Universidad de Valladolid

Escuela de Ingeniería Informática

Grado en Ingeniería Informática
Mención De Computación

TRABAJO FIN DE GRADO

Tablero de tinta electrónica de información y avisos en accesos a dependencias

Autor:
Fernando Gigosos Ronda

Tutor:
Dr. César Llamas Bello

Dedicado a la memoria de Alan Turing, un referente para mí en infinidad de aspectos.

Agradecimientos

Dedico un enorme agradecimiento a los profesores que me han guiado y ayudado: mi tutor César Llamas, con el que he pasado reuniones tan divertidas y el catedrático de electromagnetismo Jose María Muñoz que ha aportado su experto criterio técnico e increíbles habilidades de soldadura.

A mis amigos y compañeros de vida: a David Paramio, que me apoya siempre que lo merezco y está ahí para lo que sea, a Eduardo García que me ha acompañado en mi período universitario y mas allá, a Luna Krstić, que siempre eleva el nivel de profesionalidad de todo lo que hago y a Duero Joshua, la persona cuya opinión siempre considero importante.

Y, sobre todo, a mis Padres que tanto me han ayudado y aguantado: Marco Antonio Gigosos que además de padre ha ejercido de experto profesor de sistemas UNIX y Feli Ronda que además de madre ha ejercido de mi conciencia.

Resumen

Es habitual que, a pesar de los avances tecnológicos, cuando un alumno acude a un despacho de profesor que éste no se encuentre allí en ese momento. Aunque esto no suponga una gran pérdida de tiempo, la tecnología a nuestro alcance nos permite transformar esta situación de incertidumbre para el alumno en una forma de informarle de aquello que se desee: el lugar en el que se encuentra el profesor en ese momento; lo que tardará en volver; como contactar con él... etc. Este trabajo de fin de grado implementa una solución mediante tecnología simple y barata, así como de una manera sencilla e intuitiva en uso.

Los ámbitos tecnológicos incluyen la comunicación inalámbrica; del uso y control de un canal Serie aplicado a la comunicación bluetooth; del uso de un controlador gráfico sencillo; de la gestión y programación de un sistema Arduino en una placa ESP32; de la programación en C de un sistema de servicio ”demonio” con gestión de múltiples hilos; de la comunicación y gestión mediante herramientas **UNIX** de dichos hilos; de la creación de memoria compartida entre procesos y subprocesos; de programación en lenguaje bash; de la creación de servicios Java en Android así como de clases Singleton y del desarrollo de aplicaciones móvil con interfaces gráficas; del análisis y diseño software y hardware así como de la ingeniería electrónica aplicada.

Abstract

It is common that, despite technological advances, when a student goes to a teacher's office, the teacher is not there at that moment. Although this is not a significant waste of time, the currently available technology allows us to transform this situation of uncertainty for the student into a way of informing him of whatever we want to: the place where the teacher is at that moment; how long it will take him to return; how to contact him... etc. This final degree project implements a solution using simple and inexpensive technology, and does so in a simple and intuitive way.

The technological areas addressed by this project includes wireless communication; the use and control of a Serial channel applied to Bluetooth communication; the use of a simple graphic controller; the management and programming of an Arduino system on an ESP32 board; the programming in C of a "daemon" service system with multi-thread management; the communication and management through **UNIX** tools of these threads; the creation of shared memory between processes and threads; programming in bash language; the creation of Java services in Android as well as Singleton classes and the development of mobile applications with graphical interfaces; software and hardware analysis and design as well as applied electronic engineering.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

1.1. Introducción

El problema que aborda este trabajo de fin de grado es, en esencia, el de desarrollar, probar y obtener un prototipo funcional de "interactuador inteligente" con el alumno que acude físicamente a la puerta de un despacho o aula, y encuentra que el miembro docente al que había acudido a ver no se encuentra allí. La situación de no poder localizar a una persona es cada vez menos frecuente en la era de informatización en la que vivimos y, sin embargo, esta circunstancia concreta es más que probable, dado que el alumno no tiene acceso a una comunicación inmediata con el profesor si este se encuentra lejos de sus herramientas de trabajo. Esto es debido a que las herramientas de comunicación inmediata y mas directas que un profesor tiene a mano (como su dispositivo móvil) forman principalmente parte de su vida privada y, como es natural, el alumno no tiene acceso a ellas

Es en esta situación concreta en la que el alumno se ha presentado en el despacho del docente que se mantiene el adecuado equilibrio entre infrecuente y cercana, por lo que sí resulta adecuado que se pueda realizar una comunicación directa con el dispositivo personal del profesor (únicamente mediante el dispositivo allí instalado).

Partiendo de esta premisa situacional las circunstancias añadidas pueden ser infinitamente diversas, por lo que la exploración en profundidad de las posibles prestaciones de este prototipo resultan igualmente infinitas.

Es por esto que, a pesar de contar con una planificación de recursos y objetivos previos, el proyecto cuenta con la filosofía no escrita de contemplar y ejecutar mejoras no previstas en la preparación y creación del plan de trabajo. A pesar de ello, se ha realizado un modelo de planificación del proyecto en base a sus necesidades contempladas inicialmente y que otorga un punto de partida y de organización temporal que a la terminación de la creación del prototipo sigue siendo válido como descripción del trabajo realizado y su organización, a pesar de que se cuenten con prestaciones añadidas que serán descritas mas adelante

1.2. Motivación

La motivación principal para la realización de este proyecto es la de aprender conocimientos nuevos o que el alumno ha adquirido escasamente a lo largo del grado.

Algunos de estos conocimientos, por citar unos pocos, son: la instrumentación electrónica; el manejo de canales de comunicación inalámbricos; la correcta programación de un sistema Arduino o similar; profundizar en el uso de las herramientas **UNIX** y la creación de una aplicación móvil en entorno Android

Todos estos campos del conocimiento tecnológico tienen en común que, o bien pertenecen a asignaturas de la carrera que el alumno no ha cursado, o bien considera no haber aprendido lo suficiente sobre ellos y, es por esto, que la principal motivación del proyecto reside no en que resulte fácil, sino en que supone un desafío de cuya dificultad se puede extraer nuevo y refrescante conocimiento.

Otra importante motivación es la de el genuino interés en la finalidad del prototipo en si mismo, ya que se trata de una idea muy interesante y cuyo desarrollo puede llegar a ser extremadamente útil.

Y por supuesto siempre existe la motivación, difícil de realizar, de conseguir una herramienta útil y comerciable que pueda ayudar a los miembros del profesorado en su día a día y de paso impulsar la carrera profesional del alumno desde sus inicios.

1.3. Objetivos y etapas del proyecto

1.3.1. Objetivos

La manera mas inmediata de describir las facetas del problema que se aborda en este proyecto es la de afrontarlo en su complejidad técnica mas tangible y manifiesta, dejando atrás los aspectos (no menos importantes) de su planificación previa.

Podemos enfocar los objetivos del proyecto en su aspecto técnico funcional que requiere de tres componentes principales: La interfaz con la que interactúa el alumno; el servicio intermedio que gestiona la actuación del sistema en su conjunto y la interfaz en manos del profesor para manipular los parámetros y funcionalidades del sistema.

- **Pantalla**:Es un componente técnico que contiene tanto una pantalla al uso para mostrar la información como botones que permiten interactuar sobre ella y pudiendo así enviar información al otro extremo del sistema

- **Servicio *Back***: El componente integrador, es simplemente un programa o servicio en ejecución continua que se comunica con los otros dos componentes, en correcta gestión y funcionamiento

- **Aplicación Android**: El terminal en manos del profesor (en adelante **Kerkaf**) esta compuesto esencialmente por una interfaz que permite seleccionar y enviar información al otro extremo, decidiendo así el contenido en datos o incluso prestaciones que se manifiesta en la Pantalla

Se trata de tres vertientes o entornos de trabajo sobre los cuales se desarrollan soluciones de ingeniería diferentes y en algunos casos formar estructuras de desarrollo y planificación diferentes.

Evidentemente estos tres objetivos descritos se desdoblan a su vez en numerosos problemas y requisitos que pueden ser igualmente extensos. A pesar de ello, todos serán debidamente descritos en la elaboración de este documento, llegando incluso a describir problemas emergentes hipotéticos y sus soluciones también hipotéticas.

Sin embargo en lo que a este apartado introductorio se refiere, el enfoque va a reducirse a la altura de estos tres objetivos concretos.
Por lo tanto para elaborar un plan de desarrollo correcto y elaborar estos tres elementos, debemos dividirlo en varias etapas que permitan afinar la tarea real a realizar para resolver los objetivos

1.3.2. Etapas

Las etapas de desarrollo y resolución de los mencionados objetivos son las siguientes:

Primera Etapa: Pantalla Interactiva

Se debe realizar un diseño de un componente de visualización de datos y posiblemente imágenes con capacidad interactiva, este componente puede extraer su funcionalidad inicial y parcial de un producto comercial programable y se deberá programar para cumplir con los requisitos que se describen en el análisis del prototipo.

Tareas a realizar:

- Estudio de los requisitos técnicos y físicos de la Pantalla
- Estudio de sistemas equivalentes en el mercado
- Diseño de un circuito aledaño para aportarle interactividad
- Programación del sistema diseñado y/o adquirido
- Integración de la parte física montada con el sistema programado adquirido

Segunda Etapa: Aplicación *Back-End*

Se debe crear un servicio en ejecución permanente que se encuentre pendiente de la información que se quiera transmitir, tanto desde la Pantalla como desde **kerkaf**, por lo que debe tratarse de un ”demonioz un programa con sensibilidad a eventos en canales de comunicación.

Tareas a realizar:

- Estudio de los requisitos tanto hardware como en software
- Estudio de sistemas equivalentes total o parcialmente
- Obtención e instalación del componente o componentes *hardware*
- Programación de la aplicación en el sistema elegido

Tercera Etapa: Elaboración de kerkaf

Finalizando el tercer aspecto manifiesto del prototipo, se debe programar una aplicación con características propias de una interfaz de usuario que obtenga la información que se desee cargar en la Pantalla física. *Tareas a realizar:*

- Estudio de los requisitos de la aplicación, haciendo especial hincapié en los derivados de los casos de uso esperados para una interfaz visual móvil
- Estudio de sistemas equivalentes en cuanto a la comunicación elegida y el diseño visual
- Preparación e instalación del entorno de programación y carga del software en el dispositivo móvil
- Programación de la aplicación según el diseño establecido

Cuarta Etapa: Integración y prueba de los elementos

Para terminar y a modo de pulido y *testing* se deberá poner en común todas las tres piezas del prototipo y resolver las posibles incoherencias de comunicación o funcionamiento cohesionado, por lo que se podrá comprobar que el diseño previo de intercomunicación de las piezas opera correctamente

- Estudio de los requisitos físicos de comunicación necesarios para el correcto funcionamiento del prototipo final.
- diseño de los sistemas de comunicación entre las tres partes
- Implementar el protocolo de comunicación entre *back* y la Pantalla.
- Implementar el protocolo de comunicación entre **kerkaf** y el *back*
- Pruebas y comprobación de un correcto funcionamiento desde cualquier extremo del prototipo a los demás

1.4. Recursos utilizados

Para el desarrollo del prototipo con todos sus componentes y etapas se han utilizado los siguientes recursos físicos:

- Ordenador Personal con Licencia de sistema operativo Windows 10
- Raspberry Pi 2B
- Tarjeta de memoria MicroSD
- Fuente de alimentación micro USB Raspberry
- Adaptador ”dongle” USB para conectividad bluetooth
- Chip ESP32 modelo de placa TTGO T5 con Pantalla de tinta electrónica ”lilygo”
- batería de polímero de iones de litio - 3,7V 2000MAH - conector JST-PH
- Estación de soldadura 60W
- Placas de fibra de vidrio de PCB taladradas para circuito impreso
- Kit de componentes electrónicos Elegoo para entorno Arduino comprendiendo:
  - 5 resistencias de 330 ohmnios
  - 2 leds, uno rojo y uno verde
  - 4 botones de pulsación
  - Bus de 12 cables para conectar los pines de la placa ”lilygo”
- Cable de conexión de datos protocolo Micro USB
- Cable de conexión de datos protocolo Ethernet
- Regleta de conexión eléctrica

1.5. Caso de Negocio

1.5.1. Agentes implicados en el proyecto

Los siguientes actores, instituciones y organizaciones están implicados en el proyecto:

- **Cliente**: Universidad o Departamento que incorpore el prototipo a sus salas.
- **Beneficiario**:Tanto la universidad en el rendimiento de su personal como el alumnado a la hora de localizar a su personal docente.
- **Usuarios del proyecto**: El uso de este Sistema está destinado a los siguientes dos usuarios:
  - Profesor: ejecuta los casos de uso de ***Kerkaf***.
  - Alumno: ejecuta los casos de uso de la ***Pantalla***.

1.5.2. Presupuesto

En una primera aproximación al coste estimado de este prototipo y su fabricación se ha de tener en cuenta factores de costes físicos en materiales así como la valoración en horas de trabajo del desarrollador, que en este caso es tanto software como electrónico.

El desglose y detalle de este coste se puede consultar en mayor profundidad en el apartado 2.3 (memoria económica) del capitulo 3.

1.5.3. Impacto Social

La prestaciones que los microchips y sistemas informáticos en miniatura nos prestan hoy en día ponen a nuestra disposición un arsenal tecnológico que, de ser correctamente aprovechado puede mejorar enormemente el funcionamiento de las instituciones y, con ello, la vida de las personas

Estas mejoras pueden verse manifestadas en los siguientes aspectos:

Social

El impacto social que el correcto desarrollo de este proyecto puede tener se mide principalmente en su efecto indirecto en la mejora de la calidad de vida que proporciona a sus usuarios, tanto profesores como alumnos. Ya que, al ahorrar molestias y poder resolver rápidamente malentendidos en la agenda, podemos evitar malgastar nuestro tiempo y con ello evitar estrés

Económico

Asimismo, un correcto aprovechamiento del tiempo, tanto de uno como de otro, permite sacar un rendimiento económico indirecto en tanto que se evitan confusiones, tardanzas o pérdidas de tiempo, lo cual, se acaba traduciendo en una mejor productividad y rendimiento económico.

Tecnológico

Aparte de este aspecto de la calidad de vida del usuario, también cabe destacar el desarrollo en digitalización e *IoT* que se produce en el entorno universitario y que es a lo que apunta el futuro institucional público y que la sociedad nos exige para cumplir con nuestro compromiso al progreso.

1.6. Estructura de la Memoria

La creación de este documento sigue las pautas establecidas por la filosofía de documentación de Ingeniería Software, añadiendo para esta memoria en concreto un apartado de Antecedentes y desarrollando en mayor profundidad de lo habitual el apartado correspondiente a la Implementación. La estructura no obstante es la siguiente:

En el Capítulo 1 se aborda la introducción del proyecto en sí, sirviendo como extensión explicativa del resumen, incorporando la descripción de los objetivos y demas explicaciones previas y puestas en contexto incluyendo este apartado

El Capitulo 2 corresponde a la enumeración y análisis de los sistemas similares anteriores a este y estudiados para el desarrollo del prototipo, junto con las conclusiones y partes útiles de estos.

El Capítulo 3 esta dedicado a la metodología del desarrollo: su planificación, componentes y tecnologías utilizadas, lenguajes de programación entorno de desarrollo... etc En el Capítulo 4 corresponde exclusivamente al Análisis Software tal y como queda descrito en la filosofía de la ingeniería software: Casos de uso, requisitos de diferentes tipos y la explicación en detalle de los casos de uso y sus diagramas de secuencia.

El Capítulo 5 aborda la parte de Diseño Software, con el detalle de que también contiene un apartado del proceso de diseño físico del prototipo con una descripción electrónica de lo diseñado, así como las interfaces gráficas de los componentes ***Pantalla*** y ***Back***.

En el Capítulo 6, el mas elaborado, se describe tal y como sucedió el proceso de creación del prototipo y sus elementos y subsistemas, describiendo componente a componente la realización de este.

A continuación en el Capítulo 7 se describen las pruebas realizadas y los criterios seguidos para realizarlas así como sus resultados.

Finalmente en el capítulo 8 quedan reunidas las conclusiones extraídas de todo el trabajo junto a las posibles lineas futuras de desarrollo del mismo.

Al final del documento se reúnen algunos apéndices con información de interés, Manual y Guía de Uso así como la bibliografía de la memoria.

Capítulo 2

Antecedentes

Evidentemente, este proyecto no inventa la rueda, sino que en realidad inventa una nueva manera de juntar varios diseños de ruedas ya inventados. La idea de proporcionar una interfaz interactiva de acceso remoto es un concepto básico de comunicación que se usa para ordenadores cualesquiera o diferentes servicios informatizados. Uno de los componentes esenciales del diseño de un sistema computacional desde los albores de la informática ha sido el como comunicar la entrada y salida de datos al componente computacional en sí, desde perforar tarjetas de papel hasta el ”Alexa, play some music”.

Este tipo de antecedentes no será necesario que se analicen pues, como bien es sabido, son padres de nuestro tiempo y, con él, nuestros problemas y soluciones. La filosofía de Sistemas en los que más puede verse reflejado este proyecto son aquellos sistemas de entorno **IoT** (*internet of things*), a sistemas baratos y sencillos de informatización de utensilios de uso cotidiano, como es ejemplo antológico este prototipo: un cartel de información variante.

Como ya es habitual en el proyecto hasta este punto, separamos las corrientes de trabajo en tres partes, según el aspecto del prototipo con el que se esté tratando. En este caso no va a ser menos y los sistemas descritos a continuación son las respectivas inspiraciones de cada uno de los tres componentes, así como la solución escogida.

2.1. *Pantalla*

En este componente, al ser núcleo y componente de innovación del sistema desarrollado, se hace especial incapié en la investigación de sistemas similares y antecedentes tecnológicos.

Al tratarse de un sistema interactivo de cara al publico, utilizado sin supervisión de ningún tipo y ante la necesidad tratase de un dispositivo inalámbrico, la búsqueda de sistemas similares como fuente de inspiración obedece a dos principales criterios independientes: similitud funcional y tecnológica.

2.1.1. Funcional

Si observamos la similitud del sistema por la parte funcional, cualquier dispositivo interactivo que se actualice en tiempo real puede ser referente de uso de la ***Pantalla***.
Sin embargo existen numerosos ejemplos cuya funcionalidad específica se asimila enormemente a la que se busca en este proyecto y, por lo tanto sirven de inspiración.

La característica mas preciada en cuanto a funcionalidad es la facilidad de uso: un sistema con capacidad de aprendizaje sobre su utilización totalmente autodidacta o con mínimas indicaciones y cuya

sencillez haga posible un uso completo e inmediato con facilidad y rapidez.

Respecto a esta característica, es habitual ya hoy en día encontrarse numerosos ejemplos de este tipo de uso en estaciones de tren, aeropuertos, bibliotecas, etc... (figura 2.1). En general todos los entornos donde ha sido conveniente una digitalización de estos servicios reflejando, de hecho, la propia motivación que mueve este proyecto.

Resulta interesante observar como, al hacerse cada vez más frecuentes estos sistemas, las instrucciones de su uso resultan cada vez mas implícitas, por lo que la sencillez puede acompañarse cada vez más de auténtica potencia acompañada de la mejora tecnológica correspondiente.

Guiándose por esta observación en particular, resulta evidente cual es el sistema de interacción mas inmediato en comprensión para cualquier usuario hoy en día: la pantalla táctil. Esto concierne tanto a las prestaciones funcionales del prototipo como a las características tecnológicas, por lo que esa segunda parte se abordará en el siguiente punto.
En lo que a la funcionalidad concierne, un sistema de interacción táctil resultaría óptimo:

Figura 2.1: Sendos ejemplos de interfaces táctiles públicas

A pesar de todas estas ventajas, hay un aspecto de este sistema que no lo es: su precio y complejidad, haciéndolo de facto inviable para este proyecto (al menos en un prototipo) por lo que la manera de interactuar con el sistema funcionalmente, debe ser mas barata y sencilla pero mantener la ventaja de la familiaridad inmediata.

Sin necesidad de alejarnos del entorno táctil de uso mas común, el smartphone. Existe en este dispositivo que todo el mundo porta un sistema paralelo al táctil para interactuar con él. De hecho, el único del que dispone aparte del sistema táctil:

Modelo de los 3 botones:

SeleccionarMenú, anterior y siguiente. Tres simples conceptos que, dotados de la flexibilidad suficiente y comportándose de forma versátil según las circunstancias, pueden utilizarse para interactuar con casi cualquier interfaz compleja si el sistema con el que interactúa cuenta con el correcto diseño y muestra una interfaz intuitiva como el ejemplo de la figura 2.2.

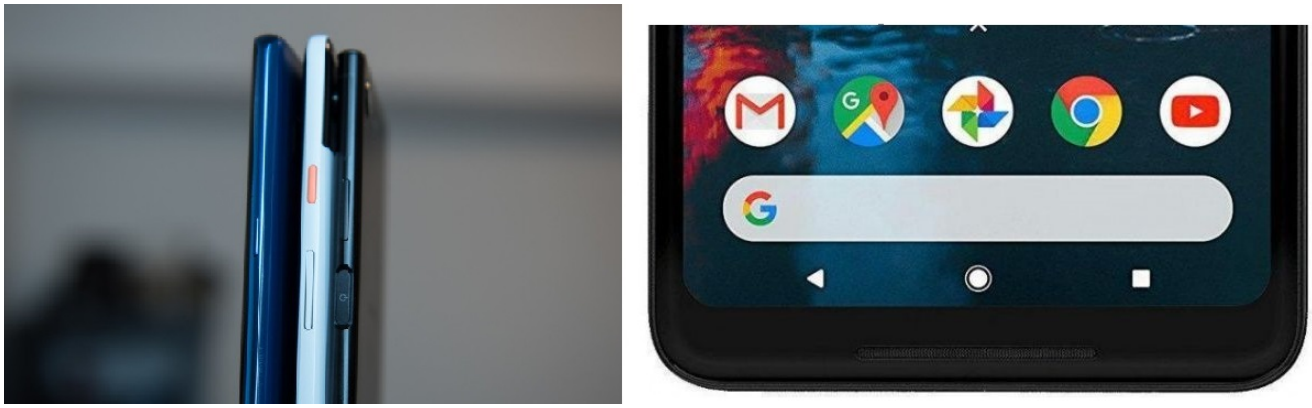

Figura 2.2: Usos del sistema de los 3 botones en smartphone

Finalmente este modelo de interacción resulta óptimo para las necesidades de la ***Pantalla*** y, desde luego, más que suficiente para el planteamiento de un prototipo inicial como el que intenta establecer este proyecto.
Sin embargo, conviene anotar que, dado que se pretende mantener una diferenciación clara de las funciones de cada botón, en el caso de esta ***Pantalla***, el número de botones es de **cuatro** ya que no es necesario que el prototipo de prueba que se va a desarrollar sea totalmente comercial y compacto, pudiendo ser así en lineas futuras de este mismo desarrollo.

2.1.2. Tecnológico

Este criterio obedece a las necesidades técnicas y físicas que imponen las circunstancias de su uso, por lo que se observan sistemas que aunque realicen una función que no sea ni remotamente similar a la de nuestro sistema final, si que comparten aspectos útiles para el prototipo y sus requisitos.

- **Comunicación inalámbrica**: Las tecnologías existentes en el mercado y al alcance de este proyecto son numerosas en diversidad y cantidad. Los requisitos son sencillos: Bajo consumo energético y corto alcance de uso.
  Suponiendo el uso de una comunicación de radio de alta frecuencia, como puede ser una conexión wifi o una red telefónica, las opciones en realidad no son demasiadas:

  - ***IoT* wifi**: El ya clásico protocolo de transmisión de datos inalámbrico, elegido como el más común de los métodos de conexión a internet de cualquier dispositivo lo suficientemente móvil como para que conectarse mediante cable (hoy en día típicamente ethernet). Esta forma de comunicación de datos resulta ideal dado su rango de uso habitual y su capacidad de comunicación. Sin embargo cuenta con inconvenientes muy importantes:

    Para empezar, su uso tan extendido de hoy en día provoca que en espacios reducidos donde se hace uso de una gran cantidad de estas redes, la pérdida de datos por interferencia sea algo muy habitual, teniendo en cuenta que el prototipo aspira a ser utilizado en grandes cantidades en un mismo pasillo, puede resultar un inconveniente. Además de esto, su coste energético puede llegar a ser demasiado alto para lo que se aspira en este desarrollo.
  - **Bluetooth**: El uso de la comunicación bluetooth como método de transmisión de datos con pequeños utensilios, herramientas o periféricos está ya muy extendido, su rango de eficacia es relativamente mas corto dado que se utiliza una potencia menor aunque su rango de frecuencia

es similar, este corto alcance evita que las interferencias sean tan frecuentes como en el caso del wifi.

De entre las posibilidades contempladas, finalmente se decidió utilizar una conexión bluetooth, por el hecho previamente sabido de que las redes wifi privadas emitidas dentro de la universidad son automáticamente contrarrestadas por los repetidores de sus instalaciones, mediante anulación por frecuencia acoplada.

- **Vista Gráfica**:

  - **display LCD**: Un visor *liquid crystal display* trabaja con la polarización de un liquido que reacciona electrónicamente dejando pasar o no la luz, los electrodos que estimulan este líquido son transparente y cuenta con un cristal polarizador de la luz que asegura que únicamente la luz polarizada cruza el display cuando este es transparente, asegurándose así que si el líquido está estimulado, obstruya correctamente la luz polarizada.

    Este tipo de visor gasta una cantidad de energía muy pequeña, por lo que su uso cubriría ese requisito del sistema que se pretende implementar. Sin embargo, su disponibilidad como pantalla pixelada y de información editable es muy limitada en el mercado de dispositivos IoT, y las opciones consultadas no resultan muy convenientes.

  - **display LED**: Esta tecnología gráfica es la mas sencilla de explicar, ya que consiste en un reparto de píxeles RGBW en malla para crear imágenes tal y como se desee. Su disponibilidad en el mercado es muy alta y su flexibilidad de uso resulta conveniente.

    Sin embargo este tipo de pantallas tienen el enorme inconveniente de utilizar una cantidad de energía excesiva para un pequeño dispositivo de bajo consumo, por lo que es descartado.

  - **Tinta electrónica**: Existen numerosos ejemplos de interfaces visuales que utilizan tinta electrónica, los sistemas estudiados además de ser muy fáciles de utilizar (manejo de píxeles en malla) están disponibles en numerosos componentes IoT programables del mercado y el resultado es muy adecuado para el objetivo del prototipo. Además su coste energético es muy reducido, ya que no se consume apenas energía mas que para variar el contenido de la pantalla, al contrario que en el display LED, esta tecnología es de uso muy común como puede verse en la figura 2.3.

Figura 2.3: Muestra de uso de pantallas de tinta electrónica en un comercio

Hay mas información sobre esta tecnología y su funcionamiento en el apartado 3.3 del capitulo siguiente y la figura 3.11.

Tras observar las diferentes tecnologías disponible, y teniendo en cuenta su disponibilidad en los diferentes chips y placas IoT o Arduino del mercado, la opción resultante fue la de la Tinta electrónica, cuyo bajo uso de energía resulta perfecto para los objetivos de este proyecto.

2.2. *Back*

Los sistemas de ”Servidor” o cualquier sistema que proporcione funcionalidad requerida remotamente son infinitamente diversos en el campo de la tecnología informática y digital. Al contrario que en el caso de la pantalla, aquí los aspectos de similitud a observar son puramente funcionales, ya que la tecnología empleada (hardware) resulta irrelevante mientras posea conexión a Internet para que la aplicación móvil pueda conectarse y a su vez posea la tecnología de comunicación inalámbrica necesaria para la Pantalla (lo que en muchos casos es flexible universalmente si posee un puerto USB).

Estando de esta manera los criterios de búsqueda, las opciones resultan prácticamente ilimitadas, pero, al ser este el componente que tiene que gozar de mayor flexibilidad del prototipo, cualquier sistema informático programable podría servir (con las prestaciones mencionadas, claro está).
Es por esto que, independientemente del entorno en el que se encuentre desplegado, los sistemas con similitud a lo que este prototipo pretende conseguir, poseen esta similitud no por sus características físicas, sino por su desempeño software de “Servidor”.

- **servicio REST**: estos tipos de sistemas de comunicación por demanda en protocolo http son tremendamente flexibles y resultan viables como solución en prácticamente todos las situaciones contempladas: cuentan con la capacidad de enviar información estructurada a placer y cuentan con numerosísimos ejemplos de funcionamiento en la inmensa mayoría sistemas y lenguajes de programación existentes.

- **Comandos SSH**: Una manera sencilla y segura de “Invocar” un servicio remotamente es la de ejecutar un comando a través de una conexión ssh, resulta un sistema menos sofisticado y con relativamente menor potencia en prestaciones, pero puede ser una solución con mas sencillez y que sin embargo aporta una capacidad de seguridad firme con poca necesidad de configuración al respecto

A pesar de que como se verá en el capítulo de conclusiones, la tecnología REST es extremadamente tentadora, añade un componente de complejidad no asumible para este modesto proyecto, y la forma de ejecución remota de servicio elegida es finalmente la de comandos SSH, dado que se trata de un entorno **UNIX** (raspbian) y las facilidades de su uso son numerosas.

2.3. *Kerkaf*

Respecto a este sistema, parte tercera del prototipo en su conjunto, puede inspirarse en sistemas informáticos de uso cotidiano con propósitos parecidos. Al igual que con el ***Back*** los aspectos similares de los sistemas que se han usado como inspiración solo lo son en funcionalidad y además, al ser la parte más flexible de las tres, no requiere una especial dedicación en cuanto a sistemas precedentes específicos, por lo que las opciones en las que se basa son pocas.

- **Aplicación Front sirviendo una página web**: Un modelo de aplicación muy flexible y que atiende a la posibilidad de realizar la comunicación con el ***Back*** en protocolo HTTP. Esta forma de creación de programas es la más común a dia de hoy y permite su acceso de manera muy flexible a través de un navegador cualquiera.

- **Aplicación Móvil**: Otra opción que también otorga gran flexibilidad es la de realizar una interfaz de uso en una aplicación móvil, la cual a su vez también puede funcionar con protocolo REST y comunicación HTTP, además cuenta con la ventaja de que se lleva encima en todo momento, por lo que resulta especialmente conveniente para los objetivos propuestos

A pesar de la sencillez y tentación que implica utilizar una página web como interfaz de funcionamiento del prototipo, la necesidad de uso ”durante el café” implica que pueda portarse con uno mismo al salir del despacho o sala y, por lo tanto, es la aplicación móvil la que finalmente se utiliza.

Capítulo 3

Metodología

3.1. Procedimiento y Desarrollo

A la hora de desarrollar este proyecto de fin de grado se ha tenido en cuenta Principalmente la necesidad exploratoria de la tecnología a utilizar, es por eso que se ha creído conveniente utilizar una metodología de desarrollo basada en una planificación previa de los objetivos y su realización, pero con la capacidad de reorientar los recursos y tiempos a lo largo del mismo, por lo que a pesar de que los requisitos en términos generales e incluso en aspectos bien matizados han podido verse ampliados tanto en profundidad como en extensión o ramificarse en nuevas necesidades derivadas.

Dadas estas previsibles circunstancias, se ha utilizado una combinación de los modelos dirigidos por plan de cascada y en espiral, permitiendo así una modificación dinámica de los requisitos y necesidades del cliente o proyecto y, además, mantenemos la estructura de etapas que permiten a su finalización, el inicio de la siguiente condicionando sus necesidades ya mejor determinadas.

Este método de trabajo, se ha seguido de manera orientativa, ya que debido a circunstancias académicas y profesionales de su autor, hay numerosos espacios temporales vacíos de actividad y desarrollo, por lo que en la explicación de la planificación del proyecto se explicará teniéndolo en cuenta. Asimismo, la flexibilidad que nos otorga tanto la metodología elegida como el propio plan de proyecto, nos permite actuar con facilidad a la hora de corregir malas interpretaciones de las necesidades del prototipo y con ello también añadir o retirar lo que mas tarde se ha visto conveniente.

Aunque como se verá en los repositorios git se ha seguido un desarrollo por iteraciones, finalmente no ha servido para reflejar correctamente el proceso de *feedback* sobre el prototipo que ha tenido el ”cliente” ya que el mayor aporte de correcciones y cambios ha sido a la terminación de las etapas descritas en la planificación, es decir: al completar cada uno de los tres componentes y a la hora de su integración en un solo prototipo unificado.

Así mismo cabe mencionar que si bien se han realizado pruebas posteriormente a cada realización de requisitos y prestaciones, las principales y mas completas pruebas de caja blanca son las realizadas al final de la creación de cada componente, coincidiendo así con el final de su etapa y el inicio del siguiente hito del modelo en cascada

3.2. Planificación del Proyecto

3.2.1. Diseño de Tareas

A partir de las etapas descritas en la introducción, que son inferidas de los objetivos propuestos, se debe elaborar y concretar un conjunto de Tareas o Fases que enruten adecuadamente el desarrollo del sistema que se pretende obtener.

Para ello, partiendo de las cuatro etapas iniciales (principales), se han definido las siguientes tareas:

- **Preparación**: La fase inicial del proyecto comprende toda la tarea de documentación y preparación tecnológica y de conocimientos que este requiere, se puede observar su despliegue en subtareas en la figura 3.1

- **Desarrollo y Ajuste de la *Pantalla***: Mediante el correcto aprendizaje realizado en la tarea inicial y tras haber fijado adecuadamente los requisitos y casos de uso necesarios del prototipo, se desarrolla y programa el funcionamiento del componente de la ***Pantalla*** para cumplirlos.

- **Desarrollo y Ajuste del *Back***: Este desarrollo resultó ser el mas crítico, ya que en aspectos técnicos de conocimiento **UNIX** resulta ser el mas exigente, contando con solo 12 días en la planificación es un elemento con riesgo de bloquear el proyecto si algo falla.

- **Desarrollo y Ajuste de *Kerkaf***: Siendo esta Fase la de mayor flexibilidad, compensa en cierta manera la presión que se ejerce sobre la anterior, ya que los recursos temporales que requiera un incumplimiento de la planificación pueden extraerse de esta Tarea.

- **Integración**: Se trata de la tarea de mayor duración, la corrección final del prototipo en su conjunto al ajustar los tres sistemas entre sí, se debe prevenir que algo falle, por lo que se otorga mucho tiempo a esta fase.

- **Documentación final**: Finalmente la fase correspondiente a la elaboración de la memoria y documentación de este proyecto a partir de las anotaciones y documentos elaborados durante la misma

A todas las fases de desarrollo y ajuste les acompaña una *deadline* de entrega que permite también ajustar los últimos cambios una vez el cliente observa el resultado del componente. Dadas estas fechas de entrega, se elabora el plan de proyecto que se refleja en la figura 3.1 y que se puede observar gráficamente en la figura 3.2.

●	Tarea de preparación y plan de proyecto	T	01/08/2021	08/08/2021	5 days	100
●	Tarea de Documentación y Aprendizaje	T	08/08/2021	15/08/2021	5 days	100
●	Tarea de Diseño y Análisis	T	15/08/2021	22/08/2021	5 days	100
○	Fase de desarrollo de la Pantalla + Corrección y ajuste	T	22/08/2021	08/09/2021	13 days	100
○	Entrega de la Pantalla	M	09/09/2021	09/09/2021	-	%
○	Fase de desarrollo del Back + Corrección y ajuste	T	09/09/2021	26/09/2021	12 days	%
○	Entrega del Back	M	27/09/2021	27/09/2021	-	%
○	Fase de desarrollo de Kerkaf + Corrección y ajuste	T	28/09/2021	14/10/2021	13 days	%
○	Entrega de Kerkaf	M	14/10/2021	14/10/2021	-	%
○	Tarea de integración de todo el prototipo	T	15/10/2021	10/11/2021	19 days	%
○	Fase de Documentación del sistema y puesta a punto	T	11/11/2021	04/12/2021	17 days	%
○	Primera entrega de la memoria	M	05/12/2021	05/12/2021	-	%
○	Entrega Final TFG	M	13/12/2021	13/12/2021	-	%

Figura 3.1: Planificación del proyecto por fases y tareas

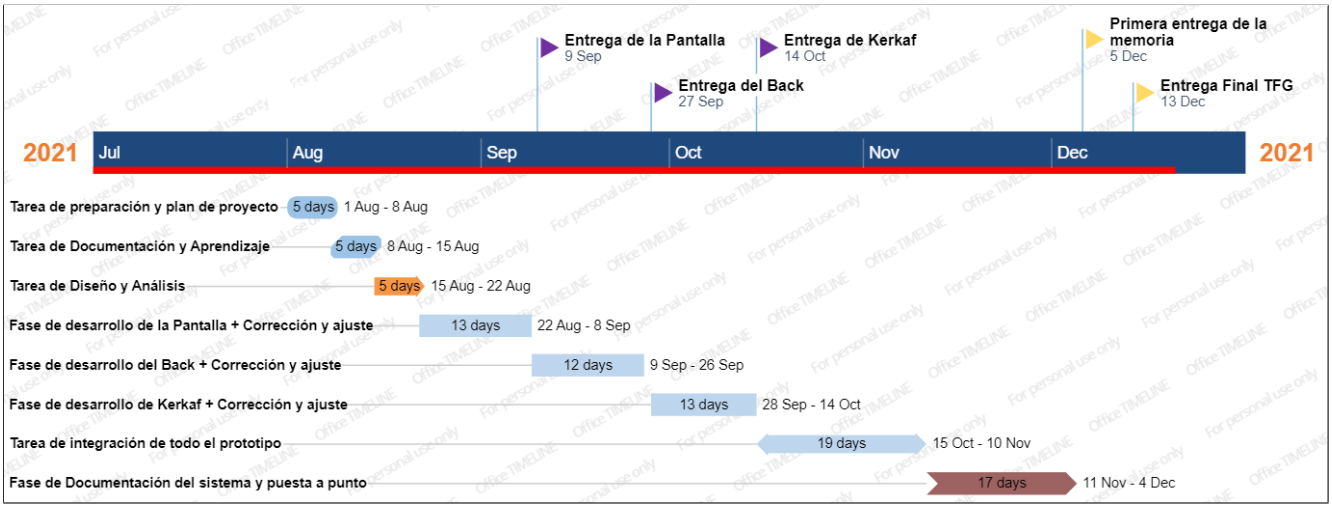

Figura 3.2: Diagrama de Gantt del plan de proyecto

Estas planificaciones, si bien son correctas para un programador e ingeniero experimentado dedicado únicamente a este proyecto, no lo han sido para el alumno con necesidad de aprender según trabajaba tanto en este proyecto como en una consultoría IT, por lo que el desarrollo real del proyecto está desajustado respecto a lo previsto fase por fase, dejando algunas de ellas en pausa durante períodos de tiempo donde no era posible continuar por motivos laborales o personales. Debe tenerse en cuenta al observar el capitulo de implementación, que describe el proceso de desarrollo tal y como sucedió.

3.2.2. Asignación de Recursos

Los recursos utilizados se dividen dependiendo de la Fase a la que correspondan. Todas las tareas correspondientes a la fase de preparación tienen un componente de coste únicamente en el gasto de tiempo del (aun no) Ingeniero Software, por lo que no se reflejan en la siguiente lista.

Fase 1: Desarrollo de la *Pantalla*

- Recursos materiales:
  - Ordenador de Sobremesa
  - TTGO ESP32 V2.3 lilygo
  - Cable USB de transmisión de datos
  - Cables dupont
  - Placa protoboard
  - Componente electrónicos simples (LED, botones, cables...)
- Recursos software:
  - Arduino IDE
  - SourceTree
  - Windows 10
  - Image2Lcd
  - Editor de imagenes bitmap
  - Adafruit librería gráfica
  - GxGD librería gráfica
- Recursos laborales:
  - Programador Arduino
  - Técnico electrónico

Fase 2: Desarrollo del *Back*

- Recursos materiales:
  - Ordenador de Sobremesa
  - Raspberry Pi 2B
  - Fuente de alimentación
  - Router con *port-forward* abierto para conexión SSH
  - Cable HDMI para la configuración inicial de la Raspberry
  - Dongle bluetooth USB
- Recursos software:
  - Visual Studio Code
  - Rasberry Pi Imager
  - SourceTree
  - Putty

- bluetoothctl

- Recursos laborales:
  - Programador c++

Fase 3: Desarrollo de *Kerkaf*

- Recursos materiales:
  - Ordenador de sobremesa
  - Móvil de testing
  - Cable USB C de conexión de datos
- Recursos software:
  - Android Studio
  - SourceTree
  - Editor de imágenes para las vistas
  - clave de firmado de las *apk* Android
- Recursos laborales:
  - Programador Android-Java

Fase 4: Integración

En esta fase, dado que esta preparada para modificar y adaptar cualquier aspecto de los elementos tratados en las fases anteriores, comprende la suma de todos los recursos anteriores, incluyendo software y personal.

3.2.3. Memoria económica

Costes de licencias Software

Descripción	Precio(€)
Visual Studio Code	0€
Arduino IDE	0€
Putty	0€
Android Studio	0€
Image2lcd	0€
Bibliotecas Arudino	0€
Notepad++	0€
SourceTree	0€
Total:	0€

Cuadro 3.1: Coste Software

Coste de componentes Físicos

Descripción	Precio(€)
Raspberry Pi 2B	32,65€
TTGO T5 V2.3 ESP32	17,40€
batería de polímero de iones	8,05€
Componentes Electrónicos	2,99€
Piezas de Cable DuPont	1,99€
dongle usb bluetooth	6,45€
placa fibra de vidrio	0,75€
conector usb	2,5€
envíos	12,5€
Total:	85,28€

Cuadro 3.2: Coste Hardware

Hay que tener en cuenta que la Raspberry puede manejar varias unidades de ***Pantalla*** al mismo tiempo, por lo que el coste de numerosos productos se vería reducido.

Costes indirectos

Estos costes relativamente relevantes se tienen en cuenta únicamente durante el tiempo en el que se ha estado realizando el proyecto

Descripción	Precio(€)
Amortización del Ordenador de sobremesa	25€/mes
Amortización de la estación de soldadura	5€/mes
Electricidad	12,45€/mes
Internet	15,99€/mes
Total(4 meses):	233,76€

Cuadro 3.3: Costes Indirectos

Costes Laborales

Los costes referentes a la mano de obra, que en el caso de este proyecto requiere de Ingeniero Software y un técnico electrónico.

Descripción	Precio(€)
Sueldo Ingeniero Software	3714,28€
Sueldo Técnico eléctrico	164,16€
Total:	3878,44€

Cuadro 3.4: Coste Laboral

Estas estimaciones se han realizando teniendo en cuenta que un Ingeniero Informático gana de media alrededor de 26000€ al año (evidentemente la dedicación de este proyecto no se mide equitativamente

con un trabajo de jornada completa) y que un técnico electrónico gana 20000€ al año o 10,26€ la hora, el coste estimado de este último se mide teniendo en cuenta que se ha hecho uso de sus servicios mucho menos.

Costes Totales

Descripción	Precio(€)
Costes Software	0€
Costes Físicos	85,28€
Costes Indirectos	233,76€
Costes Laborales	3878,44€
Total:	351,43€

Cuadro 3.5: Coste Total

3.3. Tecnologías empleadas

3.3.1. Lenguajes de programación

- **C++:**

  El lenguaje C y su ampliación C++ deben su origen a *Dennis Ritchie* que durante su estancia en los Laboratorios *Bell* dio forma y remate a esta continuación de su anterior lenguaje B.

  Con un minucioso control de las acciones del procesador y la memoria, el lenguaje C es el preferido a utilizar a la hora de programar sistemas operativos o las herramientas mas precisas de estos, y ha sido el elegido para múltiples softwares que requieren alta eficiencia y control.

  Este ha sido el lenguaje elegido para el ***Back*** al poseer la mejor sintonía con las herramientas y servicios del sistema **UNIX** en el cual está basado el Debian del que hace uso el Sistema operativo de la Raspberry Pi, por lo que a pesar de su complejidad, otorga las herramientas necesarias para el objetivo a cumplir.

  Asimismo el lenguaje que se interpreta en los programas de *Arduino* es prácticamente en su totalidad inspirado en este lenguaje C, por lo que también merece la pena mencionarlo a este respecto.

- **java:**

  El ya clásico lenguaje de programación diseñado por *James Gosling* cuya capacidad y facilidad de tipado y orientación a objetos lo convierte junto con el lenguaje *kotlin* en el lenguaje a utilizar para las aplicaciones de entorno Android, por lo que se ha elegido para crear ***Kerkaf*** en este proyecto.

- **Shell y herramientas UNIX:**

  El Shell de **UNIX**, como intérprete de comandos de prácticamente cualquier sistema Linux y por lo tanto como entorno de programación, ha permitido realizar muchas de las operaciones necesitadas en el ***Back*** el cual se aloja en un sistema operativo **UNIX**.

  Aunque el uso de las herramientas **UNIX** se sobrentiende, en el caso de este proyecto merecen una mención especial debido a que gran parte de las funcionalidades del ***Back*** no corresponden a la

programación desarrollada en este proyecto, si no más bien a la de innumerables desarrolladores de *open Source* que a lo largo de décadas han hecho posible muchas de las funcionalidades que se desempeñan el ***Back***

3.3.2. Herramientas y Tecnología Software

- **Arduino IDE**

  El *Arduino Integrated Development Environment* o entorno integrado de desarrollo de Arduino es una herramienta desarrollada en lenguaje Java que, en el caso de este proyecto, ha sido utilizada tanto para editar los programas que se ejecutan en la ***Pantalla*** tanto como el medio de introducirlo en esta. En la Figura 3.3 puede apreciarse el aspecto sencillo de la interfaz.

  Esta gran facilidad ha sido posible dado que este entorno de desarrollo de *sketches* para placas y sistemas Arduino también ha sido totalmente válido para una tarjeta TTGO con chip ESP32, que nada tiene que ver con las invenciones del originalmente Italiano proyecto de software y hardware Abierto.

  Son mas bien los fabricantes chinos de este componente los que han adaptado su diseño a los estándares tanto de Arduino como Adafruit permitiendo así a este alumno desarrollar tranquilamente sobre él como si de un Arduino cualquiera se tratase, con la evidente ventaja que esto aporta al proyecto, ya que la consulta de ejemplos y problemas similares a los planteados aquí es muchísimo mas sencilla cuanto mas estandarizado es el entorno, como Arduino IDE.

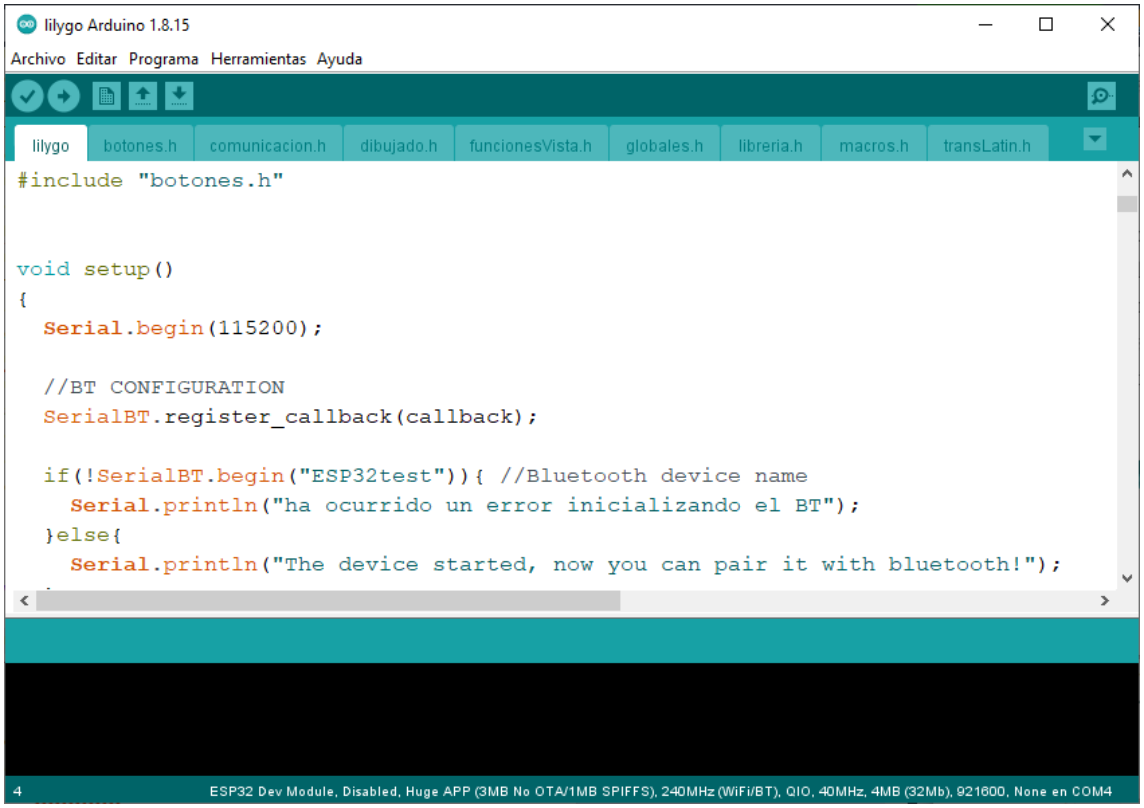

Figura 3.3: Vista de la Interfaz de Desarrollo de Arduino, donde se aprecia su simpleza

- **Visual Studio Code**

  Siendo uno de los editores que mayor comodidad aporta al programador moderno y teniendo capacidad funcional como *framework* de cualquier tipo del cual tenga extensión, Visual Studio Code ha sido la elección de muchos como su entorno de creación y desarrollo y también la de este proyecto.

  Cabe mencionar alguna que otra extensión utilizada, aunque las mas relevantes son las que atañen al seguimiento de los repositorios git que en todos los elementos del prototipo se han utilizado, así como numerosos auto-detectores de código y herramientas para facilitar una rápida escritura del mismo.

  Aunque esta herramienta se ha utilizado principalmente para el desarrollo del programa en C++, también se ha usado como forma de edición rápida de archivos de código y texto de otros elementos del proyecto por su gran comodidad. Puede observarse su distribución en la figura 3.4

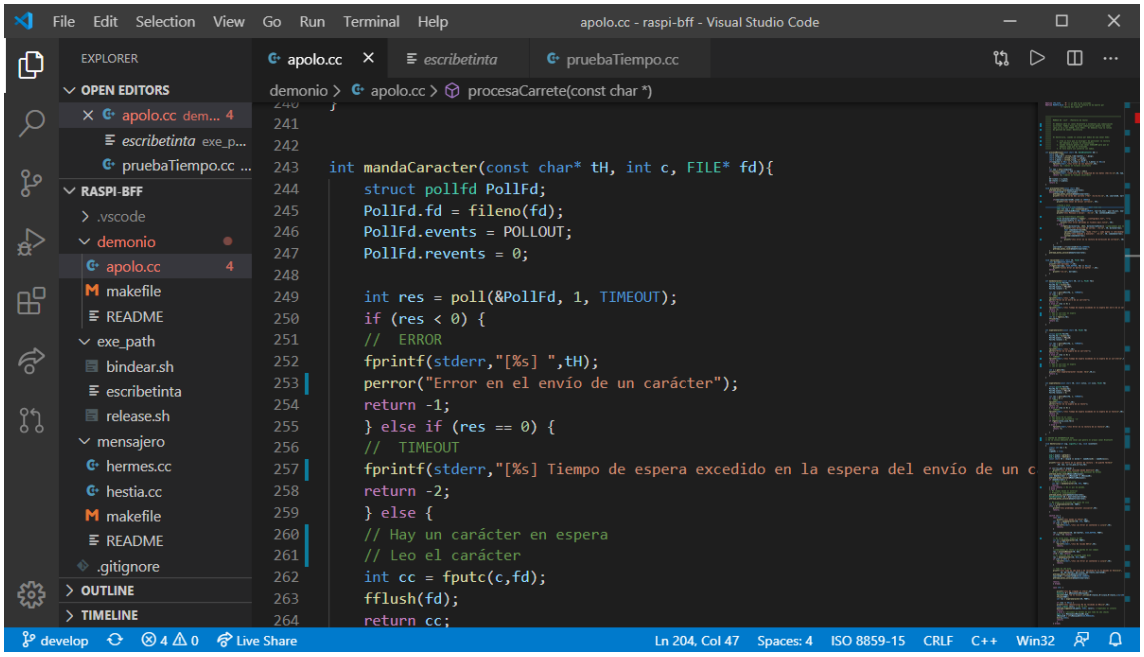

Figura 3.4: Vista del editor Visual Studio Code

- **Adafruit GFX y Librerías derivadas**

  Siendo una de las herramientas menos convencionales de todo el proyecto, Las librerías de gestión gráfica de las ***Pantalla*** han sido de enorme utilidad, ya que el entorno Arduino y la programación directa de una placa de este tipo y su chip ESP32 puede ser profundamente tediosa si se realiza desde cero y sin ayuda de ningún controlador previamente añadido.

  En este apartado cabe mencionar también la librería GxGDE y numerosas librerías de dominio y elaboración públicas sobre cuyo funcionamiento y uso se desarrollará en mayor profundidad en el capitulo de **Implementación** ya que aunque se ha creído conveniente mencionarlas aquí como herramienta utilizada, no han estado exentas de modificaciones por parte del alumno durante su integración en el proyecto y por lo tanto, la otra mitad de su historia no corresponde en este punto del documento.

- **Android Studio**

  Android Studio es un entorno de desarrollo integrado también, al igual que el mencionado antes (Arduino IDE) salvo que en este caso se trata de una herramienta con muchísima mayor potencia, aunque su diseño puede ser algo recargado figura 3.5

  Esta herramienta, que sustituyó a Eclipse como el IDE oficial de desarrollo de aplicaciones Android tiene casi tantas prestaciones en cuanto a comodidad como programa de edición como Visual Studio Code, y añadidamente un entorno de desarrollo gráfico de las *Activities* y demás elementos propios de una aplicación Android que lo convierten en el entorno para programar Android por antonomasia.

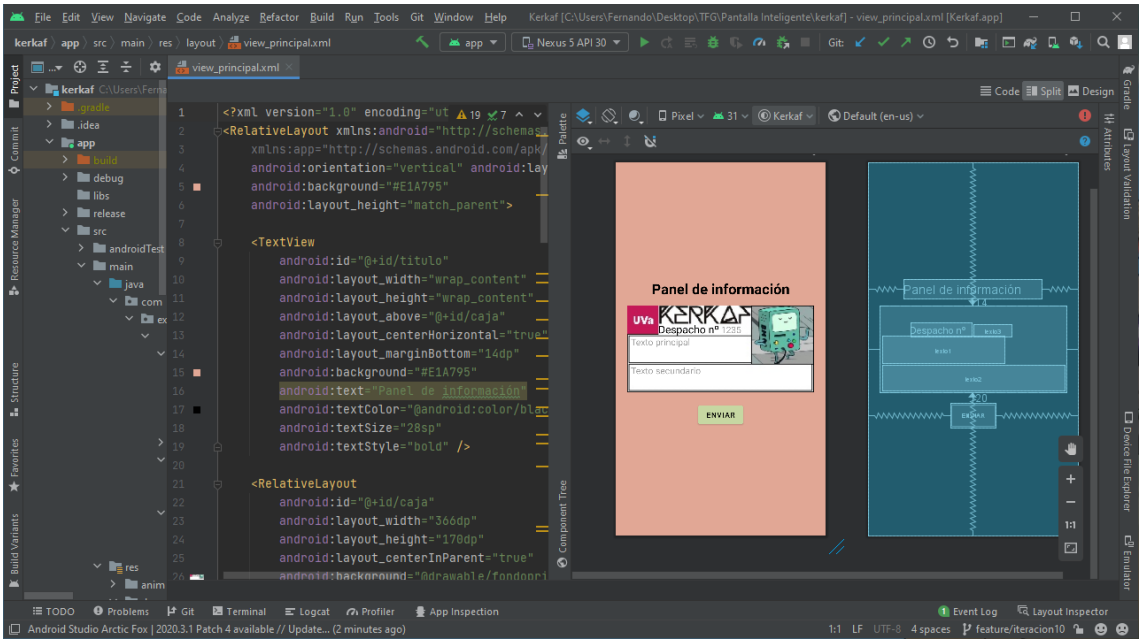

Figura 3.5: Vista del entorno de desarrollo Android Studio

- **metodología Gitflow y SourceTree**

  Los repositorios *git* como forma de almacenamiento de versiones (principalmente de texto y por ende código) son una herramienta indispensable a dia de hoy para el correcto seguimiento de las modificaciones de código y proyectos de envergadura en el que puedan participar decenas de personas sobre un mismo programa.

  Sobre este entorno de repositorios git, es habitual seguir un ”protocolo” de funcionamiento en cuanto a la gestión de sus ramas, llamado *Gitflow* el cual ya es prácticamente un estándar a la hora de utilizar repositorio git y ha sido el utilizado en este proyecto para mantener un correcto seguimiento de los cambios realizados y ahorrar tiempo a la hora de recuperar versiones anteriores o corregir errores de desarrollo.

  Aunque es posible utilizar git mediante comandos de terminal con relativa facilidad, la meta de este sistema de ahorrar tiempo al programador bien merece el uso de un entorno gráfico que otorgue facilidades de uso y potencia, y es para este fin que se ha utilizado la herramienta de uso gratuito *SourceTree* para el sistema operativo *Windows* 10 en el cual se ha desarrollado este proyecto (figura 3.6)

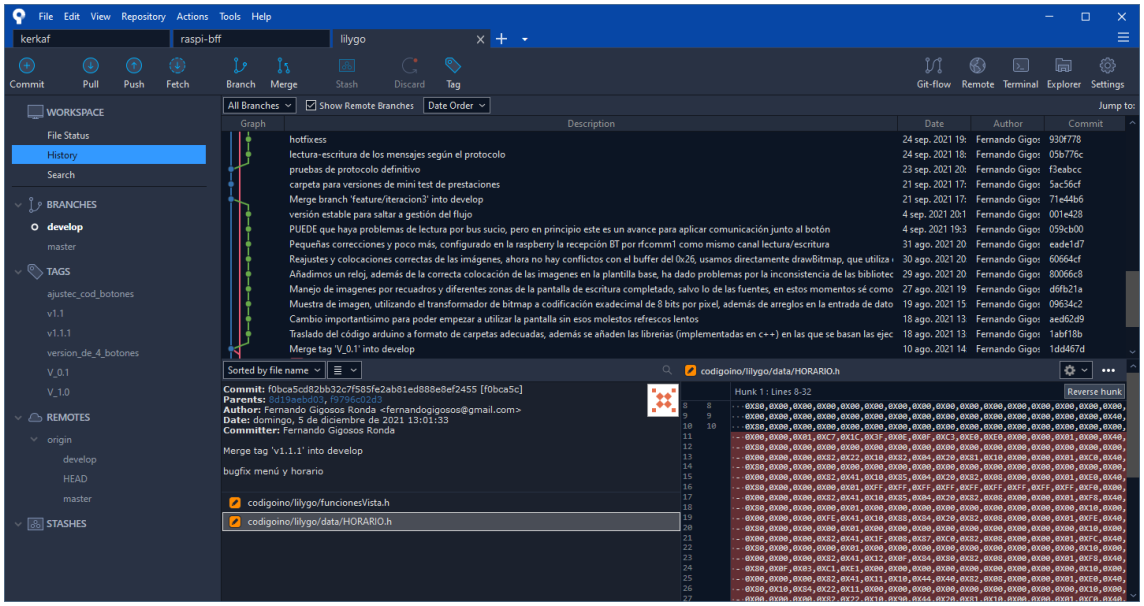

Figura 3.6: Vista del gestor git SourceTree

- **VNC y PuTTY(SSH)**

  VNC o *Virtual Network Computing* es un programa *open source* que permite una conexión cliente-servidor para controlar un ordenador remotamente, tiene la particularidad al contrario de SSH de que permite una interfaz gráfica completa del escritorio remoto.

  SSH o *Secure Shell* es un protocolo de comunicación mas que un programa como lo es el VNC que permite también una comunicación de datos cifrados.
  Su uso esta tremendamente extendido y es el protocolo de acceso predilecto de las Terminales a la mayoría de servidores.

  Estas Dos herramientas de conexión remota encriptada han sido las utilizadas para comunicarse de una manera u otra con la Raspberry Pi, permitiendo así la carga de datos, la configuración del sistema y, finalmente en el caso de SSH, el canal a través ***Kerkaf*** y el ***Back*** se comunican, lanzando los comandos que este último debe ejecutar a petición de la aplicación Android. Puede observarse la vista de inicio de ambas aplicaciones en la figura 3.7

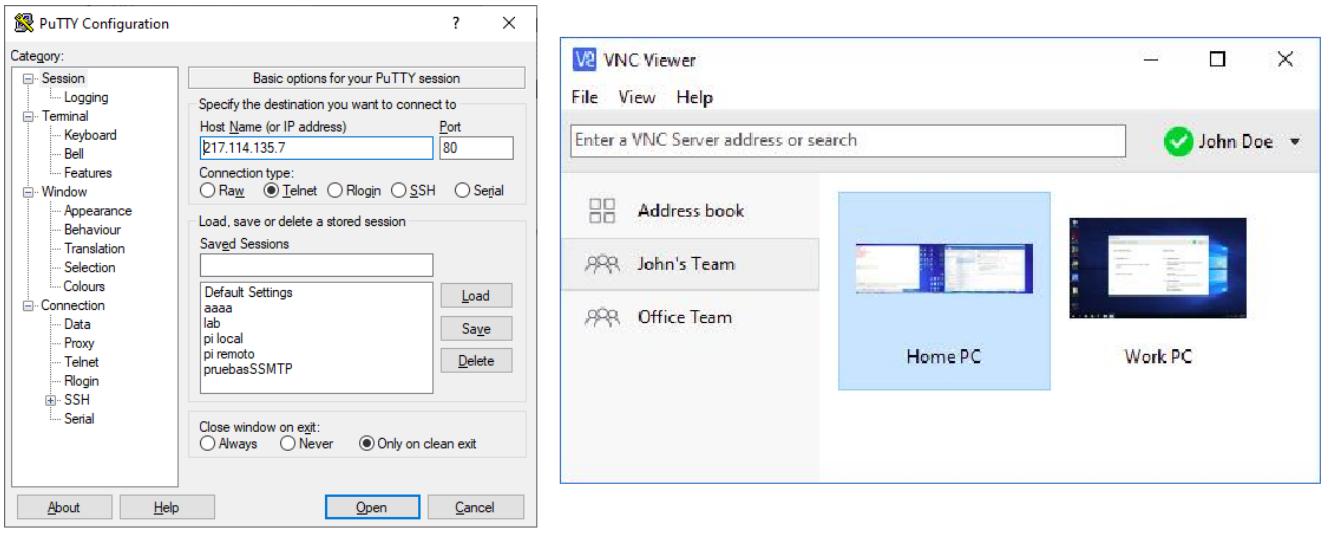

Figura 3.7: Vistas principales de VNC y PUTTY

- **msmtp**

  msmtp es un cliente de *Simple Mail Transfer Protocol* el cual permite envíar correo electrónico o solicitar envíar uno desde un servidor que gestione este servicio.
  Con filosofía *open source* este software se distribuye en la mayoría de plataformas **UNIX**, LINUX e incluso Android y ha permitido a este proyecto gestionar el envío de un correo electrónico mediante la API habilitada a este servicio del servidor de gmail, con un usuario creado a tal propósito.

- **Image2Lcd**

  Esta pequeña herramienta de *open source* (figura 3.8) fue creada para solventar un problema común a aquellos usuarios que buscan programar una pantalla gestionada por un sistema Arduino o similar: El gestor gráfico de adafruit manipula y carga las imagenes en la pantalla desde un formato de datos prácticamente en bruto: Un array de datos hexadecimales que determinan el color (blanco o negro en este caso) de 4 píxeles por cada número cargado en el array, para encajar en el espacio de un byte, el array almacena de facto bytes que agurpan cada uno dos datos hexadecimales describiendo una fila de 8 píxeles de longitud.

  Obtener de una imagen una cadena de datos semejante para cargarla en el código de un *sketch* no resulta sencillo, por lo que esta herramienta facilito enormemente la labor y rápidez de testing de diferentes imágenes.

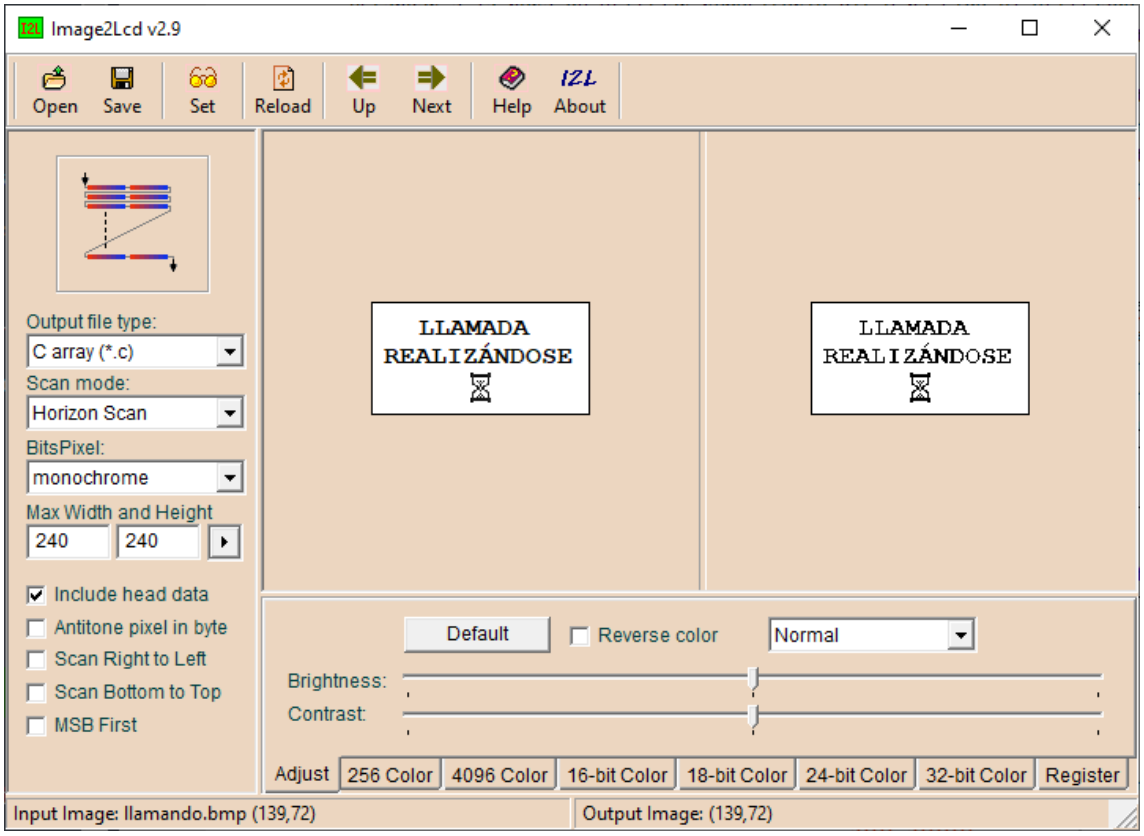

Figura 3.8: Vista principal de uso de la aplicación image2lcd. [7]

3.3.3. Herramientas y Tecnología Hardware

- **Protoboard**

  Denominada también ”Placa de pruebas” o ”Placa de inserción”, la Protoboard es un entorno de creación de circuitos donde resulta fácil probar y modificar los circuitos físicos que el proyecto requiera (ejemplo en la figura 3.9).

  A este componente se le suman por añadidura todas las resistencias, LeDs y botones que se han utilizado sobre esta herramienta

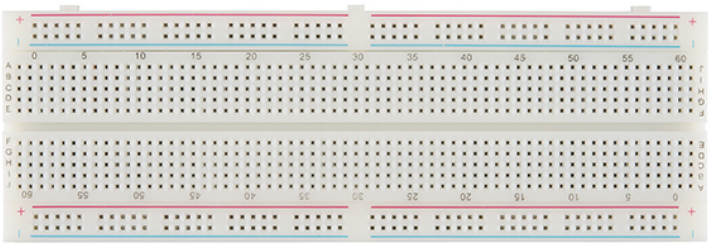

Figura 3.9: Protoboard estándar como la utilizada.

- **Estación de Soldadura**

  Una estación de soldadura es, sin duda, la herramienta básica de todo ingeniero electrónico. Se utiliza para soldar, es decir, crear conexiones eléctricas metálicas entre dos elementos.

  En la creación del prototipo y concretamente en el caso de la ***Pantalla***, Se ha utilizado para crear el circuito de interacción del usuario con los pines de la placa TTGO, ya que tras realizar el diseño del circuito en la Protoboard se pasó el diseño a un entorno físico soldado mediante esta estación.

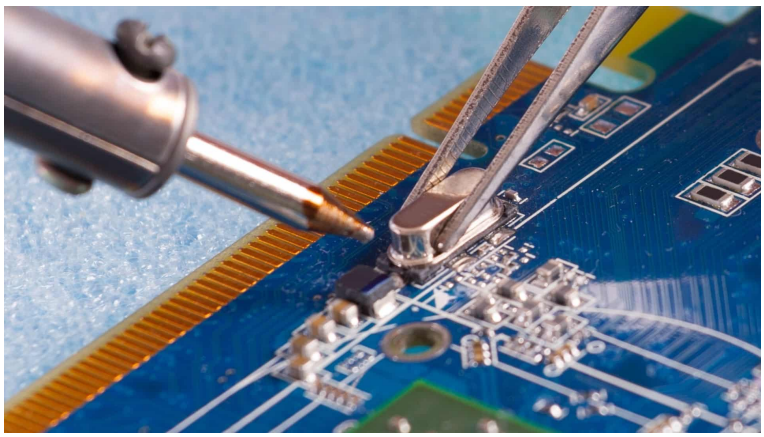

Figura 3.10: Escena de soldadura de componentes con estaño en una placa

Para las soldaduras que se han realizado se utilizó una estación de 60W de potencia la cual aporta una temperatura suficiente para realizar las soldaduras sin riesgo de hacer ”soldaduras frías”.

- **Pantalla de Tinta Electrónica**

  En el ámbito tecnológico de las mini-pantallas o pantallas de bajo consumo, la Pantalla o tecnología de Tinta electrónica resulta a día de hoy la principal apuesta en el mercado de libros electrónicos y lectura en dispositivos electrónicos para evitar la vista cansada.

  Su funcionamiento es conceptualmente muy sencillo: las pantallas están formadas por pequeñas cápsulas de un semifluido opaco (figura 3.11)las cuales se extienden en una lámina fina de una sola esfera de grosor y un área correspondiente a la resolución que se le dé a la pantalla en cuestión. Esta capa de esferas está a su vez recubierta por dos láminas de protección una de las cuales, en contacto con el circuito electrónico, sitúa debajo de cada una de las cápsulas un componente que desprende un campo eléctrico manipulable digitalmente.

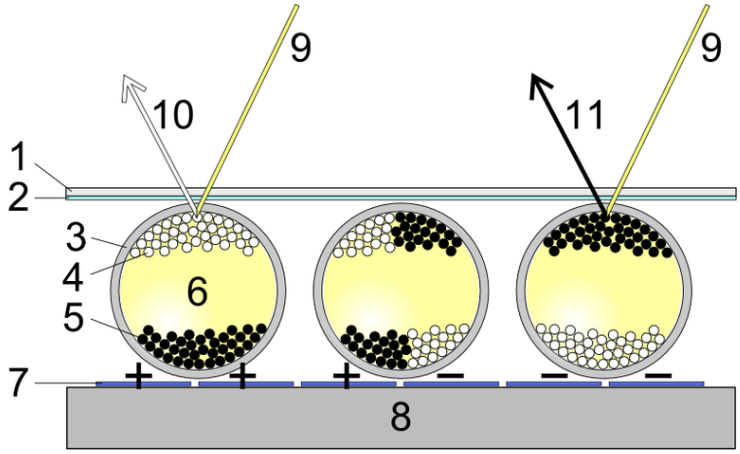

Figura 3.11: Diagrama de representación de la tecnología de tinta electrónica [1]

Leyenda	Descripción
1	Capa superior
2	Capa de electrodo transparente
3	Micro-cápsulas transparentes
4	Pigmentos blancos de carga positiva
5	Pigmentos negros de carga negativa
6	Aceite transparente
7	Capa de electrodos de los pixeles
8	Capa inferior de soporte
9	Luz
10	Blanco
11	Negro

Cuadro 3.6: Leyenda de componentes del diagrama

Estos campos eléctricos determinan si la cápsula que tienen encima es de un color u otro (la mayoría de estas pantallas son monocromáticas), permitiendo así mostrar una imagen con una frecuencia de refresco relativamente baja, pero en muchos casos suficiente para el ojo humano.

- **Conectividad Bluetooth**

  La tecnología Bluetooth, tan comúnmente usada hoy en día como protocolo de comunicación inalámbrica de datos pequeños y con facilidad.

  Esta tecnología, que no deja de ser una comunicación por radio frecuencias, se ubica en la banda de los 2,4Ghz y puede transmitir datos a una velocidad de 50 Mbits por segundo en su versión mas moderna y lo hace con un consumo energético relativamente bajo, por lo que su alcance de uso mas común es de unos pocos metros, otorgandole así una de sus mayores ventajas:
  Puede ser utilizado como método de comunicación por numerosos dispositivos en un mismo lugar o relativamente cerca en el contexto de la actividad humana, siendo así una forma de conectividad efectiva en el mundo saturado de dispositivos IoT de la actualidad.

- **Protocolo de comunicación Serie**

  La comunicación Serie, mas que a una tecnología o herramienta, alude a un concepto o filosofía de comunicación. Este concepto es el de enviar datos de manera ”secuencial”, es decir, uno detrás de otro, enviando un bit de información por cada ciclo de frecuencia.

  Este sistema tiene la desventaja de proporcionar una velocidad de comunicación de datos inferior a la de una conexión en paralelo (múltiples bits por cada ciclo) pero en algunos casos resulta preferible, ya que permite una frecuencia mayor de transmisión de datos para compensarlo y a cambio permite un control y seguridad de los datos mucho mayor, evitando interferencias o desincronizaciones.

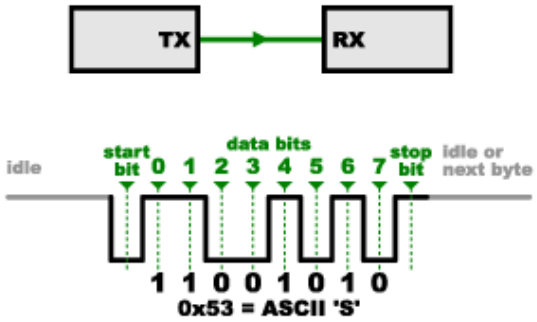

Figura 3.12: Diagrama de representación de una transmisión de datos en serie

Es sobre los terminales escuchando y emitiendo a ambos extremos de una conexión de este tipo que trata este apartado del documento, ya que existen numerosos protocolos de control del flujo de datos en la actualidad que permiten que no se pierda ninguna información durante el envío o recepción de estos, ya sea en un canal de este tipo u otro diferente.

Estos protocolos suelen ser especialmente necesarios en los casos en los que se utiliza una tecnología inalámbrica que pueda perder datos con mayor facilidad como es el caso de la conectividad Bluetooth, por lo que mas adelante se desarrollará al respecto del utilizado en el prototipo en su conexión entre el ***Back*** y la ***Pantalla*** asegurando así la correcta transmisión sin pérdidas de datos.

Capítulo 4

Análisis

Esta fase del desarrollo o iniciación del proyecto implica la obtención de información sobre las necesidades y objetivos de este, detallando de forma analítica todos los componentes y funcionalidades que el prototipo y sus sistemas requieran.

Aunque los objetivos del desarrollo estaban claros desde un principio, los pequeños requisitos que podían surgir ramificadamente de estos a lo largo del mismo no lo eran tanto. Debido a ello algunos han mutado o desaparecido y otros nuevos y en apariencia diferentes han ocupado su lugar. Sin embargo los propositos iniciales no han variado en ningún momento y siempre que se ha establecido un cambio ha sido en pro de una mejor vía de alcance de los objetivos propuestos.

Como se mencionó en la Introducción, algunas de las prestaciones finales del prototipo figuran como analizadas y sus requisitos quedan descritos pero, en algunos casos, por problemas de plazo o por resultar medidas innecesarias algunos de estos no se ven reflejados en el prototipo final pero sí en el diseño.
De igual manera, existen muchos diseños referentes a requisitos que no eran necesarios para seguir las lineas propuestas del proyecto marcadas al inicio, pero que sin embargo no solamente han sido solventadas sino que además figuran en el apartado analítico como las demás.

A pesar de toda esta flexibilidad en el desarrollo y confección de los requisitos, la amplia mayoría de ellos forma parte de la aproximación inicial al problema y surgen de las reuniones iniciales con el ”cliente”, que en el caso de este proyecto, en realidad, se trata del profesor tutor de este trabajo de fin de grado, con el cual se mantuvieron varias reuniones para concertar las necesidades del proyecto.

Resulta importante mencionar que, a pesar de que su descripción viene dada y atañe al apartado de Diseño, se mencionan los elementos del prototipo (***Kerkaf, Pantalla*** y ***Back***) para describir el contexto de ciertos casos de uso y requisitos del sistema, ya que de otra manera describirlos correctamente en el documento no sería posible.

4.1. Identificación de usuarios

El Sistema está diseñado para ser utilizado por dos usuarios principales:

- Profesor (Interactuador dígital)
- Alumno (Interactuador Físico)

Resulta importante aclarar esta condición de Interactuador ”físico” y ”digital”:
Como ya se ha descrito, este sistema esta pensado para tratar con los usuarios de dos maneras diferentes,

una de ellas implica utilizar un dispositivo Android móvil, lo cual es una ya clásica interfaz digital y por otro lado utilizando una interfaz física de botones y ***Pantalla*** las cuales aunque puedan parecer poco relacionadas con un sistema informático, en un sentido práctico se aplica la misma idiosincrasia de diseño software en lo que a metodología de interacción con el usuario se refiere, es decir, **lo único que cambia es la manera de interactuar con el sistema** por lo que el análisis y diseño software resultan igualmente convenientes aqui.

4.1.1. Profesor (Interactuador dígital)

El papel de este usuario es el de decidir que datos y como se cargan en la Pantalla.
Mediante la aplicación Android ***Kerkaf*** Sobre la cual navega para introducirlos y configurar el comportamiento del dispositivo.
Asimismo este usuario interactúa asíncronamente con el otro (alumno) mediante una de las opciones de la ***Pantalla*** que permite al alumno dejar un aviso, como si de un timbre se tratara, para el usuario profesor, el cual recibe en su dispositivo móvil.

4.1.2. Alumno (Interactuador Físico)

En el extremo del Alumno, en cambio, las principales formas de interactuar y casos de uso atañen a la ***Pantalla*** casi exclusivamente, salvo en el caso de la llamada al otro extremo dando el aviso al Profesor.
Todas los demás casos de uso se realizan utilizando el circuito físico creado para interactuar con las entradas digitales de la ***Pantalla*** y mediante estas se obtienen las respuestas visuales deseadas.

4.2. Requisitos

En esta sección del capitulo se enumeran y describen los requisitos que se desprenden de los objetivos propuestos.

4.2.1. Requisitos funcionales

Los requisitos funcionales tienen la meta de describir con exactitud las funcionalidades que el sistema o sus componentes deben implementar.

- RF-1: El sistema deberá mostrar en su componente ***Pantalla*** Un panel de información inicial que además sea su vista por defecto.
- RF-2: El sistema deberá permitir al Alumno consultar mediante un menú diferentes vistas con diferente información.
- RF-3: El sistema deberá permitir al Alumno consultar una vista con información del horario del Profesor.
- RF-4: El sistema deberá permitir al Alumno consultar una vista con información sobre el horario de tutorías del Profesor y su dirección de correo.
- RF-5: El sistema deberá permitir al Alumno consultar una vista con la lista de asignaturas que el Profesor haya cargado en la pantalla.

- RF-6: El sistema deberá permitir al Alumno realizar desde la vista del menú un aviso que llegue al dispositivo móvil del profesor.
- RF-7: El sistema deberá permitir al Profesor Identificar la ***Pantalla*** con la que interactuar.
- RF-8: El sistema deberá permitir al Profesor determinar la información que se mostrará en el panel de información de la ***Pantalla***.
- RF-9: El sistema deberá permitir al Profesor elegir a que opciones de la Vista del menú en la ***Pantalla*** tiene acceso el Alumno.
- RF-10: El sistema deberá permitir al Profesor determinar la información que se mostrará en la vista del Horario de la ***Pantalla***.
- RF-11: El sistema deberá permitir al Profesor determinar la información que se mostrará en la vista de Tutorías de la ***Pantalla***.
- RF-12: El sistema deberá permitir al Profesor determinar la información que se mostrará en la vista de Asignaturas de la ***Pantalla***.
- RF-13: El sistema deberá permitir al Profesor determinar la dirección de correo electrónico a la cual le llegará el aviso de que el Alumno le está llamando desde la ***Pantalla***.

4.2.2. Requisitos no funcionales

También llamados requisitos de Calidad, estas especificaciones determinan el ”como” que el ”qué” de los requisitos funcionales especifican, determinando así como abordar el diseño del sistema, sus características en ”no-funcionalidad” y los entornos y filosofías de funcionamiento de las soluciones dadas.

- RNF-1: El sistema deberá utilizar la tecnología de comunicación inalámbrica bluetooth para comunicar la ***Pantalla*** con el ***Back***.
- RNF-2: El sistema deberá utilizar el protocolo de comunicación SSH a través de internet para conectar ***Kerkaf*** con el ***Back***.
- RNF-3: El sistema deberá implementar físicamente una interfaz de manejo de la ***Pantalla*** mediante componentes electrónicos.
- RNF-4: El sistema deberá implementar el componente físico de la ***Pantalla*** mediante una placa TTGO v2.3 con chip ESP32 y pantalla de tinta electrónica.
- RNF-5: El sistema deberá poder ejecutar su componente ***Back*** en cualquier sistema operativo **UNIX** con acceso a una antena de conectividad bluetooth.
- RNF-6: El sistema deberá Comprobar la integridad y correcta transmisión de la información de un extremo cualquiera a otro del prototipo.
- RNF-7-: El sistema deberá realizar el aviso al profesor de la llamada del Alumno mediante correo electrónico.

- RNF-8: El sistema implementara un sistema de seguimiento o *log* para poder realizar un correcto seguimiento de las actuaciónes del ***Back***.

Aparte de los requisitos no funcionales brutos como tal, existen otras categorías de requisitos no funcionales que se aportan también al documento pertenecientes a restricciones de tipo especial:

Requisitos no funcionales de Accesibilidad

Estos requisitos determinan de que manera será accesible el sistema e indirectamente aquellos que pueden usarlo.

- RNFA-1: El sistema deberá poder instalarse y funcionar en cualquier dispositivo Android de versión 5.0 o superior.

Requisitos no funcionales de Seguridad

Estos requisitos especifican los criterios de seguridad del sistema.

- RNFS-1: El sistema deberá controlar mediante contraseña que el Profesor acceda únicamente a su ***Pantalla***.
- RNFS-2: El sistema deberá establecer un método de comunicación restrictivo entre la ***Pantalla*** y el ***Back*** evitando que cualquier dispositivo bluetooth pueda usurpar la identidad de alguno de los dos componentes.
- RNFS-3: El sistema en su parte ***Back*** requerirá de un emparejamiento Bluetooth para controlar la Pantalla.

Requisitos no funcionales de Usabilidad

Estos requisitos determinan la facilidad de uso y aprendizaje que debe mostrar el sistema a sus usuarios.

- RNFU-1: El sistema debe aportar confiadamente las respuestas esperadas.
- RNFU-2: El sistema deberá implementar una funcionalidad interactiva en sus vistas con facilidad de uso y aprendizaje.
- RNFU-3: El sistema deberá funcionar con una velocidad de respuesta y actuación aceptables para una interacción humana.

Requisitos no funcionales de Disponibilidad

Los requisitos de Disponibilidad describen en que circunstancias el sistema estará listo para ser utilizado.

- RNFD-1: El sistema deberá estar disponible todo el tiempo en el que el Profesor se encuentre en el centro y no menos.
- RNFD-2: Siempre que la conectividad bluetooth lo permita, el dispositivo ***Pantalla*** deberá ser alcanzable por el ***Back*** y por ende por ***Kerkaf*** en todo momento previsto por el RNFD-1.

Requisitos no funcionales de Información

En los requisitos de información se encuentran descritos los datos que el sistema deberá almacenar y tratar.

- RNFI-1: El sistema deberá almacenar Los siguientes datos sobre el ***Back*** y por lo tanto ***Pantalla*** al que se va a conectar el Profesor:
  - Dirección (IP)
  - Puerto de conexión
- RNFI-2: El sistema usará la siguiente información de identificación de usuario para acceder al ***Back***:
  - Usuario
  - Contraseña
- RNFI-3: El sistema deberá almacenar y utilizar para envíar el correo de aviso la siguiente información:
  - Correo electrónico deseado por el Profesor para recibir el aviso en la bandeja de su dispositivo móvil.
- RNFI-4: El sistema deberá almacenar información sobre el horario del Profesor, para cada clase:
  - Dia de la semana
  - Hora de la clase
  - Codigo de la clase (en siglas o nombre corto)
- RNFI-5: El sistema deberá almacenar la siguiente información a presentar en el panel de información de la ***Pantalla***:
  - Nº de despacho o sala que la ***Pantalla*** está identificando.
  - Nombre y titulo o descripción del personal del despacho o sala al que la pantalla identifica
  - Comentario sobre el departamento o localización o actividad actual del Profesor
- RNFI-6: El sistema deberá almacenar únicamente un campo de texto explicativo que se muestra en la vista de Tutorías
- RNFI-7: El sistema deberá almacenar la siguiente información para la vista de Asingaturas de la ***Pantalla***:
  - Cabecera de la vista, con datos sobre el departamento o nombre del Profesor.
  - Listado (como cadena de texto) de asignaturas impartidas por el Profesor.

Requisitos no funcionales de Restricción de Información

Estos requisitos imponen normas sobre los datos que el sistema trata, como su cantidad o tipo.

- RNFRI-1: El sistema deberá asegurar que solo exista una dirección de correo (a la que se envía el aviso) en todo momento.

- RNFRI-2: El sistema almacenará únicamente la información relevante a una sola ***Pantalla*** en todo momento.

- RNFRI-3: El sistema deberá asegurar que solamente pueda existir una asignatura para una misma hora y día de la semana.

- RNFRI-4: El sistema almacenará para cada vista de las interfaces de la ***Pantalla*** no más de un dato o cadena de texto por campo.

4.3. Casos de uso

Un caso de uso es una descripción de una actividad o acción que uno o mas usuarios definidos realizan en su uso del sistema.
Estos quedan descritos en la fase de análisis de la preparación y planificación de un proyecto para así determinar de manera especificada como debe reaccionar el sistema y que debe hacer ante las acciones del usuario describiendo el proceso de una manera mas concreta que los requisitos funcionales.

4.3.1. Diagramas de casos de uso

Un diagrama de caso de uso, según su definición, es ”Una descripción de las actividades que deberá realizar alguien o algo para llevar a cabo algún proceso” y añade: ”Los personajes o entidades que participarán en un diagrama de caso de uso se denominan actores”. [2]
Nótese que en este caso, los actores ”usuario” ya han sido definidos como Profesor y Alumno, por lo que sus diagramas de uso son respectivamente el de la figura 4.1 y 4.2.

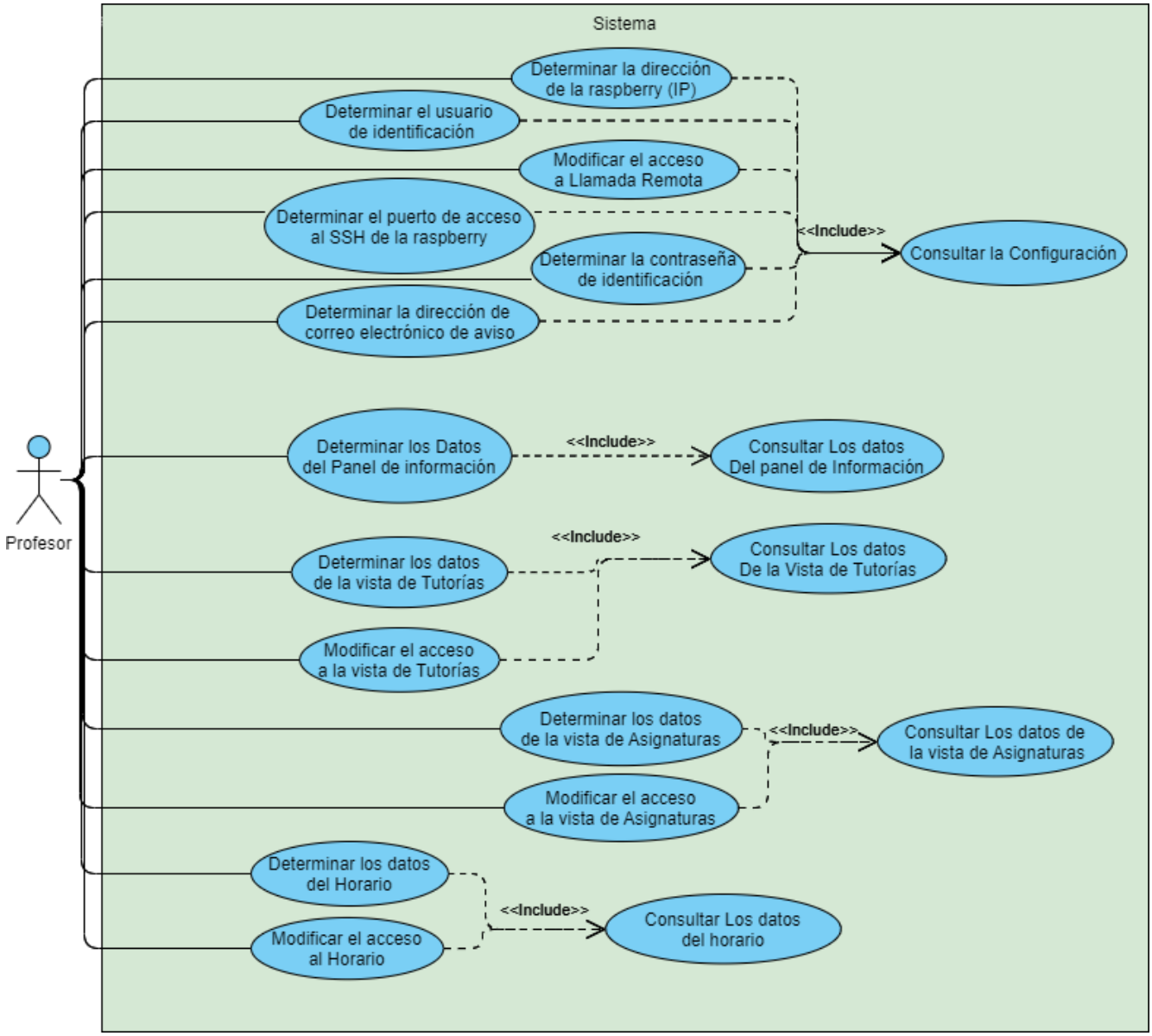

Figura 4.1: Diagrama de casos de uso del usuario Profesor

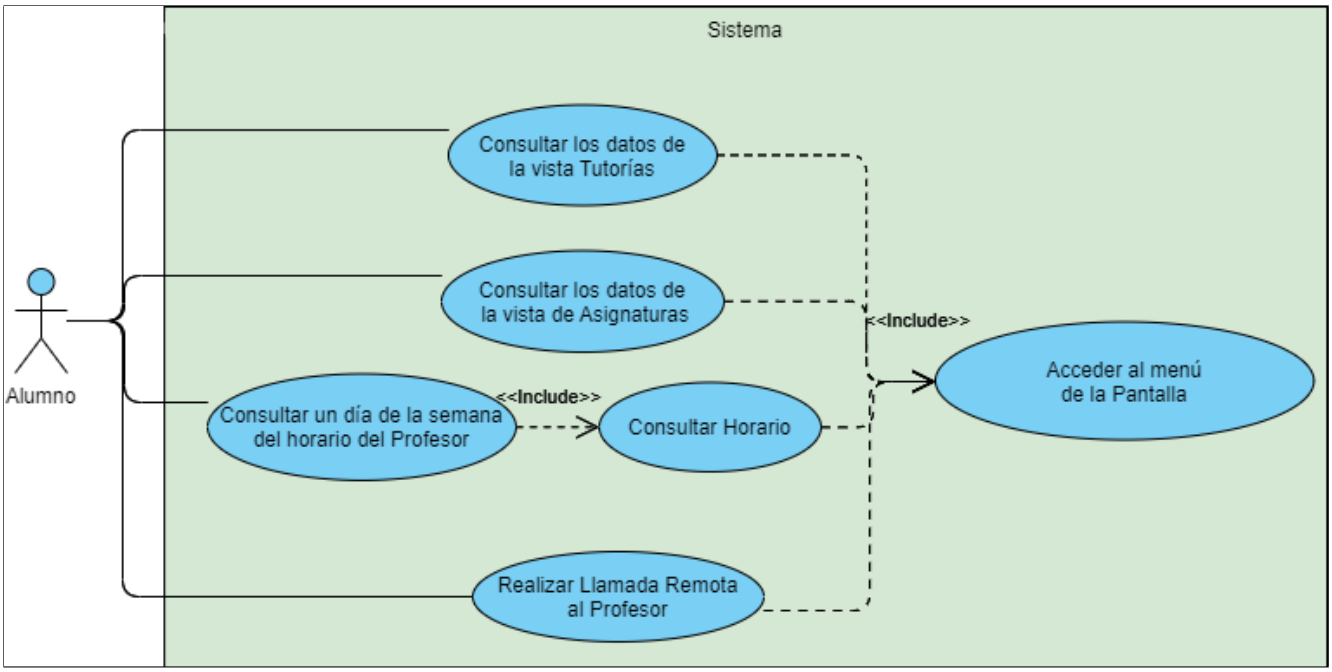

Figura 4.2: Diagrama de casos de uso del usuario Alumno

4.3.2. Especificación de los casos de uso

La lista de la totalidad de los casos de uso contemplados en la etapa analítica es la siguiente:

Profesor:

- CU-1: Consultar la configuración
- CU-2: Determinar la dirección de la Raspberry (IP).
- CU-3: Determinar el puerto de acceso al SSH de la Raspberry.
- CU-4: Determinar el usuario de identificación.
- CU-5: Determinar la contraseña de identificación.
- CU-6: Determinar la dirección de correo electrónico de aviso.
- CU-7: Modificar el acceso a la función de Llamada Remota.
- CU-8: Consultar los datos del panel de información.
- CU-9: Determinar los datos del panel de información.
- CU-10: Consultar los datos de la vista de Tutorías.
- CU-11: Determinar los datos de la vista de Tutorías.
- CU-12: Modificar el acceso a la vista de Tutorías.
- CU-13: Consultar los datos de la vista de Asignaturas.
- CU-14: Determinar los datos de la vista de Asignaturas.

- CU-15: Modificar el acceso a la vista de Asignaturas.
- CU-16: Consultar los datos del Horario.
- CU-17: Determinar los datos de la vista Horario.
- CU-18: Modificar el acceso al Horario.

Alumno:

- CU-19: Acceder al menú de la ***Pantalla***.
- CU-20: Consultar los datos de la vista de Tutorías.
- CU-21: Consultar los datos de la vista de Asignaturas.
- CU-22: Consultar un día de la semana del horario del Profesor.
- CU-23: Realizar una Llamada Remota al Profesor.

Las especificaciones del flujo de proceso de los casos de uso mostrados son las siguientes:

CU-1	**Consultar la Configuración**
Actor	**Profesor**
Descripción	El sistema deberá permitir al Profesor consultar la configuración.
Precondición	El Usuario ha accedido a ***Kerkaf*** en su dispositivo móvil y se encuentra en la vista inicial.
Secuencia Normal	1. El Profesor accede al la vista de *Configuración*. 2. El sistema muestra la configuración con todos sus datos actuales.
Postcondición	El Profesor se encuentra en la vista de *Configuración*
Excepciones	
Variación	**Acción**
1b	El usuario selecciona retroceder en su dispositivo y se abandona la aplicación

Cuadro 4.1: Caso de Uso 1 - Consultar la configuración

CU-2	**Determinar Dirección de la Raspberry**
Actor	**Profesor**
Descripción	El sistema deberá permitir al Profesor modificar la dirección IP de la Raspberry a la que se solicitan los servicios del ***Back***.
Precondición	(CU-1) El Profesor debe haber accedido a la vista de configuración de ***Kerkaf***.
Secuencia Normal	1. El Profesor modifica el campo de ”Dirección IP” 2. El Profesor pulsa el botón de ”Guardar” 3. El Sistema guarda el dato y retorna a la vista inicial de ***Kerkaf***
Postcondición	El Profesor ha identificado la dirección IP de la Raspberry ***Back***.
Excepciones	
Variación	**Acción**
2b	El Profesor se arrepiente de la modificación y no desea guardarla por lo que retrocede a la vista inicial.
3b	Si Algún campo de la configuración está vació o es incorrecto el sistema se niega a guardarlo y lo comunica al Profesor.

Cuadro 4.2: Caso de Uso 2 - Determinar dirección IP de la Raspberry

CU-3	**Determinar el puerto de acceso al SSH de la Raspberry**
Actor	**Profesor**
Descripción	El sistema deberá permitir al Profesor modificar el puerto en el que la Raspberry atiende al protocolo SSH donde se ejecutan los servicios del ***Back***.
Precondición	(CU-1) El Profesor debe haber accedido a la vista de configuración de ***Kerkaf***.
Secuencia Normal	1. El Profesor modifica el campo de ”Puerto SSH” 2. El Profesor pulsa el botón de ”Guardar” 3. El Sistema guarda el dato y retorna a la vista inicial de ***Kerkaf***
Postcondición	El Profesor ha determinado el puerto SSH al que conectarse con el ***Back***.
Excepciones	
Variación	**Acción**
2b	El Profesor se arrepiente de la modificación y no desea guardarla por lo que retrocede a la vista inicial.
3b	Si Algún campo de la configuración está vació o es incorrecto el sistema se niega a guardarlo y lo comunica al Profesor.

Cuadro 4.3: Caso de Uso 3 - Determinar el puerto SSH de la Raspberry

CU-4	Determinar el usuario de identificación
Actor	**Profesor**
Descripción	El sistema deberá permitir al Profesor determinar el usuario con el que se identifica en el ***Back*** de la Raspberry.
Precondición	(CU-1) El Profesor debe haber accedido a la vista de configuración de ***Kerkaf***.
Secuencia Normal	1. El Profesor modifica el campo de ”Usuario” 2. El Profesor pulsa el botón de ”Guardar” 3. El Sistema guarda el dato y retorna a la vista inicial de ***Kerkaf***
Postcondición	El Profesor ha identificado al usuario que se usa en el ***Back***.
Excepciones	
Variación	**Acción**
2b	El Profesor se arrepiente de la modificación y no desea guardarla por lo que retrocede a la vista inicial.
3b	Si Algún campo de la configuración está vació o es incorrecto el sistema se niega a guardarlo y lo comunica al Profesor.

Cuadro 4.4: Caso de Uso 4 - Determinar el usuario de identificación

CU-5	Determinar la contraseña de identificación
Actor	**Profesor**
Descripción	El sistema deberá permitir al Profesor determinar la contraseña con el que se identifica en el ***Back*** de la Raspberry.
Precondición	(CU-1) El Profesor debe haber accedido a la vista de configuración de ***Kerkaf***.
Secuencia Normal	1. El Profesor modifica el campo de ”Contraseña” 2. El Profesor pulsa el botón de ”Guardar” 3. El Sistema guarda el dato y retorna a la vista inicial de ***Kerkaf***
Postcondición	El Profesor ha determinado la contraseña que se usa en el ***Back***.
Excepciones	
Variación	**Acción**
2b	El Profesor se arrepiente de la modificación y no desea guardarla por lo que retrocede a la vista inicial.
3b	Si Algún campo de la configuración está vació o es incorrecto el sistema se niega a guardarlo y lo comunica al Profesor.

Cuadro 4.5: Caso de Uso 5 - Determinar la contraseña de identificación

CU-6	**Determinar la dirección de correo electrónico de aviso.**
Actor	**Profesor**
Descripción	El sistema deberá permitir al Profesor modificar la configuración al respecto de la dirección de correo electrónico a la cual se realiza la Llamada Remota.
Precondición	El Profesor debe haber accedido a la vista de configuración de ***Kerkaf*** y haber identificado correctamente su ***Pantalla***.
Secuencia Normal	1. El Profesor accede al la subvista de *Llamada Remota* de la *Configuración*. 2. El Sistema muestra la configuración actual al respecto de la opción de llamada remota y el correo al cual se realiza. 3. El Profesor Modifica el campo de ”dirección de Correo Electrónico” 4. El Profesor Pulsa el botón de ”Enviar” 5. El Sistema modifica el correo y confirma su éxito al Profesor.
Postcondición	El Profesor ha modificado la dirección de correo electrónico a la que se envía la Llamada Remota
Excepciones	
Variación	**Acción**
2b	El Sistema no posee ninguna dirección aún y muestra el campo editable en blanco.
4b	El Profesor no desea modificar el dato y retrocede volviendo a la vista inicial de ***Kerkaf***
5b	El Sistema no logra comunicarse con el ***Back*** y el caso de uso retorna al paso 2

Cuadro 4.6: Caso de Uso 6 - Modificar dirección de correo

CU-7	**Habilitar la función de la Llamada Remota**.
Actor	**Profesor**
Descripción	El sistema deberá permitir al Profesor modificar la configuración del sistema permitiendo o no a su elección que el Alumno pueda realizar la Llamada Remota.
Precondición	El Profesor debe haber accedido a la vista de configuración de ***Kerkaf*** y haber identificado correctamente su ***Pantalla***.
Secuencia Normal	1. El Profesor accede al la subvista de *Llamada Remota* de la *Configuración*. 2. El Sistema muestra la configuración actual al respecto de la opción de llamada remota y el correo al cual se realiza. 3. El Profesor Modifica la opción de ”Llamada Remota” a la deseada. 4. El Profesor Pulsa el botón de ”Enviar” 5. El Sistema modifica la opción y confirma su éxito al Profesor.
Postcondición	El sistema permite o no al Alumno seleccionar la opción de ”Llamada Remota” en función de la opción seleccionada por el Profesor.
Excepciones	
Variación	**Acción**
3b	El Profesor no desea modificarla opción y retrocede volviendo a la vista inicial de ***Kerkaf***
5b	El campo de la dirección de correo se encuentra vacío y el sistema al comprobarlo da error y el caso de uso retorna al paso 2.
5c	El Sistema no logra comunicarse con el ***Back*** o este con la ***Pantalla*** y el caso de uso retorna al paso 2.

Cuadro 4.7: Caso de Uso 7 - Habilitar la función de la Llamada Remota

CU-8	**Consultar los datos del panel de información principal**
Actor	**Profesor**
Descripción	El sistema deberá permitir al Profesor consultar los datos seleccionados en ese momento en ***Kerkaf*** que se están mostrando en el panel de información de la ***Pantalla***.
Precondición	El Usuario ha accedido a ***Kerkaf*** en su dispositivo móvil y se encuentra en la vista inicial.
Secuencia Normal	1. El Profesor accede a la vista principal pulsando ”Inicio” 2. El Sistema muestra la subvista por defecto de la vista principal con los datos actuales del panel de información.
Postcondición	El Profesor se encuentra en la subvista por defecto de la vista principal.
Excepciones	
Variación	**Acción**
2b	El Sistema no tiene cargados aún datos al respecto y muestra los campos en blanco

Cuadro 4.8: Caso de Uso 8 - Consultar los datos del Panel de información

CU-9	**Determinar los datos del panel de información**
Actor	**Profesor**
Descripción	El sistema deberá permitir al Profesor modificar los datos del panel de información desde su subvista correspondiente en ***Kerkaf*** y mostrarlos en la ***Pantalla***.
Precondición	El Profesor debe encontrarse en la vista Inicial de ***Kerkaf*** y haber identificado correctamente su ***Pantalla***.
Secuencia Normal	1. El Profesor accede a la vista principal pulsando ”Inicio” 2. El Sistema muestra la subvista por defecto de la vista principal con los datos actuales del panel de información. 3. El Profesor determina a placer los datos y campos de texto de la vista que se corresponderán con los mostrados en la ***Pantalla***. 4. El Profesor Pulsa el botón de ”Enviar” 5. El Sistema modifica los datos en la ***Pantalla*** y confirma su éxito.
Postcondición	El Profesor ha modificado los datos a mostrar en el panel de información de la ***Pantalla***.
Excepciones	
Variación	**Acción**
3b	El Profesor no desea modificar los datos y retrocede volviendo a la vista inicial de ***Kerkaf***
5b	Los campos de datos a enviar están todos vacíos y el Sistema al comprobarlo da error y el caso de uso retorna al paso 2.
5c	El Sistema no logra comunicarse con el ***Back*** o este con la ***Pantalla*** y el caso de uso retorna al paso 2.

Cuadro 4.9: Caso de Uso 9 - Determinar los datos del panel de información

CU-10	**Consultar los datos de la vista de Tutorías**
Actor	**Profesor**
Descripción	El sistema deberá permitir al Profesor consultar los datos seleccionados en ese momento en ***Kerkaf*** que se están mostrando en el vista de Tutorías de la ***Pantalla***.
Precondición	El Usuario ha accedido a ***Kerkaf*** en su dispositivo móvil y se encuentra en la vista principal.
Secuencia Normal	1. El Profesor se desplaza hasta la subvista de *Tutorías*. 2. El Sistema muestra la subvista con los datos y opciones actuales de la vista de Tutorías de la ***Pantalla***.
Postcondición	El Profesor se encuentra en la subvista de *Tutorías*.
Excepciones	
Variación	**Acción**
2b	El Sistema no tiene cargados aún datos al respecto y muestra los campos en blanco

Cuadro 4.10: Caso de Uso 10 - Consultar los datos de la Vista de Tutorías

CU-11	Determinar los datos de la Vista de Tutorías
Actor	**Profesor**
Descripción	El sistema deberá permitir al Profesor modificar los datos de la Vista de Tutorías desde su subvista correspondiente en ***Kerkaf*** y mostrarlos en la ***Pantalla***.
Precondición	El Profesor debe encontrarse en la vista principal de ***Kerkaf*** y haber identificado correctamente su ***Pantalla***.
Secuencia Normal	1. El Profesor se desplaza hasta la subvista de *Tutorías*. 2. El Sistema muestra la subvista con los datos y opciones actuales de la vista de Tutorías de la ***Pantalla***. 3. El Profesor determina a placer los datos y campos de texto de la vista que se corresponderán con los mostrados en la ***Pantalla***. 4. El Profesor Pulsa el botón de ”Enviar” 5. El Sistema modifica los datos en la ***Pantalla*** y confirma su éxito.
Postcondición	El Profesor ha modificado los datos a mostrar en la Vista de Tutorías de la ***Pantalla***.
Excepciones	
Variación	**Acción**
3b	El Profesor no desea modificar los datos y retrocede volviendo a la vista inicial de ***Kerkaf***
5b	Los campos de datos a enviar están todos vacíos y el Sistema al comprobarlo da error y el caso de uso retorna al paso 3.
5c	El Sistema no logra comunicarse con el ***Back*** o este con la ***Pantalla*** y el caso de uso retorna al paso 3.

Cuadro 4.11: Caso de Uso 11 - Determinar los datos de la Vista de Tutorías

CU-12	**Modificar el acceso a la Vista de Tutorías de la *Pantalla*.**
Actor	**Profesor**
Descripción	El sistema deberá permitir al Profesor modificar la configuración del sistema permitiendo o no a su elección que el Alumno pueda acceder a la vista de Tutorías.
Precondición	El Profesor debe encontrarse en la vista principal de ***Kerkaf*** y haber identificado correctamente su ***Pantalla***.
Secuencia Normal	1. El Profesor se desplaza hasta la subvista de *Tutorías*. 2. El Sistema muestra la configuración actual al respecto de la Vista de Tutorías. 3. El Profesor Modifica la opción de ”Permitir Acceso” a la deseada. 4. El Profesor Pulsa el botón de ”Enviar” 5. El Sistema modifica la opción y confirma su éxito al Profesor.
Postcondición	El sistema permite o no al Alumno acceder a la vista de Tutorías de la ***Pantalla*** dependiendo de lo estipulado por el Profesor.
Excepciones	
Variación	**Acción**
3b	El Profesor no desea modificarla opción y retrocede volviendo a la vista inicial de ***Kerkaf***
5b	Los campos de datos a enviar están todos vacíos y el Sistema al comprobarlo da error y el caso de uso retorna al paso 3.
5c	El Sistema no logra comunicarse con el ***Back*** o este con la ***Pantalla*** y el caso de uso retorna al paso 3

Cuadro 4.12: Caso de Uso 12 - Modificar el acceso a la Vista de Tutorías de la ***Pantalla***.

CU-13	**Consultar los datos de la vista de Asignaturas**
Actor	**Profesor**
Descripción	El sistema deberá permitir al Profesor consultar los datos seleccionados en ese momento en ***Kerkaf*** que se están mostrando en el vista de Asignaturas de la ***Pantalla***.
Precondición	El Usuario ha accedido a ***Kerkaf*** en su dispositivo móvil y se encuentra en la vista principal.
Secuencia Normal	1. El Profesor se desplaza hasta la subvista de *Asignaturas*. 2. El Sistema muestra la subvista con los datos y opciones actuales de la vista de Asignaturas de la ***Pantalla***.
Postcondición	El Profesor se encuentra en la subvista de *Asignaturas*.
Excepciones	
Variación	**Acción**
2b	El Sistema no tiene cargados aún datos al respecto y muestra los campos en blanco

Cuadro 4.13: Caso de Uso 13 - Consultar los datos de la Vista de Asignaturas

CU-14	Determinar los datos de la Vista de Asignaturas
Actor	**Profesor**
Descripción	El sistema deberá permitir al Profesor modificar los datos de la Vista de Asignaturas desde su subvista correspondiente en ***Kerkaf*** y mostrarlos en la ***Pantalla***.
Precondición	El Profesor debe encontrarse en la vista principal de ***Kerkaf*** y haber identificado correctamente su ***Pantalla***.
Secuencia Normal	1. El Profesor se desplaza hasta la subvista de *Asignaturas*. 2. El Sistema muestra la subvista con los datos y opciones actuales de la vista de Asignaturas de la ***Pantalla***. 3. El Profesor determina a placer los datos y campos de texto de la vista que se corresponderán con los mostrados en la ***Pantalla***. 4. El Profesor Pulsa el botón de ”Enviar” 5. El Sistema modifica los datos en la ***Pantalla*** y confirma su éxito.
Postcondición	El Profesor ha modificado los datos a mostrar en la Vista de Asignaturas de la ***Pantalla***.
Excepciones	
Variación	**Acción**
3b	El Profesor no desea modificar los datos y retrocede volviendo a la vista inicial de ***Kerkaf***
5b	Los campos de datos a enviar están todos vacíos y el Sistema al comprobarlo da error y el caso de uso retorna al paso 3.
5c	El Sistema no logra comunicarse con el ***Back*** o este con la ***Pantalla*** y el caso de uso retorna al paso 3.

Cuadro 4.14: Caso de Uso 14 - Determinar los datos de la Vista de Asignaturas

CU-15	**Modificar el acceso a la Vista de Asignaturas de la *Pantalla*.**
Actor	**Profesor**
Descripción	El sistema deberá permitir al Profesor modificar la configuración del sistema permitiendo o no a su elección que el Alumno pueda acceder a la vista de Asignaturas.
Precondición	El Profesor debe encontrarse en la vista principal de ***Kerkaf*** y haber identificado correctamente su ***Pantalla***.
Secuencia Normal	1. El Profesor se desplaza hasta la subvista de *Asignaturas*. 2. El Sistema muestra la configuración actual al respecto de la Vista de Asignaturas. 3. El Profesor Modifica la opción de ”Permitir Acceso” a la deseada. 4. El Profesor Pulsa el botón de ”Enviar” 5. El Sistema modifica la opción y confirma su éxito al Profesor.
Postcondición	El sistema permite o no al Alumno acceder a la vista de Asignaturas de la ***Pantalla*** dependiendo de lo estipulado por el Profesor.
Excepciones	
Variación	**Acción**
3b	El Profesor no desea modificarla opción y retrocede volviendo a la vista inicial de ***Kerkaf***
5b	Los campos de datos a enviar están todos vacíos y el Sistema al comprobarlo da error y el caso de uso retorna al paso 3.
5c	El Sistema no logra comunicarse con el ***Back*** o este con la ***Pantalla*** y el caso de uso retorna al paso 3

Cuadro 4.15: Caso de Uso 15 - Modificar el acceso a la Vista de Asignaturas de la ***Pantalla***.

CU-16	**Consultar los datos del Horario**
Actor	**Profesor**
Descripción	El sistema deberá permitir al Profesor consultar los datos seleccionados en ese momento en ***Kerkaf*** que se están mostrando en el vista del Horario de la ***Pantalla***.
Precondición	El Usuario ha accedido a ***Kerkaf*** en su dispositivo móvil y se encuentra en la vista principal.
Secuencia Normal	1. El Profesor se desplaza hasta la subvista de *Horario*. 2. El Sistema muestra la subvista con los datos y opciones actuales del Horario de la ***Pantalla***.
Postcondición	El Profesor se encuentra en la subvista *Horario*.
Excepciones	
Variación	**Acción**
2b	El Sistema no tiene cargados aún datos al respecto y muestra los campos en blanco

Cuadro 4.16: Caso de Uso 16 - Consultar los datos de la Vista del Horario

CU-17	Determinar los datos de la vista Horario
Actor	**Profesor**
Descripción	El sistema deberá permitir al Profesor modificar los datos de la Vista Horario desde su subvista correspondiente en ***Kerkaf*** y mostrarlos en la ***Pantalla***.
Precondición	El Profesor debe encontrarse en la vista principal de ***Kerkaf*** y haber identificado correctamente su ***Pantalla***.
Secuencia Normal	1. El Profesor se desplaza hasta la subvista *Horario*. 2. El Sistema muestra la subvista con los datos y opciones actuales del Horario de la ***Pantalla***. 3. El Profesor introduce la hora y día de la semana y rellena el campo de texto correspondiente a la asignatura cuyos datos cargará en la ***Pantalla***. 4. El Profesor Pulsa el botón de ”Enviar” 5. El Sistema modifica los datos en la ***Pantalla*** y confirma su éxito.
Postcondición	El Profesor ha modificado los datos a mostrar en la Vista del Horario de la ***Pantalla***.
Excepciones	
Variación	**Acción**
3b	El Profesor no desea modificar los datos y retrocede volviendo a la vista inicial de ***Kerkaf***
5b	Los campos de datos a enviar están todos vacíos y el Sistema al comprobarlo da error y el caso de uso retorna al paso 3.
5c	El Sistema no logra comunicarse con el ***Back*** o este con la ***Pantalla*** y el caso de uso retorna al paso 3.

Cuadro 4.17: Caso de Uso 17 - Determinar los datos de la Vista Horario

CU-18	**Modificar el acceso a la Vista Horario de la *Pantalla*.**
Actor	**Profesor**
Descripción	El sistema deberá permitir al Profesor modificar la configuración del sistema permitiendo o no a su elección que el Alumno pueda acceder a la vista Horario.
Precondición	El Profesor debe encontrarse en la vista principal de ***Kerkaf*** y haber identificado correctamente su ***Pantalla***.
Secuencia Normal	1. El Profesor se desplaza hasta la subvista *Horario*. 2. El Sistema muestra la configuración actual al respecto de la Vista de Horario. 3. El Profesor Modifica la opción de ”Permitir Acceso” a la deseada. 4. El Profesor Pulsa el botón de ”Enviar” 5. El Sistema modifica la opción y confirma su éxito al Profesor.
Postcondición	El sistema permite o no al Alumno acceder a la vista Horario de la ***Pantalla*** dependiendo de lo estipulado por el Profesor.
Excepciones	
Variación	**Acción**
3b	El Profesor no desea modificarla opción y retrocede volviendo a la vista inicial de ***Kerkaf***
5b	Los campos de datos a enviar están todos vacíos y el Sistema al comprobarlo da error y el caso de uso retorna al paso 3.
5c	El Sistema no logra comunicarse con el ***Back*** o este con la ***Pantalla*** y el caso de uso retorna al paso 3

Cuadro 4.18: Caso de Uso 18 - Modificar el acceso a la Vista Horario de la ***Pantalla***.

CU-19	**Acceder al Menú de la *Pantalla***
Actor	**Alumno**
Descripción	El sistema deberá permitir al Alumno acceder al menú de la ***Pantalla*** y consultar las opciones disponibles.
Precondición	El Usuario se encuentra físicamente frente a la ***Pantalla*** y le es posible manipular el panel interactivo adherido a esta.
Secuencia Normal	1. El Alumno pulsa el botón de ”Menú”. 2. El sistema muestra por ***Pantalla*** el menú de opciones disponibles.
Postcondición	El Alumno se encuentra en la vista del *Menú* de la ***Pantalla***

Cuadro 4.19: Caso de Uso 19 - Acceder al menú de la ***Pantalla***

CU-20	Consultar los datos de la vista de Tutorías
Actor	**Alumno**
Descripción	El sistema deberá permitir al Alumno consultar la Vista de Tutorías de la ***Pantalla***.
Precondición	El alumno se encuentra en el *Menú* (CU-19).
Secuencia Normal	1. El Alumno se desplaza hasta la opción de ”Tutorías” mediante los botones de la interfaz física y pulsa ”seleccionar”. 2. El sistema muestra por ***Pantalla*** la vista de Tutorías y sus datos.
Postcondición	El Alumno se encuentra en la vista de Tutorías de la ***Pantalla***
Excepciones	
Variación	**Acción**
2b	El Sistema carece de todos los datos de la vista y muestra ”SIN DATOS” o en blanco los campos.
2c	El Profesor no ha habilitado el acceso a esta vista por lo que arroja un error y retorna a la situación de precondición del caso de uso.

Cuadro 4.20: Caso de Uso 20 - Consultar los datos de la vista de Tutorías

CU-21	Consultar los datos de la vista de Asignaturas
Actor	**Alumno**
Descripción	El sistema deberá permitir al Alumno consultar la Vista de Asignaturas de la ***Pantalla***.
Precondición	El alumno se encuentra en el *Menú* (CU-19).
Secuencia Normal	1. El alumno se desplaza hasta la opción de ”Asignaturas” mediante los botones de la interfaz física y pulsa ”seleccionar”. 2. El sistema muestra por ***Pantalla*** la vista de Asignaturas y sus datos.
Postcondición	El Alumno se encuentra en la vista de Asignaturas de la ***Pantalla***
Excepciones	
Variación	**Acción**
2b	El Sistema carece de todos los datos de la vista y muestra ”SIN DATOS” o en blanco los campos.
2c	El Profesor no ha habilitado el acceso a esta vista por lo que arroja un error y retorna a la situación de precondición del caso de uso.

Cuadro 4.21: Caso de Uso 21 - Consultar los datos de la vista de Asignaturas

CU-22	Consultar un día de la semana del Horario del Profesor
Actor	**Alumno**
Descripción	El sistema deberá permitir al Alumno consultar un día concreto del horario semanal del profesor en la ***Pantalla***.
Precondición	El alumno se encuentra en el *Menú* (CU-19).
Secuencia Normal	1. El alumnos se desplaza hasta la opción de ”Horario Clases” mediante los botones de la interfaz física y pulsa ”seleccionar”. 2. El sistema muestra por ***Pantalla*** la vista de Horario en su subvista correspondiente al *lunes* y los datos de ese día. 3. El Alumno se desplaza hasta la subvista deseada mediante los botones de ”anterior” y ”siguiente” 4. El sistema muestra la subvista correspondiente al día que el alumno desee consultar.
Postcondición	El Alumno se encuentra en la vista de Horario en la subvista seleccionada en la ***Pantalla***
Excepciones	
Variación	**Acción**
2b	El Alumno ya se encuentra en el día que estaba buscando y termina el caso de uso
2c	El Profesor no ha habilitado el acceso a esta vista por lo que arroja un error y retorna a la situación de precondición del caso de uso.

Cuadro 4.22: Caso de Uso 22 - Consultar un día de la semana del horario del Profesor

CU-23	Realizar una Llamada Remota al profesor
Actor	**Alumno**
Descripción	El sistema deberá permitir al Alumno colocar un aviso al Profesor mediante el cual el Sistema enviará un correo electrónico a la dirección estipulada por el Profesor (CU-6).
Precondición	El alumno se encuentra en el *Menú* (CU-19).
Secuencia Normal	1. El alumnos se desplaza hasta la opción de ”Llamada Remota” mediante los botones de la interfaz física y pulsa ”seleccionar”. 2. El sistema muestra un mensaje de espera mientras la ***Pantalla*** se comunica con el ***Back***. 3. El sistema informa de que ha tenido éxito y retorna al panel de información.
Postcondición	El Alumno se encuentra en el panel de información de la ***Pantalla*** y un correo de aviso se ha enviado a la dirección estipulada por el Profesor.
Excepciones	
Variación	**Acción**
2b	El Profesor no ha habilitado el acceso a esta vista por lo que arroja un error y retorna a la situación de precondición del caso de uso.
3b	El Sistema no consigue comunicarse con éxito y retorna a la situación de precondición del caso de uso

Cuadro 4.23: Caso de Uso 23 - Realizar una Llamada Remota al profesor

4.4. Modelo de dominio

En la siguiente figura (4.3) se muestra un modelo abstracto de la composición lógica del sistema, mas adelante en el capítulo de diseño se detalla un diagrama de despliegue mas concreto.
También es importante detallar que las operaciones mostradas son genéricas y los componentes poseen operaciones mas específicas sobre su funcionalidad real y abstracta según los casos de uso.

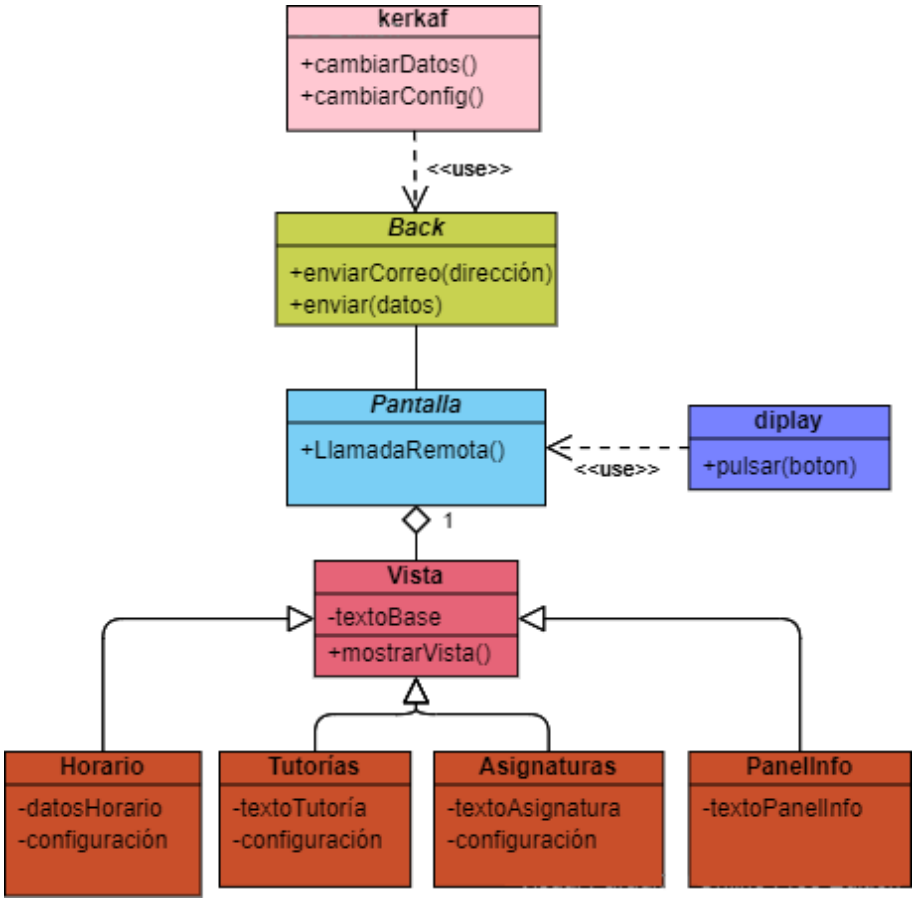

Figura 4.3: Diagrama del modelo de dominio del Sistema en su conjunto

4.5. Diagramas de secuencia

En esta sección se muestra detalladamente el transcurso de acciones en casos de uso muy representativos, se han omitido los casos de uso duplicados y similares, ya que mostrarlos resulta redundante.

4.5.1. (CU-4) Determinar el usuario de identificación:

Este caso de uso resulta especialmente representativo de la interacción del profesor con la configuración de comunicación de ***Kerkaf***, que hace las veces de identificación, ya que determina la ***Pantalla*** que se está utilizando.

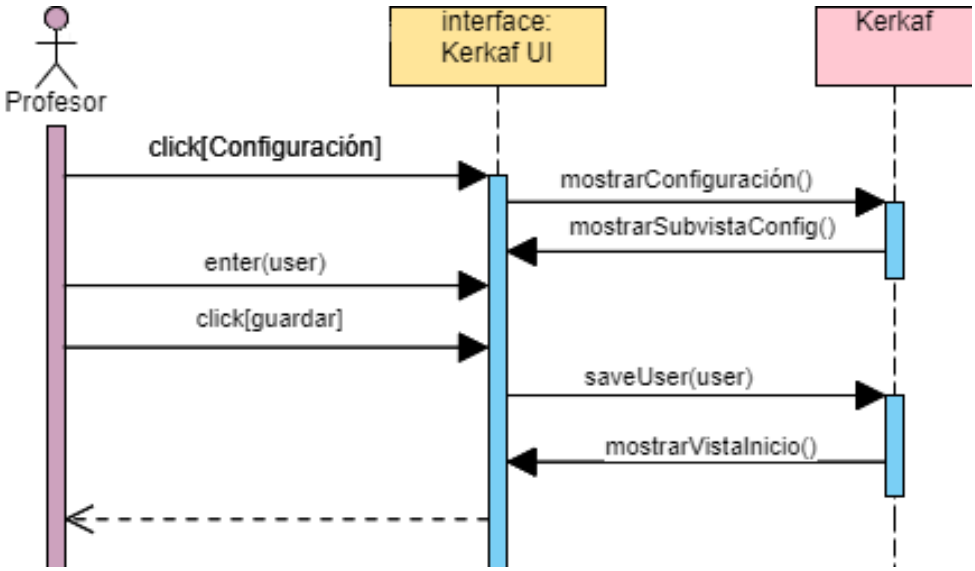

Figura 4.4: Diagrama de secuencia del caso de uso 4

4.5.2. (CU-9) Determinar los datos del panel de información:

El diagrama 4.5 se asimila a los otros 3 ejemplos de casos de uso en el que se modifican los valores a mostrar en el display de tinta electrónica. El flujo de funcionamiento es el mismo en todos salvo por lo que se mostrará en el siguiente caso de uso de Horario y tiene la particularidad junto a este de que refresca la vista con los datos nuevos tras el envío de los mismos.

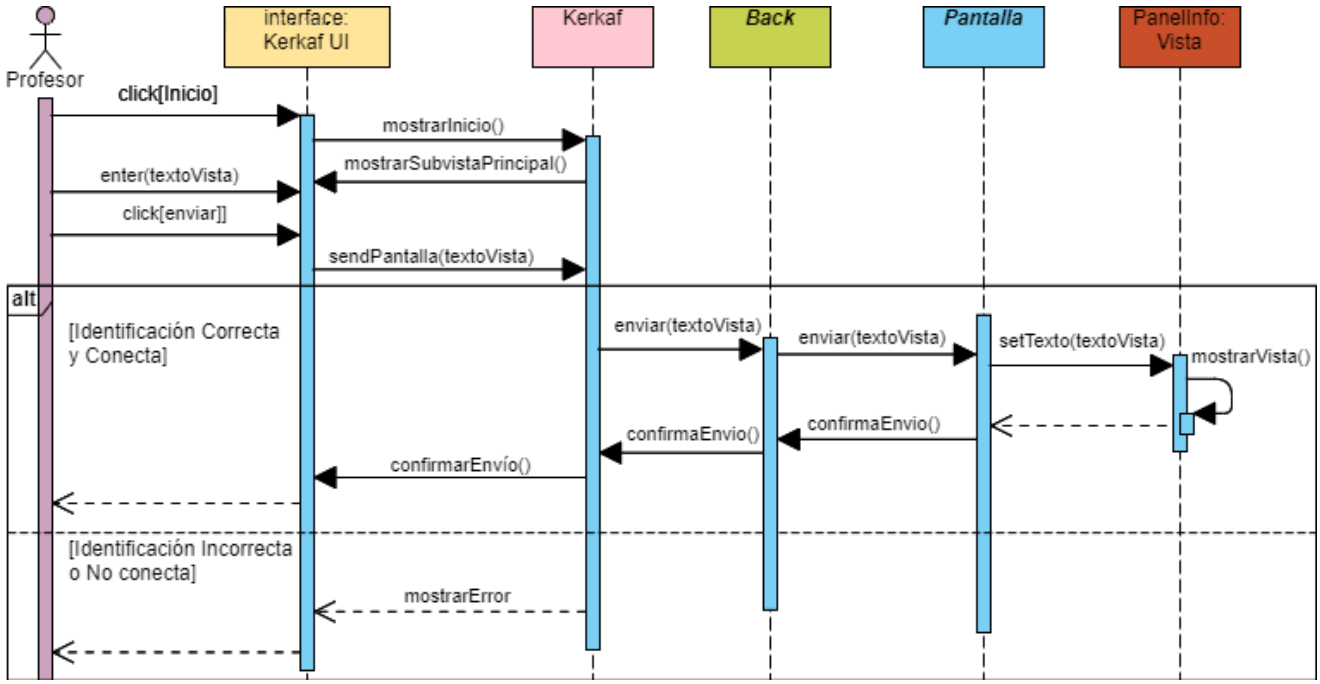

Figura 4.5: Diagrama de secuencia del caso de uso 9

4.5.3. (CU-12) Modificar el acceso a la vista de Tutorías

La modificación de la configuración tiene muchos parecidos con la de envío de datos normales, ya que gran parte del flujo de funcionamiento es exactamente igual. Sin embargo en el lado de ***Kerkaf*** la forma de interactuar visualmente debe ser muy diferenciada.

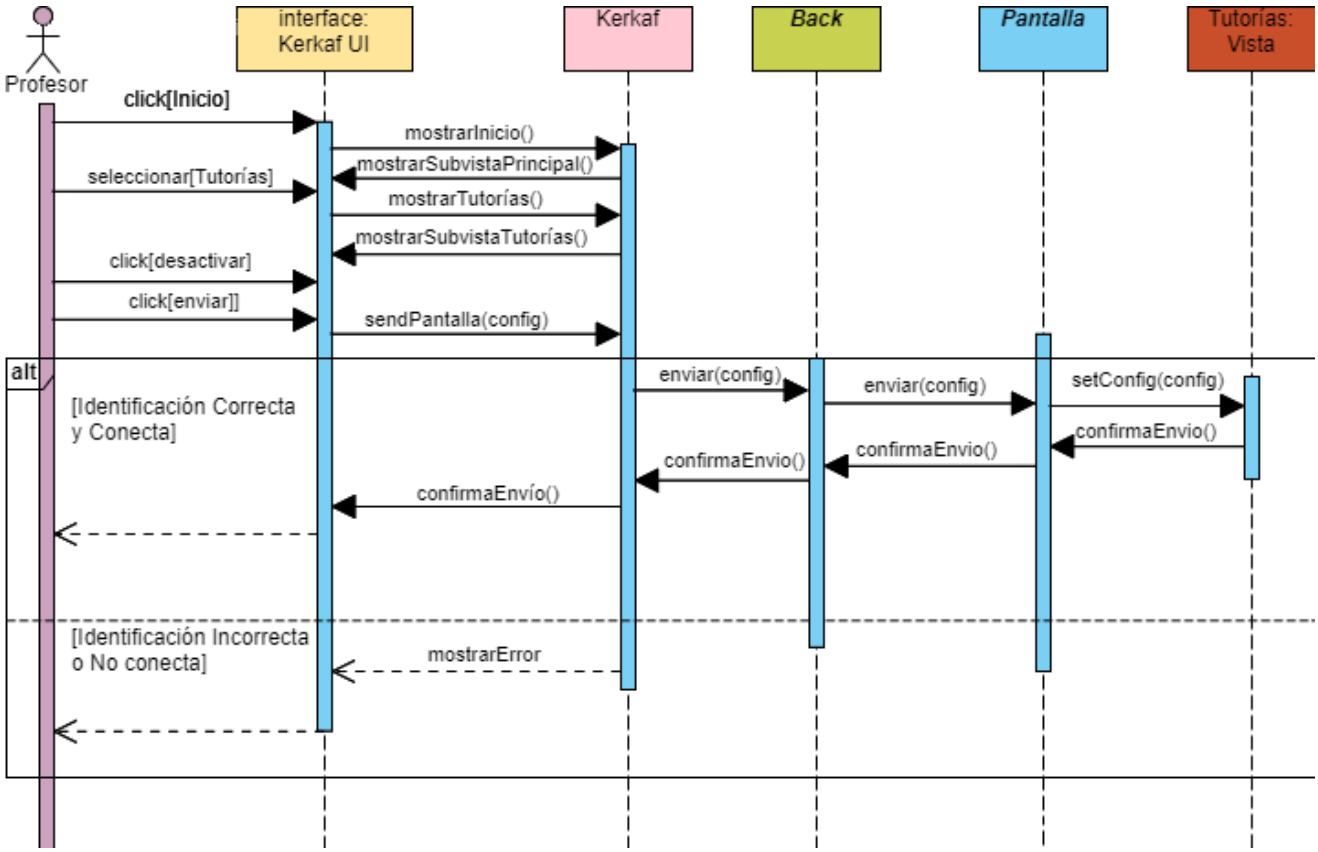

Figura 4.6: Diagrama de secuencia del caso de uso 12

4.5.4. (CU-17) Determinar los datos del Horario:

En otro ejemplo más de envió de datos a la ***Pantalla*** pero requiere especial mención ya que al contrario que en el CU-9 se debe acceder a una subvista diferente de la vista de inicio en ***Kerkaf***.

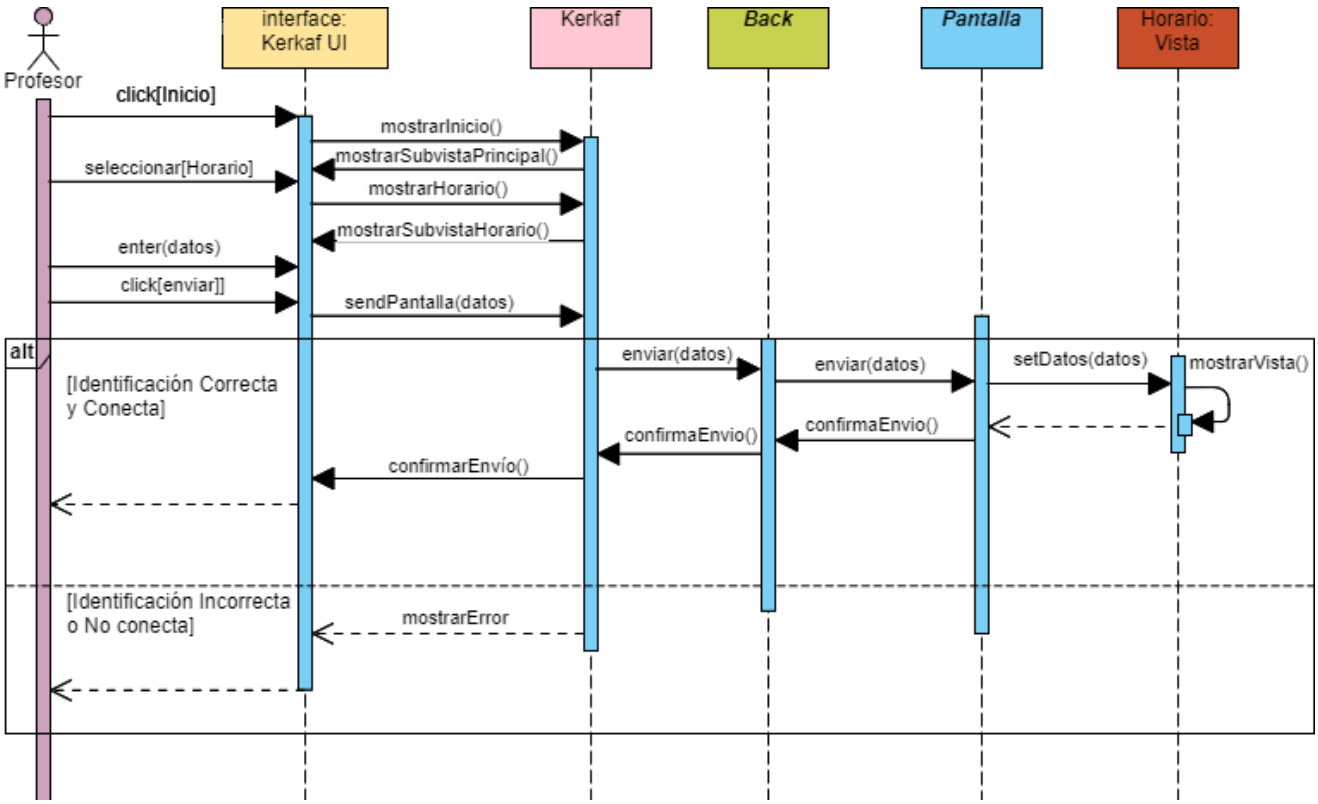

Figura 4.7: Diagrama de secuencia del caso de uso 17

4.5.5. (CU-21) Consultar los datos de la vista de Asignaturas

a continuación se muestra uno de los dos únicos ejemplos de diagrama de secuencia desde el usuario Alumno en el caso de uso de visualizar en el display los datos cargados en la ***Pantalla*** de la vista de Asignaturas, este caso de uso se repite para todas las Vistas de la ***Pantalla*** a las cuales se accede desde el menú.

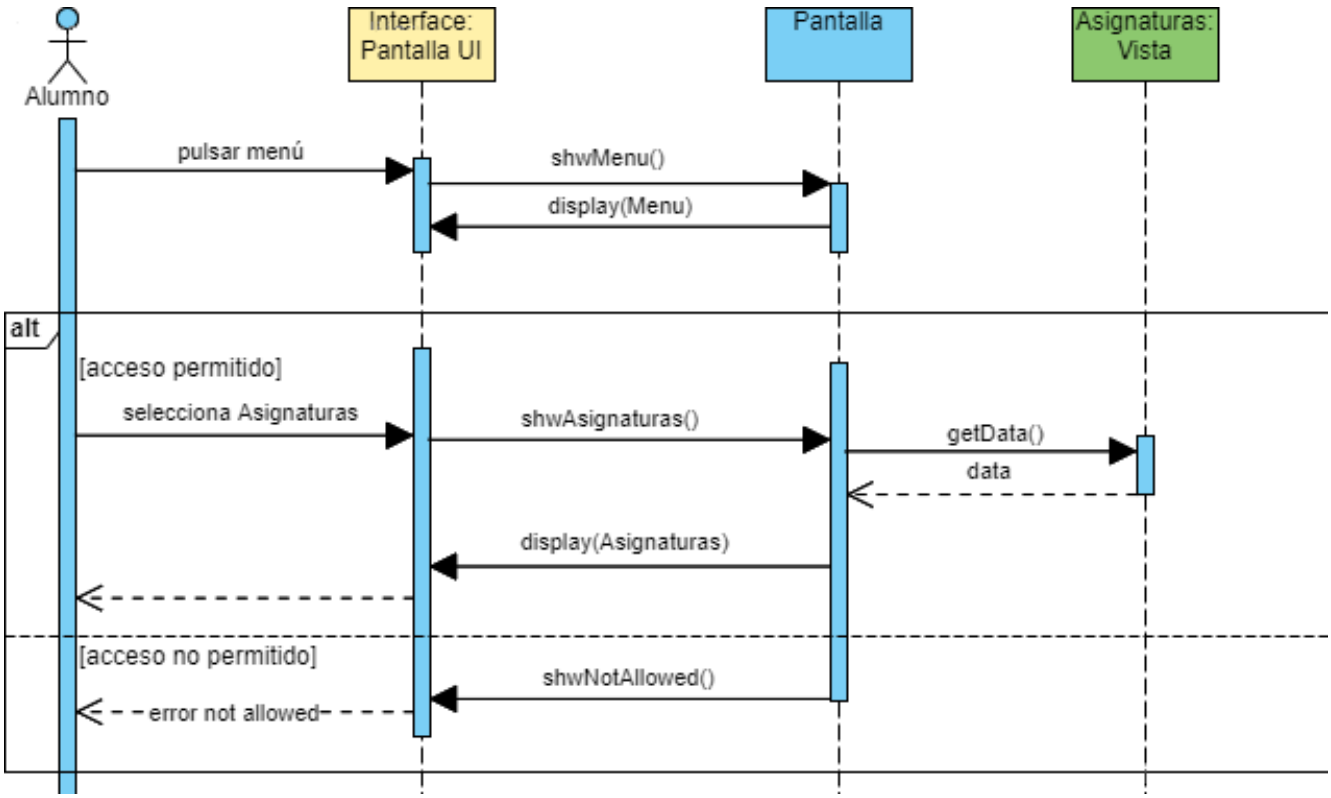

Figura 4.8: Diagrama de secuencia del caso de uso 21

4.5.6. (CU-23) Realizar una Llamada Remota al Profesor

Este caso de uso implica la comunicación a través de un servidor de correo y por lo tanto un elemento externo a la aplicación, sin embargo el diagrama lo muestra como una ejecución concreta del ***Back*** y no una interacción con Kerkaf, ya que no interviene en este proceso mas que de forma indirecta aportando la dirección de correo electrónico adecuada.

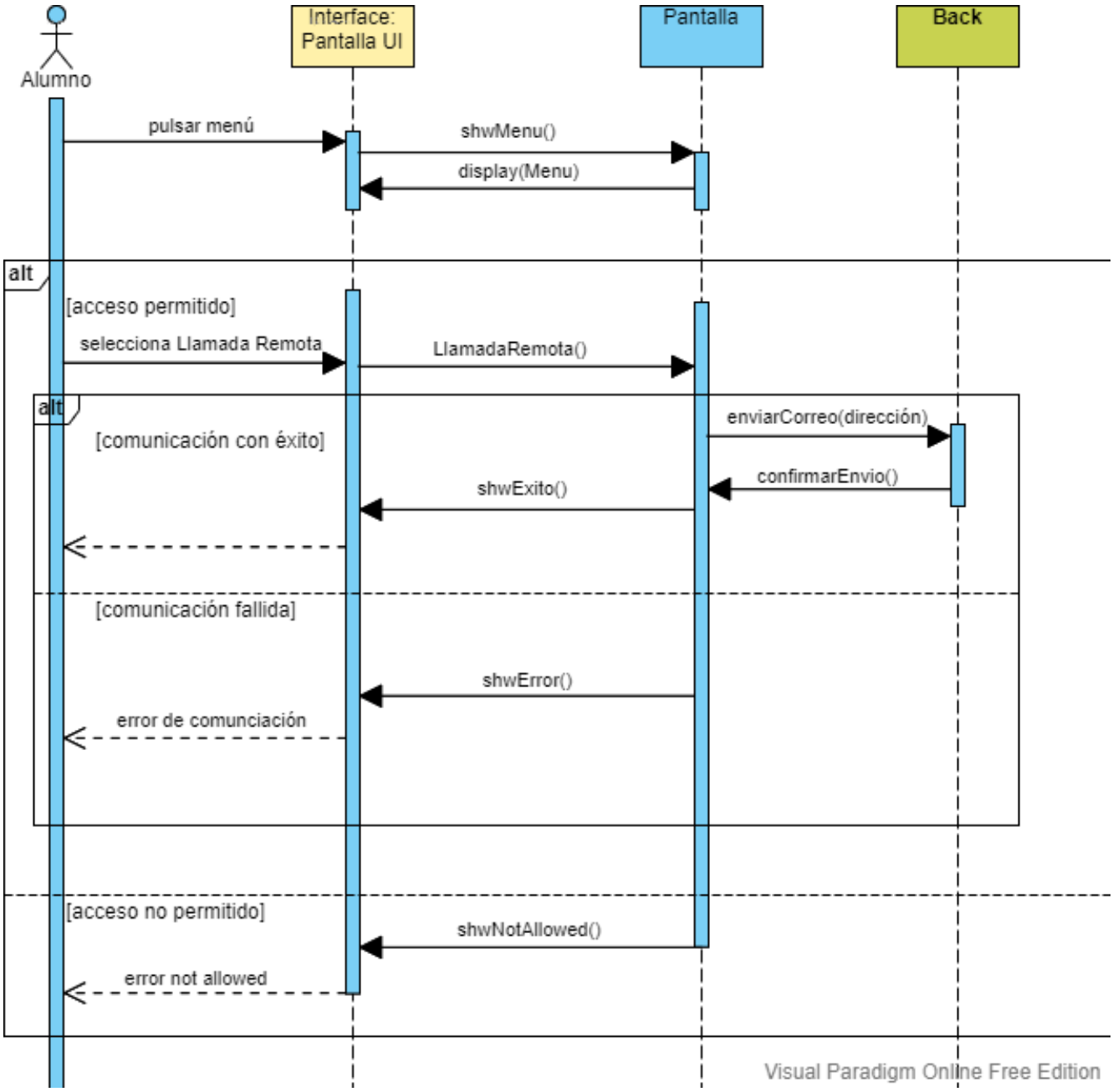

Figura 4.9: Diagrama de secuencia del caso de uso 23

Capítulo 5

Diseño

El diseño de este sistema debe contemplarse en numerosos aspectos y subapartados debido a que aunque si se mantiene una visión general del diseño del sistema al completo, también resulta importante mencionar las filosofías y procesos de diseño que se han utilizado para cada uno de los subsistemas, uno a uno.
Es debido a esto que la estructura de este capítulo desvía ligeramente de lo habitual, sin retirar ninguno de sus apartados clásicos en un documento de este tipo.

Estas variaciones se concentran especialmente al respecto de añadir un subapartado de diseño y arquitectura de diseño lógica específica para la aplicación móvil ***Kerkaf*** y también contar con un extenso apartado al final del capitulo cuya función es describir las tareas de diseño físico del componente ***Pantalla*** el cual como se ha mencionado, posee una interfaz de manejo fabricada en el contexto de este proyecto y cuyo diseño previo conviene mencionar para comprender mejor el proceso de implementación de este mismo componente que se describirá en el capítulo siguiente.

Dejando a un lado estas pequeñas anomalías, el capítulo sigue una estructura sencilla de descripción del diseño general y global del sistema, con su arquitectura física y en función de esta su descripción abstracta mediante un diagrama de despliegue; una descripción de la arquitectura lógica escogida, tanto para el sistema global en su conjunto como para ***Kerkaf***; una explicación del desarrollo de las interfaces gráficas tanto del Profesor (***Kerkaf***) como del Alumno (***Pantalla***) y finalmente un apartado de diseño físico que ahonda en la sección electrónica del proyecto.

5.1. Diseño General

El enfoque del diseño al completo del sistema y prototipo en su conjunto requiere de un enfoque especialmente abstracto, esto se debe a que los requerimientos y funcionalidades así como estructura y protocolos de comunicación son enormemente variados entre los subsistemas que lo componen.
No obstante, a pesar de encapsular diferentes filosofías arquitectónicas, el diseño general del sistema al completo cuenta con la suya propia, que se describe a continuación tanto en su aspecto físico como *software* o lógico:

5.1.1. Arquitectura física

Las condiciones físicas del proyecto han determinado una arquitectura física específica para los componentes ***Pantalla*** y ***Back***. Estas circunstancias concretas y sus respectivas soluciones son las siguientes:

Comenzando por la ***Pantalla*** es clave mencionar que se trata del componente mas alejado del usuario principal (el Profesor) y a su vez el núcleo alrededor del cual orbita todo el sistema; sus circunstancias de uso son las mas críticas y son estas las que imponen el funcionamiento del resto de los componentes. Del componente físico en concreto se requiere un uso en el exterior del despacho o sala donde se encuentre el componente ***Back*** o gestor del servicio. Al no contemplar la posibilidad de taladrar las paredes de la sala donde se instale el sistema, se decidió comunicar este elemento de manera inalámbrica. Concretamente, y debido a sus prestaciones mencionadas en la Metodología, utilizando tecnología bluetooth, con el requerimiento añadido de que el gestor del ***Back*** (la Raspberry) se encontrara a una distancia de no mas de 5-10 metros de la ***Pantalla***.

Por lo que respecta al componente ***Kerkaf***, su requerimiento de uso en la aplicación móvil del Profesor no restringe en distancia ni ubicación física concreta a este componente, salvo el poseer una conexión a internet, que fue finalmente la forma de comunicación con el ***Back*** que se eligió.

Teniendo en cuenta estas circunstancias se han determinado los siguiente elementos diferenciados de la arquitectura física:

Nodos

Como es de esperar, los elementos ya descritos hasta ahora en el documento coinciden con los nodos declarados de la arquitectura física.

- ***Kerkaf***
- ***Back***
- ***Pantalla*** compuesta físicamente por:
    - TTGO placa ESP32
    - Cuadro de mandos (botonera)
    - Display de tinta electrónica

Diagrama de Despliegue

Se puede consultar el diagrama de despliegue en la figura 5.1

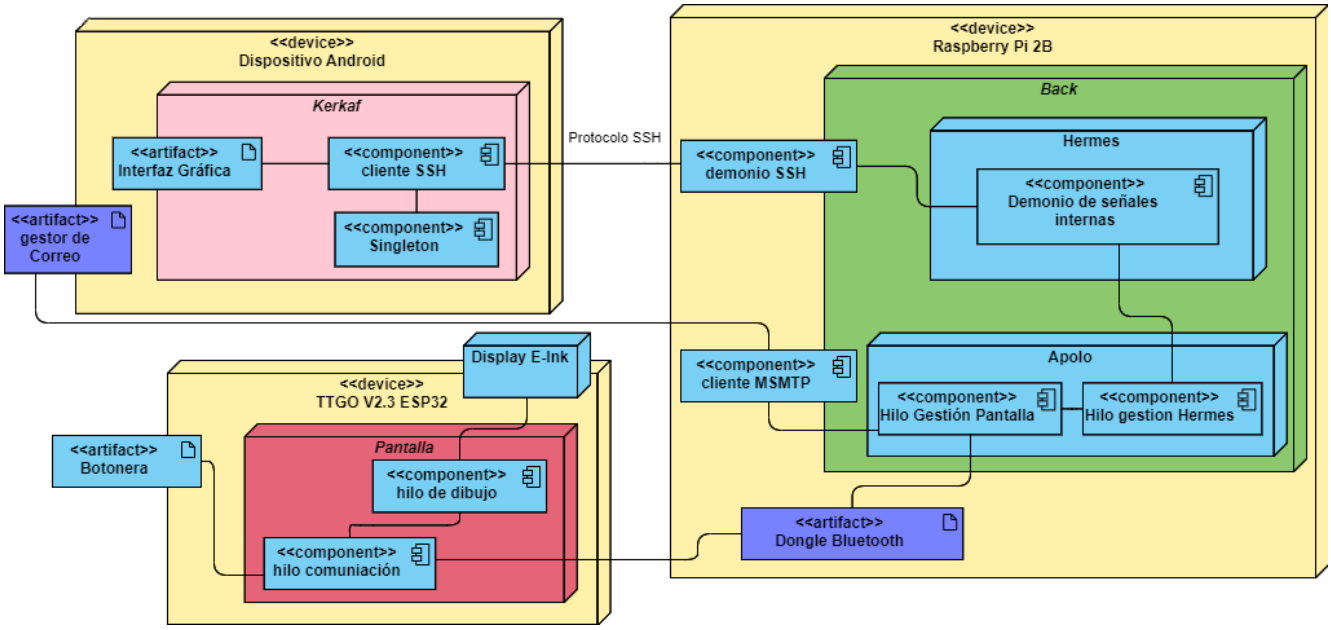

Figura 5.1: Diagrama de despliegue

5.1.2. Arquitectura lógica

Teniendo en consideración la arquitectura física contemplada en el apartado anterior se ha diseñado una arquitectura lógica basada en el modelo Vista-Controlador con su correspondiente subcapa de servicio.

Aunque habitualmente este modelo se explica con mayor facilidad con un sistema de llamadas al servicio REST, en este proyecto no se utiliza la tecnología pero si la filosofía de funcionamiento.

MVC bidireccional

Este modelo de arquitectura lógica consiste principalmente en la separación del componente que interactúa con el usuario del componente que ejecuta las tareas al respecto del caso de uso solicitado por el mismo, manteniendo en todo momento como caja negra para el usuario la capa de controlador y servicio debajo de esta.

El motivo por el cual resulta tan conveniente en un servicio de requerimiento por tecnología REST es que la parte de la vista consulta exclusivamente al controlador y servicio mediante el nombre del servicio del que quiere hacer uso y la información que le envía. Esto resulta especialmente conveniente en cuestión de seguridad, ya que el comportamiento del servicio queda definido de una manera concreta al respecto de los datos que recibe, y no es posible para el extremo del usuario variar este comportamiento al tratarse de una caja negra totalmente.

En el desarrollo de este proyecto se ha seguido esta misma filosofía de funcionamiento separando sus respectivos componentes en las diferentes partes del MVC con la siguiente particularidad:
Dependiendo del usuario con el que se esté tratando en el diseño, la lógica se aplica entre los componentes del prototipo en una dirección o en la contraria. Esto sucede principalmente debido a la particularidad de que un usuario (Profesor) tiene la capacidad de modificar además de consultar los datos del sistema, mientras que el otro usuario (Alumno) únicamente los consulta.

En esto solo se produce la excepción del caso de uso de Llamada Remota, en el cual no se produce una modificación de los datos pero si produce que el controlador deba extenderse de estar contenido en un solo componente físico del prototipo a dos.

La división en usuarios de esta filosofía arquitectónica es la siguiente:

- usuario **Profesor:**
  - **Modelo:** En el caso del Profesor, el modelo sobre el cual actúa el controlador y que contiene los datos a modificar corresponde al componente de la ***Pantalla***, la cual modifica su display cuando la solicitud del usuario Profesor llega a buen término en sus casos de uso descritos.
  - **Vista:** El componente sobre el cual el Profesor consulta los datos del sistema y mediante el cual interactúa con el mismo es ***Kerkaf*** el cual instalado en su dispositivo móvil y mediante sus interfaces gráficas muestra los datos correspondientes.
  - **Controlador:** el ***Back*** o componente *Middleware* ubicado en la Raspberry ejerce la función de controlador ante las solicitudes de la vista que ejerce ***Kerkaf***.
- usuario **Alumno:**
  - **Modelo:** En el caso del Alumno, el modelo en el que se almacenan los datos que consulta este está mayoritariamente alojado en el componente ***Pantalla***, ya que es donde los casos de uso del Profesor los ha almacenado, únicamente la excepción de la Llamada remota altera esta filosofía convirtiendo el extremo a ”modificar” el dispositivo móvil del Profesor.
  - **Vista:** el display de tinta electrónica y la interfaz electrónica de la ***Pantalla*** ejercen como primera capa de interacción y muestra de datos al Alumno, formando así la Vista de la arquitectura.
  - **Controlador:** En lo que a consulta de datos se refiere, lo cual compone la inmensa mayoría de las acciones que el Alumno puede realizar sobre el sistema, el único controlador con el que interactúa este es la ***Pantalla*** en su componente lógico. Sin embargo en el caso de utilizar la Llamada Remota, este controlador debe comprenderse en su aspecto Lógico arquitectónico como extendido al ***Back***, ya que ejerce la funcionalidad de procesar la petición de la Vista y tiene la conexión a internet necesaria para realizarla.

Kerkaf y *Singleton*

Subyacente a la Arquitectura general del sistema, existe una aplicación de esta filosofía software en la creación y funcionamiento del componente Vista en ***Kerkaf***, ya que este, al igual que un navegador a través de su caché o *cookies* almacena datos enviados o recibidos desde sí para que la acción de consultarlos no tenga porque reclamar la acción del servicio, evidentemente esto no se aplica a la modificación de estos, pero sí facilita el funcionamiento de ***Kerkaf*** a la hora de conservar la configuración de conexión al ***Back*** por ejemplo, y también los últimos datos enviados al modelo, permitiendo así saber que dato estamos modificando al hacerlo.

En el capitulo de la implementación en su apartado al respecto de la aplicación móvil Android ***Kerkaf*** se desarrollará mayormente sobre como se ha aplicado esta arquitectura.

5.1.3. Interfaz Gráfica

Al igual que en el caso de la arquitectura lógica, el diseño de las interfaces gráficas se debe realizar teniendo en cuenta que no solamente afecta a dos usuarios diferentes con casos de uso no comunes, sino que además estas pertenecen a componentes físicos diferenciados, por lo que este subapartado abarca dos Interfaces generales, una por usuario, con sus respectivas subvistas y particularidades:

Interfaz de Usuario: Profesor

En esta sección se muestran las múltiples figuras de vistas con el diseño de la interfaz para el usuario en el extremo de la aplicación Android, por lo que su estilo y funcionalidad cumple con las estandarizadas para ese entorno.

- **Vista Inicial**

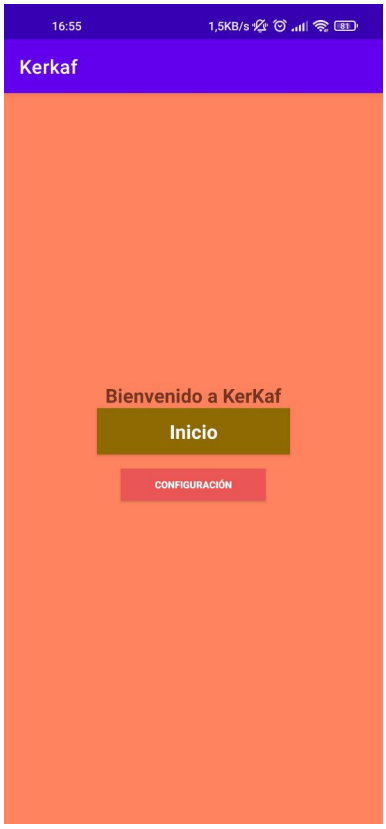

Figura 5.2: Vista inicial mostrada en ***Kerkaf***

- **Panel de configuración: subvista principal**

Figura 5.3: Configuración general de identificación del dispositivo ***Pantalla***

- **Panel de configuración: Llamada Remota**

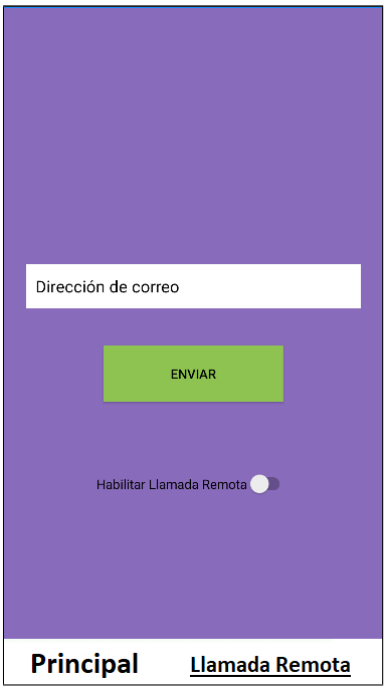

Figura 5.4: Vista de configuración secundaria de la Llamada Remota

- **Vista Principal: Panel de Información**

Figura 5.5: Vista de datos del display principal: Panel de Información

- **Vista Principal: Horario**

Figura 5.6: Subvista de datos de Horario

- **Vista Principal: Tutorías**

Figura 5.7: Subvista de datos de Tutorías

- **Vista Principal: Asignaturas**

Figura 5.8: Subvista de datos de Asignaturas

Interfaz de Usuario: Alumno

En el caso del Alumno, la interfaz gráfica es interactuable únicamente mediante los 4 botones físicos del circuito de control, los cuales son: ”menú”, ”anterior”, ”siguiente” y ”seleccionar”.
Estas interfaces se encuentran representadas en el display de tinta electrónica de la ***Pantalla*** con la característica de ser en blanco y negro y la disposición y resolución pertinentes.
Nótese que todas las imágenes no cuentan con datos cargados, los cuales se muestran en los recuadros de campos de texto de las figuras.

- **Vista principal: Panel de Información**

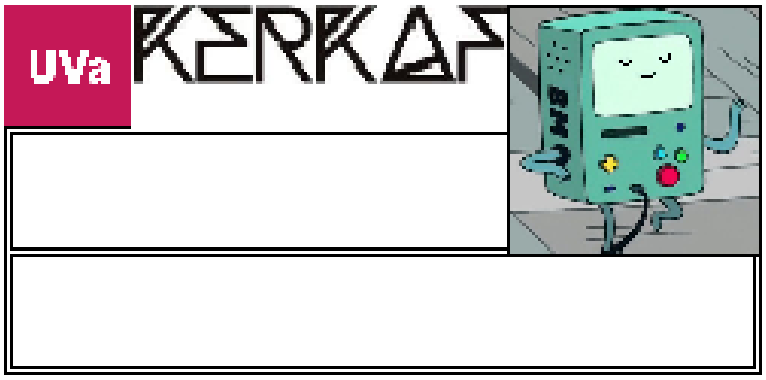

Figura 5.9: Vista mostrada por defecto en la ***Pantalla***

En esta vista hay una imagen de perfil del usuario a elegir, el prototipo utiliza la mostrada en la figura 5.9 que muestra una unidad BMO bailando.

- **Vista Menú**
  Esta vista muestra las diferentes opciones que la ***Pantalla*** ofrece en visualización de datos a las que el Alumno puede acceder

Figura 5.10: Vista del menú de selección de opciones

Las selección de las opciones que se muestran en el menú pueden estar o no disponibles dependiendo de la configuración que haya seleccionado el Profesor, en el caso de que no sea posible acceder la vista muestra la siguiente imagen emergente:

Figura 5.11: Mensaje de error emergente de acceso restringido

Si no el acceso está permitido, las vistas de las opciones mostradas en el menú son las siguientes respectivamente:

- **Vista Horario Clases**
  Esta vista en particular es iterable a lo largo de los cinco días de la semana, variando los datos mostrados y señalando el día de la semana en el campo de texto que en la figura 5.12 se encuentra bajo ”horario”.

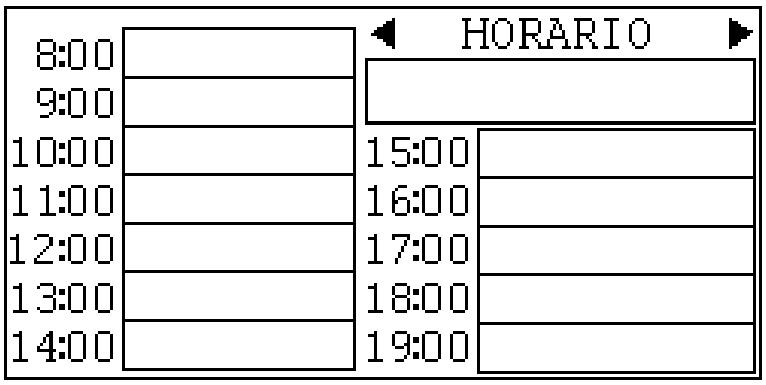

Figura 5.12: Vista genérica del horario de un día de la semana

- **Vista Tutorías**

Figura 5.13: Vista de Tutorías con el código QR de una dirección de correo electrónico

El código QR mostrado en la figura 5.13 es el correspondiente al correo del Alumno, dado que era el prototipo durante su etapa de pruebas.

- **Vista Asignaturas**

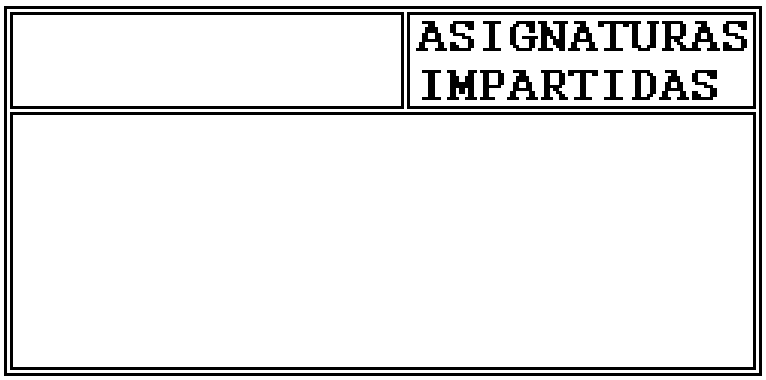

Figura 5.14: Vista de las Asignaturas con sus dos campos de texto disponibles vacíos

- **Llamada Remota**
  Al seleccionar la opción de Llamada Remota el alumno se encuentra con la siguiente imagen emergente mientras la ***Pantalla*** intenta realizar la llamada al ***Back*** 5.15:

Figura 5.15: Imagen emergente al pulsar la selección de Llamada Remota

Si la Llamada falla, la interfaz mostrará el siguiente mensaje 5.16:

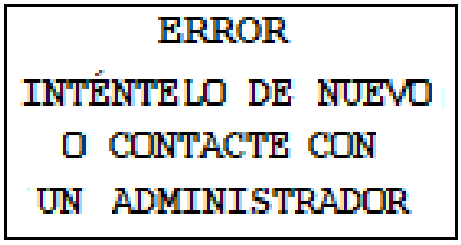

Figura 5.16: Imagen de error emergente al fallar la Llamada Remota

5.2. Diseño Hardware físico

Este proyecto y su prototipo también debe poseer un panel de control físicamente, que interactúe eléctricamente con los pines de entrada y salida de la placa sobre la cual se encuentra el display: la ***Pantalla***.
En este breve apartado de diseño físico se desarrollará sobre el criterio seguido para la realización de este componente y su conexión con la placa TTGO.

Respecto a la interacción física del alumno con la placa se ha decido que, a pesar de que se podría realizar con únicamente tres botones, esta interfaz puede manipularse con mayor comodidad y sencillez

si se colocan cuatro evitando así que un botón comparta dos funcionalidades, lo cual, dado que se trata de un prototipo, podría no ser necesario, aunque no se descarta pasarlo a tres en un modelo definitivo comercial hipotético.

Asimismo, dado el comportamiento observado de la placa a lo largo de su desarrollo (como se explicará con mayor detalle en el capitulo de *Implementación*) La latencia de respuesta de sus acciones, concretamente al modificar el contenido del display, puede ser algo lenta para los criterios de la interacción humana en tiempo real (a pesar de las inmensas mejoras que se han obtenido modificando la biblioteca de gestión).

Teniendo esto en cuenta, se ha decidido colocar un par de diodos LED para marcar mediante su encendido y apagado el tiempo de procesamiento de la placa, permitiendo así al usuario observar como si de un semáforo se tratara, cuando la placa está ”ocupada”.

El segundo diodo LED se utiliza para marcar cuando la Pantalla está en proceso de comunicación inalámbrica saliente, es decir, ”ocupada” hablando con el ***Back*** en el caso de uso de Llamada Remota.

Todas estas especificaciones y motivaciones de diseño se muestran claramente en el siguiente diagrama de representación del circuito electrónico utilizado:

5.2.1. Diagrama electrónico

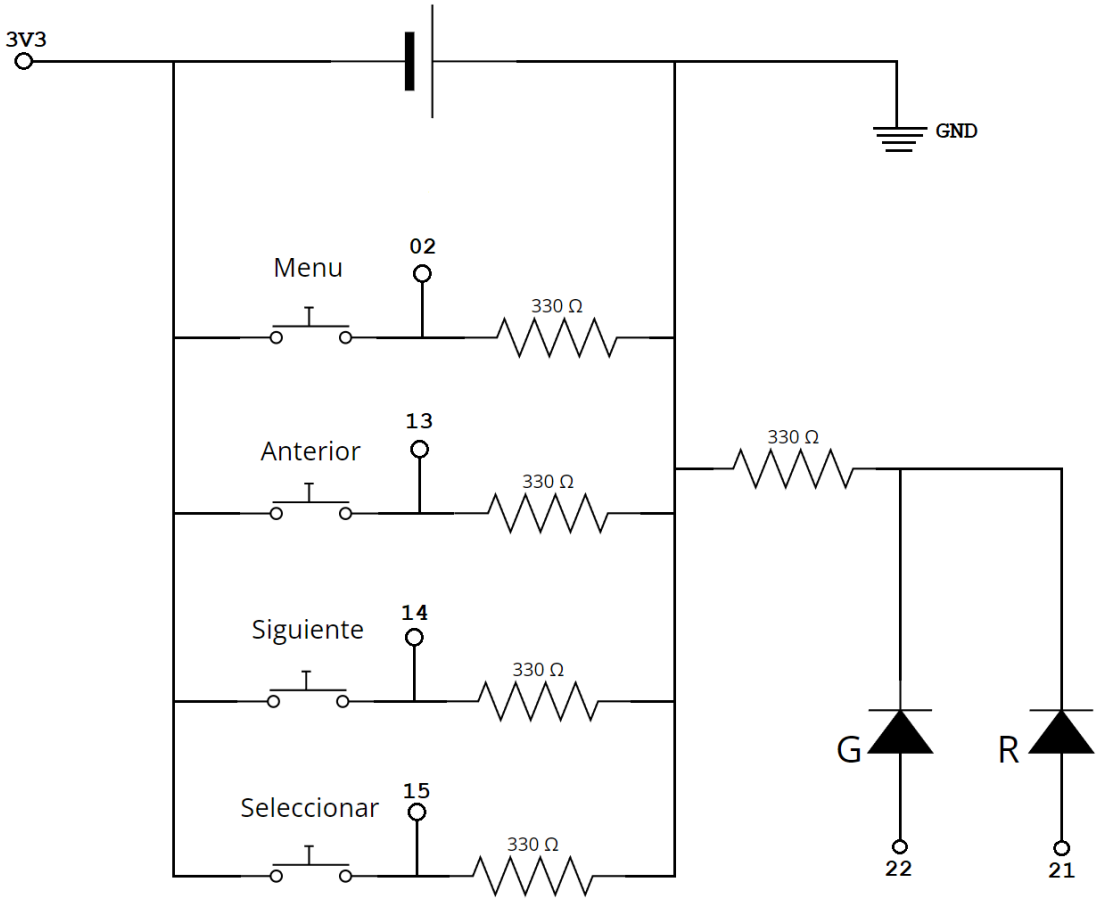

Figura 5.17: Diagrama de representación del circuito eléctrico a fabricar

Para comprender la ubicación de las etiquetas numéricas mostradas en 5.17 como algunos terminales del circuito consultar la figura 6.5, en la cual puede observarse que se utilizan la mayoría de los pines de una sola banda para facilitar su conexión.

Capítulo 6

Implementación

6.1. *Pantalla*

6.1.1. Introducción

Correspondiéndose con la primera etapa del desarrollo del prototipo, la implementación y construcción de ***Pantalla*** y sus subcomponentes supuso con diferencia el mayor de los tres retos y el mas indeterminado en conclusión.
Aunque el entorno Arduino es en realidad una comunidad software de numerosos desarrolladores que pueden echar una mano mediante sus ejemplos publicados en la red, El componente físico para este prototipo resultó ser lo suficientemente exótico como para mantener un tira y afloja con la biblioteca de gestión del display de tinta electrónica, teniendo que cambiar incluso funciones primordiales de la finalmente utilizada: **GXGD** en su subversión para este display **GxGDEM0213B74** la cual se apoya en las bibliotecas de las pantallas mas comunes de **AdaFruit** con pequeñas variaciones.

Para comprender adecuadamente el funcionamiento del programa implementado en la placa, es necesario explicar los conceptos básicos del funcionamiento de un *sketch* Arduino.
Este mini programa que corre indefinidamente en un procesador de este tipo se basa principalmente en tres fases de ejecución: Instrucciones para el compilador, *setup* y *loop*, la primera se ejecuta una sola vez, determinando las bibliotecas y datos globales que el compilador debe tener en cuenta al tratar con el código; la segunda aborda la inicialización de los servicios de la placa y las variables: en general cualquier código previo que se desee ejecutar únicamente una sola vez y, en tercer lugar, el *loop* el cual ejecuta código indefinidamente, en el cual se introducen condicionales que controlan variaciones entre los diferentes servicios establecidos en el setup: cambio de tensión en los sensores, entradas del canal serie, mensajes disponibles en el canal bluetooth... etc.
Con esta breve explicación resultara mucho mas claro el funcionamiento del programa que se describe a continuación.

Debido a las limitaciones de la gestión del servicio bluetooth de la placa y, ante los reiterados problemas de pérdida de datos se llegó a la conclusión de que era necesario un metaprotocolo gestionado por el propio programa que se asegurara de que todas las comunicaciones llegaban completas.
Esto ocasionó modificaciones en el planteamiento original del desarrollo sobre el componente y introdujo indirectamente necesidades hardware mas potentes debido a que los protocolos de comunicación sumados al procesado gráfico ocasionaban tiempos de espera excesivos para la correcta interacción humana con el usuario Alumno.

Por suerte (ya que no estaba previsto utilizarlos) el chip ESP32 con el que la placa cuenta posee dos núcleos en su procesador, y, tras indagar al respecto se encontró también entre las prestaciones de las bibliotecas Arduino un gestor de tareas equivalente a un lanzador de hilos con funciones en ejecución simultánea totalmente diferenciadas.
Para poder comunicar ambos hilos correctamente y sin percances de acceso a memoria se implementó también un sistema de semáforos *mutex* que controlan el acceso a un carrete cíclico de mensaje a procesar. De esta manera fue posible ejecutar la tarea de comunicación inalámbrica a través de uno de los dos núcleos y dejar la tarea de dibujado y procesado de mensajes al otro.

Dado que la comunicación es un evento poco común en frecuencia respecto a las otras prestaciones de la placa, el hilo encargado de gestionarla se puso también al cargo del bucle principal y de la carga de control del sistema.
Todo el código al respecto puede consultarse en la bibliografía accediendo a los repositorios del proyecto: [4]

6.1.2. Hilo de Comunicación y guardado de mensajes

Como se mencionó en la metodología, el protocolo subyacente a la comunicación bluetooth, tanto en este extremo como en el ***Back*** es el de comunicación Serie, y es mediante una librería orientada al manejo del servicio bluetooth de la placa que el programa gestiona las entradas y salidas de datos del canal imitando en prestaciones al servicio clásico de Arduino de *Serial*.

Este hilo además de atender variaciones en la entrada del canal bluetooth, también presta atención a variaciones en los botones de la interfaz del Alumno, por lo que el *loop* de este enorme sketch contiene numerosas comprobaciones:

Source Code 6.1: esqueleto de funcionamiento del *loop*

```
void loop() {
      if(button1.pressed && !boton1Procesando){
            ////////////
      }
      if(button4.pressed && !boton4Procesando){
            ////////////
      }
      if(button3.pressed && !boton3Procesando){
            ////////////
      }
      if(button2.pressed && !boton2Procesando){
            ////////////
      }
      if (SerialBT.available()){
            ////////////
      }
}
```

Estas pequeñas muestras de código sirven para ilustrar lo mencionado en el documento pero en prácticamente ningún caso se correpsponden con la auténtica implementación del programa realizado cuyo

código esta disponible en la bibliografía [4].

Al contrario de lo que se muestra en la figura 6.1 el código completo posee semáforos a cada entrada de comprobación dado que compite con otros dos hilos por la gestión de tanto las *flags* que avisan de que los botones han sido pulsados como del canal de comunicación de salida, ya que en el caso de la Llamada remota es el otro hilo (que se encarga de procesar los mensajes internos) el que lanza la comunicación saliente, encargándose este de la entrante.

Interrupción de los botones:

Para gestionar adecuadamente la comunicación inalámbrica e interna de los hilos, se ha utilizado un gestor de interrupciones adherido a las variaciones del voltaje de los pines correspondientes a la conexión de los botones físicos del panel de control diseñado para el prototipo.

Source Code 6.2: Gestor de interrupciones y función de interrupción

```
//GESTORES DE INTERRUPCIÓN
attachInterrupt(button1.PIN, boton1Pulsado, RISING);

/***
***/

void IRAM_ATTR boton1Pulsado() {
  if(!button1.pressed){
    Serial.printf("Boton1 pulsado (core = %d) : %d veces\n",
    xPortGetCoreID(),
    button1.pressed = true;
  }
}
```

Como se observa en la figura 6.2 simplemente con la función ”attachInterrupt” podemos ligar la atención inmediata del hilo a cualquier tipo de estímulo o señal que la placa sea capaz de detectar.
En sistemas mas sofisticados estas capacidades del chip se utilizan para mantener las placas en un estado de continua hibernación, encendíendose solamente ante estímulos de este estilo a traves de gestores como el utilizado aquí.

Protocolo de comunicación externa:

Como se ha mencionado anteriormente, la comunicación bluetooth, al ser poco fiable, ha provocado el desarrollo de un protocolo de control de los mensajes, asegurando principalmente que ninguno de los dos extremos (***Back*** o ***Pantalla***) comiencen a enviar información si el otro no está disponible para escucharla. Esto dio lugar al surgimiento del problema de los dos Generales [3] aunque la solución tomada ha resultado diferente en formas, pero no en filosofía. En este caso, en lugar de marcar ordinalmente los mensajes enviados para que el receptor pueda saber si le falta alguno, se sigue un ritual de comunicación en el que ambos participantes saben cual es el que tiene que ser el primer mensaje, el segundo, y así hasta que se complete el ritual, formando parte de estos mensajes predefinidos cierto códigos inspirados en propio protocolo de comunicación serie y su significado numérico en ASCII.

Entre medias de estos códigos se encuentra el mensaje de texto que se desea enviar, tanto el ”ritual” de comunicación como la estructura del mensaje de texto se describen a continuación:

-Ritual de Comunicación:

En lo que respecta a la comunicación entre el ***Back*** y la ***Pantalla*** se sigue el siguiente orden de mensajes intercambiados entre el emisor y receptor al enviar un mensaje de texto:

- 1º — (emisor): Al comenzar la comunicación el emisor envía por el canal un código RTS *request to send* (5), que queda a la espera (mediante timeout antes de volverlo a intentar) del siguiente mensaje.
- 2º — (receptor): al recibir un código RTS el receptor, si ese es el caso, envía un código CTS *clear to send* (6) informando de que está listo para recibir el mensaje.
- 3º — (emisor): tras estos dos pasos que aseguran que no se pierda la información, el emisor envía la cadena de texto al completo
- 4º — (emisor): al completar la transferencia, por último, el emisor envía un código EOT *end of transmission* (4) que marca que ha terminado.
- 5º — (receptor): al recibir el EOT, el receptor guarda el mensaje para ser procesado y envía el código final que marca la finalización de la comunicación, BELL(7). el cual al recibirse por el emisor, este da por procesada la comunicación correctamente y retorna felizmente a sus otras tareas.

Este pequeño ritual de pocos pasos y 4 códigos, aunque tedioso, asegura a la perfección que de no producirse correctamente la comunicación, el sistema emisor sea consciente de ello y con esto el usuario, ya que a no ser que toda la comunicación y procesado se produzcan adecuadamente, el sistema no finaliza con el BELL y por lo tanto el emisor reiterará sus intentos de realizar la comunicación hasta que el timeout o los intentos pongan de manifiesto que hay un error en el canal y esta comunicación finalice infructuosamente.

-Formato de los mensajes:

La comprobación (indirectamente) de la integridad de la comunicación no finaliza ahí, el mensaje transmitido, y esto afecta a todos los componentes del prototipo, mantiene el siguiente formato de cadena de caracteres con su longitud en bytes:

longitud de la cadena de texto + ’ ’	clave de un solo carácter + ’ ’	cadena de texto

El primer campo contiene un número (en ASCII, es decir, como texto) seguido de un espacio en blanco (también ASCII). Este número indica la longitud de la cadena de texto, que podría decirse que son los argumentos de la función.
El segundo campo marca la función en si, contiene una clave en forma de un solo carácter seguido de un espacio en blanco la cual determina para que debe usarse esa cadena de texto en el lado del receptor, que, en el caso de la ***Pantalla*** podría indicar por ejemplo que la cadena de texto es el mensaje que debe ubicarse en el cuadro de texto primero de la vista Principal (Panel de información).
Finalmente, como ya se acaba de explicar, el tercer campo del mensaje de longitud variable sirve para almacenar el contenido del mensaje propiamente dicho, y su longitud es comprobada en el lado del

receptor comparándola con la indicada en el primer campo antes de devolver el código BELL que indica que la transmisión se ha producido exitosamente.

Al llegar estos mensajes a la ***Pantalla*** son colocados en un buffer circular cuyo índice de escritura se desplaza, marcando así al otro hilo que hay mensajes pendientes de procesar, este protocolo también se utiliza para gestión interna de comunicación entre los dos hilos aunque no implique la llegada de un mensaje comunicado inalámbricamente, como por ejemplo al pulsarse un botón.

6.1.3. Hilo de dibujado y gestión de procesado de mensajes

Como se ha mencionado en la introducción, para asegurar la correcta disponibilidad del prototipo, las tareas de dibujado en pantalla y comunicación entrante se han separado en dos hilos diferentes. El tratamiento de la comunicación entrante, el cual incluye la interacción física del Alumno con el panel de control, esta a cargo del hilo sobre el que se acaba de desarrollar. El procesado de estas instrucciones y el dibujado, así como la comunicación saliente quedan a cargo del segundo hilo, denominado hilo de ”dibujado” o ”dibujo” y sus tareas se describen a conntinuación.

Procesado de Mensajes

La programación de una tarea específica para un hilo en el entorno Arduino pasa por otogarle el control de un bucle infinito al estilo del que maneja el hilo principal con su *loop()* aunque en este caso este bucle debe colocarse en el interior de una función que se liga al core en la fase de *setup()* del principal, en cuyas preparaciones iniciales se incluye ests funcion de ligamento:

Source Code 6.3: función de anclaje de una tarea al core

```
      xTaskCreatePinnedToCore(
          ProcesaMensaje,   /* Task function. */
          "ProcesaMensaje", /* name of task. */
          10000,            /* Stack size of task */
          NULL,             /* parameter of the task */
          5,                /* priority of the task */
          &procesa,         /* Task handle to keep track of task */
          0);               /* pin task to core 0 */
```

En otro apartado del programa se define la función a ejecutar que tiene en su interior el bucle infinito con el que este core trabajará indefinidamente localizando la existencia de un mensaje nuevo en el buffer cíclico.

Source Code 6.4: función ejecutada en el segundo core

```
void ProcesaMensaje(void * pvParameters) {
  Serial.print("ProcesaMensaje running on core ");
  Serial.println(xPortGetCoreID());
```

```
  for (;;){
    vTaskDelay(1);
    xSemaphoreTake(xMutex, portMAX_DELAY); //mutex para no tropezar
    if (currMen != currLec) {//indices de escritura y lectura buffer
      xSemaphoreGive(xMutex);
      digitalWrite(LEDROJO, HIGH);
      Serial.printf("procesado el mensaje %d,//
      que contiene la clave %c\n", currLec, Mensaje[currLec].clave);
      switch (Mensaje[currLec].clave){
          case ('A'):{
            copiaCampo(vista1.texto1, currLec);
          }break;
          case ('B'):{
            ////////////////////////////
          default:{
            Serial.printf("esto que es: %c\n",//
            Mensaje[currLec].clave);
          }
      }
      currLec = (++currLec) % NUM_MENS;
      digitalWrite(LEDROJO, LOW);
      }
      xSemaphoreGive(xMutex);
    }
 }
}
```

El funcionamiento de este código es muy simple: al evaluar la posición de los índices de escritura y lectura en el buffer circular (”carrete”) se detecta cuando el otro hilo a colocado una tarea que se debe procesar, a continuación se analiza el contenido de la clave de ese mensaje ubicando así la función que se desea ejecutar con la cadena de texto del mensaje.
En la mayoría de los casos se copia su contenido al campo de texto y vistas que corresponda según la clave utilizada, colocando así en el modelo la información que el usuario Profesor en el otro extremo a dos dispositivos y cuatro comunicaciones de distancia ha enviado.

Algunas de las claves, no obstante, interactúan con lo que la pantalla debe mostrar, introduciendo así el flujo del programa y su hilo en la tarea de gestionar el display y modificar la interfaz del Alumno, esto sucede de la siguiente manera:

Gestión y control del display

La gestión del display de tinta electrónica se confió con cierto optimismo a las funciones presentes en la biblioteca GxGD en su versión específica de este tipo de pantalla y resolución de ”GxGDEM0213B74.h”. Sin embargo, a medida que las pruebas iniciales daban paso a intentos de implementación de procesos mas complejos quedo claro que no iba a resultar adecuada tal y como se encontraba.

Problemas de baja frecuencia de refresco:

Los principales tropiezos surgieron al intentar hacer uso de la prestación de *partial refresh* o ”refresco parcial” de la pantalla, ya que la forma que figuraba en los ejemplos: *display.update()* realizaba un refresco parpadeante lento, orientado a recolocar las cargas entintadas del display lo máximo posible asegurando que la imagen se muestre con claridad.

Sin embargo, según el fabricante de la placa, esta disponía de una manera rápida de carga de imagen, mas adecuada para la interacción en tiempo real, esta carga rápida se realizaba utilizando un doble buffer de memoria.
Este doble buffer permitía (en teoría) un procesado de los datos de una mitad a la tinta mientras el otro hilo gestionaba la carga de las imágenes o texto a la otra. Esta prestación no era necesaria para nuestro proyecto dado que en el momento en el que esto se estaba intentando aún no se conocía la existencia de este segundo núcleo y de hecho, como se describe a continuación, la propia biblioteca no estaba preparada para ello.

Tras mucho indagar en GxGDE, se descubrió que al usar *partial refresh* utilizaba alternadamente uno de los dos búferes para cargar la información que se le indicaba en el programa de ***Pantalla***, pero, tras copiar su contenido al otro (mediante una instrucción de alta eficiencia del chip) se ejecutaba un borrado de datos en un segundo refresco parcial no previsto, dejando el display sin la imagen cargada.
Esto resulto ser un mayor problema de lo previsto y la adecuada modificación de las entrañas funcionales de esta biblioteca no fue gracias a su sencillez, sino a pesar de ella. Los problemas derivados del uso inadecuado de la biblioteca *Adafruit* provocaron que de facto el alumno tuviera que vérselas con la gestión mas básica de la carga de datos píxel a píxel y el funcionamiento de la memoria implicada en ello.

Finalmente, se descubrió que el problema era que este borrado de datos, apuntaba a una dirección de memoria equivocada: La del buffer cargado en ese momento.
El proceso de borrado, vinculado a un nuevo refresco parcial del display, colocaba en la tinta el contenido de la memoria del anterior buffer, por lo que la imagen deseada se perdía.

Cuando finalmente se comprendió este defecto de la biblioteca (cuyo uso está espectacularmente extendido teniendo en cuenta estos fallos) la modificación fue sencilla e inmediata: Se anuló la necesidad de este segundo refresco parcial y borrado de datos, ya que de todas maneras no se hacía uso de un segundo core en la tarea de dibujado, por lo que las prestaciones del doble buffer resultaban inexistentes.

Este cambio no solo resolvió el problema, sino que además resolvió la problemática mas acuciante del desarrollo: la imposibilidad de realizar una interacción en tiempos de respuesta humanos adecuados, debido a su lentitud.

6.1.4. Soldadura del circuito de control

En cuanto a la interfaz de su usuario directo (Alumno) como ya se ha mencionado se fabricó un panel de control de cuatro botones y dos leds, cuyo diagrama se encuentra en la figura 5.17.
El proceso de creación de este sencillo circuito pasó por su implementación y pruebas en una placa protoboard, una de las versiones intermedias de este circuito en el que solo se probó con un botón puede observarse en la figura 6.1

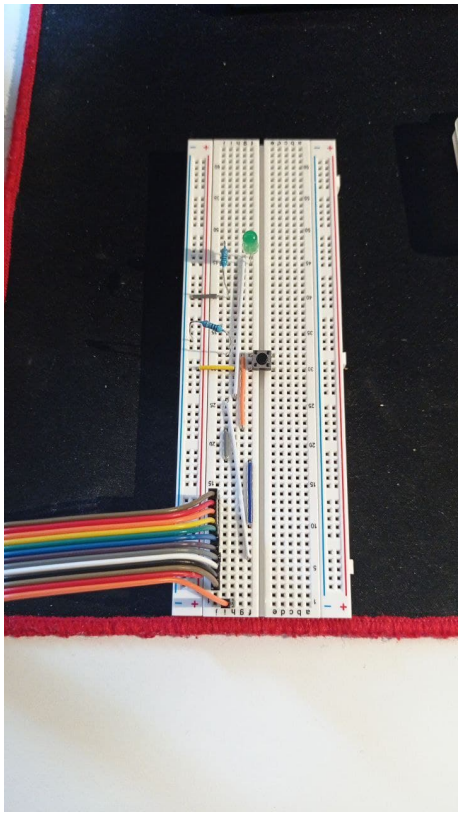

Figura 6.1: Primeros testeos de la protoboard

Desgraciadamente no se tomó ninguna fotografía del circuito al completo en la protoboard, pero no mucho después de superar las pruebas de diseño, se procedió a crear un circuito con mas solidez.

Haciendo uso de la estación de soldadura especificada en Metodología, se procedió a colocar en la placa de fibra de vidrio de impresión de circuitos todos los componentes necesarios en una disposición tal que permitieran conectarlos entre sí cómodamente. También se realizó la soldadura de una batería de litio a la placa para poder utilizarla inalámbricamente figura 6.3

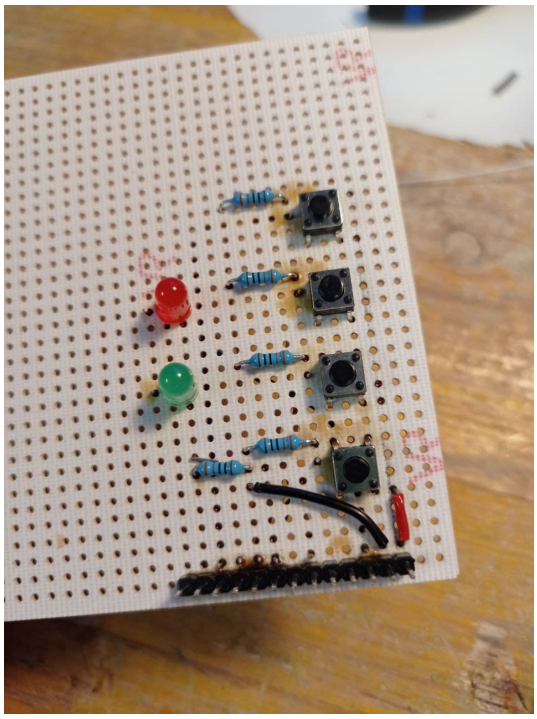

Figura 6.2: Disposición de los componentes en la placa

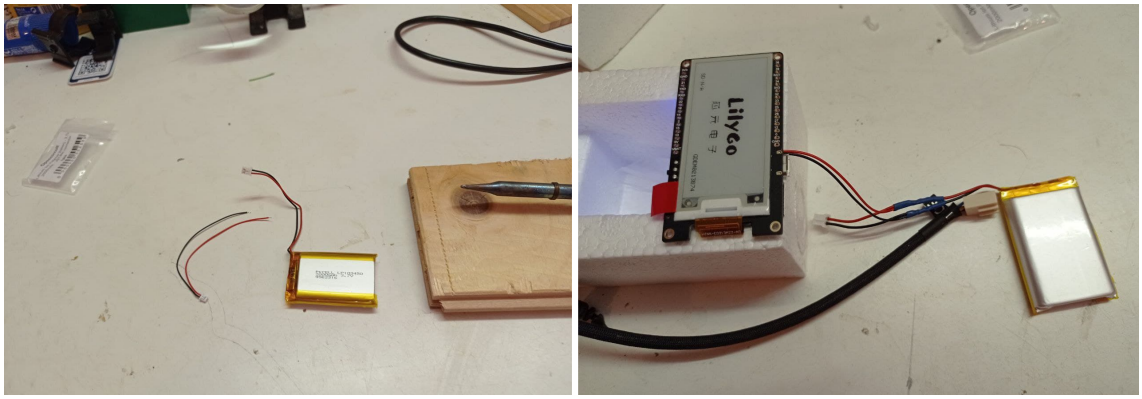

Figura 6.3: Instalación de la batería de la placa

Como se puede observar en la figura 6.2 hay una figura negra con numerosos pines metálicos en la parte baja de la placa, esta conexión es un añadido al circuito que le permite conectarse a los pines de la propia ***Pantalla***.
Estos pines en la propia placa TTGO no llegaron soldados de fábrica, sino que hubo que soldarlos a posteriori y, aunque se realizaron numerosas pruebas sin haberlos soldado por temor a estropearlo, finalmente fue posible soldarlos gracias a la inestimable ayuda y excelente habilidad del Catedrático del departamento de Electromagnetismo José María Muñoz Muñoz en una visita a la facultad de Ciencias, donde tuvo a bien echar una mano al proyecto.

Una vez colocados en su sitio, se soldaron los componentes con buen resultado:

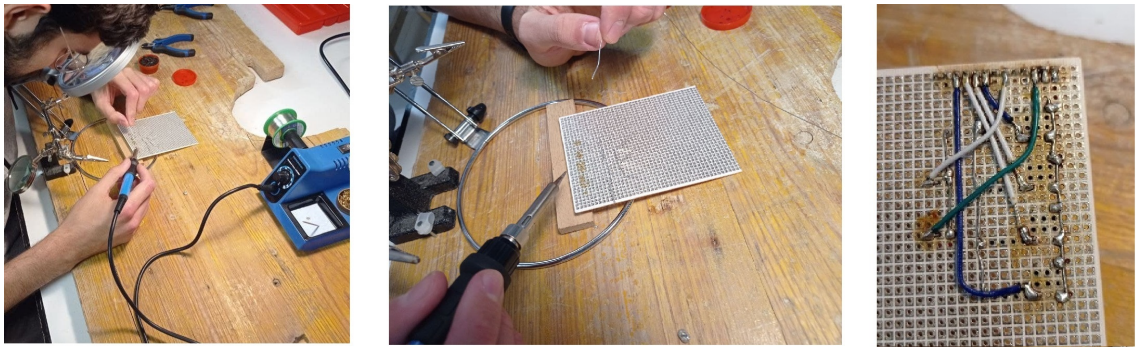

Figura 6.4: Proceso de soldado de la placa de control

Para comprender mejor la estructura de entrada de las señales digitales en la ***Pantalla***, en la figura 6.5 viene indicado en un diagrama la distribución y utilidad de sus entradas (correspondiente con los pines de las imágenes del circuito en vivo).

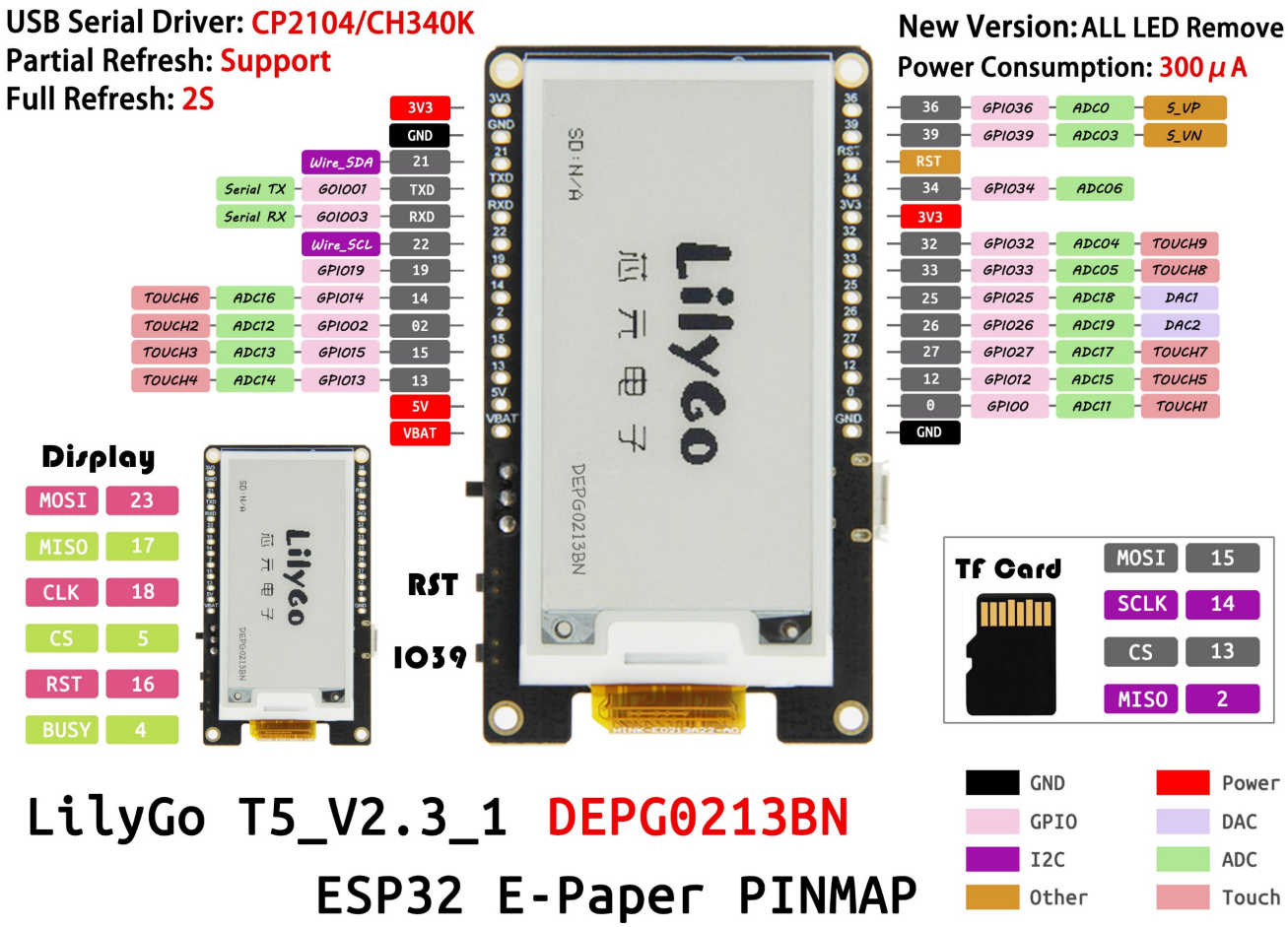

Figura 6.5: Diagrama de especificación de pines de entrada de la TTGO (o lilygo)

6.2. *Back*

6.2.1. Introducción

El *Back* o controlador y servicio ejerce la función de realizar procesar las instrucciones provenientes de la *Vista*. En el caso de este prototipo, dicho servicio es, principalmente, la correcta transmisión de un extremo al otro del sistema en su conjunto, dado que el procesamiento de la mayoría de las acciones conlleva guardar datos en la ***Pantalla***, a pesar de ello, en la Raspberry, que es donde se encuentra el componente ***Back*** también se producen acciones de servicio directas, que no solo conciernen a la correcta transmisión de los datos, vease la Llamada remota, que en este punto del sistema es donde se envía un correo electrónico mediante el msmtp, o la modificación de la dirección de correo electrónico, información que se procesa en este elemento y no alcanza la ***Pantalla***.

6.2.2. estructura

En la figura 6.6 de muestra el contenido en directorios y ficheros de todo el componente Back, tal y como está almacenado en el repositorio *git*.

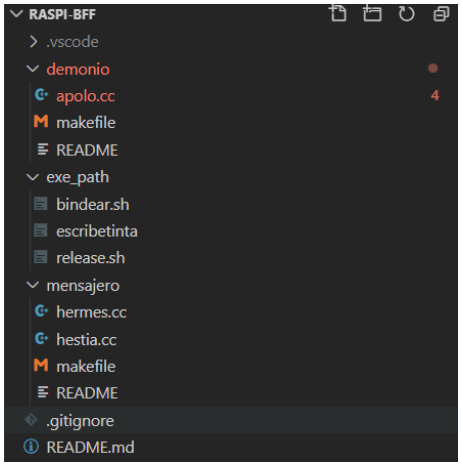

Figura 6.6: Estructura de ***Back***

A mencionar en la imagen está el pequeño script ”escribeTinte” de pruebas iniciales que aún reside en el repositorio (por intereses arqueológicos). Este script tenía la única y sencilla función de Mandar un texto bruto por el canal bluetooth.
El resto de scripts de ”exe_path” tienen la finalidad de configurar el canal de comunicación como se explicará mas adelante y los directorios de mensajero y demonio corresponden a los programas de comunicación con ***Kerkaf*** y ***Pantalla*** respectivamente.

6.2.3. Comunicación con *Pantalla*

La comunicación por bluetooth es el principal y mas grande problema a resolver al cual se enfrenta el ***Back***, como se ha explicado anteriormente, uno de los mayores desafíos emergentes e imprevistos del proyecto fue la baja fiabilidad de la comunicación bluetooth y que, debido a esta baja fiabilidad, los componentes que forman el sistema no conseguían verificar que la acción de su caso de uso se hubiera completado.
Para solventar este problema, como ya se ha explicado en la sección de la ***Pantalla*** de este capítulo, se utilizó el protocolo descrito en 6.1.2 cuya implementación aquí no supuso mayor problema o diferencia (teniendo en cuenta la diferencia de entorno) salvo por las siguientes particularidades:
Estas prestaciones y tantas otras referentes a la comunicación y gestión de mensajes de la ***Pantalla*** se colocaron el un programa nombrado **Apolo**, que se describe a continuación:

Servicio demonio: *Apolo*

Para empezar, el servicio de comunicación debía encontrarse disponible en todo momento y no ser interrumpido, por lo que debía tratarse de un demonio, y así se programó.
A continuación, este demonio, con un hilo dedicado a la comunicación con ***Pantalla*** dentro del programa, establecer el envío y recepción de mensajes vía bluetooth, por lo que se habilitó mediante el comando **UNIX** de *rfcomm* un ligamento entre la conexión bluetooth con el dispositivo emparejado y un fichero de el directorio de comunicación **UNIX** `\dev`.

Source Code 6.5: script shell de montado de la conexión bluetooth

```
#!/bin/bash

sudo rfcomm bind /dev/rfcomm1 $1
sudo stty -F /dev/rfcomm1 -echo
```

Este script, invocado desde el programa mediante *system()*, colocaba en el primer argumento *$1* la dirección MAC del dispositivo bluetooth, en este caso la ***Pantalla***.
La transformación del canal de comunicación con el dispositivo bluetooth, permitía de facto que el programa se comunicara con la ***Pantalla*** como si de un fichero se tratara, con la configuración encapsulada de un protocolo de comunicación Serie, cuyas opciones por defecto había que modificar, ya que de normal, cualquier comunicación serie del sistema (como lo es la propia terminal) tiene un eco de los caracteres emitidos, y en el caso del protocolo y ritual de comunicación, no resulta conveniente. De ahí la 4 linea del script.

El código de este programa, **Apolo** gestionando la comunicación a través de este canal está disponible en los repositorios del proyecto a libre y pública consulta de quien lo desee. Pero es importante aclarar una serie de aspectos y líneas generales de su comportamiento:

Este programa está en realidad dividido en varias funciones de ejecución continua y simultánea, de forma similar a lo explicado sobre la ***Pantalla*** ya que entre las tareas que se ejecutan en **Apolo** se encuentra la de atender al mismo tiempo el canal de comunicación bluetooth (*/dev/rfcomm1*) y las órdenes que emita ***Kerkaf***. De estos dos requisitos primordiales surgieron las preguntas iniciales del ”como” y el ”cuando”: El ”como” hace referencia a la manera mediante y el ”cuando” a como avisar al programa de que tiene tareas que realizar.

La forma mediante la cual se hace llegar la información de un punto a otro; en el caso de la ***Pantalla*** ya se ha explicado (rfcomm), pero para ***Kerkaf*** la solución requiere un poco mas de esfuerzo:
Para empezar es necesario atender la comunicación a través de internet de la aplicación móvil, para esto se utilizó la ejecución remota de comandos shell a traves del protocolo SSH, creando así a **Hermes** cuyas funciones mas específica se describen la sección siguiente, por el momento basta con entender que toma el recado y nos lo coloca en una memoria compartida.
A continuación debemos elegir de que manera se crea y se lee esta memoria compartida entre dos programas, una posibilidad era la de un fichero de común consulta, pero resultaba un canal algo tedioso para manipular simultáneamente entre dos programas.
Finalmente se utilizó la herramienta **UNIX** de POSIX *shared memory* que permite crear de facto estos ficheros compartidos, pero que en la práctica se manipulan como punteros de memoria habilitados entre los programas e identificados mediante una *label*.

Una vez resuelto este ”como” solo quedó el ”cuando” que resultó ser una pregunta más difícil: De que manera avisamos tanto de un hilo a otro, como de un programa a otro de que este tiene tareas pendientes o acciones a realizar.
La respuesta tras mucho investigar entre las herramientas a disposición del entorno *Raspbian* **UNIX** fue la de **gestión de señales**. Esta forma de comunicación entre procesos resulta especialmente práctica en algunas de sus versiones mas modernas, que permiten gestionar y transmitir información compleja a través

de decenas de códigos de señal posibles. Y no solamente para comunicación entre procesos existentes, si no que además, es posible habilitar un ”vigilante” que se encargue de monitorizar la entrada, salida, encendido o apagado de canales de comunicación, ficheros... etc, facilitando enormemente el control de la comunicación con la **Pantalla**.
Con esta potente herramienta disponible, sencillamente hubo que otorgar a un hilo o hilos concretos la tarea de ejecutar una función de comunicación (con su ritual) a la llegada de la señal establecida, y así procesar el mensaje y actuar en consecuencia, ya fuera por el canal bluetooth, o por la memoria compartida con **Hermes**.

6.2.4. Comunicación con *Kerkaf*

Una vez comprendido el núcleo característico del programa demonio **Apolo** toca mencionar también las otras tareas circundantes, es decir, la comunicación con ***Kerkaf*** mediante SSH cuyo control de mensajes corre a cargo del programa **Hermes** que también se ejecuta en este entorno de ***Back*** en la propia Raspberry:

Controlador: *Hermes*

Hermes es el programa ejecutador remotamente mediante SSH cuyos argumentos son tratados y almacenados en el formato de comunicación entre elementos del sistema: longitud, clave y cadena de texto. Es en este punto de la comunicación donde se mide la longitud de la cadena, ya que la comunicación SSH cuenta con sus propios protocolos en un contexto de comunicación TCP que asegura prácticamente de forma total la integridad de los mensajes enviados.

Una vez procesado este mensaje contenido en los argumentos de la ejecucíon del programa, **Hermes** crea una memoria compartida cuya label ya ha sido determinada de antemano, y envía una señal al proceso Apolo, que localiza mediante el comando **UNIX** *ps* y *grep*.
Una vez localizado el proceso apolo y comprobado que está funcionando correctamente (con todos los hilos adecuados) se envía una señal que indica que tiene contenido en la memoria pendiente de leer, Esta señal es gestionada por **Apolo** y la función e hilo correspondientes abren la memoria compartida, extraen la información y colocan el mensaje en el buffer circular correspondiente, confirmando a **Hermes** mediante otra señal que han leído exitosamente el mensaje.

Sin embargo, esta señal únicamente indica a **Hermes** (y por tanto a ***Kerkaf***) que el mensaje ha llegado adecuadamente al demonio **Apolo**, pero no asegura que haya llegado correctamente a la ***Pantalla*** que es lo solicitado en la mayoría de los CU.
Cuando finalmente **Apolo** confirma que el mensaje ha sido recibido por la TTGO, el hilo gestor que gestiona esta confirmación envía mediante un identificador de proceso *tid* una señal diferente a **Hermes** esta vez indicándole que sí ha llegado a buen puerto el mensaje de ***Kerkaf*** por lo que la transmisión queda confirmada, completada y **Hermes** finaliza (Solo procesa un mensaje por ejecución).

Otra importante función de **Hermes** es la de identificar la clave del mensaje recibido, ya que en el CU-6, es tarea de este programa el procesar y guardar en el fichero de configuración la dirección de correo electrónico al cual el Profesor desea que se envíen la Llamadas Remotas.
Todas las especificaciones de la creación de la memoria compartida, la gestión y envío de señales a procesos y subprocesos (hilos) está almacenada y presente en los repositorios indicados en la bibliografía al respecto del ***Back*** [6]

Llamada Remota (Apolo + Hermes)

Esta función del ***Back*** resulta muy diferente de las demás en tanto que es la única acción en la cual la comunicación de la placa TTGO es saliente. La llegada de esta comunicación, que se señaliza con la clave de mensaje 'A', indica a **Apolo** que debe enviar un correo electrónico a la dirección especificada por el Profesor mediante el CU-6.

El procedimiento es el siguiente: al recibir el mensaje, **Apolo** lo guarda en un "carrete" de buffer circular destinado a otro hilo que se encarga de gestionarlos cuando detecta una diferencia entre los punteros de escritura y lectura de este buffer, al estilo de lo que sucede en la ***Pantalla***. Una vez detectado este mensaje y su clave, se envía mediante la función *system()* un comando **msmtp** que utiliza un usuario gmail creado a tal propósito el cual envía el correo a la dirección, el código es sencillo y se trata del siguiente:

Source Code 6.6: envío de mail

```
if(*Carrete[iCarreteR].clave == CRREO){ //comprobación de la clave
    printf("[%s] señal de enviar correo\n", tH);
    //fecha y hora
    std::time_t t = std::time(0);   // get time now
    std::tm* now = std::localtime(&t);
    sprintf(contenidoMensaje, MENSAJEMSMTP, now->tm_hour,//
    now->tm_min, now->tm_mday, now->tm_mon+1, now->tm_year+1900);
    printf("[%s] Mensaje a enviar:  :%s:\n", tH, contenidoMensaje);

    //correo electrónico adecuado
    FILE* direcCorreoFile = fopen("../config/mail.txt", "r");
    if(direcCorreoFile == NULL){
        printf("[%s] error abriendo el fichero mail.txt\n", tH);
    } else {
        if(fgets(direcCorreo, 1024, direcCorreoFile)){
            printf("[%s] Dirección de correo:  :%s:\n", tH, direcCorreo);
            char comandoCorreo[550];
            sprintf(comandoCorreo, //
            "echo \"%s\" | sudo msmtp -C /etc/msmtp/config %s",//
            contenidoMensaje, direcCorreo);
            printf("[%s] Comando a ejecutar:  :%s:\n", tH, comandoCorreo);
            system(comandoCorreo);
        }else{
            printf("[%s] error en la lectura de dirección de correo\n", tH);
        }
    }
}
```

En este código se extrae la hora actual, se introduce sobre la plantilla del correo a envía y se muestra en el log (canal $1 de salida stdout redirigido al nohup). A continuación se extrae la dirección de correo que se desea utilizar del fichero "mail.txt" (no contenido en el repositorio de código ya que es un fichero

de configuración local) Y finalmente se emite el comando con los campos variables completados con la información pertinente, enviando un correo como el siguiente:

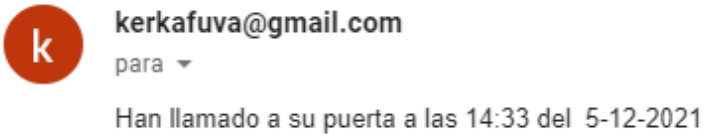

Figura 6.7: Correo de kerkaf para avisar de Llamada Remota

6.3. *kerkaf*

6.3.1. Introducción

Siendo uno de los elementos del prototipo mas sencillos en términos de desarrollo software, la aplicación Android de ***Kerkaf***, desarrollada en Android Studio, ha supuesto un esfuerzo mucho mas tranquilo y remunerado en éxitos debido a que se trata de un entorno muy trillado por infinidad de desarrolladores que ponen a disposición de todo el mundo numerosos ejemplos de como proceder ante cualquier tipo de error o tropiezo.

Este software java y xml ha sido el mas amoldado a las circustancias de entre todos los elementos del prototipo, ya que al ser el último y el mas sencillo, siempre ha supuesto una facilidad el poder cambiar o redefinir su comportamiento en función de lo que los otros dos componentes necesitaran.
Para realizar esta aplicación se partió de una plantilla de aplicación estándar a partir de la cual se ha ido progresando ampliando el número de vistas y *activities* sin demasiada variedad respecto de cualquier ejemplo estandarizado de este tipo de arquitectura Android.

Estructura

Las características de la estructura de este componente, estándar o no, son las siguientes:

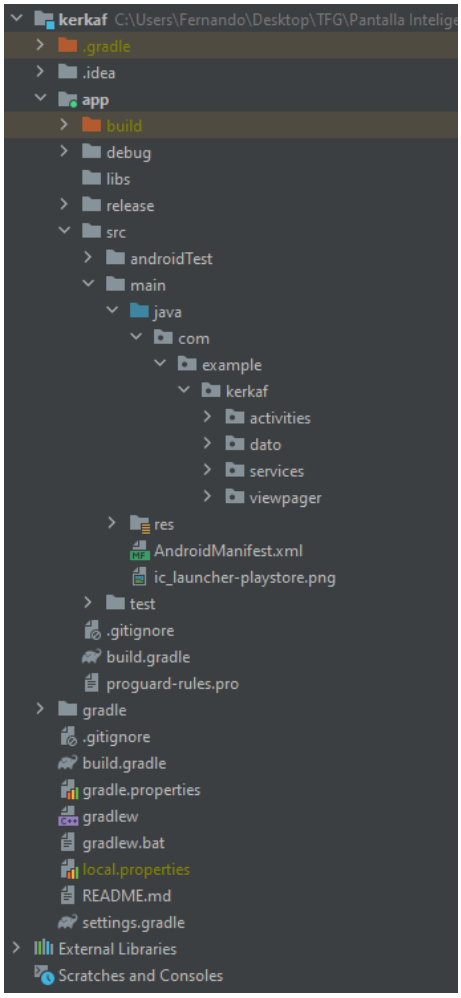

Figura 6.8: Estructura del Proyecto y Aplicación ***Kerkaf***

Aunque la mayoría se trate de componentes auto-generados por el gestor del proyecto Android Studio, cabe resaltar los siguientes:

- **Main**: Este directorio almacena todos los componentes a tener en cuenta en la aplicación sobre los cuales desarrollar propiamente, tanto el flujo de comportamiento y programa como las vistas definidas en xml hasta las ”clases” propiamente dichas que definen la pauta de almacenamiento de los datos de la aplicación.

  - **activities** Siendo el núcleo ejecutado en todo momento en la aplicación, las activities determinan que vista se muestra en todo momento y como actúan ante la interacción del usuario. Este directorio es la ubicación donde se encuentran.
  - **dato**: Este directorio concreto ha sido creado específicamente para este proyecto. Almacena las clases que definen en componente Singleton de cuyo uso se desarrollará mas adelante en este capitulo.
  - **services**: también creado a conveniencia del proyecto pero siguiendo una estructura de directorios mas estandarizada, la carpeta de servicios almacena las clases de cuyos métodos la aplicación hace uso en segundo plano, lanzando estos mientras las vistas y *activities* interactúan normalmente con el usuario.
  - **viewpager**: En el directorio viewpager se almacena la funcionalidad y vistas correspondientes a la gestión de subvistas de la aplicación: un componente de gran utilidad que permite

desplazarse entre diferentes interfaces o subinterfaces gráficas del programa sin cambiar de *activity* lo cual plantea numerosas ventajas de funcionalidad.

- **res**: La carpeta ”res” almacena y ubica todos los componentes visuales de la aplicación, desde las vistas principales ligadas a las *activities* como los pequeños logos o imágenes que en estas se cargan.
- **AndroidManifest**: Este documento en formato xml contiene las especificaciones y configuraciones básicas del proyecto y su app, en ella se determinan los servicios que esta utilizará del dispositivo móvil y las *activities* a cargar en ella.
- **build y settings**: Estos dos archivos de tipo .gradle contienen las especificaciones de compilación del proyecto y librerías a utilizar, su correcta preparación y formato son imprescindibles para que el entorno de proyecto pueda generar adecuadamente el archivo .apk de instalación Android.

A continuación se detallan los aspectos concretos de la aplicación que resultan característicos de este proyecto:

6.3.2. Almacenamiento y *Singleton*

Como se ha mencionado anteriormente en el capítulo de diseño y en este mismo capítulo, ***Kerkaf*** cuenta con un componente *Singleton* aplicando la arquitectura lógica del mismo nombre la cual nos permite implementar un sistema de almacenamiento de datos o estado de la aplicación mediante una clase cuya modificación afecta única y exclusivamente a un objeto inicializado único.
Esto otorga la ventaja de no mantener datos duplicados y en el contexto de la aplicación Android, ha sido la herramienta que ha permitido el paso de datos mas conveniente entre *activities* las cuales al funcionar como núcleos de programa relativamente independientes, forman una encapsulación de sus datos que hacen difícil su interacción con otras *activities* diferentes.

Source Code 6.7: clase singleton

```
public class MySingleton {

    private static MySingleton instance;

    public static void initInstance() {
        if (instance == null){
            instance = new MySingleton();
        }
    }
    public static MySingleton getInstance() {
        return instance;
    }

    private MySingleton() {
        // Constructor hidden because this is a singleton
    }
```

```
        public void customSingletonMethod() {
            // Custom method
        }
    }
```

Esta clase es inicializada con la aplicación exclusivamente una vez, asignandola ligadamente al componente *Application* de Android, lo cual permite el acceso al singleton desde cualquiera de las *activities*.
Los contenidos en datos que queramos que posea van incluidos en el constructor MySingleton() y los métodos también exclusivos de este en customSingletonMethod() o similares. El ejemplo mostrado está vacío en ese aspecto.

6.3.3. Comportamiento

El comportamiento de la aplicación es sencillo: se navega entre tres vistas principales y dentro de dos de ellas por sus posibles subvistas (véase viewpager) En estas el Profesor coloca los datos tal y como desee y pulsando el botón de enviar o guardar se procesa la información correspondiente.
Como se mencionó ya, esta parte del prototipo ha sido la mas sencilla de implementar según ejemplos en la red y, por lo tanto, modificable con mucha facilidad ante las modificaciónes de los requisitos que se vayan observando a lo largo del desarrollo.

Sin embargo la parte interesante del comportamiento de esta aplicación no es el ”que” sino el ”como”, el cual se describe según sus propiedades mas características a continuación.

services

Para mantener el servicio de la aplicación durante la ejecución de una tarea pesada, es preferible colocarla en un hilo aparte, del cual se encargue un core diferente del que está gestionando la interacción con el usuario visualmente, evitando reacciones torpes del dispositivo o similar.

La clase *Service* de Android permite hacer precisamente esto, ya que contiene métodos que debidamente sobre escritos por el programa tienen la capacidad de lanzar una función a placer del programador. Para ***Kerkaf*** esta función es la de realizar todo el protocolo de comunicación SSH, el cual contiene numerosos timeouts que mantienen su hilo de ejecución detenido, por lo que entorpecería el funcionamiento del resto de la aplicación. Al poseer esta prestación, se pueden realizar simultáneamente múltiples acciones.

viewpager

La artífice de las subvistas deslizables e la aplicación es una clase de Android llamada PagerAdapter que hace uso de la librería *viewpager* la cual mediante un objeto creado a gusto del programador que funciona como un modelo de estados posibles, puede mostrar en la misma *activity* numerosos *layouts* sin abandonar el mismo entorno de ejecución ligado, como suele suceder cuando se salta de un *layout* estándar a otro, ya que la activity cambia totalmente.

Esta función tiene numerosas ventajas, la principal es precisamente el ahorro de puestas en contexto de datos que habría que realizar de funcionar cada una de estas vistas ligada a una *activity* distinta.
Y, aparte de esto, proporciona un despliegue visual muy conveniente y de buena estética, cuyo uso es intuitivo y sencillo.

6.3.4. Comunicación SSH

La comunicación SSH, el núcleo de la actividad de comunicación de la aplicación y el método de conexión con el resto del sistema no sería posible sin el uso de las librerías de *apache*: **sshd.client** un demonio de conexión secure shell que realiza conexiones SSH con un servidor.

Al necesitar el sistema una ejecución remota de comandos (**Hermes**), se hace uso de la flag *CHANNEL_EXEC* que sencillamente crea la conexión con el comando de ejecución y sus argumentos como parámetros entrantes del *createChannel()*. Antes de ejecutar el comando mediante *.open()* sobre el canal creado, se ubican los *Stream* de bytes donde se almacenará la respuesta, que tras abrir el canal, es procesada para confirmar a ***Kerkaf*** que la ***Pantalla*** ha recibido adecuadamente los datos que se pretendían enviar.

Capítulo 7

Pruebas y *Testing*

7.1. Pruebas realizadas

La mayoría de las comprobaciones realizadas del correcto funcionamiento del prototipo se han realizado a lo largo del propio desarrollo y en la mayoría de los casos las pruebas descritas a continuación están compuestas por subcomponentes de funcionalidad que han sido probados uno a uno.
A pesar de esto, los cuadros de pruebas que se muestran en este apartado aluden todas a pruebas que en el momento de finalización del prototipo arrojan el resultado esperado. Y se puede inferir de estas que sus flujos intermedios de comportamiento del sistema son también correctos.

7.2. Consideraciones

Cabe mencionar también dos importantes detalles de la estructura de este capítulo: Para empezar, el orden elegido de la representación de las pruebas no se corresponde en absoluto con su orden cronológico en el desarrollo del prototipo. Se trata de una batería de pruebas completa del funcionamiento del mismo. Además, estas pruebas están prácticamente extraídas de los casos de uso esperados del sistema, si bien hay añadida alguna sobre una buena usabilidad que no queda descrita en ningún caso de uso.
El otro detalle importante al respecto de estas pruebas es que estan enteramente realizadas en un entorno de red controlado por el alumno que ha desarrollado el proyecto, por lo que su realización o las precondiciones de las mismas en el entorno para el que está previsto el prototipo (red UVA) podrían no ser iguales (como se demostrará en la presentación si es que esta sucede en el entorno universitario de Valladolid).

7.3. Pruebas en Detalle

Prueba 1	**Consultar la configuración**
Objetivo	El Profesor puede acceder correctamente a su configuración y comprobar que se ha conservado tal y como la ha modificado anteriormente
Actor	Profesor
Precondición	Existe una configuración guardada de antemano
Acción esperada	***Kerkaf*** muestra en sus dos subvistas la configuración esperada
Resultado	**Positivo**

Cuadro 7.1: Prueba 1 de funcionamiento de la vista Configuración de ***Kerkaf***

Prueba 2	**Modificar la configuración**
Objetivo	El Profesor puede modificar correctamente los campos que determinan la configuración y guardarla
Actor	Profesor
Precondición	El Profesor ha iniciado correctamente la vista de Configuración
Acción esperada	Tras pulsar el botón de ”guardar” la vista retorna al inicio y la configuración se ha guardado correctamente
Resultado	**Positivo**

Cuadro 7.2: Prueba 2 de funcionamiento de la vista Configuración de ***Kerkaf***

Prueba 3	Consulta de los datos Cargados en Panel de Información
Objetivo	El Profesor puede acceder correctamente a la vista principal de ***Kerkaf*** y consultar los datos cargados en el Panel de Información
Actor	Profesor
Precondición	Existen datos guardados (enviados) de antemano
Acción esperada	Tras entrar en la vista Principal pulsando ”Inicio” el Profesor puede consultar los datos del Panel de Información cargados en la ***Pantalla*** correctamente
Resultado	**Positivo**

Cuadro 7.3: Prueba 1 de funcionamiento de la vista Principal de ***Kerkaf***

Prueba 4	Consulta de los datos Cargados en Horario de Clases
Objetivo	El Profesor puede acceder correctamente a la vista principal de ***Kerkaf*** y consultar los datos cargados en el Horario de Clases
Actor	Profesor
Precondición	Existen datos guardados (enviados) de antemano
Acción esperada	Tras entrar en la vista Principal pulsando ”Inicio” el Profesor puede consultar los datos del horario cargados en la ***Pantalla*** correctamente
Resultado	**Positivo**

Cuadro 7.4: Prueba 2 de funcionamiento de la vista Principal de ***Kerkaf***

Como se ha mencionado antes, las siguientes pruebas contienen de manera implicita la comprobación del funcionamiento de muchos aspectos de la ***Pantalla***, como su correcta navegación por el menú, el acceso a este y la correcta respuesta del panel de control físico del prototipo.

Prueba 5	Consulta de los datos Cargados en la vista de Tutorías
Objetivo	El Profesor puede acceder correctamente a la vista principal de ***Kerkaf*** y consultar los datos cargados en la vista de Tutorías
Actor	Profesor
Precondición	Existen datos guardados (enviados) de antemano
Acción esperada	Tras entrar en la vista Principal pulsando ”Inicio” el Profesor puede consultar los datos de la vista de Tutorías cargados en la ***Pantalla*** correctamente
Resultado	**Positivo**

Cuadro 7.5: Prueba 3 de funcionamiento de la vista Principal de ***Kerkaf***

Prueba 6	Consulta de los datos Cargados en la vista de Asignaturas
Objetivo	El Profesor puede acceder correctamente a la vista principal de ***Kerkaf*** y consultar los datos cargados en la vista de Asignaturas
Actor	Profesor
Precondición	Existen datos guardados (enviados) de antemano
Acción esperada	Tras entrar en la vista Principal pulsando ”Inicio” el Profesor puede consultar los datos de la vista de Asignaturas cargados en la ***Pantalla***
Resultado	**Positivo**

Cuadro 7.6: Prueba 4 de funcionamiento de la vista Principal de ***Kerkaf***

Prueba 7	Envío de los datos modificados del Panel de Información
Objetivo	El Profesor al enviar la información modificada de la subvista del Panel de Información esta aparece correctamente en la ***Pantalla***
Actor	Profesor
Precondición	El Profesor ha identificado correctamente su ***Pantalla*** en la configuración
Acción esperada	Tras modificar los datos y pulsar ”enviar” la ***Pantalla*** se modifica con los datos esperados.
Resultado	**Positivo**

Cuadro 7.7: Prueba 1 de funcionamiento de envío de datos de ***Kerkaf*** a la ***Pantalla***

Prueba 8	Envío de los datos modificados del Horario
Objetivo	El Profesor al enviar la información modificada de la subvista del Horario esta aparece correctamente en la ***Pantalla***
Actor	Profesor
Precondición	El Profesor ha identificado correctamente su ***Pantalla*** en la configuración
Acción esperada	Tras modificar los datos y pulsar ”enviar” la ***Pantalla*** se modifica con los datos esperados.
Resultado	**Positivo**

Cuadro 7.8: Prueba 2 de funcionamiento de envío de datos de ***Kerkaf*** a la ***Pantalla***

Prueba 9	Envío de los datos modificados de las Tutorías
Objetivo	El Profesor al enviar la información modificada de la subvista de Tutorías esta aparece correctamente en la ***Pantalla***
Actor	Profesor
Precondición	El Profesor ha identificado correctamente su ***Pantalla*** en la configuración
Acción esperada	Tras modificar los datos y pulsar ”enviar” la ***Pantalla*** se modifica con los datos esperados.
Resultado	**Positivo**

Cuadro 7.9: Prueba 3 de funcionamiento de envío de datos de ***Kerkaf*** a la ***Pantalla***

Prueba 10	Envío de los datos modificados de las Asignaturas
Objetivo	El Profesor al enviar la información modificada de la subvista de Asignaturas esta aparece correctamente en la ***Pantalla***
Actor	Profesor
Precondición	El Profesor ha identificado correctamente su ***Pantalla*** en la configuración
Acción esperada	Tras modificar los datos y pulsar ”enviar” la ***Pantalla*** se modifica con los datos esperados.
Resultado	**Positivo**

Cuadro 7.10: Prueba 4 de funcionamiento de envío de datos de ***Kerkaf*** a la ***Pantalla***

Prueba 11	Consulta de la opción del menú ”Horario Clases”
Objetivo	El Alumno es capaz de consultar los datos del horario de clases del Profesor del día de la semana que desee correctamente
Actor	Alumno
Precondición	El Profesor ha cargado su información del Horario en la ***Pantalla*** y habilitado la opción de su uso
Acción esperada	Tras navegar por el menú de la ***Pantalla*** y seleccionar la opción, se muestra correctamente el Horario y es posible seleccionar correctamente el día a consultar.
Resultado	**Positivo**

Cuadro 7.11: Prueba 1 de funcionamiento de la ***Pantalla***

Prueba 12	Consulta de la opción del menú ”Tutorías”
Objetivo	El Alumno es capaz de consultar los datos de Tutorías del Profesor y consultar el código QR presente en la vista
Actor	Alumno
Precondición	El Profesor ha cargado su información correctamente en la vista de Tutorías y habilitado la opción de su uso
Acción esperada	Tras navegar por el menú de la ***Pantalla*** y seleccionar la opción ”Tutorías” se muestra correctamente y el código QR funciona correctamente.
Resultado	**Positivo**

Cuadro 7.12: Prueba 2 de funcionamiento de la ***Pantalla***

Prueba 13	Consulta de la opción del menú ”Asignaturas”
Objetivo	El Alumno es capaz de consultar los datos de la vista ”Asignaturas” del Profesor correctamente.
Actor	Alumno
Precondición	El Profesor ha cargado su información de Asignaturas que imparte en la ***Pantalla*** y habilitado la opción de acceder a ella
Acción esperada	Tras navegar por el menú de la ***Pantalla*** y seleccionar la opción de ”Asignaturas” se muestran correctamente ambos paneles de información de texto que contiene esta Vista.
Resultado	**Positivo**

Cuadro 7.13: Prueba 3 de funcionamiento de la ***Pantalla***

Prueba 14	Consulta de la opción del menú ”Llamada Remota”
Objetivo	El Alumno puede enviar un aviso al Profesor mediante correo electrónico avisando de que está esperándole.
Actor	Alumno
Precondición	El Profesor ha habilitado una dirección de correo electrónico adecuada y habilitado el uso de esta opción
Acción esperada	Seleccionando la opción del menú ”Llamada Remota” la ***Pantalla*** se comunica con el ***Back*** y este envía un correo electrónico a la dirección estipulada
Resultado	**Positivo**

Cuadro 7.14: Prueba de funcionamiento de la comunicación de la ***Pantalla*** con el resto del sistema

Capítulo 8

Conclusiones

8.1. Conclusiones Principales

La correcta interpretación de los resultados de este proyecto y sus conclusiones deben pasar primero por un escrutinio comparativo con los objetivos propuestos incialmente.
Principalmente, se han cumplido sin problemas y de forma satisfactorio, sin embargo hay matices a tener en cuenta a la hora de evaluar este éxito, algunos de estos matices implican que se ha realizado más de lo que se pretendía, y otros en cambio indican lo contrario.

Una de estas observaciones a posteriori es la reiteración de la sensación inicial al respecto de lo que este proyecto podría enseñar al alumno. Dado que como se mencionó abarca campos de la tecnología informática y casi electrónica que este no había realizado adecuadamente durante el resto del transcurso del grado. Esto es debido a que o bien no se prestó la atención suficiente a estos aspectos durante las asignaturas impartidas o que, sencillamente, se ha llegado hasta aquí sin haber realizado dichas Asignaturas, como por ejemplo las referentes a Sistemas Empotrados o Sistemas Móviles.
La implicación constructiva de este hecho es que el alumno ha tenido la oportunidad de aprender desde cero y por lo tanto hacer crecer enormemente su conocimiento en estos campos. En cambio, la implicación negativa es que a término de este proyecto, se contemplan los objetivos propuestos como poco ambiciosos e incluso ligeramente insuficientes.
De nuevo, el balance es tremendamente positivo, ya que se trata de un Sistema desarrollado en numerosos entornos muy distintos y la valoración del aprendizaje obtenido es mas que positiva.

Como bien se menciona en el capítulo de metodología, el contexto de la realización de este proyecto es el de un horario de trabajo completo y fines de semana dedicados a él, por lo que el correcto reflejo del desarrollo de un proyecto normal no se produce en este caso, como debiera ser en una situación académica normal, no obstante, el resultado final de este, a pesar de sus lagunas cronológicas, es el esperado y las conclusiones muy positivas.

8.2. Lineas Futuras

Al mencionar la posibilidad de que los objetivos propuestos puedan haber sido insuficientes o poco ambiciosos se desplaza carga del proyecto a este apartado documental, el cual ahora se debe rellenar con las consideraciones que provocan esta impresión, ya que son numerosas las mejoras u objetivos planteados a mayores durante la realización de este proyecto.

Para comenzar, indudablemente se debería reforzar el mecanismo de seguridad de la comunicación Bluetooth, ya que el hecho de que exista un protocolo secreto de comunicación, si bien evita que se pueda interceptar información sensible, no evita que se pueda perturbar el correcto funcionamiento del prototipo. Planteado como una posibilidad de solución a este problema, se contempla que la ***Pantalla*** únicamente acepte comunicaciones codificadas en la clave privada que comparta con el ***Back*** a través de un protocolo de emparejamiento. Aunque esto ya sucede de forma implícita al emparejarlos de forma normal, nada impide a un usuario malicioso el re-emparejar el dispositivo y privarlo de su comunicación con el ***Back*** ya que es este el único que tiene la posibilidad de habilitar o deshabilitar esta conexión.

Otra importante mejora que se considera muy espectacular y de bajo coste es la de la carga de imágenes seleccionadas a placer. Esta prestación recaería principalmente en cuanto a coste computacional en el ***Back***, ya que al recibir una imagen en formato *bitmap* tendría la tarea de transformarlo, tal y como hace el programa *image2lcd* en codificación hexadecimal, y luego enviar esta información a la ***Pantalla*** la cual, tomándola con un objeto como ya hace con las precargadas, podría utilizarla en su carga de imágenes de perfil.

Con respecto a esta misma mejora, se ha planteado también la posibilidad de generar directamente un código QR y no como una carga de imagen, sino que mediante las propias apis de generación de códigos QR o con un programa propio, se genere uno que igual que una imagen se envie a la ***Pantalla*** para ser el mostrado en la vista de Tutorías.

Otra modificación a realizar muy importante, que, de hecho, modificaría practicamente en su totalidad el funcionamiento de la comunicación de ***Kerkaf*** con el ***Back***, sería la de utilizar directamente un protocolo REST para envíar y recibir información.

Esta forma de comunicación, de cuya filosofía ya hace uso el actual funcionamiento del prototipo, es de extrema sencillez y tiene una ventaja antes no prevista: al igual que con la implementación de ***Kerkaf*** al ser un método tan estandarizado de comunicación se obtiene un rendimiento de desarrollo mucho mayor a la hora de resolver problemas que, indudablemente, ya han afectado a otro programador en algún momento.

Para terminar, hay una última mejora clara que abordar si se realizara una continuación de este proyecto, y es la de realizar un tercer componente del elemento ***Back*** cuya función fuera la de velar por la salud del canal de comunicación bluetooth, de manera que si se ve dañado o interrumpido, el resto del sistema tenga notificación de ello, pudiendo actuar en restaurarlo o que al menos el extremo del usuario contenga un aviso al respecto.

Esta mejora llegó a plantearse incluso mediante la implementación de un comando **UNIX** disponible cuya función es la de mantener un control vigilante y reactivo de un fichero, que al fin y al cabo es como contempla **UNIX** el canal de comunicación bluetooth al cargarlo con *rfcomm* en el directorio */dev*, sin embargo, en este sistema el propio **Hermes** se encarga de relanzar a **Apolo** al finalizarse los timeouts creados a tal fin. Por lo que no resulta necesaria esta vigilancia en lo que respecta a ***Kerkaf***, sí suponiendo un problema para la ***Pantalla*** que no tiene la capacidad de actuar de igual manera.

Manual de instalación y Guía de Uso

En este apendice se establece cuales son los pasos a seguir una vez el usuario tiene a su disposición los elementos necesarios, tanto los elementos físicos del prototipo como los programas de carga del programa Arduino, así como una interfaz git para descargar el código de los repositorios (no tiene que ser necesariamente la utilizada en los ejemplos de esta sección).

A.1. *Pantalla*

La siguiente sección alude a la instalación del programa en una placa TTGO lilygo v2.3 asumiendo que la distribución de este prototipo se realiza sin haber instalado previamente el programa que debe portar. Las siguientes etapas corresponden a la descarga del código que se va a utilizar y la preparación del entorno IDE que nos permite cargar el programa en la placa en sí.

A.1.1. Obtención del código

Lo primero es obtener el programa, para lo cual se ha habilitado como público el repositorio de almacenamiento y desarrollo (solo lectura) a través del cual podemos obtener el enlace http que permite la clonación y descarga del mismo en su versión comercial de la rama ”master”.

Tras acceder a la página del repositorio [4] debemos obtener el enlace de clonación como en la figura A.1.

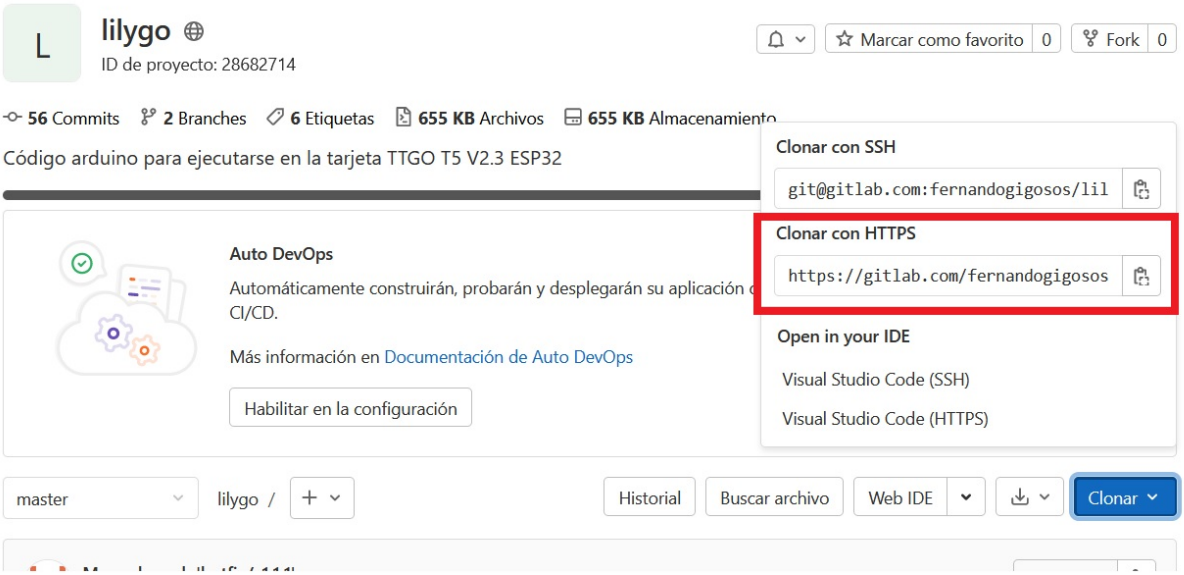

Figura A.1: Obtención de la URL de clonación

Si se está utilizando un entorno gráfico para gestionar la clonación como sourcetree la forma de añadir el repositorio es la siguiente: A.2.

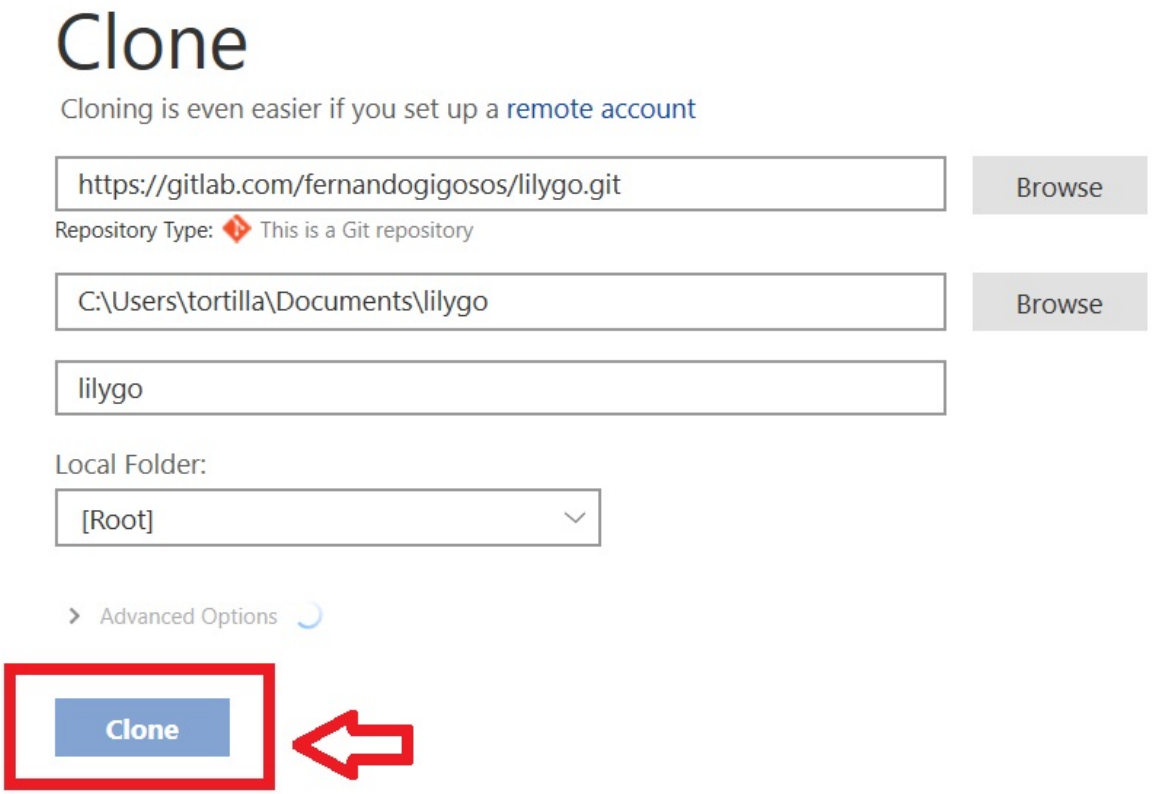

Figura A.2: vista de ”clonar” en sourceTree

Una vez clonado el repositorio y descargado, toca configurar el entorno de carga del programa, el Arduino IDE [14], para lo cual debemos añadir ciertas bibliotecas para el correcto funcionamiento del compilador. Concretamente una de ellas, Adafruit_GFX se puede cargar automáticamente mediante el gestor de bibliotecas, al cual se accede mediante la pestaña ”Programa” y cuyo aspecto es el siguiente: A.3

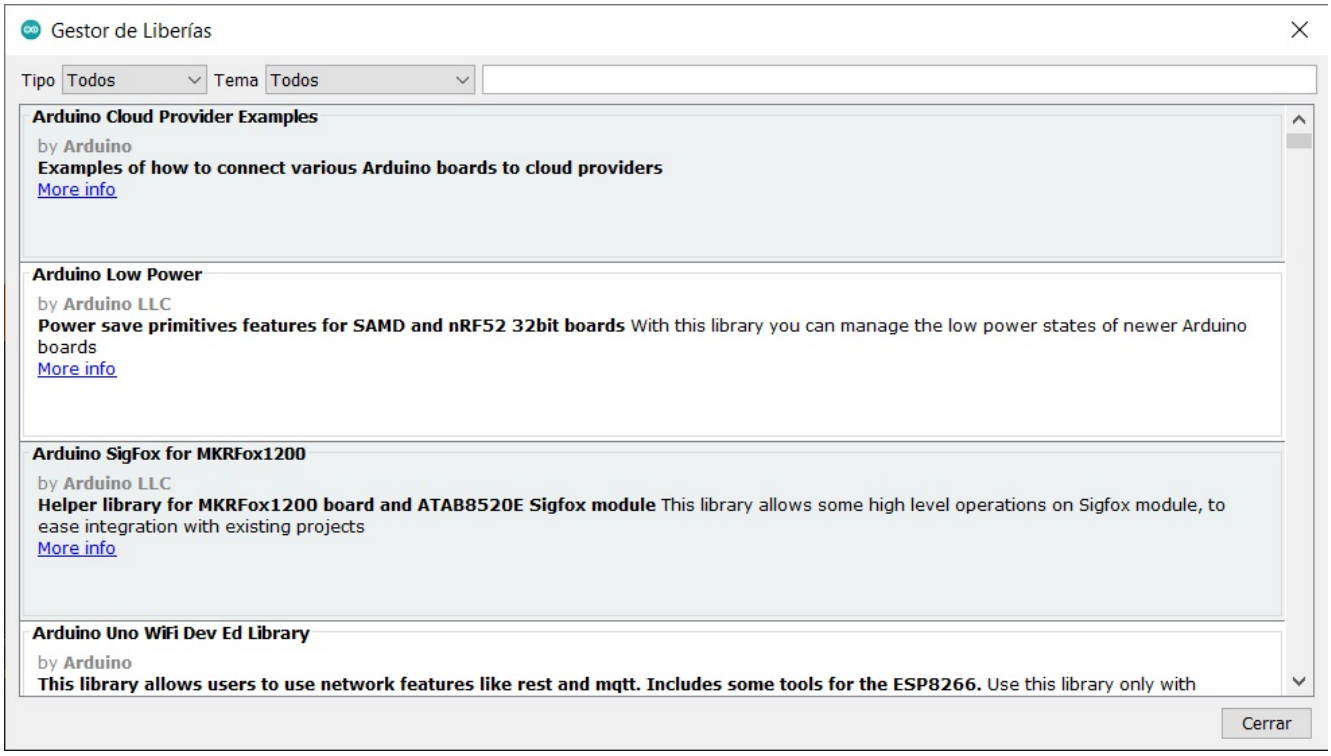

Figura A.3: vista de ”clonar” en sourceTree

En la ventana mostrada en la figura A.3 se debe escribir ”Adafruit_GFX” en el buscador e instalar la biblioteca.

Una vez instalada automáticamente esta biblioteca es necesario instalar las demás manualmente, debido a que si se hace de forma automática, no respeta adecuadamente las bibliotecas indicadas en el código, añadiendo en algunos casos sobrecargas de funciones que no deben producirse.

La biblioteca que debe añadirse manualmente es la del repositorio [8] que debe descargarse e añadirse al directorio `\Documentos\Arduino\libraries` donde, a su vez, debe añadirse al contenido de su carpeta ”src” el contenido de la carpeta del repositorio **lilygo** de ”libraries” sustituyendo los ficheros que ahí se encuentren.
Si existe alguna duda al respecto de como añadir adecuadamente estas bibliotecas, hay indicaciones en la siguiente página [15].

Además de estas bibliotecas, es necesario añadir en el panel de gestión de placas la de ”esp32” (figura A.4) y configurar en la pestaña de herramientas la configuración indicada en la figura A.5 que incluye la indicación de la placa adecuada.

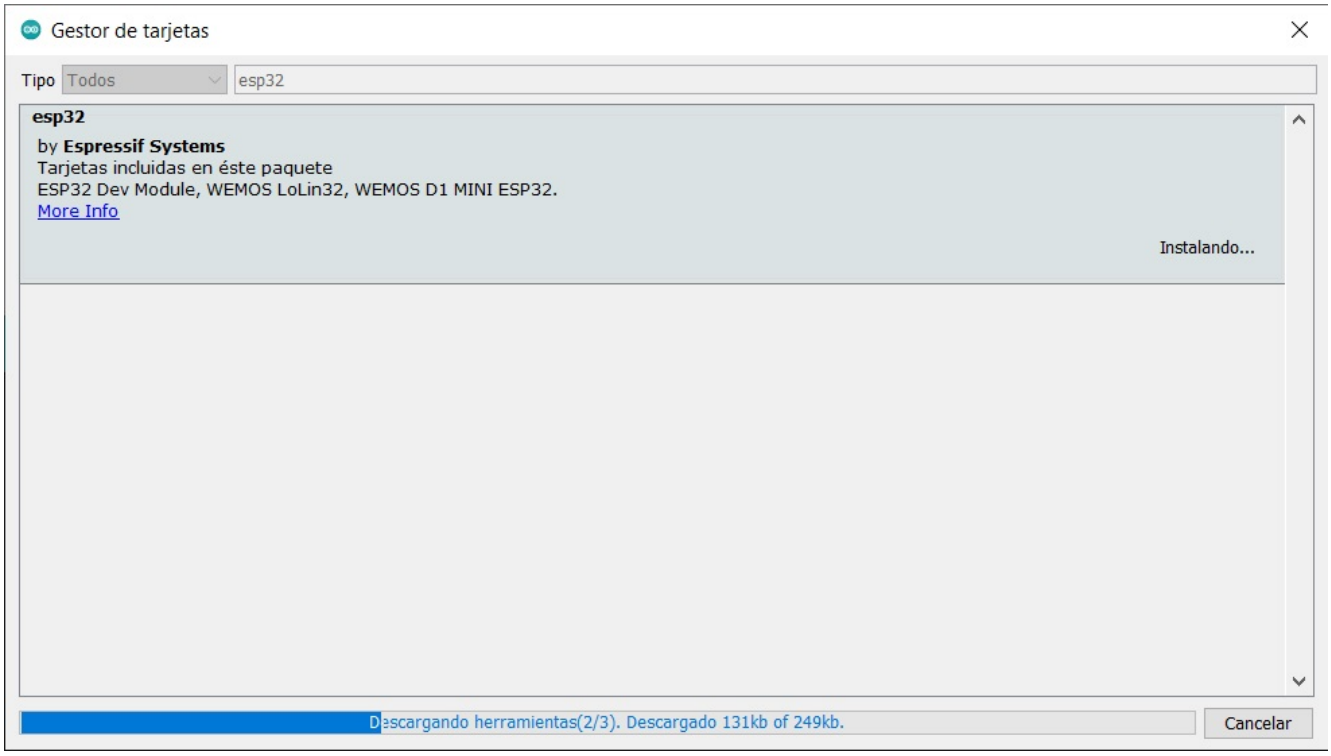

Figura A.4: Panel de gestión de los controladores de placas con la busqueda del ”esp32”

Figura A.5: configuración de la pestaña ”herramientas” para cargar el programa adecuadamente

Una vez preparadas todas estas bibliotecas y configuración, se puede comprobar que todo funciona correctamente mediante una compilación del programa (figura A.6) y acto seguido pulsar el botón adyacente para cargarla en la placa que debe estar debidamente conectada mediante un cable USB a mini-USB al ordenador.

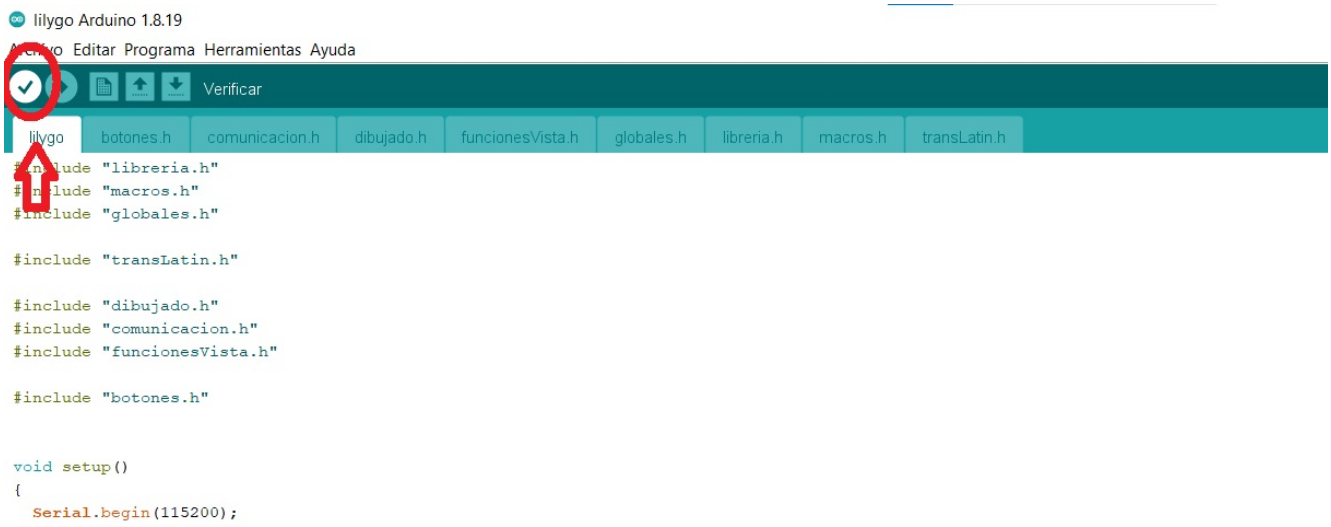

Figura A.6: Sección de compilación y carga del programa a la placa

Una vez completados estos pasos la ***Pantalla*** está lista.

A.2. *Back*

La instalación y puesta a punto del Back se solapa con una instalación y configuración estandarizada de una **Raspberry** Pi 2B, cuyos pasos pueden consultarse aquí: [16].

Una vez instalado el entorno se debe descargar el código del proyecto [6] mediante git en la **Raspberry** tal y como se ha realizado con la ***Pantalla*** y su respectivo software.

A.2.1. Emparejamiento

Para conectarse vía bluetooth con la ***Pantalla*** es necesario que se coloque el dongle bluetooth en la **Raspberry** físicamente en uno de sus puertos USB.
Una vez hecho esto, mediante la herramienta `bluetoothctl` se debe añadir y emparejar el dispositivo de nombre ”ESP32SPP”. Es posible emparejarlo de maneras menos específicas, pero esta es la que ha funcionado al proyecto sin problemas
El resto de funciones de **UNIX** necesarias vienen incorporadas en la ejecución del código del ***Back*** por lo que no es necesaria mayor instalación.

A.2.2. Compilación

Sin embargo aún falta un paso, aunque corto, para poder utilizar el prototipo desde ***Kerkaf***. Es necesario compilar los códigos ejecutando `make` en los directorios del ***Back*** de: `\mensajero` y `\demonio` compilando a **hermes** y **apolo** respectivamente. Una vez compilados, el ***Back*** está listo

A.3. *Kerkaf*

La instalación de esta parte del prototipo resulta la mas sencilla, ya que únicamente es necesario descargar el archivo *APK* de instalación en el dispositivo Android, y se debe permitir instalar aplicaciones de origenes desconocidos (al no estar publicada en la playStore).

Para indicaciones mas específicas de como hacer esto se puede consultar esta página: [51]

Para obtener la APK, dado que en el momento de elaboración de este documento no existe un repositorio de almacenamiento de esta APK, ponerse en contacto con su autor mediante la dirección de correo: **fernandogigosos@gmail.com**.

A.4. Guía de Uso

El funcionamiento de ambas interfaces de usuario, tanto ***Kerakf*** como la ***Pantalla*** han sido diseñadas de tal manera que el uso es totalmente intuitivo, especialmente para el alumno, que únicamente maneja 4 botones para interactuar con el prototipo. Aun así, se muestra un ejemplo de actuación para ambos usuarios.

Profesor: El profesor, tras haber cargado correctamente los datos de identificación y configuración de su ***Pantalla*** al pulsar ”Inicio” tiene ante sí las cuatro subvistas de carga de información y configuración en el display de tinta electrónica. Sencillamente deslizando hacia los lados se puede consultar la que se desee y tocando sus campos de texto se despliega el teclado de Android que le permite rellenar los datos como guste.

Una vez rellenos, simplemente hay que pulsar la tecla de ”enviar” y el dispositivo procesará la información guardándola y enviándola al dispositivo remoto.

En cuanto al **Alumno** La única interacción de funcionamiento de la ***Pantalla*** es mediante los botones de ”Menú”, ”Anterior”, ”Siguiente” y ”Seleccionar” que le permiten navegar por las vistas con sencillez. Tal y como se describe en el capítulo de *Diseño* las vistas navegan siempre contra el Menú, en el cual el Alumno puede con los botones seleccionar la que desee ver, salvo en el caso de la cuarta opción del menú, que le permite realizar la Llamada Remota al profesor, tras la cual, devuelve al display a la vista del Panel de Información (vista principal).

Esta guía de uso, aunque escueta, refleja la sencillez de uso que implica el prototipo y, sobre todo, el hecho de que el uso del prototipo se explica por si mismo, ya que es uno de los objetivos impuestos en el proyecto.

Bibliografía

[1] Wikipedia - Diagrama de muestra de funcionamiento de la tinta electrónica - Fundación Wikimedia Equinix:
https://es.wikipedia.org/wiki/Tinta_electr%C3%B3nica
último acceso: 17/1/2021

[2] Wikipedia - Entrada de la enciclopedia al respecto de los casos de uso - Fundación Wikimedia Equinix:
https://es.wikipedia.org/wiki/Caso_de_uso
último acceso: 17/1/2021

[3] Wikipedia - Explicación del problema de los dos Generales - Fundación Wikimedia Equinix:
https://es.wikipedia.org/wiki/Problema_de_los_dos_generales
último acceso: 17/1/2021

[4] Fernando Gigosos Ronda - Repositorio de almacenamiento y desarrollo del programa Arduino de la ***Pantalla*** - GitLab Inc:
https://gitlab.com/fernandogigosos/lilygo
último acceso: 17/1/2021

[5] Fernando Gigosos Ronda - Repositorio de almacenamiento y desarrollo del proyecto Android Studio de ***Kerkaf*** - GitLab Inc:
https://gitlab.com/fernandogigosos/kerkaf
último acceso: 17/1/2021

[6] Fernando Gigosos Ronda - Repositorio de almacenamiento y desarrollo del código c++ del ***Back*** en la raspberry - GitLab Inc:
https://gitlab.com/fernandogigosos/raspi-bff
último acceso: 17/1/2021

[7] makerbase-mks - repositorio git del programa de transformación de bitmaps a hexadecimal - Microsoft:
https://github.com/makerbase-mks/Software/tree/master/Image2LCD
último acceso: 17/1/2021

[8] lewisxhe - rama extendida de GxEDP para la placa usada en este prototipo - Microsoft:
https://github.com/lewisxhe/GxEPD
último acceso: 17/1/2021

[9] ZinggJM - Segunda versión de la libreria gráfica de GxEPD que finalmente no se usó - Microsoft:
https://github.com/ZinggJM/GxEPD2

[10] LilyGO - Repositorio del propio fabricante de la placa lilygo con ejemplos - Microsoft:
https://github.com/Xinyuan-LilyGO/LilyGo-T5-Epaper-Series

[11] ZinggJM - repositorio de la librería gráfica principal utilizada - Microsoft:
https://github.com/ZinggJM/GxEPD

[12] Alex Laurent - Solución de bugs del OS del ESP32 a las tareas de iteración vacia - Microsoft:
https://github.com/espressif/esp-idf/issues/1646 https://www.esp32.com/viewtopic.php?t=10411

[13] Usuario z8888q - Ejemplo de implementación del Singleton sobre Application - Microsoft:
https://gist.github.com/z8888q/2593576

[14] Arduino - Página de Arduino IDE para descargar el programa - Arduino:
https://www.arduino.cc/en/software
último acceso: 17/1/2021

[15] Arduino - Tutorial de como añadir las bibliotecas manualmente - Arduino:
https://www.arduino.cc/en/guide/libraries
último acceso: 17/1/2021

[16] Raspberry Pi Foundation - Instalación del Raspbian en una Raspberry PI - Raspberry Pi Foundation:
https://projects.raspberrypi.org/en/projects/raspberry-pi-setting-up
último acceso: 17/1/2021

[17] Rui & Sara Santos - Mini aplicación de aprendizaje para Bluetooth - Random Nerd Tutorials blog:
https://randomnerdtutorials.com/esp32-bluetooth-classic-arduino-ide/
último acceso: 17/1/2021

[18] Nuno Santos - Manual de uso de la librería SerialBT - techtutorialsx:
https://techtutorialsx.com/2018/12/09/esp32-arduino-serial-over-bluetooth-client-connection-event/
último acceso: 17/1/2021

[19] Phillip Burgess - Instrucciones de uso de Fuentes de texto en AdaFruit - Adafruit Industries:
https://learn.adafruit.com/adafruit-gfx-graphics-library/using-fonts
último acceso: 17/1/2021

[20] Elegoo - Página de soporte y software de elegoo, marca de distribución de herramientas arduino - Elegoo:
https://www.elegoo.com/pages/arduino-kits-support-files
último acceso: 17/1/2021

[21] Digi-Key Electronics - útil página de identificación de resistencias eléctricas por su patrón cromático - Digi-Key Electronics:
https://www.digikey.com/en/resources/conversion-calculators/conversion-calculator-resistor-color-code
último acceso: 17/1/2021

[22] Microcontrollers lab - Ejemplo sencillo de como instalar un botón en una placa ESP32 con código de ejemplo - Microcontrollers lab:
https://microcontrollerslab.com/push-button-esp32-gpio-digital-input/
último acceso: 17/1/2021

[23] RuffaloLavoisier - Manual de acceso y manipulación remota de raspberry - Raspberry Pi Foundation:
https://www.raspberrypi.com/documentation/computers/remote-access.html
último acceso: 17/1/2021

[24] ivooc28 - Instalador con instrucciones del driver de Realtek RTL8761B chip - Raspberry Pi Foundation:
https://forums.raspberrypi.com/viewtopic.php?t=294634
último acceso: 17/1/2021

[25] Google - Página principal del entorno de desarrollo Android Studio - Google:
https://developer.android.com/studio
último acceso: 17/1/2021

[26] Office Timeline - Página de planificación software - Office Timeline, LLC:
https://online.officetimeline.com
último acceso: 17/1/2021

[27] abhijeetcatches33 - Instrucciones de cómo utilizar el protocolo SSH para ejecutar comandos remotamente - geeksforgeeks:
https://www.geeksforgeeks.org/how-to-execute-commands-remotely-via-ssh-in-android/
último acceso: 17/1/2021

[28] Scott Chacon - Página de manual de uso de git y git-flow - Software Freedom Conservancy Git:
https://git-scm.com/book/es/v2/Fundamentos-de-Git-Obteniendo-un-repositorio-Git
último acceso: 17/1/2021

[29] AndroidDev - Página del foro de Android Studio indicando como firmar adecuadamente una apk para hacer posible su instalación - Google Developers:
https://developer.android.com/studio/publish/app-signing#generate-key
último acceso: 17/1/2021

[30] Fahmida Yesmin - Manual con ejemplos de scripts bash - Linux Hint LLC:
https://linuxhint.com/30_bash_script_examples/#t1
último acceso: 17/1/2021

[31] LilyGo - Página de producto y especificaciones de la gama lilygo de placas TTGO ESP32 - LILYGO:
http://www.lilygo.cn/prod_view.aspx?TypeId=50031&Id=1393&FId=t3:50031:3
último acceso: 17/1/2021

[32] G6EJD-David -Interesante vídeo sobre monitorización de la carga restante de la batería mediante medición voltaica - Youtube-Google:
https://www.youtube.com/watch?v=qKUrXwkr3cc
último acceso: 17/1/2021

[33] QrCodeMonkey - Aplicación utilizada para generar el código QR de manera que la pixelación no contenga grises - QrCodeMonkey.com:
https://www.qrcode-monkey.com/#email
último acceso: 17/1/2021

[34] Pascal Werkl - Pequeño tutorial stack overflow de demonio linux - Stack Overflow:
https://stackoverflow.com/questions/17954432/creating-a-daemon-in-linux
último acceso: 17/1/2021

[35] Nick Gammon - pequeño sketch de funcionamiento desligado de parpadeo de dos leds en Arduino - gammon.com:
http://www.gammon.com.au/blink
último acceso: 17/1/2021

[36] BhanuKiran - Instrucciones de uso correctas de rfcomm - askubuntu.com Stack Overflow:
https://askubuntu.com/questions/1025080/unable-to-create-dev-rfcomm-port
último acceso: 17/1/2021

[37] saudade - Instrucciones de uso de SSH de apache para la aplicación Android - Stack Overflow:
https://stackoverflow.com/questions/63075107/how-to-execute-remote-commands-using-apache-mina-sshd
último acceso: 17/1/2021

[38] Rui & Sara Santos - Manual de uso del dual core de ESP32 - Random Nerd Tutorials blog:
https://randomnerdtutorials.com/esp32-dual-core-arduino-ide/
último acceso: 17/1/2021

[39] Anónimo - Manual de uso del gestor de interrupciones de ESP32 - LastMinuteEngineers.com:
https://lastminuteengineers.com/handling-esp32-gpio-interrupts-tutorial/
último acceso: 17/1/2021

[40] Tech It Yourself - semaforos en ESP32 - iotsharing:
http://www.iotsharing.com/2017/06/how-to-use-binary-semaphore-mutex-counting-semaphore-resource-management.html?m=1
último acceso: 17/1/2021

[41] Angel salvador - foro explicando como instalar una fuente con acentos en 8 bits - sigmdel.ca:
https://www.sigmdel.ca/michel/program/misc/gfxfont_8bit_en.html#toc
último acceso: 17/1/2021

[42] Anupam Chugh - Gestor de ficheros internos para guardado entre ejecuciones Android - JournalDev:
https://www.journaldev.com/9383/android-internal-storage-example-tutorial
último acceso: 17/1/2021

[43] Abhinav - Tutorial para gestionar el regreso a la activity anterior - Stack Overflow:
https://stackoverflow.com/questions/4038479/android-go-back-to-previous-activity
último acceso: 17/1/2021

[44] Rishu Mishra - Manual de Servicios para Arduino - geeksforgeeks:
https://www.geeksforgeeks.org/services-in-android-with-example/
último acceso: 17/1/2021

[45] Hardcore_Graverobber - Tutorial curioso sobre ejecución de código en el último instante de la aplicación Android para no perder datos - Stack Overflow:
https://stackoverflow.com/questions/37615791/how-to-execute-code-before-android-app-is-killed/37615983
último acceso: 17/1/2021

[46] Anupam Chugh - Tutorial de uso del viewpager - JournalDev:
https://www.journaldev.com/10096/android-viewpager-example-tutorial
último acceso: 17/1/2021

[47] AndroidDev - Semáforos en Android - Google Developers:
https://developer.android.com/reference/java/util/concurrent/locks/Lock
último acceso: 17/1/2021

[48] RiccardOne - Instrucciones de uso del protocolo msmtp en la distribución raspberry utilizada - Raspberry Pi Foundation:
https://forums.raspberrypi.com/viewtopic.php?f=28&t=244147
último acceso: 17/1/2021

[49] Levente Polyak & team - Mas tutoriales de msmtp - ArchLinux:
https://wiki.archlinux.org/title/msmtp
último acceso: 17/1/2021

[50] Object Management Group - Visual Paradigm Online - Object Management Group:
https://online.visual-paradigm.com/drive/#diagramlist:proj=0&dashboard
último acceso: 17/1/2021

[51] Daniel Matus - instalación de apps externas - digitaltrends.com:
https://es.digitaltrends.com/guias/instalar-app-android-fuera-play-store/
último acceso: 17/1/2021