UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Generación de rutas utilizando sistemas empotrados Documentación Técnica

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

Índice general

Índice de figuras

Índice de tablas

Apéndice A

Plan de Proyecto Software

A.1. Introducción

En este primer apartado de los anexos se encuentra toda la información sobre cómo se ha ido desarrollando en el tiempo el proyecto. De igual forma, un análisis sobre la viabilidad tanto económica como legal.

A.2. Planificación temporal

El trabajo de fin de grado se ha ido desarrollando a base de *sprints* de 1 o 2 semanas de duración. Cada semana se hacía una revisión con el tutor de los objetivos que se habían marcado y dependiendo de si se habían alcanzado o no, se enfatizaba más sobre los mismos o se establecían nuevos objetivos. Algunas herramientas como ZenHub y GitHub han ayudado mucho en este proceso. En el GitHub del proyecto se pueden observar las diferentes fases por las que se ha ido pasando https://github.com/alejandrolampreave/GPS.

Sprint 1 - 03/10/2018-10/10/2018

La primera semana se dedica a repasar y profundizar sobre el funcionamiento de GitHub y a instalar e informarme sobre extensiones como ZenHub, para ello se ven diversos videotutoriales. Se realiza el primer *issue* de prueba asociado al primer *milestone*, me lo asigno a mi mismo y le pronostico una duración.

Además empiezo a configurar mi IDE siguiendo un tutorial a base de eclipse, librerías y plugins, pero se descubre que está obsoleto y nos

decantamos por la instalación de Kinetis Design Studio que lo integra todo.

Podemos ver con más detalle en el Sprint 1 del repositorio, las diferentes actividades que se acometieron. Se calculó que tardaría 12 horas y al final se dedicaron 13 horas.

Sprint 2 - 10/10/2018-17/10/2018

La segunda semana se termina de instalar Kinetis y se realiza la primera práctica de Sistemas Empotrados, basada en un *Hola Mundo* para familiarizarnos con el entorno de desarrollo. Una vez finalizada y ejecutada sin errores, el IDE se corrompe, no dejando cargar Processor Expert de ninguna de las maneras, quedando bloqueado.

Podemos ver con más detalle en el Sprint 2 del repositorio, las diferentes actividades que se acometieron. Se calculó que tardaría 9 horas y al final se dedicaron 16 horas.

Sprint 3 - 17/10/2018-24/10/2018

Los objetivos de este *sprint* fueron arreglar Kinetis y documentarme sobre la comunicación entre el módulo GPS hacia la placa, para ello me leí el manual *Adafruit Ultimate GPS Logger Shield*. Tras buscar mucha información por internet, hacer una reinstalación parcial de Processor Expert, reinstalar Kinetis desde cero, limpiar registros, configuraciones y parámetros, no se encuentra una solución definitiva, de forma que se opta por un formateo completo del ordenador. Además quedan soldadas las conexiones Digital I/O 0 y 1 con TX y RX respectivamente para transmitir los datos recogidos por el GPS a la placa.

Podemos ver con más detalle en el Sprint 3 del repositorio, las diferentes actividades que se acometieron. Se calculó que tardaría 45 horas y al final se dedicaron 50 horas.

Sprint 4 - 24/10/2018-31/10/2018

Una vez arreglado Kinetis, por fin se empieza con el proyecto en sí, el objetivo se marca en conseguir almacenar las coordenadas del acelerómetro en la tarjeta microSD. Después de instalar y configurar todos los componentes oportunos y de escribir el código, se consigue. Este *sprint* será el primero que se mostrará en ZenHub pero los anteriores se pueden ver en GitHub.

Podemos ver con más detalle en el Sprint 4 del repositorio, las diferentes actividades que se acometieron. Se calculó que tardaría 21 horas y al final se dedicaron 30 horas.

Sprint 5 - 07/11/2018-21/11/2018

Primer *sprint* que nos llevaría más de una semana, se plantea querer almacenar las coordenadas que arroja el GPS en la tarjeta microSD. Antes, las enviamos a la terminal Termite para ver qué muestra.

Conseguiremos almacenar información en la tarjeta microSD pero no nos daremos cuenta hasta unos días más tarde que los datos son incompletos.

Podemos ver con más detalle en el Sprint 5 del repositorio, las diferentes actividades que se acometieron. Se calculó que tardaría 42 horas y al final se dedicaron 46 horas.

Sprint 6 - 21/11/2018-28/11/2018

Esta fue la semana más caótica del proyecto, recién surgido el problema de la semana pasada se opta por avanzar sobre la funcionalidad de que solo se almacene información cuando se esté en movimiento gracias al acelerómetro. Reviso el manual del acelerómetro *FXOS8700CQ* que incorpora la placa, sin obtener ninguna conclusión válida sobre esta funcionalidad. Mientras tanto, sigo pensando cómo puedo solucionar el problema de la pérdida de información al escribir en la microSD.

Podemos ver con más detalle en el Sprint 6 del repositorio, las diferentes actividades que se acometieron. Se calculó que tardaría 2 horas y al final se dedicaron 10 horas.

Sprint 7 - 28/11/2018-05/12/2018

Finalmente, para tratar de arreglar la pérdida de caracteres, se opta por incluir en nuestro código un sistema operativo en tiempo real como *FreeRTOS*, de forma que administre las diferentes tareas. Agregamos dos componentes nuevos y escribimos las tareas para así, lograr el objetivo.

Podemos ver con más detalle en el Sprint 7 del repositorio, las diferentes actividades que se acometieron. Se calculó que tardaría 45 horas y al final se dedicaron 50 horas.

Sprint 8 - 05/12/2018-12/12/2018

Esta semana fue un alto en el camino, se decidió avanzar bastante con la memoria, con la que se llevaba algo de retraso y actualizar el *.gitIgnore* ya que estaba subiendo metadatos al repositorio.

Podemos ver con más detalle en el Sprint 8 del repositorio, las diferentes actividades que se acometieron. Se calculó que tardaría 22 horas y al final se dedicaron 26 horas.

Sprint 9 - 12/12/2018-26/12/2018

Durante estas dos semanas se elegirá la API con la que representaremos las coordenadas recogidas, finalmente nos decantamos por Mapbox y creamos una interfaz básica en html para poder probar nuestro fichero de coordenadas, después de convertirlo manualmente a la extensión *GeoJSON*.

Podemos ver con más detalle en el Sprint 9 del repositorio, las diferentes actividades que se acometieron. Se calculó que tardaría 29 horas y al final se dedicaron 30 horas.

Sprint 10 - 26/12/2018-09/01/2019

Se fijó como objetivo que la conversión entre NMEA y GeoJSON fuera transparente al usuario, había varias opciones pero finalmente se optó por ayudarnos de Mygeodata por una cuestión de tiempo, haciendo una petición a través de un *script* de Python.

Podemos ver con más detalle en el Sprint 10 del repositorio, las diferentes actividades que se acometieron. Se calculó que tardaría 8 horas y al final se dedicaron 20 horas.

Sprint 11 - 09/01/2019-16/01/2019

Ahora que la conversión del fichero funcionaba de forma óptima, era hora de juntar todo, se habilitó la subida de archivos al servidor, conseguimos ejecutar la conversión de forma autónoma y se descomprimió el directorio que generaba MyGeoData, para poder acceder al archivo convertido y representarlo en el mapa. Además se acometieron algunos cambios en la interfaz de la página.

Podemos ver con más detalle en el Sprint 11 del repositorio, las diferentes actividades que se acometieron. Se calculó que tardaría 7 horas y al final se dedicaron 15 horas.

Sprint 12 - 16/01/2019-23/01/2019

En este *sprint* se decidió que debíamos implementar alguna mejora en el código ya fuera de manera gráfica o alguna funcionalidad nueva. Conseguimos fechar los archivos subidos, añadimos el botón de "Descargar imagen" y el de "Centrar mapa en mi posición", intentamos *hostear* la página, finalizamos la memoria y empezamos los anexos.

Podemos ver con más detalle en el Sprint 12 del repositorio, las diferentes actividades que se acometieron. Se calculó que tardaría 10 horas y al final se dedicaron 11 horas.

Sprint 13 - 23/01/2019-30/01/2019

Esta semana la dedicamos a avanzar en los anexos, crear la máquina virtual con la configuración necesaria para el correcto visionado de la página web y añadimos a nuestra interfaz una plantilla HTML para que fuera más atractiva al usuario.

Podemos ver con más detalle en el Sprint 13 del repositorio, las diferentes actividades que se acometieron. Se calculó que tardaría 29 horas y se cumplió con la estimación.

Sprint 14 - 30/01/2019-06/02/2019

El objetivo primordial fue habilitar el acelerómetro para la detección de movimiento y en caso de no detectarlo, pausar el registro de coordenadas en la tarjeta microSD para no almacenar información de forma redundante. Además terminaremos los anexos y comprobaremos la calidad del código con Sonarcloud.

Podemos ver con más detalle en el Sprint 14 del repositorio, las diferentes actividades que se acometieron. Se calculó que tardaría 29 horas y se cumplió la estimación.

Sprint 15 - 06/02/2019-13/02/2019

Incorporamos excepciones al código, se hicieron los test para las pruebas del sistema, subimos los instaladores del proyecto y grabamos la ruta que se entrega con el proyecto.

Podemos ver con más detalle en el Sprint 15 del repositorio, las diferentes actividades que se acometieron. Se calculó que tardaría 16 horas y al final se dedicaron 15 horas.

A.3. Estudio de viabilidad

En este apartado se hará el análisis sobre la viabilidad económica y legal del proyecto.

Viabilidad económica

Coste de personal

Vamos a calcular el coste de tener a un programador a sueldo a lo largo de 18 semanas, o lo que es lo mismo 4 meses y medio. Promediando 22 horas a la semana a un precio por hora de 9€, el sueldo es el que se muestra:

Salario mensual
22 horas semana * 4 semanas * 15€ hora = 1320€

Tabla A.1: Salario mensual

A esos 1320€habría que sumarle los abonos a la seguridad social por parte de la empresa, en nuestro caso un 23.60 % por contingencias comunes, un 1.35 % por AT y EP (Accidente de Trabajo y Enfermedad Profesional), un 5.50 % por desempleo y un 0.60 por formación. [3] Hace un total de 31.05 %.

Salario mensual	1.320€
Seguridad Social (31.05 %)	409,86€
Coste mensual	1.729,86€
Por 4 meses y medio	
Coste total del personal	7.784,37€

Tabla A.2: Coste total del personal.

Coste de hardware

Para desarrollar el proyecto se ha hecho uso de un porátil con un Intel Core i3 M370, una Nvidia GeForce 315M, disco duro sólido de 240GB y 8GB de memoria RAM con un coste total de 700€. Consideraremos una vida útil de 7 años, por lo que el coste amortizado para 4 meses y medio son 37.5€. Calculemos ahora el coste total del hardware sumando la placa de desarrollo,el módulo GPS y la tarjeta microSD:

Portatil coste amortizado	37,5€
Placa de desarrollo FRDM-K64F [5]	32,52€
Ultimate GPS logger shield [4]	39,06€
Tarjeta microSD 16gb	3,9€
Coste total	112,98€

Tabla A.3: Coste total del hardware.

Otros costes

Tendremos en consideración otros gastos como internet:

Fibra óptica simétrica 42€/mes x 4 meses y medio = 189€

Coste total

Por lo tanto el coste total del proyecto sumando todas las cantidades anteriores es de:

Coste de personal	7.784,37€
Coste de hardware	112,98€
Otros costes	189€
Coste total	8.086,35€

Tabla A.4: Coste total del proyecto.

Beneficios

Una vez calculados los gastos por desarrollar el proyecto es hora de calcular los posibles beneficios de la venta del proyecto o bien de su explotación. Teniendo en cuenta que los costes han rondado los 8.000€, con que hubiera un comprador dispuesto a ofrecer 12.000€y sacar 4.000€de beneficios nos daríamos por satisfechos.

El sistema está orientado a toda la población en general que tenga intención de registrar una ruta, ya sea gente deportista o del mundo del motor, como también jefes a empleados en una flota de vehículos o padres controladores desconfiados de sus hijos y viceversa.

Aparte de la venta, también podría interesarnos un sistema de suscripciones al servicio de forma que, o bien compraran ellos el *hardware* o se lo alquilamos y pagan por el derecho de ingresar a la página y subir sus ficheros de coordenadas.

A un precio de 20€mensuales por suscripción, con 34 personas suscritas durante 1 año conseguiríamos recuperar todo el dinero invertido.

Sin duda es un riesgo asumible teniendo en cuenta que el coste del proyecto no es muy elevado y los ingresos pueden dispararse a corto medio plazo. Ideal para inversores con no mucho capital, dispuestos a arriesgar y con opciones a disparar sus beneficios.

Viabilidad legal

Hablando desde el punto de vista de la viabilidad legal del proyecto, éste se encuentra bajo una licencia de tipo, "GNU General Public License v3.0" la cual permite el uso comercial, la modificación, su distribución, el uso privado y el uso de patentes. Evitando así cualquier tipo de responsabilidad y garantía. [1]

Si hablamos de los términos de uso de Mapbox expresan de forma clara que no debes vender su código a terceros pero no hay problema si tu aplicación lo integra para dar servicio a tus propios usuarios finales. [7]

La plantilla HTML que hemos incluido[10] en nuestro proyecto cuenta con la licencia MIT (Massachusetts Institute of Technology), cuya única condición es que se distribuya la licencia original con todas las copias del software que se hagan.

Por último Mygeodata recalca que no debes reproducir, duplicar, vender o utilizar sus servicios si lo vas a hacer con mala intención. [9]

Apéndice B

Especificación de Requisitos

B.1. Introducción

En este apartado se incluyen los objetivos y requisitos marcados al inicio, entre el alumno y el tutor para el proyecto. De manera adicional, se han añadido algunos no establecidos al inicio pero que se han considerado pertinentes a lo largo del desarrollo.

B.2. Objetivos generales

El desarrollo de este proyecto tiene el objetivo de desentramar la cadena NMEA aportada por el módulo GPS adherido a la placa de desarrollo FRDM-K64F, almacenar los mensajes NMEA en la tarjeta microSD activándose con el movimiento y conseguir mostrar en un mapa la ruta seguida por el usuario.

B.3. Catálogo de requisitos

En esta sección detallaremos los requisitos funcionales y no funcionales del proyecto.

Requisitos funcionales

- **RF1-Conexión del GPS:** El GPS debe establecer conexión satelital.
- **RF2-Almacenar las coordenadas:** La placa guarda la información en la tarjeta.

- **RF3-Almacenamiento de archivos NMEA:** Subir el archivo del usuario al servidor.
- **RF4-Conversión de archivos:** Transformar el archivo NMEA a uno con extensión GeoJSON.
- **RF5-Descompresión del fichero:** Descomprimir el fichero generado por MyGeoData.
- **RF6-Mostrar mapa:** Representa en un mapa las coordenadas del fichero.
- **RF7-Geolocalizar posición:** Geolocaliza tu posición y centra el mapa en donde estés.
- **RF8-Descargar mapa:** Descarga el mapa que se visualice en ese momento como una imagen PNG.
- **RF9-Modificar vista del mapa:** Añade o quita zoom, rota el mapa sobre su eje o cambia de vista.
- **RF10-Mensaje de error:** En caso de subir un fichero no admitido se mostrará un mensaje de error con el motivo.

Requisitos no funcionales

- **RNF1-Usabilidad:** El usuario debe sentirse familiarizado con la interfaz y ésta ha de ser *user friendly*. Además debe ser *responsive design* de forma que la web se amolde a las diferentes pantallas.
- **RNF2-Eficiencia:** tanto el módulo GPS almacenando coordenadas, como la web mostrando la ruta, deben tener el menor tiempo de respuesta posible.
- **RNF3-Escalabilidad:** El proyecto debe ser escalable para que se puedan aumentar sus funcionalidades de forma rápida y sencilla.
- **RNF4-Mantenibilidad:** Deben proporcionarse mejoras y mantenimiento de forma periódica.
- **RNF5-Confiabilidad:** El proyecto ha de realizar las funciones para las que fue creado obteniendo los resultados esperados.
- **RNF6-Disponibilidad:** La aplicación debe estar disponible el máximo tiempo posible, minimizando posibles caídas temporales del sistema.

B.4. Especificación de requisitos

RF 1	**Conexión del GPS.**
Descripción	El GPS debe establecer conexión satelital.
Precondiciones	A la placa de desarrollo se le suministra corriente.
Acciones	El módulo GPS busca comunicación con los satélites.
Postcondiciones	El GPS transmite su posición 1 vez por segundo.
Importancia	Alta.

Tabla B.1: RF1 - Conexión del GPS.

RF 2	**Almacenar las coordenadas.**
Descripción	La placa guarda la información en la tarjeta.
Precondiciones	Se ha inicializado el sistema de almacenamiento y hay una tarjeta introducida.
Acciones	La placa comienza a escribir en la tarjeta microSD.
Postcondiciones	Obtenemos un fichero con los mensajes NMEA transmitidos por el GPS.
Importancia	Alta.

Tabla B.2: RF2 - Almacenar las coordenadas.

RF 3	**Almacenamiento de archivos NMEA.**
Descripción	Subir el archivo del usuario al servidor.
Precondiciones	El usuario selecciona el archivo que quiere subir.
Acciones	El archivo es enviado a una carpeta temporal y es movido a “subidas”.
Postcondiciones	Queda almacenado el archivo.
Importancia	Alta.

Tabla B.3: RF3 - Almacenamiento de archivos NMEA.

RF 4	**Conversión de archivos.**
Descripción	Transformar el archivo NMEA a uno con extensión GeoJSON.
Precondiciones	El usuario debe seleccionar un archivo local y subirlo.
Acciones	Se ejecuta el script en Python de la API de MyGeoData.
Postcondiciones	Se genera un fichero con extensión zip.
Importancia	Alta.

Tabla B.4: RF4 - Conversión de archivos.

RF 5	**Descompresión del fichero.**
Descripción	Descomprimir el fichero generado por MyGeoData.
Precondiciones	Debe existir el fichero comprimido de MyGeoData.
Acciones	Se descomprime el fichero.
Postcondiciones	Aparecen los archivos descomprimidos dentro de una carpeta.
Importancia	Alta.

Tabla B.5: RF5 - Descompresión del fichero.

RF 6	**Mostrar mapa.**
Descripción	Representa en un mapa las coordenadas del fichero.
Precondiciones	Disponer de un archivo en formato GeoJSON.
Acciones	La API de Mapbox utiliza ese archivo.
Postcondiciones	Se representa un mapa con la ruta.
Importancia	Alta.

Tabla B.6: RF6 - Mostrar mapa.

RF 7	**Geolocalizar posición.**
Descripción	Geolocaliza tu posición y centra el mapa en donde estés.
Precondiciones	El mapa debe estar parcialmente cargado.
Acciones	El usuario selecciona “Centrar en mi posición”.
Postcondiciones	Se obtiene un mapa centrado en tu posición actual.
Importancia	Baja.

Tabla B.7: RF7 - Geolocalizar posición.

RF 8	**Descargar mapa.**
Descripción	Descarga el mapa que se visualice en ese momento como una imagen PNG.
Precondiciones	El mapa debe estar cargado por completo.
Acciones	El usuario selecciona “Descargar mapa”.
Postcondiciones	Se inicia la descarga del mapa.
Importancia	Baja.

Tabla B.8: RF8 - Descargar mapa.

RF 9	**Modificar vista del mapa.**
Descripción	Añade o quita zoom, rota el mapa sobre su eje o cambia de vista.
Precondiciones	El mapa debe estar parcialmente cargado.
Acciones	El usuario selecciona algún botón de modificación del mapa de la vista actual o se desplaza a través de él.
Postcondiciones	Se obtiene un mapa con diferente vista.
Importancia	Baja.

Tabla B.9: RF9 - Modificar vista del mapa.

RF 10	**Mensaje de error.**
Descripción	En caso de subir un fichero no admitido se mostrará un mensaje de error con el motivo.
Precondiciones	Haber seleccionado un fichero para subir.
Acciones	El usuario presiona “Subir archivo”.
Postcondiciones	De ser un fichero inválido saltará una alerta explicando el motivo.
Importancia	Media.

Tabla B.10: RF10 - Mensaje de error.

B.5. Diagrama de casos de uso

En esta sección mostraremos el diagrama de casos de uso de nuestra aplicación.

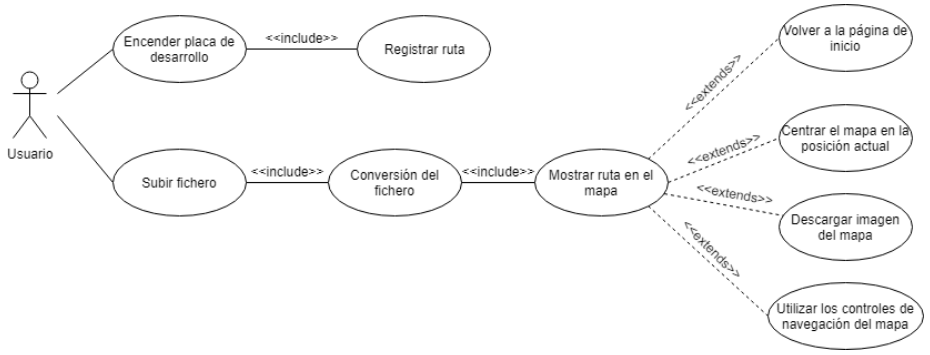

Figura B.1: Diagrama de casos de uso del proyecto

Apéndice C

Especificación de diseño

C.1. Introducción

En esta sección detallaremos el diseño procedimental y el arquitectónico.

C.2. Diseño procedimental

En este apartado veremos el diseño interno del proyecto, especificando los detalles de los diferentes algoritmos. En este caso nos vamos a ayudar de un diagrama de secuencias que explicará el caso más completo que se podría dar en el proyecto.

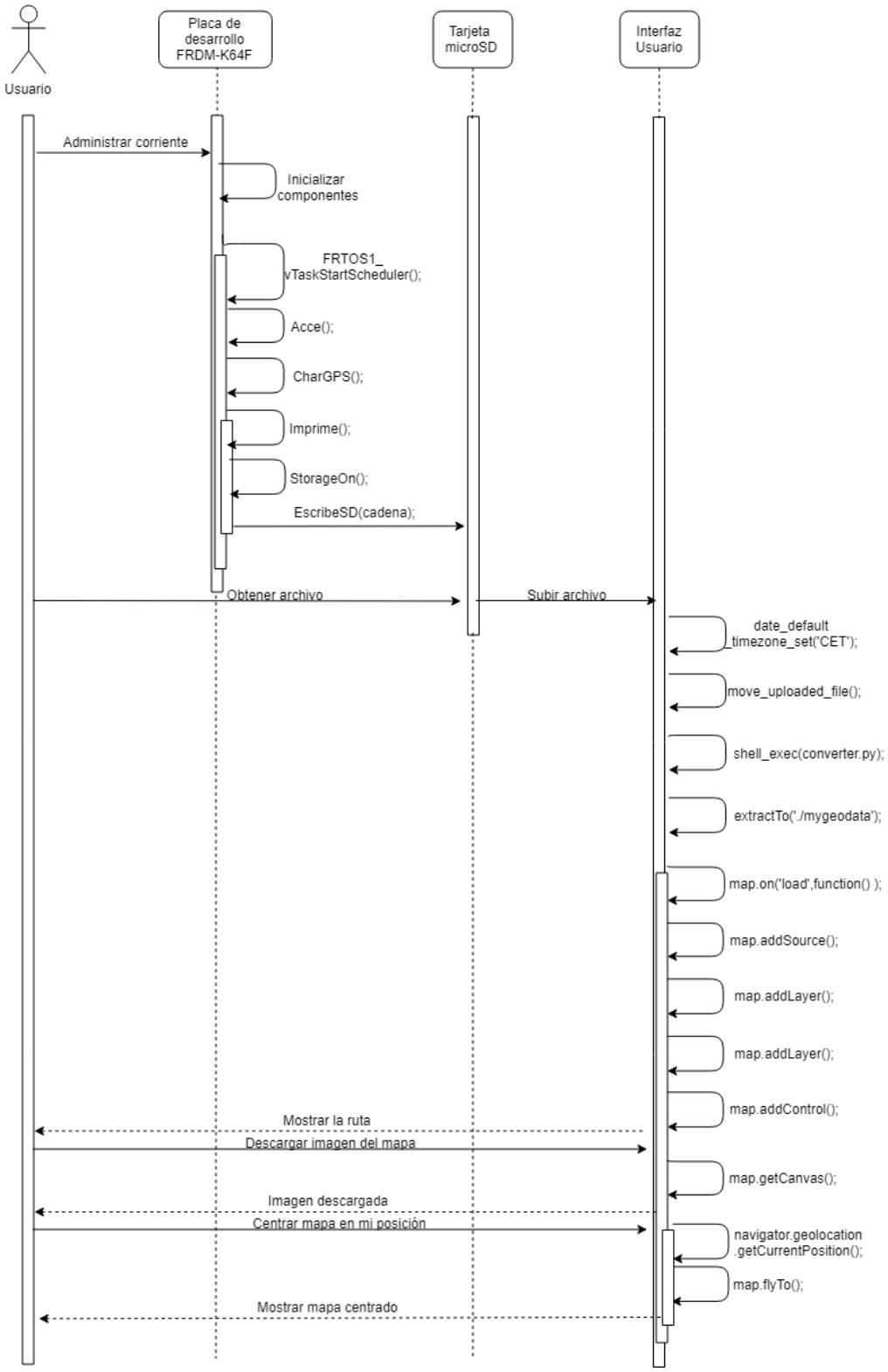

Figura C.1: Diagrama de secuencias del proyecto.

C.3. Diseño arquitectónico

En esta sección veremos como se relacionan los diferentes módulos. Desglosaremos el diseño arquitectónico en dos secciones ya que contamos con dos partes claramente diferenciadas.

Sistema operativo a tiempo real (RTOS)

La arquitectura de programación utilizada en el funcionamiento de la placa de desarrollo es comúnmente denominada como "Sistema operativo a tiempo real" o RTOS [8]. Usada en proyectos donde la complejidad es alta y es imprescindible la inclusión de requerimientos de tiempo real, el sistema operativo se encarga de decidir en que momento se debe ejecutar cada tarea a través de interrupciones también autogestionadas. En nuestro caso distinguimos las tareas de almacenar en la cadena los datos enviados por el GPS, la de comprobar los valores del acelerómetro para parar la escritura de datos en caso necesario y la de imprimir los caracteres de la cadena.

En cuanto se le transmite corriente a la placa, ejecuta el código que alberga en la memoria flash, inicializa los componentes, las tareas y por último, el sistema operativo de tiempo real que las administra.

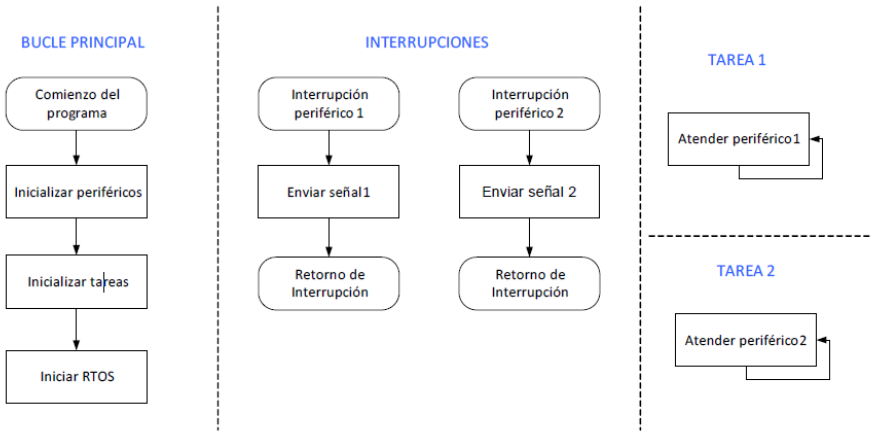

Figura C.2: Diagrama básico de la arquitectura RTOS.[8]

Modelo Vista Controlador

El patrón utilizado para la interfaz web ha sido MVC[6]. Se trata de uno de los diseños más usados y separa la vista respecto a la lógica de la aplicación. Intervienen en este diseño 3 componentes:

- **Modelo:** Se encarga de los datos y maneja la información con la que funciona el proyecto. También conocida como lógica de negocio. En nuestro caso se trata de la API de Mapbox y la de MyGeoData.
- **Vista:** Se trata de la interfaz, la parte más visible a través de la cual el usuario solicita las acciones y donde se muestra la información. Nuestra página web cumpliría con esta misión mostrando los datos, en este caso la ruta del usuario en el mapa de la forma que él ha especificado.
- **Controlador:** Es el intermediario entre los dos anteriores, recoge las peticiones del usuario para pasárselas al modelo y devuelve la información a la vista que ha transmitido éste. En nuestro proyecto cobra un papel relevante al ser el encargado de recoger el archivo al usuario, convertirlo y pasárselo al modelo con la configuración solicitada, para que éste después nos devuelva la información que debe ser mostrada en base a los datos aportados.

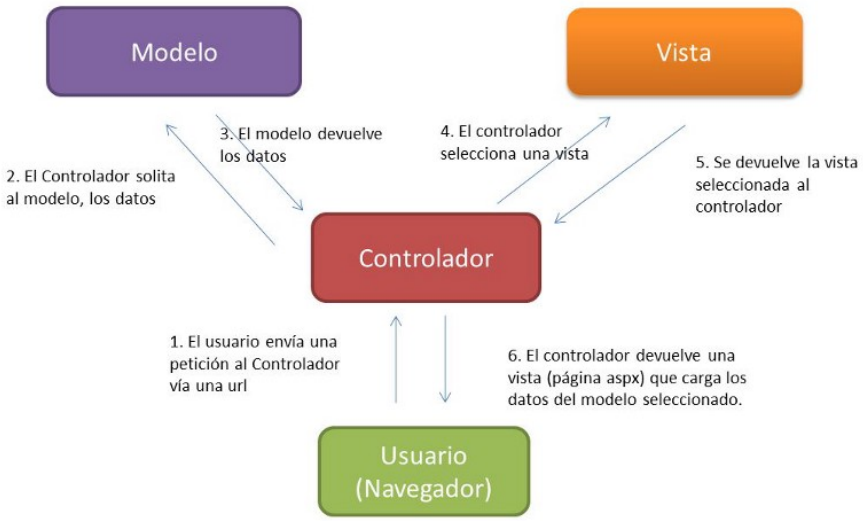

Figura C.3: Comportamiento de los 3 componentes.[2]

Apéndice D

Dossier técnico de programación

D.1. Introducción

En este manual vamos a detallar de la forma más precisa posible, cómo reanudar este proyecto para que se pueda seguir con el desarrollo si así se desea.

D.2. Estructura de directorios

Este proyecto se encuentra con la siguiente distribución de carpetas:

- **src:** se encuentra todo el código de la aplicación.
- **src/Interface:** contiene el código de la interfaz, tanto la página web como el convertidor y algunos ficheros de ejemplo.
- **src/Interface/subidas:** aloja los archivos subidos a la página web.
- **src/Interface/mygeodata:** almacena los archivos convertidos a GeoJSON y descomprimidos.
- **src/DattaLogger:** contiene el código que lleva incorporada la placa de desarrollo, encargado de almacenar las coordenadas en función del movimiento.
- **Documentación:** nos encontramos con los PDF de la memoria y anexos, los archivos de *latex* y las imágenes.

- **Selenium Test:** contiene los test creados con Selenium IDE para la interfaz.

- **Instalables:** contiene un link a todos los ejecutables que se han necesitado en el proyecto. (*Kinetis*, *Termite*, *Python* y *XAMPP*).

D.3. Manual del programador

Instalación Kinetis Design Studio

La primera parte del proyecto (la de la placa GPS) se ha desarrollado con la última versión de Kinetis (3.2.0) disponible en la página web oficial de NXP . A continuación se aporta el enlace con la última versión:

https://nxp.flexnetoperations.com/control/frse/product...

Instalación de Termite

La terminal utilizada para visualizar los datos recogidos por la placa de desarrollo ha sido Termite en su versión 3.4 *(complete setup)*. En el siguiente enlace encontraremos la última versión de Termite.

https://www.compuphase.com/software_termite.htm

Instalación de Python

Para poder ejecutar el archivo de conversión de forma local necesitaremos Python en nuestro ordenador. En este caso ha sido utilizado Python 3.7.0. En el siguiente enlace podemos encontrar la última versión de Python disponible.

http://www.oracle.com/technetwork/java/javase/downloads...

Instalación de XAMPP

Para poder ejecutar nuestra página en nuestro ordenador necesitaremos emular un servidor local, para lo que he utilizado XAMPP. Para Además necesitaremos hacer algunos ajustes de configuración para que reconozca archivos Python (extensión .py). Voy a indicar el directorio por defecto, pero cada uno tendrá que ir a su directorio de instalación.

- En el archivo de configuración C:\xampp\apache\conf\httpd.conf añadir la extensión .py en la línea del *AddHandler* de forma que quede: AddHandler cgi-script .cgi .pl .asp .py
- En la línea *Options Indexes FollowSymLinks* debe tener la configuración *ExecCGI*.
- La primera línea de script de python debe ser la ruta a python.
- Por último debemos reiniciar XAMPP para que se apliquen los cambios.

```
426     #
427     # AddHandler allows you to map certain file extensions to "handlers":
428     # actions unrelated to filetype. These can be either built into the server
429     # or added with the Action directive (see below)
430     #
431     # To use CGI scripts outside of ScriptAliased directories:
432     # (You will also need to add "ExecCGI" to the "Options" directive.)
433     #
434     AddHandler cgi-script .cgi .pl .asp .py
```

Figura D.1: Cambios necesarios en los archivos de configuración sobre el manejador.

```
# Possible values for the Options directive are "None", "All",
# or any combination of:
#   Indexes Includes FollowSymLinks SymLinksifOwnerMatch ExecCGI MultiViews
#
# Note that "MultiViews" must be named *explicitly* --- "Options All"
# doesn't give it to you.
#
# The Options directive is both complicated and important.  Please see
# http://httpd.apache.org/docs/2.4/mod/core.html#options
# for more information.
#
Options Indexes FollowSymLinks ExecCGI
```

Figura D.2: Inclusión de ExecCGI en las opciones.

Para la ejecución local, debemos meter la carpeta *interface* del proyecto en la carpeta de XAMPP *htdocs*, en mi caso: *C:\xampp\htdocs*

A continuación se encuentra el enlace a la página de Apache donde podremos descargar la última versión de XAMPP: https://www.apachefriends.org/es/download.html

D.4. Compilación, instalación y ejecución

Importar el proyecto en Kinetis

Debemos descargar el proyecto exportado comprimido y subido a GitHub que contiene no solo el código fuente si no todos los componentes y la configuración: https://github.com/alejandrolampreave/GPS. A continuación lo descomprimimos y ya podemos importarlo en Kinetis.

Para importar en Kinetis hacemos clic en *File >Import*. Seguidamente seleccionamos la opción *Existing Projects into Workspace* y seleccionamos la carpeta descomprimida.

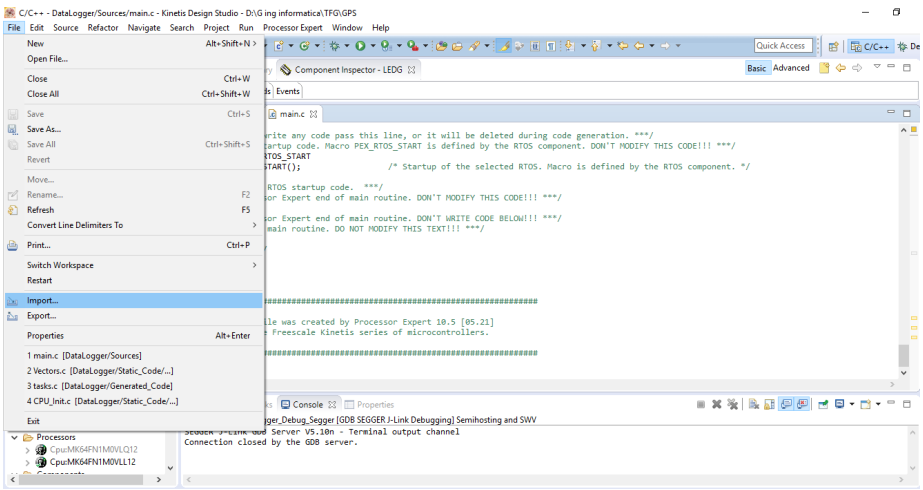

Figura D.3: File>Import para importar nuestro proyecto a Kinetis.

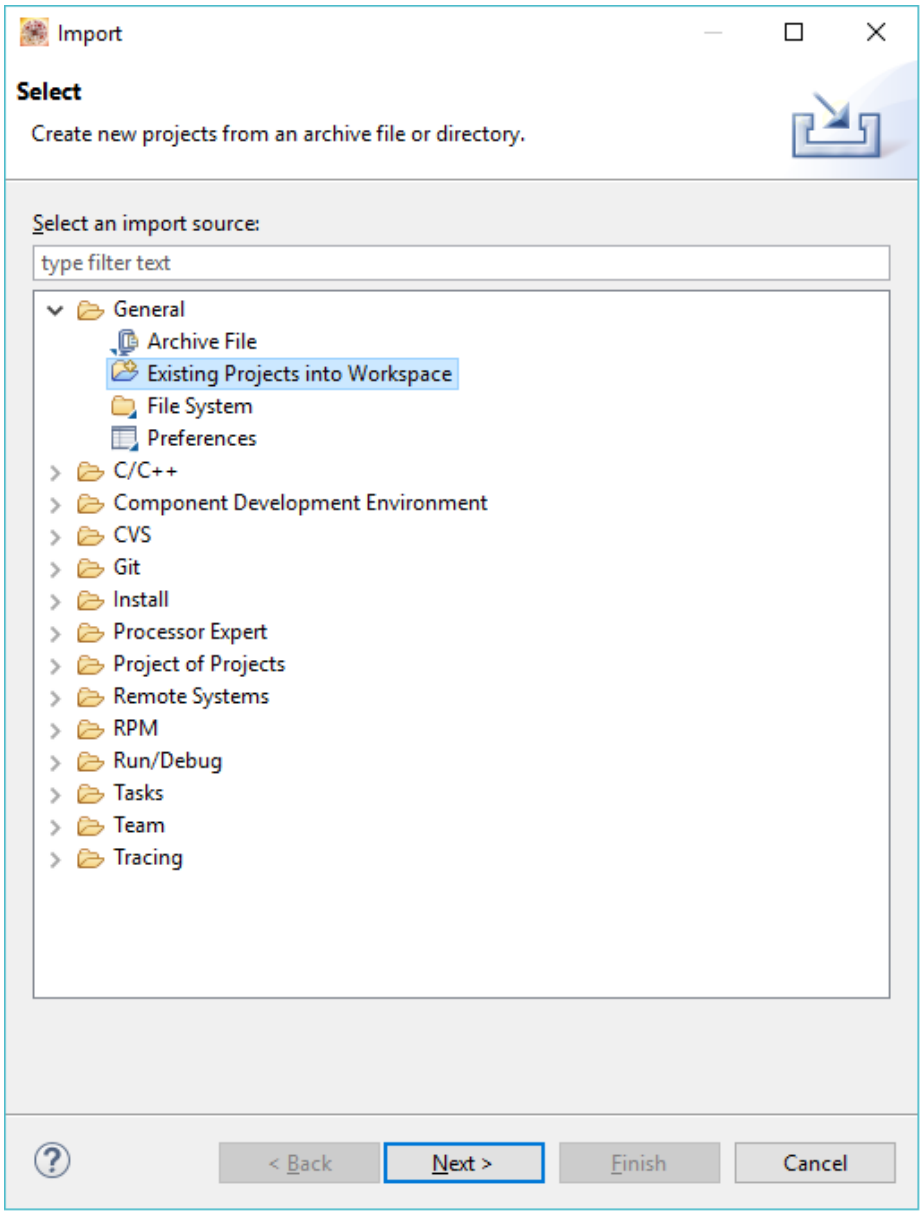

Figura D.4: Seleccionamos “Existing Projects into Workspace”.

Ya podríamos empezar a trabajar en el proyecto.

Grabar el programa en la placa

Para grabar el programa en la placa de desarrollo tan solo debemos conectar la placa de desarrollo al conector de la placa de desarrollo OpenSDA

y ejecutar nuestro programa dando al *Run*, observando que la memoria seleccionada en ese momento sea la *flash* y no la RAM.

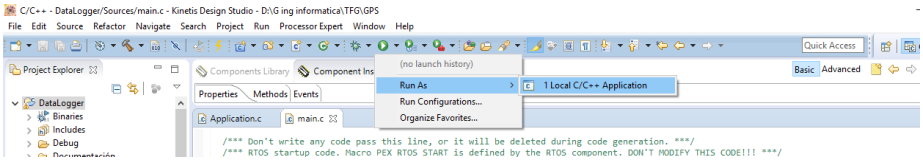

Figura D.5: Ejecutamos el programa para que quede almacenado en la memoria *Flash* de la placa.

Ejecución

Por último para ejecutar, solo tendremos que iniciar el servicio de Apache de XAMPP, abrir el navegador e introducir la siguiente url: `http://localhost/interface/index.html` para que se nos muestre la página de inicio de forma local. Una vez cargada podremos subir nuestro archivo, recogido por nuestro GPS y almacenado en la tarjeta microSD, para que se nos muestre en el mapa la ruta que hayamos hecho.

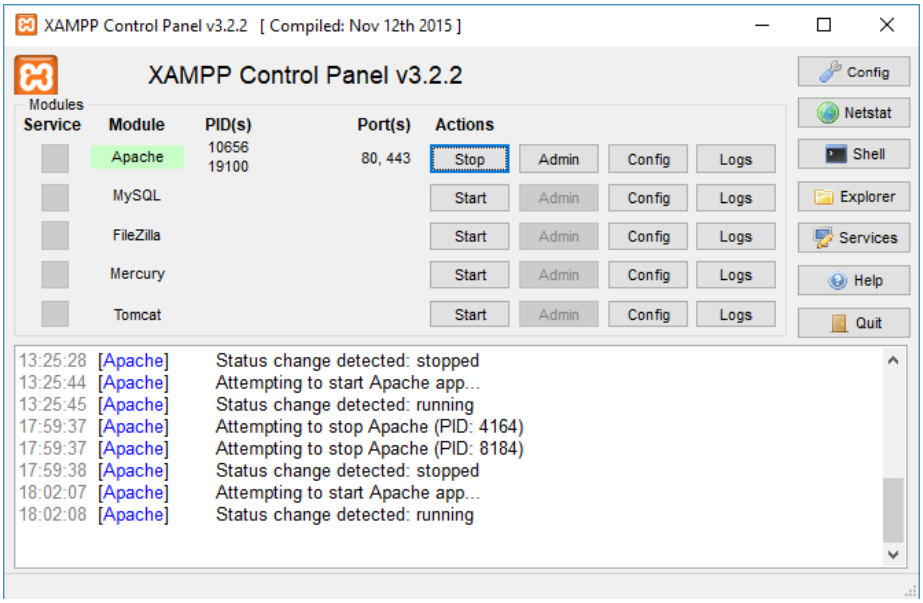

Figura D.6: Iniciamos el servicio de Apache.

D.5. Pruebas del sistema y calidad del código

A lo largo de todo el desarrollo del proyecto se ha ido probando todo lo que se iba añadiendo para asegurarnos que no se perdiera ninguna funcionalidad a medida que avanzábamos. En caso de que diera algún error o algo no saliera según lo esperado, se replanteaba lo que se quería implementar y se buscaban posibles soluciones dentro de la misma opción o se pensaba en una alternativa válida.

Tanto desde el punto de vista del usuario como para nosotros mismos se habilitó el *led* rgb de la placa de forma que nos indicase en todo momento qué estaba haciendo, *led* verde obtención de coordenadas, *led* rojo escritura en tarjeta. Además se ha comprobado que únicamente se permita la subida de archivos con extensión .txt, que es como se almacenan en la tarjeta.

De forma adicional se ha hecho uso de la herramienta de Selenium IDE, con la cual se ha desarrollado un pequeño test que comprobase la carga de los diferentes bloques de la página, incluyendo el mapa final. Se adjuntan los test junto con el proyecto.

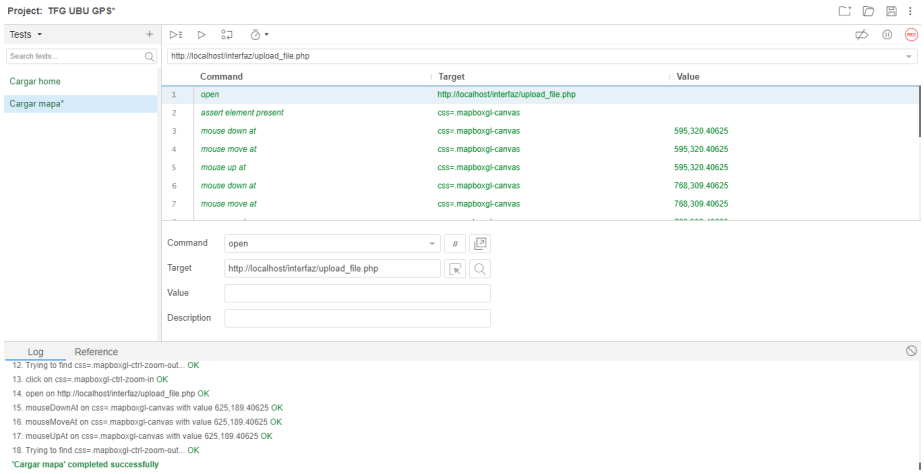

Figura D.7: Test realizado mediante Selenium IDE.

En la última fase del proyecto se realizó una prueba en un ambiente real, dentro de un coche con la placa de desarrollo, para que registrara la ruta. Luego se introdujo la tarjeta en el ordenador y se subió a la página web, mostrando de forma satisfactoria la ruta seguida.

En cuanto a la calidad del código se refiere barajamos varias herramientas, entre ellas: SonarQube, Codacy, CodeClimate y CodeBeat. Finalmente seleccionamos SonarQube por ser una de las más importantes y que mejor se adaptaba al proyecto. En su página web seleccionamos el método *online*, nos redireccionó a una página llamada Sonarcloud y tras descargarnos el "sonar-scanner-3.3.0.1492" ejecutamos un comando que nos dieron para generar el análisis. Cabe destacar que solo hemos podido pasar el escáner a los archivos de la interfaz, ya que al pasar el *build wraper* proporcionado por la herramienta por nuestros ficheros fuentes en C no los reconocía. Diría que es porque la *Build Tool* utilizado en Kinetis no es GCC si no uno específico llamado *Cross ARM GCC*.

Obtuvimos buenos resultados pero tras unas pequeñas mejoras, incluyendo algunas de la plantilla HTML utilizada, conseguimos mejorar la calidad de nuestro código sustancialmente. Algunos de los errores fueron marcados como falsos positivos, pero fueron acompañados por una explicación en tal caso. El link para comprobar la calidad del código es: https://sonarcloud.io/dashboard?id=alejandrolampreave_GPS

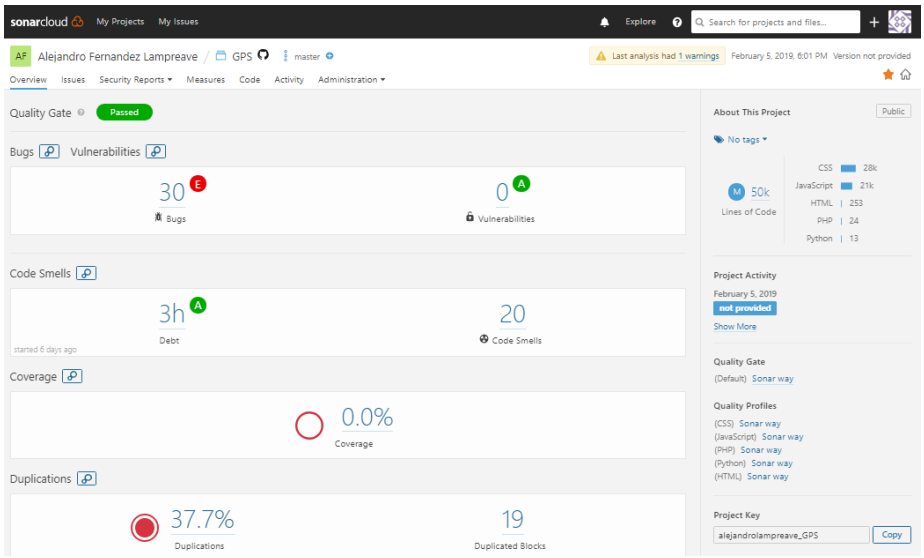

Figura D.8: Escaneo realizado antes de corregir el código.

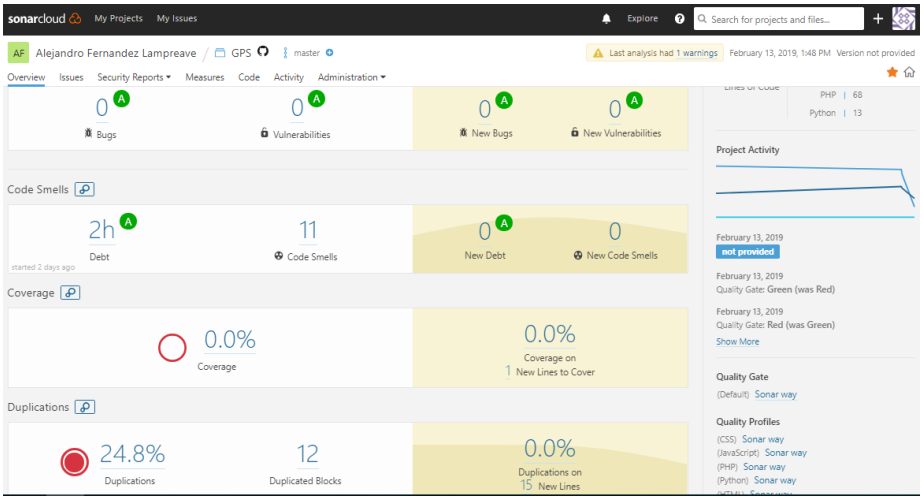

Figura D.9: Escaneo realizado tras corregir la mayoría de errores.

Apéndice E

Documentación de usuario

E.1. Introducción

Vamos a explicar cómo grabar una ruta con la placa GPS, así como la subida del fichero generado a la web para poder visualizarla en el mapa.

E.2. Requisitos de usuarios

Para poder registrar una ruta necesitamos:

- Disponer de una placa de desarrollo FRDM-K64F con un módulo GPS incorporado y cableado como se ha detallado en la memoria.
- Tener grabada en su memoria flash (como se entrega) el programa que le va a permitir almacenar las coordenadas.
- Disponer de un ordenador con una correcta instalación de XAMPP y Python (dado que no está *hosteada*).
- Un navegador instalado y conexión a internet.
- Una tarjeta microSD.
- Un soporte *hardware* que permita leer tarjetas microSD.

E.3. Instalación

El proyecto, propiamente dicho, no requiere ser instalado previamente en un ordenador para funcionar, dado que se trata de una página web y la

placa GPS va con el programa incorporado, pero sí se debe haber hecho una correcta configuración previa, como ya se ha hecho referencia.

E.4. Manual del usuario

Una vez que está todo correctamente configurado, solo queda poder utilizar el proyecto. Para ello vamos a dividirlo en dos partes claramente diferenciadas.

Grabar la ruta

Debemos suministrar corriente a la placa de desarrollo, para lo cual utilizaremos una batería externa conectada por micro USB, que nos brinda máxima movilidad. Debemos esperar a que consiga conexión con los satélites, el led *fix* rojo lo indicará cuando deje de parpadear de forma constante, y lo haga cada 3 segundos aproximadamente.

Se recomienda que el módulo GPS esté lo menos cubierto posible, ya que ayudará a su rápida conexión, de no ser así podría tardar hasta media hora. Se recuerda que en caso de necesidad se le podría acoplar una antena GPS externa, que le otorga mejor conexión. Ahora solo queda empezar a moverse y automáticamente se empezarán a almacenar las coordenadas, el led rojo será en encargado de indicar que se están almacenando las coordenadas correctamente en la tarjeta microSD. En caso de detenerse se parará el almacenamiento, para una mayor eficiencia.

Mostrar la ruta

Una vez finalizada la ruta, simplemente debemos extraer la tarjeta microSD de la placa e introducirla a nuestro ordenador. Veremos un archivo llamado "LOG_GPS_NMEA.txt" ese será nuestro archivo, abrimos la página web, `http://localhost/interface/index.html` y daremos a "Seleccionar archivo", escogeremos nuestro archivo y seleccionaremos "Subir archivo".

De forma automática nuestras coordenadas serán almacenadas en el directorio */subidas*, las convertirá a formato GeoJSON y las almacenará en */mygeodata* y se mostrará la ruta en el mapa.

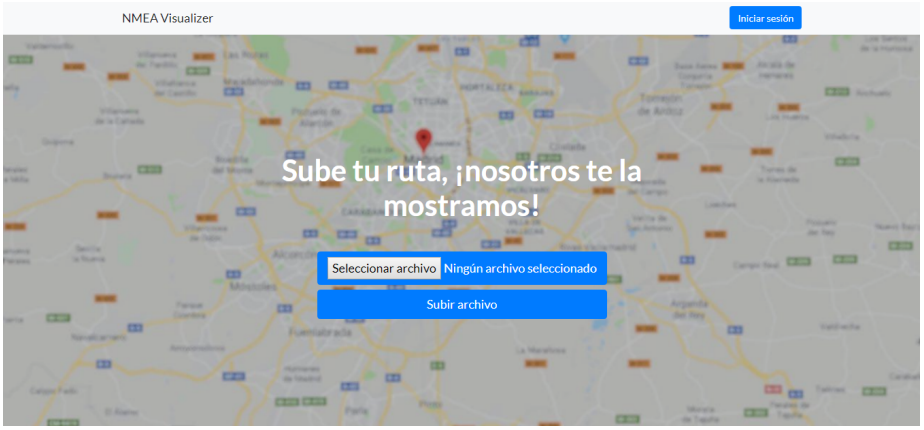

Figura E.1: Página principal donde subiremos los archivos.

Información de la página

A modo informativo se nos muestra una serie de cajas con información sobre el proyecto. Así como de presentación para posibles nuevos usuarios.

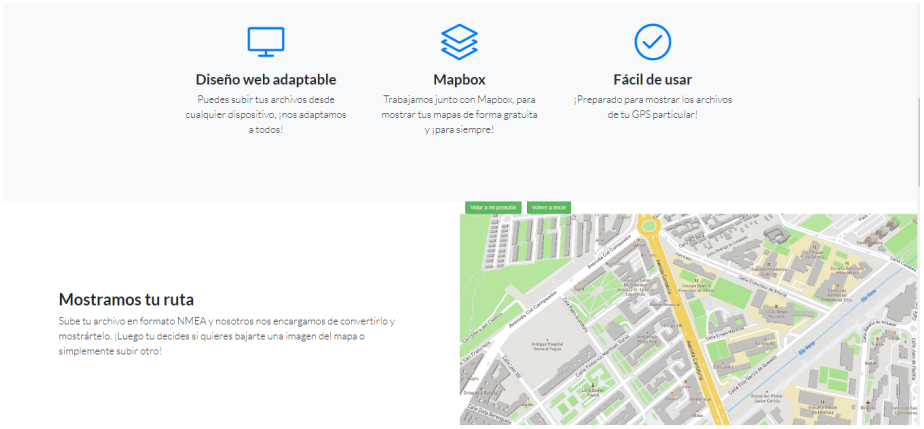

Figura E.2: Características generales de la página y funcionalidad básica.

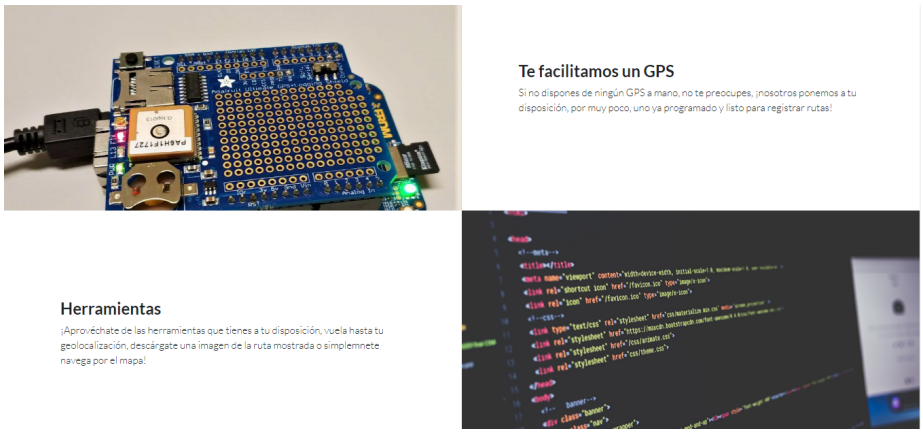

Figura E.3: Herramientas del mapa y opción a GPS.

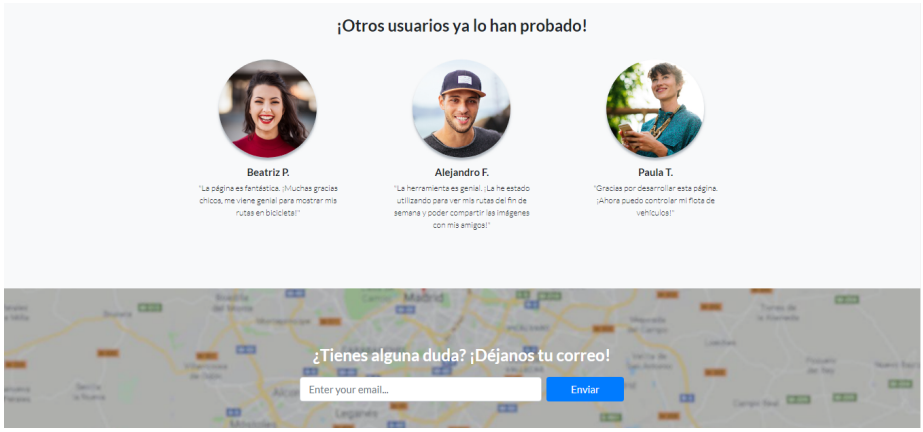

Figura E.4: Opiniones y preguntas.

Opciones de la interfaz

Se nos presentan algunas herramientas en la ventana donde se muestra la ruta. La primera “Centrar en mi posición”, te saltará una solicitud para poder detectar tu posición, tras aceptarla, el mapa se centrará en tu localización. También tienes la opción de moverte por el mapa, tienes a tu disposición unos controles a la derecha para poder añadir o quitar zoom y volver a orientarte hacia el norte. Además, podemos descargar la imagen con la ruta en alta resolución si seleccionamos “Descargar imagen”.

Seleccionando con el botón derecho del ratón se nos permite cambiar en ángulo de visión del mapa y rotarlo. De esta forma podemos visualizar los

edificios en 3D.

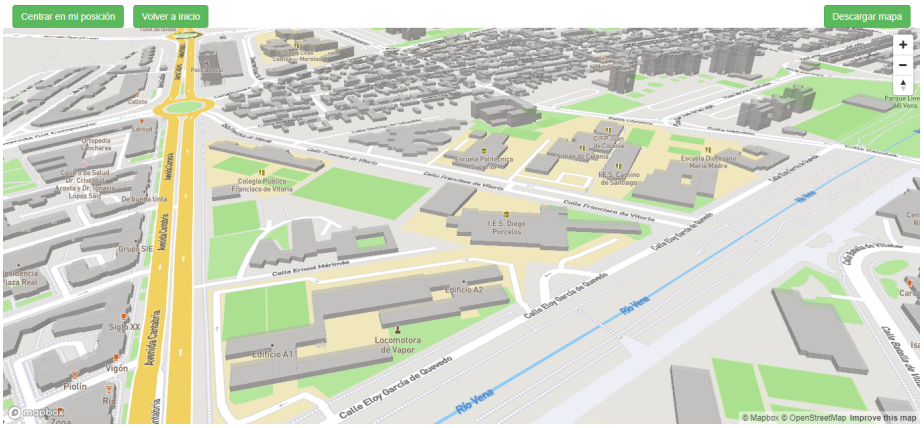

Figura E.5: Edificios en 3D tras haber cambiado la vista.

Bibliografía

[1] Fernández Alejandro. Gps/license. https://github.com/alejandrolampreave/GPS/blob/master/LICENSE, 2019.

[2] Davidenq. Entendiendo m de mvc y sus problemas. https://medium.com/@davidenq/entendiendo-m-de-mvc-y-sus-problemas-ebc0cbf518ec, 2016.

[3] Lorenzana Diego. ¿cuánto paga tu empresa por ti? quizá más de lo que piensas. https://www.pymesyautonomos.com/fiscalidad-y-contabilidad/cuanto-paga-tu-empresa-por-ti-quiza-mas-de-lo-que-piensas, 2017.

[4] Digikey. Información general del producto. https://www.digikey.es/product-detail/es/adafruit-industries-llc/1272/1528-1045-ND/5011061?utm_adgroup=Evaluation+Boards+-+Expansion+Boards%2C+Daughter+Cards&mkwid=s&pcrid=278710408307&pkw=&pmt=&pdv=c&productid=5011061&slid=&gclid=Cj0KCQiA7IDiBRCLARIsABIPohjofgS_TvYYhJeI5Rp_R-9t6Cai-CTBbLWxLFP16HRyekHkX-Qm0ysaAjeUEALw_wcB, 2019.

[5] Farnell. Frdm-k64f - placa de desarrollo, mk64fn1m0vll12 ethernet/usb, acelerómetro y magnetómetro. https://es.farnell.com/nxp/frdm-k64f/placa-desarrollo-ethernet-usb/dp/2406741, 2018.

[6] Uriel Hernandez. Mvc (model, view, controller) explicado. https://codigofacilito.com/articulos/mvc-model-view-controller-explicado, 2015.

[7] Mapbox. Geocoding api. https://www.mapbox.com/tos/#geocoding, 2014.

[8] Alejandro Merino. *Diseño de aplicaciones embebidas*, chapter 3, page 18. Universidad de Burgos, 2017. Máster Universitario en Ingeniería Informática, Sistemas Empotrados y Ubicuos.

[9] Mygeodata. General conditions. https://mygeodata.cloud/terms-of-service/, 2018.

[10] Startbootstrap. Landing page. a simple, elegant, and beautifully responsive landing page theme for bootstrap 4 websites. https://startbootstrap.com/template-overviews/landing-page/, 2019.