un registro de rutas.

Conclusiones y Líneas de trabajo futuras

7.1. Conclusiones

Una vez concluido el trabajo de fin de grado, podría decir que he aprendido más de lo esperado en un primer momento, sobre todo por el tipo de trabajo escogido, tan centrado en los sistemas de posicionamiento y en los sistemas embebidos.

El resultado (al margen de la captura de datos que se da por finalizada), es una interfaz básica, sin pretensiones de ser una herramienta completamente finalizada y con muchas líneas de trabajo futuras, pero con una buena base ya realizada, que agilice su crecimiento y multiplique sus opciones en poco tiempo.

He aprendido mucho sobre los sistemas empotrados, su versatilidad y su gran utilidad a bajo costo. También he aprendido lenguajes nuevos para mí como PHP, herramientas como XAMPP, Termite, Kinetis, Mapbox, MyGeodata o LaTeX y la importancia de las API. He reforzado mis conocimientos en C, HTML y sistemas en tiempo real y he profundizado sobre el hecho de realizar una página de forma local para luego poder subirla a un hosting.

He alcanzado un nivel de entendimiento sobre posicionamiento GPS, que me permite comprender de una forma más clara cómo funcionan todos los GPS que nos acompañan en el día a día y lo importantes que llegan a ser, en actividades tan diferentes como operaciones de rescate o bancarias.

La organización en un proyecto de esta envergadura es primordial, de forma que planificarme y utilizar una herramienta de gestión como es GitHub, ha formado parte de mi día a día estos últimos meses y, a pesar de que

requiere tiempo, finalmente quedan muy bien expuestas todas las fases por las que se ha ido pasando, más si se revisan las diferentes *issues* y *milestones*.

7.2. Líneas de trabajo futuras

- Permitir la subida múltiple de ficheros.
- Crear un login de acceso.
- Crear una base de datos con todos los archivos subidos.
- Poder acceder a los archivos subidos y mostrar el que se desee.
- Añadir otros idiomas a la página.
- Implementar los botones de la interfaz que carecen de funcionalidad.
- Incluir gráficos y estadísticas.
- Cambiar la API de pago por otra gratuita o un hacer un parser que convierta los ficheros.
- Dotar a la placa de conexión a Internet para poder mostrar la ruta en tiempo real.

Bibliografía

- [1] 5hertz. Abc del acelerometro. https://www.5hertz.com/index.php?route=tutoriales/tutorial&tutorial_id=2, 2019.
- [2] Mygeodata Cloud. Mygeodata cloud... a place where your gis data lives! <https://mygeodata.cloud/>, 2018.
- [3] CompuPhase. Termit: a simple rs232 terminal. https://www.compuphase.com/software_termite.htm, 2018.
- [4] Mantenimiento De Una Computadora. ¿qué es xampp y para que sirve? <https://mantenimientosdeunapc.blogspot.com/2011/11/que-es-xampp-y-para-que-sirve.html>, 2011.
- [5] DePriest Dale. Nmea data. <https://www.gpsinformation.org/dale/nmea.htm>, 2018.
- [6] Departamento de ingeniería eléctrica electrónica y de control de la UNED. Ingeniería de los sistemas embebidos. http://www.ieec.uned.es/investigacion/Dipseil/PAC/archivos/Informacion_de_referencia_ISE5_3_1.pdf, 2011.
- [7] Gobierno de los Estados Unidos. Sistema de posicionamiento global. al servicio del mundo. <https://www.gps.gov/spanish.php>, 2018.
- [8] Alonso Diego. Qué son los códigos epsg / srid y su vinculación con postgis. <https://mappinggis.com/2016/04/los-codigos-epsg-srid-vinculacion-postgis/>, 2016.
- [9] EcuRed. Sistema operativo de tiempo real. https://www.ecured.cu/Sistema_Operativo_de_Tiempo_Real, 2014.

- [10] Endomondo. Endomondo - correr & ciclismo. <https://play.google.com/store/apps/details?id=com.endomondo.android&hl=es>, 2018.
- [11] Gakstatter Eric. What exactly is gps nmea data? <https://www.gpsworld.com/what-exactly-is-gps-nmea-data/>, 2015.
- [12] Styger Erich. Tutorial: Data logger with the frdm-k64f board. <https://mcuoneclipse.com/2014/05/26/tutorial-data-logger-with-the-frdm-k64f-board/>, 2014.
- [13] Styger Erich. Tutorial: Freedom board with adafruit ultimate gps data logger shield. <https://mcuoneclipse.com/2014/05/31/tutorial-freedom-board-with-adafruit-ultimate-gps-data-logger-shield/>, 2014.
- [14] FreeRTOS. The freertos™ kernel market leading, de-facto standard and cross platform rtos kernel. <https://www.freertos.org/>, 2018.
- [15] Freescale. Kinetis design studio v3.0.0- user's guide. <https://www.nxp.com/docs/en/user-guide/KDSUG.pdf>, 2015.
- [16] Freescale. Processor expert® software and embedded components. https://www.nxp.com/support/developer-resources/software-development-tools/processor-expert-and-embedded-components:BEAN_STORE_MAIN, 2018.
- [17] GitHub. The world's leading software development platform. <https://github.com/>, 2018.
- [18] Serra Jose Maria. ¿qué es el standar nmea. <http://www.informaticaabordo.com/2011/10/%C2%BFque-es-el-standar-nmea/>, 2011.
- [19] LaTeX. Agile project management for github. <https://www.latex-project.org/>, 2018.
- [20] Alegesa Leandro. Definición de microsd (tarjeta de memoria). <http://www.alegsa.com.ar/Dic/microsd.php>, 2017.
- [21] Mapbox. Initialize a map in an html element with mapbox gl js. <https://docs.mapbox.com/mapbox-gl-js/example/simple-map/>, 2019.

- [22] Alejandro Merino. *Interfaces y modos de operación*, chapter 2, pages 9–12. Universidad de Burgos, 2017. Máster Universitario en Ingeniería Informática, Sistemas Empotrados y Ubicuos.
- [23] Alejandro Merino. *Interfaces y modos de operación*, chapter 2, pages 12–13. Universidad de Burgos, 2017. Máster Universitario en Ingeniería Informática, Sistemas Empotrados y Ubicuos.
- [24] Alejandro Merino. *Interfaces y modos de operación*, chapter 2, pages 13–14. Universidad de Burgos, 2017. Máster Universitario en Ingeniería Informática, Sistemas Empotrados y Ubicuos.
- [25] Alvarez Miguel Angel. Qué es php. <https://desarrolloweb.com/articulos/392.php>, 2001.
- [26] Alvarez Miguel Angel. Qué es python. <https://desarrolloweb.com/articulos/1325.php>, 2003.
- [27] MiKTeX. About miktex. <https://miktex.org/about>, 2018.
- [28] MyGeoData. Mygeodata cloud api manual. <https://mygeodata.cloud/static/doc/MyGeodata-Cloud-API-Manual.pdf>, 2019.
- [29] Omega. Acelerómetro. ¿qué es un acelerómetro? <https://es.omega.com/prodinfo/acelerometro.html>, 2019.
- [30] Guardiola Pablo. Mapbox: el sdk de mapas abierto. <https://www.genbeta.com/desarrollo/mapbox-el-sdk-de-mapas-abierto>, 2017.
- [31] PulsómetroSinBanda. Las 6 mejores aplicaciones para correr en 2019. <https://www.pulsometrosinbanda.com/mejores-aplicaciones-para-correr-2018/>, 2019.
- [32] QGIS. Sistema de coordenadas de referencia. https://docs.qgis.org/2.14/es/docs/gentle_gis_introduction/coordinate_reference_systems.html, 2019.
- [33] Relive. ¿cómo funciona relive? <https://www.relive.cc/support?hl=es>, 2019.
- [34] Richard. Mapbox gl js: Export map to png or pdf? <https://stackoverflow.com/questions/42483449/mapbox-gl-js-export-map-to-png-or-pdf>, 2017.

- [35] Seleniumhq. What is selenium? <https://www.seleniumhq.org/>, 2019.
- [36] SemanticWebBuilder. Sistemas embebidos: Innovando hacia los sistemas inteligentes. http://www.semanticwebbuilder.org.mx/es_mx/swb/Sistemas_Embebidos_Innovando_hacia_los_Sistemas_Inteligentes_, 2016.
- [37] Sonarcloud. Clean code rockstar status. <https://sonarcloud.io/about>, 2019.
- [38] Startbootstrap. Landing page. a simple, elegant, and beautifully responsive landing page theme for bootstrap 4 websites. <https://startbootstrap.com/template-overviews/landing-page/>, 2019.
- [39] Google support. Cronología en google maps. <https://support.google.com/maps/answer/6258979?co=GENIE.Platform%3DDesktop&hl=es>, 2019.
- [40] Texmaker. Free cross-platform latex editor since 2003. <http://www.xmlmath.net/texmaker/index.html>, 2003.
- [41] Wikipedia. C (lenguaje de programación). [https://es.wikipedia.org/wiki/C_\(lenguaje_de_programaci%C3%B3n\)](https://es.wikipedia.org/wiki/C_(lenguaje_de_programaci%C3%B3n)), 2018.
- [42] Wikipedia. Licencia mit. https://es.wikipedia.org/wiki/Licencia_MIT, 2018.
- [43] Wikipedia. Sistema operativo de tiempo real. https://es.wikipedia.org/wiki/Sistema_operativo_de_tiempo_real, 2018.
- [44] ZenHub. Agile project management for github. <https://www.zenhub.com/>, 2018.