KEEPY

Grupo 6

- *Miguel Ángel Martínez Mateos*
- *Pablo Rubio Noguera*
- *Alberto Gil Rodríguez*
- *Lucas de Miguel Anguita*
- *Jon Aizkorreta Carro*

ÍNDICE:

1) MOTIVACIÓN

Keepy es un producto desarrollado por estudiantes de la Universidad Politécnica de Madrid del Grado de Ingeniería de Tecnologías y Servicios de las Telecomunicaciones.

Todo el mundo se ha visto en alguna de las siguientes situaciones:

- Riesgo al volver a casa por la noche en zonas marginales.
- Situaciones de peligro durante el día como atracos o agresiones.
- Sufrir algún tipo de accidente ya sea al practicar deporte de alta montaña, por ser una persona dependiente o por diversos casos distintos.

O bien ha sentido miedo de verse en esta situación así.

KEEPY busca evitar estas situaciones mediante la tecnología, permitiendo ser socorrido a tiempo. Nuestro producto nace de pequeñas ideas sugeridas durante las primeras reuniones de brainstorming. Queríamos encontrar un producto que cumpliese las siguientes **metas**:

- Producto de fácil acceso.
- Un producto muy útil que soluciona problemas existentes en la sociedad.
- Producto simple y con un diseño estético.
- Un producto que se pudiese ajustar a las necesidades económicas de nuestros clientes.
- Mejorar algún producto ya existente.
- Proyecto tecnológico.

A partir de tener la idea y distribución de nuestro producto, queremos ofrecer un servicio amplio y abierto a nuestros clientes, los cuales podrán entrar en contacto con la empresa a través de las múltiples redes sociales.

2) CARACTERÍSTICAS DEL PRODUCTO

Se trata de un dispositivo electrónico conectado a nuestro teléfono mediante conexión inalámbrica Bluetooth Low Energy (BLE). El dispositivo permite enviar una señal a nuestro teléfono, el cual deberá tener instalada una aplicación que interpretará dicha señal como una señal de alarma y enviará a través del servicio de mensajería SMS nuestra ubicación al contacto elegido previamente en la misma aplicación.

a) ESPECIFICACIONES TÉCNICAS.

Primero analizaremos el componente electrónico, como microcontrolador central contamos con la **Wemos D1 Mini ESP32** (puede consultar su datasheet en la bibliografía final para mayor información).

Operating Voltage	3.3 V
Digital I/O Pins	11
Analog Input Pins	1(3.2V Max)
Clock Speed	80/160 MHz
Flash	4M Bytes
Size	34.2*25.6mm
Weight	3g

Dicho microcontrolador se encuentra alimentada por una **batería** de 1000 mAh, esta batería nos ofrece una autonomía más que suficiente, alimentando a la placa principal y ofreciendo una tensión de entrada de 3.3V. La batería está pensada para ofrecer una duración suficiente para que el cliente no tenga que preocuparse de estar cargando el producto.

La batería, para garantizar un perfecto funcionamiento y protección del sistema completo, ha sido conectada a un **battery shield** a la placa. Nos permite alimentar a la placa con una tensión de 3.3V y corriente máxima de 1A, además el shield tiene incorporado un módulo de carga de la batería. La batería queda conectada y se carga introduciendo un cargador con conector micro de tipo B.

b) MANUAL DE INSTRUCCIONES.

El uso de nuestro dispositivo está destinado a aquellas personas que tras alguno de los problemas antes mencionados se encuentra sola y decide mandar su ubicación, para ello debe haber configurado la aplicación previamente de la siguiente forma:

Cuando te instalas la aplicación por primera vez, ésta te pedirá permisos de ubicación, de acceso a tus contactos y al servicio de mensajería SMS. El siguiente paso sería elegir tu contacto de emergencia a través del botón que ofrece la interfaz de usuario con el texto *elija un contacto de emergencia*, tras pulsarlo se mostrará una lista con todos los contactos almacenados en la agenda del dispositivo móvil, en la cual habrá que seleccionar el contacto al que será enviado el mensaje pulsando sobre él.

Tras seleccionar el contacto habrá que permitir la conexión entre el dispositivo móvil y el producto, para ello, deberá habilitar el bluetooth del móvil y a través del botón *conectar* seleccionar el dispositivo de entre la lista ofrecida (si no ha sido modificado, el nombre de serie es KEEPY_BT).

Nuestra placa está encendida en todo momento, pero no está conectada todo el rato. Esto es porque la placa se encuentra en modo DeepSleep, más tarde explicaremos los beneficios de esta funcionalidad. Para salir de este modo, vale con pulsar una vez el botón, en ese momento se conecta a nuestro móvil y espera durante un tiempo a un nuevo estímulo de entrada, si tras un tiempo no detecta ninguna pulsación vuelve a entrar en modo DeepSleep.

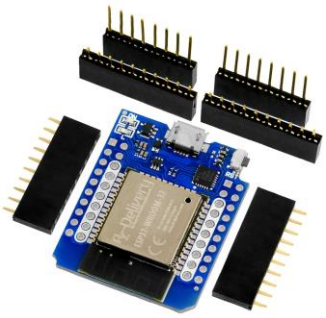
Cuando estamos en modo normal, basta con pulsar 5 veces consecutivas para mandar la señal de emergencia, es importante no dejar demasiado tiempo entre pulsaciones ya que en este caso se detectarán como pulsaciones erróneas y se descartarán con el fin de evitar pulsaciones accidentales.

Tras estas pulsaciones, se envía una señal al teléfono el cual se encargará de mandar un SMS al contacto elegido.

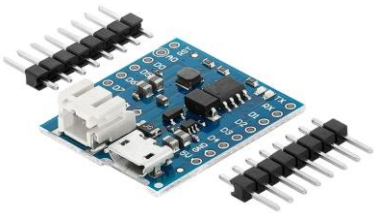
3) Diseño del producto

a) Diseño Hardware

Nuestro producto está formado por la placa de desarrollo Wemos D1 Mini ESP32, una placa que nos ofrece todas las necesidades del producto a un precio muy competente. Es una placa que permite conexiones Bluetooth Low Energy (BLE).



En el apartado de la autonomía se ha optado por el uso de una batería de 1000 mAh y 3,7V de Li Po, ofreciendo a nuestro producto, en utilización del modo Deep Sleep de la ESP32, una autonomía calculada de más de 3 meses.



Para conectar la batería de manera segura, hemos optado por introducir un battery shield de la Wemos ESP32. Este shield viene incluido con un módulo de carga con adaptador micro USB. El shield protege nuestra placa de altas temperaturas que puedan dañar a la placa.

Necesitaremos también un botón de 3 pines para mandar la señal desde la placa hasta nuestro dispositivo. Este botón ofrece la ventaja frente a los de dos pines de evitar rebotes alternando el paso de VCC y GND.

b) Diseño Software

El diseño software consta de dos partes: un código desarrollado en Arduino y una aplicación Android implementada mediante la herramienta *AppInventor*.

Primero, en el código Arduino, proporcionado al microcontrolador en C++, se implementan las siguientes características:

- Configuración de la conexión bluetooth e implementación del sistema de envío de mensajes.
- Desarrollo del protocolo de socorro.
- Extracción de la información sobre la capacidad de la batería.
- Implementación del protocolo Deep Sleep para reducir el consumo de batería.

La configuración de la conexión bluetooth está basada en las librerías proporcionadas por el fabricante: "BluetoothSerial.h" y <esp_bt.h>. Con estas librerías se permite enviar mensajes por el puerto *Serial BT.* Estos mensajes se envían desde el método *enviarMensaje(int code),* dependiendo del código que reciba, envía información del estado de la batería o el código de socorro, el teléfono recibirá esta información y llevará a cabo los protocolos correspondientes.

El protocolo de socorro se lleva a cabo en los siguientes métodos: *pulsador()* y *newPulse()*.

El método *pulsador()* se encarga de leer si el botón está pulsado o no. También se encarga de evitar rebotes: si las pulsaciones se están produciendo en un margen de tiempo menor de 50ms se descartan. Cuando se detecta que el botón está pulsado se llama a *newPulse()*, el cual cuenta pulsaciones, si se producen 5 pulsaciones en menos de 10 segundos comienza el protocolo de auxilio: se llama a *enviarMensaje(int code)* con el código *Alarm_COD* y se envía el mensaje de socorro al teléfono para que ejecute las siguientes fases y acabé enviando un mensaje de auxilio con la ubicación de teléfono.

Para saber la capacidad de la batería se ha implementado el siguiente método. Por el pin 35 de la placa se obtiene la información del voltaje que está proporcionando la batería, se le aplica un factor de conversión: a la tensión medida se le asigna un valor entre 0 y 4095, en el que 0 V corresponde a 0 y 3,3 V corresponde a 4095. Los valores medidos se enmarcan en unos márgenes, y dependiendo del rango en el que se encuentre se informa de un porcentaje correspondiente llamando a *enviarMensaje(int code)* con el código *Bat_COD*.

Por último, el método *deepSleep()* configura como salir y entrar de este modo de bajo consumo. El dispositivo sólo entrará en modo sueño profundo si ha pasado más de 1 minuto sin usarse. Para salir de este modo se ha configurado el pin de entrada del botón del pulsador de la siguiente forma:

esp_sleep_enable_ext0_wakeup(GPIO_NUM_34,1);

El '1' significa que se despertará al dispositivo cuando el botón se pulse (*1 = High, 0 = Low*).

Por otro lado, la aplicación ha sido desarrollada en **AppInventor**, una herramienta online de fácil uso que permite el desarrollo de aplicaciones Android de manera simple y sencilla. A continuación, mostramos el resultado final de nuestra APP.

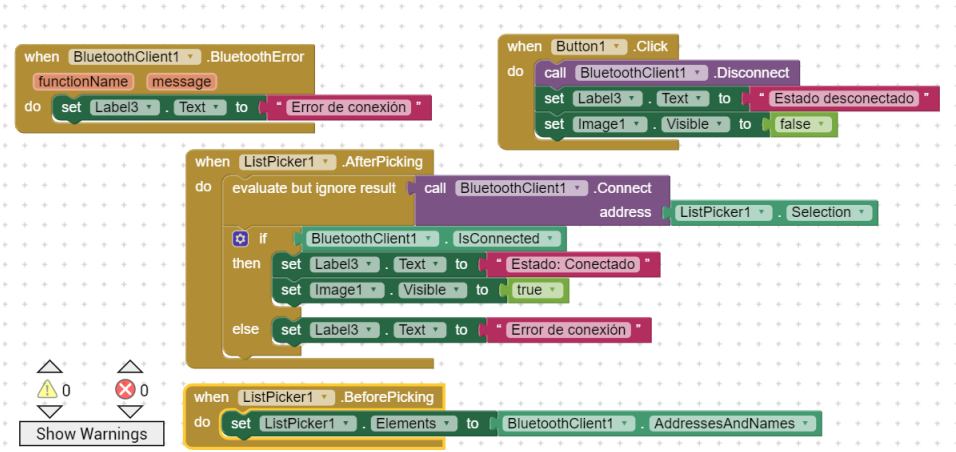
Para que Keepy funciona necesita lo siguiente:

- Comunicación Bluetooth
- Mensajes SMS
- Sensor de Localización
- Base de datos
- Botones
- Reloj
- Imagen
- Etiqueta

El código se divide en dos partes: Una asociada a la vinculación de nuestro microcontrolador con el móvil mediante bluetooth y otra que se encarga de reconocer la información recibida desde la ESP32 para enviar mensajes por SMS.

VINCULACIÓN MICROCONTROLADOR-MÓVIL

Para poder conectar vía bluetooth ambos dispositivos, es necesario de un botón (conectar), que, al pulsarlo, muestra una lista de los dispositivos bluetooth que ya han sido guardados en nuestro dispositivo móvil. Seleccionamos aquel identificador que pertenezca a la ESP32, y a continuación, nos aparecerá una etiqueta con el siguiente texto "Estado conectado" que indica la conexión entre ambos dispositivos. Si por cualquier razón el dispositivo se desconecta de manera accidental o no logra conectarse correctamente, en la etiqueta aparecerá "Error de conexión". Por último, si presionamos el botón "desconectar", desemparejaremos los dos dispositivos. Todas estas funciones aparecen en la foto siguiente.



Como podemos ver al pulsar sobre nuestro "ListPicker1", nos aparecerá la lista ya mencionada, y una vez elijamos el identificador, conectará el dispositivo a la key seleccionada. Como se ha mencionado antes, en caso de error, la app nos avisará y si pulsamos el "Button1" nos desconectaremos de la ESP32.

GESTIÓN DE INFORMACIÓN RECIBIDA Y ENVÍO DE SMS

Nuestra app dispone de un reloj para la comprobación de información recibida desde el microcontrolador cada 1 segundo. Durante este chequeo cíclico, la API debe primero:

Ver primero si ha llegado algún tipo de información, y en caso de que lo haya, el tipo de mensaje recibido. Los mensajes recibidos son de dos tipos, o bien de nivel de batería, o bien de envío de un SMS.

La app dispone de una serie de variables locales con un número distinto (por ejemplo, el envío de un mensaje SMS es un 10 y el nivel de batería agotado es un 20). Durante un periodo de reloj, la app compara el mensaje recibido con las variables locales para ver si debe cambiar el nivel de batería del dispositivo(cambiar la imagen de la batería que aparece en la app) o si bien debe enviar un mensaje.

Las variables que indican el nivel de batería son:

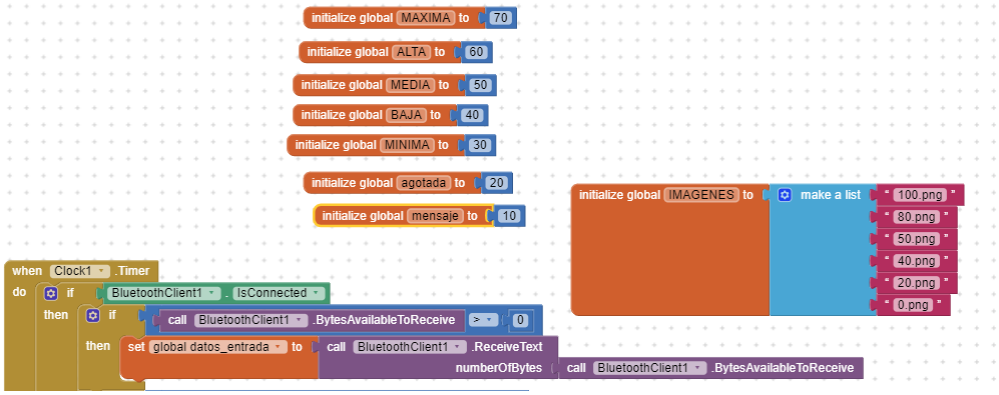
Batería	Mensaje
Máxima	70
Alta	60
Media	50
Baja	40
Mínima	30
Agotada	20

Si Keepy recibe un mensaje que concuerde con alguno de estos, lo que hará será seleccionar de una lista de imágenes aquella que corresponda con el mensaje recibido y cambiar la imagen que aparece en la API.

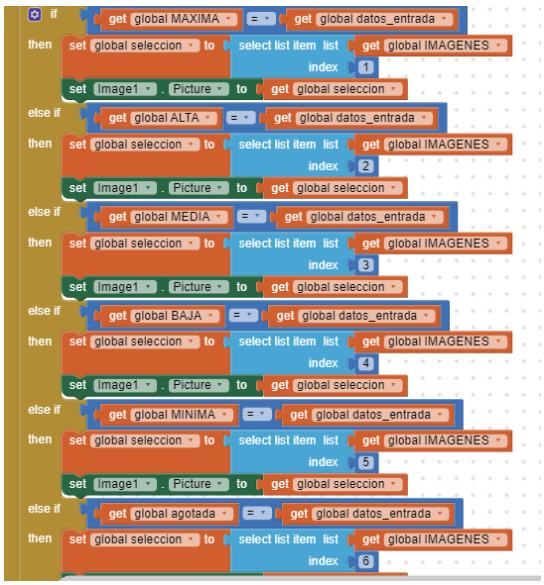
Por último, si se recibe un mensaje de envío de SMS (un **10**), se crea una lista(que es la que se envía a nuestro contacto de emergencia) con la siguiente estructura:

"SOS estoy en:" + "https://www.google.es/maps/place/" + "Latitud,Longitud"

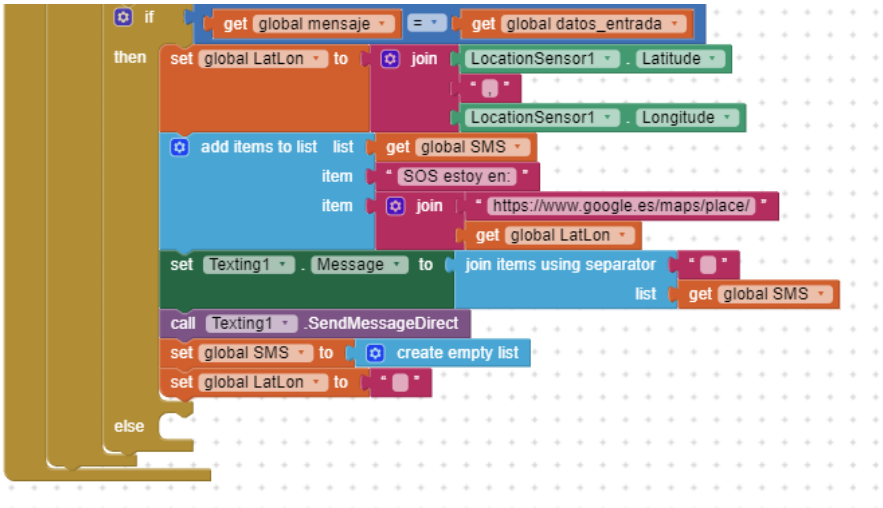
Dónde Latitud y Longitud corresponden a nuestra posición actual con un pequeño margen de error. La posición se detalla gracias al GPS del móvil Android.



Variables locales, lista de imágenes y comprobación de la llegada de la información a nuestra app.



Chequeo nivel de batería



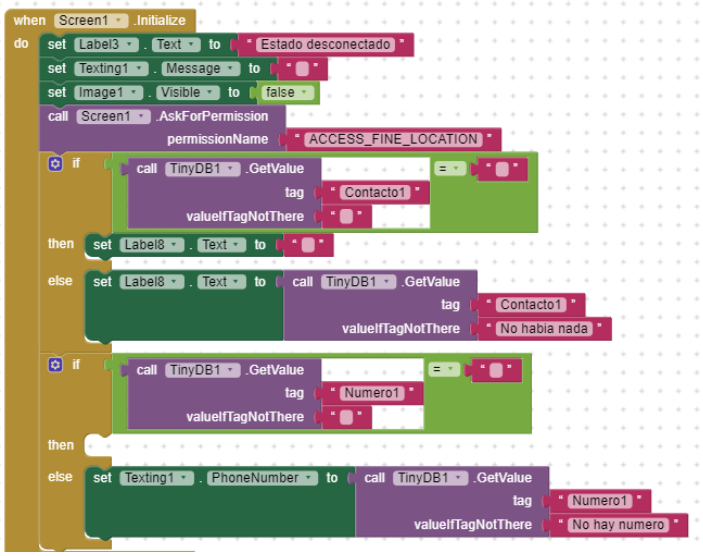
Envío de mensaje por SMS

FUNCIONALIDADES EXTRAS

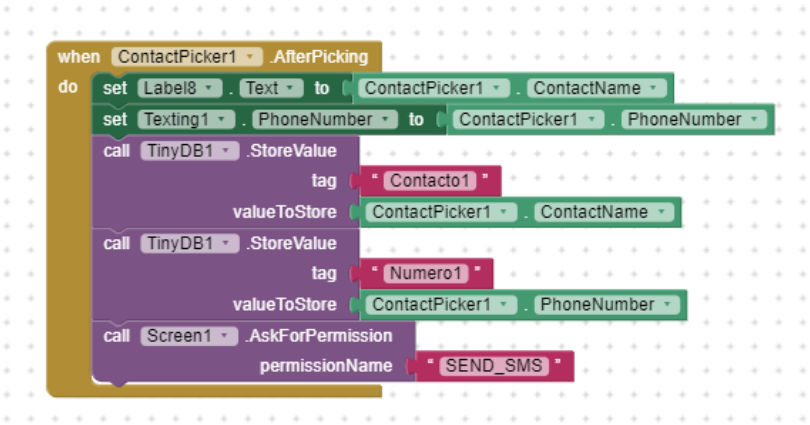
Además de todo lo descrito anteriormente, nuestra API debe disponer de un botón para elegir los contactos de emergencia que él/ella quiera. También debe aceptar los correspondientes permisos para que pueda usar la app sin violar su privacidad.

Cuando pulsamos el botón de contacto de emergencia, keppy nos pide el acceso a nuestra agenda, y una vez hemos elegido el contacto, guardamos tanto nombre como número de teléfono en la base de datos, así como aparece a la derecha del botón el nombre del contacto seleccionado. Finalmente, nos pide acceso a la mensajería de nuestro móvil para en un futuro poder enviar un mensaje.

Al iniciar nuestra API, solicitaremos al usuario el acceso al GPS de su dispositivo móvil, así como la creación de una mini base de datos en la que guardamos el contacto de emergencia del usuario para que no tenga que estar seleccionando cada vez que se meta la app (si no es la primera que se accede a la app, la base de datos se mantiene respecto a la que había cuando entramos por primera vez). Por último, en la etiqueta que muestra la conexión que hay con la ESP32, aparece en "desconectado" y también ponemos en "false" la visibilidad de la imagen de la batería ya que de momento no está el microcontrolador emparejado.



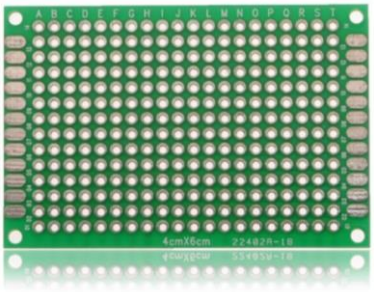
Inicialización app



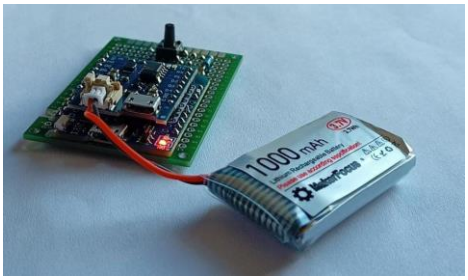
Selección contacto de emergencia

c) FABRICACIÓN

El primer paso para la fabricación de nuestro producto pasa por realizar el montaje sobre una pcb. Este primer prototipo ha sido montado sobre una placa de 4x6cm, pero como comentaremos más adelante tenemos la intención de reducir este tamaño.



La Wemos D1 Mini ESP32 irá unida de forma muy compacta a su correspondiente shield junto con el montaje del botón y la batería estarán unidos gracias a esta pcb. quedando el montaje final de la siguiente forma:



Finalmente, todo el producto estará dentro la siguiente carcasa que hemos diseñado e impreso en una impresora 3D, usando la herramienta de diseño Fusion 360.

El producto final será el siguiente:

4) MERCADO OBJETIVO

Nuestro mercado objetivo estará determinado por distintas situaciones en las que se requiera usar el dispositivo.

Cuando concebimos la idea pensamos en la situación de una chica joven que pudiera prevenir una situación de peligro. Aun así, *Keepy* puede ser útil en muchas más situaciones, accidentes en personas dependientes, en una travesía de montaña o víctimas de agresiones.

El criterio de segmentación del mercado se ve delimitado por género y edad.

Por tanto, nuestra clientela potencial serán mujeres jóvenes de entre 15 y 35 años con teléfono móvil Android. De forma secundaria estará formado por jóvenes de entre 18 y 35 años que les gusten los deportes de riesgo o alta montaña y por personas dependientes como minusválidos o ancianos.

5) NUESTRA EMPRESA

Keepy será el primer producto de la empresa que lleva el mismo nombre, se trata de un startup formado por estudiantes de último año de Ingeniería de tecnologías y servicios de las telecomunicaciones. Los cinco fundadores estaremos en la dirección y puestos de responsabilidad de la empresa.

ANÁLISIS DAFO

Debilidades	Fortalezas
Dependencia total de los proveedores Precio elevado Pocos recursos iniciales Dependencia de capital interno	Equipo humano con gran conocimiento y motivación Confianza en el producto por parte de los clientes Productos y servicios claros y sencillos
Amenazas	**Oportunidades**
Existencia de productos similares en el mercado Idea fácilmente replicable por otras empresas externas	Necesidad existente y demandada por gran cantidad de clientes Nuestro producto resuelve un problema clave en la sociedad moderna Producto con mucho potencial de desarrollo y evolución.

PROVEEDORES Y SERVICIOS

A continuación, expondremos todos los costes que se producen para poder crear nuestro producto, además de los proveedores. Cabe recalcar que pondremos precio de una unidad y de 1000 unidades, al ser una empresa con expectativas de producción masiva, haríamos los pedidos por la segunda opción.

Material	Proveedor	Precio por unidad	Precio estimado 1000 Uds.
Shield	AZ-Delivery	1,87€	748€ - 1309€
Placa de desarrollo	Aliexpress	2,86€	1144€ - 2002€
Batería 100mAh	MakerFocus	4€	1600€ - 2800€
Proto PCB	Sourcing map	1,096€	438,4€ - 767,2€
Botón	Mouser	0,13€	52€ - 91€
resistencia	Mouser	0,11€	44€ - 77€
Cable	Mouser	1,50€/m * 0,2m = 0,3€	120€ - 210€
Tornillos	Walfront	2€/30uds = 0,067€	3 * 55€ = 165€
Filamento PLA impresión 3D	Sunlu	23€/Kg*0,005Kg=0,115€	23€/Kg * 5Kg=115€
Servicio PlayStore	Google Play	20,71€ (darse de alta como desarrollador)	

El precio de venta al por mayor se ha estimado con un descuento de entre el 30% y el 60% aunque generalmente es de un 50%. Este cálculo no es aplicable a los tornillos ya que se vende la bolsa de 1000 unidades por 55€ (serán necesarias 3 bolsas) y la compra mínima es la bolsa de 30 unidades, ni al filamento PLA ni a los servicios de PlayStore.

La fabricación de 1000 unidades tiene un coste de 5483,71€, por tanto, el coste por unidad es 5,484€.

DISTRIBUCIÓN

Entre los costes añadidos que encontramos a la hora de ofrecer nuestro servicio se encuentran:

- Envío y embalaje del producto completo: El coste medio por cada envío por Correos desde Madrid hasta cualquier lado de la península es de 4€ de media. Y para el embalaje, tendremos un coste por caja de 29 céntimos.

Sin embargo, podríamos vender el producto a través de Amazon ya que de esta forma los gastos de envío podrían ser nulos.

CONCLUSIÓN Y PRECIO FINAL DE VENTA

Hemos decidido por tanto lanzar el producto al mercado con un precio de venta de 10,95€.

Con lo que tendríamos un beneficio de **5,466€/producto**. Que incluyendo los gastos de envío supondría un coste final para el usuario de 14,95€. Sin embargo, si vendiéramos el producto a través de Amazon el precio del producto final para el usuario sería de **10,95€**, un precio mucho más competente para el mercado.

6) POSIBLES MEJORAS Y TRABAJO FUTURO

El producto que sacamos es una primera versión, con el tiempo y tras recibir *feedback* por parte de los usuarios, iremos añadiendo mejoras al producto para así proporcionar la mejor experiencia posible para nuestros compradores.

Dichas mejoras irán enfocadas a distintas partes de nuestro proyecto, desde la aplicación hasta cambios en el diseño hardware del producto.

APLICACIÓN

Mejorar la interfaz de usuario y ofrecer una aplicación más agradable estéticamente con transiciones o animaciones entre los diferentes menús y apartados.

IOS

Desarrollando una aplicación para iOS estaríamos abriendo nuestro mercado a un gran número de nuevos posibles usuarios que utilizan móviles de la marca Apple.

FUNCIONALIDADES

Entre las funcionalidades que podemos añadir se encuentran, indicadores de batería y señal más complejos. Conseguir que el producto sea independiente de nuestro teléfono para enviar mensajes de emergencia. Baterías con carga inalámbrica para que los compradores no tengan que enchufar el cargador y así evitar puntos débiles en el sistema.

Otra funcionalidad muy interesante sería que una de las líneas del producto incluya un Acelerómetro con el fin de detectar caídas fuertes y en este caso enviar el mensaje de socorro automáticamente. Siendo un producto más orientado al alpinismo o personas mayores, pues podría darse la situación de que la caída sea muy grave y el individuo sea incapaz de pulsar el botón de emergencia.

COLABORACIÓN Y ACUERDOS

Se tratará de lograr una colaboración con los organismos públicos pertinentes para que los mensajes SMS sean tramitados por los diferentes cuerpos de seguridad (policía, guardia civil, bomberos...). También se podría llegar a un acuerdo con empresas de seguridad como por ejemplo *Prosegur* o *Segurservi* para incorporar el producto en su cartera de servicios a cambio de una remuneración.

TAMAÑO

Por último, una de las mejoras más clara y sencilla sería la disminución del tamaño del producto, para así poder convertirlo en un producto más estilístico, ergonómico, ligero y llamativo para el usuario.

FIABILIDAD

Desarrollar algún método de detección de calidad de la conexión para avisar al usuario en caso de fallo y restablecer la conexión automáticamente y lo más rápidamente posible.

7) CONCLUSIONES

A día de hoy falta camino para llegar a ser un producto competente en el mercado, pero cabe apreciar que implementando y desarrollando las mejoras mencionadas en los apartados anteriores y consiguiendo un formato de distribución adecuado seremos capaces de reducir costes y conseguir un producto que no pase desapercibido.

En cualquier caso, independientemente de lo competente que pueda ser nuestro producto en el mercado, el valor académico de este proyecto ha sido inmenso a nivel personal. Hemos podido desarrollar competencias, por primera vez en el grado, a muchos niveles que nos serán muy útiles en el futuro. A destacar:

- Comunicación entre teléfono y Arduino vía Bluetooth.
- Desarrollo de aplicaciones Android.
- Desarrollo de software Arduino.
- Diseño de piezas en 3D.

Por tanto, querríamos finalizar la memoria agradeciendo esta oportunidad a todos los profesores de la asignatura, especialmente a Antonio Pérez Serrano ya que sin su consejo y ayuda no habría sido posible este proyecto.

8) BIBLIOGRAFÍA

- Wemos D1 Mini Página fabricante
- Página Web Compra Wemos D1 Mini
- Wemos D1 Mini Shield
- https://github.com/wemos/D1_mini_Examples/blob/master/examples/01.Basics/ReadAnalogVoltage/ReadAnalogVoltage.ino
- https://randomnerdtutorials.com/esp32-adc-analog-read-arduino-ide/
- https://programarfacil.com/blog/arduino-blog/medidor-de-carga-de-baterias-pilas-arduino/
- https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/sleep_modes.html
- https://randomnerdtutorials.com/esp32-external-wake-up-deep-sleep/#:~:text=External%20wake%20up%20is%20one,up%3A%20ext0%2C%20and%20ext1
- https://www.instructables.com/ESP32-Deep-Sleep/
- https://diyprojects.io/esp32-arduino-code-for-deep-sleep-and-wake-ups-timer-touch-pad-gpio/#.YIWYAZAzZPZ
- Página de embalaje del producto
- Servicio de envío de paquete