ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

IMPLEMENTACIÓN DE UN SISTEMA DE LOCALIZACIÓN PARA PERSONAS EN SITUACIONES DE RIESGO MEDIANTE UNA INTERFAZ WEB Y APLICACIÓN EN ANDROID

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGOS ELECTRÓNICA Y TELECOMUNICACIONES

ADRIAN JOHAO BENAVIDES ALMACHI
adrian.benavides@epn.edu.ec

DENNIS EDISON VILLAMARÍN PERACHIMBA
dennis.villamarin@epn.edu.ec

DIRECTOR: ING. LEANDRO ANTONIO PAZMIÑO ORTIZ, MSC.
leandro.pazmino@epn.edu.ec

CODIRECTOR: ING. MÓNICA VINUEZA RHOR, MSC.
monica.vinueza@epn.edu.ec

QUITO, ENERO 2021

CERTIFICACIÓN

Certificamos que el presente trabajo fue desarrollado por los señores Benavides Almachi Adrian Johao y Villamarín Perachimba Dennis Edison bajo nuestra supervisión:

Ing. Leandro Pazmiño, MSc.	**Ing. Mónica Vinueza, MSc.**
DIRECTOR DEL PROYECTO	**CODIRECTOR(A) DEL PROYECTO**

DECLARACIÓN

Nosotros Adrian Johao Benavides Almachi con CI: 172674520-9 y Dennis Edison Villamarín Perachimba con CI: 172767391-3 declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y que hemos consultado las referencias bibliográficas que se incluyen en este documento.

Sin prejuicio de los derechos reconocidos en el primer párrafo del artículo 144 del Código Orgánico de la Economía Social de los Conocimientos, Creatividad e Innovación – COESC-, somos titulares de la obra en mención y otorgamos una licencia gratuita, intransferible y no exclusiva de uso con fines académicos a la Escuela Politécnica Nacional.

Entregamos toda la información técnica pertinente, en caso de que hubiese una explotación comercial de la obra por parte de la EPN, se negociará los porcentajes de los beneficios conforme lo establece la normativa nacional vigente.

Adrian Johao Benavides Almachi

Dennis Edison Villamarín Perachimba

DEDICATORIA

Dedico esta tesis a mis padres Franklin y Beatriz, quienes indudablemente me han apoyado en cada uno de mis pasos, poniendo todo su esfuerzo para lograr la motivación que permite que hoy en día sea la persona que soy y por su amor incondicional.

~ Dennis

AGRADECIMIENTO

Primero y como más importante, me gustaría agradecer sinceramente a mi director y tutor de Tesis, Ing. Leandro Pazmiño, por su esfuerzo y dedicación.

También me gustaría agradecer los consejos recibidos a lo largo de los últimos años por otros profesores de la Escuela de Formación de Tecnólogos de la EPN, que de una manera u otra han aportado su granito de arena a mi formación.

A mi hermana, abuelos y tíos quienes de alguna manera aportaron a la culminación de este proyecto y formaron parte de mi desarrollo personal.

Y, por último, pero no menos importante, estaré muy agradecido a mi compañero de trabajo, Adrian, al cual considero una excelente persona y profesional.

Para ellos, muchas gracias por todo.

~ Dennis

DEDICATORIA

Como dignos merecedores de cualquier esfuerzo y fruto bueno que de mi pudiera provenir, dedico este trabajo a Juan y Martha, mis padres; y a Erick, mi hermano y primer maestro.

~ Adrian

AGRADECIMIENTO

Tantos a quienes agradecer en tan corto espacio.

Para empezar mi sentido de agradecimiento a aquellas personas quienes, por lo general, resultan pasadas por alto pero que son, a mi manera de ver, parte importante del desarrollo del estudiante. Entre estas encontramos a los servicios gastronómicos, como las «poliburguers» de la esquina, los cevichochos de «el Duro», «la Seño» del arroz de tarrina, así como otros, que, en no pocas ocasiones, fueron el soporte para las maratones de clases.

Otro reconocimiento a esos compañeros de aula que estuvieron dispuestos a compartir sus tareas y trabajos sin pedir algo a cambio, pero, aún más a aquellos quienes no lo hicieron, que sin querer incentivaron el trabajo y la dedicación propia.

Mención especial a los y las «Inges», promotores del conocimiento y de manera general a toda la comunidad educativa de mi querida «Poli».

Finalmente, y por tanto más importantes, gracias Pa, Ericksin, Ma y especialmente, gracias, Jah.

~ Adrian

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ECUACIONES

RESUMEN

El proyecto desarrollado en este documento describe el análisis, diseño, desarrollo e implementación de un sistema de localización para personas en situaciones de riesgo mediante una interfaz web y una aplicación en Android.

En la sección introducción se presenta el objetivo general, objetivos específicos y la justificación del tema seleccionado sumados a un referente teórico que abarca los tópicos más destacados para el desarrollo del proyecto. Además, se destacan las características de los dispositivos del sistema.

El segundo apartado describe la metodología, tipo de investigación, actividades y secuencia sistemática usada para, en conjunto, facilitar la ejecución y el cumplimiento de las etapas encaminadas a la implementación del sistema de localización.

En la sección de resultados y discusión se presentan los hechos más relevantes producto de la ejecución de las distintas actividades llevadas a cabo en la realización del sistema. Se detallan temas como: las necesidades del sistema, los elementos utilizados, los esquemas y diagramas de funcionamiento y el desarrollo de las interfaces. Se exponen, además, las pruebas de funcionamiento del prototipo.

A continuación, se presentan las conclusiones inferidas luego de haber cumplido con los objetivos propuestos para el proyecto integrador, en base a la experiencia adquirida en la ejecución de este. Finalmente, en los apartados referencias bibliográficas y anexos se presentan las fuentes de apoyo en la materialización del proyecto y sus respaldos documentales.

PALABRAS CLAVE: sistema de localización, Arduino, Android.

ABSTRACT

The project approached in this document describes the analysis, design, development and deployment of a localization system for people at risk using a web interface and an Android application.

The introductory section shows the general objective, specific objectives and justification of the chosen topic including the theoretical reference that reviews the highlighted topics for developing the project. Also, the devices' features are mentioned.

The second section describes the methodology, type of investigation, the activities and systematic sequence used for easing the realization and fulfilment of the phases required to get the job done.

The results and discussion section presents the most significant facts as the result of performing the activities to build the system. The system needs, the devices used, operating sketches and diagrams, and development of interfaces are detailed. Additionally, the test runs are included.

After fulfilling the project objectives, the conclusions obtained based on the experience gained by building the project are presented in this section. Finally, the references and annexes sections expose the sources of support and the documentary evidence.

KEYWORDS: *localization system, Arduino, Android.*

1 INTRODUCCIÓN

1.1 Marco teórico

GPS (*Global Positioning System*)

El sistema de posicionamiento global o GPS, es un sistema de utilidad perteneciente a Estados Unidos operado por la Fuerza Aérea de dicho país. Además, es capaz de proveer los servicios de: posicionamiento, navegación y sincronización (*PositioningNavigationTiming*) [1]. La funcionalidad del sistema radica en el uso de la constelación de al menos 24 satélites (31 en total) que rotan alrededor del planeta en un lapso de 12 horas emitiendo señales continuas de navegación. Estos satélites están situados en 6 planos inclinados exactamente a 55 grados [2].

Para que el sistema opere, es necesaria la interacción de tres segmentos: espacio, control y usuario como se observa en la Figura 1.1. El segmento espacio comprende en su totalidad el arreglo de satélites mencionado, el uso de una frecuencia atómica estándar, el procesador para el almacenamiento de información, la señal de ruido pseudoaleatoria que genera la señal de alcance y la antena de transmisión de banda L, son los componentes de cada uno de los satélites ubicados en este segmento. Los elementos que representan el segmento de control son las estaciones de control, antenas de tierra y la red de estaciones de monitoreo. La función de este conjunto es monitorear y controlar todos los aspectos referentes al segmento espacio. Finalmente, el segmento de usuario está integrado por una antena y un receptor/procesador.

Los dispositivos receptores adquieren la señal proveniente de cuatro o más satélites para medir su tiempo de propagación y cambios de frecuencia de *Doppler* para luego de procesar esta información obtener los datos de posición en tres dimensiones incluyendo la velocidad. El equipo de usuario puede incluir otros sensores o sistemas con el fin de mejorar la precisión en entornos dinámicos [2].

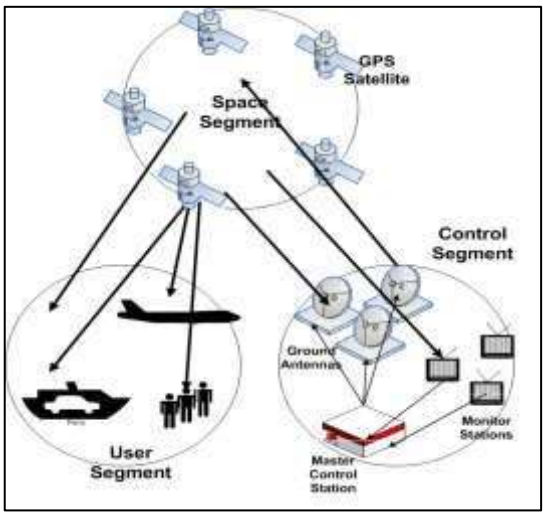


Figura 1.1 Segmentos de red GPS [26]

Una de las ventajas de este sistema de posicionamiento es su elevada precisión, que como ya se mencionó, puede ser mejorada mediante la inclusión de otros sistemas. Otra característica favorable de GPS es su resistencia a climas desfavorables lo que garantiza su operación en un 95% del tiempo [1].

Transmisión GPS

El envío del mensaje de navegación se realiza desde los satélites por medio de una señal modulada en BPSK en la primera banda (L) de radiocomunicaciones espaciales, comprendida entre los 1.525 a 1.710 (GHz). Las frecuencias utilizadas dependen del tipo de satélite del que proviene la señal. En la Tabla 1.1 se muestran las frecuencias centrales correspondientes a las bandas que usan los satélites. La o las frecuencias en las que operen los dispositivos GPS, tanto receptores como satélites, depende del diseño de estos, pueden existir dispositivos que trabajen solo con la portadora L1, en L1 y L2 o con las tres portadoras. [2].

Tabla 1.1 Frecuencias centrales banda L [2]

Portadora L	Frecuencia Central [MHz]	Límite inferior	Límite superior
L1	1575.42 ± 15.345	1559	1610
L2	1227.6 ± 15.345	1215	1300
L5	1176.45 ± 12	1164	1215

Estructura de la señal GPS

Puesto que las señales provenientes del segmento espacio son moduladas poseen una estructura determinada: una señal portadora que puede contener otras señales con parámetros adicionales de acuerdo con la frecuencia central (L1, L2), señales de ruido PRN (*Pseudo Random Noise*) que permiten identificar el satélite del que proviene la señal y el mensaje de navegación.

GPRS

La tecnología GPRS (*General Packet Radio System*) aparece como una evolución a la segunda generación de redes celulares en la que se propone una mejora a la transmisión de datos antes que a las señales de voz. Dicha mejora se hace con el propósito de volver a los dispositivos, que trabajen en GSM, compatibles con la navegación en Internet, así como con redes LAN y WAN [3].

La principal característica de GPRS es el envío de información por medio de conmutación de paquetes. En esta modalidad la información es dividida en paquetes de determinada longitud en los que se incluyen datos de destino y orden, llamados datos de señalización. Cada paquete viaja por caminos distintos de la red según la disponibilidad de estos y son liberados al terminar su trayecto. De esta manera no se monopolizan los canales de la red lo que vuelve más eficiente a GPRS frente a GSM [4].

Debido al tipo de transmisión que utiliza conmutación de paquetes, GPRS resulta compatible con aplicaciones en donde el tamaño de los mensajes es pequeño o poco frecuente, y también cuando el envío de datos es intermitente, como en el caso de navegación en Internet por medio de WWW [4].

GPRS trabaja compartiendo la arquitectura de 2G y además incluye elementos como el SGSN, GGSN y PCU, responsables de la conectividad de esta tecnología [4]. La Figura 1.2 permite observar los tres principales segmentos de la arquitectura divididos en: estación móvil (MS), sistema de estación base (BSS), subsistema de red y conmutación (NSS).

El primer segmento correspondiente a MS integra los equipos de usuario (SIM, equipo móvil, terminal móvil) y cumple con las funciones de transmisión, recepción, autenticación y codificación de canal.

En el segmento BSS interactúan los elementos BTS (*Base Transceiver Station*), BSC (*Base Station Controller*) y PCU (*Packet Control Unit*). Estos elementos cumplen con las

tareas de transmisión por medio de antenas y minimizar los problemas de transmisión por aire (BTS), conexión entre BTS y MS, control de los recursos de la radiotransmisión, manejo de cambios de celdas de usuarios (*handover*), recolección de datos (BSC), control de paquetes y canales de datos (PCU).

Finalmente, el segmento NSS divide la arquitectura para los sistemas GSM y GPRS. Por el lado de GSM, el MSC maneja los *gateways* de conexión a otras redes, usualmente a la PSTN. Mientras que en GPRS se utilizan esencialmente dos elementos que cumplen las funciones de entrega de paquetes al MS (SGSN) y conexión con redes de datos externas (GGSN), por lo general, Internet [4] [5] [6].

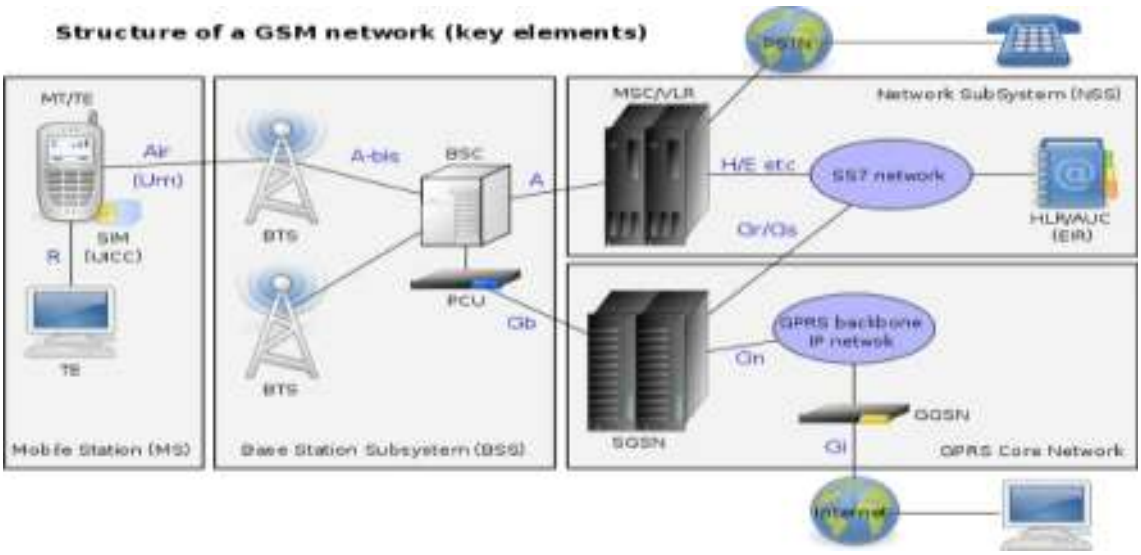


Figura 1.2 Arquitectura de red GSM/GPRS [27]

Arduino

Es una plataforma de desarrollo electrónico basada en el concepto de *open-source* enfocada en el fácil manejo de *software* y *hardware.* El producto principal de Arduino son sus placas de desarrollo que permiten al usuario implementar distintos proyectos que van desde encender *LEDs* o motores hasta, por medio de un set de comandos, realizar impresiones 3D, manejo de aplicaciones de IoT (*Internet of Things*) o distintas ideas más.

Las placas Arduino se componen básicamente por un microcontrolador y varios periféricos de entrada y salida, que dependerán del modelo de placa que se esté utilizando. El manejo de la placa Arduino se realiza mediante un *software* propio llamado Arduino IDE y un lenguaje de programación compuesto por tres partes principales que son: funciones, variables y estructura, correspondientes al esquema de programación C++ [7].

Batería

Los dispositivos capaces de convertir la energía química en eléctrica por medio de reacciones químicas llevan el nombre de baterías. Son compuestas, generalmente, por una o más celdas a su vez conformadas por: ánodo, cátodo y un electrolítico que depende del tipo de batería. El tipo de electrolítico del que esté constituido la betería es el responsable de designar el tipo de la reacción, en otras palabras, la vuelve primaria (no recargable) o secundaria (recargable) [8] [9].

La energía que es proporcionada por una batería está directamente ligada a los materiales que contenga. Además, cada celda es caracterizada por un valor de capacidad que obedece a la cantidad de material reactivo que contiene. Así, por medio de estos valores teóricos, el potencial de los electrodos medidos en voltios y la capacidad de las celdas expresada en culombios o amperios/hora, se puede conocer un valor aproximado de energía contenida por la batería. El valor real varía en función de parámetros como el peso y volumen de los materiales, los elementos no reactivos usados para la construcción, el balanceo de los materiales, el valor de descarga de la batería entre otros [9]. La Ecuación 1.1 expresa la energía teórica entregada por una batería dada en vatios/hora.

$$E_{teor} = V_{teor} * C_{teor}$$

Ecuación 1.1 Valor teórico de una batería

Donde:

E_{teor}: (Wh) energía teórica

V_{teor}: (V) voltaje teórico

C_{teor}: (Ah) capacidad teórica

Batería de litio

En el campo de las baterías secundarias (recargables), el elemento al que se han dedicado mayores esfuerzos para su desarrollo es el litio por diversos motivos de los que destaca el peso.Las baterías de litio o *Li-Ion* se caracterizan por ser capaces de entregar mayor energía con menor peso lo que se traduce en una alta potencia específica (W/kg) y energía específica (Wh/kg). Además, las baterías de iones de litio poseen un bajo efecto memoria, buen rendimiento a altas temperaturas en ambientes con baja presencia de humedad, larga vida útil y son amigables con el ambiente [9] [10].

Batería de Polímero de Litio (Li-Po)

A finales de los años noventa se empezaron a producir baterías basadas en el conjunto Litio-Polímero que con el paso del tiempo y gracias a sus mejoradas características eléctricas y físicas, llegaron a ubicarse en una posición favorecida en el campo de baterías secundarias [11].

Una de las características más representativas de LiPo es su alta gama de formas y tamaños frente a la tecnología de ion de litio. Esto se debe a que el armazón de tipo bolsa aislada por una lámina de aluminio le da flexibilidad al momento de realizar el diseño de la batería. Además de esto, las baterías LiPo pueden ser distinguidas por su densidad energética y auto descarga. El en caso de la densidad energética LiPo ofrece una ligera mejora a las baterías de litio y en respecto a la velocidad de auto descarga tiene ventaja por su relativa baja velocidad. Sin embargo, este tipo de baterías también presentan desventajas y una de las mayores es que requieren un manejo cuidadoso acerca del tipo de carga, descargas, sobrecargas y temperatura de trabajo [11].

2 METODOLOGÍA

El tipo de investigación que caracteriza al presente proyecto es aplicativo, puesto que busca, mediante la aplicación de conocimientos obtenidos previamente y por medio de la ejecución de distintas tareas sistemáticas, dar solución a situaciones que aquejan a cierto grupo [12]. En este caso, el sistema de localización está destinado a personas que se encuentren en situaciones de vulnerabilidad como personas con deficiencia neurológica o menores de edad que requieran supervisión.

El desarrollo del sistema de localización principió desde la obtención y análisis de datos acerca de la situación de personas desaparecidas y las condiciones en las que se suscitan estas, lo que permitió conseguir el enfoque adecuado para el proyecto. Posterior a esto, se procedió a estudiar los aspectos técnicos que requiere el sistema de localización en función del enfoque determinado. Se revisaron temas como: redes de navegación, redes de telefonía celular, sistemas de control, interfaz de usuario, consumo energético y características físicas de los dispositivos de localización del mercado, para en función de los resultados del estudio seleccionar los elementos que mejor se ajusten a estos.

Luego, se procedió a estructurar el sistema por medio de diagramas de flujo que dieron paso al diseño del circuito del dispositivo localizador para después de adquirir los elementos y materiales necesarios, elaborar la programación de la rutina principal y

ejecutar las primeras pruebas de funcionamiento con el fin de que una vez analizados los problemas existentes sean corregidos.

Se examinaron, asimismo, proveedores de *hosting* para levantar los servicios necesarios en donde almacenar los datos que se envíen desde el dispositivo de localización y números de emergencia, así como los archivos de apoyo para las aplicaciones web y Android. Para este propósito se recopiló información relacionada a la creación, manejo y programación de aplicaciones web y Android.

Finalmente, se puso en marcha la fase de pruebas que contó con evaluaciones en distintas etapas del sistema. Se evaluaron aspectos como la conectividad, el envío de datos al servicio de almacenamiento en la nube, la comunicación del dispositivo con las aplicaciones y viceversa, la generación de alertas, la autonomía del dispositivo y pruebas del funcionamiento en general.

3 RESULTADOS Y DISCUSIÓN

3.1 Identificación de los requerimientos para el diseño e implementación del sistema de localización

Necesidades del sistema

De acuerdo con lo descrito en la guía para productos y servicios de la información y comunicación, con el transcurso de los años la población de adultos mayores tiende a tener un deterioro en campos como destreza manual, movilidad, fuerza y resistencia. Los que van acompañados de un declive en la habilidad de procesamiento de información que desemboca en pérdidas de memoria. Este hecho hace que tengan mayor dificultad en el manejo de dispositivos electrónicos y la tecnología en general [13].

Por el contrario, los menores de edad tienden a manejar de mejor manera la tecnología y poseen mayor facilidad para aprender a utilizarla.

En vista que el sistema se destinó a menores de edad y adultos mayores y dado a que las necesidades específicas de estos dos grupos difieren en gran manera, se optó por seguir los principios de “Diseño para todos” de productos y servicios que se base en 7 principios a los que se ligó las necesidades del sistema [13]. Estos principios se detallan en la Tabla 3.1

Tabla 3.1 Principios de "Diseño para todos" [13]

Principio	Descripción
Uso igualitario	Útil para cualquier grupo de usuarios
Flexibilidad de uso	Apto para un amplio rango de habilidades
Simple e intuitivo	Fácil de usar y entender
Información perceptible	Comunicar la información de uso de forma efectiva
Tolerancia a errores	Minimizar riesgos de errores
Bajo esfuerzo físico	Uso con efectividad y el menor esfuerzo
Tamaño y espacio	Manipulable independiente del tipo de usuario o postura

Requisitos del sistema de localización

Como respuesta a los principios de “Diseño para todos” se determinaron los requisitos del sistema como se detallan en la Tabla 3.2

Tabla 3.2 Requisitos del sistema de localización

Principio	Requisito
Uso igualitario	Apto para adultos mayores y menores de edad
Flexibilidad de uso	Interfaz sencilla
Simple e intuitivo	Pocos botones
Información perceptible	Poseer indicadores de funcionamiento
Tolerancia a errores	Resistente
Bajo esfuerzo físico	Ligero, portátil, larga duración
Tamaño y espacio	Tamaño pequeño, fácil de transportar.

Por esta razón se requiere de un sistema de localización que, disponga de un dispositivo rastreador que sea de tamaño pequeño, ligero, fácil de transportar, con un armazón resistente y una batería de larga duración. Este localizador deberá tener un botón que genere alertas a un cuidador de forma inalámbrica y entregue su posición en tiempo real desde distintas localizaciones. Poseerá un selector que permita seguir de forma continua al localizador o no, así como de un *switch* de encendido o apagado. Deberá informar cuando requiera recargar la batería y el modo de operación que se encuentre por medio de indicadores LED.

Del lado del cuidador, el sistema proporcionará una interfaz sencilla para observar el recorrido del localizador. Permitirá configurar un número celular para recibir notificaciones de emergencia independientes de los datos móviles.

Los requerimientos se dividieron en 3 etapas principales: adquisición de datos, almacenamiento y presentación.

Para la etapa de adquisición de datos se requirió un dispositivo compatible con alguno de los sistemas de radionavegación (GPS, Galileo, GLONASS), un microcontrolador que disponga de interfaces seriales, análogas, buenas capacidades de procesamiento, un sistema de alimentación portátil de alto rendimiento, una interfaz de red celular que tolere la conexión de datos móviles con amplia cobertura, y antenas de radionavegación y celular de buena ganancia.

La etapa de almacenamiento requiere de un sitio de alojamiento en la nube o *hosting* estable, de bajo costo, que garantice disponibilidad, bases de datos, un dominio para el acceso independiente a los datos.

En la etapa de presentación se necesita una plataforma de desarrollo para aplicaciones web y *Android* de lenguaje fácil de utilizar. Estas aplicaciones deberán asegurar el seguimiento en tiempo real del localizador y opciones de configuración de números de emergencia en un entorno de desarrollo sencillo e intuitivo.

3.2 Análisis del hardware según los requerimientos identificados para el diseño del prototipo

Elementos del prototipo

Para la selección del sistema embebido se realizó la comparativa que se muestra en la Tabla 3.3

Tabla 3.3 Comparativa sistemas microprocesados *[14] [15] [16]*

Característica	Arduino Pro Mini 3.3 V	Arduino Nano	Raspberry Pi Zero
Voltaje Alimentación	3.3 (V)	5 (V)	5 (V)
Voltaje de Trabajo	3.3 (V)	5 (V)	5 (V)
Microcontrolador	ATmega328	ATmega328	BCM 2835 SOC
Memoria	32 (Kb) Flash	32 (Kb) (Flash)	512 (MB) RAM
Número de pines	22	22	40
Dimensiones	33*18 (mm)	43.2*18.5 (mm)	65*30 (mm)
Precio	4 ($)	6.50($)	10.($)

Como se puede observar en la comparativa la placa de control Arduino Pro mini 3.3 (V) se adapta mejor a las necesidades del sistema puesto que posee las dimensiones y el voltaje de alimentación más reducidos. Además, el número de pines para utilizar son 10 por lo que los 22 pines del Arduino Pro mini dejan solvencia para añadir periféricos. En cuestión de costo se notó que el costo menor es el correspondiente al Pro mini. Se descartan los sistemas Arduino Nano y *Raspberry Pi Zero* por su voltaje de alimentación y dimensiones respectivamente. Las especificaciones técnicas de la placa se encuentran en el **Anexo A1.**

Respecto al sistema de radionavegación, se selecciona al sistema GPS por su mayor avance respecto a sus posibles competidores como GLONASS, GALILEO o *BeiDou* que a pesar de su desarrollo no alcanzan el nivel del sistema americano [17]. En base a esto se seleccionó un módulo de radionavegación GPS, para lo que se contrastaron los modelos de la Tabla 3.4. Aquí se puede observar que el módulo NEO 6M maneja los valores de sensibilidad y tiempo de adquisición óptimos, además de tener las dimensiones más reducidas por lo que fue seleccionado para el sistema de localización. Las demás características técnicas del GPS NEO 6M se pueden hallar en el **Anexo A2**

Tabla 3.4 Comparativa módulos GPS *[18] [19]*

Característica	NEO 6M	SIM808
Interfaces	UART, USB, SPI, DDC	UART
Dimensiones	30*20*11.4 (mm)	40-55 (mm)
Tiempo de Adquisición (*Cold start*)	27 (s)	30 (s)
Sensibilidad de navegación	-161 (dBm)	-165 (dBm)
Voltaje de alimentación	2.7-3.6 (V)	3.4-4.2 (V)
Corriente máxima	67 (mA)	2 (A)

Con el propósito de cumplir con el requerimiento de notificar al usuario se establecieron los mensajes de texto y llamadas de voz como medios de alerta. Por esta razón se hizo necesaria emplear la red celular que acceder a Internet y para el envío de las notificaciones mencionadas. Las tecnologías que permiten esto van desde la segunda hasta la cuarta generación y su cobertura se muestra en la Tabla 3.5. Por esta comparativa se selecciona la tecnología 2G como red celular a ser empleada en el sistema.

Tabla 3.5 Cobertura celular según la tecnología *[20]*

Tecnología	Sector Urbano (%)	Sector Rural (%)	Nivel Nacional (%)
2G	99.85	87.99	96.81
3G	99.87	71.46	92.58
4G	77.71	23.32	63.75

Para la selección del módulo celular se consideró los discriminantes mostrados en la Tabla 3.6 donde se realizó una comparación de 3 módulos que permiten usar los servicios de la red celular y enviar datos por la red GPRS.

Tabla 3.6 Comparativa módulos GPRS *[19] [21] [22]*

Característica	SIM800L	SIM808	SIM900
Voltaje máximo	4.4 (V)	4.4 (V)	4.8 (V)
Corriente máxima	2 (A)	2 (A)	2 (A)
Bandas de Frecuencia	GSM 850, EGSM 900, DCS 1800, PCS 1900	GSM 850, EGSM 900, DCS 1800, PCS 1900	GSM 850, EGSM 900, DCS 1800, PCS 1900
Velocidad máxima de transmisión	85.6 (kbps)	85.6 (kbps)	85.6 (kbps)
Conectividad GPRS	Multi-slot 12	Multi-slot 12	Multi-slot 10
Protocolos	PAP, TCP/IP, PBCCH	PAP, TCP/IP, PBCCH	TCP/IP, PBCCH
Dimensiones	25*23 (mm)	7.4*5.1 (mm)	80*66 (mm)

Tras observar la esta comparación se seleccionó el módulo SIM800L debido a ser el dispositivo de menores dimensiones en comparación con los demás presentados. Además, presenta el menor voltaje de alimentación junto con una clase de transmisión multi-ranura 12 que representa una mejor velocidad de transmisión y está en conformidad con la frecuencia de operación de las redes 2G de Ecuador 850 (MHz). Otras características técnicas se pueden encontrar en el **Anexo A3**.

La selección de las antenas se realizó en base a sus características técnicas tales como la ganancia y al espacio disponible dentro de la carcasa del dispositivo de rastreo, por lo que se optó usar por las antenas qué se incluyen de fábrica en los módulos, esto debido a su tamaño compacto y al hecho de que son antenas pasivas, es decir, que no necesitan un circuito de alimentación extra y a sus desempeños en un entorno urbano.

Los requisitos para la selección de la batería son: voltaje dentro del rango de 3.3 a 4.4 (V) y capacidad de más de 1758 (mAh). La sección Análisis de consumo energético explica el porqué de estos valores. En función de estos discriminantes la batería seleccionada fue una celda Li-Po 903052 de 3.7 (V) y 2000 (mAh), provista de un módulo BMS (*Battery Management System*) para la prevención de sobrecargas y sobredescargas.

En la Figura 3.1 se muestra un resumen de los elementos principales que conforman el dispositivo localizador junto con los periféricos que acompañan a cada uno de estos.

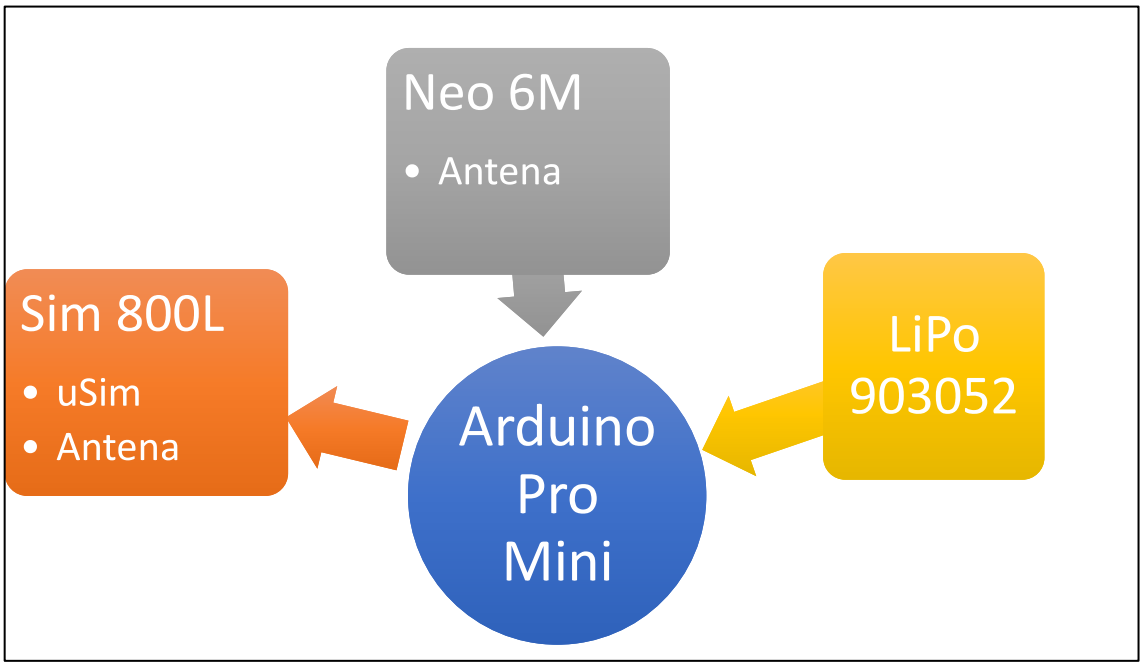


Figura 3.1 *Diagrama de bloques del prototipo*

Funcionamiento del sistema

El esquema de funcionamiento se puede encontrar en Figura 3.2 Esquema del sistema de localización. El proceso inicia en la etapa de adquisición, basada en el dispositivo localizador. El localizador tiene como misión emplear su receptor GPS por medio del módulo NEO 6M para recibir las señales de tipo NMEA provenientes de la red de radionavegación, que son decodificadas a través de algoritmo ejecutado por el Arduino Pro Mini, el mismo que es alimentado por la batería LiPo que faculta de independencia energética al localizador y por tanto su movilidad.

Posterior a la decodificación, los datos en términos de longitud y latitud son enviados con la ayuda del módulo Sim800L usando la tecnología celular GPRS hasta la base de datos ubicada en el servidor Tracker2020.xyz. Además, el microcontrolador dispone de

un indicador que le advierte al usuario que la batería está próxima a agotarse. Para la carga de la batería del localizador se puede utilizar un cargador tipo micro USB a 5 (V).

Ya en el servidor los datos de posición se alojan en la base de datos correspondiente. El servidor contiene, además, los guiones o *scripts* y archivos de soporte para las aplicaciones.

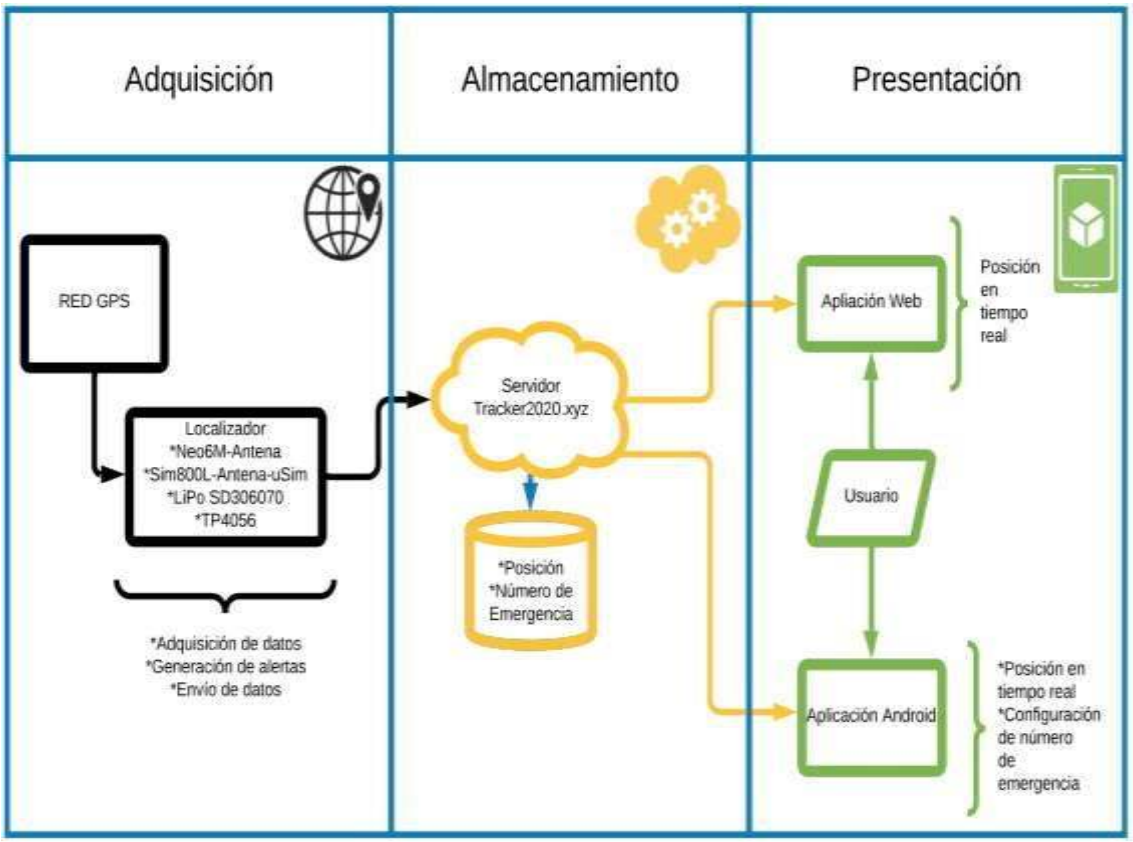


Figura 3.2 Esquema del sistema de localización

Del lado del usuario o cuidador se tienen 2 aplicaciones Android y web. Estas aplicaciones cumplen con el objetivo de permitirle al cuidador observar el desplazamiento en tiempo real del localizador. La aplicación para Android adicionalmente le propicia al usuario un campo para ingresar un número celular de cualquier operadora de Ecuador al que llegarán los avisos de emergencia.

Por último, el sistema ofrece dos modos de uso espía y emergencia. En el caso del modo espía los datos se envían desde el localizador de forma continua con períodos de 180 (s) sin generar las notificaciones de emergencia al cuidador, esto permite que las aplicaciones realicen un seguimiento inadvertido del propietario del localizador.

El modo emergencia entra en funcionamiento inmediatamente después de accionar el pulsador del localizador. Una vez que esto sucede se cambia el período de envío de datos de posición a la base a un tiempo menor: 30 (s). Además, se realizan dos notificaciones de emergencia, una de estas por medio de una llamada al número ingresado por el cuidador que se cuelga de forma autónoma luego de 10 (s) y el envío de un SMS para utilizar la aplicación Android y un enlace de acceso a la aplicación web. Este modo de operación se mantiene en funcionamiento mientras no se cambie el modo en el localizador.

Diseño del circuito localizador

El prototipo de localizador es controlado mediante el sistema de Arduino Pro Mini que tiene como núcleo el microcontrolador ATmega328. A este dispositivo se conectaron las interfaces GPS y GRPS así como la fuente de energía y elementos electrónicos, como se muestran en la Figura 3.3. En la Tabla 3.7 se describen los pines utilizados en la placa Arduino. Los pines no anotados en la tabla no son utilizados.

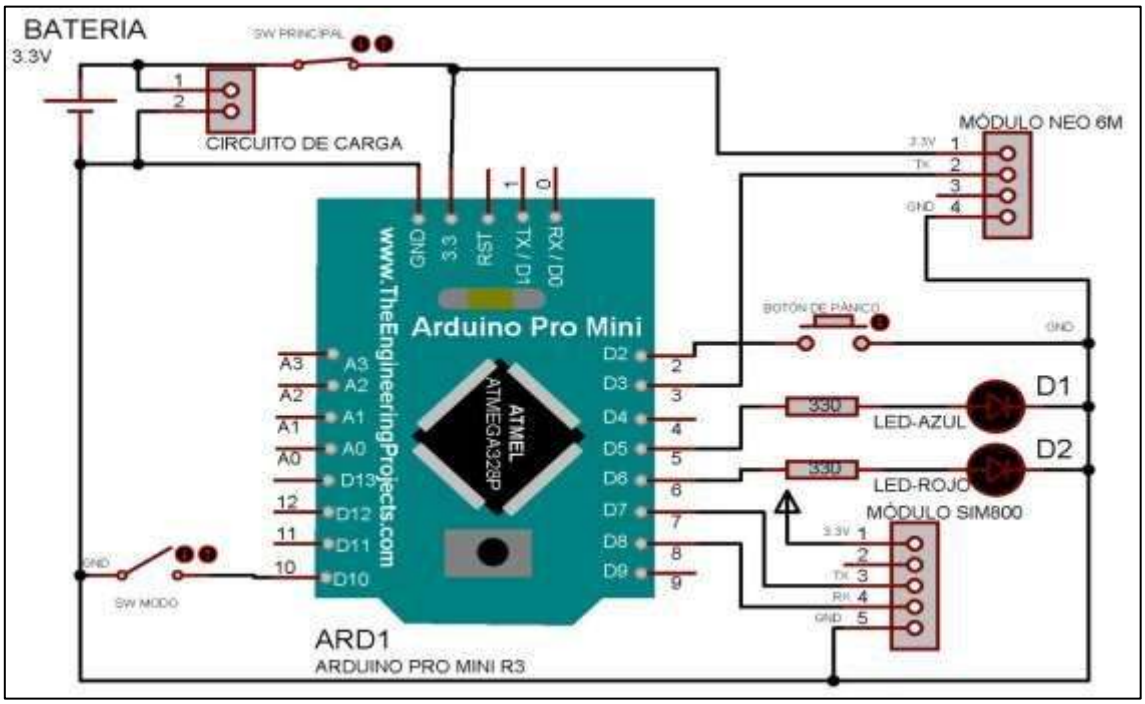


Figura 3.3 Diagrama de conexión localizador

Tabla 3.7 Distribución de pines Arduino Pro mini

Pin	Arduino Pro Mini	Pin	Arduino Pro Mini	Pin	Arduino Pro Mini
3.3	Batería+	D3	Tx NEO 6M	D7	Tx SIM800L
GND	Batería-	D5	LED azul	D8	Rx SIM800L
D2	Botón de pánico	D6	LED rojo	D10	Switch

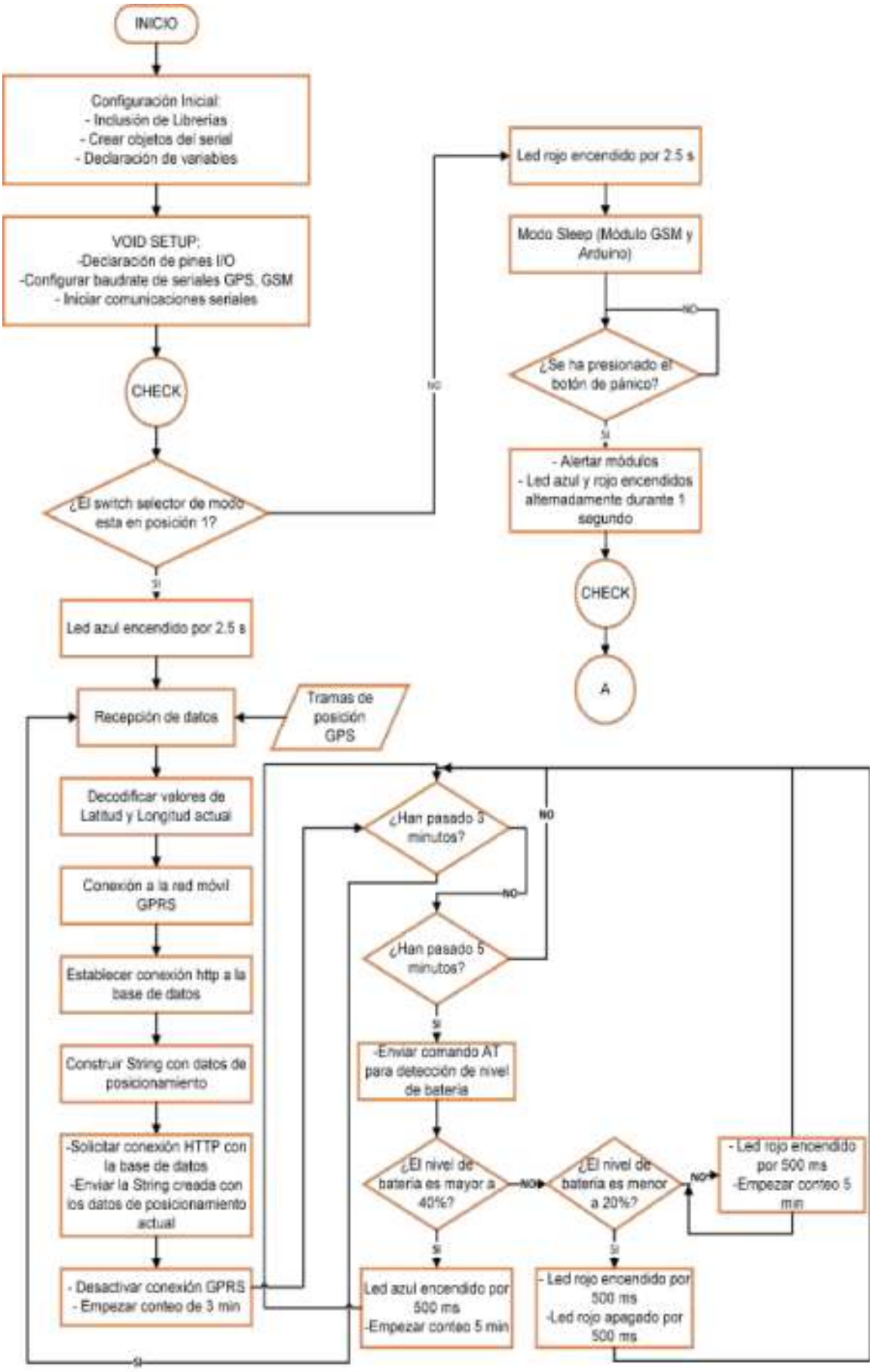
Programa de control

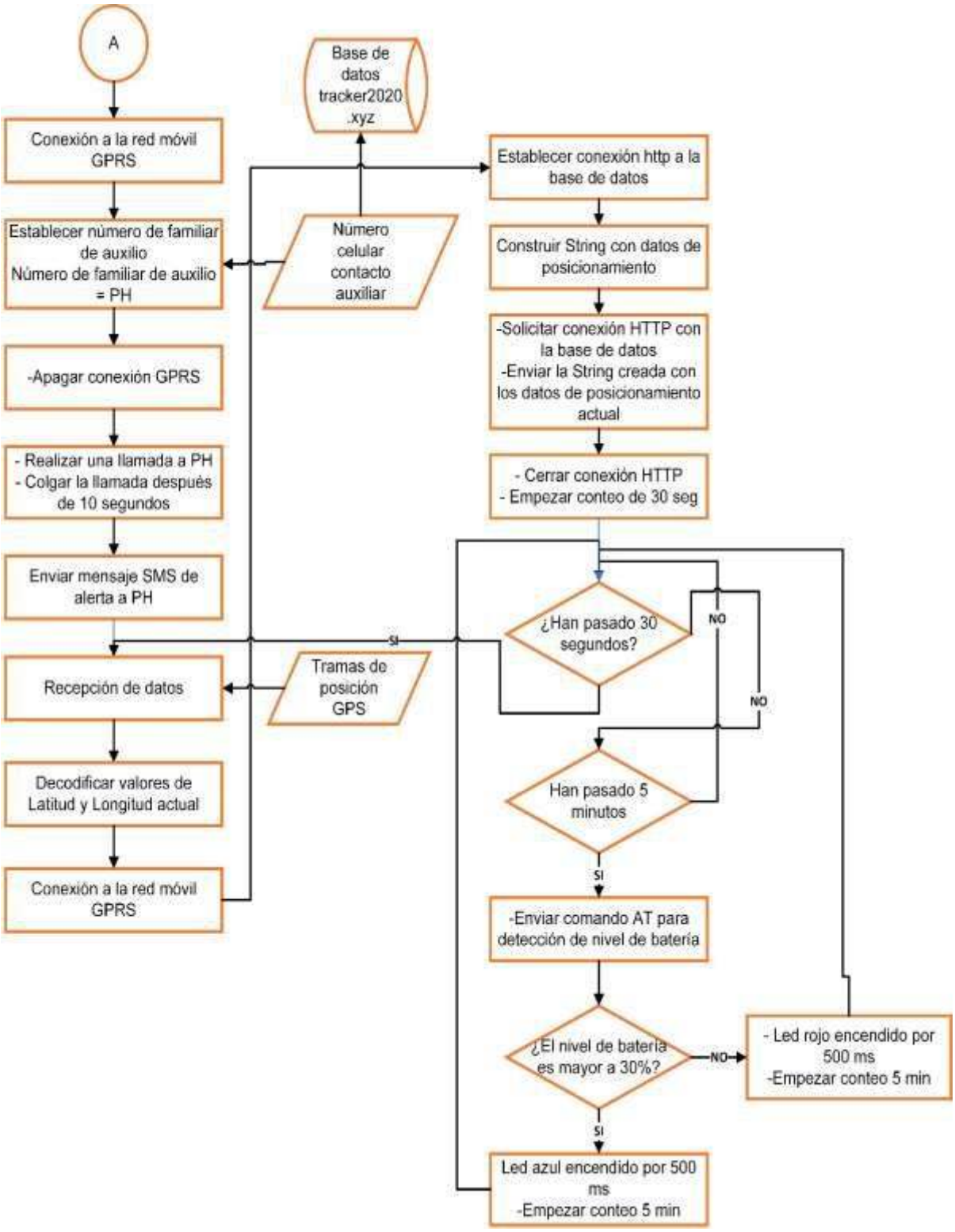
Para la escritura del código se utilizó el entorno de programación Arduino IDE y un bus de datos que sirvió de puente entre el ordenador y el microcontrolador. El algoritmo almacenado en la memoria del ATmega328 es el que se observa en la Figura 3.4. En un inicio, se agregaron las librerías para el control de dispositivos, así como la creación de objetos y declaración de variables. Después, se establecieron los pines de entrada y salida, valores de *baud rate* y se iniciaron las comunicaciones seriales.

Luego de definir los parámetros iniciales, se llama a la subrutina "*CHECK*" en la que se verifica la comunicación con el módulo GSM. De no lograr comunicación con el módulo se enciende el LED rojo por 500 (ms) y se ejecuta un nuevo intento. En caso de lograr comunicarse con el módulo se enciende el indicador azul por 500 (ms) para luego evaluar la red celular. Si no se consigue conexión con la red celular se enciende el LED rojo por 500 (ms) y se lanza una nueva prueba. En el caso favorable, el LED azul es encendido por 500 (ms) y se definen los parámetros del APN (*Access Point Name*) del proveedor celular.

Una vez configurados estos valores es revisada la posición del selector. Cuando el selector se encuentra en posición uno el LED azul es prendido por 2.5 (s) que indica que el modo espía está en funcionamiento. Luego, se reciben los datos de posición GPS que a su vez son decodificados para inmediatamente después establecer la conexión a la red GPRS por medio de la cual se utiliza el protocolo HTTP para el envío de las cadenas de datos de posicionamiento. La red GPRS es desconectada luego de hacer el envío y se inicia un contador por un total de 180 (s) para después repetir el proceso de adquisición y envío de la información de posicionamiento.

Para actualizar los de datos en el servidor se cumple con el proceso que se observa en la Figura 3.5. El *script* "*write_data*" inicia con la configuración de los datos de acceso a la base, luego lee los valores que continúan a "*value* 1" y "*value* 2" de la cadena de caracteres del tipo "http://tracker2020.xyz/write_data.php?value1=-0.2028999&value2=-78.4910538". A continuación, se inicia la conexión y se verifica que no existan errores en esta. De no haber errores en la conexión, se procede con la inserción de los datos de latitud y longitud en la tabla "Sensor" y se verifica que no existan fallos. Al no encontrar error, se devuelve el mensaje de éxito en la subida de datos y se finaliza la conexión. En el **Anexo B2** se halla el código correspondiente.


INICIO
Configuración Inicial:
- Inclusión de Librerías
- Crear objetos del serial
- Declaración de variables
VOID SETUP:
-Declaración de pines I/O
-Configurar baudrate de seriales GPS, GSM
- Iniciar comunicaciones seriales
CHECK
¿El switch selector de modo esta en posición 1?
NO
SI
Led rojo encendido por 2.5 s
Modo Sleep (Módulo GSM y Arduino)
NO
¿Se ha presionado el botón de pánico?
SI
- Alertar módulos
- Led azul y rojo encendidos alternadamente durante 1 segundo
CHECK
A
Led azul encendido por 2.5 s
Recepción de datos
Tramas de posición GPS
Decodificar valores de Latitud y Longitud actual
Conexión a la red móvil GPRS
Establecer conexión http a la base de datos
Construir String con datos de posicionamiento
-Solicitar conexión HTTP con la base de datos
-Enviar la String creada con los datos de posicionamiento actual
- Desactivar conexión GPRS
- Empezar conteo de 3 min
¿Han pasado 3 minutos?
NO
¿Han pasado 5 minutos?
NO
SI
-Enviar comando AT para detección de nivel de bateria
¿El nivel de bateria es mayor a 40%?
NO
¿El nivel de bateria es menor a 20%?
NO
- Led rojo encendido por 500 ms
-Empezar conteo 5 min
SI
Led azul encendido por 500 ms
-Empezar conteo 5 min
SI
- Led rojo encendido por 500 ms
-Led rojo apagado por 500 ms


A
Base de datos tracker2020 .xyz
Conexión a la red móvil GPRS
Establecer conexión http a la base de datos
Establecer número de familiar de auxilio
Número de familiar de auxilio = PH
Número celular contacto auxiliar
Construir String con datos de posicionamiento
-Apagar conexión GPRS
-Solicitar conexión HTTP con la base de datos
-Enviar la String creada con los datos de posicionamiento actual
- Realizar una llamada a PH
- Colgar la llamada después de 10 segundos
- Cerrar conexión HTTP
- Empezar conteo de 30 seg
Enviar mensaje SMS de alerta a PH
¿Han pasado 30 segundos?
NO
SI
Recepción de datos
Tramas de posición GPS
NO
Decodificar valores de Latitud y Longitud actual
Han pasado 5 minutos
SI
Conexión a la red móvil GPRS
-Enviar comando AT para detección de nivel de batería
¿El nivel de batería es mayor a 30%?
NO
- Led rojo encendido por 500 ms
-Empezar conteo 5 min
SI
Led azul encendido por 500 ms
-Empezar conteo 5 min

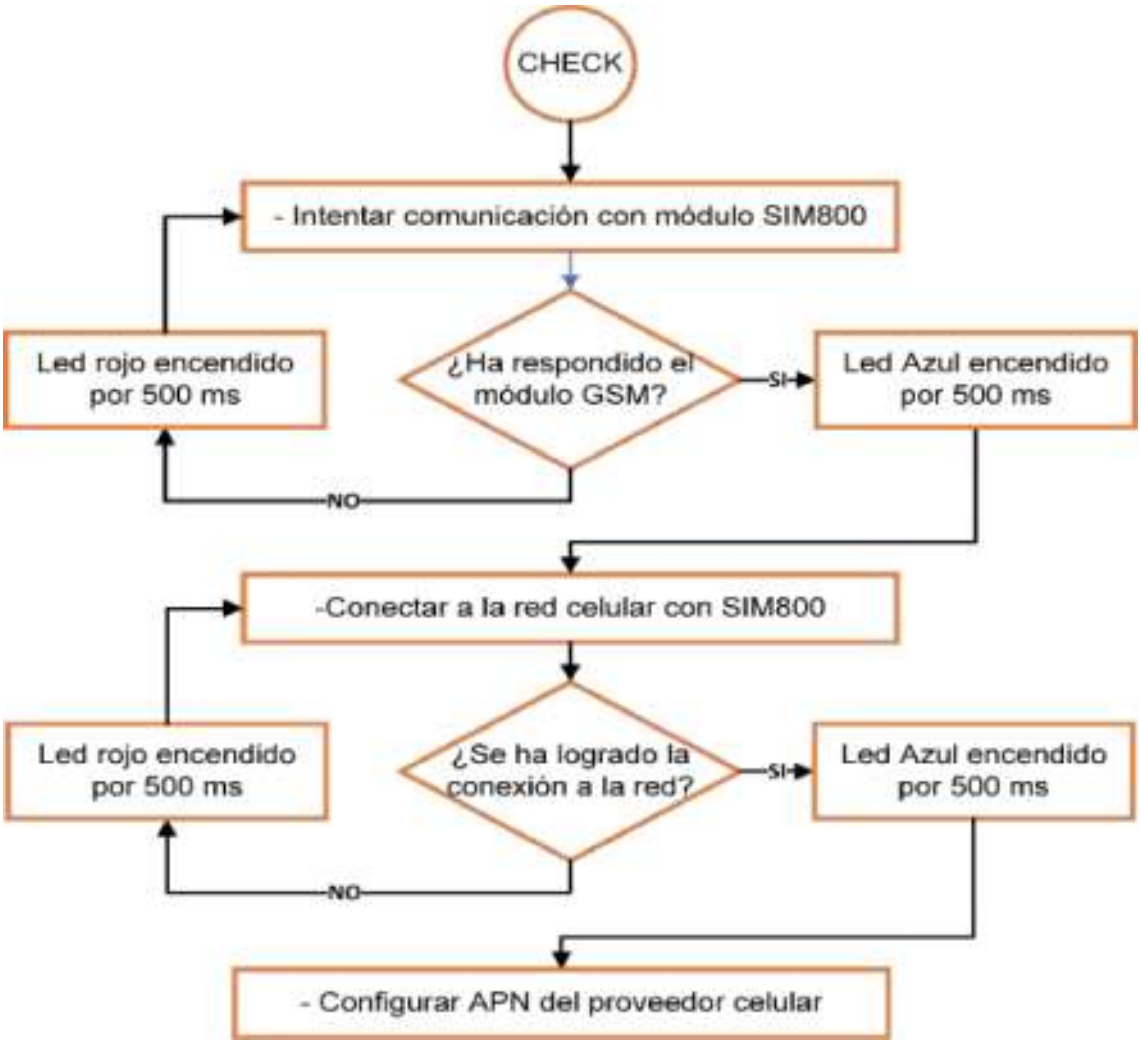


Figura 3.4 Diagrama de flujo Arduino Pro mini

Si se sitúa el selector del localizador en la posición contraria el controlador da la orden, luego de encender el LED rojo por 2.5 (s), a los módulos GSM y GPRS de entrar en modo *sleep* hasta que el botón de pánico se presionado. Esto producirá una activación de los módulos, el parpadeo de los 2 LEDs, la ejecución de la subrutina *CHECK*, la obtención del número de emergencia y la generación de alerta por medio de una llamada y un SMS a este, para luego de esto realizar la toma de datos con un período menor y efectuar la revisión del nivel de batería.

El nivel de batería es revisado en los dos modos cada 5 minutos. Un nivel de carga sobre 30% del valor total será evidenciado por la luz del indicador azul por 500 (ms) por otro lado, si es menor se encenderá el indicador rojo por el mismo tiempo. La toma del número de emergencia se lleva a cabo con el guion “vercell.php”. Ver **Anexo B3**.

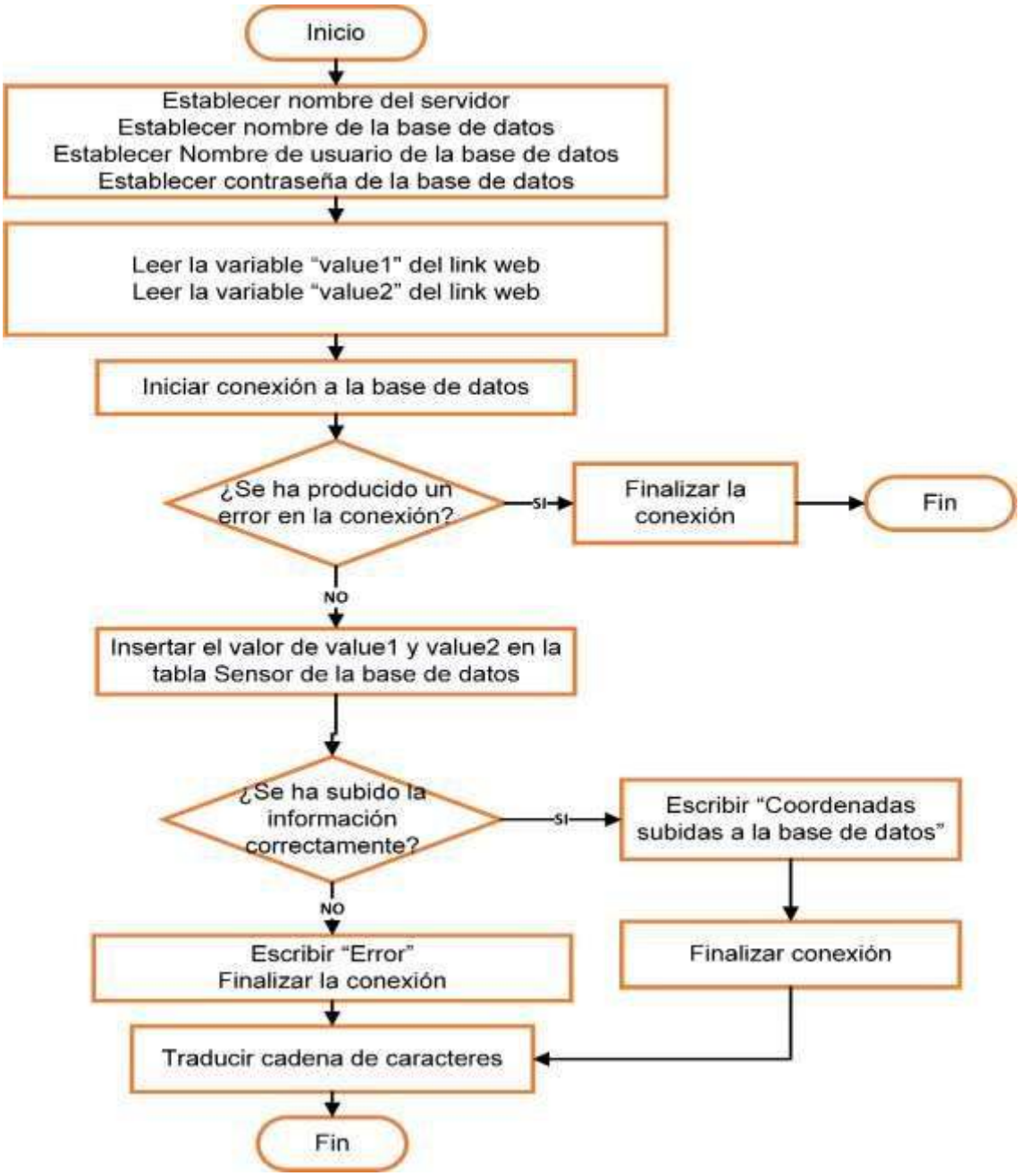


Figura 3.5 Diagrama de flujo "*write_data.php*"

Como la Figura 3.6 señala luego de colocar los datos del servidor e iniciar la conexión se solicita el dato con el último valor de identificador de la tabla "celular" y los coloca en un arreglo, luego de obtener los datos se constata que el arreglo disponga de valores, de ser así, se escribe el número y se vuelve a revisar el arreglo hasta que no tenga información y se cierra el bucle

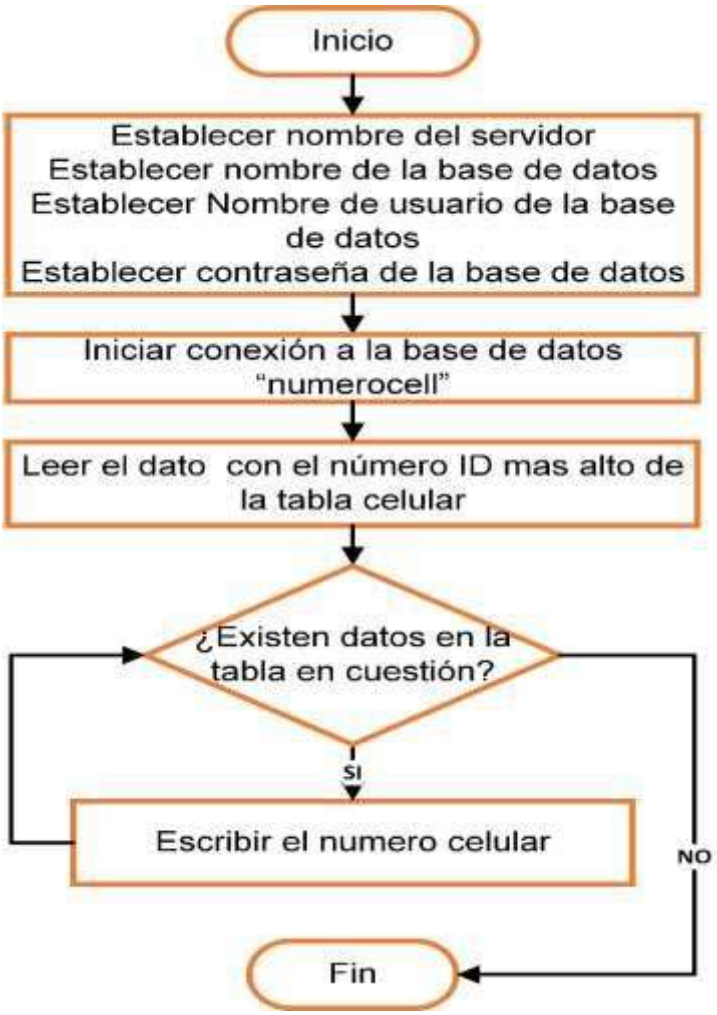


Figura 3.6 Diagrama de flujo "vercell.php"

Análisis de consumo energético

Para poder determinar el consumo y tiempo de operación del dispositivo localizador se analizaron los consumos energéticos de los componentes del circuito. Para este análisis se descartan los elementos que tengan un consumo bajo. Con respecto al voltaje de alimentación se puede observar en la Tabla 3.8 que los valores de alimentación de cada dispositivo que se encuentran en un intervalo entre 3.3 a 4.4 (V), siendo aceptable cualquier valor dentro de este rango.

Tabla 3.8 Voltajes de alimentación elementos principales *[14] [23] [24]*

	Arduino Pro Mini	SIM800L	NEO 6M
Voltaje alimentación (V)	3.3	3.4 - 4.4	2.7 - 3.6

Con el fin de analizar el consumo energético la Figura 3.7 se puede observar que el modo de mayor consumo es el *standby* con 6.41 (mA) y el más bajo el modo *power down* con 0.1 (mA).

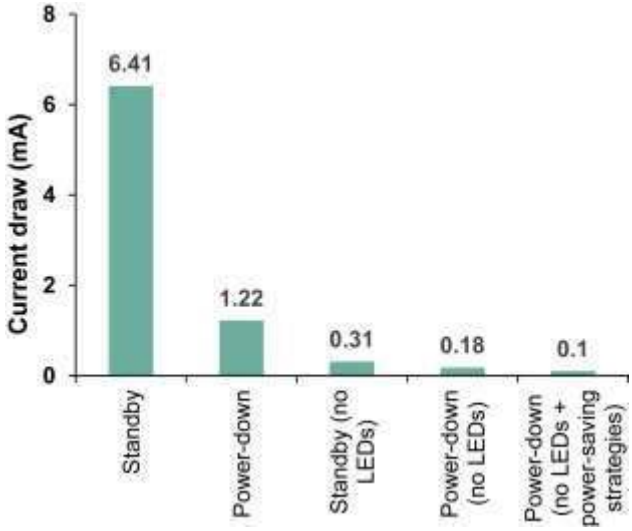


Figura 3.7 Consumo promedio vs modo de operación del Arduino Pro mini

A continuación, en la Tabla 3.9 se muestran los distintos modos de operación con su respectivo consumo para el módulo GPS NEO 6M

Tabla 3.9 Consumo por modo de operación NEO 6M *[24]*

Parámetro	Función		Valor típico	Valor máximo	Unidad
Corriente de alimentación máxima				67	mA
Corriente de alimentación promedio	Adquisición		47		
	Tracking	Modo máximo rendimiento	39		
		Modo ECO	37		
		Modo ahorro de energía	11		

En la Tabla 3.10 se presentan los distintos modos de operación junto al consumo promedio del módulo GPRS.

Tabla 3.10 Consumo por modo de operación SIM800L *[23]*

Parámetro	Condición		Valor típico	Valor máximo	Unidad
Corriente Promedio	Modo Ahorro de energía			60	uA
	Modo Datos	GSM 850	212.69		mA
		EGSM 9000	227.95		
		DCS1 1800	158.19		
		PCS1 1900	148.19		

Modo Espía

En esta modalidad de funcionamiento el dispositivo localizador envía los datos de posición de manera continua lo que en términos de consumo energético representa una alimentación constante de los módulos. En función de esto se plantea que el tiempo de autonomía del dispositivo localizador es de al menos siete horas.

En este escenario el módulo Arduino se configura en modo *standby*, el módulo GPS en modo de máximo rendimiento, cabe mencionar que para el cálculo el fabricante recomienda utilizar el valor pico de consumo, y el módulo GPRS se configura en modo de transmisión de datos 4Rx y 1Tx en la banda de frecuencia GSM850. En la Tabla 3.11 se pueden observar los valores de corriente eléctrica por dispositivo. Las actualizaciones en la base de datos se realizan cada 180 segundos.

Tabla 3.11 Consumo del localizador en modo espía *[14] [23] [24]*

	Arduino Mini Pro	GPS NEO 6M	SIM 800L
Consumo (mA)	6.41	67	212.69

Con estos valores se puede calcular el consumo del localizador con la ***Ecuación 3.1***.

$$I_{Sis} = I_{Arduino} + I_{GPS} + I_{SIM800L}$$

Ecuación 3.1 Corriente total del sistema

Donde:

$I_{Arduino}$: 6.41 (mA) corriente Arduino Pro mini

I_{GPS} : 67 (mA) corriente NEO 6M

$I_{SIM800L}$: 219.69 (mA) corriente SIM800L

Usando la Ecuación 3.1 se obtiene:

$$I_{Sis} = 293.1\ (mA)$$

Para seleccionar la batería adecuada la corriente total que podrá suministrar la batería utilizada y la corriente total del sistema. Para esto se utilizará la Ecuación 3.2.

$$Autonomía = \frac{C_{Bat}}{I_{Sis}}$$

Ecuación 3.2 Tiempo de autonomía del sistema

Puesto que la autonomía deseada es 6 (h) y se desea conocer la capacidad de la batería se despeja de la Ecuación 3.2:

$$C_{Bat} = Autonomía * I_{Sis}$$

Donde:

$Autonomía$	:	6 (h) tiempo de duración de la batería
I_{Sis}	:	293.1 (mA) Corriente total del sistema

Por lo tanto:

$$C_{Bat} = 1\ 758\ (mAh)$$

El resultado señala que para una duración de 6 horas se requiere una batería con capacidad de 1758 (mAh). La batería seleccionada ofrece una carga de 2000 (mAh).

Modo Emergencia

El modo emergencia reduce el período de actualización de datos a la base a 30(s). Además, los modos de configuración energética de los módulos que componen el localizador cambian de modo y por lo tanto de consumo.

Este cambio representa dos escenarios del funcionamiento del localizador: 1) Antes de presionar el botón y 2) Después de presionar el botón. En el primer escenario el localizador se encuentra en modo de ahorro de energía por lo que los módulos se configuran de la siguiente manera: Arduino en modo *Power-down*, GPRS en modo *Power-down* y el módulo GPS en modo ahorro de energía, los valores correspondientes a estos modos se encuentran resumidos en la Tabla 3.12 Después de presionar el botón los módulos consumen la misma cantidad de energía que en el modo Espía.

Tabla 3.12 Consumo del localizador en modo emergencia *[14] [23] [24]*

	Arduino Mini Pro	GPS NEO 6M	SIM 800L
Consumo (mA)	1.22	11	0.06

Se utiliza la Ecuación 3.1:

$$I_{Sis} = I_{Arduino} + I_{GPS} + I_{SIM800L}$$

Ecuación 3.3 Corriente total del sistema

Donde:

$I_{Arduino}$: 1.22 (mA) corriente Arduino Pro mini

I_{GPS} : 11 (mA) corriente NEO 6M

$I_{SIM800L}$: 0.06 (mA) corriente SIM800L

De esta se obtiene:

$$I_{Sis} = 12.28\ (mA)$$

Para el escenario 1) la corriente es 12.28 (mA) mientras para el escenario 2) es 293.1 (mA) como ya se calculó. El tiempo de autonomía para este modo variará en función de cuánto tiempo transcurra sin que se pulse el botón de emergencia.

Para conocer el valor máximo de duración de la batería se utiliza la Ecuación 3.2:

$$Autonomía = \frac{C_{Bat}}{I_{Sis}}$$

Ecuación 3.4 Tiempo de autonomía del sistema

Donde:

C_{Bat} : 2000 (mAh) carga de la batería

I_{Sis} : 12.28 (mA) Corriente total del sistema

Lo que da como resultado:

$$Autonomía = 168.87\ (h)$$

El tiempo máximo de duración del dispositivo encendido en modo emergencia sin presionar el botón es 168.87 (h).

Implementación del localizador

Para darle soporte a los elementos del circuito localizador se realizó una placa base mediante el software de simulación Proteus y el programa Ares para el diseño de pistas electrónicas. El tamaño de dicha placa es de 46*24 (mm) y el diseño creado se puede observar en la Figura 3.8.

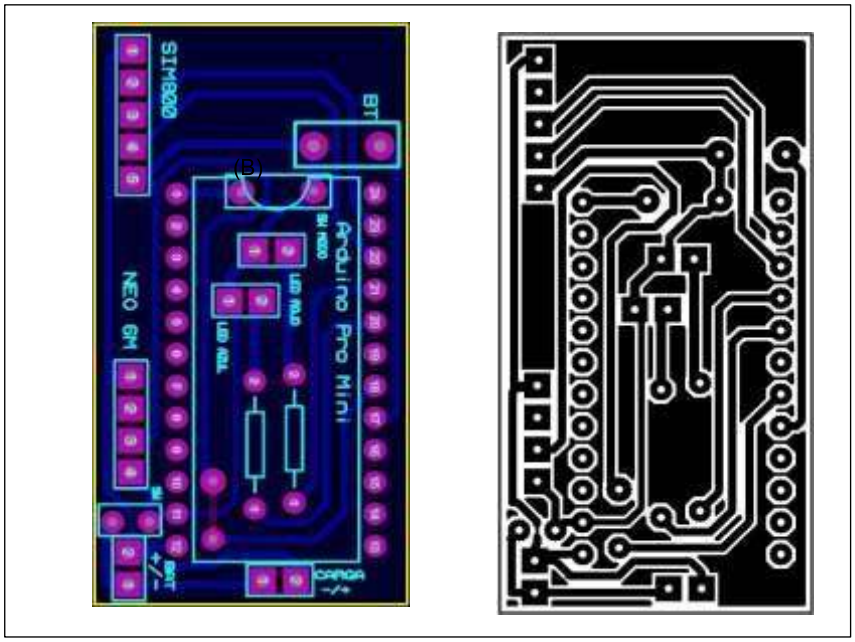
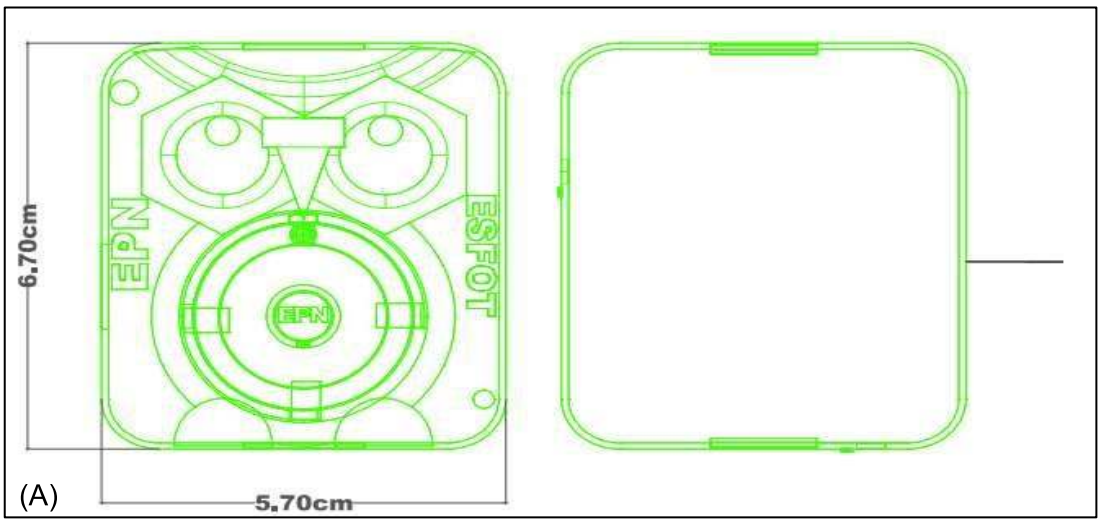


Figura 3.8 (A) Diseño de pistas electrónicas (B) Diseño para impresión

La creación del armazón para el dispositivo localizador se realizó en el software de diseño asistido AutoCAD. Por medio de este se creó la estructura en la que se colocaron los elementos modulares del circuito electrónico además de elementos como leds, pulsadores, *switches* y puertos de carga. El modelo 3D fue impreso en filamento PLA (poliácido láctico), el mismo se puede observar en la Figura 3.9.



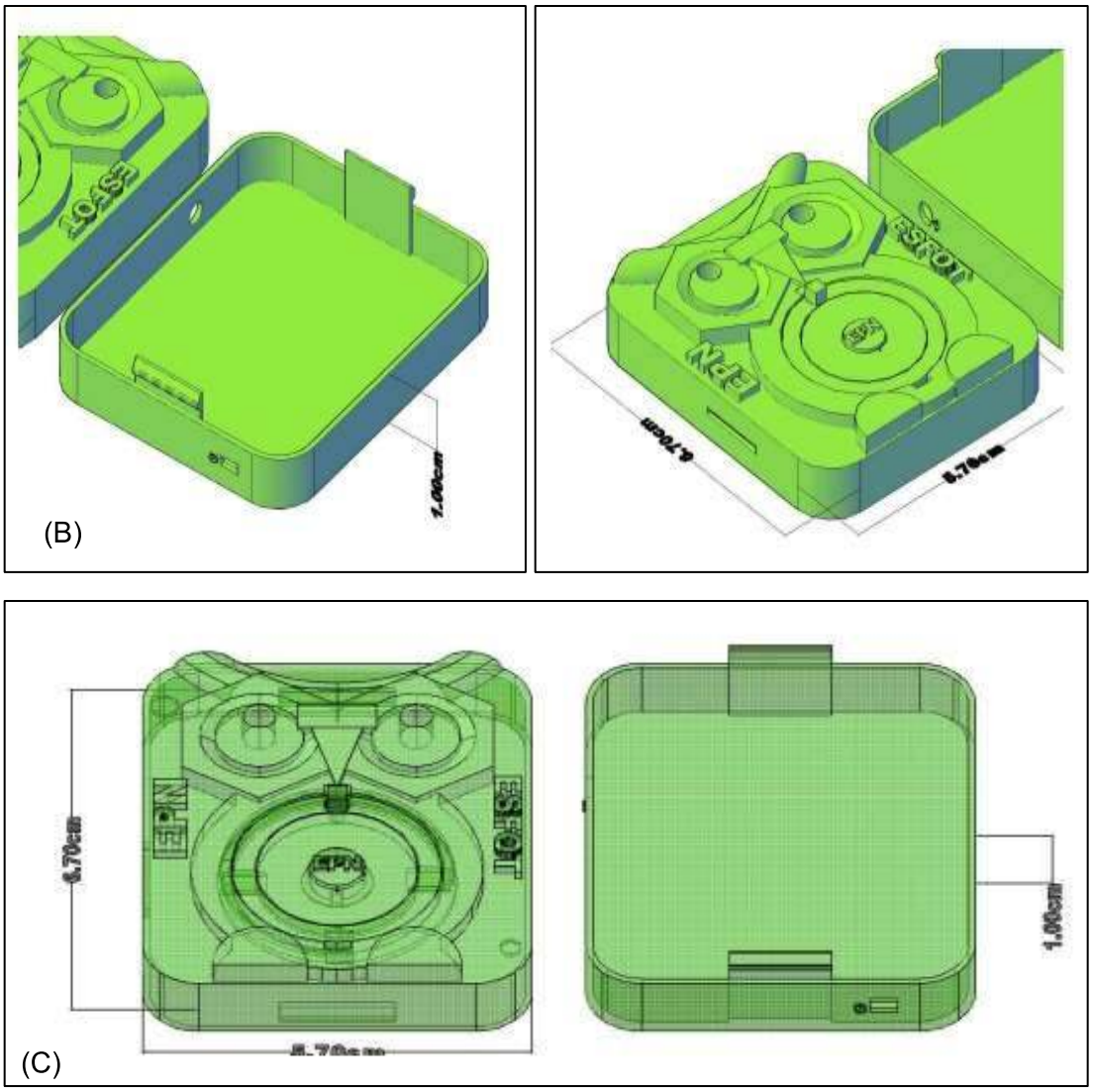


Figura 3.9 (A) Vista superior 2D (B) Vistas laterales 3D (C) Vista superior 3D

3.3 Desarrollo de las interfaces de usuario que permitan la visualización de la posición del individuo en tiempo real

Implementación de aplicación web

Para el desarrollo de la aplicación web fue necesario obtener un espacio para el alojamiento de datos, crear un nombre de dominio, configurar las bases para el almacenamiento de la información adquirida por el localizador, escribir los scripts y archivos de funcionamiento de la aplicación, además ofrezca una herramienta para la creación y diseño de la interfaz de usuario.

El resultado final fue una aplicación que permite al usuario observar la posición del dispositivo localizador en tiempo real sobre el mapa de Google en sus formatos callejero

y satelital con funciones de autocentrado y autozoom. En un entorno sencillo, fácil de usar y con claras instrucciones de uso como la Figura 3.10.

Figura 3.10 Interfaz aplicación web

Levantamiento del servidor

Para el levantamiento del servidor, fue esencial la contratación de un servicio de *hosting* en línea por lo que se comparó el plan básico de los tres proveedores que se halla en la Tabla 3.13. Producto de la comparación se escogió al proveedor *Hostinger* por la cantidad de servicios incluidos, capacidad de almacenamiento elevado, nivel de complejidad bajo, alta disponibilidad y menor precio entre los proveedores comparados.

En este entorno se crearon: el espacio de almacenamiento y nombre de dominio. Como dominio del servidor se asignó el nombre “tracker2020.xyz”.

Tabla 3.13 Comparativa proveedores de *hosting*

Característica	Wix	GoDaddy	Hostinger
Servicios Incluidos	Dominio+*Hosting*	Dominio	Dominio+*Hosting*+SSL
Almacenamiento en nube	10Gb	8Gb	10Gb
Tipo de almacenamiento	SSD	HDD	SSD
Disponibilidad	99.97%	99.98%	99.98%
Manejo de la Interfaz	Complejo	Poco complejo	Poco complejo
Pago mensual (usd)	4.50	6.02	2.99

Base de datos en MySQL

Los datos recopilados por el localizador y la aplicación Android crearon la necesidad de contar con un espacio de almacenamiento. Para solventar esta exigencia del sistema se crearon dos bases de datos ligadas al dominio Tracker2020.xyz. Una de ellas para alojar los números de celular configurados por el usuario y otra que albergue los datos de posición provenientes del localizador.

La base de números celulares llamada "*numerocell*" contiene una tabla de nombre "celular" en la que se crearon dos columnas, una dedicada al identificador de dato y la segunda perteneciente al número insertado. La columna ID es de tipo entero y con autoincremento y la columna siguiente de tipo texto, para mantener el formato del número celular. El contenido de la base "*numerocell*" se muestra en la Figura 3.11

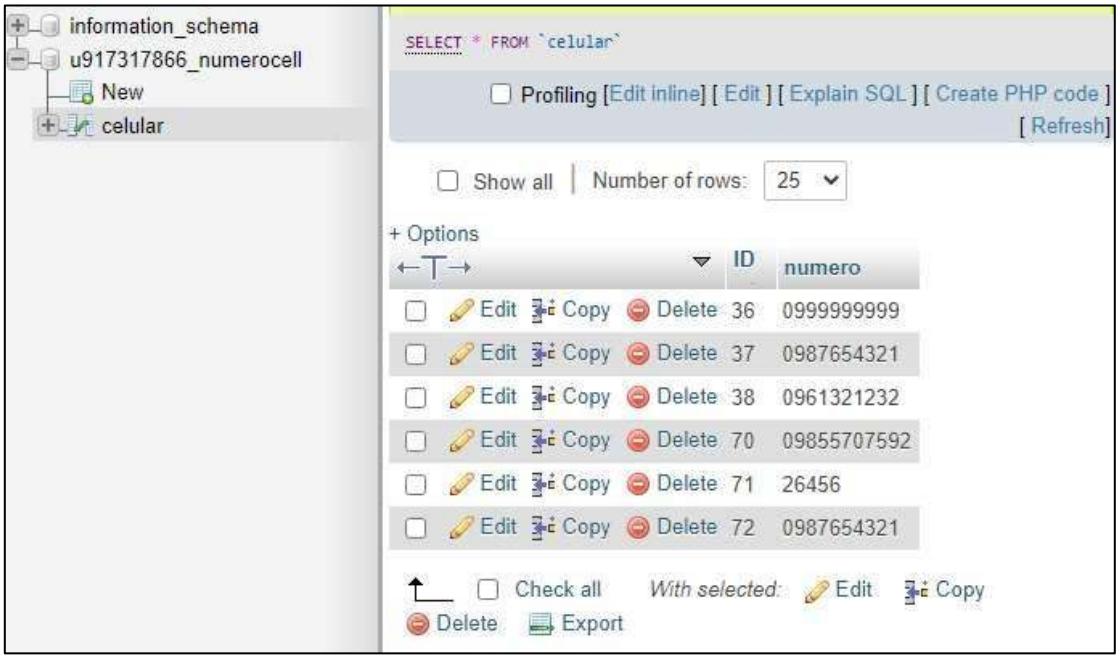


Figura 3.11 Base de datos "*numcell*"

La Figura 3.12 muestra el contenido de la base de datos "*tracker_data*" en la que se creó la tabla "Sensor" que cuenta con 4 columnas: ID, *Value_1*, *Value_2* y TIEMPO_LECTURA. La primera de estas con el fin de dar un valor identificador de tipo número entero con autoincremento. La columna *Value_1* de tipo punto flotante en donde son insertados los datos de latitud. Al igual que el anterior, *Value_2* es de tipo punto flotante para los datos de longitud. La cuarta columna, TIEMPO_LECTURA registra la fecha y hora en el que cada uno de los datos ingresa a la tabla con referencia al uso horario de Greenwich (GTM +0).

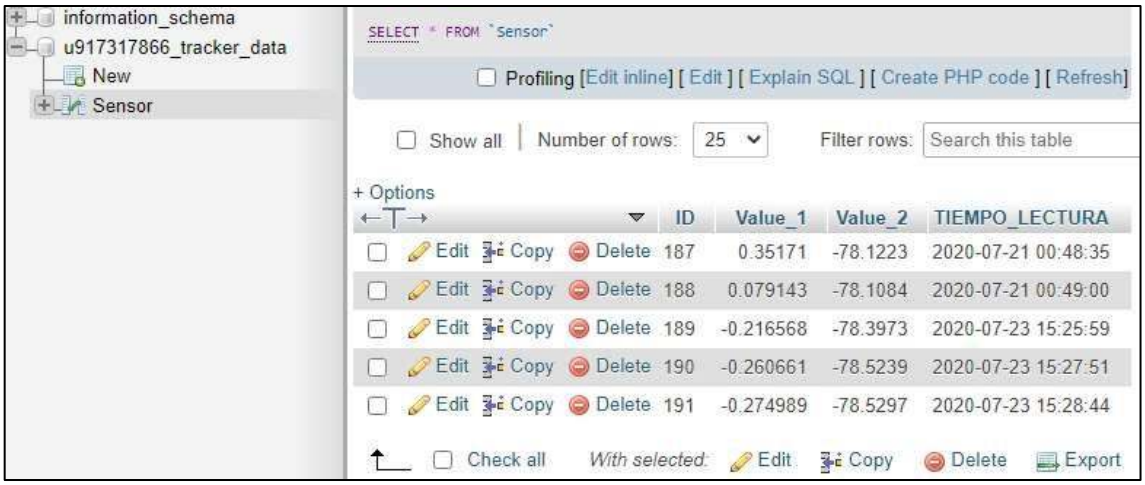


Figura 3.12 Base de datos "*numerocell*"

La Figura 3.13 muestra el contenido de la base de datos "*tracker_data*" en la que se creó la tabla "Sensor" que cuenta con 4 columnas: ID, *Value_1*, *Value_2* y TIEMPO_LECTURA. La primera de estas con el fin de dar un valor identificador de tipo número entero con autoincremento. La columna *Value_1* de tipo punto flotante en donde son insertados los datos de latitud. Al igual que el anterior, *Value_2* es de tipo punto flotante para los datos de longitud. La cuarta columna, TIEMPO_LECTURA registra la fecha y hora en el que cada uno de los datos ingresa a la tabla con referencia al uso horario de Greenwich (GTM +0).

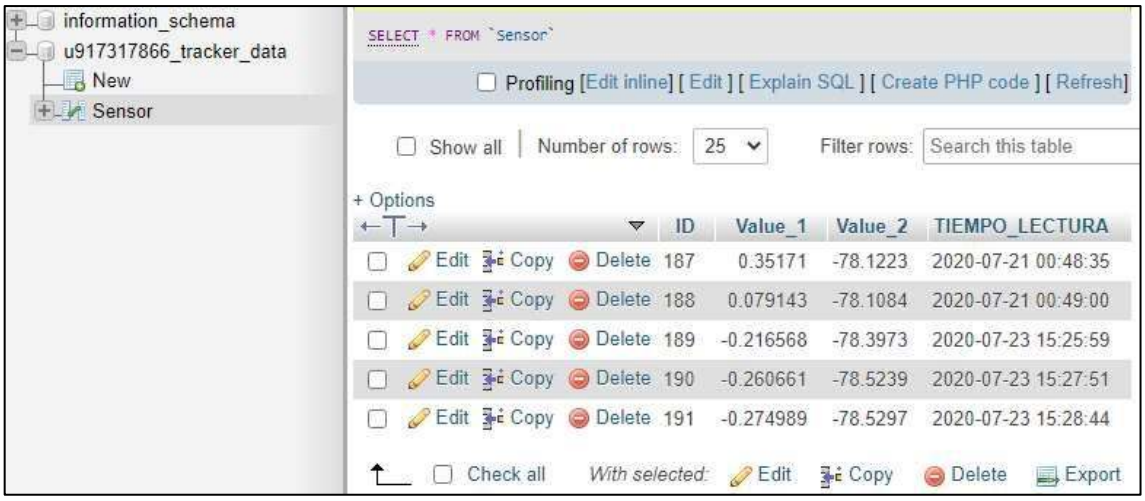


Figura 3.13 Base de datos "*tracker_data*"

Aplicación Web

Con el fin de desarrollar el mapa de la aplicación *web* fue necesaria la elaboración del código en lenguaje HTML, que se incluye en el **Anexo B4,** el resto de elementos gráficos

y texto fueron desarrollados con la ayuda de la herramienta de creación de páginas web incluida en el servicio de *hosting*. La aplicación usa el mapa de Google Maps y permite seleccionar entre el estilo del mapa calle o satélite. Sobre el mapa se coloca un marcador señalando la posición del localizador. Para cumplir con esta labor se utiliza el *script* php de nombre *xmlfile.php* en formato XML. El código de este se halla en el **Anexo B5**. La toma de información desde la base se lleva a cabo haciendo uso de la solicitud *GET*, por lo que se establece como ruta el enlace: http://tracker2020.xyz/

Se observa en la Figura 3.14 el proceso que cumple el mapa de la aplicación web para mostrar la posición del localizador sobre el mapa. En primera instancia, se definen los parámetros de idioma, título, estilo de página y la creación de variables. Luego de esto, se determina la función “InitMap”, encargada de configurar las características del mapa, en donde se establecen: el nivel de *zoom* y estilo de mapa. Se crean, además, las variables de geolocalización y de mapa.

A continuación, se llama al archivo XML, que es el contenedor de las direcciones obtenidas de la base de datos. Se realiza la eliminación de los marcadores que hayan estado situados previamente y se lee la cantidad de coordenadas para proceder con la creación de los marcadores que señalen las últimas posiciones guardadas. Una vez hecho esto, se efectúa un acercamiento al marcador que indique la última posición del localizador. Finalmente, se crea una ventana de información en la que se mostrará la dirección obtenida desde el servicio de Google, que será desplegada únicamente si el usuario “clickea” el marcador. Para esto último se genera un evento en el código que le permite estar a la expectativa de dicho suceso.

En el siguiente punto, el código muestra el marcador y espera la activación del evento. En caso de no llevarse a cabo se traducen las coordenadas a direcciones y se envía el mensaje de dirección no encontrada. Por otro lado, al detonarse el evento los resultados se muestran en la consola luego de haber traducido las coordenadas del último marcador a direcciones leíbles y se incluye el marcador al conjunto “*markers*”. Este resultado se mostrará por 3 segundos mismo tiempo en el que se realiza una actualización de direcciones. Por último, después de haber transcurrido 30 segundos el mapa se centrará en la posición del último marcador.

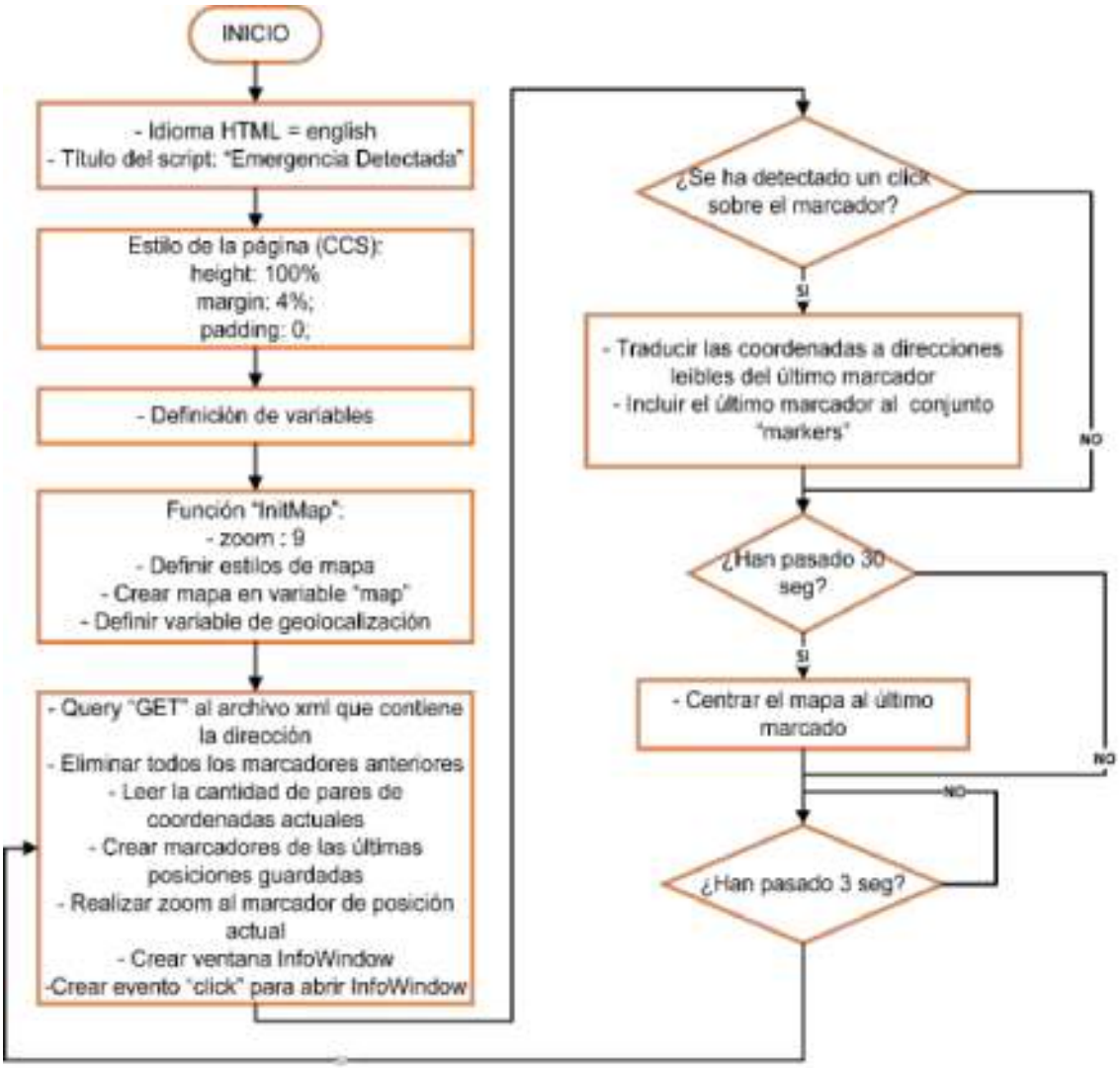


Figura 3.14 Diagrama de flujo del mapa de la aplicación web

La Figura 3.15 muestra el diagrama de flujo del código del archivo *xmlfile.php*. Allí se observa que en un inicio se llama al *script* de nombre "phpsqlajax_dbinfo.php" cuyo contenido comprende los datos de la base y servidor. Después de esto se realiza un reemplazo de caracteres especiales para iniciar la conexión a la base y revisar el estado de esta. Una vez descartados los fallos en la conexión, se selecciona la tabla de nombre "Sensor" de la que se toman los datos para, luego de asegurarse que no hay errores en la adquisición, construir los datos en formato XML hasta que se tomen todos los datos.

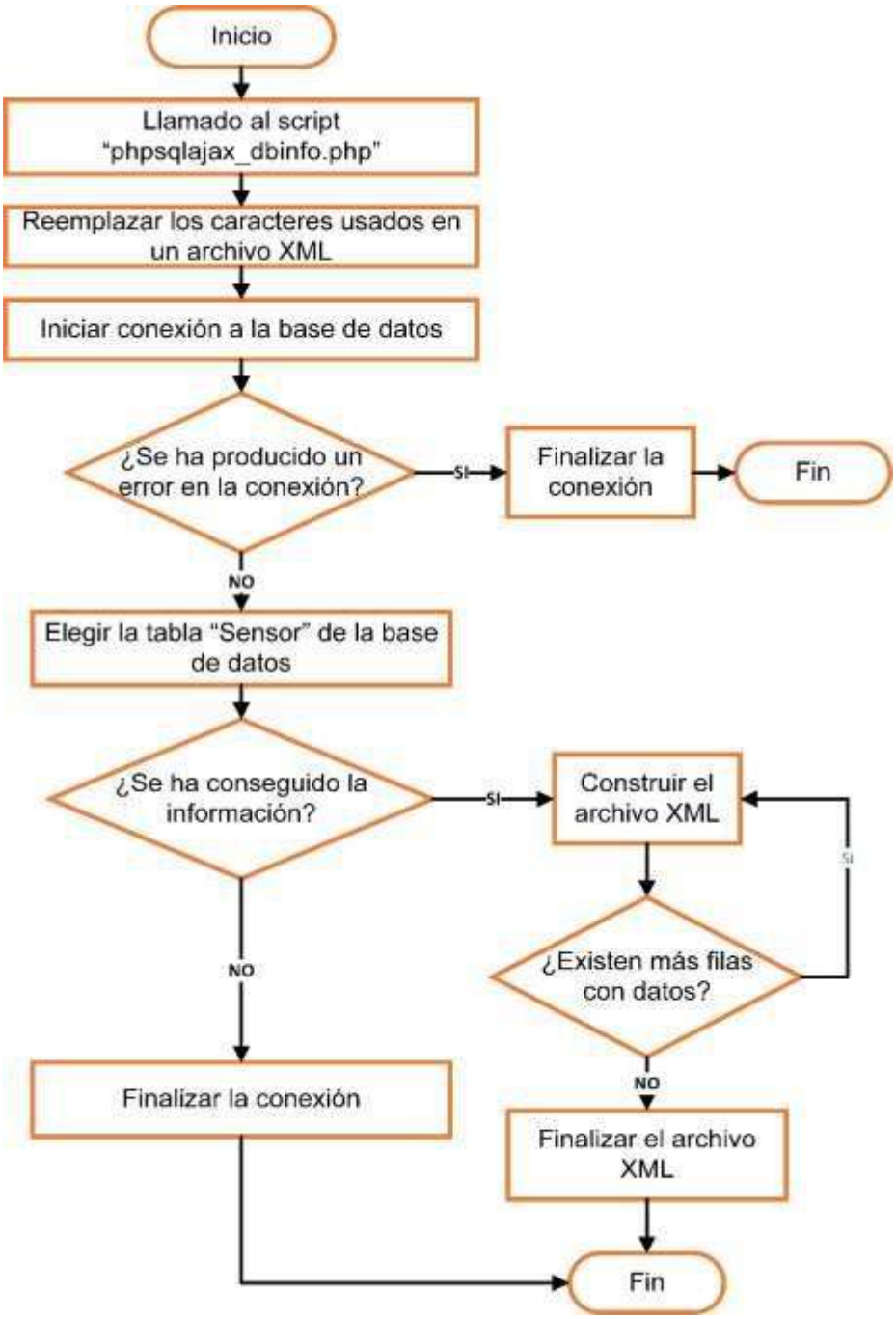


Figura 3.15 Diagrama de flujo archivo "*xmlfile*"

Desarrollo de *scripts* PHP

Para que la aplicación tenga conectividad con la base de datos fue necesaria la creación de guiones o *scripts* PHP. Se optó por la creación de *scripts* PHP puesto que PHP puede ser añadido al lenguaje de programación HTML, el cual es generalmente empleado en el desarrollo web [25]. Dos *scripts* PHP son los encargados de realizar el envío y

recepción de datos a la base asegurando la comunicación entre la base de datos y la aplicación Android. El envío de datos se realizó por medio del método *POST*, mientras que para la recepción de datos el PHP toma los datos solicitados y los envía en formato JSON.

En primer lugar, se establece el tipo de formato de intercambio de datos como JSON para luego definir los parámetros necesarios para la conexión con el servidor y base de datos. Después, se establecen dichos parámetros y se verifica el estado de conexión. En caso de que no se haya completado la conexión se notifica el error, caso contrario se solicitan los valores de latitud y longitud almacenados en la base de nombre “Sensor” correspondientes al último valor de *ID.* Estos valores se almacenan en una variable y a su vez se colocan en forma de JSON dentro de un bucle de repetición, de manera que cada vez que se soliciten datos nuevos se repitan estas tareas.

Finalmente, la respuesta de PHP son los datos JSON que serán usados por la aplicación. El diagrama de flujo correspondiente a este guion se observa en Figura 3.16. El **Anexo B6** presenta el contenido del archivo php

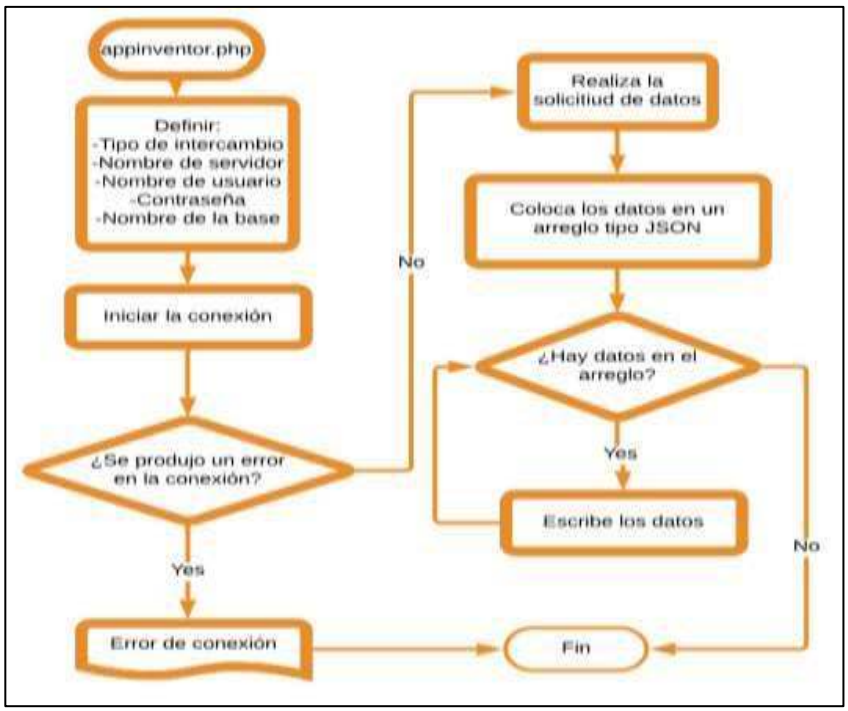


Figura 3.16 Diagrama de flujo "appinventor.php"

El registro del número de emergencia proveniente de la aplicación Android, es llevado a cabo por el proceso mostrado en la Figura 3.17. Primero, se escriben los datos del servidor, nombre de la base, usuario, contraseña y método de intercambio, se inicia la comunicación con el *link web* y se comprueba el estado de conexión. Sino existen

problemas de conexión, se postea el número de celular ingresado en la base respectiva y verifica si la información enviada se almacenó de forma correcta. Si se ejecuta de manera correcta, se responde con el mensaje “Número celular subido a la base” y se procede a cerrar la conexión. De no ser así se muestra el error y se finaliza la conexión. Se incluye el código php en el **Anexo B7**.

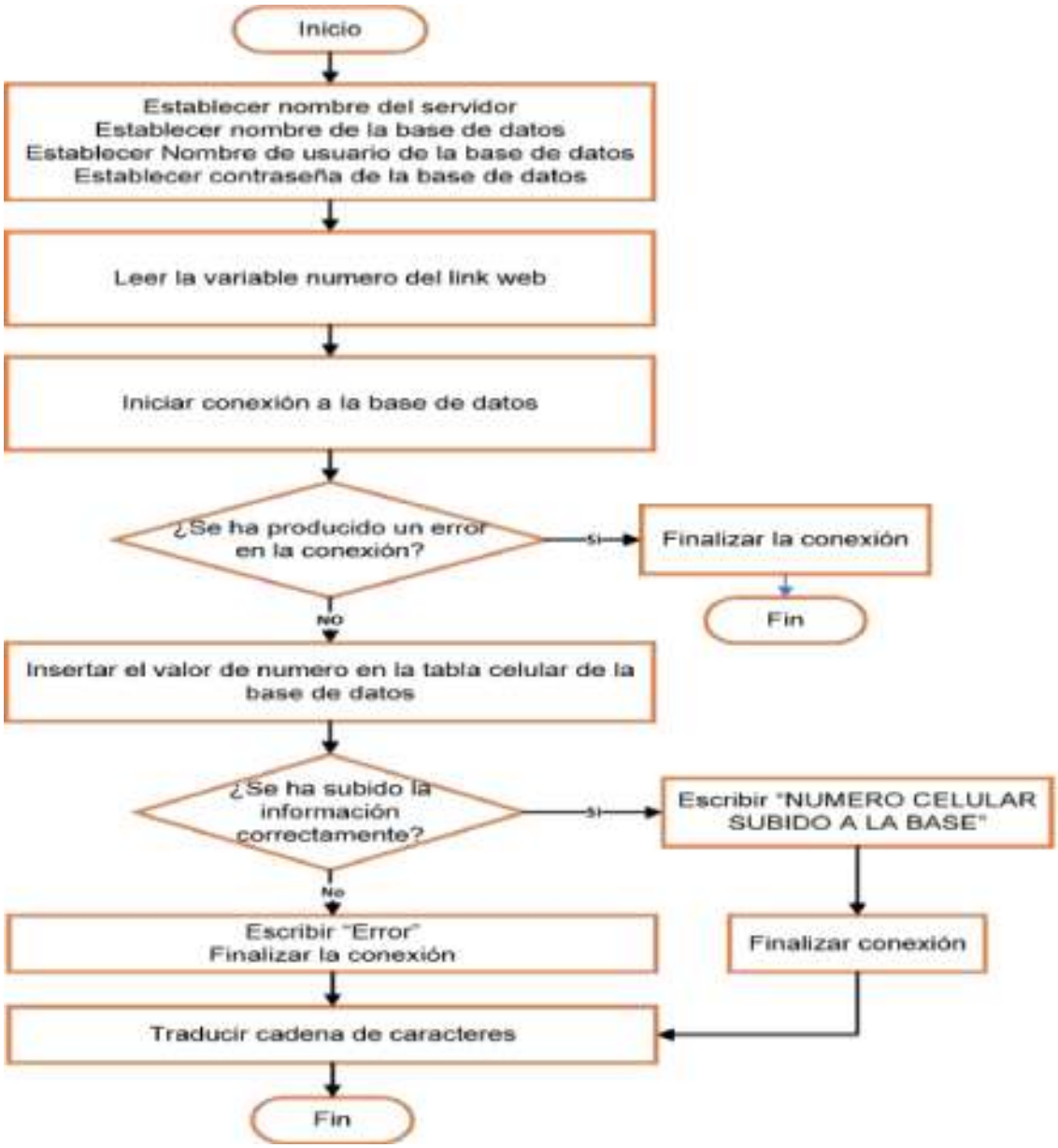


Figura 3.17 Diagrama de flujo "*write_cell.php*"

Aplicación *Android*

Para la manera de interfaz de usuario se desarrolló una aplicación compatible con dispositivos móviles con sistema operativo *Android* 2.3 (*Gingerbread*) o superior. Esta

aplicación de 13.97 (MB) cuenta con una interfaz sencilla e intuitiva que le permite al usuario observar en tiempo real la localización de la persona que envía la alerta por medio del dispositivo GPS sobre un mapa en formato calle sobre el que se sitúan tres marcadores de diferentes formas y colores: una persona, un marcador verde y uno rojo. Estos señalan: la posición del dispositivo que abre la aplicación, el localizador, y el punto de donde inició a moverse el localizador, respectivamente. Tal como se indica en la Figura 3.18.

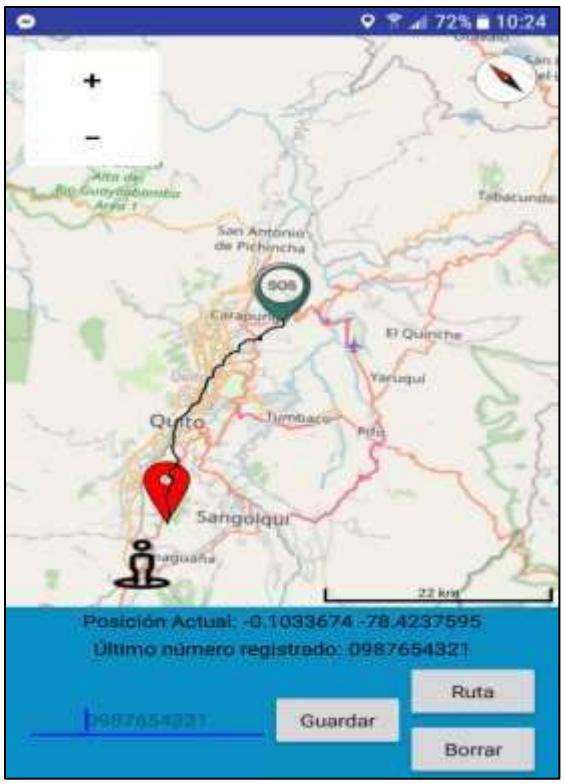


Figura 3.18 Pantalla principal aplicación *Android*

Programación

El programa principal de la aplicación se elaboró sobre la plataforma educativa *MIT App Inventor* desarrollada en colaboración por Google y el Instituto Tecnológico de Massachusetts. *MIT App Inventor* ofrece un entorno de creación de aplicaciones Android dividida en dos secciones los que permiten ubicar elementos visuales en la aplicación y desarrollar el código para el funcionamiento. El tipo de programación que ofrece este entorno es en bloques lo que facilita el desarrollo para usuarios poco experimentados.

Después de abrir la aplicación se muestra la pantalla inicial correspondiente al diagrama de flujo que se muestra en la Figura 3.19. Cuando la aplicación se inicia se muestra el notificador1 y posteriormente se espera en la pantalla inicial hasta que el usuario accione el botón que permite el cambio de pantalla.

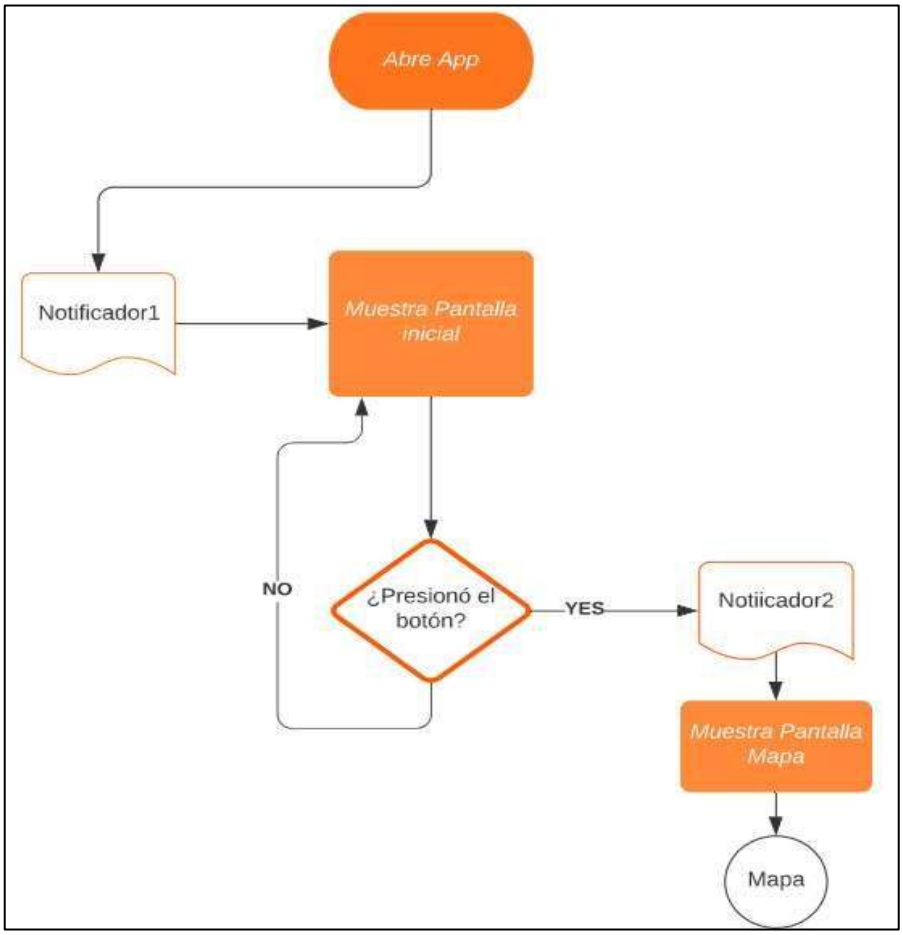


Figura 3.19 Diagrama de flujo pantalla inicial

El programa denominado “Mapa” se muestra en la Figura 3.20 en la que se puede ver que al momento de desplegarse la pantalla principal se crean 4 variables globales vacías: Tabla, Coordenadas, *CELL* y *NumLong*, estas 3 primeras de tipo tabla y la última de tipo número.

En este punto se muestra al usuario un mensaje de notificación recordándole que ingrese un número de contacto. A continuación, se inician los contadores de nombre “*REFRESH*” y “*CENTER*” que cumplen la función de actualizar constantemente la adquisición de datos y ubicar el mapa en el centro de la pantalla. Los contadores “*REFRESH*” y “*CENTER*” son configurados a 1000 y 30000 (ms) respectivamente. Cada vez que uno de estos contadores llega a su valor predeterminado convocará la subrutina de su mismo nombre.

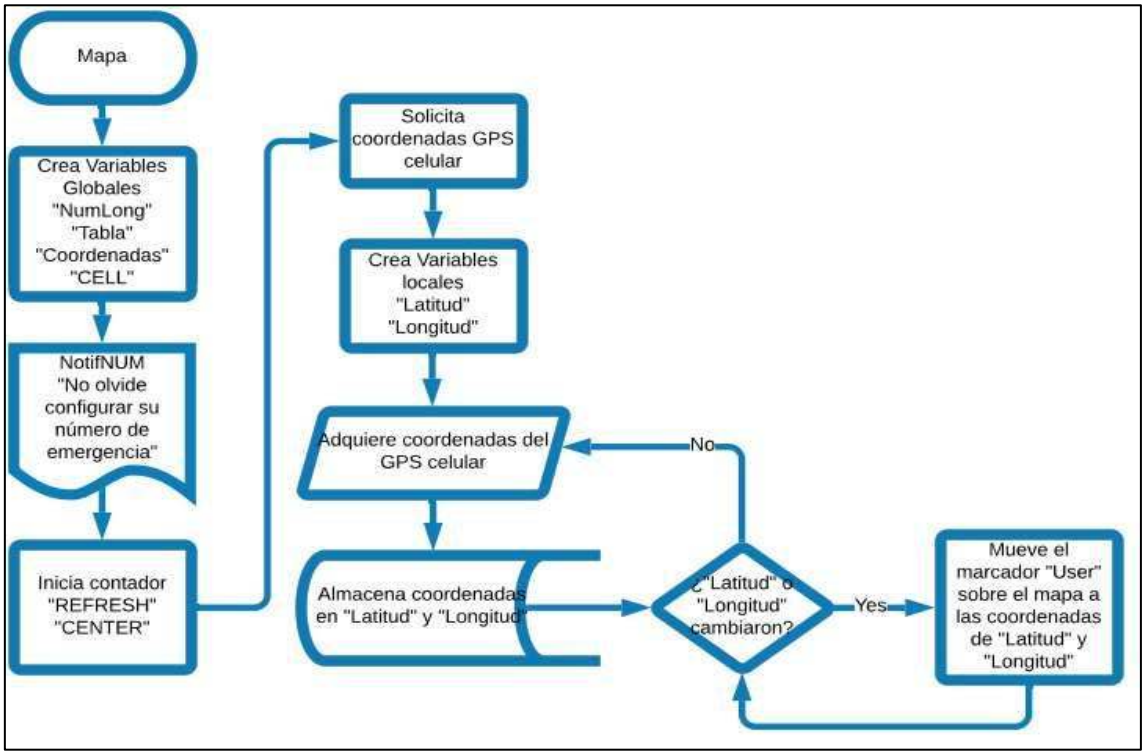


Figura 3.20 Diagrama de flujo pantalla principal

Una vez que se iniciaron los contadores, la aplicación solicita al dispositivo Android los datos de su GPS para almacenar estos en variables locales “Latitud” y “Longitud”, según corresponda. Luego de esto, verifica si estos valores han variado para, de ser el caso, mover el marcador “*User*” correspondiente al *smartphone* que abre la aplicación. Esta verificación se continuará realizando en caso de que los valores no cambien.

La subrutina asociada al contador “*REFRESH*” se muestra en la Figura 3.21 En primer lugar, se vacía la variable “Coordenadas” para después preparar la conexión con el servidor al establecer el enlace de donde se adquirirán los valores almacenados en la base de nombre “Sensor”. Luego de ser adquiridos, el archivo tipo JSON es decodificado y guardado en la variable “Tabla”.

A continuación, se ejecuta una nueva subrutina llamada “Formato” que es la encargada de seleccionar los datos a usar e inmediatamente se ejecuta la subrutina “ubiMark” responsable de colocar en posición al marcador “*Help*” Después, se limpia la variable “*CELL*” y se coloca el enlace que permite obtener el número ingresado por el usuario de la base de nombre “celular” y se decodifican para ser guardados en “*CELL*”. El valor almacenado es concatenado junto a “0” y se coloca en la etiqueta “*PHONE*”.

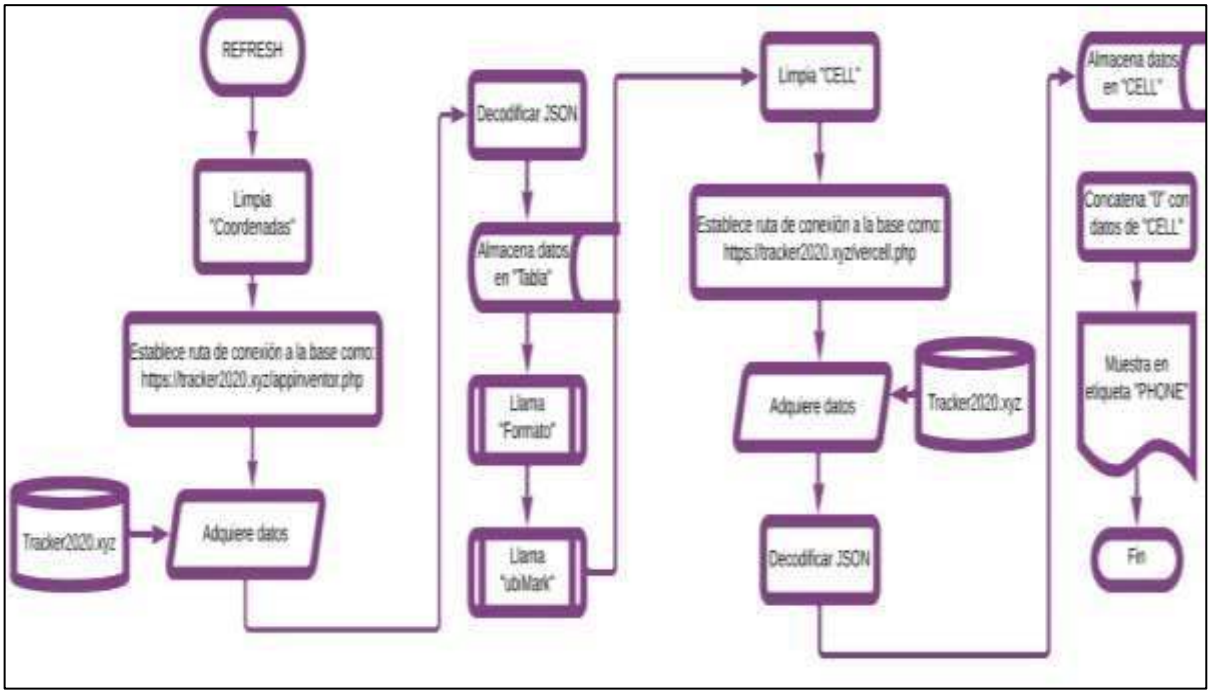


Figura 3.21 Diagrama de flujo subrtina *REFRESH*

La Figura 3.22 muestra el diagrama de flujo de “Formato”. Esta subrutina selecciona los datos recibidos con el esquema que se observa en la Figura 3.23 y los separa con el fin de aislar los datos que serán usados para establecer una posición.

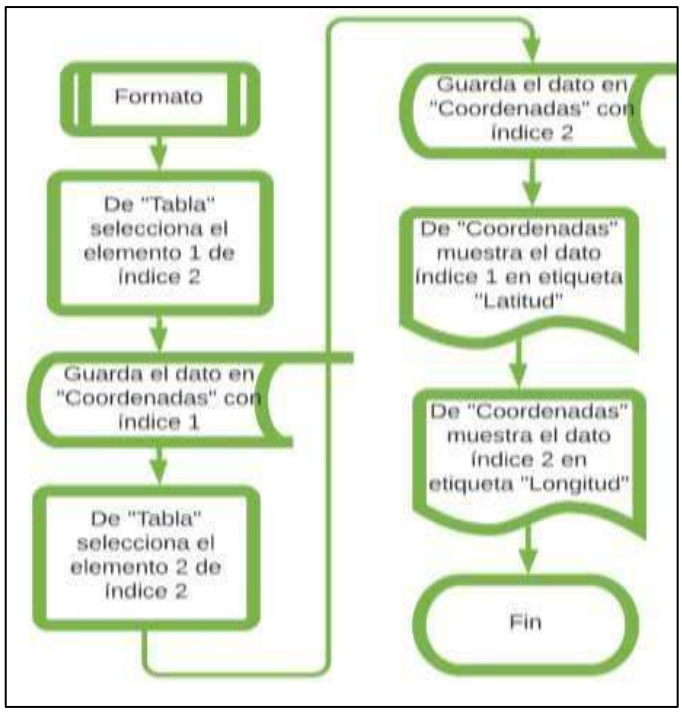


Figura 3.22 Diagrama de flujo subrutina “Formato”

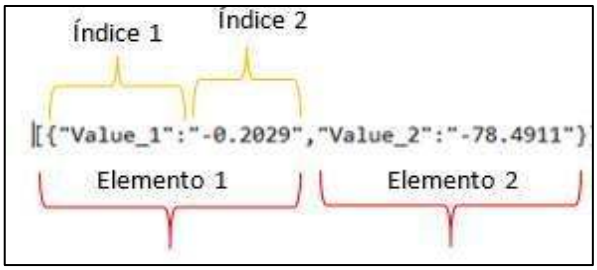


Figura 3.23 Esquema de datos recibidos

Los datos, resultado de “Formato”, son utilizados para mover el marcador ligado al localizador, llamado “Help”. Para esto se ejecuta la subrutina de la Figura 3.24 en donde se toman los valores de latitud y longitud y almacenan en las variables “Latitud1” y “Longitud1”, necesarias para posicionar el marcador “*Help*” y mostrarlos al usuario en las etiquetas correspondientes.

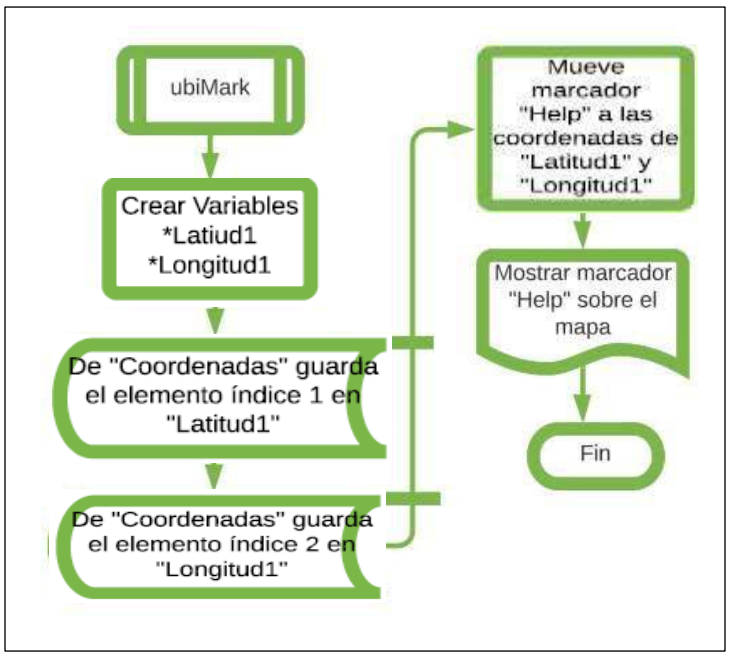


Figura 3.24 Diagrama subrutina "*ubiMark*"

En la Figura 3.25 se muestra la secuencia de pasos que sigue la subrutina número que se ejecuta cuando el usuario presiona el botón “Guardar” para configurar el número de emergencia. Una vez que se presiona el botón se obtiene la longitud de caracteres ingresados en el campo de texto “*Phone*” para guardarlos en la variable “NumLong”. Se evalúa “NumLong” y en caso de tener un valor distinto a 10 se muestra el Notificador Error y no se configura el número. Caso contrario, se verifica si el segundo dígito, de izquierda a derecha, ingresado es 9. De no ser así se notifica el error y no se guarda el número en la base.

Estos pasos se realizan con el fin de reducir la posibilidad de ingresar un número incorrecto. Luego de confirmar que la longitud del número y su segundo dígito son correctos se concatena la dirección web del servidor junto al número colocado en el campo de texto y se realiza el *POST* de dicho número en la base de datos, para a continuación mostrar en la etiqueta "Último número registrado" el número enviado a la base y finalmente se vacía el cuadro de texto.

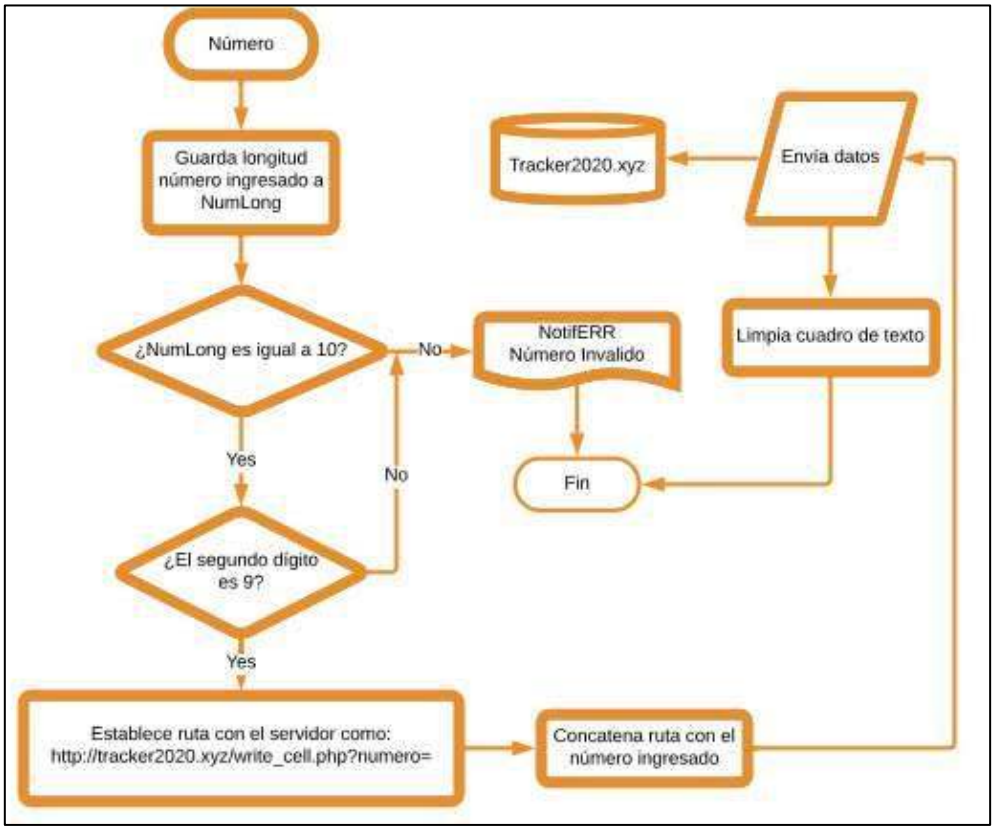


Figura 3.25 Diagrama de flujo subrutina "Numero"

El diagrama de flujo de la Figura 3.26 señala el proceso a ejecutar de la interrupción del contador "*CENTER*". Aquí los valores de las variables "Latitud1" y "Longitud1" son tomados para centrar el mapa y establecer un valor de acercamiento de 10. Esta subrutina se ejecuta con el propósito de mantener visible el marcador de auxilio.

El usuario dispone de la opción de trazado de ruta que señala las calles a recorrer para llegar al dispositivo localizador. Para hacer uso de esta función se debe pulsar el botón Ruta de la pantalla Mapa, lo que ejecutará la secuencia de eventos señalada en la Figura 3.27.

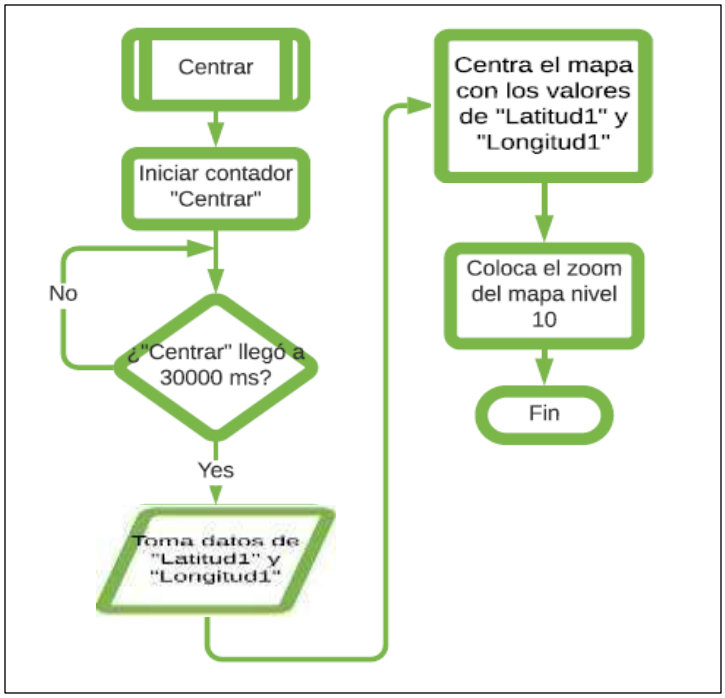


Figura 3.26 Diagrama de flujo interrupción "Centrar"

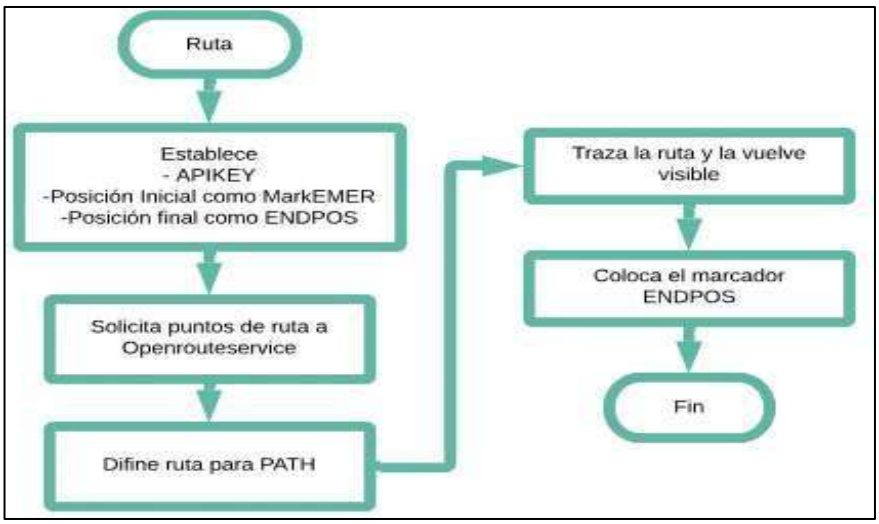


Figura 3.27 Diagrama de flujo subrutina "Ruta"

Primero, se toman como puntos inicial y final a la posición de los marcadores *ENDPOS* (Marcador rojo) y *MarkEMER* (marcador verde) respectivamente. Además, se establece el api *key* del servicio de ruta prestado por *OpenRoute Service.* Luego, se realiza la solicitud de los puntos a este servicio y se define la ruta a seguir. Además, se coloca la ruta mediante una línea de color negro sobre el mapa, así como el marcador *ENDPOS*.

Puesto que la solicitud para trazar la ruta entre los marcadores se acciona de forma manual por el usuario una vez el marcador verde cambie de ubicación la ruta sobre el

mapa no se cambiará con el marcador. Con el fin de actualizar la ruta será necesario volver a pulsar dicho botón.

En caso de que el usuario desee dejar de observar la ruta sobre el mapa debe pulsar el botón Borrar. Esto pondrá en marcha los pasos indicados en el diagrama de la Figura 3.28. Para empezar, se limpian las variables Latitud y Longitud. Después, se retira del mapa el marcador *ENDPOS* y la línea que señala la ruta, para inmediatamente almacenar los valores actuales de Latitud y Longitud, que serán reservados para un nuevo trazado de ruta.

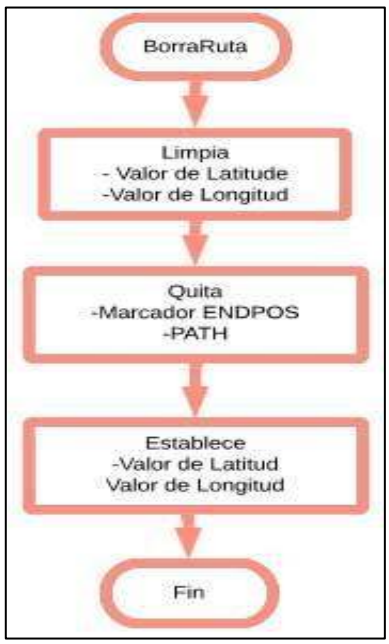


Figura 3.28 Diagrama de flujo subrutina "BorraRuta"

El código en bloques desarrollado para cada una de las rutinas explicadas se puede encontrar en el **Anexo B8**

3.4 Pruebas de funcionamiento del sistema de localización

Prueba GPRS y GPS sobre placa de prueba

Con el fin de revisar el funcionamiento de los módulos SIM 800L y NEO 6M, asegurar la recepción de datos NMEA y verificar la salida a Internet del sistema se realizaron varias pruebas operativas por medio de conectar los 2 módulos al controlador sobre una placa de pruebas como la Figura 3.29 muestra.

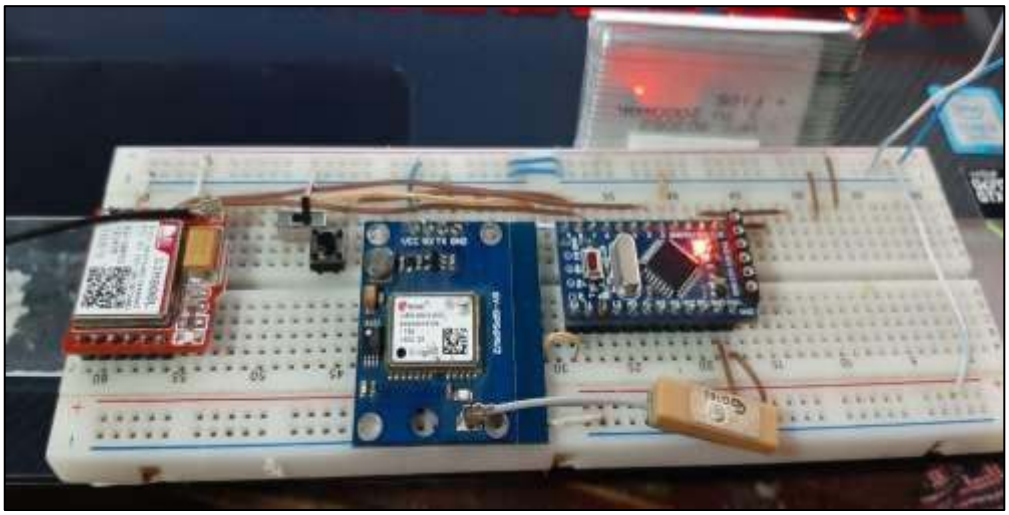


Figura 3.29 Prueba GPRS y GPS sobre *protoboard*

En primer lugar, se revisó el funcionamiento eléctrico de los dos módulos para luego de esto proceder con las pruebas individuales. Se ejecutó un código breve que permitió testear las funciones de envío y recepción de datos, envío de SMS y salida de llamadas para el SIM 800L. Mientras que para el módulo NEO 6M se evaluó la recepción de datos NMEA.

Para la conexión a la red celular se usó la uSIM de la operadora Tuenti, a su vez se emplearon comandos AT enviados a través de la placa *Arduino* al módulo SIM800 mediante un enlace serial. En la Figura 3.30 se muestran los mensajes adquiridos por el monitor serial presente en la interfaz gráfica *Arduino*, donde las líneas de texto iniciadas con “AT” son comandos enviados por la placa *Arduino* y las restantes son las respuestas por parte del módulo GSM.

La línea señalada con el cuadro rojo señala indica que el resultado de la prueba fue exitoso. En la línea +HTTPACTION:0,200,38 el valor 0 representa que se ejecutó el método GET sobre la URL definida. Además, el valor 200 indica el resultado positivo y el valor 38 denota el número bytes que fueron enviados.

```
at+csq
+CSQ: 13,0

OK
AT+SAPBR=3,1,"CONTYPE","GPRS"
OK
AT+SAPBR=3,1,"APN","internet.tuenti.ec"
OK
AT+SAPBR =1,1
OK
AT+HTTPINIT
OK
AT+HTTPPARA="CID",1
OK
AT+HTTPPARA="URL","http://tracker2020.xyz/write_data.php?value1=-147&value2=1568"
OK
AT+HTTPACTION=0
OK

+HTTPACTION: 0,200,38
AT+HTTPTERM
OK
```

Figura 3.30 Monitor serial prueba envío datos

La prueba del módulo GPS se llevó a cabo mediante una rutina de obtención y decodificación de datos de posicionamiento recibidos con codificación NMEA como se observa en la Figura 3.30.

```
COM6

13:52:32.439 -> $GPGLL,0004.75097,N,07806.50197,W,185233.00,A,A*7E
13:52:33.108 -> $GPRMC,185234.00,A,0004.75106,N,07806.50210,W,0.893,,191220,,,A*6E
13:52:33.203 -> $GPVTG,,T,,M,0.893,N,1.653,K,A*20
13:52:33.203 -> $GPGGA,185234.00,0004.75106,N,07806.50210,W,1,04,3.82,2963.4,M,13.9,M,,*75
13:52:33.298 -> $GPGSA,A,3,23,16,26,10,,,,,,,,,6.03,3.82,4.67*08
13:52:33.346 -> $GPGSV,2,1,05,10,46,150,37,16,58,212,28,20,12,147,22,23,15,147,27*7C
13:52:33.440 -> $GPGSV,2,2,05,26,81,348,26*4A
13:52:33.440 -> $GPGLL,0004.75106,N,07806.50210,W,185234.00,A,A*7C
13:52:34.101 -> $GPRMC,185235.00,A,0004.75094,N,07806.50207,W,0.632,,191220,,,A*66
13:52:34.197 -> $GPVTG,,T,,M,0.632,N,1.170,K,A*23
13:52:34.197 -> $GPGGA,185235.00,0004.75094,N,07806.50207,W,1,04,3.82,2963.9,M,13.9,M,,*75
13:52:34.289 -> $GPGSA,A,3,23,16,26,10,,,,,,,,,6.03,3.82,4.67*08
13:52:34.336 -> $GPGSV,2,1,05,10,46,150,35,16,58,212,29,20,12,147,21,23,15,147,26*7D
13:52:34.428 -> $GPGSV,2,2,05,26,81,348,26*4A
13:52:34.475 -> $GPGLL,0004.75094,N,07806.50207,W,185235.00,A,A*71
```

Figura 3.31 Tramas GPS codificadas

Una vez decodificados los datos, se extraen los valores de longitud y latitud y se exponen de manera decimal, este proceso se muestra en la Figura 3.32.

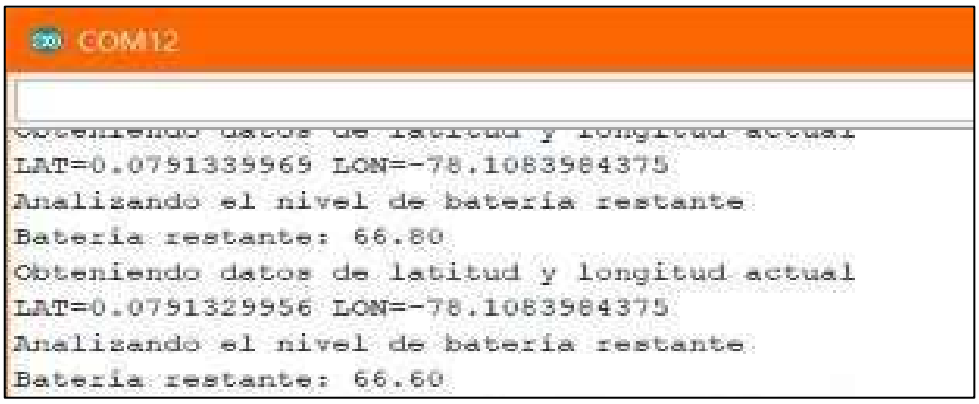


Figura 3.32 Tramas GPS decodificadas

Pruebas de envío de SMS y llamadas

Una vez presionado el botón de pánico, el dispositivo de localización debe realizar una llamada de 10 segundos y enviar un mensaje de alerta a un contacto. En la Figura 3.33 se muestran los mensajes del monitor serial, donde se evidencia este proceso de manera satisfactoria, recibiendo en primera instancia el número, y luego realizando la llamada al celular de contacto y finalmente enviando un SMS al mismo destinatario como se observa en la Figura 3.34

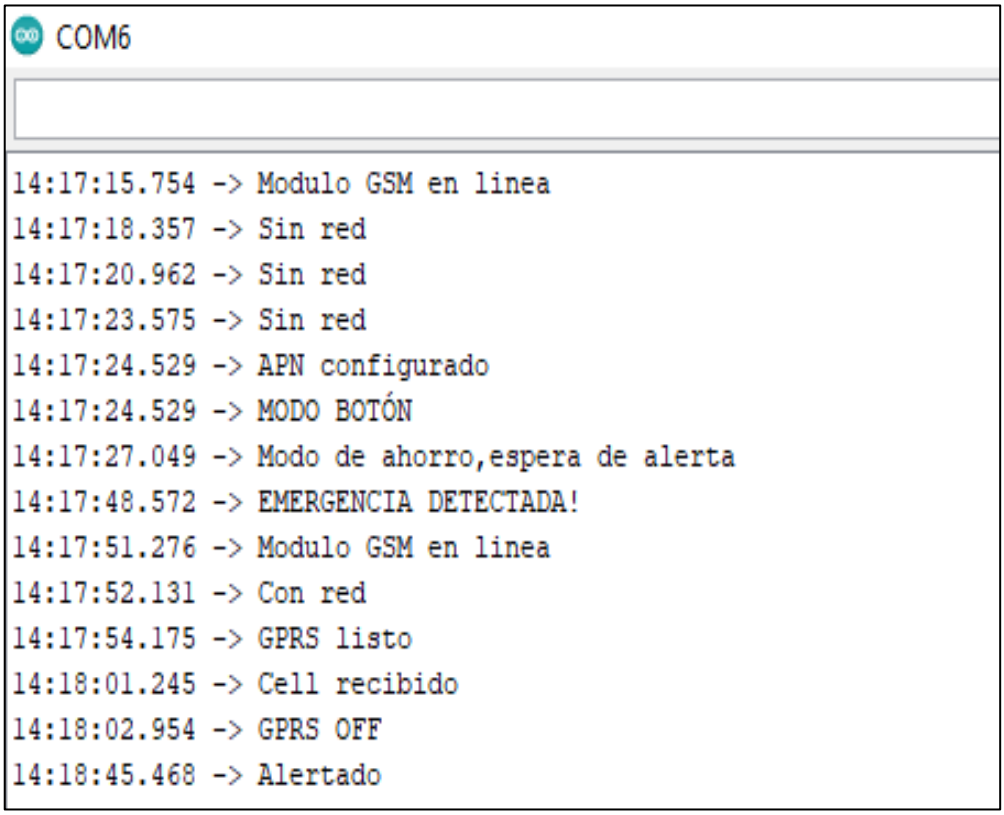


Figura 3.33 Prueba llamada y SMS

Figura 3.34 Llamada y SMS de emergencia recibidos

Prueba de aplicaciones

Para comprobar el correcto funcionamiento de las interfaces de usuario se realizó una verificación manual y otra automática. El procedimiento consistió en la introducción repetida de direcciones al azar a la base de coordenadas de forma manual con el fin de revisar las características de actualización en tiempo real, autocentrado, *autozoom* y, en el caso de la aplicación Android, se comprobó también la función de introducción de números de emergencia. Se evaluaron los mismos parámetros para el caso de la actualización de datos automática.

La aplicación web fue puesta a prueba desde un computador de escritorio con sistema *Windows* en el navegador *Google Chrome*, mientras para la aplicación *Android* se realizó desde un *smartphone* con sistema *Android Marshmallow* 6.0.

En los dos casos los resultados para actualización de datos manual como automática fueron positivos como las Figura 3.35 y Figura 3.36 señalan. Las dos pruebas mostraron una actualización en tiempo real y buen desempeño de las funciones de autocentrado y *autozoom.*

Figura 3.35 Prueba aplicación Web

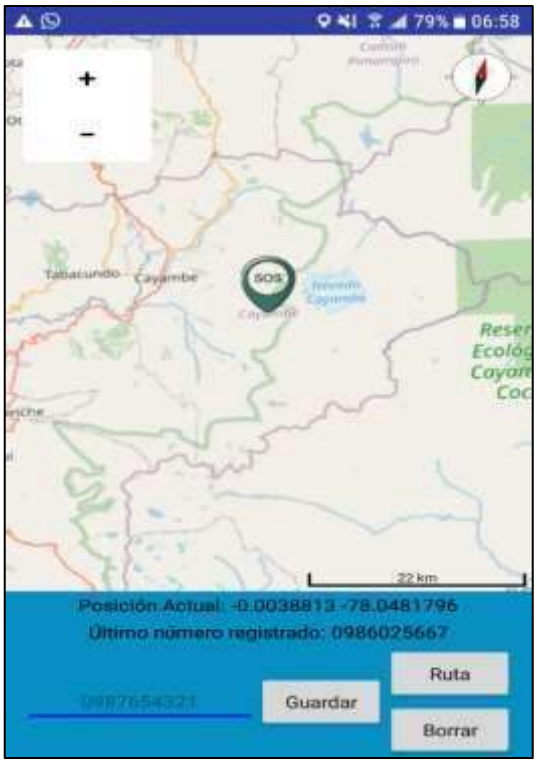


Figura 3.36 Prueba aplicación Android

La Figura 3.37 muestra la prueba del algoritmo para números erróneos de la aplicación Android que como se evidencia funciona correctamente al tratar de introducir números que no empiezan por “09” y no tengan extensión de 10 dígitos.

Figura 3.37 Prueba algoritmo número celular

Prueba sistema de localización

La Figura 3.38 indica los resultados de la ejecución del modo espía en la placa de pruebas. Se observa que para iniciar se revisa el funcionamiento del módulo GSM y se configura el APN. Posterior a esto, se revisa el nivel de batería y el modo de operación. A continuación, se construye el *string* de valores con los datos decodificados del GPS y son enviados a la base de datos para su visualización en la aplicación web o *Android*.

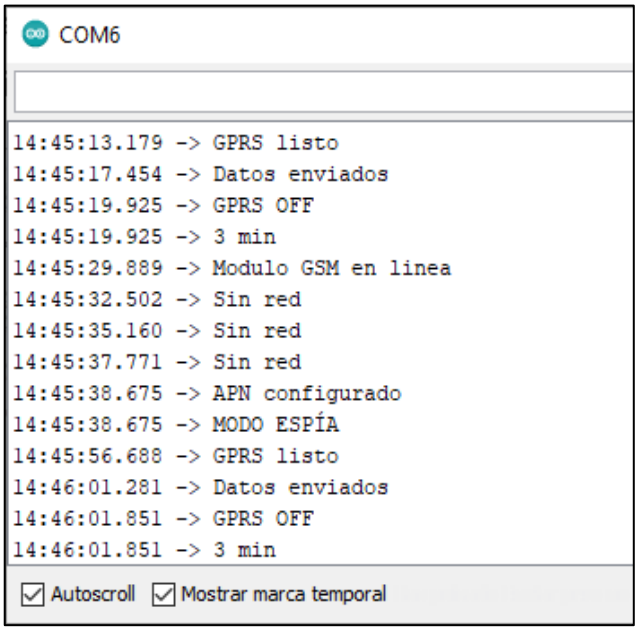


Figura 3.38 Prueba funcionamiento modo espía

De igual manera en la Figura 3.39 se observa el funcionamiento del sistema en modo botón de pánico, en el cual se realiza una subrutina extra en comparación al modo espía.

En primera instancia se recibe el número celular de contacto desde la base de datos, luego se realiza una llamada de corta duración, luego de colgar la llamada se construye un *string* con el texto que se envía vía SMS, dando paso a la lectura de datos GPS y la construcción de *string* de datos que se envía a la base de datos. Este proceso se repite cada 30 segundos, cabe aclarar que la llamada y el SMS se envían la primera vez de este ciclo.

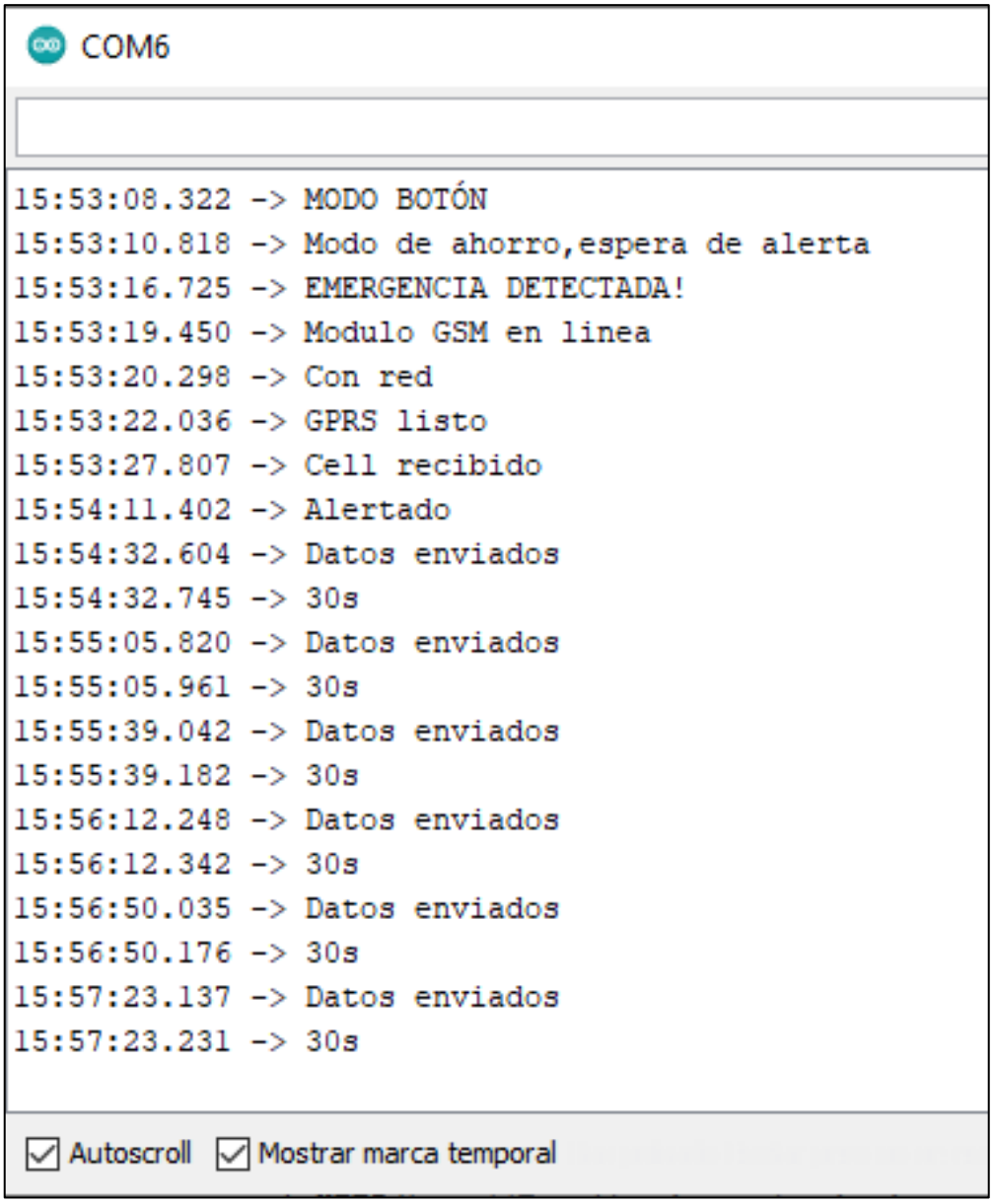


Figura 3.39 Funcionamiento modo emergencia

Costos

En la Tabla 3.14 se puede encontrar el costo por materiales, mano de obra y diseño que estuvo involucrado en la implementación del proyecto. Cabe resaltar que el tiempo de mano de obra es referente al tiempo dedicado al armado del localizador y programación de las aplicaciones.

Tabla 3.14 Costos por implementación

Detalle del material	Cantidad	Precio unitario	Total
Módulo GPS Neo 6M	1	14.5	14.5
Módulo GPRS Sim800L	1	13	13
Arduino Pro mini 3.3 (v)	1	4	4
Programador Arduino	1	1.5	1.5
Cargador USB Li-Po	1	5.5	5.5
Batería Li-Po 903052	1	13	13
Impresión 3D	1	10	10
uSim Claro	1	5	5
Hosting por 1 año	1	8.82	8.82
Aislante termoretráctil 2 (m)	1	2	2
Baquelita 10*10(cm)	1	2.5	2.5
Baquelita perforada 10*10 (cm)	1	2.5	2.5
LED rojo	1	0.1	0.1
LED verde	1	0.1	0.1
Resistencias			0
Espadines hembra	2	2.2	4.4
Pulsador	1	0.15	0.15
Switch	2	0.15	0.3
Cable AWG 24 1 (m)	2	0.4	0.8
Varios (estaño, cloruro férrico, pomada, etc)	1	14.5	14.5
Gastos de envío	3	5	15
Mano de obra 100 horas	2	250	500
Total			617.67$

3.5 Manual de Uso y Mantenimiento

Vínculo: https://youtu.be/BotoO64x5NA

4 CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

- La implementación del sistema de localización representa una ayuda efectiva para los cuidadores de personas como menores de edad, adultos mayores y personas con dificultades cognitivas, debido a que permite al cuidador realizar una supervisión constante de la persona y proporciona alertas de auxilio enviadas a un número celular establecido.

- El análisis de las necesidades del grupo objetivo permitió reconocer que el dispositivo de localización debe tener un equilibrio entre la relación tamaño y la facilidad de transporte. Por lo tanto, fueron considerados en la implementación del prototipo con la finalidad que el usuario del localizador pueda transportar el artefacto de manera sencilla y que sus dimensiones no hagan que este sea fácil de extraviarse.

- El uso de programación en bloques para el desarrollo de la aplicación Android resulta más sencillo, en comparación con los lenguajes de programación habituales como Java, para desarrolladores con poca experiencia y ofrece la oportunidad de obtener aplicaciones con diversas características completamente funcionales.

- Aprovechar la precisión de trabajo de la red GPS, la amplia gama de dispositivos compatibles con esta tecnología y la amplia cobertura de la red de datos celular 2G facilitó la creación de un sistema de localización fiable y con buen desempeño en distintas zonas urbanas que resulta efectivo en el cuidado de adultos mayores y menores de edad.

- El proyecto desarrollado hizo uso de los conocimientos adquiridos en la carrera en campos como el manejo de microcontroladores y sensores, al igual que fundamentos teóricos en lo relacionado con la transmisión y manejo de información, pero, además, añade habilidades en el manejo de bases de datos, creación y desarrollo de aplicaciones que benefician el perfil de egreso del estudiante.

- Luego de realizadas las pruebas de funcionamiento e interacción de los módulos GSM y GPS con la placa Arduino, se concluye que los módulos interactúan de manera correcta con el sistema que los comanda en este caso Arduino, recibiendo comandos a través de sus puertos seriales uno a la vez y enviando los datos de control al microprocesador el cual se encarga de entenderlas y seguir ejecutando el flujo de trabajo en función a estas.

4.2 Recomendaciones

- Para trabajos futuros se recomienda incluir otras tecnologías que posibiliten una cobertura mayor a la que ofrece la red celular con el fin de aumentar la confiabilidad del sistema de localización en un mayor número de escenarios.

- Para la mejora del proyecto en trabajos futuros es recomendable añadir características de seguridad para las aplicaciones como delimitar áreas seguras para prevenir que el usuario del localizador salga del perímetro seguro o establecer más de un número de emergencia al que lleguen las alertas de auxilio.

- En ciertas ocasiones resulta de gran utilidad poder tener un control remoto del funcionamiento de un dispositivo, es decir, activarlo o solicitar algún dato a la distancia, por esto, se recomienda implementar un método que permita activar el dispositivo de rastreo de manera remota, sin necesidad de tener acceso físico a este.

- Dado el avance tecnológico en los sistemas de radionavegación se recomienda examinar las nuevas tecnologías y sus mejoras con el propósito de asegurar la selección de la mejor alternativa de localización, en la búsqueda de realizar sistemas más precisos y confiables de ayuda para personas que necesiten de una supervisión debido a circunstancias médicas o de otra índole.

- Con la finalidad de obtener resultados precisos y evitar cualquier error de diseño se recomienda dejar al menos 0.2 (mm) de tolerancia en los diseños para impresión 3D.

5 REFERENCIAS BIBLIOGRÁFICAS

[1] National Coordination Office for Space-Based Positioning, Navigation and Timing, «The Global Positioning System,» NOAA, abril 2020. [En línea]. Available: https://www.gps.gov/systems/gps/. [Último acceso: mayo 2020].

[2] Intenational Telecomunication Union, «Description of systems and networks in the radionavigation-satellite service (space-to Earth and space-to-space) and technical characteristics of transmitting space stations operating in the bands 1 164 MHz-1 125 MHz, 1 115 MHz-1 300 Mhz and 1 559- 1 610M,» marzo 2018. [En línea]. Available: https://www.itu.int/dms_pubrec/itu-r/rec/m/R-REC-M.1787-3-201803-I!!PDF-E.pdf. [Último acceso: mayo 2020].

[3] J. Sánchez, «Análisis y Estudio de Redes GPRS,» 2005. [En línea]. Available: http://cybertesis.uach.cl/tesis/uach/2005/bmfcis211a/doc/bmfcis211a.pdf. [Último acceso: mayo 2020].

[4] E. C. i. Farré, «Introducción al sistema GPRS y a su gestión de recursos radio,» abril 2003. [En línea]. Available: https://upcommons.upc.edu/bitstream/handle/2099/9886/Article008.pdf?sequence=1&isAllowed=y. [Último acceso: mayo 2020].

[5] G. Maguire Jr., «GSM,GPRS, SMS, International Roaming, OAM,» marzo 2002. [En línea]. Available: https://www.kth.se/social/files/54803dfef27654772ca84b5a/P4-Lecture3-2002.pdf. [Último acceso: mayo 2020].

[6] GSMTheory, «GSM Architecture,» Google Sites, NF NF. [En línea]. Available: https://sites.google.com/site/gsmtheory/home/4-2-bearer-independent-circuit-switched-core-network-umts-release-4-/3-base-station-subsystem-bss-. [Último acceso: mayo 2020].

[7] Arduino, «What is Arduino?,» NF NF. [En línea]. Available: https://www.arduino.cc/en/Guide/Introduction. [Último acceso: mayo 2020].

[8] N. Casaña y P. Gómez, «Baterías de litio. La alternativa al plomo y al cadmio,» abril 1996. [En línea]. Available: https://digital.csic.es/bitstream/10261/15686/1/ctslibat.pdf. [Último acceso: mayo 2020].

[9] R. Iglesias, A. Lago, A. Noguerias, C. Martínez-Peñalver, J. Marcos, C. Quintans, M. J. Moure y M. D. Valdés, «Modelado y simulación de una batería de Ion-litio Comercial Multicelda,» enero 2012. [En línea]. Available: https://www.researchgate.net/publication/234588217_MODELADO_Y_SIMULACION_DE_UNA_BATERIA_DE_ION-LITIO_COMERCIAL_MULTICELDA. [Último acceso: mayo 2020].

[10] J. Fonseca, «Celdas, Pilas y Baterías de Ion de Litio una alternativa para?,» enero 2011. [En línea]. Available: https://www.kimerius.com/app/download/5783123155/Celdas%2C+pilas+y+bater%C3%ADas+de+I%C3%B3n-Litio+una+alternativa+para....pdf. [Último acceso: mayo 2020].

[11] J. Heydecke, «Introduction to Lithium Polymer Battery Technology,» noviembre 2018. [En línea]. Available: https://www.jauch.com/downloadfile/5c5050fa5b6510e9a8ad76299baae4e53/white_paper_introduction_to_lipo_battery_technology_11-2018_en.pdf. [Último acceso: mayo 2020].

[12] Z. Vargas C., «La investigación aplicada: una forma de conocer realidades con evidencia científica,» 2009. [En línea]. Available: https://revistas.ucr.ac.cr/index.php/educacion/article/download/538/589/0. [Último acceso: mayo 2020].

[13] ETSI European Innovation Paternship, «Human Factors (HF) - Guidelines for ICT products and services - "Design for All",» 2009. [En línea]. Available: https://www.etsi.org/deliver/etsi_eg/202100_202199/202116/01.02.02_60/eg_202116v010202p.pdf. [Último acceso: noviembre 2020].

[14] EdwinRobotics, «Arduino Pro Mini 328 - 3.3V/8MHz DEV-11114,» [En línea]. Available: https://shop.edwinrobotics.com/boards/97-arduino-pro-mini-328-33v8mhz-dev-11114.html. [Último acceso: mayo 2020].

[15] Arduino, «Arduino Nano,» NF. [En línea]. Available: https://store.arduino.cc/usa/arduino-nano. [Último acceso: octubre 2020].

[16] SparkfunElectronics, «Raspberry Zero,» NF. [En línea]. Available: https://cdn.sparkfun.com/assets/learn_tutorials/6/7/6/PiZero_1.pdf. [Último acceso: octubre 2020].

[17] BBC Mundo, «Cómo funciona GLONASS y porqué el sistema de navegación ruso no tiene el éxito del GPS estadounidense,» octubre 2017. [En línea]. Available: https://www.bbc.com/mundo/noticias-41596292#:~:text=La%20m%C3%A1s%20obvia%20de%20las,el%20caso%20del%20sistema%20ruso.. [Último acceso: octubre 2020].

[18] RoboticsBD, «U-Blox NEO-6M GPS Module,» [En línea]. Available: https://store.roboticsbd.com/plc-cable/797-u-blox-neo-6m-gps-module-robotics-bangladesh.html. [Último acceso: mayo 2020].

[19] MarkerFabs, «Sim808 GPS tracker v1.5 User manual,» NF. [En línea]. Available: https://www.robotshop.com/media/files/pdf/sim808-gps-module-datasheet.pdf. [Último acceso: octubre 2020].

[20] ARCOTEL, «Boleteín estadístico diciembre 2018,» diciembre 2018. [En línea]. Available: https://www.arcotel.gob.ec/wp-content/uploads/2015/01/BOLETIN-ESTADISTICO-Diciembre-2018-v4_4.pdf. [Último acceso: mayo 2020].

[21] Megatronica, «Módulo Sim800l GSM GPRS,» [En línea]. Available: https://www.google.com/url?sa=i&url=https%3A%2F%2Farticulo.mercadolibre.com.ec%2FMEC-425425049-modulo-sim800l-gsm-gprs-arduino-_JM&psig=AOvVaw08eRcbVNdzyTPCZb6HiLb0&ust=1590772901139000&source=images&cd=vfe&ved=0CA0QjhxqFwoTCPDAxoGJ1-kCFQAAAAAdAAAAABAD. [Último acceso: mayo 2020].

[22] TECmikro, «Modulo shield GSM Sim900,» NF. [En línea]. Available: https://tecmikro.com/modulos-shields/483-modulo-shield-gsm-sim900.html. [Último acceso: octubre 2020].

[23] SIMCom Wireless Solutions, «SIM800L_Hardware_Design_V1.00,» agosto 2013. [En línea]. Available: https://img.filipeflop.com/files/download/Datasheet_SIM800L.pdf. [Último acceso: mayo 2020].

[24] u-blox, «NEO-6 u-blox 6 GPS Modules,» 2011. [En línea]. Available: https://www.u-blox.com/sites/default/files/products/documents/NEO-6_DataSheet_(GPS.G6-HW-09005).pdf. [Último acceso: mayo 2020].

[25] The PHP Group, «¿Qué es PHP?,» NF. [En línea]. Available: https://www.php.net/manual/es/intro-whatis.php. [Último acceso: junio 2020].

[26] S. Vílchez y G. Seco, «Descripción y representación de las señales GNSS,» junio 2019. [En línea]. Available: http://openaccess.uoc.edu/webapps/o2/bitstream/10609/97367/7/svilchez10TFM0619memoria.pdf. [Último acceso: mayo 2020].

[27] J.-P. Montillet, «Precise Positioning in Urban Canyons: Applied to the Localisation of Buried Assets,» octubre 2008. [En línea]. Available: https://www.google.com.ec/url?sa=i&url=https%3A%2F%2Fwww.researchgate.net%2Ffigure%2FOverview-of-the-GPS-segments-USArmy-1996_fig3_325626210&psig=AOvVaw0aOaSLrb_9Xrc7q04r4lXp&ust=1590616803352000&source=images&cd=vfe&ved=0CA0QjhxqFwoTCMD_kcHD0ukCFQAAAAAdA. [Último acceso: mayo 2020].

[28] Indonesia Dokumen, «Jarlokar,» agosto 2015. [En línea]. Available: https://fdokumen.com/document/jarlokar.html. [Último acceso: mayo 2020].

[29] A. Ali, D. D. Zachary Zanzinger y B. Stephens, «Open Source Building Science Sensors (OSBSS): A low-cost Arduino-based platform for long-term indoor environmental data collection,» mayo 2016. [En línea]. Available: https://www.sciencedirect.com/science/article/pii/S0360132316300476. [Último acceso: mayo 2020].

6 ANEXOS

Anexo A: ESPECIFICACIONES TÉCNICAS

Anexo A1 Neo 6M

La Figura 6.1 mostrada a continuación presenta la distribución de pines de la placa embebida Arduino Pro Mini

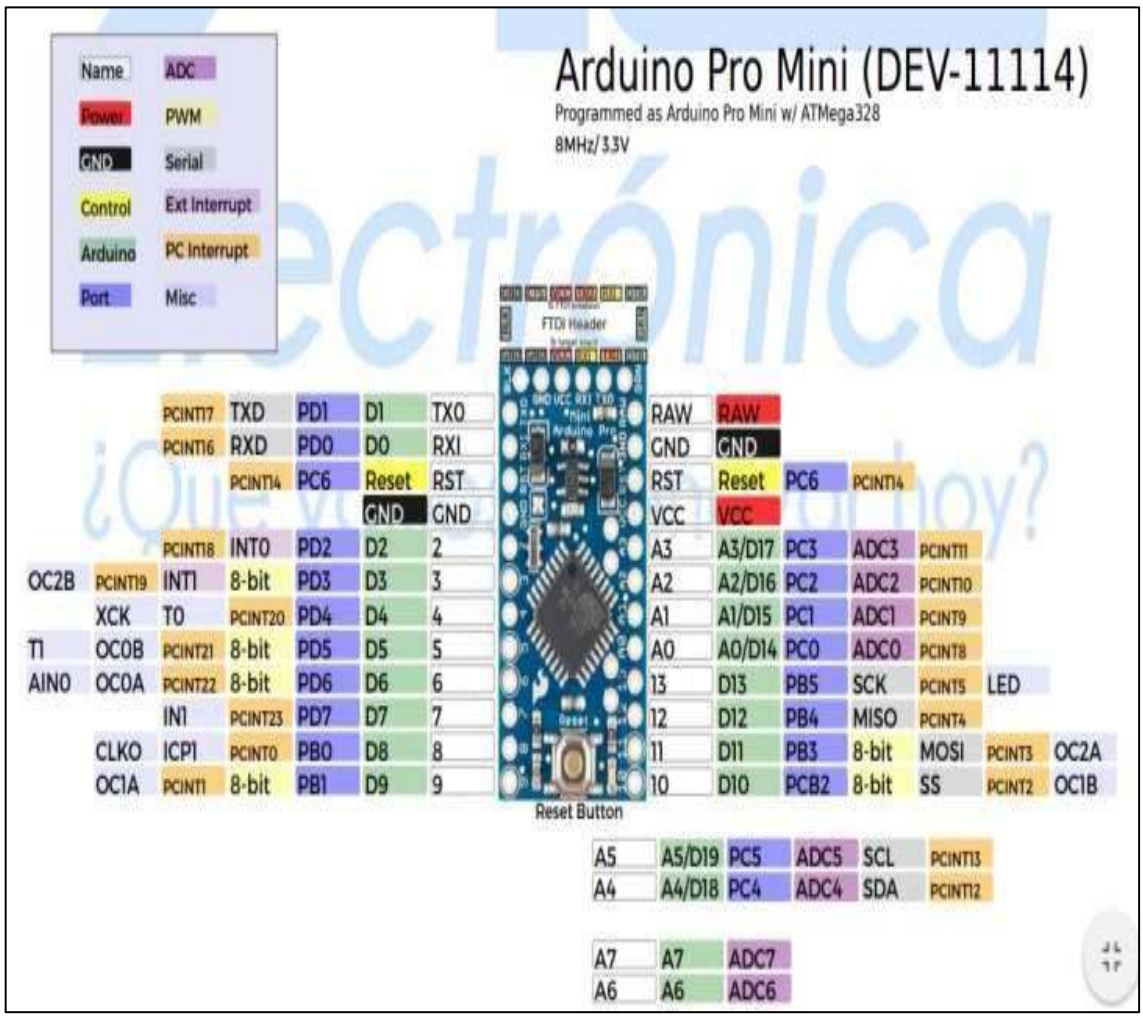


Figura 6.1 Arduino Pro mini [14]

La siguiente Tabla 6.1 indica los parámetros de funcionamiento del Arduino Pro Mini a 3.3 (V)

Tabla 6.1 Especificaciones técnicas Arduino Pro mini [14]

Microcontrolador	ATmega328
Memoria Flash	32 [kB]
Memoria SRAM	2 [kB]
Memoria EEPROM	1 [kB]
Voltaje Operación	3.3 [V]
Corriente Operación	150 [mA]
Pines (I/O) digitales	14
Pines (Entrada) analógicos	8
Comunicación	TTL / RX / TX
Dimensiones	33*18 [mm]
Alimentación Externa	3.3-12 [V] (CD)

Anexo A2 Sim800L

El SIM800L, cuyas frecuencias y datos técnicos se muestran en la Tabla 6.2, es un módulo de conectividad *quad-band* GSM/GPRS. de tamaño reducido que ofrece compatibilidad con la mayoría de los periféricos del usuario entre los que se notan: teclados 5*5*2, USB, canales de audio para micrófono y altavoz de 8 (Ω), slot para micro SIM card y radio FM [23]. Los pines de conexión se detallan en la siguiente Figura 6.2.

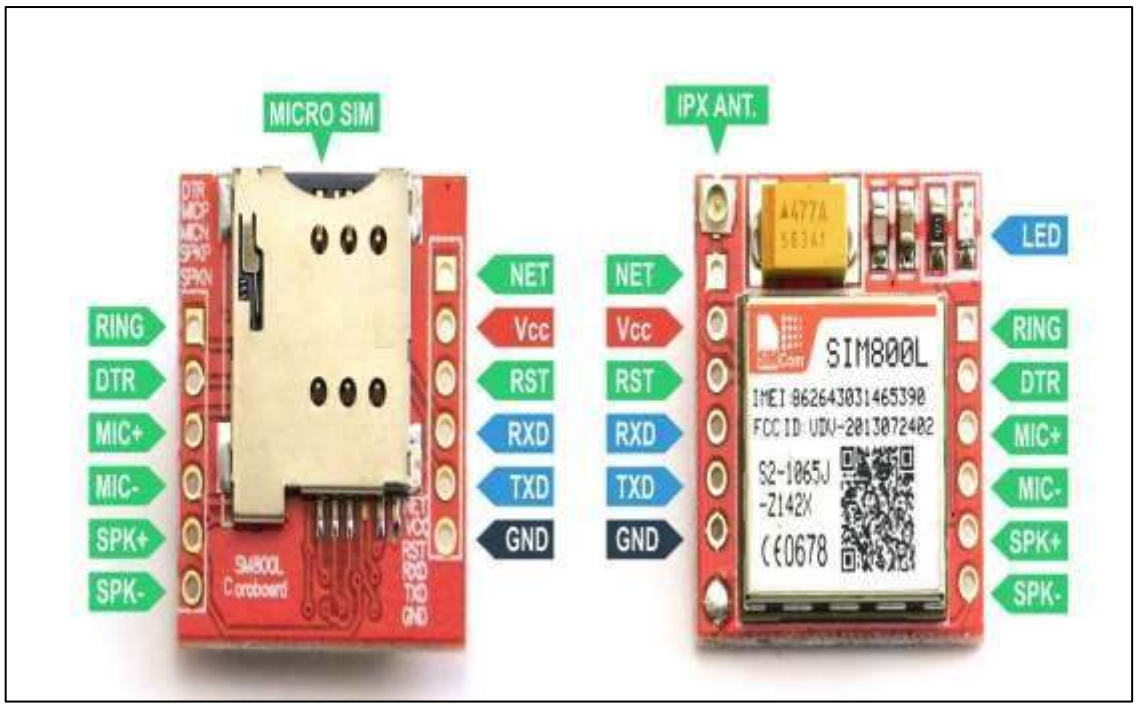


Figura 6.2 Módulo Sim800L [23]

Los valores que se presentan en la Tabla 6.3 son correspondientes al funcionamiento del módulo en modo de envío de datos sin periféricos conectados. Cada periférico conectado al módulo incurrirá en un aumento al consumo de corriente. El consumo de corriente promedio variará de acuerdo con la frecuencia en la que el módulo se encuentre trabajando [23].

Tabla 6.2 Especificaciones técnicas Sim 800L [23]

Características		SIM 800L
Bandas de Frecuencia		GSM 850, EGSM 900, DCS 1800, PCS 1900
Esquemas de Codificación		CS-1, CS-2, CS-3, CS-4
Velocidad de transmisión (máxima)	CS-1	36.2 (kbps)
	CS-2	53.6 (kbps)
	CS-3	62.4 (kbps)
	CS-4	85.6 (kbps)
Conectividad GPRS		Protocolo PAP, TCP/IP, PBCCH
Interfaz SIM		Micro SIM a 1.8 o 3 (V)
SMS		MT, MO, CB, Texto y Modo PDU
Características de Audio		Supresión de ruido, cancelación de eco, Codecs ETS 06.20/06.10/06.50/06.60/06.80 y AMR
Puertos		Puerto serial 1200 a 115200 [bps] Puerto Debug USB_DM y USB_DP
Dimensiones		25x23 (mm)

Tabla 6.3 Especificaciones eléctricas Sim 800L [23]

Características		SIM 800L
Alimentación	Máximo	4.4 (V)
	Promedio	4 (V)
	Mínimo	3.4 (V)
Consumo	Pico Máximo	2 (A)
	Sleep	0.7 (mA)

Anexo A3 Neo 6M

El módulo NEO 6M fabricado por u-blox, mostrado en la Figura 6.3, es un dispositivo de posicionamiento de alto rendimiento, bajo consumo y opciones de conectividad como se puede ver en Tabla 6.4. Este módulo es capaz de operar con una exactitud de 2.5 (m) en el plano horizontal [18]. Las características operacionales se presentan en la Tabla 6.5

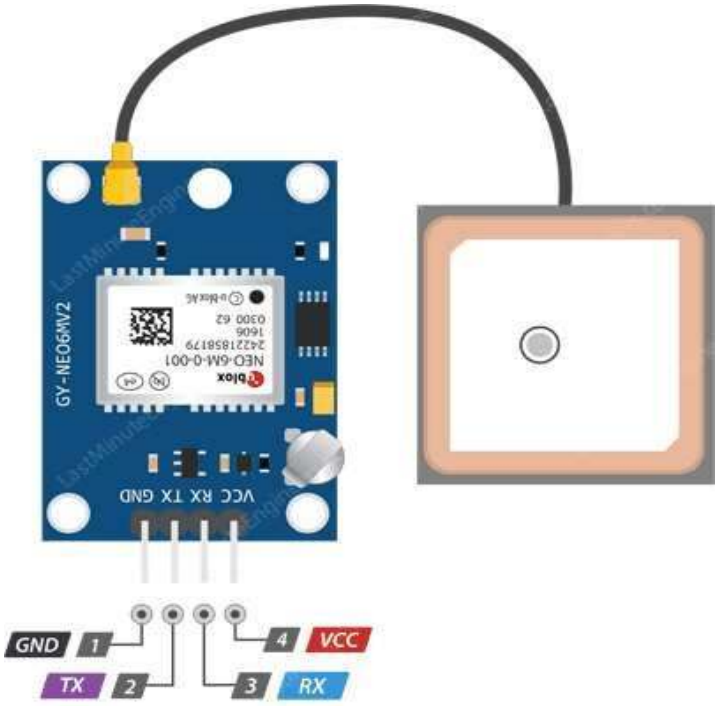


Figura 6.3 Módulo Neo 6M [24]

Tabla 6.4 Especificaciones técnicas Neo 6M [24]

Características		NEO 6M
Sistema Localización		GPS, A-GPS
Frecuencia de Operación		L1
Frecuencia de Actualización		5 (Hz)
Tiempo Primera Adquisición	*Cold Start*	27 (s)
	Warm Start	27 (s)
	Hot Start	1 (s)
	Aistido	27 (s)
Interfaces		UART, USB, SPI, DDC
Protocolos		NMEA, UBX, RTCM
Dimensiones		30 x 20 x 11.4 (mm)

Tabla 6.5 Especificaciones operativas Neo 6M [24]

Características		NEO 6M
Sensibilidad	Navegación/Rastreo	-161 (dBm)
	Readquisición	-160 (dBm)
	Cold Start	-147 (dBm)
	Hot Start	-157 (dBm)
Exactitud	Posición Horizontal	2.5 (m)
	Velocidad	0.1 (m/s)
	Azimut	0.5 (º)
Altitud Máxima		50 (km)
Velocidad Máxima		500 (m/s)

La **Tabla 6.6** expone las características de consumo en condiciones de rastreo en modo de máximo rendimiento. Además, el módulo NEO 6M es capaz de configurar tres modos de consumo: máximo rendimiento, ECO y ahorro de energía por medio de los que se puede llegar a obtener un consumo de rastreo mínimo de 11 (mA) [24].

Tabla 6.6 Especificaciones eléctricas Neo 6M [24]

Características		NEO 6M
Alimentación		2.7- 3.6 (V)
Consumo	Máximo	67 (mA)
	Adquisición	47 (mA)
	Rastreo	39 (mA)

ANEXO B: CÓDIGOS DE PROGRAMACIÓN

Anexo B1 Código *Arduino*

```
#include <SoftwareSerial.h> //Inclusión librería para comunicación serial
#include <TinyGPS.h> //Inclusión librería para decodificación de datos GPS
#include "LowPower.h"
#define interruptPin 2 //Definición de pin para interrupción provocado por la librería LowPower

TinyGPS gps;   //Objeto GPS
SoftwareSerial serialSIM800(7, 8);       //Configuración de pin 7 (RX), 8(TX) para comunicación serial GSM
SoftwareSerial serialGPS(3, 4);          //Configuración de pin 3 (RX), 4(TX) para comunicación serial GPS

float flat, flon; //Variables no enteras que contiene el valor de longitud y latitud
unsigned long periodo = 180000;  //Intervalo de tiempo en ms de actualización de posición
unsigned long tiempoAnterior = 0; //Variable auxiliar de tiempo en ms
unsigned long periodoBAT = 300000;  //Intervalo de tiempo en ms de actualización de nivel de batería
unsigned long AnteriorBAT = 0;  //Variable auxiliar de tiempo en ms
void setup() {
  pinMode(2, INPUT_PULLUP); //Pin 2 como entrada (Boton de Pánico)
  pinMode(10, INPUT_PULLUP); //Pin 10 como entrada (Selector de modo)
  pinMode(5, OUTPUT); //Pin 5 como salida (Led azul)
  pinMode(6, OUTPUT); //Pin 6 como salida (Led rojo)

  Serial.begin(9600); //Inicio serial PC-Arduino a 9600 baudios
  serialGPS.begin(9600);  //Inicio serial GPS-Arduino a 9600 baudios
  serialSIM800.begin(19200); //Inicio serial SIM800(GSM)-Arduino a 19200 baudios
  pruebaSIM800(); //Función de prueba de inicio correcto del módulo SIM800
  pruebaRED();  //Función de prueba de registro en la red celular
  setAPN(); //Función para establecer la APN del proveedor de telefonía
}

void loop() {
```

```
  int selectorMODO = digitalRead(10); //Variable que contiene la lectura del switch de selección de modo
  if(selectorMODO == 1) { //switch abierto (modo espía)
    Serial.println("MODO ESPÍA"); //Mensaje de inicio de modo
    while(1){
    if(millis()-AnteriorBAT >= periodoBAT){ //Condición de cumplimiento del tiempo para mostrar la bateria restante
      nivelBATERIA(); //Función de verificación del porcentaje de batería restante
      AnteriorBAT = millis(); //guarda el tiempo actual como referencia
    } //Fin de la condición de cumplimiento de tiempo para el porcentaje de batería
    if(millis()-tiempoAnterior>=periodo || tiempoAnterior == 0){//Condición de cumplimieno del tiempo para envío de datos
      datosGPS(); //Función de decodificación de datos gps
      iniciarGPRS(); //Función de encendido GPRS
      envioHTTP();  //Función de envío de datos
      finCONECCION();
      tiempoAnterior=millis();  //guarda el tiempo actual como referencia
      Serial.println("3 min"); //Mensaje de espera
      }
    } //Fin de la condición de cumplimiento de tiempo

  } else {  //switch cerrado (modo botón de pánico)
      Serial.println("MODO BOTÓN"); //Mensaje de inicio de modo
      tiempoAnterior=0; //Variable auxiliar
      periodo = 45000; //Tiempo en ms de actualización de posición
      modoSLEEP(); //Función de ahorro de energía
      int estado = digitalRead(2); //variable estado igual al valor del pin con el botón de pánico
      if ( estado==LOW ) { //Botón de pánico presionado
        Serial.println("EMERGENCIA DETECTADA!"); //Mensaje de alerta
        pruebaSIM800(); //Función de prueba de inicio correcto del módulo SIM800
        sendATcommand("AT+CSCLK=0", "OK", 1000);// Evitar que módulo SIM800 entre en ahorro de energía
        sendATcommand("AT+CSCLK=0", "OK", 1000);// Evitar que módulo SIM800 entre en ahorro de energía
        pruebaRED();  //Función de prueba de registro en la red celular
```

```
        iniciarGPRS(); //Función de encendido GPRS
        alertaSMS_CALL(); //Función de envió de mensaje sms y llamada al contacto
        finCONECCION();
        while(1){  //Bucle infinito de envio de posición
          if(millis()-AnteriorBAT >= periodoBAT){ //Condición de cumplimiento del tiempo para mostrar la bateria restante
            nivelBATERIA(); //Función de verificación del porcentaje de batería restante
            AnteriorBAT = millis(); //guarda el tiempo actual como referencia
          } //Fin de la condición de cumplimiento de tiempo para el porcentaje de batería
          if(millis()-tiempoAnterior>=periodo || tiempoAnterior == 0){//Condición de cumplimieno del tiempo para envío de datos
            datosGPS(); //Función de decodificación de datos gps
            setAPN(); //Función para establecer la APN del proveedor de telefonía
            iniciarGPRS(); //Función de encendido GPRS
            envioHTTP();  //Función de envío de datos
            finCONECCION();
            tiempoAnterior=millis();  //guarda el tiempo actual como referencia
            Serial.println("30s"); //Mensaje de espera
          }

        } //Etiqueta de terminación del bucle infinito
      } //Fin condición de botón de pánico presionado
    } //Fin de la condición de modo botón de pánico
} //Final del loop

/*FUNCIÓN PARA PROBAR QUE EL MÓDULO SIM800 ESTÁ FUNCIONANDO*/
void pruebaSIM800(){  //Nombre de la función e inicio
  serialSIM800.listen(); //Atender serial SIM800
  uint8_t answer=0; //variable de 1 byte llamada "answer"
  answer = sendATcommand("AT", "OK", 2000); //Enviar comando "AT" y esperar hasta 2 segundos, si la respuesta del módulo es "OK", answer = 1
  if (answer == 0){ //Condicional de fallo
    //Serial.println("Error al comunicarse con el módulo"); //Imprimir mensaje de error
    while(answer == 0){ //Bucle mientras el módulo no responda
      digitalWrite(6,HIGH); //Led rojo encendido por 1 segundo
      delay(1000);
```

```
    digitalWrite(6,LOW); //Led rojo apagado por 1 segundo
    delay(1000);
    answer = sendATcommand("AT", "OK", 2000); //Enviar comando "AT" y esperar hasta 2 segundos la respuesta
    } //Fin del bucle de no respuesta del módulo
  }
  else{  //Condición de éxito de la conexión al módulo SIM800
    Serial.println("S800 L"); //Imprimir mensaje de éxito
    digitalWrite(5,HIGH); //Led azul encendido por 2 segundo
    delay(2000);
    digitalWrite(5,LOW); //Led azul apagado
  } //Fin de la condición de éxito de conexión al módulo
 }//Fin de la función de Prueba

 /*FUNCIÓN PARA PROBAR EL REGISTRO DEL MÓDULOS SIM800 A LA RED MÓVIL*/
void pruebaRED(){ //Inicio de función de prueba de conexión a la red celular
  serialSIM800.listen();  //Atender serial SIM800
  uint8_t answer1=0; //variable de 1 byte llamada "answer1"
  answer1 = sendATcommand("AT+CREG?", "+CREG: 0,1", 2000); //Enviar comando de comprobación de registro a la red, y comparación por 2 segundos
                                        //si el valor de respuesta del módulo coincide, answer1 = 1
  if (answer1 == 0){  //Condicional de red celular no alcanzada
   while(answer1==0){ //Bucle de espera a la conexión
    Serial.println("Sin red");  //Imprimir en el monitor serial mensaje de error
    answer1 = sendATcommand("AT+CREG?", "+CREG: 0,1", 2000);  //Actualización de la variable answer1
    }  //Fin del bucle de espera a la conexión
  } //Fin de la condición de error
  else{ //Condición inversa (conexión exitosa a la red celular)
    Serial.println("Con red");  //Imprimir mensaje de éxito en la operación
    delay(2000);  //Retraso de 2 segundos
    }//Fin de la condición
  } //Fin de la función de prueba de conexión a la red celular
```

```
/*FUNCIÓN DE CONFIGURACIÓN DEL APN DEL PROVEEDOR*/
void setAPN(){  //Inicio de la función
  serialSIM800.listen(); //Atender serial SIM800
  sendATcommand("AT+SAPBR=3,1,\"Contype\",\"GPRS\"", "OK", 2000); //Establecer el tipo de conexión a GPRS
  sendATcommand("AT+SAPBR=3,1,\"APN\",\"internet.tuenti.ec\"", "OK", 2000);//Establecer APN del proveedor de telefonía
  Serial.println("APN configurado"); //Mensaje de éxito en la operación
  } //Fin de la función

/*FUNCIÓN DE OBTENCIÓN DE NIVEL DE BATERIA RESTANTE*/
void nivelBATERIA(){ //Inicio de la función para verificar el porcentaje de batería restante
  serialSIM800.listen();  //Atender serial SIM 800
  sendATcommand("AT+CBC", " ", 1000); //Enviar comando AT que devuelve la capacidad de batería restante
  String bateria_raw = serialSIM800.readString(); //Guardar el String completo
  int bateria = getValue(bateria_raw,',',1).toInt();   //Filtrar el valor de interés del String entero
  Serial.print("Nivel de batería: "); //Mensaje indicando el valor de batería restante en porcentaje
  Serial.println(bateria);
  if (bateria <= 50){ //Condición de baja batería
    digitalWrite(6, HIGH);  //Encender led rojo
    delay(500); //Retraso de 500 ms
    digitalWrite(6, LOW); //Apagar el led rojo
    } //Fin de la condición de bateria baja
  else{ //Condición inversa de batería aceptable
    digitalWrite(5, HIGH);  //Encender led azul
    delay(500); //Retraso de 500 ms
    digitalWrite(5, LOW); //Apagar led azul
    } //Fin de la condición de batería aceptable
} //Fin función de verificación de estado de la batería

/*FUNCIÓN DE OBTENCIÓN DE DATOS DE POSICIÓN DEL MÓDULO GPS (NEO 6M)*/
void datosGPS(){ //Inicio de la función
```

```
  serialGPS.listen(); //Activar comunicación serial al módulo GPS
   bool newData = false; //variable booleana para aviso de nueva información
   for (unsigned long start = millis(); millis() - start < 1000;){ // Intentar recibir secuencia por 1 seg
      while (serialGPS.available()) //Si en el buffer existe datos
      {
         char c = serialGPS.read(); //Los datos en el buffer serial se almacenan en "c"
         if (gps.encode(c)) //Condicional de decodificación exitosa de "c"
            newData = true; //Variable de nueva información, verdadera
      } //Fin del bucle mientras exista datos en el puerto serial
   } //Fin del bucle de intento de recibir datos por 1 segundo

   if (newData){ //Condicional de existencia de nuevos datos
      gps.f_get_position(&flat, &flon); //Obtener de la trama gps los datos de longitud y latitud en forma "float"
      /*Serial.print("LAT="); //Imprimir la variable de latitud con 10 decimales
      Serial.print(flat == TinyGPS::GPS_INVALID_F_ANGLE ? 0.0 : flat, 10);
      Serial.println(flat, 10);
      Serial.print(" LON=");  //Imprimir la variable de longitud con 10 decimales
      Serial.println(flon == TinyGPS::GPS_INVALID_F_ANGLE ? 0.0 : flon, 10);
      Serial.println(flon, 10);*/
      Serial.println("Datos GPS obtenidos");
   }
  }

/*FUNCIÓN DE CREACIÓN DEL STRING PARA REALIZAR EL REQUEST HTTP*/
String gpsSTRING(){ //Inicio de la función
  String linkHTTP; //Variable de cadena de valores auxiliar
  String sflat = String(flat, 7);  //Convertir de float a string con 7 decimales el valor de latitud
  String sflon = String(flon, 7);  //Convertir de float a string con 7 decimales el valor de longitud
  linkHTTP ="";  //Vaciar de caracteres el string
  linkHTTP                                                                                    += "AT+HTTPPARA=\"URL\",\"http://tracker2020.xyz/write_data.php?value1=";    //Agregar el texto inicial del link
```

```
  linkHTTP += sflat; //Añadir el string con 10 decimales de latitud
  linkHTTP += "&value2=";
  linkHTTP += sflon;  //Añadir el string con 10 decimales de longitud
  linkHTTP += "\""; //Final del String
  return linkHTTP; //retornar el valor
  } //Fin de la función de construcción de comando AT con datos de posicionamiento

/*FUNCIÓN DE ACTIVACIÓN DE RED GPRS*/
void iniciarGPRS(){ //Inicio de la función
  serialSIM800.listen();    //Selección del bus serial conectado al módulo
  uint8_t answer2=0;    //variable de 1 byte llamada "answer2"
  sendATcommand("AT+SAPBR=1,1", "OK", 40000); //Enviar comando de activación
GPRS y espera respuesta 2 segundos
  answer2 = sendATcommand("AT+SAPBR=2,1", "+SAPBR: 1,3,\"0.0.0.0\"", 2000);
//Enviar comando de comprobación de conexión
  if (answer2 == 1){    //Condicional de red GPRS no alcanzada
   while(answer2==1){   //Bucle de espera a la conexión
    Serial.println("Fallo GPRS...");   //Imprimir en el monitor serial mensaje de error
    sendATcommand("AT+SAPBR=1,1", "OK", 40000);   //Reintentar conexión GPRS
    answer2 = sendATcommand("AT+SAPBR=2,1", "+SAPBR: 1,3,\"0.0.0.0\"",
2000);//Actualización de la variable answer2
    }   //Fin del bucle de espera a la conexión
  }   //Fin de la condición de error
  else{   //Condición inversa (conexión exitosa a la red GPRS)
    Serial.println("GPRS listo");  //Imprimir mensaje de éxito en la operación
    delay(2000);  //Retraso de 2 segundos
    }//Fin de la condición
  } //Fin de la función

/*FUNCIÓN PARA ENVIAR LOS DATOS A TRAVÉS DEL SERVICIO HTTP*/
void envioHTTP(){ //Inicio de la función
  serialSIM800.listen();  //Atender serial SIM 800
  String comandoAT = gpsSTRING(); //String comando AT guarda el comando generado
en la función gpsString()
  int str_len = comandoAT.length() + 1; //Longitud del String
  char char_array[str_len]; //Crear char de longitud igual a la del String
```

```
  comandoAT.toCharArray(char_array, str_len);  //Conversión de String a char
  uint8_t answer=0; //Variable de  byte
  answer = sendATcommand("AT+HTTPINIT", "OK", 20000); //enviar comando para inicio de servicio HTTP
  if (answer == 1){ //Si es exitosa la operación
    answer = sendATcommand("AT+HTTPPARA=\"CID\",1", "OK", 5000);// Entrar a la configuración
    if (answer == 1){ //Si es exitosa la operación
      answer = sendATcommand(char_array, "OK", 5000); //Enviar comando con datos de posicionamiento
      if (answer == 1){ //Si es exitosa la operación
        answer = sendATcommand("AT+HTTPACTION=0", "+HTTPACTION: 0,200,38", 10000);  //Enviar un post del ur indicado
        if (answer == 1){ ////Si es exitosa la operación
          Serial.println("Datos enviados"); //Mensaje de éxito
        }
        else{ //Si existe un error al acceder en el URL indicado
          errorURL(); //Función de mensaje de error en el acceso
        }
      }
      else{ //Si existe un error en el URL indicado
        errorCURL(); //Función de mensaje de error en el URL
      }
    }
    else{ //Si existe un error en acceder a la configuración
      errorCID(); //Función de mensaje de error en el acceso
    }
  }
  else{ //Si existe un error en acceder al iniciar servicio HTTP
    errorHTTP();  //Función de mensaje de error en el inicio del servicio
    }
  //finCONECCION();
  }
/*Función de mensaje de error en el acceso al URL*/
void errorURL(){
    Serial.println("Error al acceder al url indicado");
```

```
 }
/*Función de mensaje de error en el URL*/
void errorCURL(){
   Serial.println("Error al configurar el url");
   }
/*Función de mensaje de error en el acceso a la configuración*/
void errorCID(){
   Serial.println("Error configurando los parámetros CID");
   }
/*Función de mensaje de error en el inicio del servicio HTTP*/
void errorHTTP(){
   Serial.println("Error iniciando el servicio HTTP");
 }
/*Función de mensaje de GPRS apagado*/
void offGPRS(){
   Serial.println("GPRS OFF");
 }

/*FUNCIÓN DE OBTENCIÓN DE NÚMERO CELULAR DE CONTACTO*/
String recepHTTP(){ //Inicio de la función que retorna un String
 serialSIM800.listen(); //Atender serial SIM 800
 Serial.println("Obteniendo número"); //Mensaje de inicio del proceso
 uint8_t answer=0; //variable de 1 byte
 answer = sendATcommand("AT+HTTPINIT", "OK", 20000); //enviar comando para inicio de servicio HTTP
 if (answer == 1){  //Si es exitosa la operación
  answer = sendATcommand("AT+HTTPPARA=\"CID\",1", "OK", 5000); // Entrar a la configuración
  if (answer == 1){  //Si es exitosa la operación
    answer = sendATcommand("AT+HTTPPARA=\"URL\",\"http://tracker2020.xyz/vercell.php\"", "OK", 5000);  //Enviar comando con link de la base de datos de contacto
    if (answer == 1){  //Si es exitosa la operación
      answer = sendATcommand("AT+HTTPACTION=0", "+HTTPACTION: 0,200,10", 10000); //Enviar un post del ur indicado
      if (answer == 1){  //Si es exitosa la operación
```

```
        Serial.println("Cell recibido"); //Mensaje de éxito en la operación
       }
        else{ //Si existe un error al acceder en el URL indicado
          errorURL(); //Función de mensaje de error en el acceso
        }
      }
      else{ //Si existe un error en el URL indicado
        errorCURL(); //Función de mensaje de error en el URL
      }
    }
    else{ //Si existe un error en acceder a la configuración
      errorCID(); //Función de mensaje de error en el acceso
    }
  }
  else{ //Si existe un error en acceder al iniciar servicio HTTP
    errorHTTP();  //Función de mensaje de error en el inicio del servicio
  }
  sendATcommand("AT+HTTPREAD", "", 2000); //Leer contenido de request HTTP
  String numerocell = serialSIM800.readString(); //Guardar la lectura
  /*sendATcommand("AT+HTTPTERM", "OK", 5000); //Finalizar servicio HTTP
  sendATcommand("AT+SAPBR=0,1", "OK", 2000);  //Apagar GPRS
  offGPRS();  //Función de mensaje de GPRS apagado*/
  //finCONECCION();
  numerocell.remove(0,29);  //Eliminar caracteres iniciales innecesarios
  numerocell.remove(9,6); //Eliminar caracteres finales innecesarios
  //Serial.println(numerocell);
  return numerocell; //Retornar número celular de contacto
  }

void finCONECCION(){
  sendATcommand("AT+HTTPTERM", "OK", 5000); //Finalizar servicio HTTP
  sendATcommand("AT+SAPBR=0,1", "OK", 5000);  //Apagar GPRS
  offGPRS();  //Función de mensaje de GPRS apagado
  }

/*FUNCIÓN DE ENVIO DE SMS Y LLAMADA AL CONTACTO*/
```

```
void alertaSMS_CALL(){  //Inicio de la función
  serialSIM800.listen();  //Atender serial SIM800
  String ph = recepHTTP();  //Usar el valor de retorno de la función recepHTTP()
  String stringAT ="";  //String que contendrá el comando para una llamada al contacto
  stringAT += "ATD+593";  //Agregar caracteres de inicio del comando
  stringAT += ph; //Agregar número de contacto
  stringAT += ";"; //Agregar un finalizar de comando
  int str_len = stringAT.length() + 1; //Longitud del char igual a la del String + 1
  char char_array[str_len]; //Crear un char de longitud str_len
  stringAT.toCharArray(char_array, str_len);  //Convertir String a char
  serialSIM800.println(stringAT);
  delay(20000);
  serialSIM800.println("ATH");
  delay(5000);
  serialSIM800.println("ATH");
  delay(5000);
  /*
  stringAT =""; //Vaciar nuevamente el Sring
  stringAT += "AT+CMGS=\"593";  //Inicio del comando para SMS
  stringAT += ph; //Añadir nuemro de contacto
  stringAT += "\""; //Finalizador de comando
  str_len = stringAT.length() + 1;  //Longitud del char igual a la del String + 1
  char_array[str_len];  //Crear un char de longitud str_len
  stringAT.toCharArray(char_array, str_len);  //Convertir String a char
  //Serial.println("AT SMS");
  //Serial.println(char_array);
  sendATcommand("AT+CMGF=1", "OK", 2000); //Entrar a modo de mensajes de texto
  sendATcommand(char_array, "OK", 2000);  //Enviar el número de celular a enviar el
mensaje
  serialSIM800.println("Se ha detectado una emergencia, siga la ubicación de su familiar
en el siguiente link: http://tracker2020.xyz/muestramap.html");  //Mensaje de alerta
enviado por SMS
  serialSIM800.write(0x1a); //Enviar un Ctrl+z para finalizar el mensaje
  Serial.println("Alertado");
  delay(10000);
  */
```

```
  Serial.println("Alertado");
  } //Fin de la función

void nada(){}  //Función vacia necesaria para la librería LowPower

/*FUNCIÓN PARA AHORRO DE ENERGÍA EN MÓDULO SIM800 Y ARDUINO*/
void modoSLEEP(){ //Inicio de función de ahorro
  Serial.println("Modo de ahorro,espera de alerta");  //Mensaje de inicio de función
  serialSIM800.listen(); //Atender serial SIM 800
  sendATcommand("AT+CSCLK=2", "OK", 1000);  //Enviar comando AT para colocar el módulo SIM800 EN "LOW POWER MODE"
  attachInterrupt(0, nada, LOW); //Declarar pin de interupción para despertar placa arduino
  LowPower.powerDown(SLEEP_FOREVER, ADC_OFF, BOD_OFF);  //Entrar en modo de ahorro de energía hasta presionar el pin 2
  detachInterrupt(0); //Placa arduino ha despertado, quitar la interuupción del pin 2
  } //Fin de la función de ahorro de energía

/*******************FUNCIONES AUXILIARES*************************/
/*FUNCIÓN DE AYUDA PARA ENVIO DE COMANDOS AT*/
int8_t sendATcommand(char* ATcommand, char* expected_answer1, unsigned int timeout){
uint8_t x=0, answer=0;
char response[100];
unsigned long previous;

memset(response, '\0', 100); // Initialize the string

delay(100);

while( serialSIM800.available() > 0) serialSIM800.read(); // Clean the input buffer

serialSIM800.println(ATcommand); // Send the AT command


x = 0;
```

```
previous = millis();

// this loop waits for the answer
do{
if(serialSIM800.available() != 0){
response[x] = serialSIM800.read();
x++;
// check if the desired answer is in the response of the module
if (strstr(response, expected_answer1) != NULL)
{
answer = 1;
}
}
// Waits for the asnwer with time out
}
while((answer == 0) && ((millis() - previous) < timeout));

return answer;
}

String getValue(String data, char separator, int index){
  int found = 0;
  int strIndex[] = {0, -1};
  int maxIndex = data.length()-1;

  for(int i=0; i<=maxIndex && found<=index; i++){
    if(data.charAt(i)==separator || i==maxIndex){
        found++;
        strIndex[0] = strIndex[1]+1;
        strIndex[1] = (i == maxIndex) ? i+1 : i;
    }
  }

  return found>index ? data.substring(strIndex[0], strIndex[1]) : "";
}
```

Anexo B2 *WRITE_DATA.PHP*

```
<?php //Etiqueta de inicio del script php
$servername = "localhost";  //Nombre de servidor
$dbname = "u917317866_tracker_data"; //Base de datos
$username = "u917317866_drygorbr"; //Nombre de usuario de la base de datos
$password = "drygorbr.2015"; //Contraseña de la base de datos


    $value1 = test_input($_GET["value1"]); //Variable value1 = al valor introducido en el link
    $value2 = test_input($_GET["value2"]); //Variable value2 = al valor introducido en el link

    $conn = new mysqli($servername, $username, $password, $dbname); //Inicio de la conexión a la base de datos
    if ($conn->connect_error) { //Condición si existe un error en la conexión
        die("Connection failed: " . $conn->connect_error); //Terminar la conexión
    } //Fin de la condición
    $sql = "INSERT INTO Sensor (Value_1, Value_2)
    VALUES ('" . $value1 . "', '" . $value2 . "')"; //Inserta el valor de "value1" y "value2" en la fila correspondiente de la tabla sensor

    if ($conn->query($sql) === TRUE) {  //Condición de éxito en el posteo de la información
        echo "Coordenadas subidas a la base de datos";  //Escribir mensaje de notificación
    }  //Fin del condicional
    else {  //Condición opuesta
        echo "Error: " . $sql . "<br>" . $conn->error;  //Escribir error y terminar la conexión
    }  //Fin de la condición opuesta

    $conn->close(); //Finalizar la conexión a la base de datos

function test_input($data) {  //Verificar la cadena de valores del link http
  $data = trim($data); //Eliminar cualquier espacio en blanco en la cadena de carácteres
```

```
    $data = stripslashes($data); //Quitar las barras por comillas escapadas
    $data = htmlspecialchars($data); //Cambiar caracteres especiales
    return $data;  //Regreso de la función con los caracteres corregidos
}  //Fin de la función de verificación
```

Anexo B3 *VERCELL.PHP*

```
<?php //Inicio de script PHP
/* DEFINICION DE CREDENCIALES DE LA BASE DE DATOS*/
$servername= "localhost"; //Variable que contiene el nombre del servidor
$username= "u917317866_drygorbr2015"; //Variable que contiene el nombre de usuario
$password= "drygorbr2015"; //Variable que contiene la contraseña de la base de datos
$dbname= "u917317866_numerocell"; //Variable que contiene el nombre de la base de datos

$conn=mysqli_connect($servername,$username,$password,$dbname); //Establecer la conexión a la base
$q = mysqli_query($conn, "SELECT numero FROM celular WHERE ID=(SELECT MAX(ID) FROM celular)"); //Selección de la tabla apuntando al valor máximo del parametro ID.

while($data = mysqli_fetch_array($q)) //Bucle para escribir los datos mientras exista alguno en el buffer
{
   echo "" .$data["numero"]."""; //Escribir el parámetro "número"
}  //Fin del bucle
//Final del script php
?>
```

Anexo B4 MUESTRAMAP.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
	<meta charset="UTF-8">
	<title>Emergencia Detectada</title>
	<style>
		#map {
			height: 100%;
		}
		/* Optional: Makes the sample page fill the window. */
		html, body {
			height: 90%;
			margin: 4%;
			padding: 0;
		}
	</style>
	<script>
	function initMap() {
		var mapOptions = {
			zoom: 5,
			center: new google.maps.LatLng(-0.210556, -78.488889),
			mapTypeId: 'roadmap'
		};
		var map = new google.maps.Map(document.getElementById('map'),
mapOptions);
		$.ajax({
	type: "GET",
	async: true,
	url: "http://tracker2020.xyz/mapa/xmlfile.php",
	dataType: "xml",
	success:
	function (xml) {
	   var places = xml.documentElement.getElementsByTagName("sensor");
	   for (var i = places.length-1; i < places.length; i++) {
```

```
            var lat = places[i].getAttribute('Value_1');
            var long = places[i].getAttribute('Value_2');
        var latLng = new google.maps.LatLng(lat, long);
        bounds  = new google.maps.LatLngBounds(); //R
            var marker = new google.maps.Marker({
                position:  latLng,
                map: map,
                label:places[i].name
            });

            loc = new google.maps.LatLng(marker.position.lat(), marker.position.lng());
    bounds.extend(loc);//

    map.fitBounds(bounds);       //# auto-zoom
    map.panToBounds(bounds);     //# auto-center
  }

        }
    });
        }
        </script>
        <script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
</head>
<body>
<div id="map"></div>
<script async defer

        src="https://maps.googleapis.com/maps/api/js?key=AIzaSyC8spZzQ8nNBR4dn
lpxoynH-OKgeaQ16VM&callback=initMap">
</script>
</body>
</html>
```

Anexo B5 *Script XMLFILE.PHP*

```
<?php //Etiqueta de inicio del script php
require("phpsqlajax_dbinfo.php"); //Llamado al archivo de credenciales de la base de datos

function parseToXML($htmlStr)//Función para reemplazo de caracteres a los comprendidos por un archivo XML
{
$xmlStr=str_replace('<','&lt;',$htmlStr); //Reemplazo del caracter "<"
$xmlStr=str_replace('>','&gt;',$xmlStr); //Reemplazo del caracter ">"
$xmlStr=str_replace('"','&quot;',$xmlStr); //Reemplazo del caracter "
$xmlStr=str_replace("'",'&#39;',$xmlStr); //Reemplazo del caracter '
$xmlStr=str_replace("&",'&amp;',$xmlStr); //Reemplazo del caracter "&"
return $xmlStr;  //Retorno de la función con la variable
} //Fin de la función de reemplazos

// Iniciar una conexión con la base de datos MySQL
$connection=new mysqli ("localhost", $username, $password, $database); //Cargar las credenciales de la base para inicio de conexión
if (!$connection) {  //Condicional de desconexión a la base de datos
  die('Not connected : ' . mysql_error()); //Cierre del vínculo en caso de error
} // Fin de la condición

// Selección de la tabla que contiene los datos
$sql = "SELECT * FROM Sensor"; //Elección de la tabla Sensor
$result = $connection->query($sql); //Variable de validez de la conexión
if (!$result) { //Condición en caso de que la operación sea inválida.
  die('Invalid query: ' . mysql_error()); //Cierre de la conexión con un mensaje de error.
} //Fin de la Condición

header("Content-type: text/xml"); //Encabezado del archivo XML

//Construcción del archivo XML
echo '<sensors>'; //Escribir la etiqueta "sensors"
while ($row = @mysqli_fetch_assoc($result)){ //Bucle para leer todas las filas de la tabla
```

```
  // Se escribe los datos de las filas existentes en la tabla
  echo '<sensor '; //Etiqueta de inicio de nueva fila
  echo 'ID="' . $row['ID'] . '" '; //Se escribe el valor de la variable "ID"
  echo 'Value_1="' . parseToXML($row['Value_1']) . '" '; //Se escribe el valor de la variable
"Value_1"
  echo 'Value_2="' . parseToXML($row['Value_2']) . '" '; //Se escribe el valor de la variable
"Value_2"
  echo '/>'; //Etiqueta de fin de fila

} //Final del bucle una vez acabadas las filas
echo '</sensors>'; //Escribir la etiqueta de fin del archivo xml
//Etiqueta de final del archivo php
?>
```

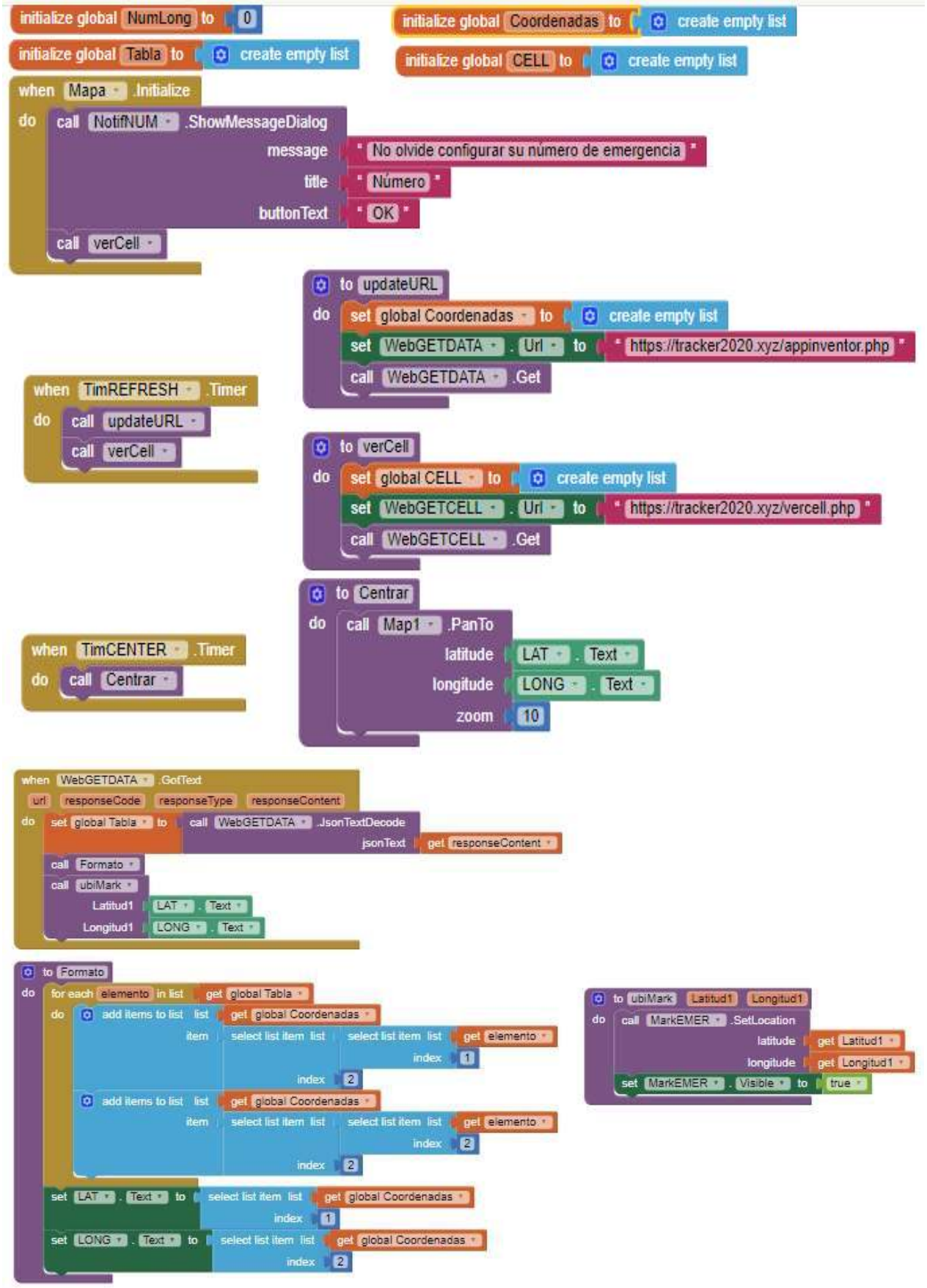
Anexo B6 *Script APPINVENTOR.PHP*

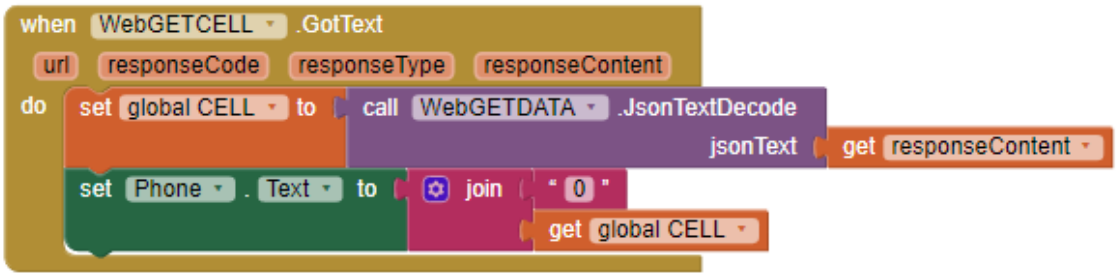
```
<?php

header('Content-Type: aplication/json'); //Define el tipo de contenido en formato JSON
$servername= "localhost";               //Define variable y asigna el nombre del servidor
$username= "u917317866_drygorbr";      //Define variable y asigna el nombre de usuario
$password= "drygorbr.2015";            //Define variable y asigna la contraseña de la base
$dbname= "u917317866_tracker_data";     //Define variable y asigna el nombre de la base de datos
$conn = mysqli_connect($servername,$username,$password,$dbname);       //Preprara la cenexión a la base con los datos respectivos
if (!$conn){            //Si no se entabla la conexión se delueve un error
   die ("connection failed: ". mysqli_connect_error());
}
$query =  mysqli_query($conn, "SELECT Value_1, Value_2 FROM Sensor WHERE ID=(SELECT MAX(ID) FROM Sensor)"); //Envía una consulta a la base solicitando los datos asociados al último registro en Value_1 y Value_2
while($data = mysqli_fetch_assoc($query))       //Inicia bucle de repetición
                                    //Devuleve un arreglo con los valores solicitados con los índices asociativos
{

   $json[] = $data;                         //Coloca el arreglo de datos en formato JSON
}
echo json_encode( $json);                       //Devuelve el JSON
?>
```

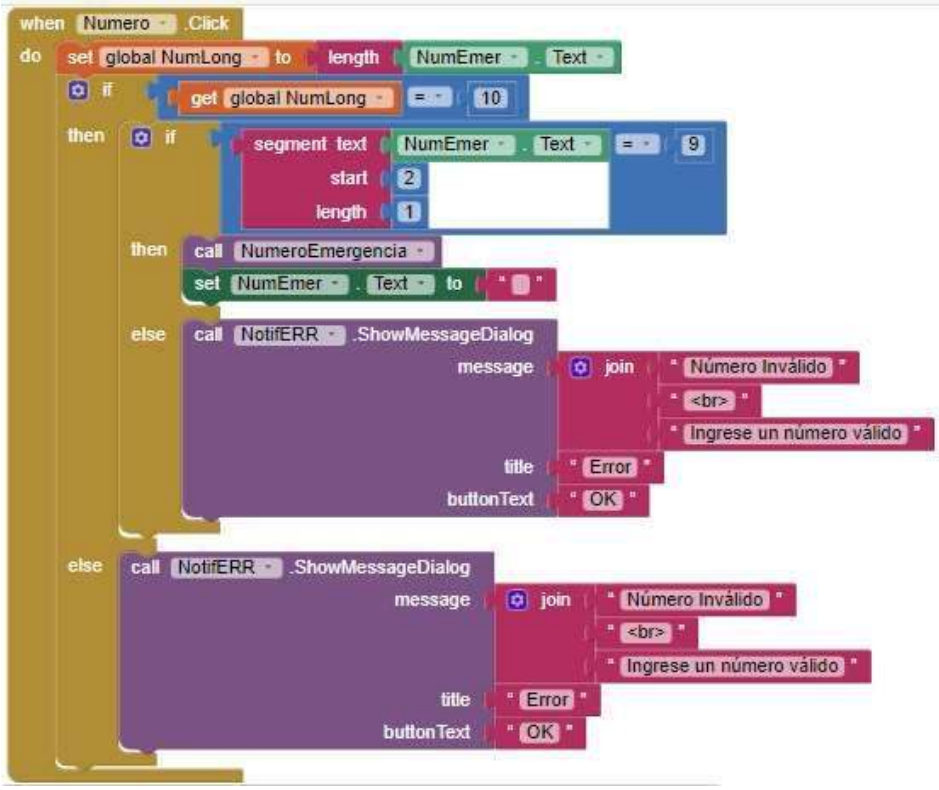
Anexo B7 *Script WRITE_CELL.PHP*

```
<?php //Etiqueta de inicio del script php
$servername = "localhost"; //Nombre de servidor
$dbname = "u917317866_numerocell"; //Base de datos
$username = "u917317866_drygorbr2015";//Nombre de usuario de la base de datos
$password = "drygorbr2015";  //Contraseña de la base de datos

    $numero = test_input($_GET["numero"]); //Variable numero = al valor introducido en el link
    $conn = new mysqli($servername, $username, $password, $dbname);//Inicio de la conexión a la base de datos
    if ($conn->connect_error) { //Condición si existe un error en la conexión
        die("Connection failed: " . $conn->connect_error);   //Terminar la conexión
    }  //Fin de la condición
    $sql = "INSERT INTO celular (numero)
    VALUES ('" . $numero . "')";  //Inserta el valor de "numero" en la fila numero de la tabla celular
    if ($conn->query($sql) === TRUE) { //Condición de éxito en el posteo de la información
        echo "NUMERO CELULAR SUBIDO A LA BASE"; //Escribir mensaje de notificación
    }   //Fin del condicional
    else {      //Condición opuesta
        echo "Error: " . $sql . "<br>" . $conn->error; //Escribir error y terminar la conexión
    } //Fin de la condición opuesta

    $conn->close(); //Finalizar la conexión a la base de datos
function test_input($data) { //Verificar la cadena de valores del link http
  $data = trim($data);   //Eliminar cualquier espacio en blanco en la cadena de carácteres
  $data = stripslashes($data); //Quitar las barras por comillas escapadas
  $data = htmlspecialchars($data); //Cambiar caracteres especiales
  return $data; //Regreso de la función con los caracteres corregidos
} //Fin de la función de verificación
```

Anexo B8 Bloques de programación Aplicación Android




when WebGETCELL .GotText
url responseCode responseType responseContent
do set global CELL to call WebGETDATA .JsonTextDecode
jsonText get responseContent
set Phone . Text to join " 0 "
get global CELL


when Numero .Click
do set global NumLong to length NumEmer . Text
if get global NumLong = 10
then if segment text NumEmer . Text = 9
start 2
length 1
then call NumeroEmergencia
set NumEmer . Text to " "
else call NotifERR .ShowMessageDialog
message join " Número Inválido "
"
 "
" Ingrese un número válido "
title " Error "
buttonText " OK "
else call NotifERR .ShowMessageDialog
message join " Número Inválido "
"
 "
" Ingrese un número válido "
title " Error "
buttonText " OK "


to NumeroEmergencia
do set WebPOST . Url to join " http://tracker2020.xyz/write_cell.php?numero= "
NumEmer . Text
call WebPOST .PostText
text " "