Universidad Acción Pro Educación y Cultura

Decanato de Ingeniería e Informática
Escuela de Ingeniería

Tesis de Grado para Optar por el Título de:
Ingeniero Electrónico Mención Comunicaciones

DISEÑO DE UN SISTEMA ELECTRÓNICO PARA LOCALIZAR PERSONAS CON ALZHEIMER

Sustentantes:

Br. Omielant Muñoz Núñez	2012-2043
Br. Andrickson R. Vicente Núñez	2012-2059
Br. Natividad J. Ortiz Pol	2013-0352

Asesor:

Ing. Emin Rivera

"Los conceptos expuestos en esta investigación son de la exclusiva responsabilidad de su(s) autor(es)."

Distrito Nacional
República Dominicana
Noviembre 2016

RESUMEN

Este trabajo de grado trata acerca del diseño de un sistema electrónico localizador, como una solución tecnológica al problema que muchos pacientes con Alzheimer atraviesan en determinadas circunstancias, siendo la más común de éstas, que se extravíen y sufran lesiones permanentes en algún momento de descuido que hayan tenido sus cuidadores.

Dado que el Alzheimer es una enfermedad que se continúa estudiando, se parte de documentación actualizada sobre esta patología y quienes la padecen para evaluar los trastornos psicológicos y conductuales que afectan a los pacientes con esta enfermedad. Esto ha permitido la elaboración de parámetros y consideraciones, que han de tomarse en cuenta para el diseño del sistema, además, se presentan los cimientos tecnológicos sobre localización geográfica que facilitan el análisis de tecnologías y dispositivos electrónicos aplicables en este ámbito. Posteriormente, se presentan los criterios para el diseño del sistema electrónico localizador, y el desarrollo de un prototipo como elemento de validación del mismo. A su vez, se describen los detalles de los componentes y módulos utilizados para su integración.

Entre los hallazgos más importantes de los resultados obtenidos, destaca el potencial de estas tecnologías para trabajar en conjunto con el Sistema Nacional de Atención a Emergencias y Seguridad (911), y la posibilidad de operar con elementos de muy bajo costo, como el envío de SMS y solicitudes de posición geográfica vía el servicio GPRS, por medio de convenios con alguna compañía telefónica.

DEDICATORIAS

Omielant Muñoz Núñez

A Dios
A mis padres, Emilio Muñoz y Ana Victoria Núñez
A mis hermanos, Donato e Hypatia.

Andrickson Vicente Núñez

A Dios
A mis padres, Altagracia Núñez y Carlos Vicente
A mis abuelos, Altagracia Mieses y Freddy Núñez
A mis hermanos Scarled Núñez y Edison Núñez
A los que me apoyaron siempre.

Natividad Ortiz Pol

A Dios
A mi madre, Virginia Victoria Pol Sanquintin
A mis hermanos, Violeta y Julio
A mi familia, especialmente, mi tía Quenia Pol
A mi novio, Miguel Damirón
A mi asesor y gran amigo, Emin Rivera
A mi compañero de trabajo y gran amigo, Pedro Garabitos.

AGRADECIMIENTOS

Omielant Muñoz Núñez

Agradezco a Dios, por no abandonarme, guiarme y fortalecerme cada día; por haberme permitido alcanzar esta meta y darme la posibilidad de tener nuevas. A mis padres, por sus enseñanzas, consejos y amor. A mis amigos Osias Féliz, Reynaldo Hernández, Agustín Asencio, Franlismer Martes, Ismael Volquez, Silvia Pimentel, Miguel Medina y Gustavo Sánchez, por la amistad, cariño y compresión que han compartido conmigo a lo largo de estos años. A mis compañeros, por ser un gran equipo y luchar juntos para lograr nuestros objetivos. Al profesor Emin Rivera, por ser nuestro guía y mentor, por los conocimientos transmitidos a lo largo de este trabajo de grado. Y, por último, a todos aquellos que estuvieron al pendiente y nos brindaron su ayuda cuando fue necesario.

Andrickson Vicente Núñez

Agradezco a Dios, por permitirme lograr esta meta y darme la fortaleza de superar cada dificultad. A mis padres Altagracia Núñez y Carlos Vicente por estar siempre presentes y darme la motivación de seguir adelante. A mis abuelos, por todo lo que han forjado en mí. A mis tíos/as Martín Núñez, Delio Núñez, Freddy Núñez, Olga Mieses, Bienvenido Núñez, por su apoyo incondicional. A mis amigos Miguel Medina, Emely Rodríguez, Gustavo Sánchez, Reynaldo Hernández, Agustín Asencio, Osias Féliz, Robert Pérez, Misael Martínez, por brindarme su granito de arena durante mi trayecto. A mis compañeras, por su dedicación y entrega. Al profesor Emin Rivera, por sus consejos y enseñanzas. A Luis De La Rosa, a Francisco Francisco, y a todos los que nos apoyaron.

Natividad Ortiz Pol

Agradezco a Dios y a la Virgen María quienes me otorgaron la fuerza necesaria para concluir la tesis. A mi madre, quien me apoyo física y emocionalmente en todo el proceso. A mi familia, quienes me dieron fortaleza. A mi novio, quien me brindó su apoyo y comprensión en esta etapa. A mi jefe, Pedro Garabito, que tuvo paciencia y compresión con mi tiempo. A nuestro asesor, Emin Rivera, quien mantuvo la motivación y el ánimo de todos. Agradezco a todos los que nos apoyaron y motivaron en esta importante etapa, a todos ellos le dedico este gran logro.

TABLA DE CONTENIDO

RESUMEN i

DEDICATORIAS ii

AGRADECIMIENTOS iii

TABLA DE CONTENIDO iv

LISTA DE TABLAS x

LISTA DE FIGURAS xi

INTRODUCCIÓN 16

1. COMPORTAMIENTO DEL PACIENTE CON ALZHEIMER Y TECNOLOGÍAS APLICADAS PARA SU CUIDADO 18

1.1. Trastornos psicológicos y del comportamiento 18

1.1.1. Concepto de Alzheimer y sus etapas 18

1.1.1.1. Primera etapa: Ausencia de daño cognitivo 19

1.1.1.2. Segunda etapa: Disminución cognitiva muy leve 19

1.1.1.3. Tercera etapa: Disminución cognitiva leve 19

1.1.1.4. Cuarta etapa: Disminución cognitiva moderada 20

1.1.1.5. Quinta etapa: Disminución cognitiva moderadamente severa 20

1.1.1.6. Sexta etapa: Disminución cognitiva severa 20

1.1.1.7. Séptima etapa: Disminución cognitiva muy severa 21

1.1.2. Consecuencias patológicas del Alzheimer 21

1.1.2.1. Trastornos cognoscitivos 22

1.1.2.2. Trastornos psicológicos 23

1.1.2.3. Trastornos conductuales 26

1.2. Consideraciones del comportamiento 29

1.2.1. Generalidades 29

1.2.2. Apreciación conductual 30

1.3. Localización geográfica 33

1.3.1. Caracterización de los sistemas de posicionamiento inalámbricos 34

1.3.2. Sistemas de posicionamiento y navegación GPS/ GSM 36

1.3.2.1. Posicionamiento con sistema GPS 36

1.3.2.2. Posicionamiento con sistema GSM 36

1.4. Módulo GPS 37

1.4.1. Fundamentos de la tecnología GPS 37

1.4.2. Componentes 38

1.4.3. Funcionamiento del GPS 39

1.4.4. Receptor GPS 40

1.5. Módulo GSM 42

1.5.1. Fundamentos de la tecnología GSM 42

1.5.2. Arquitectura 42

1.5.3. Módem GSM 45

1.5.4. Servicio SMS 46

1.5.5. Modalidades 46

2. DISEÑO DE SISTEMA ELECTRÓNICO LOCALIZADOR (SEL) 47

2.1. Diagramas generales del sistema 47

2.1.1. Hardware 47

2.1.2. Software 47

2.2. Integración del microcontrolador y módulos GSM-GPS 48

2.2.1. Módulos GSM y GPS 48

2.2.1.1. Módulo SIM808 49

2.2.1.2. Módulo SIM800 52

2.2.2. Microcontrolador ATMEGA328P 53

2.2.3. Arduino 54

2.3. Protocolo y comandos de comunicación 55

2.3.1. Trasmisor-Receptor Asíncrono Universal (UART) 55

2.3.2. Comandos AT 56

2.4. Programación del procesador de datos y de la pulsera electrónica 57

2.4.1. Módulo procesador de datos 58

2.4.1.1. Raspberry Pi 58

2.4.1.1.1. Disposición de pines 59

2.4.1.2. Lenguaje de programación: Python 59

2.4.1.2.1. Librerías 59

2.4.1.3. Programas modulares: Descripción de funcionamiento 60

2.4.1.3.1. Residentes 61

2.4.1.3.1.1. Alerta de estado del paciente 61

2.4.1.3.1.2. Análisis de coordenadas 63
2.4.1.3.1.3. Actualización de tiempo 65
2.4.1.3.1.4. Recepción de mensaje 66
2.4.1.3.2. De proceso por solicitud 67
2.4.1.3.2.1. Mensaje cuidador 67
2.4.1.3.2.2. Mensaje pulsera 68
2.4.1.3.2.3. Envío de mensaje 69
2.4.1.3.2.4. Realización de llamada 70
2.4.1.4. Base de datos 71
2.4.1.5. Interfaz de administrador 75
2.4.2. Pulsera electrónica 80
2.5. Interfaz interactiva del usuario 82
2.5.1. Sistema operativo: Android 82
2.5.2. Software: Android Studio 82
2.5.3. Lenguaje de programación: Java 83
2.5.4. Diseño de interfaz gráfica: Aplicación móvil 83
2.5.4.1. Opciones de usuario 84
2.5.4.2. Zonas seguras del paciente 85
2.5.4.3. Envío de mensaje al módulo procesador de datos 86
2.5.4.4. Mostrar ubicación geográfica 87
2.5.4.5. Recepción de mensajes del módulo procesador de datos 88
2.6. Prototipado del sistema 89
2.6.1. Prototipo de la pulsera electrónica 89

2.6.1.1. Programa en Arduino 90
2.6.2. Prototipo del módulo procesador de datos 91
2.6.2.1. Programas modulares: Líneas de código principales 92
2.6.2.1.1. Residentes 92
2.6.2.1.2. De proceso por solicitud 97
2.6.3. Prototipo de la aplicación móvil 100
3. VERIFICACIÓN Y CONTRASTE DEL DISEÑO 101
3.1. Resultados 101
3.1.1. Prueba funcional del sistema 101
3.1.1.1. Condiciones iniciales 101
3.1.1.1.1. Hardware 102
3.1.1.1.1.1. Módulo procesador de datos 102
3.1.1.1.1.2. Pulsera electrónica 103
3.1.1.1.1.3. Teléfono móvil del cuidador 106
3.1.1.1.2. Software 106
3.1.1.1.2.1. Módulo procesador de datos 106
3.1.1.1.2.2. Pulsera electrónica 108
3.1.1.1.2.3. Aplicación móvil 109
3.1.1.2. Información de pacientes y cuidadores 110
3.1.1.2.1. Registro de pulsera 111
3.1.1.2.2. Registro de paciente 112
3.1.1.2.3. Registro de cuidador 112
3.1.1.2.4. Asignación de zonas seguras 113

3.1.1.3. Funcionamiento de la pulsera electrónica 114
3.1.1.4. Funcionamiento del módulo procesador de datos 116
3.1.1.5. Funcionamiento de la aplicación móvil 120
3.2. Evaluación de hallazgos 123
CONCLUSIONES 125
RECOMENDACIONES 126
REFERENCIAS BIBLIOGRÁFICAS 128
GLOSARIO 135
ANEXOS 137

LISTA DE TABLAS

Tablas Página

Tabla 1.1. Funciones y características genéricas de un receptor GPS 41

Tabla 1.2. Características de hardware del módulo SIM808 49

Tabla 1.3. Características técnicas del chip SIM808 51

Tabla 1.4. Especificaciones de la batería y el cargador para el chip SIM808 51

Tabla 1.5. Características de hardware del módulo SIM800H 52

Tabla 1.6. Características y prestaciones técnicas del chip SIM800H 53

Tabla 1.7. Características técnicas del microcontrolador ATMEGA 328P 54

Tabla 1.8. Especificaciones técnicas de la placa Arduino UNO 55

Tabla 1.9. Comandos AT utilizados con frecuencia en el SEL 57

Tabla 1.10. Características técnicas de la Raspberry Pi 2 Modelo B V1.1 58

Anexos

Tabla A.1. Diferencias entre Alzheimer y demencia senil 138

Tabla A.2. Síntomas del Alzheimer y recomendaciones de prevención 138

Tabla B.1. Clasificación de los sistemas de posicionamiento basados en mediciones disponibles 141

Tabla B.2. Comparación entre diferentes sistemas de posicionamiento 142

Tabla C.1. Formato de los datos de la ubicación geográfica 145

LISTA DE FIGURAS

Figuras **Página**

Figura 1.1. Posicionamiento general de una red 34

Figura 1.2. Componentes del sistema GPS 39

Figura 1.3. Partes de un receptor GPS 40

Figura 1.4. Arquitectura de una red GSM 44

Figura 1.5. Partes de un módem GSM 45

Figura 2.1. Estructura física del SEL. 47

Figura 2.2. Programas del SEL. 47

Figura 2.3. Módulo GPS-GSM, SIM808. 49

Figura 2.4. Chip SIM808. 50

Figura 2.5. Módulo GSM, SIM800 (modelo H).. 52

Figura 2.6. Chip SIM800H.. 52

Figura 2.7. Raspberry Pi 2 Modelo B V1.1. 58

Figura 2.8. Programas modulares del módulo procesador de datos. 61

Figura 2.9. Flujo de procesos del programa “Alerta de estado del paciente”. 62

Figura 2.10. Flujo de procesos del programa “Análisis de coordenadas” 64

Figura 2.11. Flujo de procesos del programa “Actualización de tiempo” 65

Figura 2.12. Flujo de procesos del programa “Recepción de mensaje” 66

Figura 2.13. Flujo de procesos del programa “Mensaje cuidador” 67

Figura 2.14. Flujo de procesos del programa “Mensaje pulsera” 68

Figura 2.15. Flujo de procesos del programa “Envío de mensaje” 69

Figura 2.16. Flujo de procesos del programa “Realización de llamada” 70

Figura 2.17. Tablas de la base de datos del sistema 71

Figura 2.18. Tabla “Log” 72

Figura 2.19. Tabla “Dispositivos” 72

Figura 2.20. Tabla “Paciente” 73

Figura 2.21. Tabla “Cuidador” 73

Figura 2.22. Tabla “Usucuidador” 74

Figura 2.23. Tabla “Zona” 74

Figura 2.24. Tabla “Estado” 75

Figura 2.25. Interfaz de administrador del módulo procesador de datos 76

Figura 2.26. Pestaña “Datos de pacientes” de la base de datos 76

Figura 2.27. Pestaña “Datos de cuidadores” de la base de datos 77

Figura 2.28. Pestaña “Registro de ubicaciones” de la base de datos 78

Figura 2.29. Pestaña “Registro de pulseras” de la base de datos 78

Figura 2.30. Pestaña “Resultado de análisis de las ubicaciones” de la base de datos 79

Figura 2.31. Pestaña “Opciones” de la base de datos 79

Figura 2.32. Flujo de procesos del programa de la pulsera electrónica 81

Figura 2.33. Módulos de la aplicación móvil 83

Figura 2.34. Flujo de procesos del programa “Opciones de usuario” 84

Figura 2.35. Flujo de procesos del programa “Zonas seguras del paciente” 85

Figura 2.36. Flujo de procesos del programa “Envío de mensaje al módulo procesador de datos” 86

Figura 2.37. Flujo de procesos del programa “Mostrar ubicación geográfica” 87

Figura 2.38. Flujo de procesos del programa “Recepción de mensajes del módulo procesador de datos” 88

Figura 2.39. Diagrama en bloques de la pulsera electrónica 89

Figura 2.40. Diagrama en bloques del módulo procesador de datos 91

Figura 2.41. Diagrama en bloques de un teléfono móvil 100

Figura 3.1. Ensamblaje del módulo SIM800 102

Figura 3.2. Conexión Raspberry Pi – Módulo SIM800 103

Figura 3.3. Ensamblaje del módulo procesador de datos 103

Figura 3.4. Ensamblaje del módulo SIM808. a) Parte delantera. b) Parte trasera 104

Figura 3.5. Conexión del pin V_{io} al pin V_{bat} en el módulo SIM808 104

Figura 3.6. Pulsera electrónica: Conexiones Arduino – Módulo SIM808 105

Figura 3.7. Pulsera electrónica conectada vía cable USB a una computadora 105

Figura 3.8. Ejecución de la terminal de línea de comandos en Raspbian (Jessie) 106

Figura 3.9. Programas modulares ejecutados en su estado inicial 108

Figura 3.10. Pulsera electrónica: Inicialización del programa vista desde el “Monitor Serial” de Arduino 109

Figura 3.11. Interfaz de la aplicación móvil 110

Figura 3.12. Interfaz de administrador: Inicio de sesión del administrador del SEL 111

Figura 3.13. Interfaz de administrador: Ventana “Menú” 111

Figura 3.14. Interfaz de administrador: Ventana “Registro de Pulsera” 111

Figura 3.15. Interfaz de administrador: Ventana “Datos de Paciente”. 112

Figura 3.16. Interfaz de administrador: Ventana “Datos de Cuidador” 113

Figura 3.17. Interfaz de administrador: Ventana “Registro de Ubicaciones” 114

Figura 3.18. Pulsera electrónica: Búsqueda de arreglos satelitales 115

Figura 3.19. Pulsera electrónica: Arreglo satelital encontrado 115

Figura 3.20. Pulsera electrónica: Guardado de los datos geográficos y envío vía SMS al procesador de datos 116

Figura 3.21. Módulo procesador de datos: En espera de mensaje SMS 116

Figura 3.22. Módulo procesador de datos: Recibiendo SMS de la aplicación móvil y enviando respuesta de solicitud 117

Figura 3.23. Módulo procesador de datos: Recibiendo SMS de la pulsera electrónica 118

Figura 3.24. Módulo procesador de datos: Estado “Peligro” en la tabla de “Resultados de análisis de las ubicaciones” 118

Figura 3.25. Módulo procesador de datos: Estado “Seguro” en la tabla de “Resultados de análisis de las ubicaciones” 119

Figura 3.26. Base de datos con la información de los pacientes registrados en el SEL 119

Figura 3.27. Aplicación móvil. a) Solicitud de acceso a los mensajes de textos. b) Solicitud de acceso a la ubicación del GPS 120

Figura 3.28. Aplicación móvil. a) Seleccionar zona segura. b) Confirmar zona segura. c) Establecer diámetro de zona segura 121

Figura 3.29. Zona segura ingresada 121

Figura 3.30. Aplicación móvil. a) Barra de herramientas. b) Solicitud de la ubicación del paciente. c) Ubicación actualizada del paciente 122

Figura 3.31. Aplicación móvil. a) Paciente fuera de la zona segura. b) Estado de alerta en peligro. c) Respuesta del cuidador a la alerta enviada al procesador de datos 123

Anexos

Figura A.1. Comparación entre un cerebro sano y uno enfermo 139

Figura A.2. Tipos de demencia 140

Figura B.1. Niveles de calidad, cobertura y precisión en las tecnologías de posicionamiento geográfico 141

Figura C.1. Esquemático de la placa SIM808 utilizada para el prototipo de la pulsera electrónica 143

Figura C.2. Esquemático de la placa SIM800 utilizada en el prototipo del módulo procesador de datos. 144

Figura C.3. Disposición de pines de la Raspberry Pi 2 Modelo B V1.1 145

Figura F.1. Módulo procesador de datos: Tabla de la base de datos que muestra los resultados y análisis del sistema sobre el estado del paciente 183

Figura F.2. Módulo procesador de datos: Tabla de la base de datos con la información de cuidadores registrados 183

Figura F.3. Módulo procesador de datos: Tabla de la base de datos con las credenciales de los custodios 183

Figura F.4. Módulo procesador de datos: Tabla de la base de datos con la información de zonas seguras y coordenadas geográficas del paciente 183

Figura G.1. Artículo sobre estudios de la enfermedad de Alzheimer en dominicanos 184

INTRODUCCIÓN

El Alzheimer es una enfermedad que afecta el cerebro de las personas que la padecen, cuya manifestación puede ser progresiva y conforme evoluciona provoca en los individuos grandes cambios psicológicos y conductuales, siendo estos últimos la razón por la cual los enfermos en las primeras etapas tienden a deambular olvidando el motivo y hacia donde se dirigen. Actualmente, en la República Dominicana se tienen 120,000 casos detectados de personas con este padecimiento, una cifra que se pronostica con los años aumentará.

El objetivo principal que persigue este trabajo de grado es diseñar un sistema electrónico que permita conocer la ubicación de aquellos individuos que lo utilicen.

Para la elaboración de este trabajo se tomaron datos estadísticos acerca de la situación actual de la enfermedad de Alzheimer en República Dominicana y sobre cómo ha afectado a sus habitantes. Partiendo de esto, se identificó la problemática principal de los pacientes, que es básicamente la tendencia a deambular lo que muchas veces conlleva a daños físicos y/o emocionales a lo largo de sus recorridos. Por consiguiente, se limitó el alcance del diseño, dedicándolo para aquellos con fases tempranas y medias de la enfermedad. Por tanto, se identificaron las herramientas y tecnologías existentes en la actualidad que sirven de soporte a los individuos con Alzheimer y sus allegados.

También se evaluaron las limitaciones que implican los dispositivos y/o métodos respecto a la problemática presentada, y se seleccionaron los elementos necesarios para el diseño y elaboración de un sistema funcional capaz de mantener a los custodios y sus pacientes conectados.

Luego se analizaron las posibles estructuras, tanto de hardware como de software para el diseño del prototipo del sistema y un futuro diseño final, seleccionando los elementos necesarios para la creación del hardware en función de los parámetros más apropiados, compuesto en esencia por la integración de sistemas embebidos y plataformas móviles. Desde el punto de vista del software se determinó la creación de una base de datos, una interfaz de administrador y una aplicación móvil o de usuario para el manejo del sistema. Partiendo de las ideas anteriores, se plantearon los diagramas en bloques y diagramas de flujo con respecto al funcionamiento que caracterizan al sistema y sus componentes.

Los sistemas embebidos fueron programados en diferentes lenguajes de programación para el funcionamiento de las plataformas. El sistema de gestión utilizado para la base de datos fue SQLite, en conjunto con el diseño de la interfaz de administrador que fue elaborado en Python. Se realizaron pruebas de compatibilidad entre estas dos partes del sistema y por último se elaboró la aplicación de usuario utilizando el lenguaje de programación Java para dispositivos Android. Posteriormente, se probó la integración de cada elemento tanto de hardware como software en conjunto con la interfaz y se comprobó su funcionamiento e interacción.

1. COMPORTAMIENTO DEL PACIENTE CON ALZHEIMER Y TECNOLOGÍAS APLICADAS PARA SU CUIDADO

1.1. Trastornos psicológicos y del comportamiento

1.1.1. Concepto de Alzheimer y sus etapas.

El Alzheimer es una enfermedad degenerativa del cerebro, y la causa más común de demencia en las personas de edad avanzada, por lo que dicha manifestación se concibe como un síndrome normalmente crónico o progresivo que se caracteriza por un daño en la memoria, lenguaje, intelecto y otras habilidades como la orientación espacial, que afectan el modo de vida de las personas. Esto ocurre porque las células nerviosas en las zonas del cerebro que participan en las funciones cognitivas han sido afectadas, dañadas o destruidas.[1]

Las personas que sufren la enfermedad, tienen un pronóstico de vida de ocho años luego de que otras personas noten algunos síntomas. Sin embargo, la esperanza de vida se encuentra entre tres y veinte años, dependiendo de factores como la edad del individuo y las condiciones de salud (véase más información sobre el Alzheimer en el Apéndice A).[2]

El experto Barry Reisberg, M.D., director del Centro de Investigación de Demencia y Envejecimiento Silberstein de la Escuela de Medicina de la Universidad de Nueva York, define la Escala de Deterioro Global (GDS), donde se muestran las 7 etapas de la demencia

(secciones 1.1.1.1. a 1.1.1.7.), que permiten observar la progresión de los diversos cambios en las habilidades cognitivas de los pacientes, a medida que se desarrolla la enfermedad, además, presenta la etapa temprana, media y severa del Alzheimer.[3]

1.1.1.1. Primera etapa: Ausencia de daño cognitivo.

La persona se considera sana mentalmente, no experimenta dificultades para memorizar y no se presentan evidencias médicas sobre la enfermedad. En esta etapa se encuentran los individuos que no tienen demencia.

1.1.1.2. Segunda etapa: Disminución cognitiva muy leve.

La persona percibe algunos problemas de memoria, tales como el olvido de palabras comunes y conocidas, así como el lugar de las pertenencias de uso cotidiano. Estas situaciones no ponen en evidencia para los parientes cercanos y amigos las causas de la enfermedad, pero pueden ser los primeros indicios de envejecimiento o de Alzheimer.

1.1.1.3. Tercera etapa: Disminución cognitiva leve.

Empiezan a presentarse evidencias de fallas de memoria y atención en las actividades diarias. En este punto, las características de estos problemas pueden analizarse con más detalle por parte de un especialista. En algunos casos, estos síntomas pueden formar parte de un diagnóstico para la etapa temprana del Alzheimer y pueden ser:

- Menor capacidad de planificación y organización.
- Dificultad para pensar, retener información al leer, recordar una palabra o un nombre.
- Pérdida de objetos.

1.1.1.4. Cuarta etapa: Disminución cognitiva moderada.

Etapa Leve o Temprana del Alzheimer. En esta etapa es posible detectar ante una evaluación médica varias deficiencias notables, tales como:

- Falla en la memoria a corto plazo.
- Mayor dificultad para elaborar, planificar o realizar tareas complejas.
- Variaciones de humor ante desafíos mentales y sociales.
- Olvidar los acontecimientos y la secuencia de estos.

1.1.1.5. Quinta etapa: Disminución cognitiva moderadamente severa.

Etapa moderada o media del Alzheimer. La persona ha desarrollado dificultades para hacer uso de las habilidades mentales como el lenguaje, la percepción y la memoria, y es prudente que reciba algunas atenciones para realizar sus actividades diarias. Algunas de las características de esta etapa son:

- Incapacidad de recordar su propia casa e informaciones relevantes como su número de teléfono, la institución donde estudió, entre otros.
- Confundir los días de la semana y el lugar en el que está presente.
- Necesidad de ayuda para escoger la ropa adecuada para la ocasión.
- Aún se recuerdan detalles importantes sobre sí mismo y familiares.

1.1.1.6. Sexta etapa: Disminución cognitiva severa.

En este punto, continúan empeorando los problemas de memoria y pueden generarse algunos cambios significativos en la personalidad y en el comportamiento. Es requerido que el individuo tenga un cuidador y que le asista en todas sus tareas habituales.

Entre las consecuencias más comunes de esta etapa se encuentran:

- Pérdida de la conciencia sobre hechos recientes, el entorno y de experiencias.
- Experimentar una conducta compulsiva.
- Desarrollar una tendencia a deambular.
- Necesitar ayuda para hacer uso del baño.
- Cometer errores como colocarse incorrectamente la ropa y zapatos.

1.1.1.7. Séptima etapa: Disminución cognitiva muy severa.

Etapa severa o tardía del Alzheimer. Los pacientes pierden la capacidad de reconocer el entorno, se deteriora su capacidad de hablar y, con el tiempo, hasta las habilidades motoras. En estas circunstancias, el individuo necesita ayuda inclusive con las actividades más básicas, tales como comer, asearse, sentarse adecuadamente, sonreír, entre otras.

1.1.2. Consecuencias patológicas del Alzheimer.

Debido al deterioro que esta enfermedad produce, puede generarse una diversidad de trastornos que afectan psicológica y conductualmente al paciente, causando cambios conflictivos en su diario vivir, lo que hace necesaria la compañía de un cuidador.

Estos cambios pueden presentarse en cualquier etapa del Alzheimer.[4] Además, los trastornos no dependerán, necesariamente, de qué tan avanzada esté la enfermedad en el paciente, sino de los síntomas, los cuales, pueden variar durante el desarrollo de la enfermedad.[5]

Por otra parte, las alteraciones psicológicas y conductuales, van ligadas al estado del paciente y a otros tipos de enfermedad, además de los factores ambientales del entorno en que éste se encuentre. De ahí que los síntomas difieren en cada caso de Alzheimer.[6]

1.1.2.1. Trastornos cognoscitivos.

Estas alteraciones afectan a las funciones del cerebro vinculadas con los procesos de atención, la memoria, el lenguaje las habilidades visuales-espaciales y la función motora.[7]

- **Amnesia**

Es un trastorno que afecta a la memoria y sus características principales son la incapacidad de recordar información.[8] Puede ser de dos formas, orgánica o funcional. La amnesia orgánica se debe a alguna patología como el Alzheimer, que puede deteriorar el cerebro, mientras que la amnesia funcional, es debida a un mecanismo psicológico.[9]

Esta alteración es considerada uno de los síntomas comunes en la etapa temprana del Alzheimer, donde se puede denotar cómo la persona tiene mucha dificultad para recordar algo aunque se le brinden pistas.[10] Al principio, la memoria a corto plazo es eventualmente, más afectada que la memoria a largo plazo, dado que esta última sufre un daño más lento durante los inicios de la enfermedad.[11]

- **Agnosia**

Es la incapacidad de reconocer e identificar a las personas, lugares y objetos a pesar de que las funciones sensoriales se encuentran en correcto estado. Este padecimiento es debido a la interrupción de las señales o estímulos que llegan al cerebro.[12]

1.1.2.2. Trastornos psicológicos.

Este tipo de trastorno afecta al estado mental de los individuos, y se presenta como anormalidades en la capacidad de razonamiento que dificultan la correcta interpretación de la realidad y la adaptación en el entorno.[13]

Las alteraciones pueden ocurrir con frecuencia y desarrollar diversos patrones psicológicos y conductuales, afectando en esencia a las habilidades cognitivas del individuo, las cuales, van desde delirios hasta estados de ansiedad y agresividad.[14] Esto implica que la enfermedad no representa únicamente un problema de daño progresivo a la memoria y de capacidades mentales sino también una situación difícil para sus allegados.

- **Delirio**

Es una patología propia de las enfermedades de demencia que consiste en una idea o creencia incorrecta, generada a partir de la inferencia errónea de la realidad donde la información del pensamiento carece de lógica y coherencia. También es considerada uno de los trastornos más comunes en las personas con Alzheimer que se encuentran en las etapas media y severa.[15]

Entre los delirios más frecuentes en pacientes con demencia están:

- Sentir que está siendo observado o vigilado.
- Creer que su hijo es una persona desconocida o que su pareja le es infiel.
- Estar convencido de que sus parientes les abandonarán.

- **Alucinación**

Es una percepción que se presenta sin ningún estímulo externo a uno o varios sentidos (vista, oído, olfato, gusto o tacto) dando la impresión de que algo es real. Este padecimiento, no es tan usual como los delirios en los individuos afectados y son más comunes las alucinaciones visuales y auditivas.[16]

- **Ilusión**

Es una distorsión de la percepción sensorial, debido a estímulos externos que provoca la interpretación incorrecta de la realidad. Como uno de los trastornos más comunes en los pacientes de Alzheimer, afecta también las habilidades lógicas y de razonamiento, además de producir anomalías en la visión o en la audición. Un ejemplo frecuente de ilusión, corresponde con la percepción de un objeto pequeño que realmente es grande.[17]

- **Depresión**

Es un síndrome caracterizado por un conjunto de síntomas tales como tristeza patológica[18], vacío existencial, falta de motivación y un estado de ánimo muy bajo que puede ser reversible. En las primeras fases de la demencia, se denota con frecuencia una tristeza irracional mientras que en la etapa media se observan reacciones repentinas sobre emociones variables y desproporcionadas tales como risas impropias y cambios de humor. Este trastorno mental, genera resultados negativos en la persona, como insomnio y falta de interés a causa del Alzheimer.[19]

Los pacientes, presentan una remota posibilidad de intento de suicidio dado que para llevar a cabo este objetivo es necesario planificar, y puede resultar complicado dadas las características de la enfermedad, pero éstos puede provocarse lesiones.[20]

- **Ansiedad**

Es una respuesta frente a un peligro o miedo, caracterizada por generar inquietud, sentimientos desagradables y cambios físicos-tensionales. De forma natural, es una reacción que sirve para alertar sobre una amenaza permitiendo al individuo afectado tomar decisiones en torno al problema.[21]

En los pacientes de Alzheimer puede producirse un aumento innecesario de ansiedad, provocando cambios negativos en los patrones psicológicos y conductuales de la persona.[22]

Algunas de las circunstancias en las que se denota esta patología son:

- Estar consciente de la pérdida de sus capacidades mentales.
- Frustración frente a la dificultad de no comprender ni responder adecuadamente.
- Incapacidad de encontrar cualquier objeto.

- **Agresividad**

Es un estado emocional que produce una tendencia a actuar de forma violenta, manifestando con intensidad variable reacciones de ataque físico y/o verbal, tales como gritos, lenguaje malsonante, emisión de ruidos extraños y cambios de temperamento. Asimismo, los pacientes pueden sentirse amenazados o en peligro en lugares que una persona sana considera seguros.[23]

Esta reacción se produce como respuesta a diversas situaciones tales como:

- Cansancio.
- Incapacidad de comprender lo que sucede o expresar emociones o sentimientos.
- Expresar la frustración al tener que recibir ayuda para hacer las cosas.
- Respuesta a la ansiedad o angustia.
- Respuesta ante un delirio o una alucinación.

1.1.2.3. Trastornos conductuales.

Estas alteraciones tienen repercusión en los hábitos y costumbres del paciente, provocando que el enfermo tenga cambios en sus patrones de conducta. Estos pueden ser consecuencia de otros tipos de trastornos o enfermedades distintas al Alzheimer.[24]

- **Deambulación errática**

Es una práctica común entre los individuos que padecen algún tipo de demencia en cualquiera de sus etapas. Consiste en caminar sin algún rumbo aparente, siendo capaces de realizar esta acción por horas sin manifestar alguna expresión de cansancio o fatiga.[25]

Algunas de sus causas pueden ser:

- La incapacidad del enfermo de orientarse e identificar su entorno acostumbrado. Las inseguridades que esto provoca hacen que el individuo se muestre ansioso y agitado, motivándolo a deambular.
- La falta de distracción y la poca atención del custodio. Caminar sin cesar es una forma de llamar la atención y a la vez encontrar entretenimiento.

- **Trastornos del sueño**

Estas alteraciones se caracterizan por los cambios en los hábitos de sueño de la persona en cuestión. La intensidad de este trastorno variará según la rutina del individuo, si éste duerme todo el día, esto afectará su sueño en la noche, por el contrario, si tiene un día relativamente activo, es muy posible que esté cansado y pueda conciliar el sueño con facilidad. También dependerá del estado del paciente o la fase que tenga la enfermedad.[26]

Consecuentemente, una parte significativa de los individuos tiende a desorientarse y perderse en sus hogares. Asimismo, éstos pueden levantarse y vestirse con el objetivo de ir a realizar alguna actividad que quizá antes de enfermarse hacían de forma rutinaria (trabajar, ir de compras, pagar deudas o cocinar). No obstante, hay pacientes que pueden dormir con normalidad pero despiertan a muy tempranas horas del día.[27]

- **Alteraciones en la alimentación**

Estas consisten en cambios en los hábitos alimenticios, yendo desde demostrar mucho apetito hasta ser indiferente a la comida.[28]

Este trastorno puede ser el resultado de algún problema bucal (problemas en los dientes, dentaduras postizas, sensibilidad, úlceras), pérdida del gusto o el olfato y también, una consecuencia de otras enfermedades o trastornos que están asociadas al Alzheimer como son, la depresión y la apatía, provocando que el paciente no quiera ingerir alimentos o que los consuma de manera compulsiva.[29]

- **Desinhibición y lenguaje soez**

Este trastorno conductual está relacionado con la actividad sexual del paciente y la capacidad crítica. El individuo con Alzheimer u otro tipo de demencia, es incapaz de evaluar el entorno en que se encuentra y es por esto que en ocasiones, puede desnudarse porque no se siente cómodo o puede orinar/defecar en cualquier lugar sin notar que no están en el baño. En este sentido, el paciente al no recordar o tomar en cuenta las normas de la sociedad en general, tiende a utilizar lenguaje inadecuado.[30]

- **Síndrome de *Klüver-Bucy***

Es una patología que consiste en la necesidad del individuo de llevarse objetos a la boca desconociendo la diferencia entre lo comestible y lo nocivo. Esta provoca en los pacientes, agnosias visual, táctil y auditiva; las cuales no permiten que las personas reconozcan las cosas con la vista, tacto y oído y a su vez, una metamorfosis[31] que suscita un impulso por conocer el entorno y sorprenderse ante cualquier estímulo visual.[32]

- **Reacciones catastróficas**

Son ataques emocionales relacionados a un estímulo interno o externo al que el paciente se somete, (ya sea un delirio o una alucinación, algún problema, limitación o malestar, alguien desconocido, ruidos, situaciones nuevas o por el hecho de demandarle que realice alguna acción que no reconoce u olvidó como hacer) provocando en éste, reacciones desproporcionadas (enfado, agitación, angustia o violencia) de acuerdo a sus causas.[33]

1.2. Consideraciones del comportamiento

1.2.1. Generalidades.

A pesar de que el Alzheimer es una enfermedad con diversas etapas compuestas por un grupo sintomático particular, cada paciente es diferente al otro y sin importar que se encuentren en una misma fase del Alzheimer o no, estos presentan distintos comportamientos. En consecuencia, cada cuidador y familia del individuo debe tomar en cuenta ciertos aspectos del entorno en donde el paciente estará habitando.

- **Hogar**

Es necesario que el hogar del paciente esté adecuado a las necesidades que éste manifiesta de forma progresiva en su enfermedad, evaluando y aplicando las medidas de seguridad necesarias con el fin de mantener una calidad de vida apropiada para el enfermo.[34] En este sentido, los ajustes que deben realizarse son:

- Colocar cerrojos de protección en las ventanas y puertas con acceso al exterior.
- Remover los cerrojos de los cuartos de baños y aposentos (esto evita accidentes).
- Implementar seguros para infantes en cada lugar donde almacene productos químicos como los de limpieza.
- Almacenar cualquier objeto que represente un riesgo para el paciente.
- Procurar que el interior y exterior del hogar estén correctamente iluminados.
- Mantener control sobre la cocina y desperfectos que puedan ser inseguros para el enfermo.

- **Vestir**

A medida que la enfermedad evoluciona, las tareas más simples que el paciente podía realizar sin ayuda de un tercero, como hacer sus necesidades, bañarse y vestirse, se convierten en situaciones difíciles y estresantes para ambos.[35] Por tanto, los custodios deben emplear métodos como:

- Crear una costumbre con el paciente para que este se vista diariamente a la misma hora.
- Permitir y motivar al enfermo a que se vista por sí mismo sin apresurarlo.
- Escoger una variedad reducida de ropa.
- Instruir al paciente en cómo debe vestirse prenda a prenda, un paso a la vez.
- Seleccionar prendas que sean simples de utilizar y puedan conservarse higiénicas.

1.2.2. Apreciación conductual.

Uno de los grandes retos a los que se enfrentan los custodios de una persona con Alzheimer es el tener que lidiar con el nuevo comportamiento que el paciente desarrolla como parte de la enfermedad o como consecuencia de algún padecimiento previo. Por consiguiente, los cuidadores deben adoptar diferentes métodos para proteger tanto al paciente como a ellos mismos de los efectos que pueden provocar ciertas conductas.

- **Delirio y alucinaciones**

El enfermo con Alzheimer puede manifestar estas alteraciones en cualquiera de las etapas de la enfermedad y a pesar de ser trastornos psicológicos, éstas se demuestran con comportamientos ilógicos o comentarios fuera de lugar.[36]

En consecuencia, los custodios están en la responsabilidad de:

- Prescindir de discusiones con el paciente y consolarlos cuando sea necesario.
- Entretener al individuo con un tópico distinto o alguna actividad, por ejemplo, dándole un pequeño recorrido.
- Evitar que el enfermo vea programas de televisión que incluyan violencia o que puedan alterar su estado, ya que algunos individuos no son capaces de distinguir si lo que ven es real o no.
- Retirar cualquier objeto que el paciente pueda utilizar para afectar a un tercero o a sí mismo.
- Explicar al doctor el comportamiento que presenta el enfermo y compartir su historial médico, debido a que pueden ser resultado de otra enfermedad o algún medicamento que haya estado ingiriendo.

- **Trastornos del sueño**

Estos causan alteraciones en la conducta del paciente, ya sea inquietándolo, agitándolo o provocándole mal humor.[37] Es por esto que se recurre a:

- Motivar al paciente a ejercitarse durante el día para que esté cansado cuando llegue la noche. Éste debe tomar una serie de descansos durante cada rutina.
- Realizar un horario para hacer actividades en el día que exijan algún esfuerzo físico.
- Intentar eliminar las fuentes de ruido e implementar lámparas de mesa en su aposento en caso de que el enfermo tema a la oscuridad.
- Crear una rutina de sueño para el paciente como resultado de programar una hora específica para ir a dormir.
- Evitar que el enfermo ingiera alimentos que contengan cafeína.

- **Ansiedad**

El paciente ansioso, en muchas de las ocasiones, no es capaz de informar al custodio lo que le sucede y qué lo está causando. Esta puede presentarse con diversas intensidades, que van desde sentimientos del enfermo hasta episodios de agitación, que pueden ser resultado de otros trastornos tales como reacciones catastróficas y alteraciones del sueño.[38] Por tal razón, el cuidador debe:

- Prudentemente, tratar de conocer las causas de sus preocupaciones o ansiedad.
- Evitar realizar alguna acción que le cause ansiedad.
- Permanecer en calma siempre que interactúe con el paciente.
- Compartir con el doctor cualquier cambio en el humor del enfermo.

- **Deambular**

Es muy común en los individuos con Alzheimer, siendo uno de los trastornos más perjudiciales para el paciente debido a los daños irreparables que este puede causarle.[39] Por esto los cuidadores tienen la necesidad de aplicar medidas como:

- Cerciorarse de que el enfermo tenga una identificación consigo.
- Inscribir al paciente en algún programa de retorno seguro de la nación, si existe. Con éste, si el individuo se extravía y tiene dificultades para comunicarse, alertarán sobre la condición del enfermo.
- Dar a conocer a los allegados y a las autoridades pertinentes que el paciente suele deambular.
- Tener fotografías recientes del enfermo para proveer a la policía en caso de extravío.
- Asegurar las puertas y colocar sistemas de alarma, para que cuando sean abiertas el custodio pueda notarlo.

1.3. Localización geográfica

A menudo, las personas que cuidan a los pacientes con Alzheimer suelen tener algunas dificultades para ejercer la función de custodio sin que se vea afectada su calidad de vida, dado que no siempre permanecen junto al paciente cuando van a realizar sus actividades personales y otras diligencias. Esto es debido a las implicaciones propias de la enfermedad, las cuales afectan al individuo y con el tiempo resultan problemáticas y preocupantes ante la ausencia del cuidador.

De esta manera, surge la necesidad de conocer periódicamente la ubicación del paciente utilizando tecnologías modernas que implementen técnicas para determinar la ubicación de objetos de uso cotidiano tales como prendas de vestir, dispositivos móviles, entre otros.

En este sentido, el posicionamiento o capacidad de determinar una posición, ha significado parte importante de los servicios implicados en áreas de la salud, que van desde conocer la ubicación física de ambulancias o vehículos especiales hasta casos más específicos como delimitar la ubicación de pacientes con demencia. Por este motivo, se implementan tecnologías que tienen como característica principal la capacidad de transmitir información de forma inalámbrica y efectiva que mejoran la calidad de vida de las personas. Esto a su vez, supone que muchos sistemas electrónicos se basen en el concepto de localización, el cual, se define como el proceso de identificar con cierta precisión, cualquier punto de la superficie terrestre mediante el uso de coordenadas geográficas. El mismo emplea un sistema de numeración posicional con dos valores angulares denominados latitud y longitud.

La tecnología de localización más popular se conoce como el Sistema de Posicionamiento Global, por sus siglas en inglés, GPS (Global Positioning System), que consiste en un sistema de posicionamiento por satélite y receptores en tierra. Su uso permite que las personas puedan determinar su ubicación geográfica (en el Apéndice B, Figura 1, se muestran varios parámetros de las tecnologías de posicionamiento geográfico). Entre las aplicaciones de posicionamiento se encuentran los sistemas inteligentes de transporte, que permitirían la gestión de accidentes, asistencia en las carreteras y el seguimiento de vehículos. También es considerada la seguridad como un objetivo clave para la localización de civiles o para ayudar a pacientes mediante los sistemas del 911 utilizado en la mayoría de los países y 112 en Europa.[40]

1.3.1. Caracterización de los sistemas de posicionamiento inalámbricos.

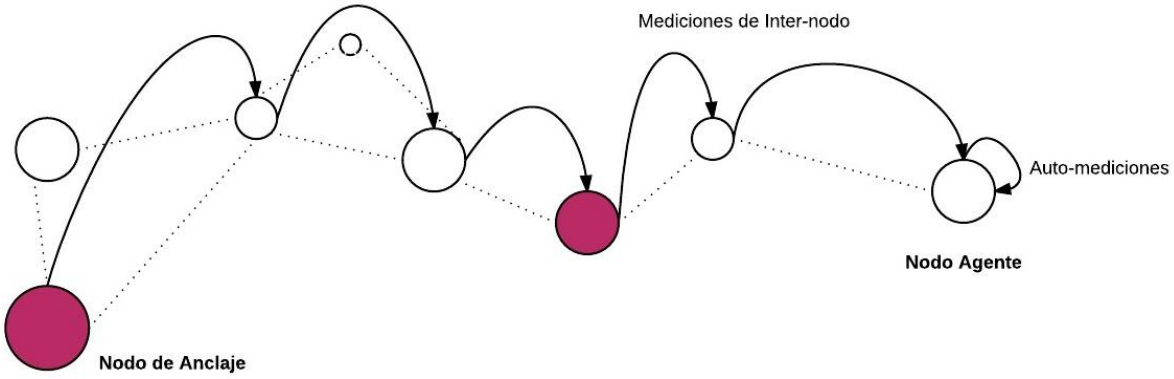


Figura 1.1. Posicionamiento general de una red.[41]

En principio, la localización de posición de forma inalámbrica implica que un terminal "activo" esté presente en el sistema, cuya posición tiene que ser obtenida. Esta situación difiere de la radiolocalización, en que esta última busca determinar la posición de un objeto distante considerado "pasivo" que no participa en el proceso de localización.[42]

[40] En la República Dominicana el número telefónico 9-1-1 corresponde al Sistema Nacional de Atención a Emergencias y Seguridad (911). Fuente: https://911.gob.do/

Un ejemplo de esto son los radares. Es debido a esto que la radiolocalización se relaciona con frecuencia con sistemas militares o de seguridad, mientras que la localización de un objeto de posición "activo", implica la participación del mismo para determinar su propia posición mediante el intercambio inalámbrico de información con algunas estaciones que sirvan de referencia. En estos sistemas la información sobre la posición es utilizada por el propio objeto "activo" y puede ser enviada a alguna estación de control que regule las funciones del mismo.[43]

En lo que respecta a los términos localización y navegación, existe una diferencia en este contexto. La navegación está orientada a recoger, registrar y planificar información geográfica sobre un vehículo y realizar un seguimiento de su trayectoria a partir de una posición inicial. En este proceso dicha trayectoria consiste en un algoritmo que determina una serie de posiciones estimadas independientes. Mientras que la localización consiste en determinar una ubicación geográfica.[44]

Los sistemas de posicionamiento inalámbricos se caracterizan por poseer una estructura con nodos a los que se les llama "nodos de anclaje" con ubicaciones fijas y conocidas. Estas coordenadas sirven de referencia a uno o más nodos móviles que tienen ubicaciones variables, a los que se conoce como "nodo agente".[45] La combinación de estos nodos define el posicionamiento en una red, como se muestra en la Figura 1.1, y se produce normalmente en dos pasos:

- Primero, las mediciones específicas se ejecutan entre los nodos.
- Segundo, las mediciones obtenidas son procesadas para determinar la posición de los nodos móviles o nodos agentes.

1.3.2. Sistemas de posicionamiento y navegación GPS/ GSM.

1.3.2.1. Posicionamiento con sistema GPS.

Para que en un proyecto sea implementado el sistema GPS es necesario comprender el funcionamiento del mismo. Actualmente, los receptores GPS están diseñados para ser muy fáciles de utilizar. Sin embargo, un conocimiento básico de las partes del sistema y aprender sobre la interacción entre éstas, permite que los diseñadores puedan trabajar con módulos receptores GPS de acuerdo a sus necesidades (véase el Apéndice B, Tabla 1 para conocer acerca de las medidas en sistemas como GPS).

El sistema fue originado con el objetivo de satisfacer las necesidades tácticas militares para la navegación de barcos y aviones, pero gracias a los avances en la integración de circuitos, los módulos GPS resultan muy económicos en una gamma muy variada de aplicaciones. Esta tecnología se caracteriza por el uso de técnicas que ofrecen una gran precisión respecto a otros sistemas de posicionamiento que puede ir desde unos pocos metros a centímetros (en el Apéndice B, Tabla 2, se comparan varios sistemas de posicionamiento).[46] Actualmente se puede encontrar en vehículos grandes y pequeños, equipos médicos y equipos de construcción, contribuyendo a la seguridad de las personas.

1.3.2.2. Posicionamiento con sistema GSM.

El Sistema Global para las Comunicaciones Móviles, por sus siglas en inglés, GSM (Global System for Mobile communications), realiza sus operaciones en las redes celulares, las cuales se basan en determinadas estaciones base (BSS), que poseen cada una, un radio de cobertura que puede abordar varias decenas de kilómetros. Es la tecnología más extendida en redes celulares basada en TDOA[47] ampliamente utilizada en todos los países desarrollados.[48]

La localización GSM analiza la diferencia de tiempo entre las señales transmitidas por las estaciones base que estén visibles, que son necesarias para determinar la posición de un móvil con el método de triangulación[49]. La ubicación final puede ser definida con una precisión de 50 a 500 metros. Esta precisión dependerá de los parámetros del ambiente que se estén considerando y la topología de la red.[50]

1.4. Módulo GPS

1.4.1. Fundamentos de la tecnología GPS.

El Sistema de Posicionamiento Global (GPS), por sus siglas en inglés, Global Positioning System, es un sistema que permite conocer la ubicación geográfica de una persona o un objeto en la Tierra. De esta forma, proporciona servicios a los usuarios tales como posicionamiento y navegación.[51]

El nombre oficial del sistema es NAVSTAR GPS (NAVigation Satellite Timing And Ranging Global Positioning System) y pertenece a los Estados Unidos. El mismo es administrado y operado por la Fuerza Aérea de EE.UU. en Los Ángeles, California, para proporcionar información de navegación de alta precisión a las fuerzas militares. No obstante, un gran porcentaje de los usuarios de GPS en todo el mundo son civiles por lo que el número de productos comerciales cada vez es mayor.[52]

Históricamente, desde el primer lanzamiento en 1978, ha habido cuatro generaciones de satélites GPS:[53]

Los Satélites del Bloque I (1978 y 1985) sirvieron de prueba para los principios del sistema, y verificar lo aprendido de los primeros 11 satélites en los bloques posteriores.

Los Satélites del Bloque I1 y IIA entre 1989 y 1997, formaron parte de la primera constelación en pleno funcionamiento de los cuales, varios aún se encuentran operando.

Los Satélites del Bloque IIR incorporados entre 1997 y 2007, se utilizaron para reponer satélites del Bloque II / IIA que llegaron al final de su vida útil. Los satélites del Bloque IIR-M colocados entre 2005 y 2008 efectuaron mejores técnicas.

Los Satélites del Bloque IIF entre 2010 y 2012, fueron los satélites de cuarta generación y son utilizados para operaciones y mantenimiento de reposición.

Los Satélites del Bloque I11 se pretende que sean los satélites de quinta generación, con capacidades mucho más modernas de las de los satélites Bloque IIF.

1.4.2. Componentes.

El sistema GPS está constituido por tres partes (como se muestra en la Figura 1.2):[54]

- ***Segmento Espacial.*** Es una constelación de 24 satélites en la que cada uno transmite señales que permiten calcular la posición y el tiempo de cada satélite del sistema GPS.
- ***Segmento de Control.*** Es un conjunto de estaciones que se encargan de dar seguimiento en toda la superficie terrestre con el propósito de asegurar la posición de los satélites en las órbitas adecuadas.
- ***Segmento del Usuario***. Consiste en el módulo receptor del GPS que recibe las señales de los satélites del sistema y las procesa para determinar la posición tridimensional y la hora precisa.

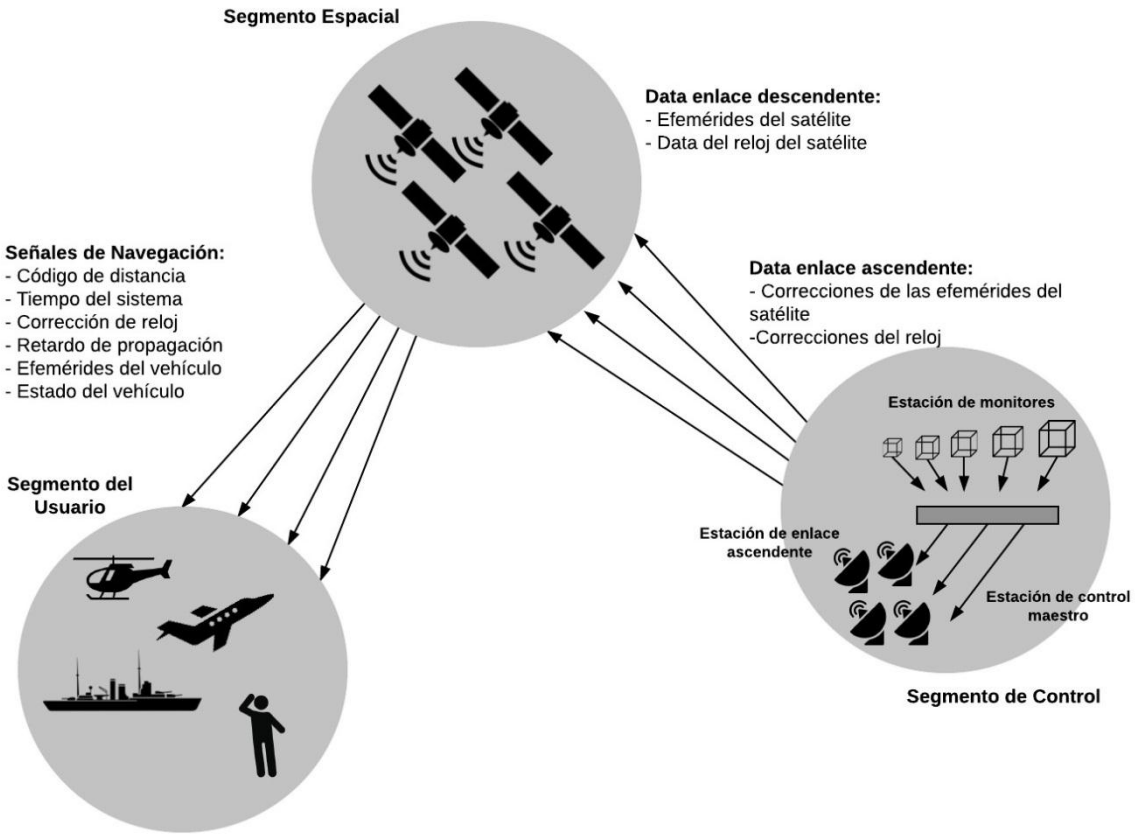


Figura 1.2. Componentes del sistema GPS.[55]

1.4.3. Funcionamiento del GPS.

A pesar de la complejidad que puede significar el desarrollo de un sistema GPS, el principio de funcionamiento consiste en un criterio elemental. Se basa en la ecuación de la velocidad, igual al cociente de la distancia por el tiempo. Esto nos dice que conociendo que tan rápido viaja algo y el tiempo que se toma se puede calcular la distancia recorrida.[56]

En el caso del sistema GPS, lo que realiza el viaje es una onda de radio que se propaga a la velocidad de la luz. Este proceso es llamado oscilación (*ranging*), con el que se transmiten múltiples señales de radio de varios satélites con el uso de la triangulación como concepto geométrico para determinar la posición exacta de un usuario.[57]

Para calcular distancias, el sistema GPS guarda la información exacta de la señal de radio que se emite en el momento de su transmisión, así mismo, las ubicaciones de los satélites se conocen con precisión de centímetros. Además, los receptores incluyen un registro de las posiciones teóricas de cada uno de éstos. Las correcciones de estas posiciones, también se transmiten por los satélites para determinar la ubicación geográfica.[58]

1.4.4. Receptor GPS.

Un receptor GPS es un circuito electrónico que opera en base a la información transmitida por los satélites para determinar su ubicación geográfica. En la Figura 1.3, se observa un diagrama en bloques de un receptor GPS).

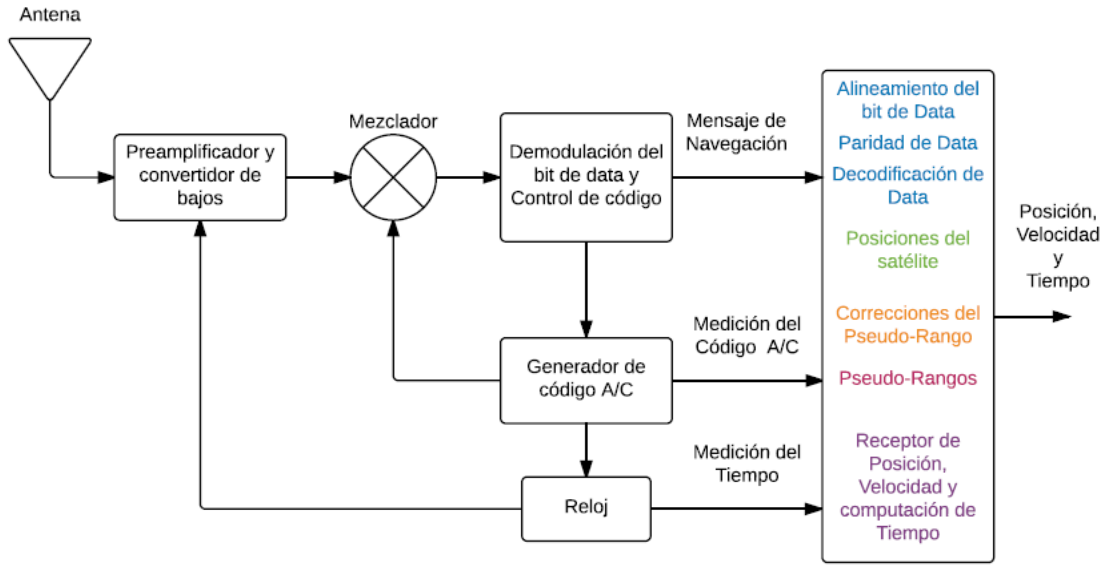


Figura 1.3. Partes de un receptor GPS.[59]

Las partes del diagrama se definen a continuación:

- **Antena.** Permite la recepción de las señales de los satélites al módulo GPS.
- **Preamplificador y convertidor de bajos.** Dado que las señales recibidas por los satélites se caracterizan por ser de alta frecuencia y baja potencia, éste filtro permite la adecuación de las señales para su posterior tratamiento en el receptor.

- **Mezclador.** Este bloque combina las frecuencias recibidas por los satélites y el Generador de código A/C para obtener una única señal. Esta señal luego será demodulada para generar una palabra binaria que posee las posiciones satelitales y otros datos, conocida como Mensaje de Navegación.
- **Demodulación del bit de data y Control de código.** Se encarga de eliminar la portadora de la señal combinada y extraer la información de la misma.
- **Generador de código A/C.** Permite la obtención del código A/C en función del patrón aleatorio de datos de cada satélite. Este pseudo-código sirve de retroalimentación permitiendo conocer el tiempo invertido por la señal al recorrer la distancia entre un satélite y el receptor.
- **Reloj.** Permite la sincronización entre las señales recibidas y las señales procesadas por el módulo GPS.

En la Tabla 1.1, se muestran las funciones y características de un receptor GPS.

Tabla 1.1. Funciones y características genéricas de un receptor GPS.[60]

Receptor GPS		
Funciones elementales	**Características técnicas**	**Permiten**
Sintonizarse con las señales de los satélites	La primera posición bidimensional se puede obtener en menos de 120 segundos	Crear rutas de navegación
Decodificar e interpretar el código de navegación	La primera posición tridimensional se puede obtener en menos de 150 segundos	Disponer de almacenamiento de los datos
Calcular el retardo de la señal que llega del transmisor hacia receptor	Muestreo de la posición con tiempos de 0.5 a 1 segundos	Mostrar mapas
Presentar información de la posición del receptor	Precisión alrededor de 15 metros	

1.5. Módulo GSM

1.5.1. Fundamentos de la tecnología GSM.

Es el primer estándar de los sistemas de telefonía móvil conocido como Sistema Global para Comunicaciones Móviles (GSM), correspondiente a la segunda generación (2G) de tecnología celular. Este permite la transición de la comunicación de radio analógico a radio digital, y se introdujo comercialmente a principios de la década de 1990. Actualmente, GSM continúa siendo el estándar más común en el mundo y es utilizado principalmente para la telefonía, transmisión de datos con conmutación de paquetes y mensajería corta (SMS).[61]

Los teléfonos GSM utilizan una tarjeta SIM (Módulo de Identidad del Suscriptor), que consiste en una identidad digital criptográfica ligada al número telefónico del usuario. Esta tarjeta cuenta con recursos de almacenamiento limitado y permite un registro de direcciones que la persona puede utilizar para comunicarse con sus contactos. La ventaja más llamativa de esta tecnología es la posibilidad de retirar la tarjeta SIM de un teléfono para ser utilizado en otro teléfono móvil manteniendo toda la información importante del usuario.[62]

1.5.2. Arquitectura.

Una red celular puede resultar más compleja de describir debido a la gran cantidad de elementos que la conforman. Sin embargo, la arquitectura GSM puede ser agrupada en tres componentes básicos: Estación móvil (MS), Subsistema de estaciones base (BSS) y Subsistema de red y conmutación (NSS). Esta arquitectura de red se observa en la Figura 1.4.[63]

- **Estación móvil (MS)**

Esta estación se compone de dos elementos que son, el teléfono móvil con la batería y el Módulo de Identidad del Suscriptor (SIM). Técnicamente, el teléfono es uno de los dispositivos más complejos de la red GSM, y puede identificarse mediante un código permanente conocido como Identidad del Equipo Móvil Internacional (IMEI) que a su vez, permite determinar si el dispositivo es legítimo o ha sido robado.[64]

Por otra parte, la tarjeta SIM permite identificar al usuario en la red. Es un chip que puede ser removido con facilidad de un dispositivo móvil, y cada tarjeta posee un código único llamado Identidad Internacional del Abonado a un Móvil (IMSI). El número telefónico de un celular puede estar entre 10 y 15 dígitos, siendo los tres primeros el código del país, los siguientes dos representan la central de conmutación y los números restantes representan el teléfono del abonado.[65]

- **Subsistema de Estaciones Base (BSS)**

En una red GSM la comunicación con los dispositivos móviles está relacionada principalmente con las estaciones base. Estas se conforman de dos elementos, denominados Estación de transferencia de base (BTS) y el controlador de estaciones base (BSC).[66]

Las BTS se encargan de procesar los canales de radio en una interfaz Um[67] en una celda, con protocolos asociados a la comunicación utilizada, mientras que las BSC comunican y controlan las BTS mediante una interfaz de canales de 64 kbits/s.[68]

- **Subsistema de Red y Conmutación**

Se encarga de la parte más compleja de la red. Está asociada al control y administración del sistema, que consiste en dos elementos:[69]

- **Centro de conmutación de servicios móviles (MSC),** se encarga de gestionar conexiones de los usuarios en la red y en la Red Telefónica Pública Conmutada (PSTN).
- **Puerta de centro de conmutación de servicios móviles (GMSC),** se encarga de administrar las conexiones de los usuarios cuando realizan una comunicación fuera de la red móvil.

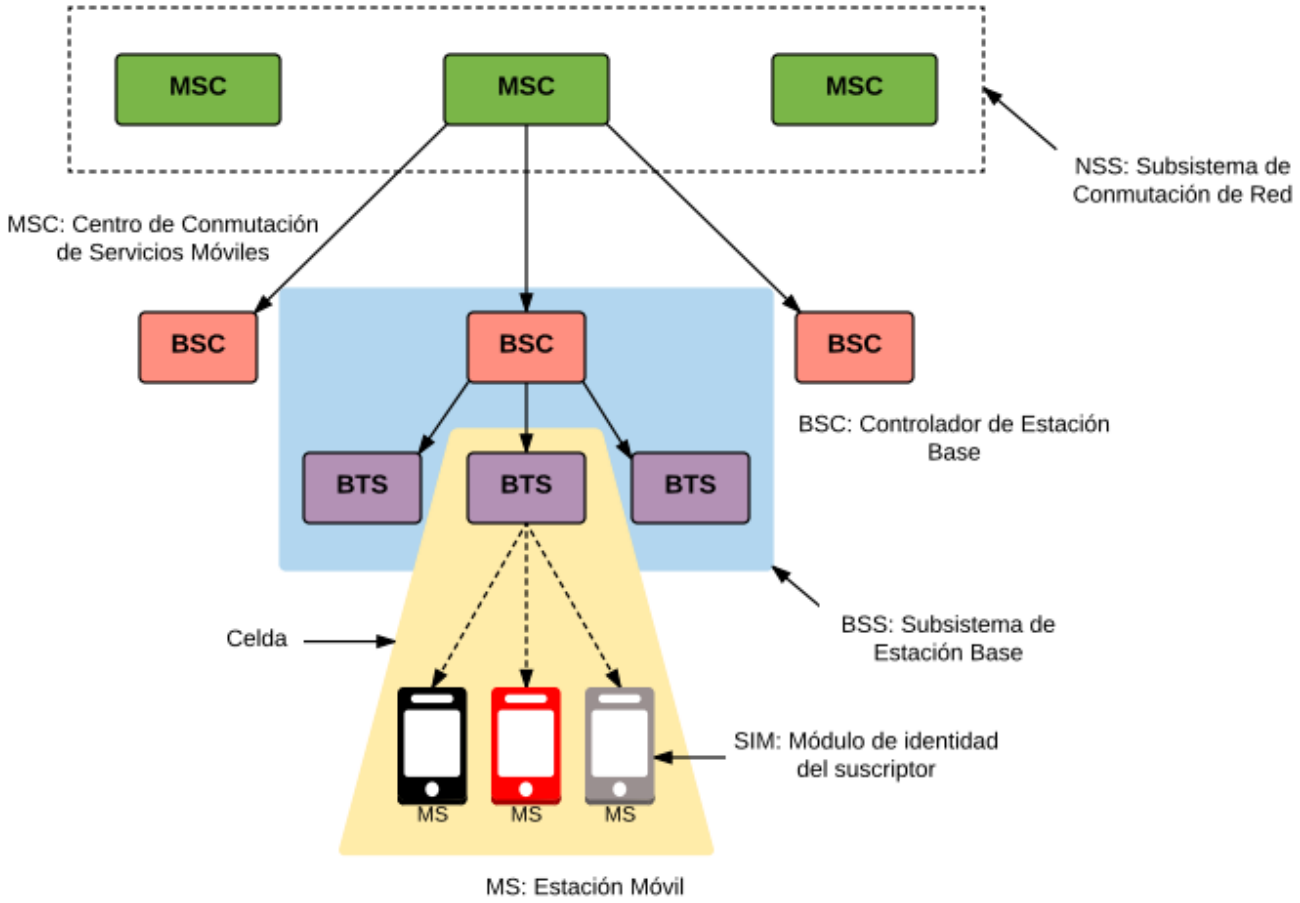


Figura 1.4. Arquitectura de una red GSM.[70]

1.5.3. Módem GSM.

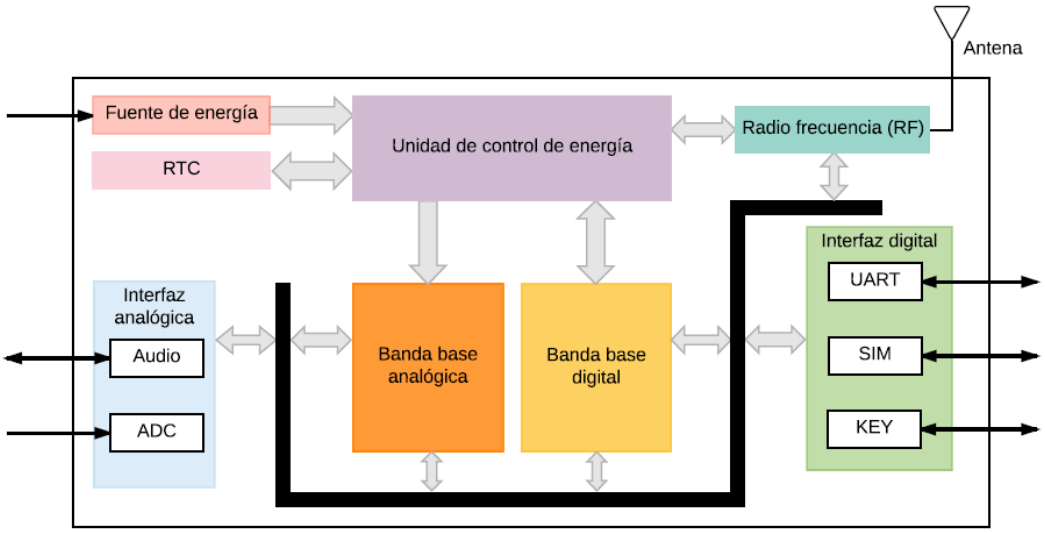


Figura 1.5. Partes de un módem GSM.[71] *(SIMCOM, - Hardware Design 2015).*

El sistema GSM se ha introducido para transmitir voz, pero también, se ha convertido rápidamente en una herramienta para la transmisión de datos.[72] Por otra parte, la red GPRS[73] posee gran popularidad dado que incorpora algunas mejoras a la tecnología GSM orientadas a la conmutación de paquetes.[74] Varios módulos están disponibles en el mercado y se utilizan en diferentes equipos industriales, para transferir datos de forma remota. Los módems GSM son los dispositivos encargados de la comunicación entre los móviles celulares, los cuales se enlazan a las redes telefónicas para brindar servicios como el envío/recepción de mensajes vía SMS y la realización/recepción de llamadas de voz.[75] Como se muestra en la Figura 1.5, un módulo GSM se encuentra constituido básicamente por:

- **Unidad de control de energía**. Permite el manejo y distribución de alimentación.
- **Interfaz y banda base analógica.** Posibilita el audio en el dispositivo.
- **Interfaz y banda base digital.** Se utiliza para el control del módem mediante protocolos de comunicación.

- **Circuito de radiofrecuencia (RF).** Se encarga de procesar los datos enviados y recibidos por el dispositivo a través de la antena.
- **Antena**. Permite la transmisión y recepción de información de forma inalámbrica.

1.5.4. Servicio SMS.

Este servicio utilizado en la red GSM, permite la transferencia de mensajes de texto entre estaciones móviles y otra estación. Para el caso de los dispositivos celulares, ambas estaciones son estaciones móviles y se logran mediante una comunicación extremo a extremo. Algunas aplicaciones de este servicio pueden ser: reportes de falla en determinados equipos, soporte técnico, suscripción de diversos servicios y control de sistemas vía mensajes SMS.[76]

1.5.5. Modalidades.

En la actualidad, destacan dos tipos de módems GSM, que pueden utilizarse según las necesidades del usuario:

- **Módems disponibles en PCB**[77]**.** Estos dispositivos son módems realizados en tarjetas electrónicas y sirven para desarrollar hardware de forma más específica. Son especialmente útiles para diseñadores de sistemas electrónicos.
- **Módems para computadoras.** Son aparatos que permiten a los computadores conectarse a internet a través de la red móvil.

2. DISEÑO DE SISTEMA ELECTRÓNICO LOCALIZADOR (SEL)

2.1. Diagramas generales del sistema

El diseño de este sistema electrónico contempla el desarrollo de sus partes tanto físicas como virtuales, desde un punto de vista funcional. En las figuras 2.1 y 2.2, se observan diagramas en bloques del hardware y software que componen el sistema.

2.1.1. Hardware.

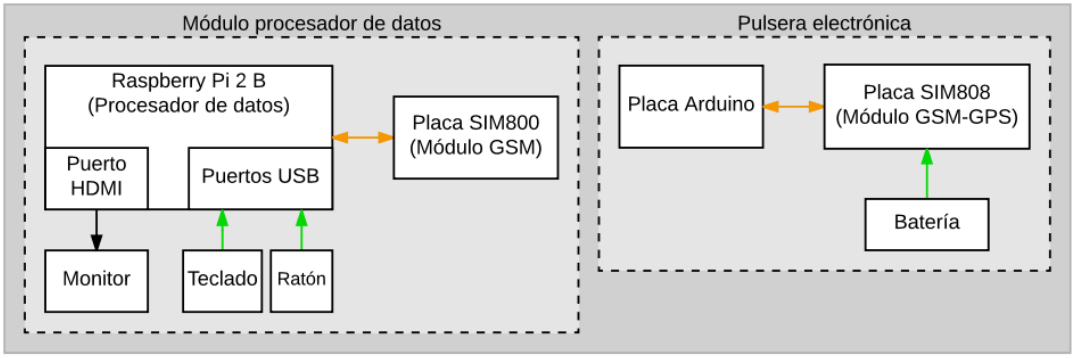


Figura 2.1. Estructura física del SEL.

2.1.2. Software.

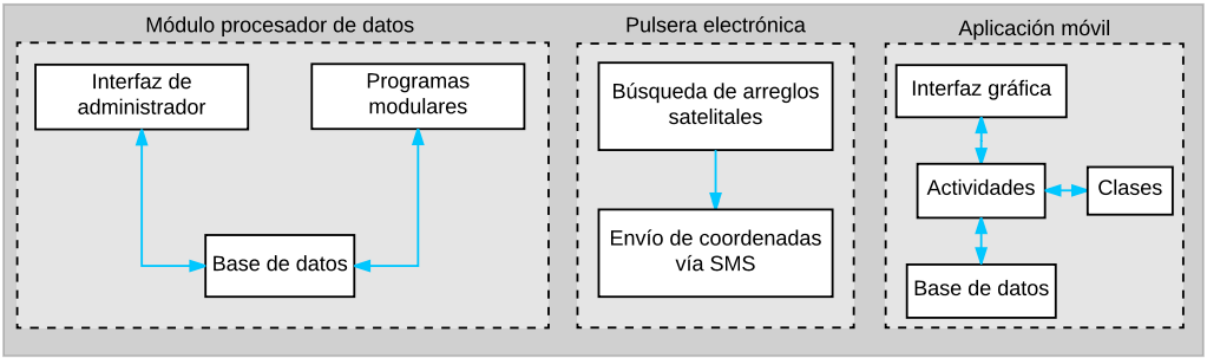


Figura 2.2. Programas del SEL.

2.2. Integración del microcontrolador y módulos GPS-GSM

El diseño del sistema supone el análisis de los elementos que permiten su aplicación y operatividad. En este sentido, el proyecto se desarrolla con un enfoque modular, empleando dispositivos electrónicos que permitan una estructura adaptable, compatible y funcional.

2.2.1. Módulos GSM y GPS.

El sistema electrónico para la localización, consiste en una estructura compuesta por dispositivos (un procesador de datos y un localizador), que trabajan en conjunto para obtener, procesar, enviar, recibir y almacenar la localización de un individuo en cuestión.

La localización es un proceso realizado por el módulo GPS, el cual es el encargado de recibir las señales satelitales para calcular las coordenadas geográficas de la posición del mismo. En adición, los módulos GSM corresponden a la etapa responsable de establecer comunicación con otros dispositivos de la red móvil. La pulsera electrónica (dispositivo localizable), dispone de un módulo SIM808 que conlleva tanto la etapa de localización como la de comunicación vía GSM (el esquemático de este circuito se muestra en el Apéndice C, Figura 1).

Por otra parte, el procesador de datos emplea un módulo SIM800 que le permite establecer comunicación con la pulsera electrónica (véase el Apéndice C, Figura 2 para conocer el esquemático de este módulo).

2.2.1.1. Módulo SIM808.

Figura 2.3. Módulo GPS-GSM, SIM808.

Esta es una plataforma de teléfono celular, utilizada como placa de prueba funcional para la creación del prototipo de la pulsera electrónica. Las razones de su selección se deben a las características de hardware que posee, de las cuales, únicamente se aprovecharon las presentadas en la Tabla 1.2.

Tabla 1.2. Características de hardware del módulo SIM808.[78]

Módulo SIM808
Características
Conector para Batería LIPO (Litio-Polímero) para que el dispositivo pueda ser portátil
Puerto USB para cargar la batería
Chip SIM808
Compartimiento instalado para tarjeta SIM
Conectores uFL de antenas externas GPS y GSM
Conectores IDC para establecer comunicación con la placa
Antena GSM de 38mm con conector uFL. Ganancia: 3 dBi, Peso: 0.5g
Antena GPS de 9mm x 9mm x 6.5mm con conector uFL. Ganancia: -2Bi. Peso: 2.4g

El objetivo de utilizar este módulo en las pruebas funcionales, es demostrar la posibilidad de obtener la ubicación geográfica del dispositivo de forma periódica, empleando tecnología de localización GPS que posteriormente, pueda ser enviada de forma inalámbrica a través de SMS mediante tecnología GSM.

Figura 2.4. Chip SIM808.[79]

El chip SIM808, (véase la Figura 2.4) es el elemento principal de la placa de prueba presentada en la Figura 2.3, y el componente central de la etapa GSM-GPS de la pulsera electrónica, que ha de utilizarse en el diseño final debido a las consideraciones presentadas en la Tabla 1.3.

- **Batería:**

Es importante destacar que el módulo SIM808 está diseñado para funcionar con un tipo de batería en particular.[80] Por lo tanto, es necesario considerar las características del cargador y la batería a utilizar. Para que la pulsera electrónica sea portátil (tanto en el prototipo como en el diseño final) es relevante el uso de la misma y su capacidad de carga. Las especificaciones y requisitos del cargador se muestran en la Tabla 1.4.

Tabla 1.3. Características técnicas del chip SIM808.[81] *Términos del glosario: TTFF*[82].

Generales	GSM	GPS
Posee tecnologías GSM y GPS integradas que pueden utilizarse de forma independiente	Quad-band 850/900/1800/1900MHz. Selecciona entre cuatro bandas de frecuencia automáticamente	Sensibilidad: - Rastreo: -165 dBm - Arranque en frío: -147 dBm
Bajo consumo: 1.07 mA aproximadamente, en modo dormido (sleep)	Potencia de transmisión: Clase 4 (2W) para GSM 850 y EGSM 900, Clase 1 (1W) para DCS 1800 y PCS 1900	TTFF: - Arranque en frio: 30s - Arranque en temperatura ambiente: 28s - Arranque en caliente: 1s
Interfaz serial (UART) utilizada a una tasa de	Almacenamiento de SMS en la tarjeta SIM	Precisión: - Posición Horizontal: <2.5 m
Autobaudios entre 1200 bps hasta 115200 bps	Interfaz SIM	
Control vía comandos AT		
Rango de voltaje de funcionamiento: 3.4 ~ 4.4V		
Sus dimensiones son: 24.0 x 24.0 x 2.6 mm		
Pines para conexión de antenas GPS y GSM		
Peso: 3.30 g		

Tabla 1.4. Especificaciones de la batería y el cargador para el chip SIM808.[83]

Especificaciones de la batería	**Requisitos del cargador**
Tipo: Litio – Polímetro (LIPO)	Voltaje de salida: 5.0 V a 7.0 V
Voltaje normal: 3.7 V	Corriente mínima de la fuente: 1.0 A
Capacidad: Normal a 1100 mAh	Voltaje pico máximo de 10 V permitido durante 1 ms mientras no se esté cargando
Voltaje de carga: 4.200 +- 0.050 V	Corriente pico máxima de 1.6 A, permitida durante 1 ms mientras está cargando
Corriente de carga máxima: 1.0A	Circuito Integrado MCP73831 Cargador de baterías LIPO
Método de carga: CC/CV (Corriente constante/ Voltaje constante)	
Voltaje Cut-off de descarga: 3.0V / cell	
Resistencia interna: <= 150 m ohms	

2.2.1.2. Módulo SIM800.

Figura 2.5. Módulo GSM, SIM800 (modelo H).

El módulo SIM800H es una placa electrónica utilizada junto al procesador de datos para la elaboración del prototipo del módulo procesador de datos. Sus características son similares al módulo SIM808, con la diferencia de que no posee tecnología GPS. El motivo de su selección es debido a las características de hardware mostradas en la Tabla 1.5.

Tabla 1.5. Características de hardware del módulo SIM800H.[84]

Módulo SIM800
Características
Conector para Batería LIPO para que el dispositivo pueda ser portátil
Chip SIM800
Compartimiento instalado para tarjeta SIM
Conector uFL de antena externa GSM
Antena GSM de 38mm con conector uFL. Ganancia: 3 dBi
Conectores IDC para establecer comunicación con la placa y para cargar la batería

Figura 2.6. Chip SIM800H.[85]

El chip SIM800 (véase la Figura 2.6), constituye la parte más importante de la placa de prueba mostrada en la Figura 2.5, y el componente electrónico de control requerido en el diseño final de la etapa GSM del módulo procesador de datos. Su selección se basa en las consideraciones y prestaciones técnicas mostradas en la Tabla 1.6.

Tabla 1.6. Características y prestaciones técnicas del chip SIM800H.[86]

Generales	**GSM**
Posee tecnología GSM	Quad-band 850/900/1800/1900MHz
Dimensiones: 15.8 x 17.8 x 2.4 mm	Potencia de transmisión: - Clase 4 (2W) para GSM 850 y EGSM 900 - Clase 1 (1W) para DCS 1800 y PCS 1900
Interfaz serial (UART) utilizada a una tasa de 115200 bps	Interfaz y Almacenamiento de SMS en la tarjeta SIM
Autobaudios entre 1200bps hasta 115200 bps	Interfaz SIM
Control vía Comandos AT	
Rango de Voltaje de funcionamiento: 3.4 ~ 4.4V	
Bajo consumo: (0.83-0.92) mA aproximadamente, en modo dormido (sleep)	
Puerto para tarjeta SIM tamaño estándar	
Pin para conexión de antena GSM	
Permite la depuración y la actualización del firmware mediante la interfaz USB	

2.2.2. Microcontrolador ATMEGA328P.

Como parte de la selección de tecnologías, se ha determinado el uso de un microcontrolador (MCU[87]) como dispositivo de control en la pulsera electrónica. Este proyecto hace uso del ATMEGA 328P (PDIP[88]), un microcontrolador de la familia ATMEL que cumple con los parámetros necesarios para controlar el módulo de comunicación GSM-GPS. Las características de este circuito integrado se muestran en la Tabla 1.7.

Tabla 1.7. Características técnicas del microcontrolador ATMEGA 328P.[89]

MCU ATMEGA328P
Características
Arquitectura de 8 bits
Comunicación Serial: UART (Software y Hardware)
Rango de Voltaje de funcionamiento: 1.8 ~ 5.5 V
Frecuencia de operación máxima: 20 MHz
Memoria Flash de 32 KB

Este microcontrolador es utilizado en la plataforma Arduino y es ampliamente recomendado para pruebas y prototipos electrónicos.

2.2.3. Arduino.

La plataforma Arduino, consiste en una placa de bajo costo, práctica y fácil de utilizar para el desarrollo de proyectos. Este sistema embebido se caracteriza en lo que respecta al hardware, por disponer de puertos digitales y analógicos de entrada-salida, así como puertos especiales (I2C[90], UART, SPI[91] y PWM[92]). Además, posee un entorno de desarrollo (IDE) conocido como software Arduino basado en Processing y su propio lenguaje de programación fundamentado en Wiring.[93]

Como dispositivo de control, el Arduino establece comunicación mediante la interfaz UART utilizando comandos AT[94] con el módulo GSM-GPS (SIM808), al que envía solicitudes y procesa sus respuestas. Una de las diferencias de esta plataforma respecto a otras placas, es el uso de un controlador USB[95]-a-Serial para conectar Arduino con una computadora y programarlo.

En adición, es importante mencionar que el desarrollo del diseño final contempla el uso del ATMEGA328P en su versión de encapsulado SMT[96] como sustituto de esta plataforma, con el propósito de integrarse con el chip SIM808 y constituir la pulsera electrónica. Las especificaciones técnicas de esta placa se observan en la Tabla 1.8.

Tabla 1.8. Especificaciones técnicas de la placa Arduino UNO.[97]

Placa Arduino
Especificaciones técnicas
Voltaje de operación: 5V
Voltaje de entrada recomendado: 7-12V
Voltaje límite: 6-20V
Pines digitales I/O: 14 incluyendo 6 de salida PWM
Corriente DC por pines I/O: 20mA, 50mA para el pin de 3.3V
Frecuencia del reloj: 16MHz
Memoria flash: 32Kb, (0.5 utilizados por el bootloader)

2.3. Protocolo y comandos de comunicación

2.3.1. Transmisor-Receptor Asíncrono Universal (UART).

Es un protocolo de comunicación asincrónico full-dúplex que sirve como interfaz de comunicación para la transmisión y recepción de información de forma serial entre dos dispositivos. Su funcionamiento se basa en la estructuración de paquetes de datos con un tamaño de 8 bits (1 byte) para ser transmitidos bit por bit de forma secuencial a otro dispositivo, cuyo UART al recibirlo realiza el mismo proceso inverso. Este tiene incluido un registro de desplazamiento el cual es un método básico para realizar las conversiones de la información con formato paralelo a serial y viceversa. Cabe destacar que esta comunicación serial puede utilizarse no solo por hardware sino también por software.[98]

En el caso de los microcontroladores y microprocesadores, si se va a utilizar por hardware, generalmente tienen los puertos habilitados como digitales de forma predeterminada. Por lo que es requerido que en primera instancia sean configurados como puerto serial. La razón para hacer uso de la comunicación UART se debe principalmente a su popularidad y compatibilidad con otros dispositivos, tales como computadoras, periféricos y módulos electrónicos.

2.3.2. Comandos AT.

Los comandos AT o comandos Hayes conforman un lenguaje utilizado para establecer comunicación con los módems. Sin embargo, su aplicación se ha expandido a los móviles con tecnología GSM.[99]

Estos comandos se encuentran pre-grabados en estos dispositivos y consisten en instrucciones que permiten a los dispositivos móviles realizar acciones específicas tales como realizar y recibir llamadas, enviar y recibir mensajes SMS y configurar diversas opciones de la terminal. Además, las características de funcionamiento de estos comandos hacen posible que puedan ser empleados para la comunicación entre los sistemas embebidos y los módulos adyacentes.[100]

En este caso particular, tanto en el prototipo como en el diseño final del sistema, los comandos AT tienen la función de comunicar al microcontrolador con el chip SIM808 y al microprocesador con el SIM800. Los comandos más utilizados en la programación de la pulsera electrónica y el módulo procesador de datos se presentan en la Tabla 1.9.

Tabla 1.9. Comandos AT utilizados con frecuencia en el SEL.[101] *Términos del glosario: ICCID*[102].

Comandos	Función
AT	Es el comando básico de los comandos AT. Se utiliza para confirmar que el módulo se encuentra disponible
ATI	Este comando permite mostrar información del dispositivo
AT+CCID	Permite mostrar el número ICCID de la tarjeta SIM
AT+GSN	Presenta el número de IMEI del dispositivo
AT+CREG?	Sirve para mostrar el estado de la red
ATD<n>	Es utilizado para la realización de llamadas
ATH	Sirve para colgar una llamada
AT+CMGF=1	Habilita el modo lectura de los SMS guardados
AT+CMGR=<>	Permite seleccionar un SMS para leerlo
AT+CMGD=<n>	Permite seleccionar un SMS para borrarlo
AT+CMGS	Permite enviar mensajes SMS
AT+CPMS?	Sirve para conocer la cantidad de mensajes guardados
AT+CGPS=1	Habilita el uso del GPS
AT+CGPSSTATUS?	Permite conocer si se ha encontrado un arreglo de satélites

2.4. Programación del procesador de datos y de la pulsera electrónica

La estructura del sistema electrónico no corresponde únicamente al desarrollo de las etapas de hardware que permiten designar las condiciones de operación y las consideraciones físicas del mismo, sino también, a los procesos lógicos que definen el software de un procesador de datos y un dispositivo localizable.

2.4.1. Módulo procesador de datos.

2.4.1.1. Raspberry Pi.

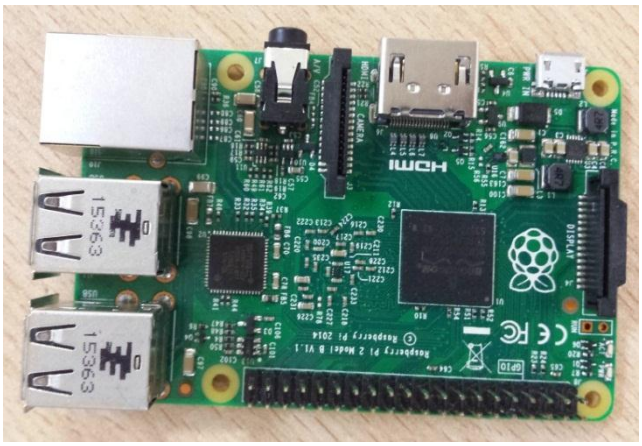


Figura 2.7. Raspberry Pi 2 Modelo B V1.1.

La Raspberry Pi es un computador de bajo costo diseñado con el propósito de enseñar a programar. Esta plataforma incorpora Python como lenguaje de programación, y se caracteriza por disponer de un software de código libre que utiliza un sistema operativo oficial basado en Linux, llamado Raspbian. En este sistema se implementa la versión 2 Modelo B V1.1, como módulo de procesamiento y almacenamiento de datos.[103] Un ejemplar de esta plataforma se muestra en la Figura 2.7, y sus características generales se observan en la Tabla 1.10.

Tabla 1.10. Características técnicas de la Raspberry Pi 2 Modelo B V1.1.[104] *Términos del glosario: SoC*[105], *ARM*[106].

Raspberry Pi2 B
Especificaciones técnicas
SoC: BCM2836
Microprocesador CPU: ARM- Cortex A7
Arquitectura: ARM
Familia (Set de Instrucciones): ARMv7-A
Ancho de bits del núcleo: 32 bits
Frecuencia de Operación: 900 MHz
Microprocesador GPU: VideoCore IV

2.4.1.1.1. Disposición de pines.

La comunicación con el módulo GSM mediante el protocolo UART por hardware, requiere que se interconecten los pines específicos para la recepción y envío de datos. En la Raspberry Pi 2 estos son: pin de T_X[107] (8), pin de R_X[108] (10) y pin de GND (6, 9, 14, 20, 25, 30, 34 o 39). Ver Distribución de pines de la Raspberry Pi 2 en el Apéndice C, Figura 3.

2.4.1.2. Lenguaje de programación: Python.

Es un lenguaje de programación de tipo intérprete de código abierto, utilizado en el procesador de datos por su legibilidad y facilidad de uso. Se caracteriza por estar orientado a objetos y por el uso de sangría (*indentation*) como elemento sintáctico en la elaboración de programas. Se utiliza en el módulo procesador de datos debido a que es el lenguaje predeterminado en la Raspberry Pi.[109]

2.4.1.2.1. Librerías.

- **os:** El módulo permite acceder a funcionalidades dependientes del Sistema Operativo. Sobre todo, aquellas que permiten manipular la estructura de directorios. Es importada para abrir archivos (programas) desde otro archivo en ejecución.[110]
- **sys:** El módulo es el encargado de proveer variables y funcionalidades, directamente relacionadas con el intérprete. Su función en el sistema es manipular (trasladar y obtener) los parámetros de un archivo al ejecutarlo.[111]
- **tkinter:** Es un módulo que permite construir interfaces gráficas de usuario multiplataforma en Python. Esta librería crea las ventanas y los componentes visuales de la interfaz gráfica del administrador.[112]

- **ttk:** Es un módulo que permite configurar la apariencia de una ventana en el modo gráfico para separar la codificación de las acciones de una ventana, de las de su apariencia. En el sistema es utilizada para establecer la apariencia (tipo de letra, tamaño de letra, colores de fondo, entre otros) de todas las ventanas de la interfaz gráfica del administrador.[113]
- **math:** Es un módulo que contiene todas las funciones matemáticas de Python, con éste se ejecutan los cálculos matemáticos del sistema.[114]
- **time:** Este módulo posee diversas funciones relacionadas con el tiempo. Se encarga de ejecutar los retardos de los programas, y suministrar la fecha y hora del sistema.[115]
- **serial:** Este módulo tiene las funciones que permiten establecer contacto con el puerto serial. Es utilizado para la comunicación entre el módulo GSM y la Raspberry Pi.[116]
- **sqlite3:** Contiene las funciones pertinentes para establecer comunicación entre una base de datos y Python.[117]

2.4.1.3. Programas modulares: Descripción de funcionamiento.

En el módulo procesador de datos se han desarrollado determinados procesos que se mantienen en constante operación con la finalidad de tomar decisiones respecto a los datos que recibe de la pulsera electrónica y la aplicación móvil, estos procesos se llaman *programas modulares residentes*. También se utilizan otros procesos, que se ejecutan al ser llamados bajo circunstancias determinadas, denominados *programas modulares de proceso por solicitud*. Dado que algunos de estos procesos, pueden realizarse de forma simultánea a otros, se utilizan franjas de colores en esta sección, para indicar esta coincidencia entre los programas que tengan el mismo color. En la Figura 2.8, se observa una distribución de los programas modulares.

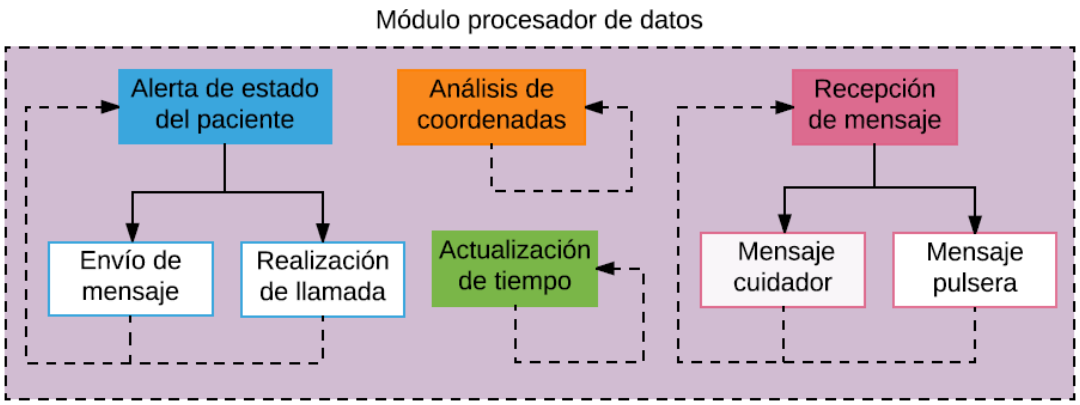


Figura. 2.8. Programas modulares del módulo procesador de datos.

2.4.1.3.1. Residentes.

2.4.1.3.1.1. Alerta de estado del paciente.

La función de este programa es iniciar un proceso de alerta en caso de que el campo 'estado' en la tabla "Resultado de análisis de las ubicaciones" de la base de datos sea igual a 'Peligro' o 'Desconocido'. Ambos estados representan una condición de atención urgente para el paciente.

El proceso consiste en emitir una alerta al cuidador a través de la aplicación móvil (ver Figura 2.9) que tendrá instalada en sus dispositivos inteligentes compatibles con Android. Partiendo de esto, si luego de la alerta transcurre un (1) minuto sin obtener respuesta, el módulo procesador de datos enviará un SMS al responsable del paciente, si posterior al mismo período de tiempo el custodio no da una respuesta de confirmación, se realizará una llamada al Sistema Nacional de Atención a Emergencias y Seguridad 9-1-1 (de aquí en adelante 911).

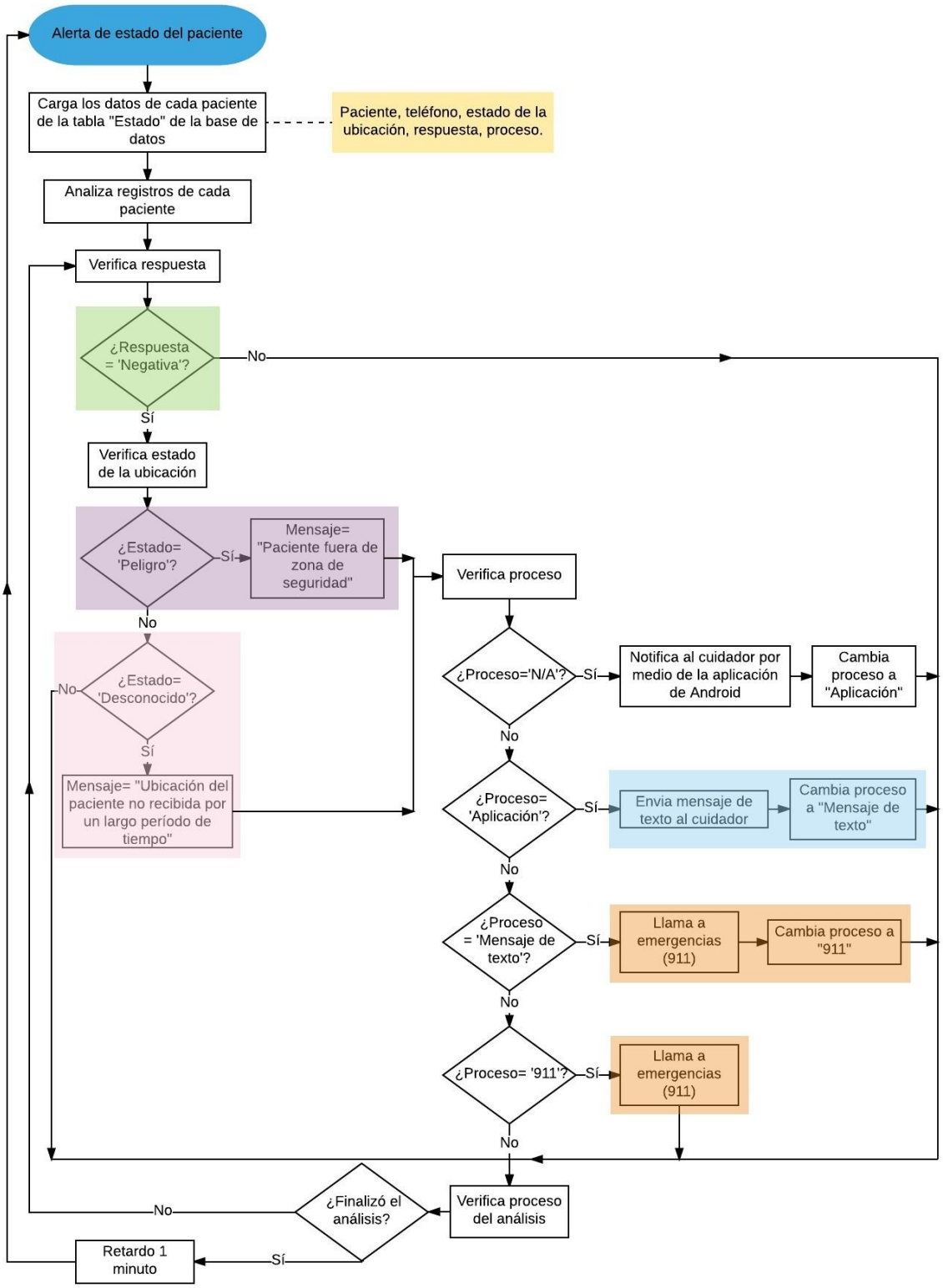
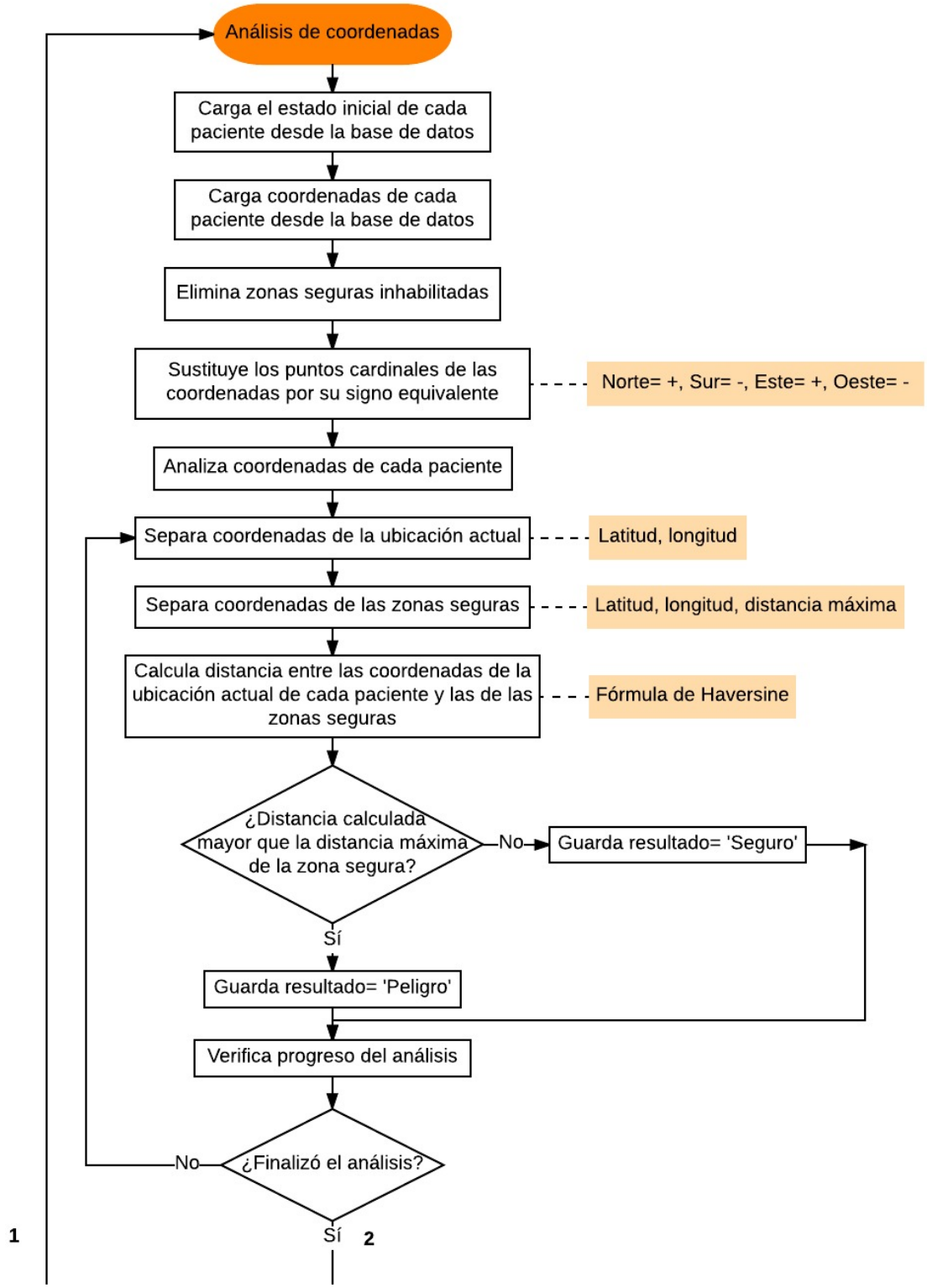


Figura 2.9. Flujo de procesos del programa "Alerta de estado del paciente". *Se encarga de ejecutar una alerta, de acuerdo al estado del paciente en el procesador de datos.*

Véase la descripción de los pasos de este programa en el Apéndice D, sección 1.

2.4.1.3.1.2. Análisis de coordenadas.



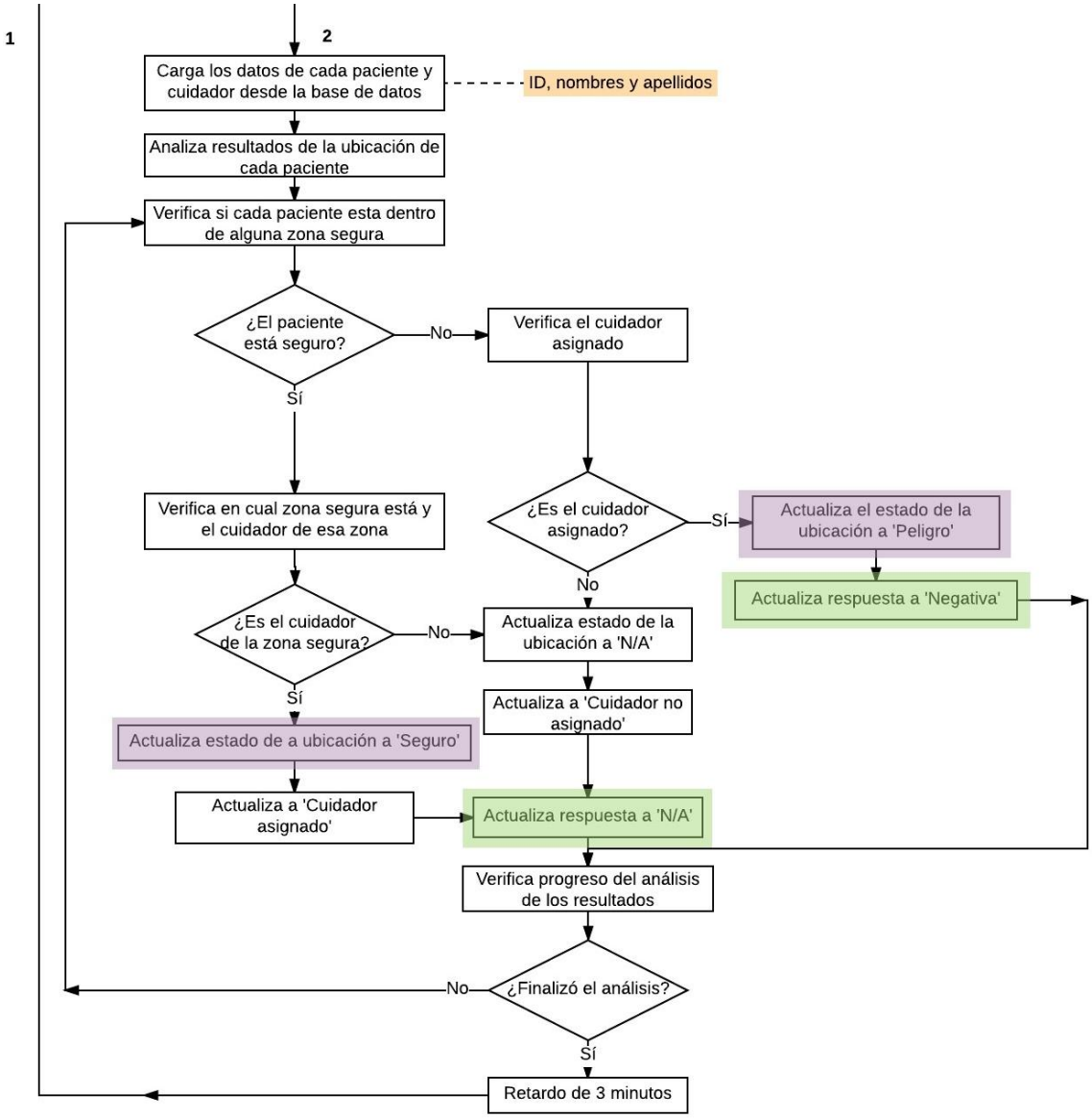


Figura 2.10. Flujo de procesos del programa "Análisis de coordenadas".

El flujo correspondiente a la Figura 2.10, permite analizar la ubicación actual del paciente comparándola con las coordenadas registradas de las zonas seguras. La descripción de este proceso y la fórmula de Haversine[118] se presentan en el Apéndice D, sección 2 y 2.1, respectivamente.

2.4.1.3.1.3. Actualización de tiempo.

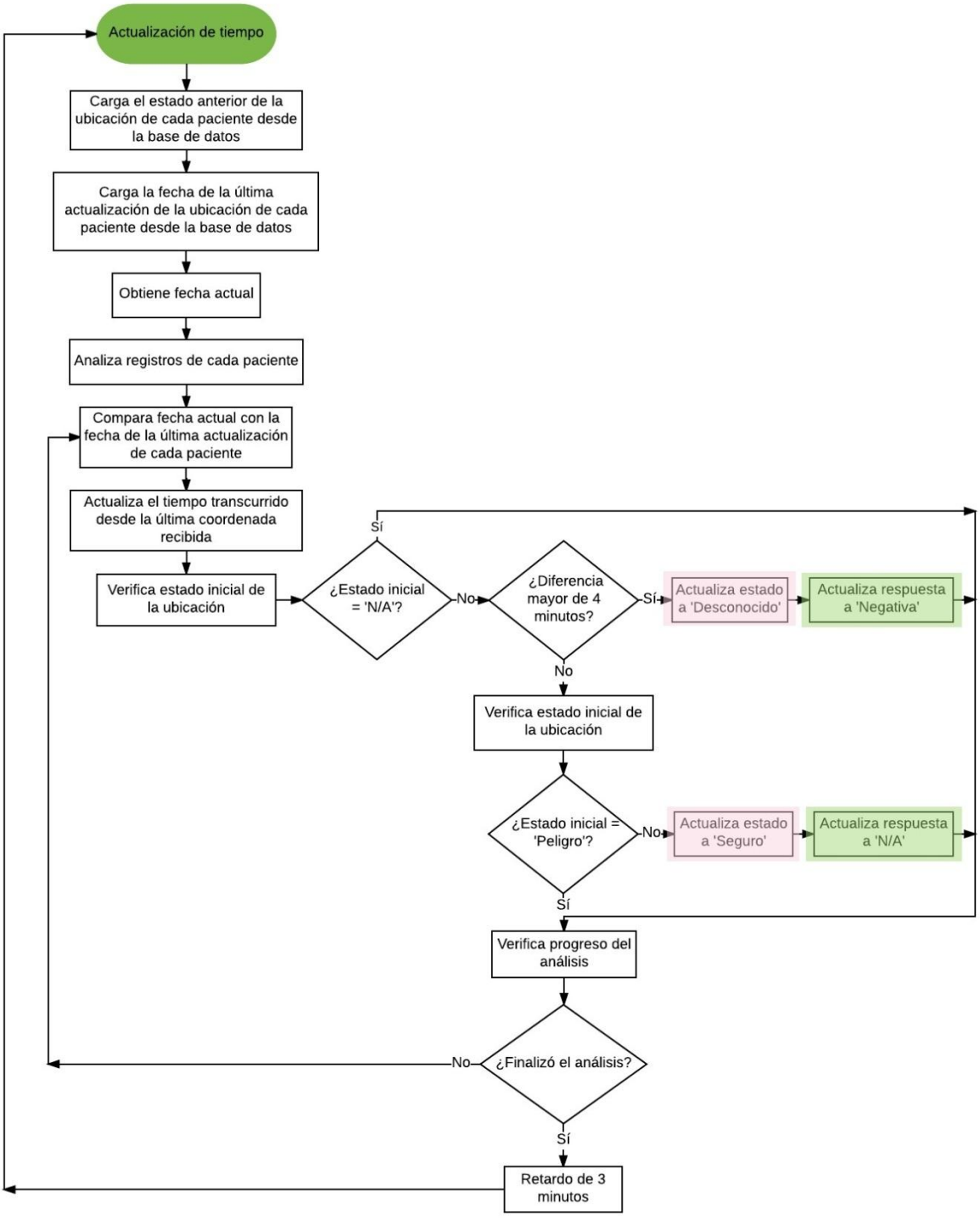


Figura 2.11. Flujo de procesos del programa "Actualización de tiempo".

El programa modular residente mostrado en la Figura 2.11, tiene como función determinar el tiempo transcurrido desde la última actualización de la ubicación del paciente. La descripción de los pasos de este programa se muestra en el Apéndice D, sección 3.

2.4.1.3.1.4. Recepción de mensaje.

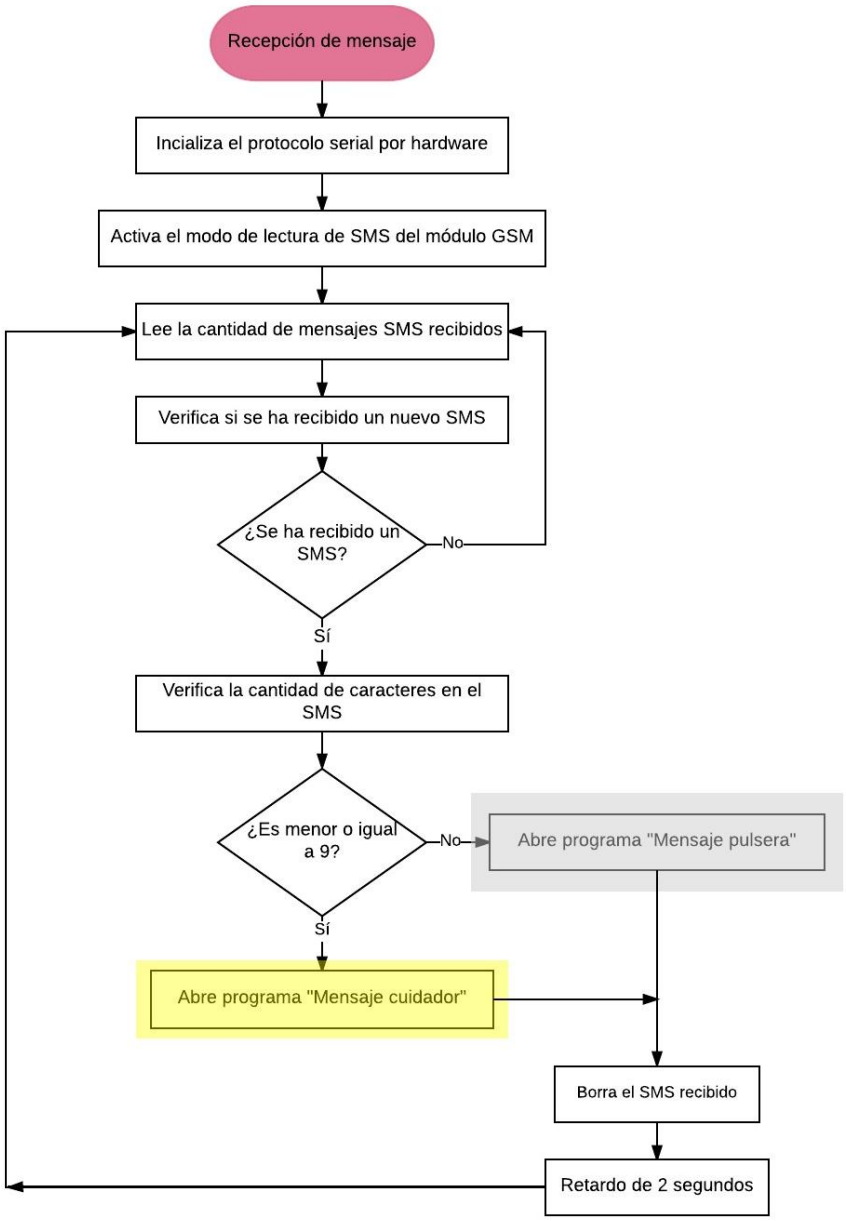


Figura 2.12. Flujo de procesos del programa "Recepción de mensaje".

El módulo procesador de datos, se encuentra constantemente a la espera de nuevos SMS. El programa “Recepción de mensaje”, se mantiene en ejecución para verificar si el mensaje proviene de la aplicación móvil de un cuidador, o de una pulsera electrónica, y luego extraer la información recibida (véase la Figura 2.12). La descripción de los pasos de este programa se puede observar en el Apéndice D, sección 4.

2.4.1.3.2. De proceso por solicitud.

2.4.1.3.2.1. Mensaje cuidador.

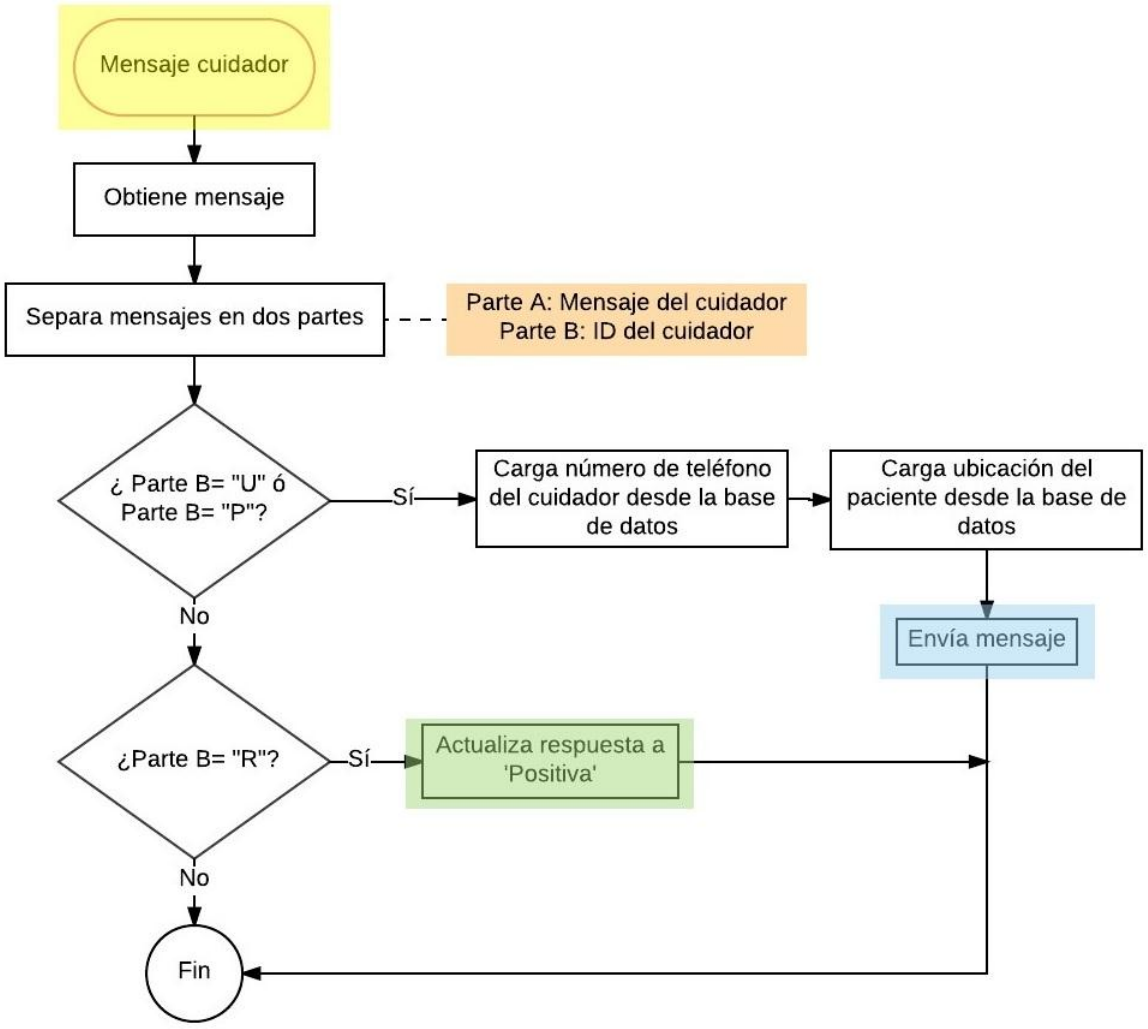


Figura 2.13. Flujo de procesos del programa “Mensaje cuidador”.

Este programa (mostrado en la Figura 2.13), es ejecutado cuando el mensaje de texto corresponde a algún cuidador previamente registrado en el sistema. El mensaje de texto, es analizado para determinar la información del mensaje.

La información del mensaje tienes tres posibles contenidos: 'U', el cuidador solicita la ubicación del paciente; 'P', la ubicación solicitada del paciente no fue recibida; 'R', el cuidador recibió la notificación de que el paciente está en peligro. Los pasos de este programa se muestran en el Apéndice D, sección 5.

2.4.1.3.2.2. Mensaje pulsera.

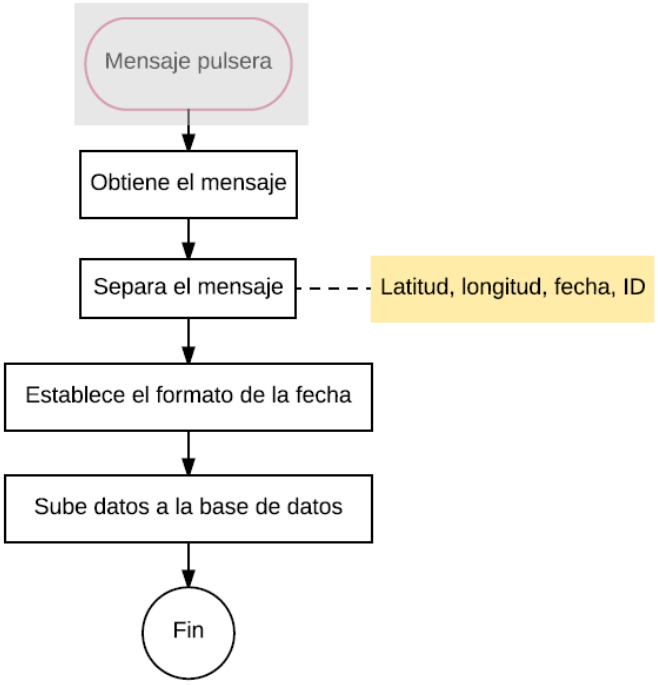


Figura 2.14. Flujo de procesos del programa "Mensaje pulsera".

Este programa (Observe la figura 2.14), es ejecutado cuando el mensaje de texto proviene de la pulsera electrónica. El mensaje de texto es guardado en la base de datos, ésta información se presenta al administrador en la tabla de "Registro de ubicaciones" de la interfaz gráfica. Ver la descripción de los pasos en el Apéndice D, sección 6.

2.4.1.3.2.3. Envío de mensaje.

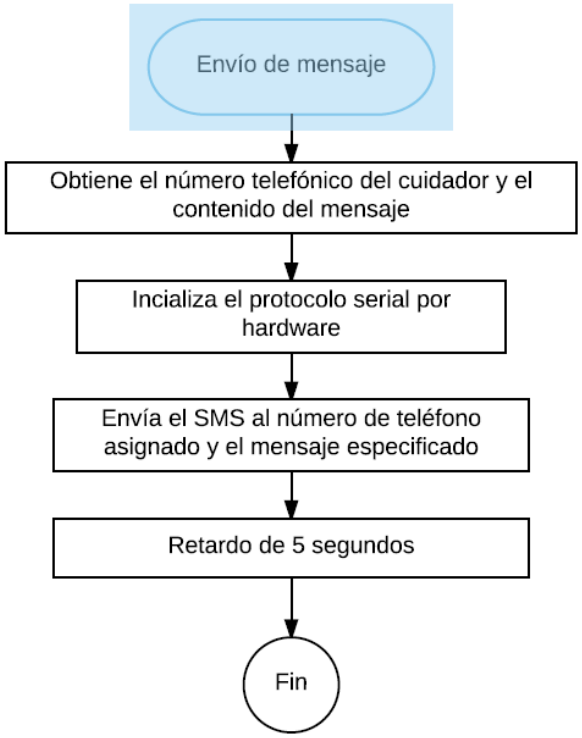


Figura 2.15. Flujo de procesos del programa "Envío de mensaje".

El programa "Envío de mensaje", mostrado en la Figura 2.15, es del tipo modular, pero no se encuentra en ejecución constantemente. Es llamado por el programa "Alerta de estado del paciente" cuando requiere avisar a un cuidador acerca de la situación del paciente. La descripción de este proceso se observa en el Apéndice D, sección 7.

2.4.1.3.2.4. Realización de llamada.

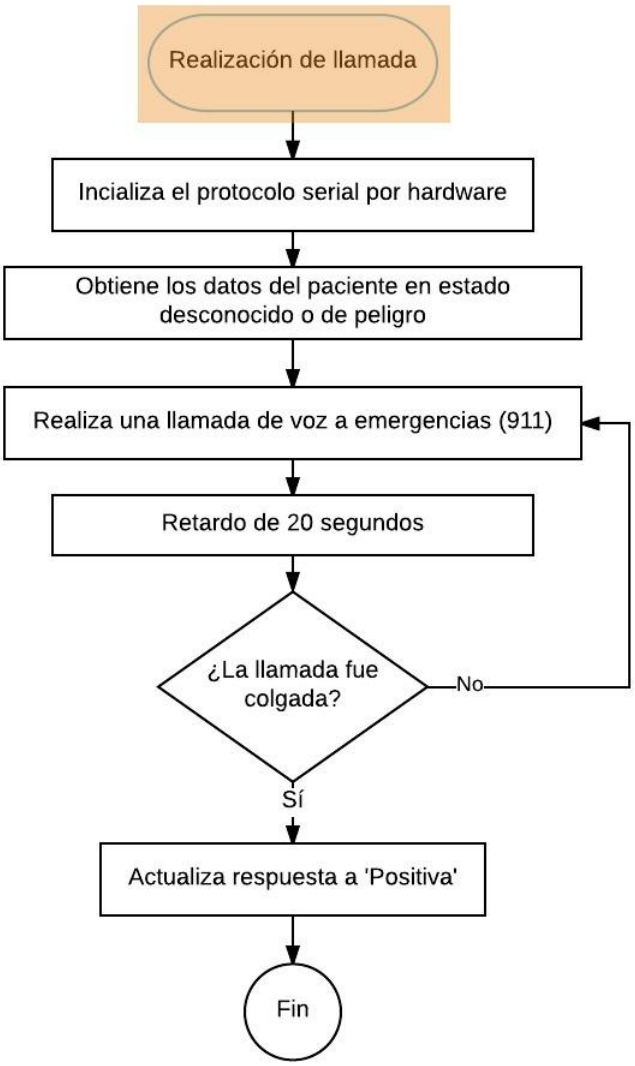


Figura 2.16. Flujo de procesos del programa "Realización de llamada".

Este programa no se ejecuta constantemente. Tiene la función de realizar llamada al número de emergencias "911", cuando el programa "Alerta de estado del paciente" detecte que el estado del paciente sea "Desconocido" ó "Peligro" (Figura 2.16). La descripción de este proceso se presenta en el Apéndice D, sección 8.

2.4.1.4. Base de datos.

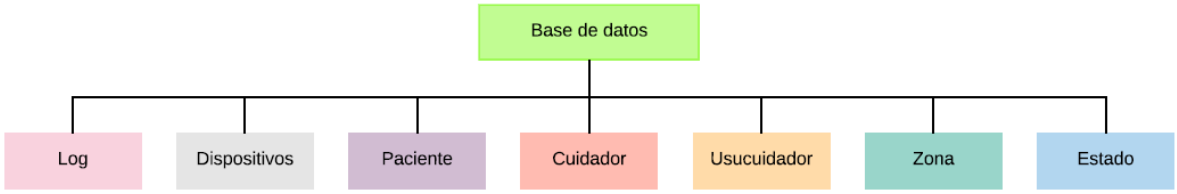


Figura 2.17. Tablas de la base de datos del sistema.

La base de datos del sistema es gestionada por el software de código abierto "SQLite". Esta herramienta permite almacenar información de forma rápida y eficaz con la particularidad especial, de correr sin inconvenientes en equipos con poca capacidad de hardware. Además, el acceso y manipulación con Python es sencillo (tiene su propia librería) y para su interacción se realizan los siguientes pasos:

1. Abrir una conexión con la base de datos.
2. Crear un cursor a través del cual se ejecutarán los comandos.
3. Ejecutar el comando.
4. Cerrar el cursor.
5. Cerrar la conexión con la base de datos.

Como se muestra en la Figura 2.17, la base de datos del SEL contiene siete (7) tablas: Log, Dispositivos, Paciente, Cuidador, Usucuidador, Zona y Estado.

- **Log**

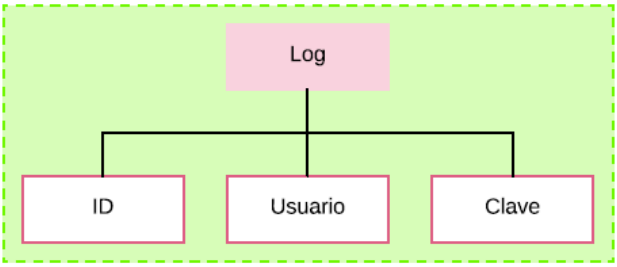


Figura 2.18. Tabla "Log".

"Log", es una tabla no visualizada en la interfaz gráfica del administrador, la cual contiene el número de identificación (ID), el nombre de usuario y la clave de cada usuario (administrador). Observe la Figura 2.18.

- **Dispositivos**

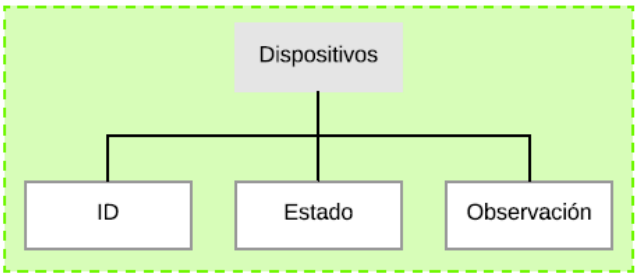


Figura 2.19. Tabla "Dispositivos".

"Dispositivos", es una tabla que contiene el registro de las pulseras electrónicas vigentes. Su versión en la interfaz gráfica del administrador es conocida como "Registro de pulsera", la cual se compone por el número de identificación (ID), tiempo de uso (Estado) y los comentarios (Observación) de los dispositivos. Véase la Figura 2.19.

- **Paciente**

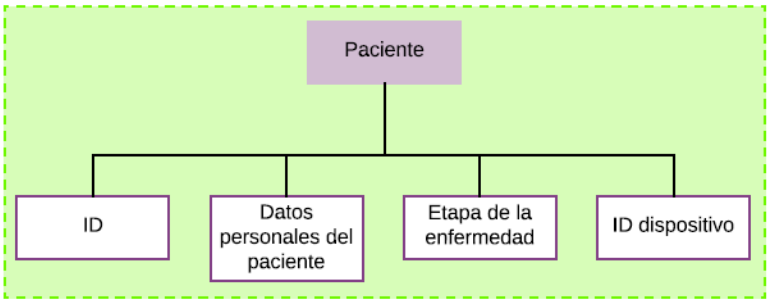


Figura 2.20. Tabla "Paciente".

"Paciente", es una tabla que tiene los datos significativos del paciente. Se presenta en la Interfaz gráfica del administrador como "Datos de paciente". Como se observa en la Figura 2.20, contiene el número de identificación, datos personales (nombres, apellidos, edad, cédula, dirección y teléfono), la etapa de la enfermedad del paciente y el número de identificación de la pulsera electrónica.

- **Cuidador**

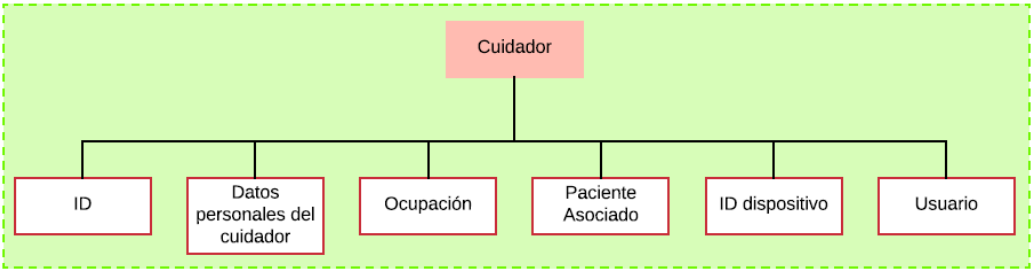


Figura 2.21. Tabla "Cuidador".

"Cuidador", es una tabla de datos (mostrada en la Figura 2.21) que almacena información acerca del cuidador. Es visualizada en la interfaz gráfica del administrador como "Datos de cuidador". Contiene el número de identificación, los datos personales, ocupación, usuario del cuidador, el paciente de quien está a cargo.

- **Usucuidador**

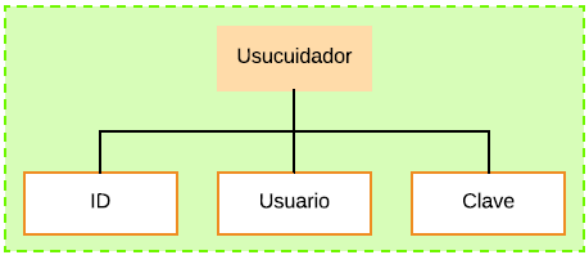


Figura 2.22. Tabla "Usucuidador".

"Usucuidador", es una tabla no visualizada en la interfaz gráfica del administrador, la cual, contiene el número de identificación (ID), el usuario y la clave de cada usuario (cuidador), como se muestra en la Figura 2.22.

- **Zona**

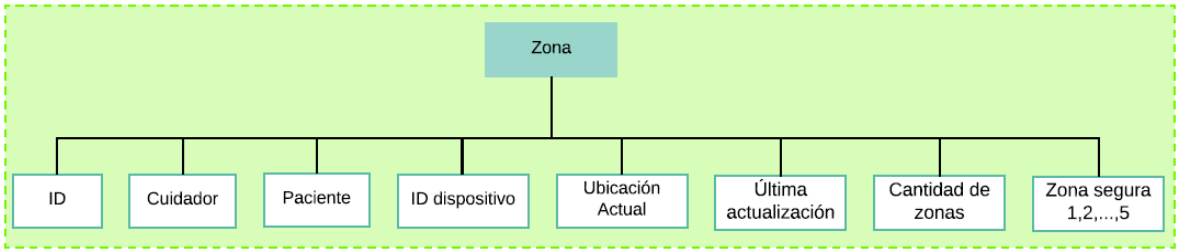


Figura 2.23. Tabla "Zona".

"Zona", es una tabla (ver la Figura 2.23) que contiene los datos de la ubicación del paciente y sus zonas seguras. En la interfaz gráfica del administrador se conoce como "Registro de ubicaciones". La misma está compuesta por un número de identificación de la relación enfermo-custodio y un segundo perteneciente a la pulsera electrónica, datos personales de cada cuidador y paciente, la fecha de la última actualización de la ubicación del paciente, las zonas seguras y la cantidad restante de éstas para ser registradas.

- **Estado**

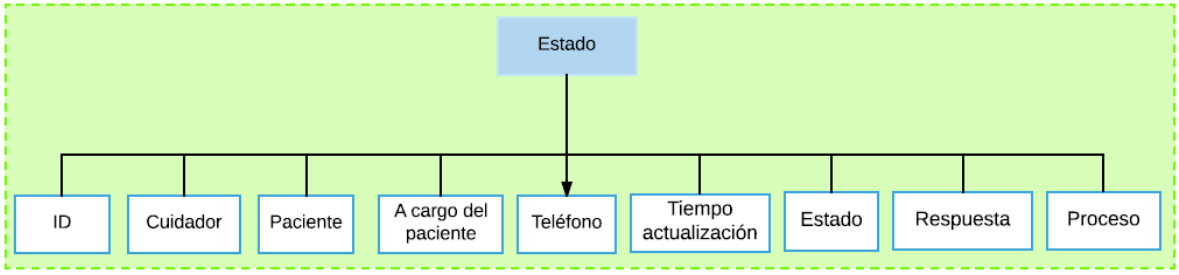


Figura 2.24. Tabla "Estado".

"Estado" es una tabla que almacena las conclusiones de los procesos ejecutados por el sistema. En la interfaz gráfica del administrador se puede visualizar como "Resultado de análisis de las ubicaciones". Como se muestra en la Figura 2.24, ésta contiene un número de identificación de la relación enfermo-custodio, los datos personales de cada cuidador y paciente, teléfono, el tiempo transcurrido desde que se recibieron las últimas coordenadas, el estado de la ubicación del paciente, la respuesta de la emisión de alerta y el proceso que se está ejecutando en función de dicha alerta de estado.

2.4.1.5. Interfaz de administrador.

Es una herramienta elaborada con la finalidad de controlar los datos recibidos en el módulo procesador de datos. La misma consiste en programas realizados en el lenguaje Python. Este último es seleccionado tanto para el prototipo y el diseño final de la interfaz debido a que es el lenguaje utilizado por el computador seleccionado (Raspberry Pi). La interfaz del administrador abarca una estructura funcional mostrada en la Figura 2.25.

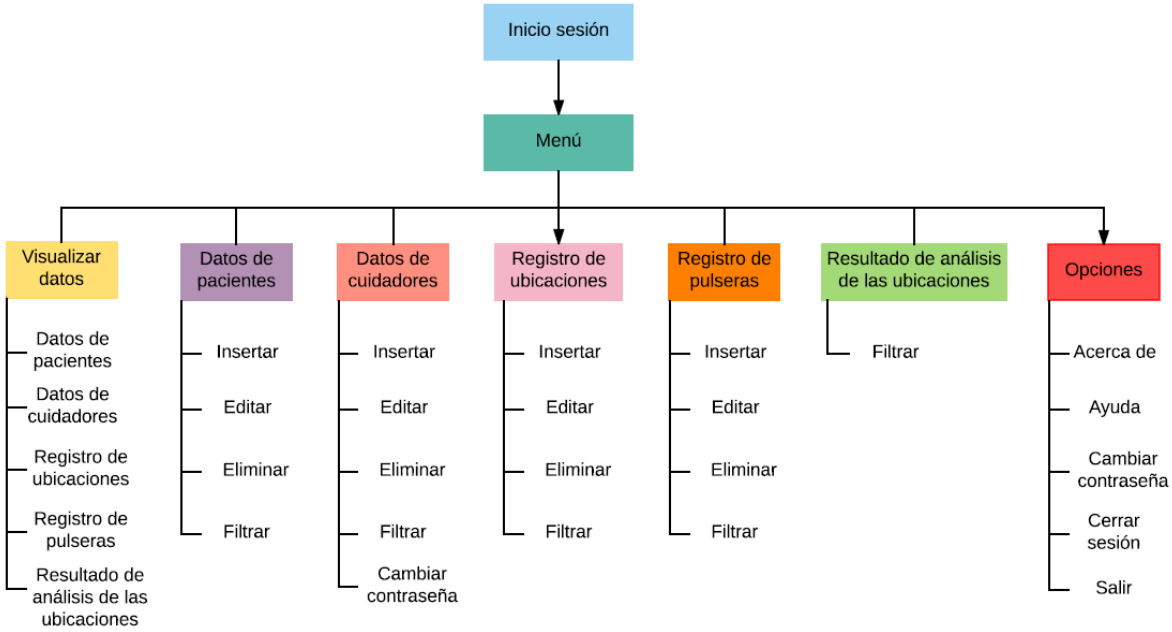


Figura 2.25. Interfaz de administrador del módulo procesador de datos.

La Interfaz de administrador parte del inicio de sesión para poder acceder a la misma. Luego, se presentará una ventana de menú que a su vez estará compuesta por las opciones: Visualizar datos, Datos de pacientes, Datos de cuidadores, Registro de ubicaciones, Registro de pulseras, Resultado de análisis de las ubicaciones y Opciones, las cuales a su vez, contienen diversas acciones que se muestran en el siguiente grupo de figuras, significando la franja color *aqua* un despliegue del menú.

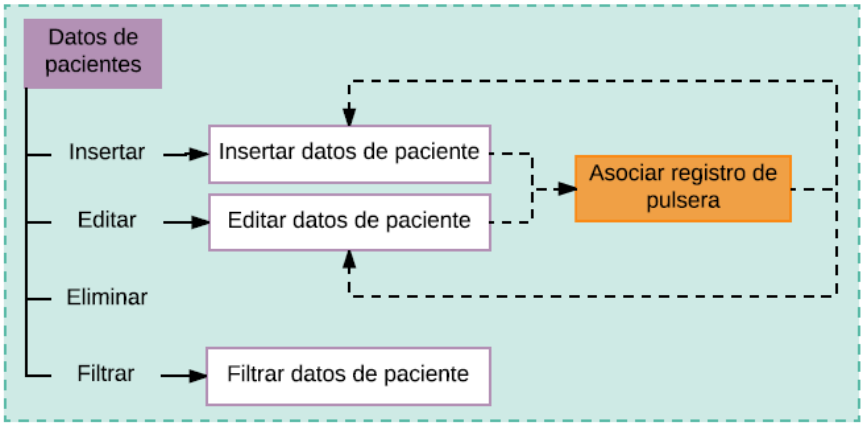


Figura 2.26. Pestaña “Datos de pacientes” de la base de datos.

- ***Datos de pacientes.*** Contiene los datos de los pacientes. Esta pestaña presentada en la Figura 2.26, le da las opciones al administrador de insertar, editar, eliminar y filtrar informaciones relacionadas a los pacientes. Las funciones insertar y editar, permiten a su vez, que se pueda asociar un dispositivo, en este caso la pulsera electrónica, a las informaciones del enfermo que se estén añadiendo o editando.

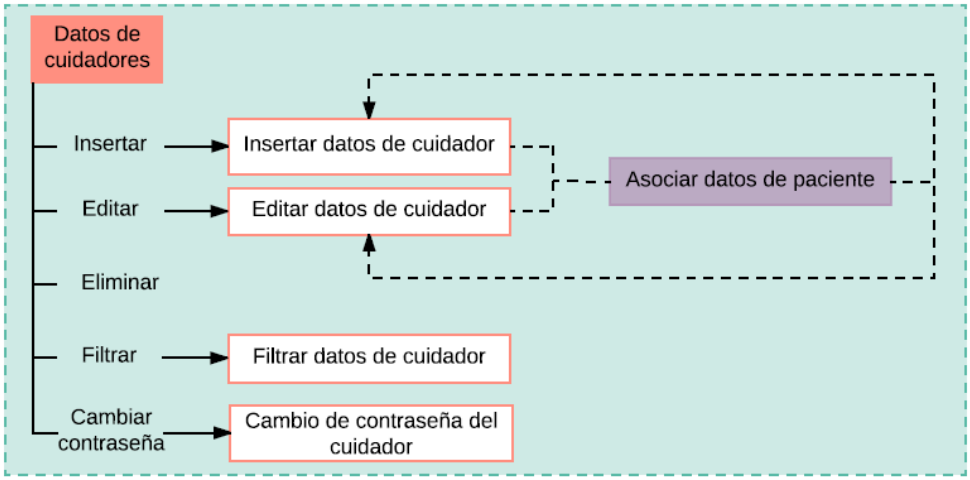


Figura 2.27. Pestaña "Datos de cuidadores" de la base de datos.

- ***Datos de cuidadores.*** Contiene detalles acerca de los custodios de los pacientes. Como se muestra en la Figura 2.27, esta pestaña le da al administrador las opciones de insertar, editar, eliminar y filtrar los datos relacionados a los cuidadores, además, facilita el cambio de contraseña de acceso de éste. Las funciones insertar y editar, permiten también, que se pueda asociar al paciente a su cargo, las informaciones que se estén añadiendo o editando. Además, filtrar, habilita la discriminación de datos con el propósito de realizar búsquedas acerca del custodio.

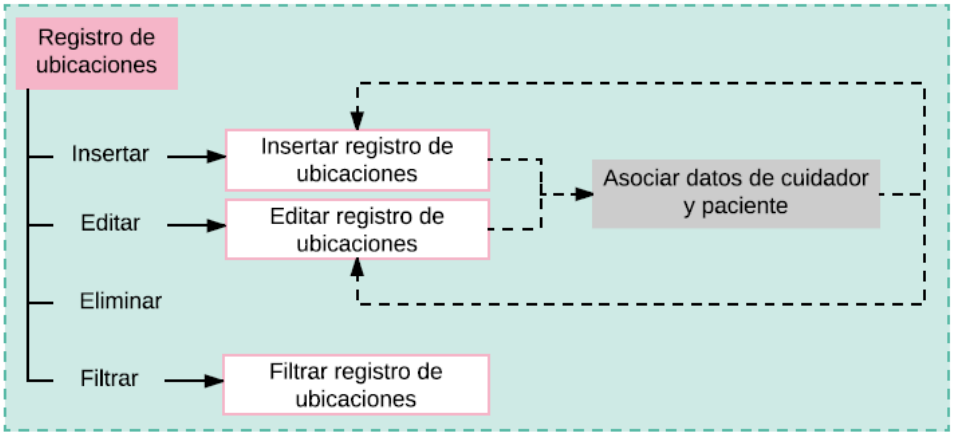


Figura 2.28. Pestaña "Registro de ubicaciones" de la base de datos.

- ***Registro de ubicaciones.*** Contiene los datos relacionados a la ubicación del paciente, y sus zonas de seguridad predeterminadas. Al ingresar a éste, se crea otro campo adyacente para almacenar los resultados de los análisis de las ubicaciones. Esta pestaña presenta al administrador las opciones de insertar, editar, eliminar y filtrar los registros de ubicaciones de los enfermos obtenidos del dispositivo localizable que poseen. Las funciones insertar y editar, permiten que se agreguen o modifiquen los datos de las localizaciones de las áreas de resguardo, respectivamente. Además, facilita la eliminación y la búsqueda de dichos registros mediante la selección de eliminar o filtrar, en el mismo orden (véase Figura 2.28).

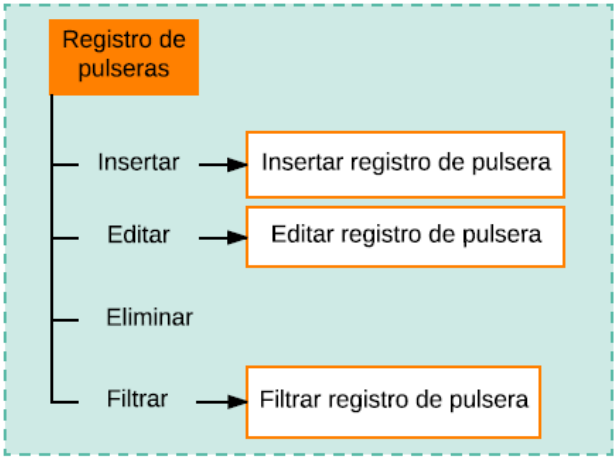


Figura 2.29. Pestaña "Registro de pulseras" de la base de datos.

- ***Registro de pulseras.*** Contiene los datos de las pulseras. Esta pestaña (ver Figura 2.29) presenta al administrador, las opciones de insertar, editar, eliminar y filtrar los registros de los dispositivos localizables que se posean. Las funciones insertar y editar, permiten que se agreguen o modifiquen los datos de las pulseras electrónicas en vigencia, respectivamente. Además, facilita la eliminación y la búsqueda de dichos registros mediante la selección de eliminar o filtrar, en el mismo orden.

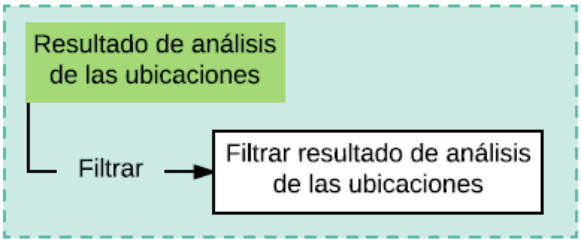


Figura 2.30. Pestaña "Resultado de análisis de las ubicaciones" de la base de datos.

- ***Resultado de análisis de las ubicaciones.*** Esta pestaña, como se muestra en la Figura 2.30, permite al administrador, filtrar datos sobre los análisis realizados para conocer la ubicación de los pacientes, presentar si están en peligro, y delimitar sus zonas de seguridad.

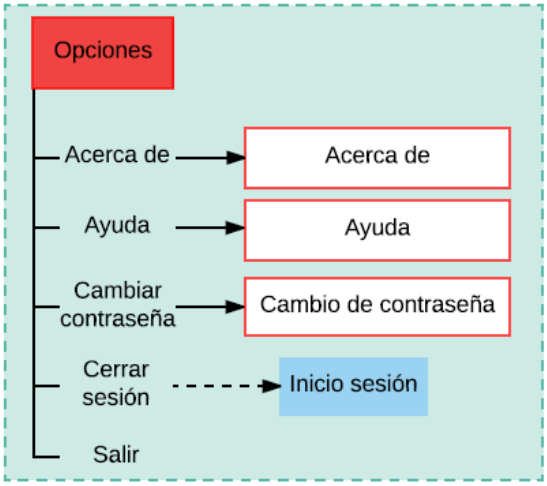


Figura 2.31. Pestaña "Opciones" de la base de datos.

- ***Opciones.*** Contiene funciones extras de la interfaz. Como se observa en la Figura 2.31, esta pestaña muestra al administrador las opciones de conocer y aprender sobre el programa, recibir asistencia para utilizarlo, cambiar su contraseña de acceso, cerrar su sesión o salir del programa.

2.4.2. Pulsera electrónica.

La pulsera electrónica es un dispositivo localizable capaz de proveer la ubicación del usuario que la porte. Como parte del sistema, corresponde a la primera fase del funcionamiento general, la cual consiste en la determinación de las coordenadas geográficas para ser enviadas periódicamente al Procesador de datos.

En principio, la pulsera presenta un retardo notable durante la búsqueda de las señales satelitales debido a las características del chip GPS-GSM (SIM808). Posteriormente, el microprocesador realiza un proceso de almacenamiento de los datos obtenidos, y se ordenan en un formato para ser escritos y enviados en mensajes SMS. Esto representa la función elemental de este dispositivo.

Para la fase de obtención como la de envío de coordenadas del paciente, se ha definido el formato de Latitud, Longitud, Tiempo, Número de Identificación de la pulsera Electrónica mostrado en el Apéndice C, Tabla C.1.

- **Diagrama de flujo de la pulsera electrónica**

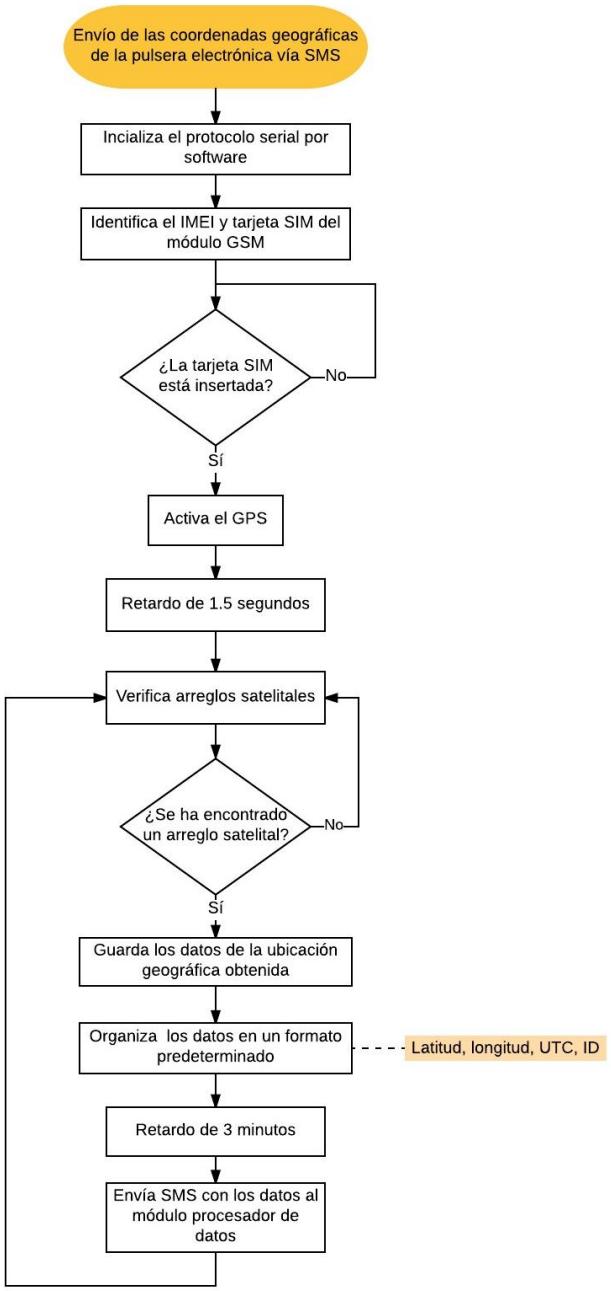


Figura 2.32. Flujo de procesos del programa de la pulsera electrónica.

En lo que respecta al prototipo de la pulsera electrónica, el conjunto de acciones que realiza es determinado por el programa del microprocesador (véase la Figura 2.32). La descripción de este proceso se muestra en el Apéndice D, sección 9.

2.5. Interfaz interactiva del usuario

2.5.1. Sistema operativo: Android.

La interfaz diseñada para el usuario (custodio del paciente) se basa en Android, un sistema operativo fundamentado en Linux utilizado principalmente en dispositivos móviles y/o portátiles. Su aspecto es de gran simplicidad y dispone de una amplia gama de software de programación y aplicaciones con licencia libre para el usuario, lo que permite una gran aceptación por parte del público de hoy en día.[119]

2.5.2. Software: Android Studio.

Android Studio es el entorno oficial de desarrollo integrado de Android, el cual facilita el desarrollo y creación de aplicaciones para los dispositivos que utilizan este sistema operativo. A su vez, permite depurar, editar, perfilar y probar los códigos realizados.[120]

El uso de este software de programación para el diseño de la aplicación móvil del SEL, se debe principalmente a la compatibilidad con la mayoría de los dispositivos, a su flexibilidad y a la gran comunidad que aporta información sobre el sistema operativo.

2.5.3. Lenguaje de programación: Java.

Android permite la creación de aplicaciones utilizando Android Runtime (ART), un derivado de Java, suministrando las interfaces necesarias para el diseño de aplicaciones capaces de tener acceso directo a funciones de los dispositivos (cámara, multimedia, GPS, entre otros).[121]

2.5.4. Diseño de interfaz gráfica: Aplicación móvil.

El diseño de una aplicación para dispositivos móviles consiste en una interfaz gráfica elaborada con el propósito de dar opciones al custodio, para conocer la posición de un paciente determinado de forma periódica y para solicitarla cuando desee. Su función en el sistema está determinada por la recepción y envío de SMS. En la Figura 2.33, se observan las partes de la aplicación.

Figura 2.33. Módulos de la aplicación móvil.

En esta sección, las franjas con un mismo color sobre los procesos de varios diagramas, indican que son dependientes entre sí.

2.5.4.1. Opciones de usuario.

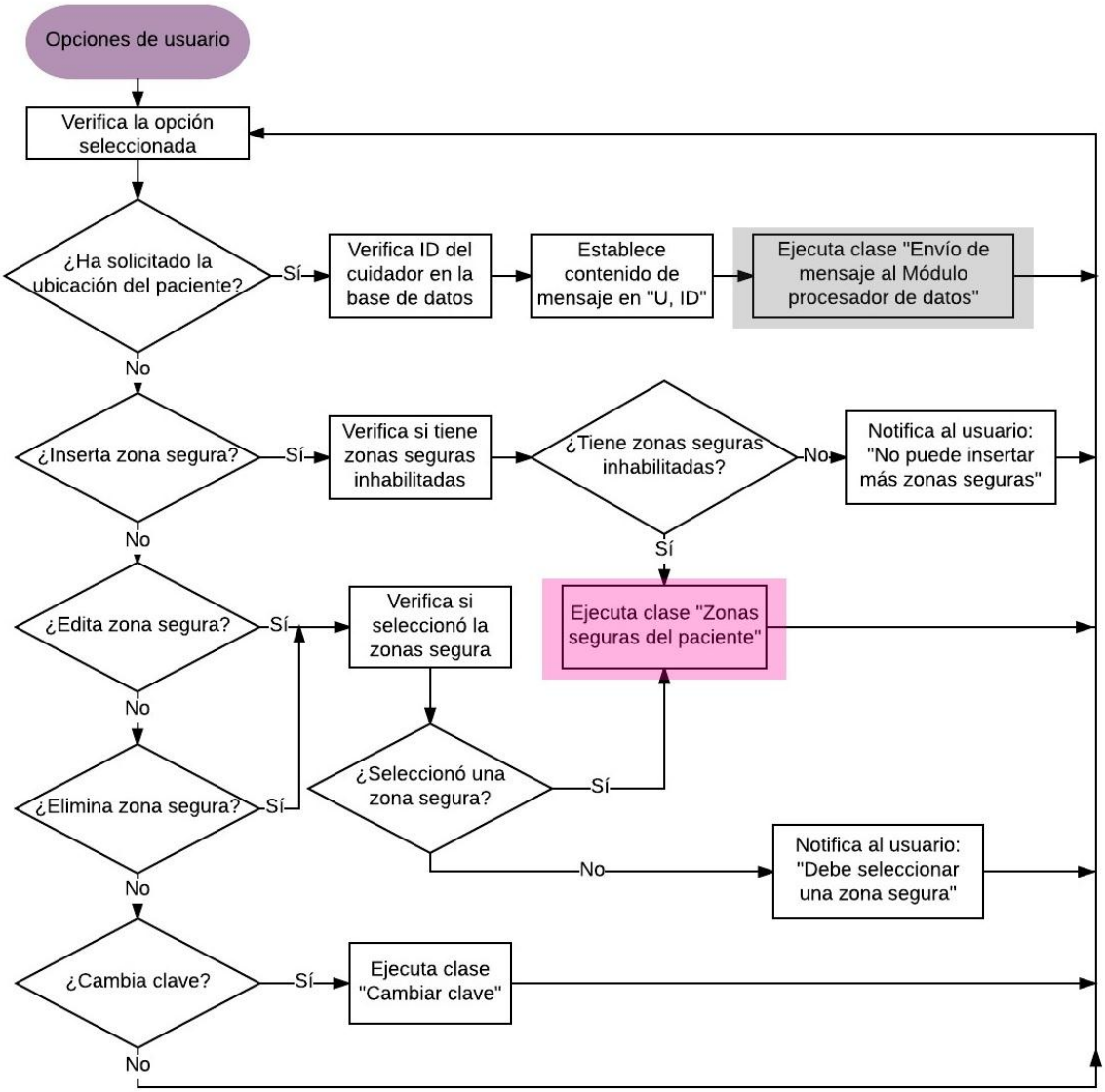


Figura 2.34. Flujo de procesos del programa "Opciones de usuario".

El flujo de procesos presentado en la Figura 2.34, se mantiene en ejecución en el dispositivo móvil. Su función es ofrecer al usuario diversas opciones del sistema. La descripción de los pasos de este flujo de procesos, se muestra en el apéndice D, sección 10.

2.5.4.2. Zonas seguras del paciente.

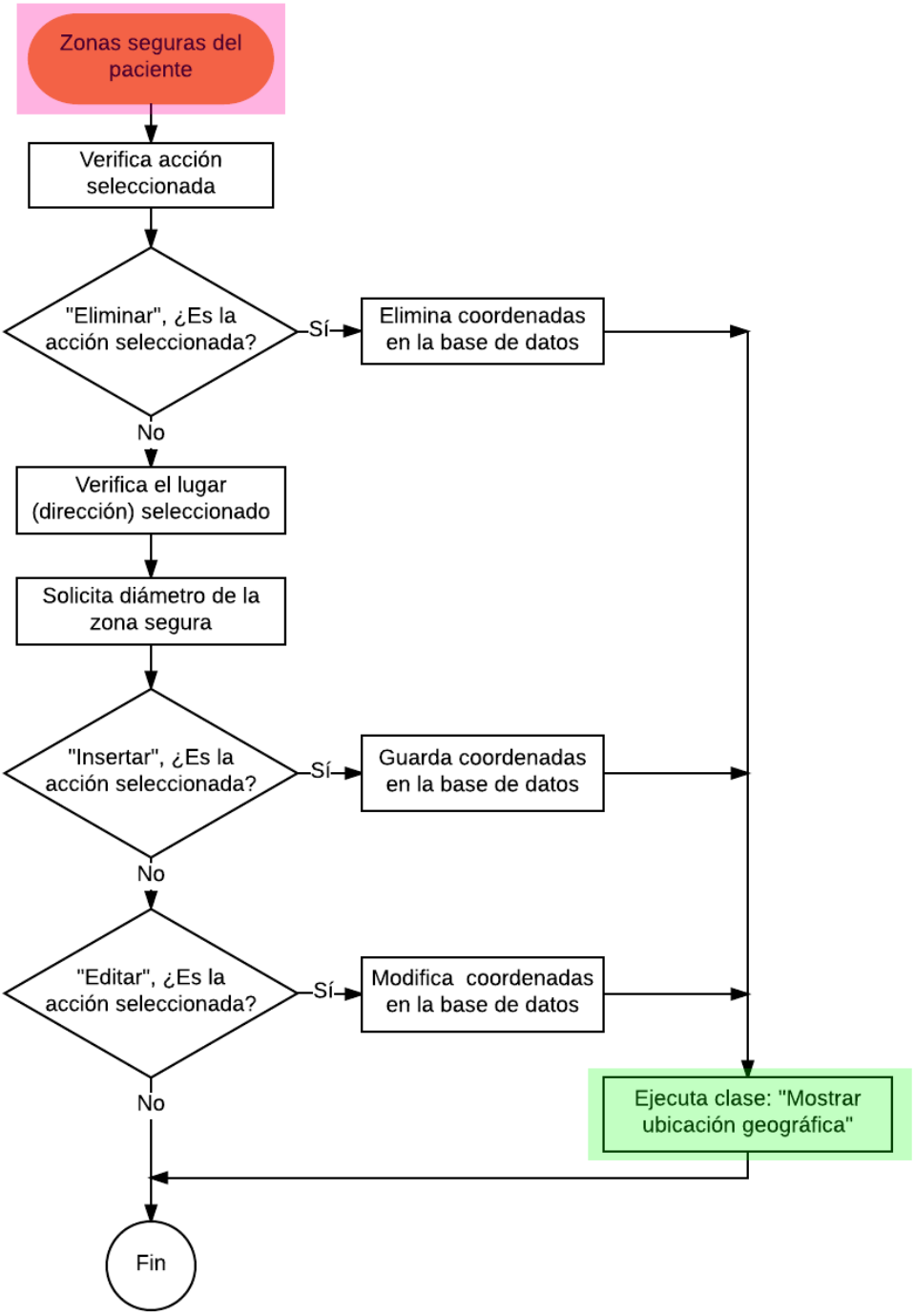


Figura 2.35. Flujo de procesos del programa "Zonas seguras del paciente".

Este programa, mostrado en la Figura 2.35, solo es ejecutado cuando se necesita realizar alguna acción con respecto a las zonas seguras (insertar, editar o eliminar). Los pasos del mismo se presentan en el Apéndice D, sección 11.

2.5.4.3. Envío de mensaje al módulo procesador de datos.

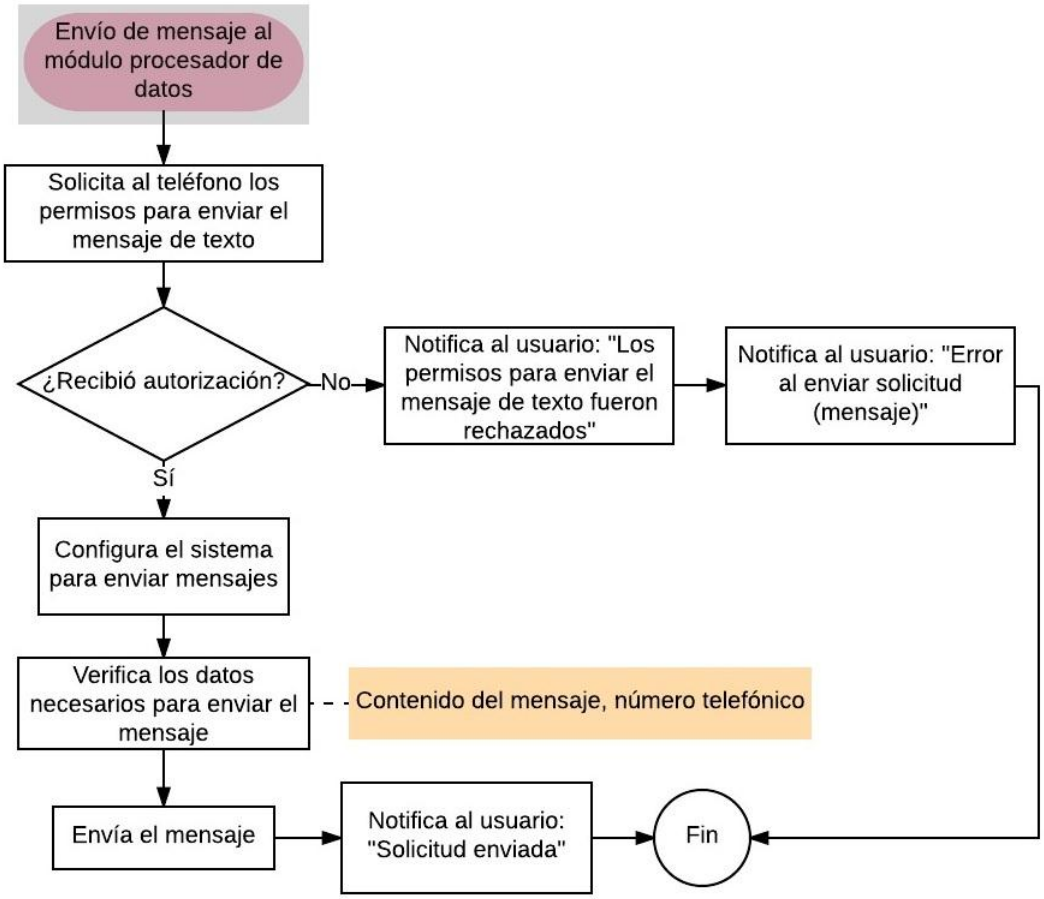


Figura 2.36. Procesos del programa "Envío de mensaje al módulo procesador de datos".

Este programa (observe la Figura 2.36) solo se ejecuta por solicitud de otro programa. Su función es interactuar con el procesador de datos mediante el envío de mensajes de textos. Los pasos del mismo se muestran en el Apéndice D, sección 12.

2.5.4.4. Mostrar ubicación geográfica.

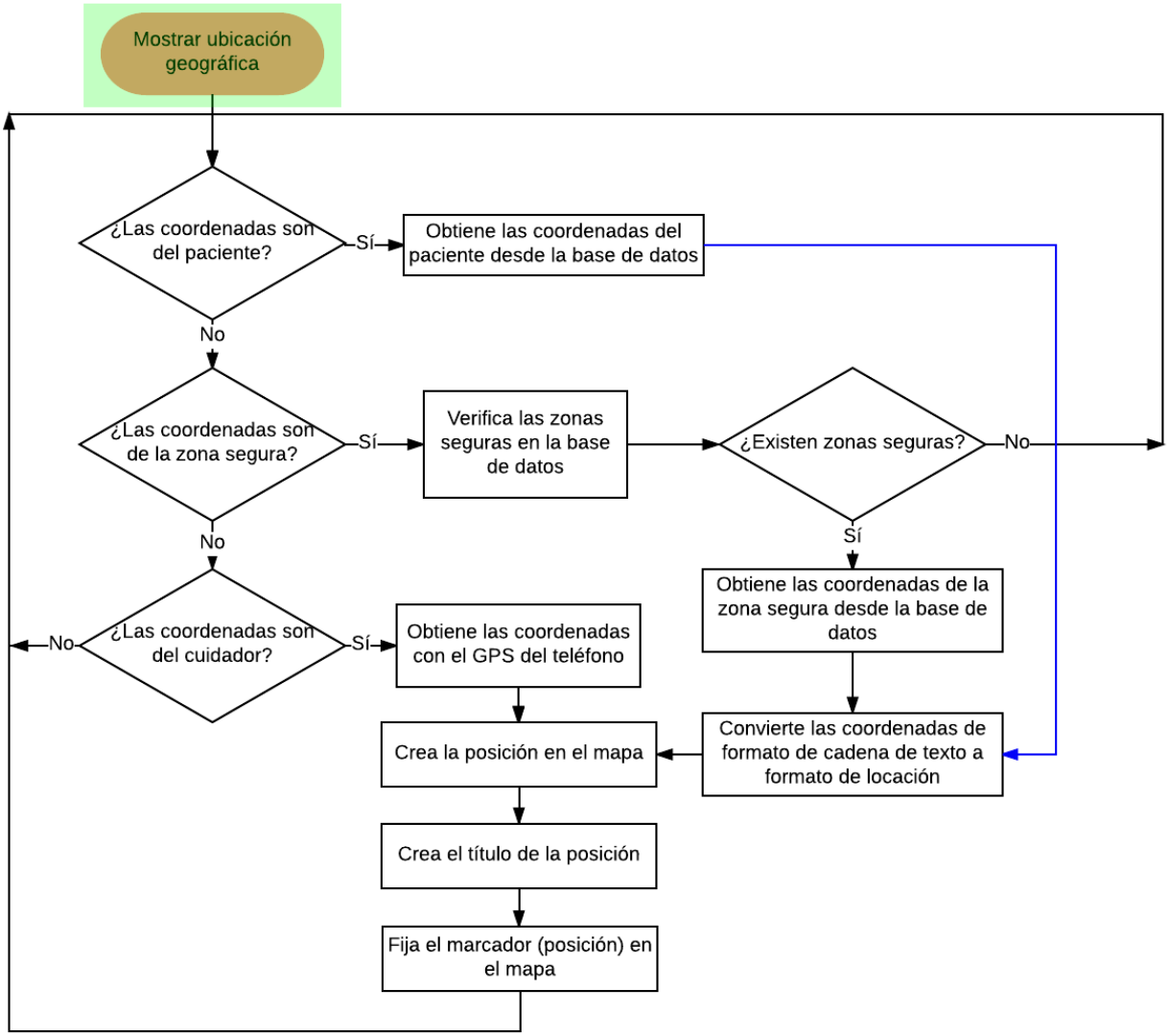


Figura 2.37. Flujo de procesos del programa "Mostrar ubicación geográfica".

El flujo de procesos de la Figura 2.37, es un programa que se mantiene en ejecución de forma constante, en el sistema de la aplicación. Su función es mostrar las ubicaciones geográficas del paciente, cuidador y las zonas seguras. Los pasos de este programa se muestran en el Apéndice D, sección 13.

2.5.4.5. Recepción de mensajes del módulo procesador de datos.

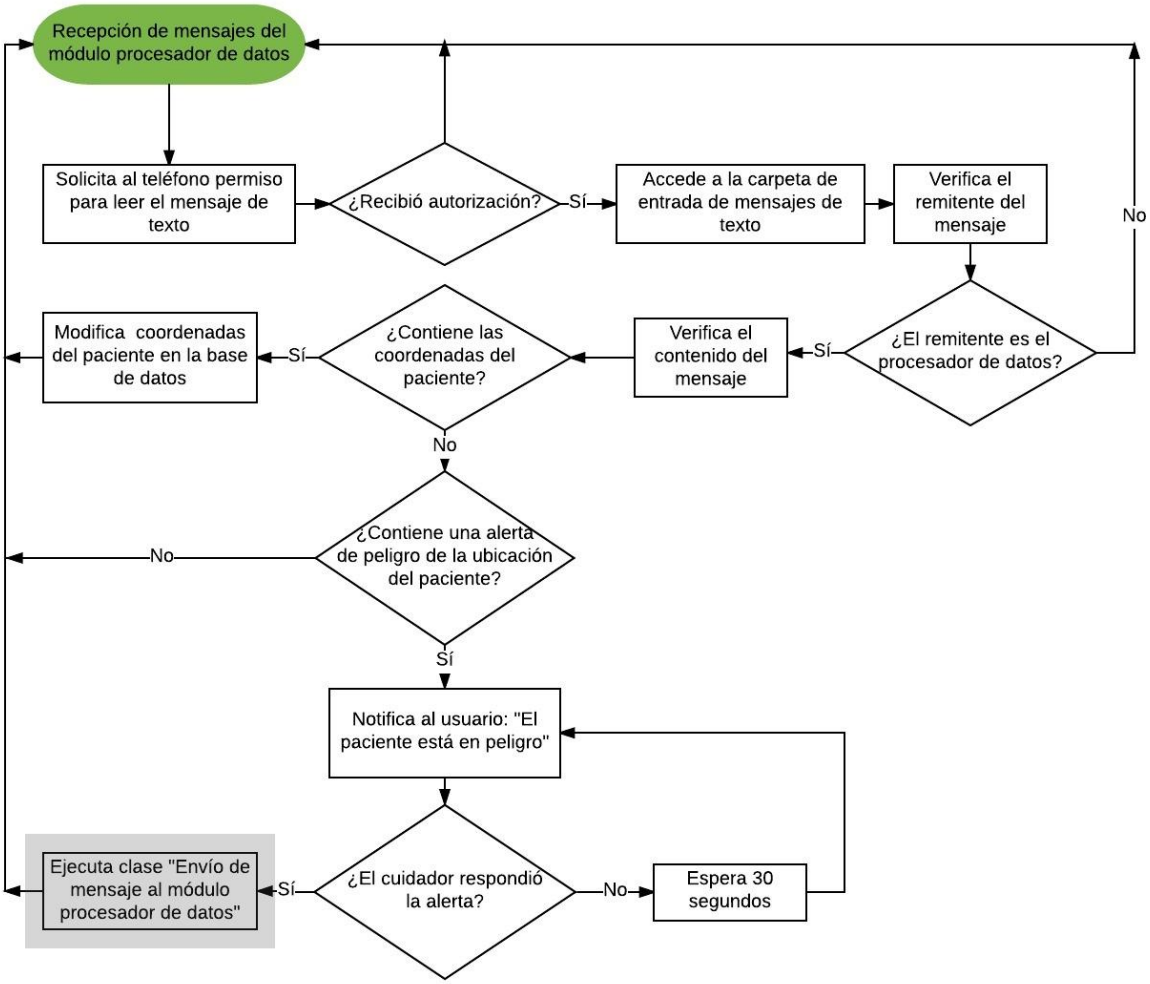


Figura 2.38. Flujo de procesos del programa "Recepción de mensajes del módulo procesador de datos".

Este programa (véase la Figura 2.38) se ejecuta constantemente en el sistema de la aplicación. Su función es interactuar con el procesador de datos al recibir los mensajes de textos provenientes de éste. Observe los pasos de estos procesos en el Apéndice D, sección 14.

2.6. Prototipado del sistema

2.6.1. Prototipo de la pulsera electrónica.

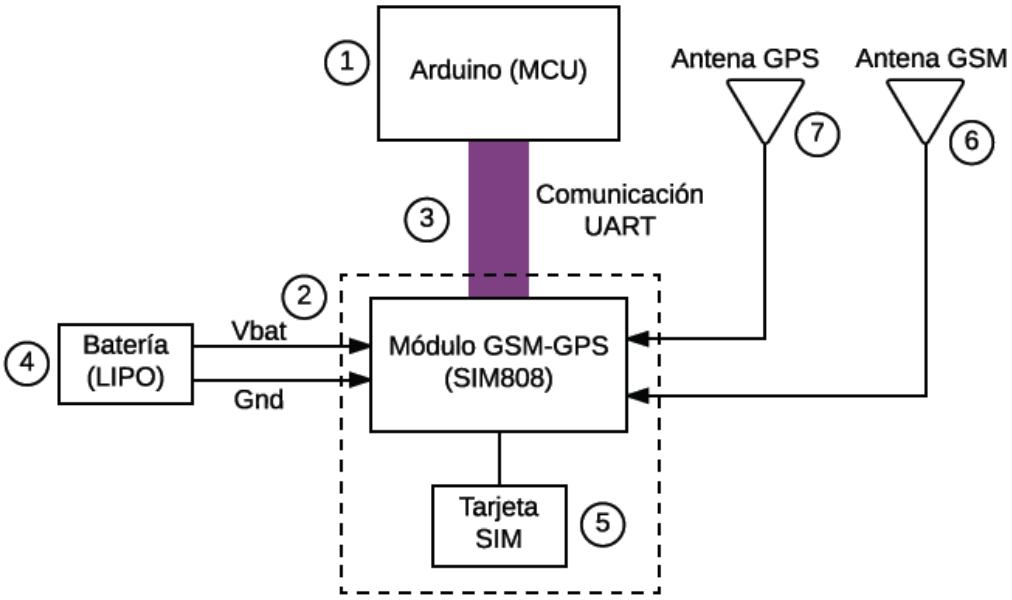


Figura 2.39. Diagrama en bloques de la pulsera electrónica.

Según se observa en la Figura 2.39, la estructura de la pulsera está conformada por:

1. El microcontrolador de la plataforma Arduino. Controla al módulo GPS-GSM para obtener su ubicación y enviar mensajes SMS periódicamente al procesador de datos, las coordenadas del dispositivo.
2. El modulo GPS-GSM. Se encarga de brindar las coordenadas dadas por el GPS y enviarlas mediante solicitudes de comandos AT por el Arduino.
3. El protocolo UART. Es necesario para controlar el módulo GSM-GPS y sus parámetros son la tasa de baudios: 115,200 y el tamaño de bit: 8.
4. La batería del SIM808. Se requiere para funcionar adecuadamente y mantener estable su alimentación. Además, sirve para brindar autonomía y portabilidad al dispositivo.
5. La tarjeta SIM. Permite el envío y recepción de mensajes SMS en la red móvil.

6. La antena GSM. Posibilita al módulo establecer conexión con la red móvil.
7. La antena GPS. Permite al módulo encontrar un arreglo de satélites para obtener coordenadas geográficas.

2.6.1.1. Programa en Arduino.

El código realizado en la plataforma Arduino permite a la pulsera electrónica realizar todas las operaciones de control y comunicación (véase el código completo en el Apéndice E, sección 1). Las partes vitales de este programa se denotan con las siguientes líneas.

- *Esta línea de código permite la inicialización del protocolo UART por software.*

```
Serial.begin(115200);
```

- *Se verifica si el módulo GSM posee una tarjeta SIM.*

```
if (String(replybuffer) == "ERROR") { while(1) {} }
```

- *Verifica que el número ICCID de la tarjeta SIM es el asignado por el sistema.*

```
fona.getSIMCCID(replybuffer);
if (String(replybuffer) != "01234567890123456789") { while(1) {} }
```

- *Esta función se encarga de habilitar el GPS.*

```
if (!fona.enableGPS(true)) delay(1500);}
```

- *Este bloque de código permite al módulo SIM808 iniciar la búsqueda de un arreglo satelital[122] 2D o 3D cuando se ha solicitado la ubicación de la pulsera electrónica.*

```
loop:
delay(1000); stat = fona.GPSstatus();
  if (stat < 0) { Serial.println(F("Fallo en la consulta de datos")); }
  if (stat == 0) { Serial.println(F("GPS apagado")); }
```

```
if (stat == 1) { Serial.println(F("No se ha encontrado un arreglo"));
goto loop; }
```

- *Se encarga de guardar las coordenadas geográficas y el tiempo UTC[123] en la variable gpsdata. Esta registra los datos en la memoria RAM del ATMEGA328P.*

```
char gpsdata[120]; fona.getGPS(0, gpsdata, 120); int n=0;
for (int i = 2; i <= 25; i++) { message[n]= gpsdata[i]; n++;} n=24;
for (int i = 37; i <= 55; i++) { message[n]= gpsdata[i]; n++;}
```

- *Se guarda el ID de la pulsera en la posición determinada de la variable message.*

```
message[43]='0'; message[44]='0'; message[45]='0'; message[46]='0'; message[47]='1';
```

- *Proceso de envío de SMS con los datos geográficos y el ID de la pulsera.*

```
if (!fona.sendSMS(sendto, message)) { Serial.println(F("No enviado."));}
  else { Serial.println(F("Enviado.")); }
```

2.6.2. Prototipo del módulo procesador de datos.

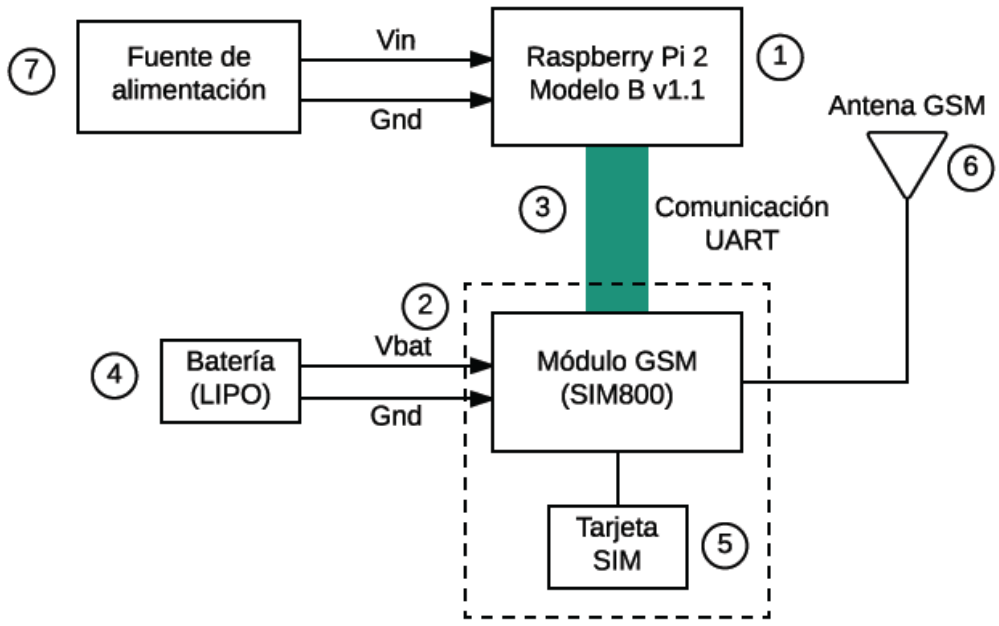


Figura. 2.40. Diagrama en bloques del módulo procesador de datos.

Según sea observa en la Figura 2.40, las partes del módulo procesador de datos son:

1. La Raspberry Pi 2 Modelo B V1.1. Se encarga de procesar los datos recibidos por el módulo GSM, el cual es controlado mediante comandos AT a través del protocolo UART. Recibe las coordenadas en SMS y las convierte en variables para guardarlas en la base de datos y procesarlas para tomar decisiones.
2. El módulo GSM. Recibe las coordenadas de la pulsera localizable a través de SMS.
3. El protocolo UART. Es necesario para controlar el módulo GSM y sus parámetros son la tasa de baudios a 115,200 y el tamaño de bit = 8.
4. La batería del SIM800. Es requerida por el SIM800 para funcionar adecuadamente.
5. La tarjeta SIM. Permite el envío y recepción de mensajes SMS en la red móvil.
6. La antena GSM. Posibilita al módulo GSM establecer conexión con la red móvil.
7. Fuente de alimentación. Suministra un voltaje regulado de 5V.

2.6.2.1. Programas modulares: Líneas de código principales.

El Procesador de datos basa su programación en el intérprete de Python (versión 2.7) para la elaboración de los diversos programas que utiliza. Estos códigos (véase el Apéndice E, secciones 2-9) conforman la mayor parte de las operaciones independientes y dependientes del sistema, los cuales, pueden ser, “Residentes” ó “De proceso por solicitud”.

2.6.2.1.1. Residentes.

- **Alerta de estado del paciente.**

De las funciones y líneas de código que componen este programa modular, la función más importante es la que decide si es pertinente emitir una notificación de peligro del paciente, y en caso de que lo sea inicia el proceso de alerta denominado ‘Decision’.

- *Función utilizada para evaluar el estado de la ubicación y el proceso de notificación.*

```
def Decision():
```

- *Realiza un recorrido del contenido de la variable "Paciente" que contiene sus datos.*

```
 for Recorrido in range(len(Paciente)):
```

- *Verifica el estado del paciente, si es igual a 'Peligro' o 'Desconocido' .*

```
        if Estado[Recorrido] == 'Peligro' and Respuesta[Recorrido] == 'Negativa':
        elif Estado[Recorrido] == 'Desconocido' and Respuesta[Recorrido] == 'Negativa':
```

- *Verifica si algún proceso (dependiendo del proceso se puede enviar un mensaje o llamar al Sistema Nacional de Atención a Emergencias y Seguridad 9-1-1) ha iniciado o no ha iniciado.*

```
            if Proceso[Recorrido] == 'N/A':
            elif Proceso[Recorrido] == 'APLICACION':
                os.system('sudo python Enviar_Mensajes ' +
                Telefono[Recorrido].encode('cp1252') + ' ' + Mensaje.replace(" ",
                "~").encode('cp1252'))
            elif Proceso[Recorrido] == 'MENSAJE DE TEXTO':
                os.system('sudo python Realizar_Llamada ' +
                Paciente[Recorrido].replace(" ", "~").encode('cp1252'))
            elif Proceso[Recorrido] == '911':
                os.system('sudo python Realizar_Llamada ' +
                Paciente[Recorrido].replace(" ", "~").encode('cp1252'))
```

- **Análisis de coordenadas.**

De las líneas de código que componen este programa modular, la función más importante es el cálculo de la distancia entre las coordenadas actuales del paciente con las coordenadas de la zona segura mediante la fórmula de Haversine, denominada 'AnalizarDatos'.

- *Función utilizada para calcular la distancia entre la ubicación actual del paciente y cada zona segura asignada por cada cuidador.*

```
def AnalizarDatos(CoordActLat, CoordActLon):
```

- *Se convierten los datos de las coordenadas en números decimales.*

```
lat1= (float(CoordActLat)* (math.pi))/180
lon1= (float(CoordActLon)* (math.pi))/180
```

- *Determina la cantidad de zonas que existen.*

```
Cantidad = len(DistanciasZonas)
```

- *Se convierten los datos de las coordenadas en radianes.*

```
lat2 = (float(Coordenadas[RecorridoCoord]) * (math.pi))/180
lon2 = (float(Coordenadas[RecorridoCoord+1]) * (math.pi))/180
```

- *Realiza la operación para obtener "a".*

```
a=(math.pow((math.sin(diflat/2)),2)) + (math.cos(lat1)) * (math.cos(lat2)) *
(math.pow((math.sin(diflon/2)),2))
```

- *Realiza la operación para obtener "c" y la operación para obtener la distancia entre ambas coordenadas, luego las convierte a kilómetros.*

```
c = 2 * math.atan2(math.sqrt(a),math.sqrt(1-a))
d = r*c
d = d*1000
```

- *Determina si la distancia está dentro de la zona segura.*

```
if (float(d) < float(DistanciasZonas[Recorrido])):
    Resultado[Recorrido]="Seguro"
```

- *Se cambia la condición del paciente a "Seguro" o a "Peligro" dependiendo de la distancia calculada.*

```
        if ("Seguro" in Resultado) == True:
            ResultFinal = "Seguro"
        else:
            ResultFinal = "Peligro"
```

- **Actualizar tiempo.**

De las funciones y partes que componen este programa modular, la función más importante es la que determina (calcula) la diferencia de tiempo entre dos fechas denominada 'Determinar Tiempo'.

- *Función utilizada para calcular la diferencia de tiempo entre dos fechas.*

```
def DeterminarTiempo():
    Fecha2 = datetime.now()
```

- *Realiza un recorrido del contenido de la variable Fecha1 (la cual contiene la última actualización de la coordenada de cada paciente.*

```
for Analisis in range(len(Fecha1)):
    Fecha1[Analisis] =datetime.strptime(Fecha1[Analisis], "%d/%m/%Y %H:%M:%S")
    Diferencia = Fecha2 - Fecha1[Analisis]
```

- *Comprueba que el tiempo sea menor o igual a cuatro (4) minutos (Días = 0 y Minutos <=4), también verifica que el estado inicial del paciente sea diferente de peligro o 'N/A'. Si se cumplen lo anterior, actualiza el estado a 'seguro'.*

```
if Diferencia.days == 0 and (Diferencia.seconds/60) <= 4 and EstadoAnterior[Analisis] != "Peligro" and EstadoAnterior[Analisis] != "N/A":
```

- *Comprueba que el tiempo sea mayor de cuatro (4) minutos (Dias >= 1 y Minutos >4), también verifica que el estado inicial del paciente sea diferente de desconocido o 'N/A'. Si se cumple lo anterior, actualiza a 'Desconocido'.*

```
elif (Diferencia.days > 0 or (Diferencia.seconds/60) > 4) and EstadoAnterior[Analisis] != "Desconocido" and EstadoAnterior[Analisis] != "N/A":
```

- *Función utilizada para actualizar los datos en la tabla "Resultado de análisis de las ubicaciones".*

```
EditarEstado(Analisis, Campos)
```

- **Recepción de mensaje.**

De las funciones y códigos que componen este programa, la fase más importante es la que recibe el mensaje de texto y determina su proveniencia. (cuidador o pulsera).

- *Habilita el puerto serial.*

```
ser = serial.Serial('/dev/ttyAMA0', 115200)
```

- *Activa el modo lectura de datos del módulo GSM.*

```
ser.write("AT+CMGF=1\r")
```

Lee el puerto serial, y guarda los datos en la variable indicada. (Contenido del mensaje).

```
response_3 = ser.readline()
```

- *Obtiene la longitud del contenido del mensaje de texto.*

```
tam = len(response_3)
    if tam > 9:
```

- *Lee y guarda los datos del puerto serial, luego ejecuta "Mensaje_Pulsera.py".*

```
Mensaje = response_3[0:45]
os.system('sudo python Mensaje_Pulsera.py ' + Mensaje)
    if tam <=9:
```

- *Obtiene las coordenadas de los datos del SMS , luego ejecuta "Mensaje_Cuidador.py".*

```
 Mensaje = response_3[0:8]
 os.system('sudo python Mensaje_Cuidador.py ' + Mensaje)
```

2.6.2.1.2. De proceso por solicitud.

- **Mensaje cuidador.**

Este programa consiste básicamente en procesar un SMS recibido correspondiente a un cuidador. Posteriormente, la información es guardada en la base de datos.

- *Separa los datos del mensaje y obtiene el identificador del cuidador.*

```
Texto = Mensaje.split(',')[0]
ID = Mensaje.split(',')[1]
```

- *Verifica si el contenido del mensaje es igual a la letra 'U' o la letra 'P'.*

```
if Texto == 'U' or Texto == 'P':
```

- *Se obtiene el teléfono del cuidador y la ubicación actual del paciente desde la base de datos, luego se envía un mensaje al cuidador con los datos solicitados.*

```
Datos = CargarDatos()
```

os.system('sudo python Enviar_Mensajes ' + Datos[0] + ' ' + Datos[1].replace(" ", "~"))

- *Verifica si el contenido del mensaje es igual a la letra 'S'.*

  elif Texto == 'S':

- *Verifica si el contenido del mensaje es igual a la letra 'R'.*

  elif Texto == 'R':

- *Función utilizada para cambiar la respuesta a 'positiva' en la tabla "Resultado de análisis de las ubicaciones".*

  ModificarEstado()

- **Mensaje pulsera.**

De las partes que componen este programa, los bloques de código más importantes son los que identifica el mensaje del paciente y guardan la información en la base de datos.

- *Obtiene la latitud de la coordenada de la pulsera.*

  latitude = Mensaje.split(',')[0]

- *Obtiene la longitud de la coordenada de la pulsera.*

  longitude = Mensaje.split(',')[1]

- *Obtiene la fecha de la coordenada de la pulsera.*

  Fecha = Mensaje.split(',')[2]

- *Obtiene el identificador de la pulsera.*

  ID = Mensaje.split(',')[3]

- *Aplica formato a la fecha.*

  Fecha = Fecha[6:8] + '/' + Fecha[4:6] + '/' + Fecha[0:4] + ' ' + Fecha[8:10] + ':' + Fecha[10:12] + ':' + Fecha[12:14]

- *Comando a ejecutar en la tabla de la base de datos.*

```
comando = "update " + tabla + " set " + value + " where Id_Dispositivo='" + ID + "'"
```

- *Establece la conexión con la base de datos y maneja un cursor para ejecutar comandos.*

```
with sqlite3.connect('Tesis.s3db') as con:
    cursor = con.cursor()
    cursor.execute(comando)
    cursor.close()
con.close()
```

- **Envío de mensaje.**

De las funciones y partes que componen este programa, las líneas de código más importantes son las que envían el mensaje de texto.

- *Habilita el puerto serial.*

```
ser = serial.Serial('/dev/ttyAMA0', 115200)
```

- *Escribe un comando con el número telefónico al cual se le enviará el mensaje de texto.*

```
ser.write("AT+CMGS=" + "" + Telefono + "" + "\r")
```

- *Envía el mensaje de texto.*

```
ser.write(Mensaje + '\x1a')
time.sleep(5)
```

- *Cierra el puerto serial.*

```
ser.close()
```

- **Realización de llamada.**

De las funciones y partes que componen este programa, las líneas de código más importantes son los que ejecutan la llamada telefónica:

- *Habilita el puerto serial.*

```
ser = serial.Serial('/dev/ttyAMA0', 115200)
```

- *Establece el número telefónico al que se llamará.*

```
ser.write("ATDXXXXXXXXXX;\r")
```

Comando que cuelga la llamada (utilizado exclusivamente para la prueba funcional).

```
ser.write("ATH\r")
```

- *Lee las respuestas del módulo GSM.*

```
res = ser.readline()
```

- *Cierra el puerto serial.*

```
ser.close()
```

2.6.3. Prototipo de la aplicación móvil.

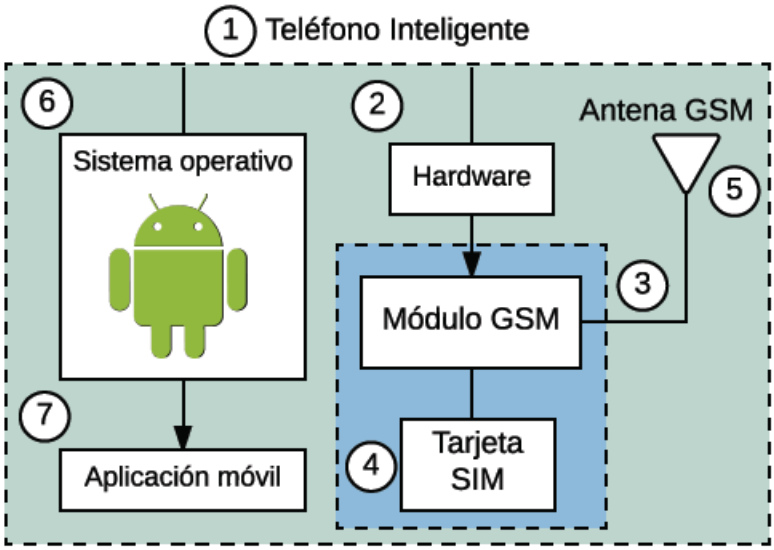


Figura 2.41. Diagrama en bloques de un teléfono móvil. *Este bloque únicamente, indica el lugar que ocupa la aplicación en un teléfono móvil y no forma parte del sistema.*

Según sea observa en la Figura 2.41, un teléfono inteligente con sistema operativo Android está constituido por las siguientes partes:

1. Dispositivo móvil.
2. Conjunto de periféricos y partes físicas que conforma el dispositivo.
3. El módulo GSM. Recibe las coordenadas de la pulsera localizable en mensajes SMS.
4. La tarjeta SIM. Permite el envío y recepción de mensajes SMS.
5. La antena GSM. Posibilita al módulo GSM establecer conexión con la red móvil.
6. Plataforma de aplicaciones y software del dispositivo.
7. Aplicación para hacer solicitudes al procesador de datos y recibir notificaciones.

En el Apéndice E, secciones 10-13 se muestran las líneas de códigos más relevantes de la aplicación móvil.

3. VERIFICACIÓN Y CONTRASTE DEL DISEÑO

3.1. Resultados

3.1.1. Prueba funcional del sistema.

3.1.1.1. Condiciones iniciales.

En esta fase de prueba, las partes del sistema realizan determinados procesos de inicialización que les permiten preparar tanto su nivel físico como el entorno virtual en pos de mantener su funcionamiento.

3.1.1.1.1. Hardware.

3.1.1.1.1.1. Módulo procesador de datos.

El módulo procesador de datos (ver Figura 3.1) es el primer elemento del sistema que inicia sus operaciones, debido a que es el encargado de evaluar las respuestas provenientes de la pulsera electrónica y de la aplicación del cuidador.

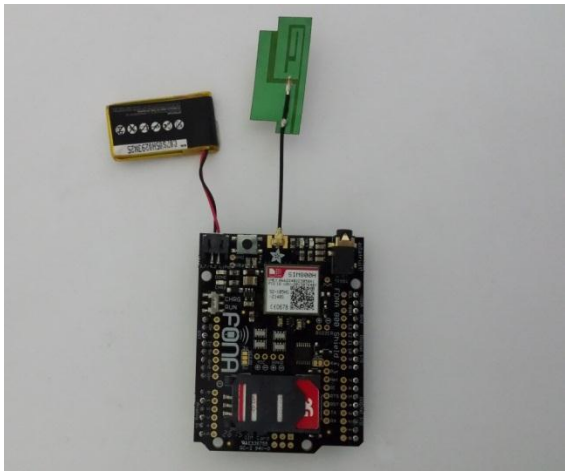


Figura 3.1. Ensamblaje del Módulo SIM800.

En primer lugar, se coloca la antena GSM, la batería y la tarjeta SIM al módulo SIM800. Luego, la Raspberry Pi debe enlazarse con éste módulo. Las conexiones (ver Figura 3.2) se realizan desde la Raspberry Pi hasta el módulo SIM800 de la siguiente forma:

4. Desde el pin GND hasta el pin GND.
5. Desde el pin 10 (R_X) hasta el pin T_X.
6. Desde el pin 8 (T_X) hasta el pin R_X.
7. Desde el pin 5V hasta el pin 5V.

Posteriormente, se conectan los periféricos teclado y ratón a puertos USB, un Monitor al puerto HDMI[124] y el cable de alimentación de la Raspberry Pi. Véase la Figura 3.3.

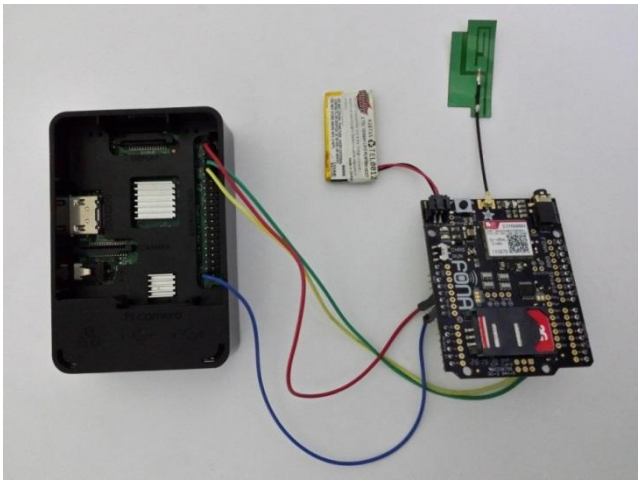


Figura 3.2. Conexión Raspberry Pi – Módulo SIM800.

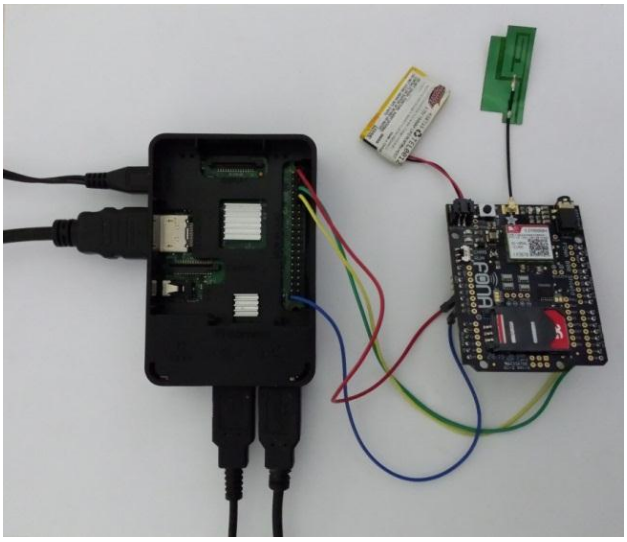
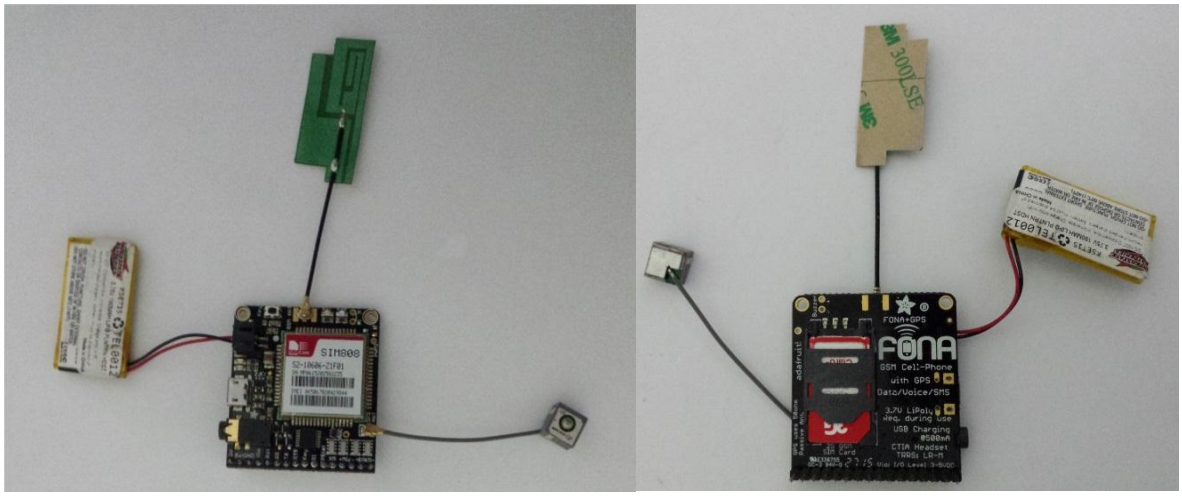


Figura 3.3. Ensamblaje del módulo procesador de datos.

3.1.1.1.1.2. Pulsera electrónica.

La pulsera electrónica es el dispositivo portátil del sistema. El primer paso para preparar el hardware es conectar las antenas GPS y GSM, colocar la tarjeta SIM y conectar la batería al módulo SIM808. Observe la Figura 3.4.

Luego, la placa Arduino debe enlazarse con éste módulo. Por otra parte, el módulo SIM808 también requiere la interconexión del pin V_{io} al pin V_{bat} en el mismo dispositivo. Ver conexión en la Figura 3.5.



a) b)

Figura 3.4. Ensamblaje del módulo SIM808. a) Parte delantera. b) Parte trasera.

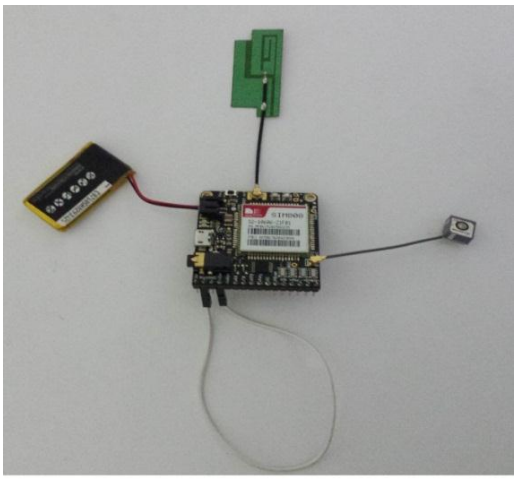


Figura 3.5. Conexión del pin V_{io} al pin V_{bat} en el módulo SIM808.

Se deben realizar además, conexiones desde la placa de Arduino hasta el módulo SIM808 de la siguiente forma:

8. Desde el pin GND hasta el pin GND.
9. Desde el pin 3 ($R_{X)}$ hasta el pin T_X.
10. Desde el pin 2 (T_X) hasta el pin R_X
11. Desde el pin 5V hasta el pin 5V.

En la Figura 3.6, se muestra el prototipo funcional de la pulsera electrónica.

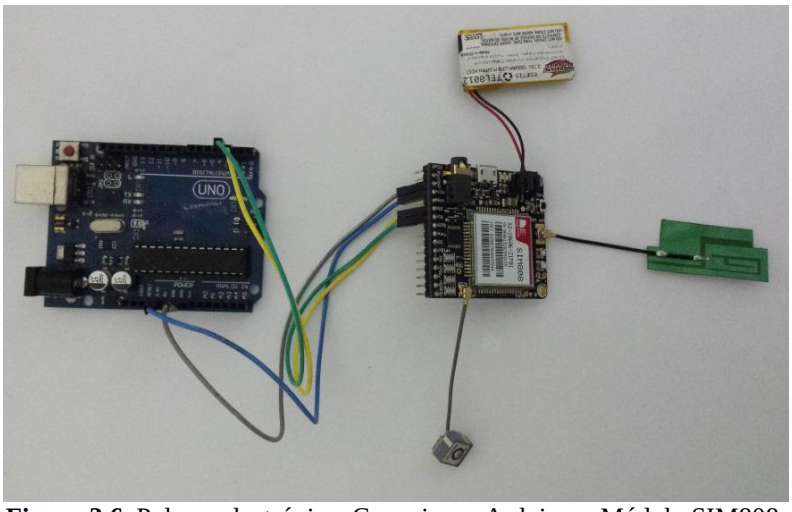


Figura 3.6. Pulsera electrónica: Conexiones Arduino – Módulo SIM808.

El siguiente paso es encender el módulo SIM808 presionando por 3 segundos el botón Key, y luego, se conecta el Arduino a un computador, que no forma parte del Sistema Localizador de pacientes con Alzheimer, pero que servirá para mostrar los procesos que se realizan en la pulsera electrónica. Ver en la Figura 3.7.

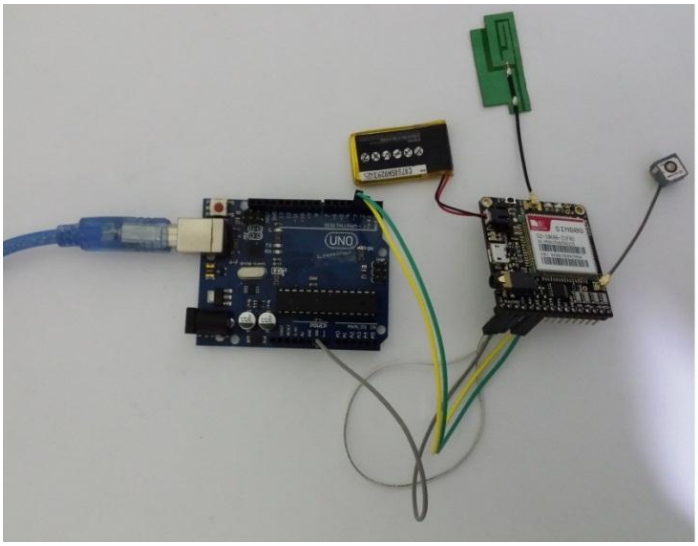


Figura 3.7. Pulsera electrónica conectada vía cable USB a una computadora.

3.1.1.1.1.3. Teléfono móvil del cuidador.

Para la ejecución de la aplicación, el dispositivo móvil a utilizarse en la prueba funcional debe cumplir con los siguientes requisitos:

1. Que el dispositivo disponga de una tarjeta SIM activa y registrada en una red móvil.
2. Dispositivo encendido con el sistema operativo Android iniciado.
3. Conexión a la red móvil.

3.1.1.1.2. Software.

3.1.1.1.2.1. Módulo procesador de datos.

Con las condiciones iniciales de hardware aplicadas, el módulo procesador de datos se alimenta con una fuente de voltaje y enciende inmediatamente. La siguiente etapa del módulo procesador de datos corresponde con la ejecución de los programas residentes. Para continuar, se inicia la terminal de línea de comandos del sistema operativo Raspbian (como se muestra en la Figura 3.8). Esta se puede ejecutar haciendo click en el ícono de *Terminal* de la barra de tareas o accediendo a *Menú de aplicaciones/Accesorios/Terminal*.

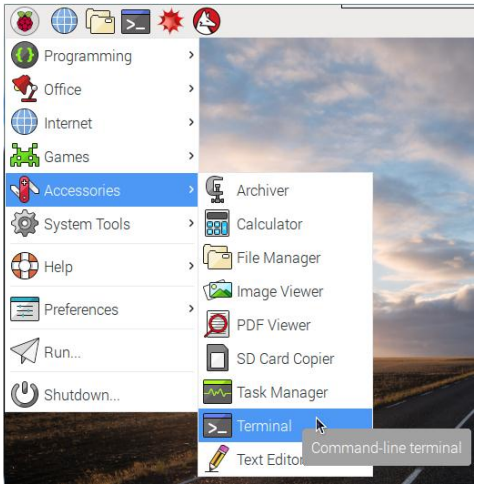


Figura 3.8. Ejecución de la terminal de línea de comandos en Raspbian (Jessie).

Con la terminal en proceso, se procede a escribir las órdenes que permiten ejecutar programas. En el Procesador de Datos, los programas fueron realizados en Python por lo que, las líneas de comandos iniciarán el intérprete de este lenguaje de programación para correr los archivos "*.py*".

En primer lugar, se le indica a la terminal que los archivos se encuentran en la carpeta *RaspBerry*.

```
pi@raspberrypi:~ $ cd RaspBerry
```

Luego, se ejecutan los programas modulares residentes en cualquier orden. Estos se muestran en la Figura 3.9.

Programas modulares residentes.

```
pi@raspberrypi:~/RaspBerry $ sudo python Alerta_Estado.py
pi@raspberrypi:~/RaspBerry $ sudo python Analizar_Coordenadas.py
pi@raspberrypi:~/RaspBerry $ sudo python Actualizar_Tiempo.py
pi@raspberrypi:~/RaspBerry $ sudo python Recibir_Mensajes.py
```

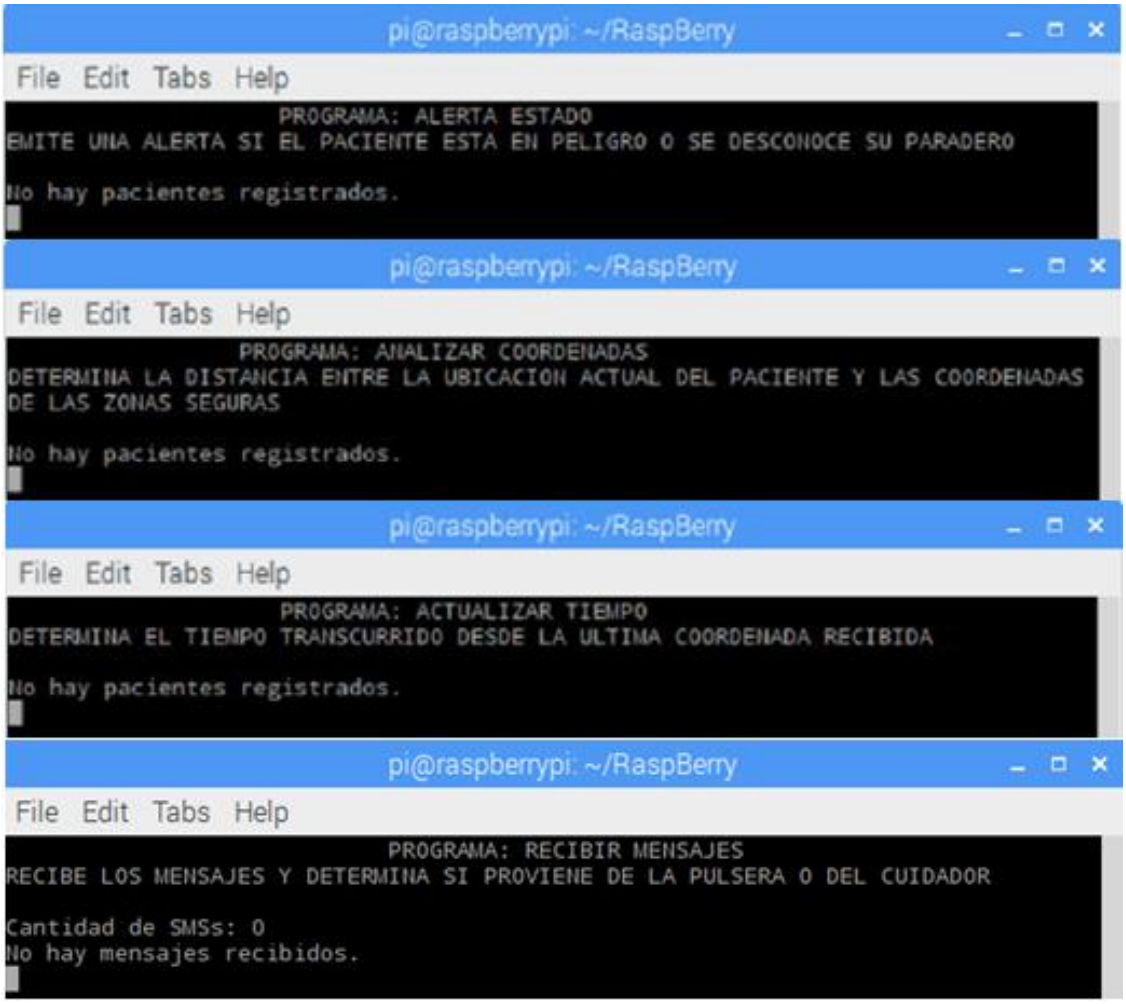


Figura 3.9. Programas modulares ejecutados en su estado inicial.

3.1.1.1.2.2. Pulsera electrónica.

Con las condiciones iniciales de hardware aplicadas, este dispositivo se encuentra listo para su funcionamiento, y se ejecuta el programa de inicialización de la pulsera electrónica. Ver Figura 3.10.

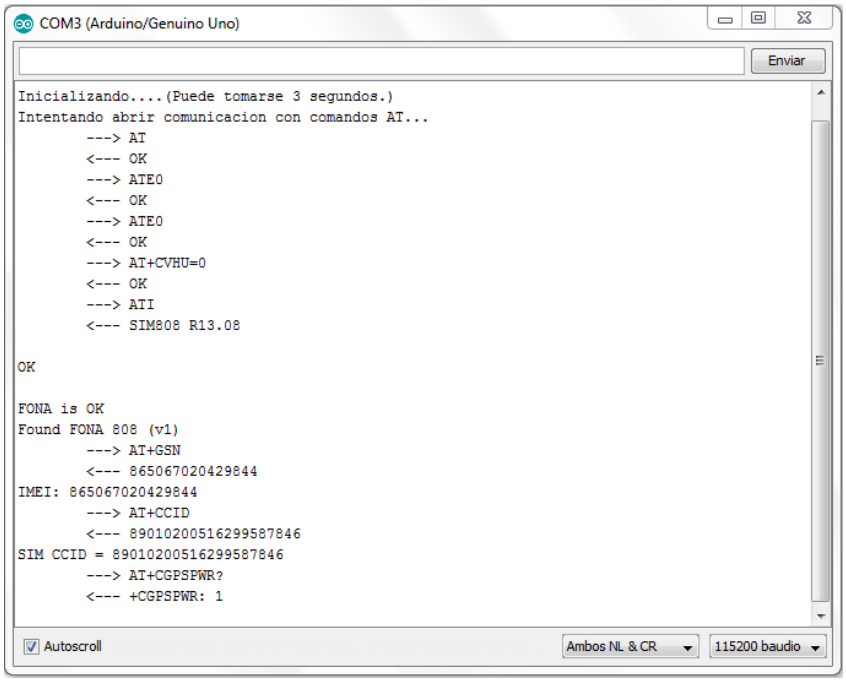


Figura 3.10. Pulsera electrónica: Inicialización del programa vista desde el "Monitor Serial" de Arduino.

El momento en que el Arduino es alimentado por el cable USB, el microcontrolador ATMEGA328P inicia el programa para comunicarse con el módulo SIM808. Posteriormente, comprueba el IMEI del dispositivo y el ICCID de la Tarjeta SIM. Luego, enciende el receptor GPS para recibir señales satelitales.

3.1.1.1.2.3. Aplicación móvil.

En la Figura 3.11, se muestra la interfaz de la aplicación de Android, la cual, es un elemento no físico del sistema que requiere ser instalado en un dispositivo móvil. El único paso para ser utilizada por el cuidador, es ejecutarla y dejarla funcionando en segundo plano.

Figura 3.11. Interfaz de la aplicación móvil.

3.1.1.2. Información de pacientes y cuidadores.

En el módulo procesador de datos es necesario disponer de información acerca de la pulsera electrónica, paciente, cuidador y zonas seguras que ofrezcan una referencia al sistema para realizar sus funciones. En primer lugar, hay que iniciar la Interfaz de administrador (observe la Figura 3.12), mediante las siguientes líneas de comando:

```
pi@raspberrypi:~$ cd RaspBerry
pi@raspberrypi:~/RaspBerry $ sudo python 1Log.py
```

Figura 3.12. Interfaz de administrador: Inicio de sesión del administrador del SEL.

El siguiente paso es escribir el nombre de usuario y la contraseña de administrador para acceder a la ventana de menú presentada en la Figura 3.13.

Figura 3.13. Interfaz de administrador: Ventana "Menú".

3.1.1.2.1. Registro de pulsera.

A continuación, se accede a la pestaña "Registro de Pulsera" y se hace click en la opción "Insertar" (observe la ventana en la Figura 3.14). Luego se llenan todos los campos de información de la Pulsera a agregar y al finalizar, se presiona el botón "Aceptar".

Figura 3.14. Interfaz de administrador: Ventana "Registro de Pulsera".

3.1.1.2.2. Registro de paciente.

El siguiente paso es registrar al paciente (se muestra esta ventana en la Figura 3.15). Primero, se selecciona la pestaña “Datos de Pacientes” y se hace click en la opción “Insertar”. Luego, se llenan todos los campos solicitados con la información del paciente. Después se presiona el botón “Pulsera” y se escoge el dispositivo que se asociará al paciente. Finalmente hacemos click en el botón “Aceptar”.

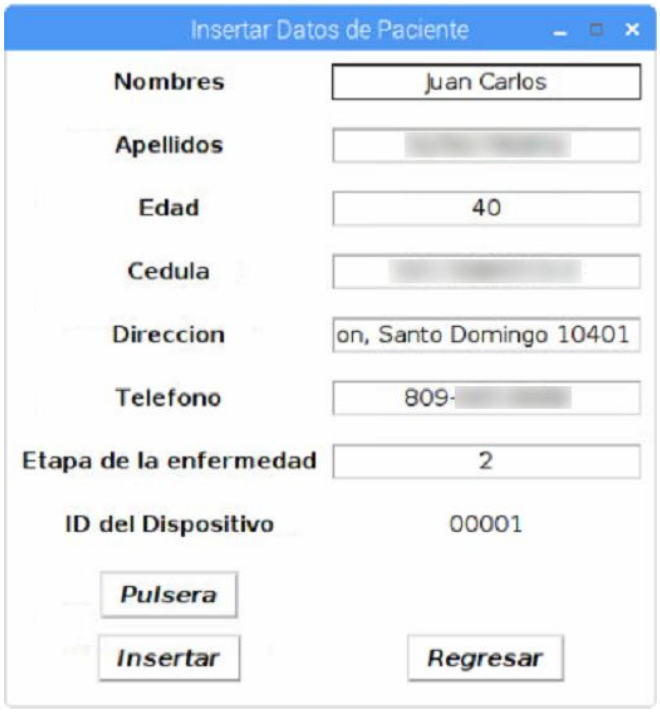


Figura 3.15. Interfaz de administrador: Ventana “Datos de Paciente”.

3.1.1.2.3. Registro de cuidador.

El registro de cuidador consiste en un proceso similar al registro de paciente, con la singularidad de que el cuidador debe registrar más datos. En primer lugar seleccionamos la pestaña “Datos de Cuidador” y hacemos click en la opción “Insertar”, donde se completarán los campos con información del cuidador, al finalizar, presionamos el botón “Paciente” y seleccionamos los datos del paciente que se asociará al cuidador. Finalmente, hacemos click en el botón “Aceptar”. Véase la Figura 3.16.

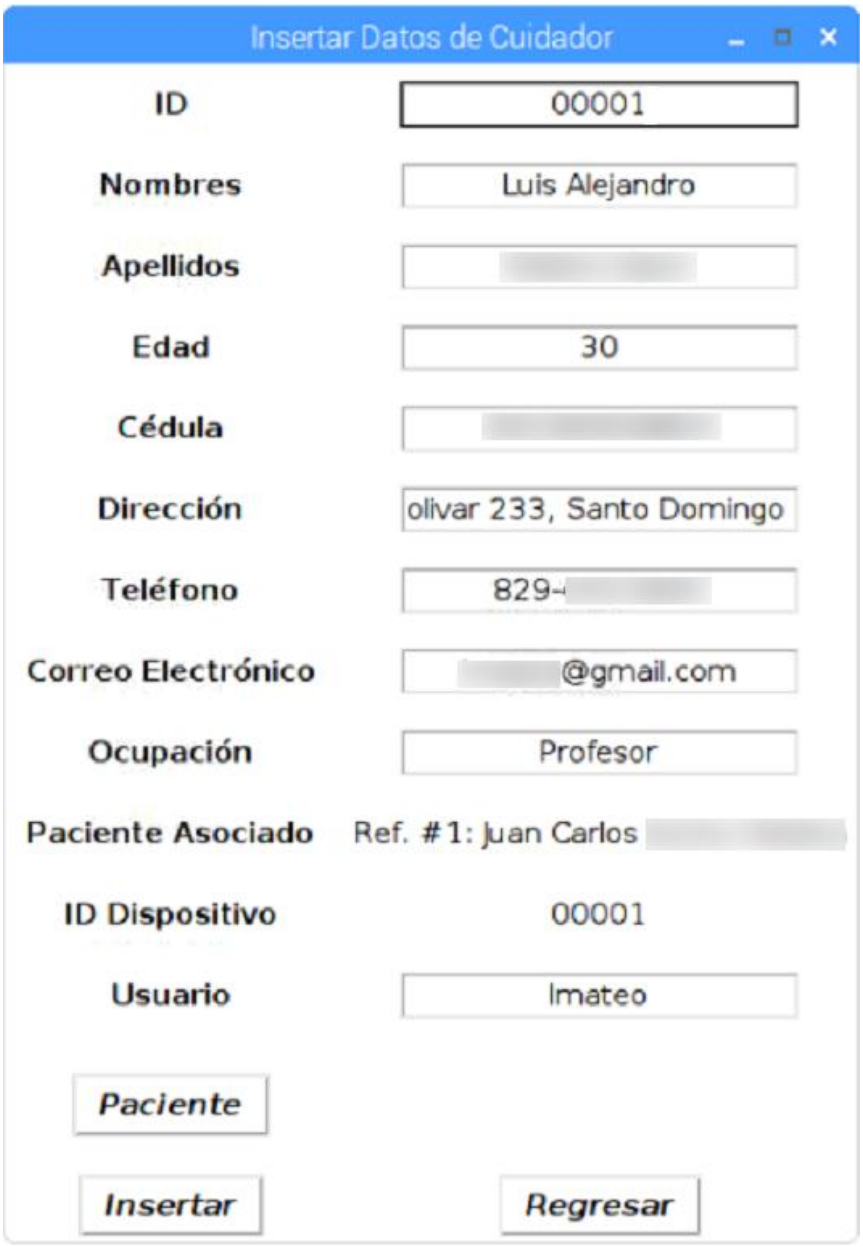


Figura 3.16. Interfaz de administrador: Ventana "Datos de Cuidador".

3.1.1.2.4. Asignación de zonas seguras.

Como parte de la información del paciente, es necesaria la asignación de las zonas seguras en las que este debe encontrarse. Para esto, se selecciona la pestaña "Registro de Ubicaciones" y se hace click en la opción "Insertar", luego, se completan los campos requeridos y se presiona el botón "Cuidador/Paciente". Posteriormente, se seleccionan los

datos del cuidador que asignará las zonas seguras y se presiona el botón “Aceptar”. Ver Figura 3.17.

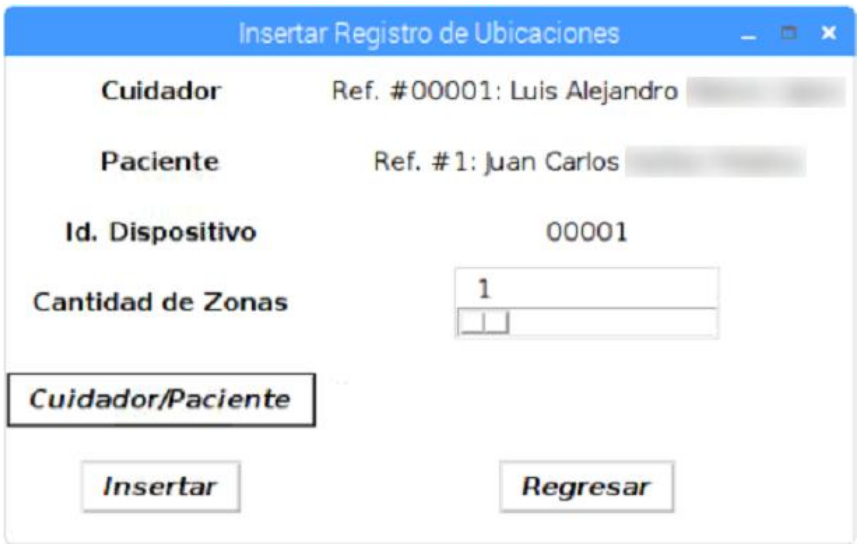


Figura 3.17. Interfaz de administrador: Ventana “Registro de Ubicaciones”.

3.1.1.3.Funcionamiento de la pulsera electrónica.

La pulsera electrónica se encuentra preparada para el envío de información geográfica vía SMS. El primer paso que debe realizar, consiste en encontrar señales GPS que permitan hacer el cálculo de distancias entre satélites de referencia para obtener su posición geográfica, este proceso puede tardar un promedio de cuatro (4) minutos. Durante este proceso, el “Monitor Serial” arroja el estado de “No Fix” (como se observa en la Figura 3.18), cuando logre encontrar los arreglos se presentarán los estados “2D Fix” o “3D Fix” (ver Figura 3.19).

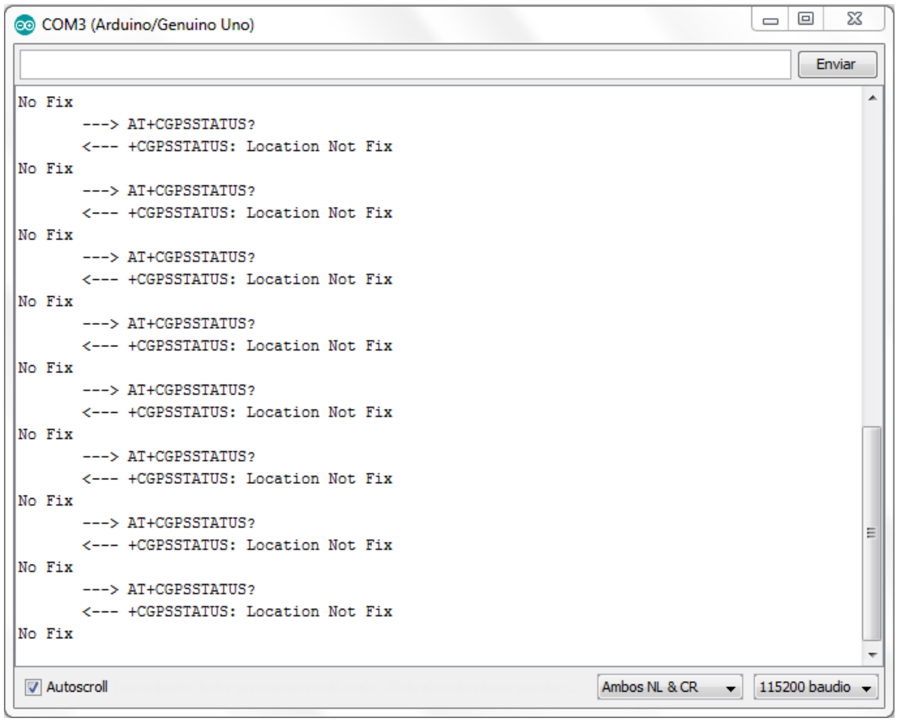


Figura 3.18. Pulsera electrónica: Búsqueda de arreglos satelitales.

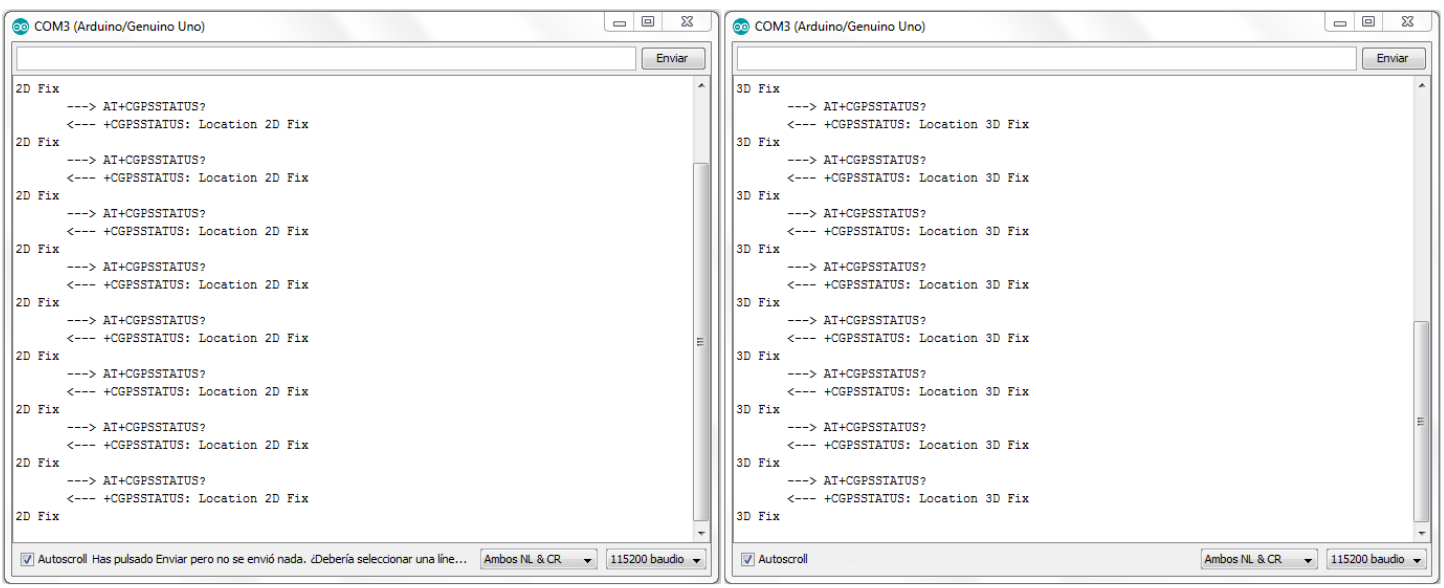


Figura 3.19. Pulsera electrónica: Arreglo satelital encontrado.

Luego de este proceso, la pulsera electrónica guarda los datos obtenidos en el formato establecido (ver Figura 3.20) y se envían en una línea de texto como mensaje SMS al Módulo procesador de datos.

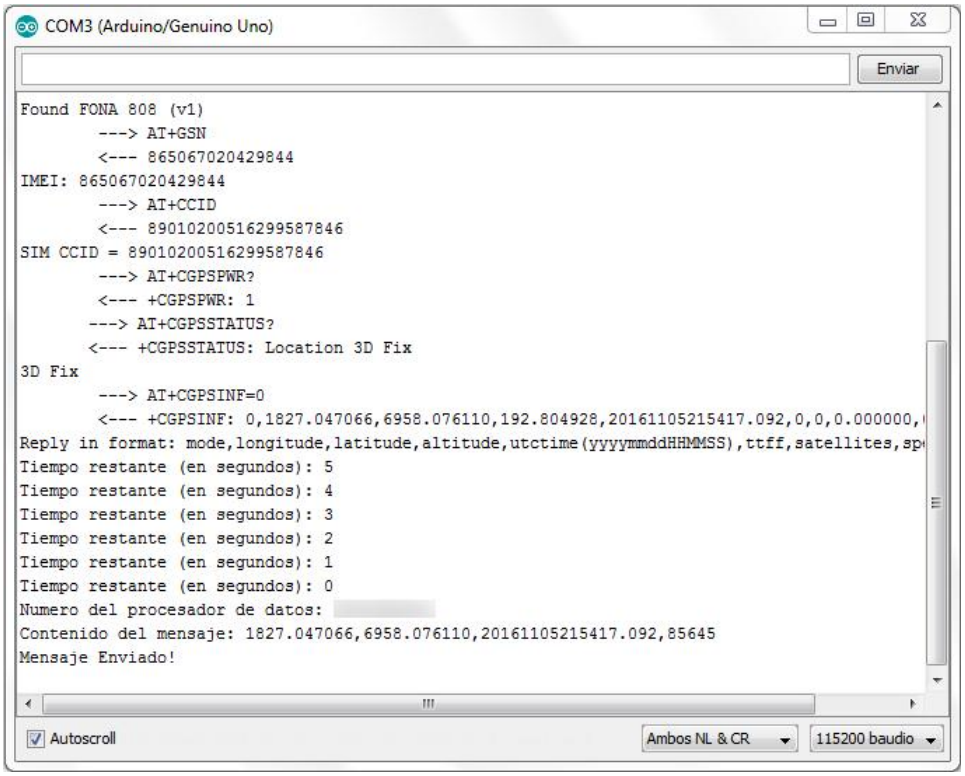


Figura 3.20. Pulsera electrónica: Guardado de los datos geográficos y envío vía SMS al procesador de datos.

3.1.1.4. Funcionamiento del módulo procesador de datos.

El módulo procesador de datos dispone de la configuración necesaria para recibir los datos geográficos de la pulsera electrónica. Cuando recibe un SMS, el mensaje es analizado por el programa "Recibir_Mensajes.py", éste determina si el mensaje proviene de la aplicación del cuidador o de la pulsera electrónica. La Figura 3.21 presenta el proceso de recepción.

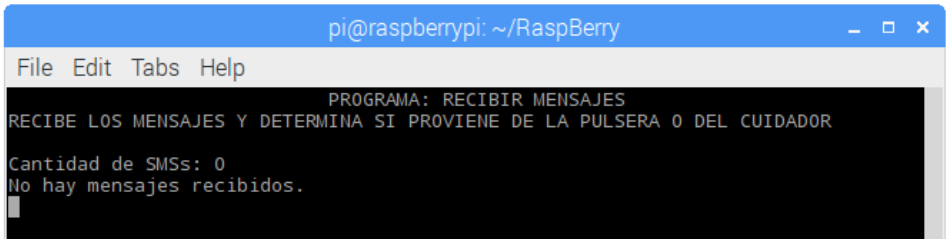


Figura 3.21. Módulo procesador de datos: En espera de mensaje SMS.

Si el mensaje de texto recibido fue enviado por la aplicación móvil del cuidador, "Recibir_Mensajes.py" ejecuta automáticamente el programa "Mensaje_Cuidador.py" para analizar la información que envía la aplicación. Ver ejemplo de SMS recibido en la Figura 3.22.

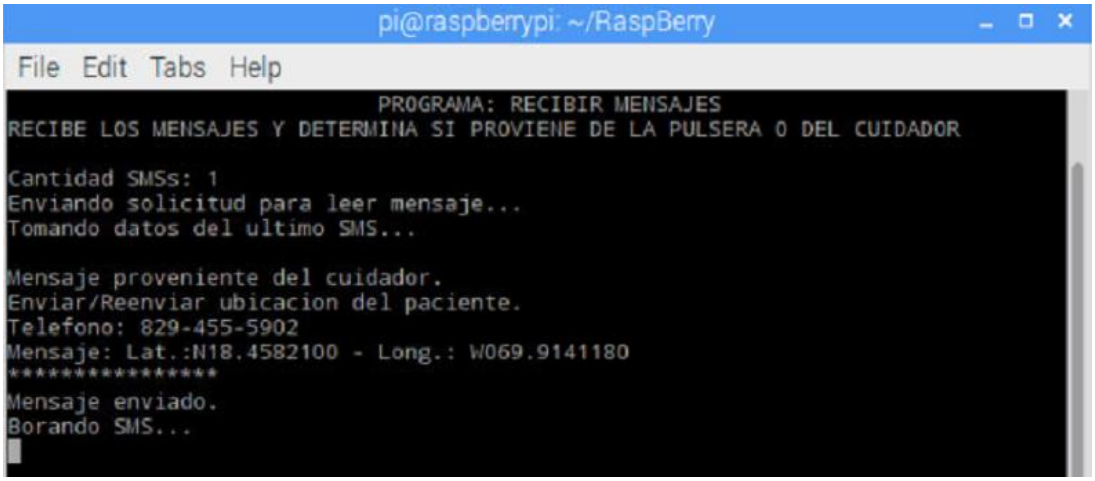


Figura 3.22. Módulo procesador de datos: Recibiendo SMS de la aplicación móvil y enviando respuesta de solicitud.

El programa "Mensaje_Cuidador.py" puede tomar las siguientes decisiones de acuerdo al contenido del mensaje:

a. Informar que recibió la alerta de peligro del paciente: Procede a cambiar el campo "Respuesta" a "Positiva" en la tabla "Resultado de análisis de las ubicaciones".
b. Solicitud de la ubicación del paciente por parte del cuidador: Procede a abrir el archivo "Enviar_Mensajes.py" y envía un mensaje al cuidador con la ubicación geográfica más reciente del paciente.

Si el mensaje de texto recibido fue enviado por la pulsera electrónica (como se muestra en la Figura 3.23), "Recibir_Mensajes.py" ejecuta automáticamente el programa

"Mensaje_Pulsera.py" para analizar la información recibida y guardarla en la base de datos del sistema.

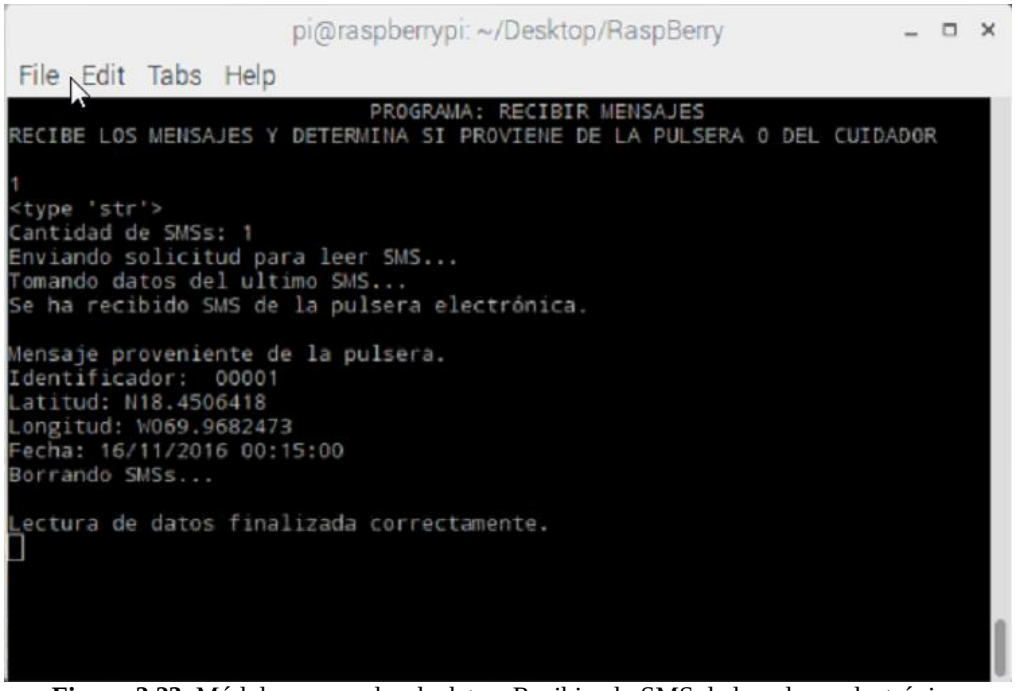


Figura 3.23. Módulo procesador de datos: Recibiendo SMS de la pulsera electrónica.

El programa modular residente "Analizar_Coordenadas.py", analiza las coordenadas guardadas en la base de datos. Si las coordenadas indican que el paciente se encuentra fuera de las zonas seguras, procederá a colocar en el campo de "Estado" la condición "Peligro" (como se observa en la Figura 3.24) en la tabla "Resultados de análisis de las ubicaciones".

Figura 3.24. Módulo procesador de datos: Estado "Peligro" en la tabla de "Resultados de análisis de las ubicaciones".

Si las coordenadas indican que el paciente se encuentra dentro de una de las zonas seguras, se procede a colocar en el campo de "Estado" la condición "Seguro" (como se muestra en la Figura 3.25) en la tabla "Resultados de análisis de las ubicaciones".

Figura 3.25. Módulo procesador de datos: Estado "Seguro" en la tabla de "Resultados de análisis de las ubicaciones".

El programa "Alerta_Estado.py" es el programa modular residente encargado de verificar si algún paciente está en estado "Peligro". De ser así, puede proceder a tomar las siguientes medidas:

1. Alertar al cuidador mediante la aplicación móvil. Si el sistema no ha recibido respuesta por parte del cuidador mediante un SMS enviado por la aplicación, se realizará una llamada al Sistema Nacional de Atención a Emergencias y Seguridad 9-1-1.

El programa "Actualizar_Tiempo.py" es el programa modular residente encargado de verificar la fecha de la última actualización de las coordenadas geográficas recibidas de la pulsera electrónica. Este determina el tiempo transcurrido y lo guarda en la base de datos.

En la Figura 3.26, se muestra la información de los pacientes registrados en la base de datos.

SQL Query | Result | Edit Data

Table: paciente

Filterfield: Filter:

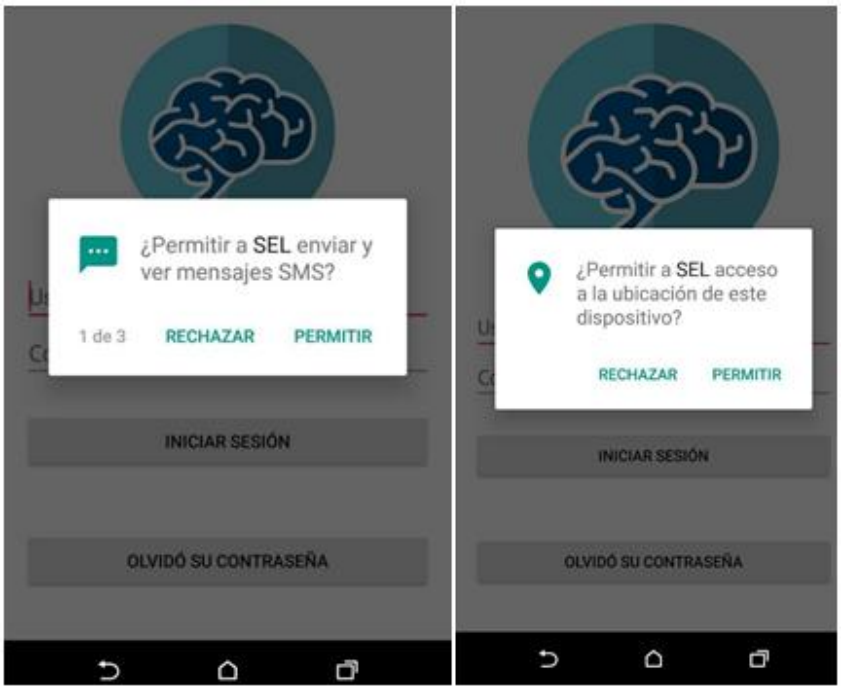
ID	Nombres	Apellidos	Edad	Cedula	Direccion	Telefono	Etapa_de_la_Enfermedad	Id_Dispositivo
1	Juan Carlos	Nuñez Medina	40	001-5984515-4	Autopista 30 de Mayo Esq. Heroes de Luperon, Santo Domingo 10401	809-685-8888	2	85985
2	Carmen María	Pérez Báez	50	001-5615155-8	Av Venezuela 92, Santo Domingo Este 11906	809-666-4894	2	66767

Figura 3.26. Base de datos con la información de los pacientes registrados en el SEL.

En el Apéndice F, se muestran las demás bases de datos del sistema.

3.1.1.5. Funcionamiento de la aplicación móvil.

La aplicación móvil del cuidador necesita la autorización del usuario para el uso del GPS y el acceso a los mensajes de texto los cuales son solicitados únicamente la primera vez que se ejecute la aplicación móvil. Esto se muestra en la Figura 3.27.



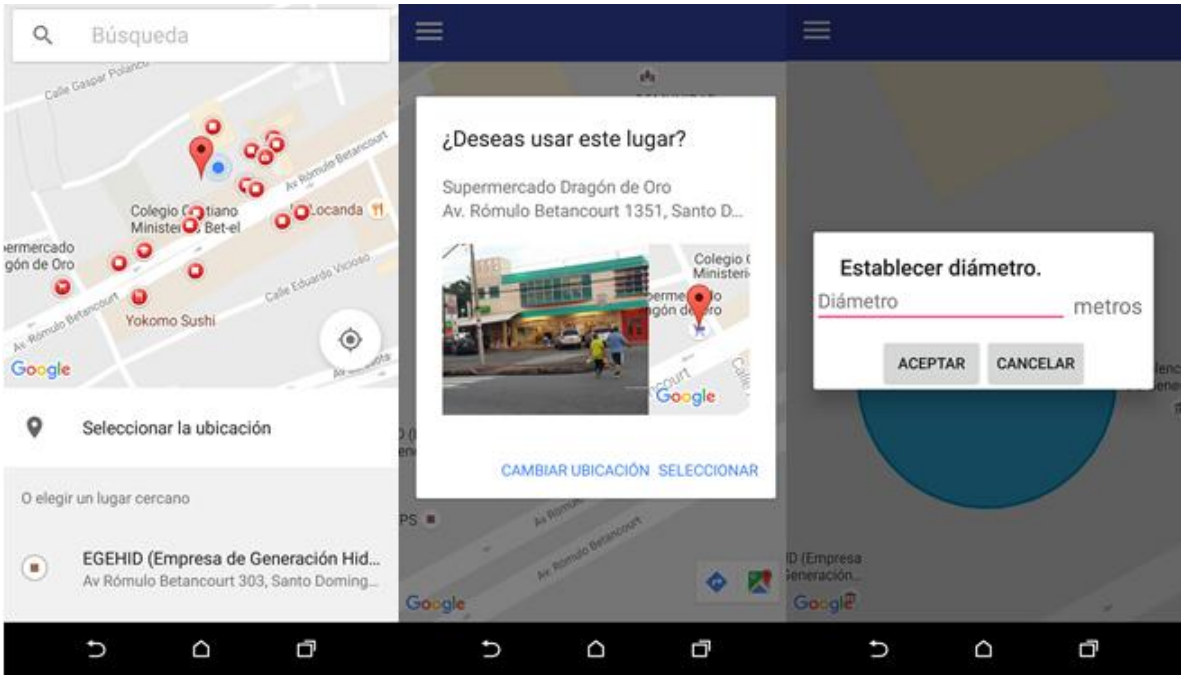
a) b)

Figura 3.27. Aplicación móvil. a) Solicitud de acceso a los mensajes de textos. b) Solicitud de acceso a la ubicación del GPS.

En la Figura 3.28, se observa el procedimiento para establecer una zona segura. Primero, el usuario selecciona en el mapa la ubicación del área segura.[125] Luego, la aplicación solicita la confirmación para guardar el lugar seleccionado como una zona segura. Si su respuesta es positiva, el usuario procede a insertar el diámetro del espacio que ocupa la zona.

Posteriormente, la aplicación notifica al cuidador que la zona segura ha sido registrada correctamente, y envía las coordenadas geográficas, mediante SMS al procesador de datos. Véase la Figura 3.29.

[125] La interacción Aplicación – Google Maps se logró mediante la utilización de una API (Application Programming Interface) de la plataforma de *Google* para Android Studio. https://github.com/googlesamples/android-play-places



a) b) c)

Figura 3.28. Aplicación móvil. a) Seleccionar zona segura. b) Confirmar zona segura. c) Establecer diámetro de la zona segura.

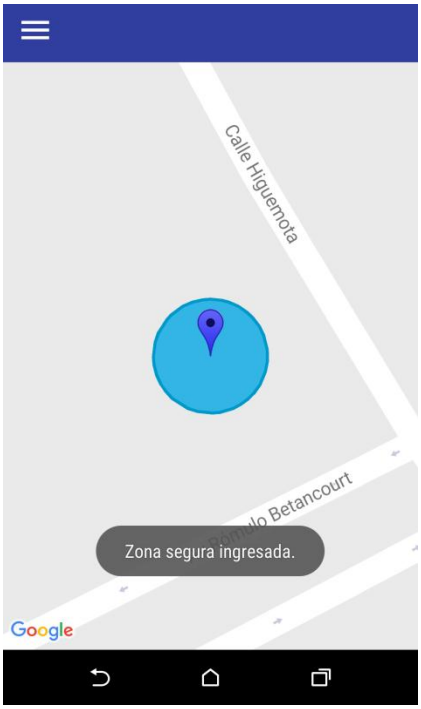
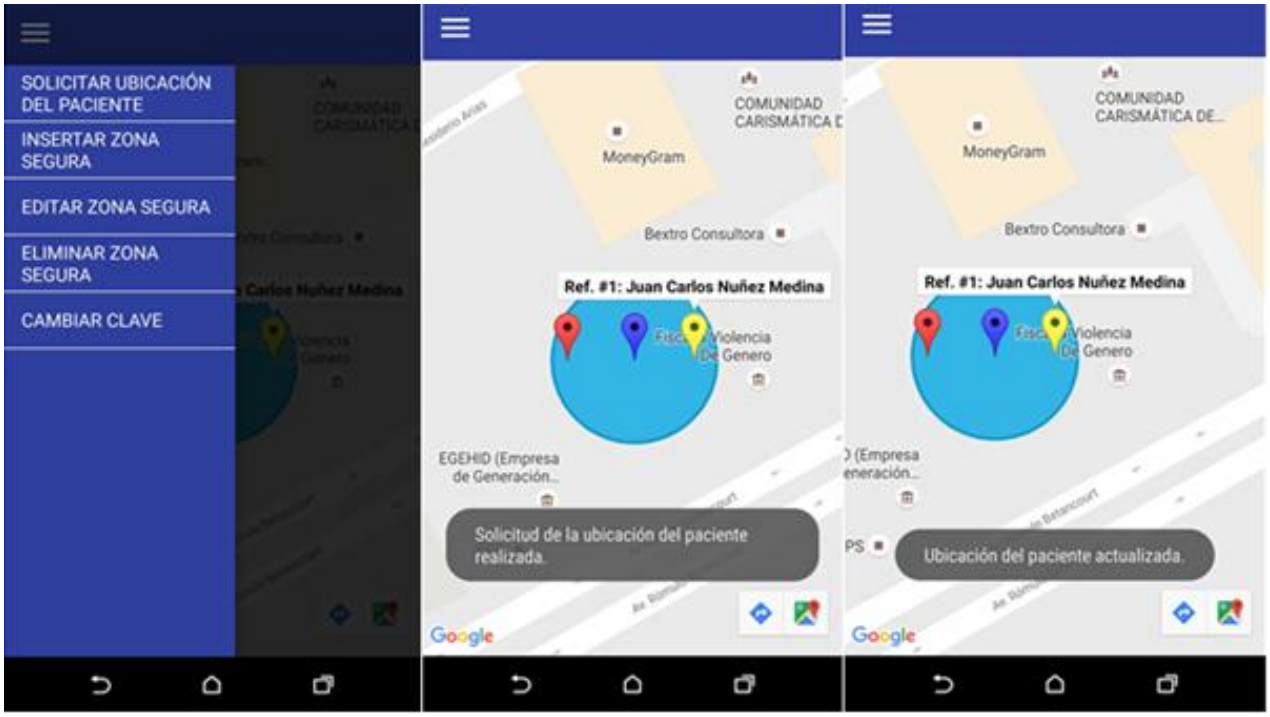


Figura 3.29. Zona segura ingresada. *El marcador y el área azul representan la zona segura.*

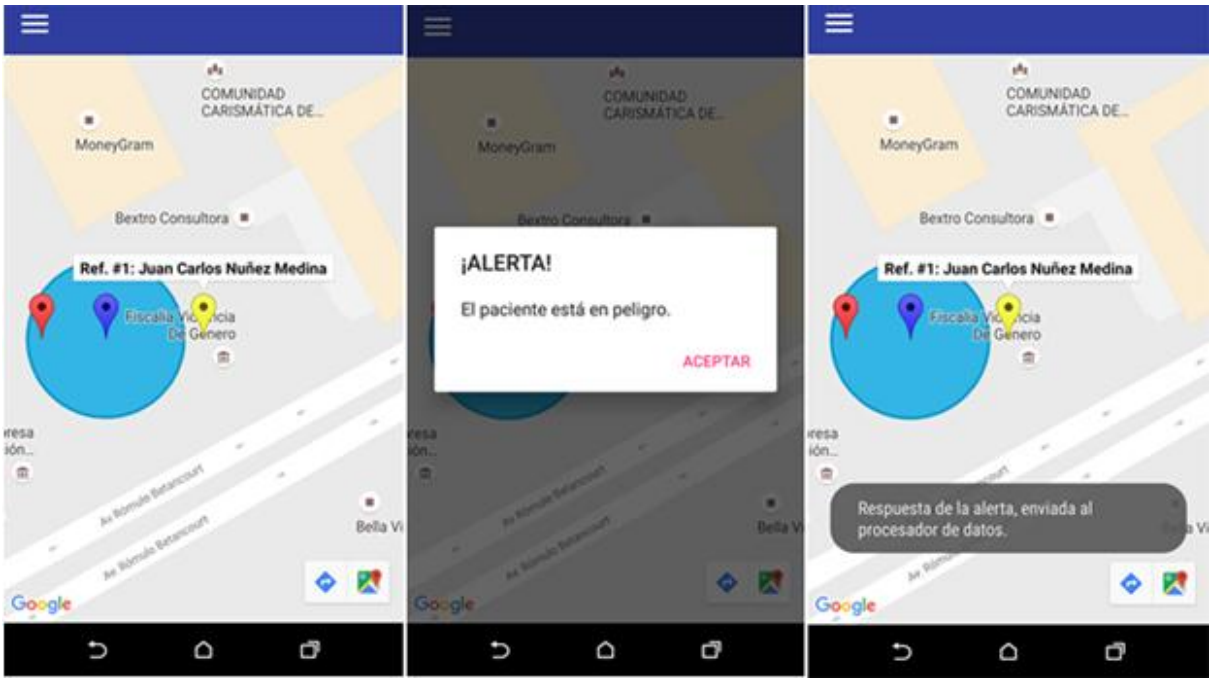
Al solicitar la ubicación del paciente, la aplicación envía un SMS al procesador de datos y este último, verifica el mensaje y notifica al usuario si la petición se ha realizado correctamente. En otro orden, si el paciente se encuentra en peligro, el procesador de datos notifica a la aplicación y ésta última indica que recibió el SMS. Ver Figura 3.30.



a) b) c)

Figura 3.30. Aplicación móvil. a) Barra de herramientas. b) Solicitud de la ubicación del paciente. c) Ubicación actualizada del paciente. *El cuidador corresponde al marcador rojo y el paciente al marcador amarillo.*

Finalmente, cuando la aplicación móvil reciba el mensaje, verifica si el remitente es el procesador de datos, de ser así, evalúa su contenido para determinar si es la ubicación del paciente o una alerta de peligro. Por otro lado, si el mensaje de texto posee una indicación de alerta, entonces, notifica al usuario de la misma, y si contiene la ubicación del paciente, la actualiza automáticamente en el mapa. Observe la Figura 3.31 donde se muestra las posiciones del cuidador, zona segura y paciente, respectivamente.



a) b) c)

Figura 3.31. Aplicación móvil. a) Paciente fuera de la zona segura. b) Estado de alerta en peligro. c) Respuesta del cuidador a la alerta enviada al procesador de datos.

3.2. Evaluación de hallazgos

Durante el desarrollo de las pruebas funcionales del sistema y a partir de los resultados obtenidos, se encontraron algunos hallazgos que aportaron a la generación de nuevas ideas.

Se pudo observar que el sistema tiene la capacidad de adaptarse a instalaciones informáticas y/o electrónicas más sofisticadas y con diferentes propósitos, debido a que la información de mayor relevancia (datos de pacientes, cuidadores y dispositivos) es guardada en bases de datos que tienen compatibilidad con una gran variedad de software.

En este sentido, las condiciones anteriores implican que el sistema pueda ser integrado en áreas de la salud, como parte del Sistema Nacional de Atención a Emergencias, 9-1-1 y Seguridad para monitorizar el estado y ubicación física de pacientes con enfermedades degenerativas que afectan al cerebro. Además, el 9-1-1 ha puesto en ejecución la instalación de cámaras de video-vigilancia en diversas zonas de la provincia de Santo Domingo, República Dominicana, que podrían asistir al sistema localizador en el reconocimiento e identificación de enfermos.

De los resultados obtenidos con las operaciones de la pulsera electrónica, el tiempo de obtención de las coordenadas geográficas en ocasiones puede ser muy grande (superior al tiempo promedio) cuando un obstáculo físico afecta las mediciones realizadas por el receptor GPS, reduciendo la tasa de envío de los SMS hacia el módulo procesador de datos. Este inconveniente puede ser resuelto mediante el establecimiento de un contrato con una red móvil que reduzca los gastos por uso de paquetes de datos, al utilizar el servicio GPRS del módulo GSM-GPS para obtener coordenadas geográficas.

A partir de los resultados obtenidos con la aplicación Android para el cuidador, se determinó que la misma posee una interfaz sencilla de utilizar. En este aspecto, durante las pruebas de solicitud y recepción de la ubicación del paciente mediante mensaje de texto, al recibir respuesta de dicha solicitud, la aplicación móvil colocó automáticamente en el mapa, un marcador de color amarillo con el nombre “Ubicación del paciente”, esto permitió tener una mejor apreciación de la ubicación del enfermo, además de indicar si el mismo estaba dentro de una zona segura.

CONCLUSIONES

En este trabajo de tres capítulos, se ha logrado compactar el desarrollo coherente de un sistema electrónico localizador, que responde a situaciones reales y de carácter puntual, dirigido a aquellos individuos que tengan dependencia de un tercero y dificultades para orientarse. Su composición estructural ha sido formada mediante la implementación precisa de módulos de comunicación inalámbrica y dispositivos de control, que han generado resultados adecuados para la elaboración de un diseño final con las tecnologías utilizadas.

Particularizando los elementos y procesos de la prueba funcional, destacan los siguientes:

- La función del programa modular "Analizar_Coordenadas.py", corresponde con una fase de la prueba, considerada elemental, que determina de forma directa el propósito del sistema. Los datos de este proceso son guardados en una base de datos que permite luego, condicionar la condición ("Seguro" o "Peligro") de estado del paciente.

- El programa de la plataforma Arduino para controlar el módulo SIM808, determina la forma en que es enviada la información del paciente al procesador de datos y se compone de todas las operaciones y procesos de la pulsera electrónica.

- La interfaz de administrador realizada en la Raspberry Pi 2, permite la interacción entre los usuarios y el sistema. En adición, utiliza Python como lenguaje de programación y la librería Tkinter para presentar del procesador de datos.

RECOMENDACIONES

1. Incluir en la pulsera electrónica, un método de reproducción de música o melodías para tranquilizar y entretener al paciente con Alzheimer.
2. Hacer un acuerdo con el Sistema Nacional de Atención a Emergencias y Seguridad 9-1-1, para asociar el sistema electrónico con sus servicios, de forma que puedan asistir a los custodios al momento de recuperar a sus pacientes.
3. Añadir en la base de datos un historial médico del paciente, medicamentos que consume, etc., para que los doctores (como usuarios del sistema) puedan tener acceso a esta información al momento de atender al enfermo.
4. Permitir al custodio, la posibilidad de programar alarmas en la aplicación móvil para recordar las rutinas que sigue con el paciente. Y que además, éstas pueden ser reproducidas en la pulsera electrónica para que el individuo en la etapa temprana del Alzheimer sepa que acción realizar.
5. Enviar correo electrónico al cuidador cuando el paciente esté en peligro.
6. Gestionar acuerdos con las telefónicas del país para reducir los costos del consumo mensual por el envío de mensajes SMS del sistema.
7. Dependiendo de las necesidades particulares del paciente, darle la opción al cuidador de que el dispositivo localizable sea un collar, unos zapatos, una correa o cualquier otro tipo de prenda.
8. En el aspecto de la seguridad, introducir un método de cifrado del contenido de los SMS en sus procesos de transmisión y recepción.

9. Diseñar la pulsera electrónica para un reconocimiento autónomo de las zonas seguras, y que solo envíe mensajes cuando el paciente las abandone. Esto reduciría la cantidad de SMS remitidos desde el dispositivo localizable al procesador de datos.
10. Agregar una opción en la aplicación móvil, en la cual el custodio sea capaz de seleccionar áreas específicas del hogar como seguras o peligrosas.
11. En la pulsera electrónica, habilitar la función de localización del módulo GSM, como un proceso asistido para el GPS en la obtención de coordenadas geográficas. Esto tendría una gran eficiencia, si se realiza con una compañía telefónica un contrato que minimice los costos durante el uso del servicio GPRS.

REFERENCIAS BIBLIOGRÁFICAS

Publicación periódica:

(136) Batista, L. (29 de octubre del 2016). *Estudio demuestra que dominicanos son proclives a sufrir de Alzheimer.* Diario Libre. Consultado el 29 de octubre de 2016, de http://www.diariolibre.com/noticias/salud/estudio-demuestra-que-dominicanos-son-proclives-a-sufrir-de-alzheimer-YD5306551

Libros:

(11) Carlesimo GA, Oscar-Berman M (junio de 1992). *Memory deficits in Alzheimer's patients: a comprehensive review*. Neuropsychol Rev 3 (2): 119-69.

(40-46, 48, 50, 51, 130-132) Dardari, Davide Falletti, Emanuela Luise, Marco. (2012). *Satellite and Terrestrial Radio Positioning Techniques - A Signal Processing Perspective* (p. 1-14). Elsevier. Versión en línea: http://app.knovel.com/hotlink/toc/id:kpSTRPTAS1/satellite-terrestrial/satellite-terrestrial

(52, 53, 55-58) Ogaja, Clement A. (2011). *Applied GPS for Engineers and Project Managers* (p. 4-11). American Society of Civil Engineers (ASCE). Versión en línea: http://app.knovel.com/hotlink/toc/id:kpAGPSEPMD/applied-gps-engineers/applied-gps-engineers

(61, 62) Epstein, Joseph. (2009). *Scalable VoIP Mobility - Integration and Deployment* (p. 3-21). Elsevier. Versión en línea: http://app.knovel.com/hotlink/toc/id:kpSVIPMIDC/scalable-voip-mobility/scalable-voip-mobility

(63-69, 72-76) Dalal, Upena. (2015). *Wireless Communication and Networks* (p.413-451). Oxford University Press. Versión en línea: http://app.knovel.com/hotlink/toc/id:kpWCN00011/wireless-communication/wireless-communication

(94) Berral, I. (2010). *Equipos microinformáticos* (p. 136). Paraninfo. Versión en línea: https://books.google.com.do/books?id=mqm-FYlidSMC&lpg=PA136&ots=RnXX4tweqO&dq=comandos%20AT%20Attention&hl=es&pg=PA136#v=onepage&q=comandos%20AT%20Attention&f=false

Documentos publicados en internet:

(1,5) Instituto Nacional Sobre el Envejecimiento. (2010, agosto). *La enfermedad de Alzheimer.* Recuperado el 1 de octubre de 2016, de https://www.nia.nih.gov/espanol/publicaciones/enfermedad-alzheimer

(2,3) Alzheimer's association. (2016). *Etapas*. Recuperado el 4 de octubre de 2016, de http://www.alz.org/espanol/about/etapas.asp

(4, 6, 13, 14, 20-22, 24-33, 38, 39) afate. (2016). *Alteraciones Psicológicas y del Comportamiento.* Recuperado el 4 octubre de 2016, de http://afate.es/alteraciones-psicologicas-y-del-comportamiento

(7-10, 12-17, 19, 23) Eun-Shim Nahm. (2007). *La enfermedad de Alzheimer y otras demencias.* Recuperado el 4 de octubre de 2016, de http://lightbridgehealthcare.com/caregiver_resources/knowledgebase/S-Chapter2.pdf

(18) García, C. (2016). *¿Cuándo la tristeza es normal o patológica?* Recuperado el 4 de octubre de 2016, de http://www.actualpsico.com/cuando-la-tristeza-es-normal-o-patologica/

(34-37) Instituto Nacional Sobre el Envejecimiento. (2010, agosto). *La enfermedad de Alzheimer.* Recuperado el 1 de octubre de 2016, de https://www.nia.nih.gov/espanol/publicaciones/guia-quienes-cuidan-personas-alzheimer

(47) Bensky, A. (s.f). *Chapter 7: Time of Arrival and Time Difference of Arrival*. Recuperado el 2 de octubre de 2016, de http://www.globalspec.com/reference/81358/203279/chapter-7-time-of-arrival-and-time-difference-of-arrival

(49) masmar. (5 de mayo del 2008). *GPS: Triangulación desde los satélites*. Recuperado el 2 de octubre de 2016, de http://www.masmar.net/esl/N%C3%A1utica/Seguridad-Naval-Naufragios/Sistema-de-Posicionamiento-Global.-%C2%BFQue-es-un-GPS/Triangulaci%C3%B3n-desde-los-sat%C3%A9lites

(54) GPS.gov. (2016). *El Sistema de Posicionamiento Global.* Recuperado el 5 de octubre de 2016, de http://www.gps.gov/systems/gps/spanish.php

(59, 60) mecinca.net. (s.f.). *Tratamiento de la señal GPS.* Recuperado el 6 de octubre de 2016, de http://www.mecinca.net/Presentaciones/GPSsencillo/index4.htm

(70) Belmonte, D. (2014). *Universidad de Valencia Rogelio Montañana Ampliación Redes 7-1 Tema 7 Redes Inalámbricas y Movilidad.* Recuperado el 5 de octubre de 2016, de http://images.slideplayer.es/1/106604/slides/slide_84.jpg

(71) SIMCOM.EE. (2016). *SIM800_Hardware_Design_V1.08* (p. 13). Consultado el 9 de octubre de 2016, de http://simcom.ee/documents/SIM800/SIM800_Hardware%20Design_V1.08.pdf

(84, 85, 86) SIMCOM.EE. (2016). *SIM800* (p. 1-23). Recuperado el 9 de octubre de 2016, de http://simcom.ee/modules/gsm-gprs-gnss/sim800/

(77) Definitions.net. (s.f.). *Printed Circuit Board*. Recuperado el 10 de octubre de 2016, de http://www.definitions.net/definition/Printed%20Circuit%20Board

(78-81, 83) SIMCOM.EE. (2016). *SIM808.* Recuperado el 9 de octubre de 2016, de http://simcom.ee/modules/gsm-gprs-gnss/sim808/

(82) Zahradnik, F. (2016) *Time to First Fix (TTFF).* Recuperado el 10 de octubre de 2016, de https://www.lifewire.com/time-to-first-fix-ttff-1683313

(87) Mayank. (2016). *What is a microcontroller? And how does it differ from a microprocessor?* Maxembedded.com. Recuperado el 10 de octubre de 2016, de http://maxembedded.com/2011/06/mcu-vs-mpu/

(88) Dictionary.com. (2016). *Dual in-line package.* Recuperado el 10 de octubre de 2016, de http://www.dictionary.com/browse/dual-in-line-package

(89) Atmel. (2016). *Atmega 328/P Datasheet Complete.* Recuperado el 6 de octubre de 2016, de http://www.atmel.com/Images/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_datasheet.pdf

(90, 91) sparkfun. (s.f.). *I2C.* Recuperado el 12 de octubre de 2016, de https://learn.sparkfun.com/tutorials/i2c

(92) Arduino.cc. (s.f.). *PWM.* Recuperado el 12 de octubre de 2016, de https://www.arduino.cc/en/Tutorial/PWM

(93) Arduino.cc. (s.f.). *Arduino/Processing Language Comparison.* Recuperado el 12 de octubre de 2016, de https://www.arduino.cc/en/Reference/Comparison

(95) CCM. (s.f.). *USB (Bus de serie universal).* Recuperado el 6 de octubre de 2016, de http://es.ccm.net/contents/407-usb-bus-de-serie-universal

(96) radio-electronics.com. (s.f.). *What is SMT Surface Mount Technology - Tutorial.* Recuperado el 6 de octubre de 2016, de http://www.radio-electronics.com/info/data/smt/what-is-surface-mount-technology-tutorial.php

(97) Arduino. (2016). *Technical Specs.* Recuperado el 6 de octubre de, https://www.arduino.cc/en/Main/ArduinoBoardUno

(98) Pérez, M. (s.f.) *PUERTOS Y BUSES 1: I2C Y UART.* Geekytheory.com. Recuperado el 6 de octubre de https://geekytheory.com/puertos-y-buses-1-i2c-y-uart/

(99-101) Agnihotri, N. (2012). *AT Commands, GSM AT command set.* Recuperado el 6 de octubre de 2016, de http://www.engineersgarage.com/tutorials/at-commands

(102) redyc.com. (2014, septiembre). *NÚMEROS DE MÓVIL.* Recuperado el 6 de octubre de, http://www.redyc.com/rastros/2014/09/05/numeros-de-movil/

(103, 104, 135) Raspberrypi.org. (s.f.). *RASPBERRY PI HARDWARE*. Recuperado el 5 de octubre de 2016, de https://www.raspberrypi.org/documentation/hardware/

(105) Sánchez, I. (s.f.) *¿Qué significa SOC? System on a chip.* About. Recuperado el 6 de octubre de http://computadoras.about.com/od/conocer-mi-computadora/a/Que-Es-La-Tarjeta-Grafica.htm

(106) Technopedia. (s.f.). *RASPBERRY PI HARDWARE*. Recuperado el 5 de octubre de 2016, de https://www.techopedia.com/definition/5900/advanced-risc-machine-arm

(107, 108, 109) Raspberrypi.org. (s.f.). *RASPBERRY PI HARDWARE*. Recuperado el 5 de octubre de 2016, de https://www.raspberrypi.org/documentation/

(110-117) Python Software Foundation. (2016, septiembre). *The Python Language Reference*. Recuperado el 4 de octubre de 2016, de https://docs.python.org/2/library/

(118) Hedges, A. (2016). *Finding distances based on Latitude and Longitude.* Recuperado el 10 de octubre de 2016, de http://andrew.hedges.name/experiments/haversine/

(119) androidcentral.com. (s.f.) *Android Apps.* Recuperado el 10 de octubre de 2016, de http://www.androidcentral.com/apps

(120-121) Android Studio. (s.f.) *El IDE oficial Android*. Recuperado el 10 de octubre de 2016, de https://developer.android.com/studio/index.html?hl=es-419

(122) jcea.es. (17 de abril del 2005). *Definiciones GPS*. Recuperado el 10 de octubre de 2016, de https://www.jcea.es/artic/gps-definiciones.htm

(123) www.timeanddate.com. (s.f.) *UTC – The World´s Time Standard*. Recuperado el 10 de octubre de 2016, de https://www.timeanddate.com/time/aboututc.html

(124) xataka. (s.f.) *HDMI.* Recuperado el 10 de octubre de 2016, de http://www.xataka.com/alta-definicion/hdmi

(125) Github. (2016). *Google Places API for Android Samples*. Recuperado el 26 de octubre de 2016, de https://github.com/googlesamples/android-play-places

(126-129) cuadroscomparativos.com. (2016). *LA ENFERMEDAD DE ALZHEIMER Y LA DEMENCIA: CUADROS COMPARATIVOS, IMÁGENES E INFORMACIÓN.* Recuperado el 15 de noviembre de 2016, de http://cuadroscomparativos.com/la-enfermedad-de-alzheimer-y-la-demencia-cuadros-comparativos-imagenes-e-informacion/

(133) adafruit. (s.f.). *Adafruit FONA 808 Cellular + GPS Breakout.* Recuperado el 3 de octubre de 2016, de https://learn.adafruit.com/adafruit-fona-808-cellular-plus-gps-breakout/overview?view=all

(134) adafruit. (s.f.). *Adafruit FONA 800 Shield.* Recuperado el 3 de octubre de 2016, de https://learn.adafruit.com/adafruit-fona-800-shield/downloads

GLOSARIO

ARM.- Del inglés, *Advanced RISC Machine.* Es una arquitectura RISC de 32 bits utilizada en microprocesadores y microcontroladores. *(Pág. 58)*

Arreglo satelital.- Es una posición, o ubicación geográfica calculada a partir de las mediciones de las distancias de las señales satelitales. *(Págs. 90, 116)*

AT.- Abreviatura en inglés de la palabra *Attention*. *(Págs. 54, 56, 57, 89, 92)*

GPRS.- Del inglés, *General Packet Radio Service.* Es una extensión de GSM para la conmutación de paquetes de datos. *(Págs. 45, 124)*

HDMI.- Del inglés, *High-Definition Multimedia Interface.* Es una tecnología utilizada para la transferencia de audio y video digital. *(Pág. 102)*

Metamorfosis.- Transformación de algo en otra cosa. *(Pág. 28)*

I2C.- Del inglés *Inter-Integrated Circuit*. Es un protocolo de comunicación utilizado generalmente entre microcontroladores. *(Pág. 54)*

ICCID.- Del inglés, *Integrated Circuit Card Identifier*. Es un número guardado en la tarjeta SIM que sirve para identificarla por parte de una operadora. *(Págs. 57, 90, 109)*

Interfaz Um.- Es una interfaz que utiliza el sistema de estaciones base (BSS) para establecer comunicación con la red GSM. *(Pág. 43)*

MCU.- Del inglés, *MicroController Unit*. Unidad de microcontrolador. *(Pág. 53)*

PCB.- Del inglés, *Printed Circuit Board*, Placa de circuito impreso. *(Pág. 46)*

PDIP.- Del inglés, Plastic Dual In-Line Package. Se refiere a un tipo de encapsulado para circuitos integrados utilizados comúnmente para Placas de proyectos. *(Pág. 53)*

PWM.- Del inglés, *Pulse Width Modulation*. Se refiere a una técnica conocida como modulación por ancho de pulso. *(Pág. 54)*

Rx.- Abreviatura de la palabra en inglés *Reception* para telecomunicaciones. *(Pág. 59)*

SMT.- Del inglés, *Surface Mount Technology*. Se refiere a los circuitos integrados con encapsulados utilizados en montajes de superficie. *(Pág. 55)*

SPI.- Del inglés, *Serial Peripheral Interface*. Es un protocolo de comunicaciones utilizado para la transferencia de información entre microcontroladores y otros dispositivos electrónicos. *(Pág. 54)*

SoC.- Del inglés, *System-on-chip*. Es una connotación referida a los sistemas electrónicos integrados en un solo chip. *(Pág. 58)*

TDOA.- Del inglés, *Time Difference of Arrival*. Es una técnica basada en mediciones de diferencia de distancia entre dos o más estaciones que generan señales en tiempos conocidos. *(Pág. 36)*

TTFF.- Del inglés, *Time To First Fix*. Es una medición que consiste en determinar el tiempo que se toma un receptor GPS para calcular una posición geográfica. *(Pág. 51)*

Tx.- Abreviatura de la palabra en inglés, *Transmission* para telecomunicaciones. *(Pág. 59)*

Triangulación.- Es un método utilizado para medir la distancia de las señales recorridas entre los satélites y una posición relativa, de forma que pueda calcularse dicha posición. *(Págs. 37, 39)*

Tristeza patológica.- Es un síntoma de trastornos mentales, caracterizado por la tendencia de un individuo a interpretar y percibir de forma negativa lo que le rodea y así mismo. *(Pág. 24)*

UTC.- Del inglés, *Coordinated Universal Time*. Es el estándar de tiempo que regula el tiempo a nivel mundial. *(Pág. 91)*

ANEXOS

APÉNDICE A: Más sobre Alzheimer

Tabla A.1. Diferencias entre Alzheimer y demencia senil.[126]

Diferencias	
Demencia Senil	Alzheimer
Olvida detalles de una situación	Olvida situaciones completa
Con frecuencia recuerda más tarde	No recuerda más tarde
Puede seguir indicaciones	Le cuesta seguir indicaciones
Puede usar notas	Le cuesta usar notas para recordar
Puede cuidar de sí mismo	Le es difícil cuidar de sí mismo

Tabla A.2. Síntomas del Alzheimer y recomendaciones de prevención.[127]

Engaños de la memoria		
Sintomas		Recomendaciones para prevenir enfermedad
Primeras etapas	Etapas posteriores	Intentar mantener una vida socialmente activa.
Breves pérdidas de la memoria.	Confusión, desorientación temporal.	Llevar un estilo de vida sano sin abusar del alcohol y el tabaco.
Cambios de personalidad (apatía, desinterés en la actividad social).	Divagación, dificultad para conversar.	A partir de los 50 años, es necesario controlar periódicamente las habilidades mentales realizando una evaluación de la memoria y restantes funciones cognitivas.
Menor capacidad intelectual.	Erráticos cambios de humor.	Realizar ejercicio físico periódicamente.
Irritabilidad y dificultades motrices.	Serio deterioro de la salud.	Mantener una amplia gamma de intereses y hobbies.
	Incapacidad para cuidarse solo.	Evitar situaciones de estrés emocional y mantener una vida relajada que le permita disfrutar de las actividades de ocio.
Dieta: los especialistas recomiendan carnes blancas en lugar de rojas, consumir frutas y verduras y beber mucha agua, ya que cuidar la dieta puede ayudar a detener la progresión del Alzheimer.		

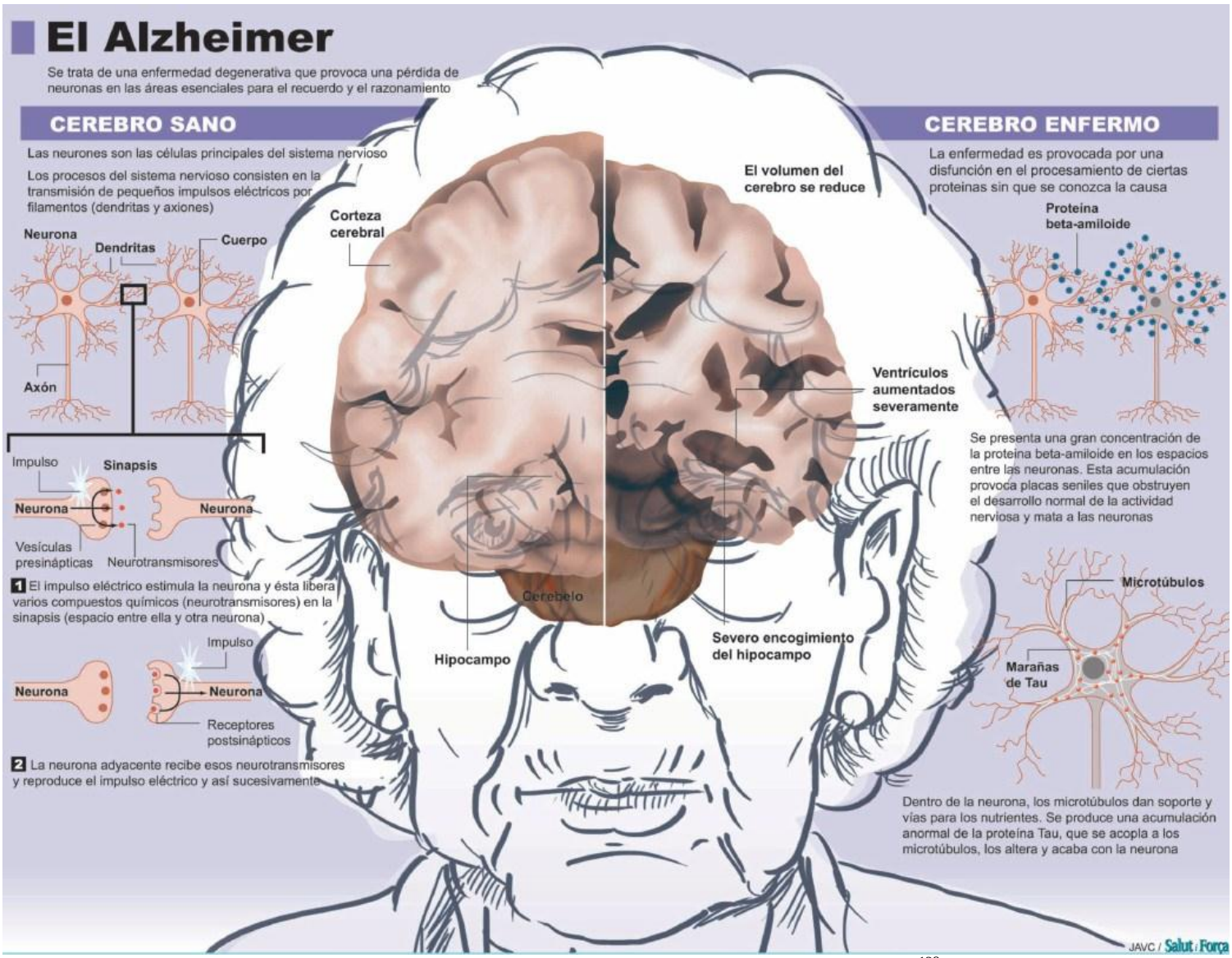


Figura A.1. Comparación entre un cerebro sano y uno enfermo.[128]

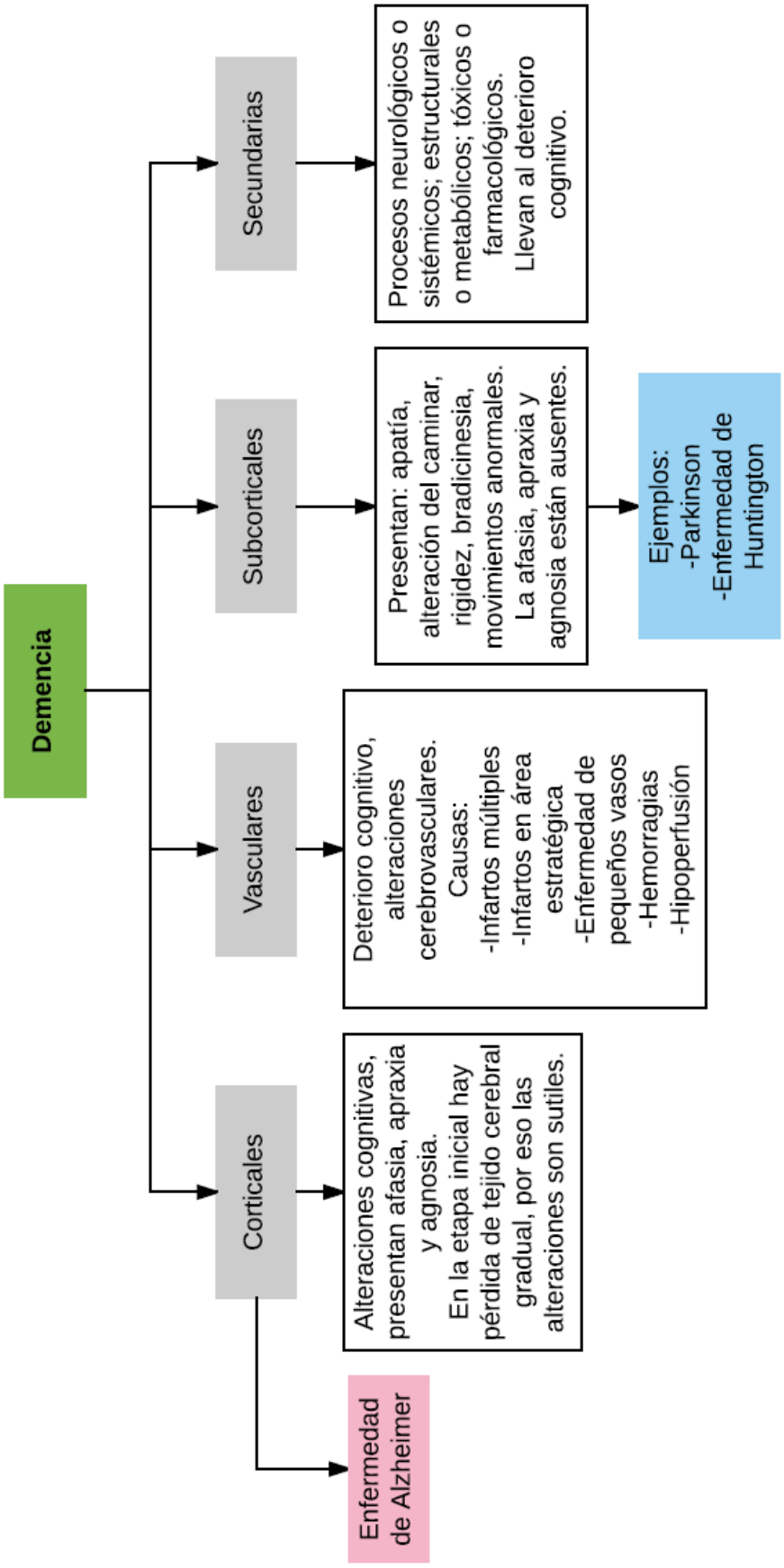


Figura A.2. Tipos de demencia.[129]

APÉNDICE B: Localización geográfica

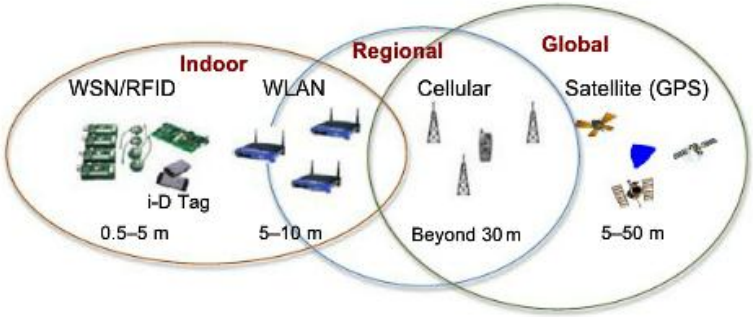


Figura B.1. Niveles de calidad, cobertura y precisión en las tecnologías de posicionamiento geográfico.[130]

Tabla B.1. Clasificación de los sistemas de posicionamiento basados en mediciones disponibles.[131]

Classification of Positioning Systems Based on Available Measurements		
Measured Quantity	**Positioning Scheme**	**Characteristic Aspect**
Angle of arrival (AOA)	Angle based	Characterizes the direction of propagation. Usually antenna arrays are required.
Received signal strength (RSS)	Range based Fingerprinting Interferometric	Measurement of the received power.
Time of arrival (TOA)	Range based	Measurement of the signal
Time difference of arrival(TDOA)	Range difference based	Measurement of the signals propagation delay difference.
Near-field ranging (NFR)	Range based	Relates the distance to the angle between the electric and magnetic fields in near-field conditions.
Radio visibility	Proximity range-free	Connectivity.
Acceleration and angular velocity	Inertial	Provides linear and angular displacement.
Earth magnetic field	Magnetic	Provides orientation information.

Tabla B.2. Comparación entre diferentes sistemas de posicionamiento.[132]

Comparison of Existing Positioning Systems				
Technology	**Measurement technique**	**Accuracy**	**Pros**	**Cons**
GPS	TDOA	10-20m	Earth scale coverage	Expensive infrastructure, only outdoor
Gallleo	TDOA	1-5m	Earth scale coverage	Expensive infrastructure, only outdoor
A-GNSS	TDOA	<5m	Country coverage	Scarce indoor accuracy
Cellular	E-OTD/OTDOA	50-500m	Country coverage	Requires synchronized base stations
Cellular	CellID	cell size	Country coverage	Scarce accuracy
WLAN	RSS-fingerprinting	1-5m	Indoor coverage, low cost	Database required for fingerprinting, low accuracy
WSN (ZigBee)	RSS	1-10m	Indoor coverage, low power consumption, low cost	Low accuracy
WSN (UWB)	TOA/TDOA/AOA	0.1-1m	Indoor coverage,high accuracy	Short range (<20m), problems in NLOS
RFID	Proximity	Conectivity range	Indoor coverage, low power consumption, low cost	Low accuracy, one tag per location
NFR	E.M. near-field characteristics	1-5m	Indoor coverage, low cost	Low frequency, large antennas
INS	Acceleration Angular rate Earth Magnetic field	1-6% o the traveled distance/ angle	Works everywhere	Position/orientation drift, magnetic disturbance in indoor

APÉNDICE C: Esquemáticos, distribución de pines y formato de envío y recepción de datos

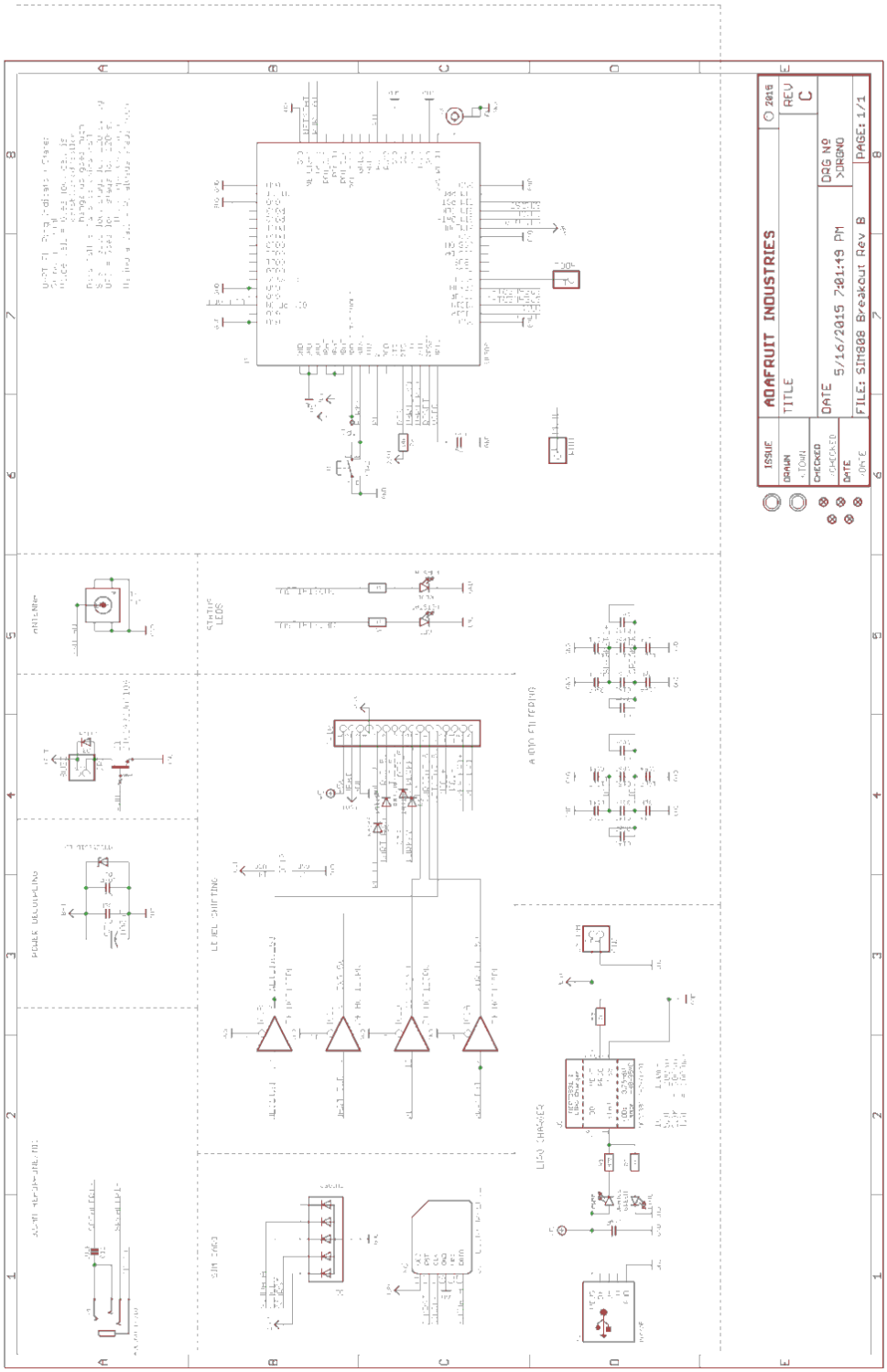


Figura C.1. Esquemático de la placa SIM808 utilizada para el prototipo de la pulsera electrónica.[133]

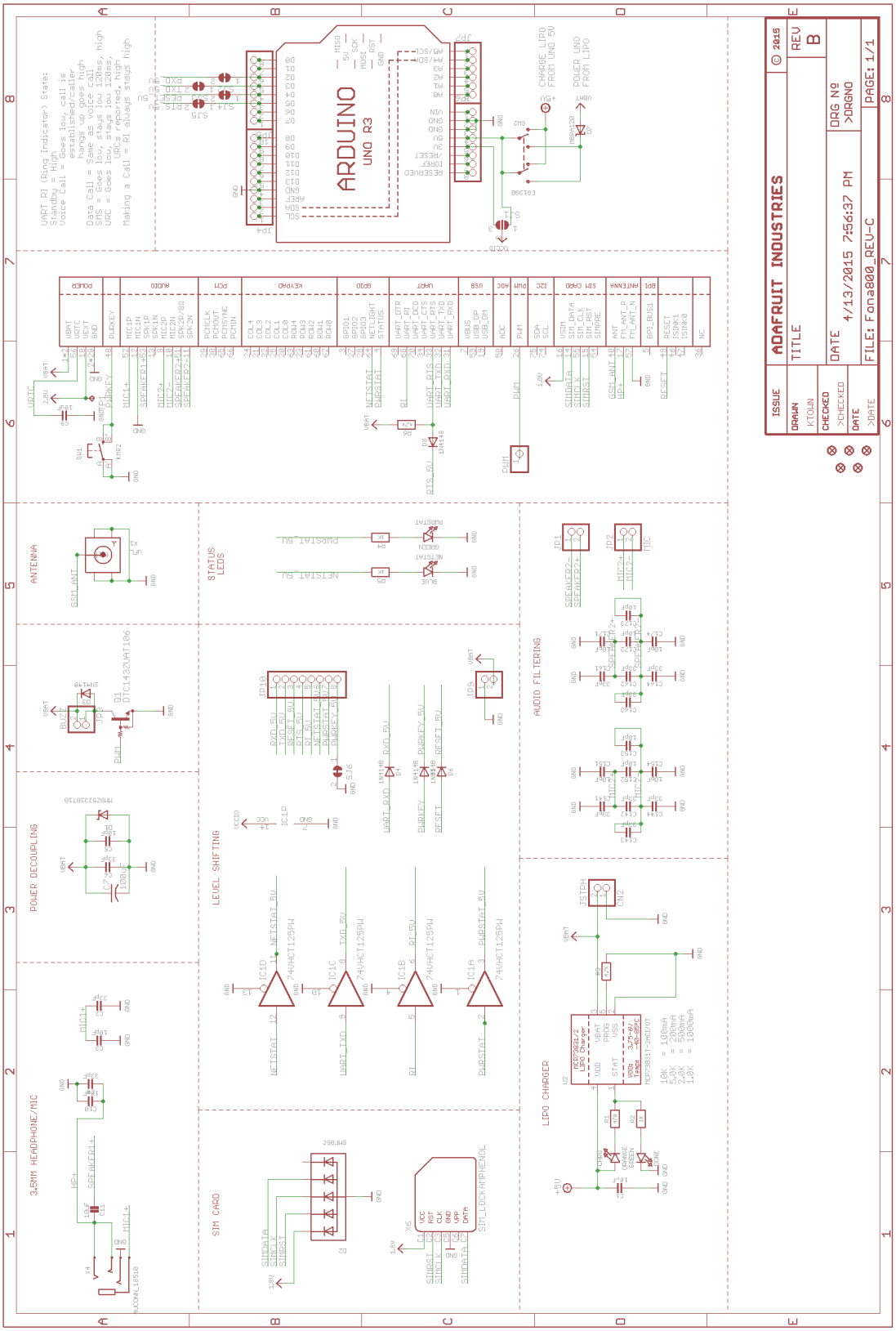


Figura C.2. Esquemático de la placa SIM800 utilizada en el prototipo del módulo procesador de datos.[134]

Figura C.3. Disposición de pines de la Raspberry Pi 2 Modelo B V1.1.[135]

Tabla C.1. Formato de los datos de la ubicación geográfica.

Formato	
IGG.MMSSWWW,LGGG.MMSSWWW,yyyymmddhhmmss,XXXX	
Variables	**Significado**
I	N (Norte) o S (Sur)
L	E (Este) u O (Oeste)
GG	Grados sexagesimales
MM	Minutos sexagesimales
SS	Segundos sexagesimales
WWW	Milisegundos sexagesimales
yyyy	Año
mm	Mes
dd	Día
hh	Horas
mm	Minutos
ss	Segundos
XXXXX	Número en formato hexadecimal

APÉNDICE D: Descripción de diagramas de flujo

Programas modulares residentes

D.1. Alerta de estado del paciente.

Los pasos del programa se describen a continuación:

1. Obtiene los campos ‘Paciente’, ‘Teléfono’, ‘Estado’, ‘Respuesta’, ‘Proceso’ desde la tabla “Resultado de análisis de las ubicaciones” en la base de datos.
2. Analiza los estados de cada paciente. Recorre las variables que contienen los datos.
3. Verifica la respuesta, si es ‘Positiva’ entonces alguien conoce el estado de peligro del paciente y verifica si finalizó el análisis.
4. Si la respuesta es ‘Negativa’, verifica el estado de la ubicación del paciente.
5. Si el estado es ‘Peligro’, se establece el mensaje “Paciente fuera de zona de seguridad”.
6. Si el estado es ‘Desconocido’, entonces se establece el mensaje “Ubicación del paciente” no recibida por un largo período de tiempo”.
7. Verifica el proceso de notificación del ‘Peligro’ del paciente.
8. Si el proceso es igual a “N/A”, notifica al cuidador mediante la aplicación y actualiza el proceso a “Aplicación” para indicar que ya ejecutó ese paso. Luego verifica si el análisis finalizó.
9. Si el proceso es igual a “Aplicación”, indica que ya se realizó la notificación mediante la aplicación móvil, notifica al cuidador por SMS y actualiza el proceso a “Mensaje de texto” para indicar que ya ejecutó ese paso. Luego verifica si el análisis finalizó.
10. Si el proceso es igual a “Mensaje de texto”, indica que ya realizó la notificación mediante SMS, luego, llama al ‘911’ y espera un período de tiempo hasta que la

llamada sea colgada, luego actualiza el proceso a ‘911’ para indicar que ya ejecutó ese paso, finalmente, verifica si el análisis terminó.

11. Si el proceso es igual a “911”, indica que se realizó la llamada al ‘911’ pero no hubo respuesta, y continúa hasta obtener una. Luego verifica si el análisis finalizó.
12. Si el análisis finalizó, espera un minuto y regresa al inicio del programa.

D.2. Análisis de coordenadas.

1. Obtiene el estado inicial de la ubicación del paciente desde la tabla “Resultado de análisis de las ubicaciones” en la base de datos.
2. Obtiene la ubicación actual y las coordenadas de la zona segura desde la tabla “Registro de ubicaciones” en la base de datos.
3. Elimina las zonas seguras que no tienen coordenadas establecidas o están inhabilitadas.
4. Sustituye los puntos cardinales de las coordenadas asociadas a las zonas seguras y a la ubicación actual.
5. Analiza las coordenadas. Recorre las variables que contienen los datos.
6. Separa las coordenadas de la ubicación en dos cadenas: latitud y longitud.
7. Separa las coordenadas de las zonas seguras en tres cadenas: latitud, longitud y distancia máxima.
8. Calcula la distancia entre las coordenadas de la ubicación actual y de cada zona segura utilizando la *fórmula de Haversine*.
9. Verifica si la distancia calculada es mayor que distancia máxima establecida en la zona.

Si la distancia es menor, indica que está dentro de la zona segura, establece el resultado igual a ‘Seguro’ y verifica si finalizó el análisis de las coordenadas.

10. Si la distancia es mayor, indica que está fuera de la zona segura, establece el resultado igual a ‘Peligro’ y verifica si finalizó el análisis de las coordenadas.
11. Si no finalizó el análisis de las coordenadas, entonces retorna al paso 6.
12. Si finalizó el análisis de las coordenadas, procede a otra etapa del análisis.
13. Obtiene datos de cada paciente y cuidador (ID, nombres y apellidos) desde la tabla “Registro de Ubicaciones” en la base de datos.
14. Analiza los resultados. Recorre las variables que contiene los datos.
15. Verifica si el paciente está en alguna zona segura, si lo está, el resultado es ‘Seguro’.
16. Si el resultado es ‘Seguro’, verifica en cual zona segura se encuentra y el cuidador que estableció dicha zona.

A. Si es el cuidador asignado (el que estableció la zona segura), actualiza el estado de la ubicación a ‘Seguro’ y la respuesta a ‘N/A’, además, lo registra como cuidador a cargo en la tabla “Resultado de análisis de las ubicaciones”.
B. Si no es el cuidador asignado, actualiza el estado de la ubicación a ‘N/A’ y la respuesta a ‘N/A’, posteriormente, establece que no es el cuidador a cargo en la tabla “Resultado de análisis de las ubicaciones”.

17. Si el resultado no es ‘Seguro’, verifica el cuidador asignado.

A. Si es el cuidador asignado, actualiza el estado de la ubicación a ‘Peligro’ y la respuesta a ‘Negativa’ en la tabla “Resultado de análisis de las ubicaciones”.
B. Si no es el cuidador asignado, actualiza el estado de la ubicación a ‘N/A’ y la respuesta a ‘N/A’.

18. Verifica si el análisis finalizó.
19. Si finalizó, espera tres (3) minutos y repite todo el programa.
20. Si no finalizó, retorna al paso ‘16’.

D.2.1. Ecuación para calcular la distancia de un punto geográfico a otro:

La distancia (d) ortodrómica entre dos coordenadas (ubicación actual y zona segura) es obtenida mediante la fórmula de Haversine, la cual está expresada como una función tangente inversa de la latitud en radianes y de la longitud en radianes:

$$\Delta Latitud = Latitud_2 - Latitud_1 \quad ; \quad \Delta Longitud = Longitud_2 - Longitud_1$$

$$a = \left[\sin\left(\frac{\Delta latitud}{2}\right)\right]^2 + \cos(Latitud_1) * \cos(Latitud_2) * \left[\sin\left(\frac{\Delta longitud}{2}\right)\right]^2$$

$$c = 2 * \operatorname{atan} 2\left(\sqrt{a}, \sqrt{(1-a)}\right)$$

$$r = Radio\ de\ la\ tierra \quad ; \quad d = rc$$

D.3. Actualización de tiempo.

1. Obtiene el estado inicial de la ubicación del paciente desde la tabla "Resultado de análisis de las ubicaciones" en la base de datos.
2. Obtiene la fecha de la última actualización de la ubicación del paciente desde la tabla "Registro de Ubicaciones" en la base de datos.
3. Obtiene la fecha actual del sistema.
4. Analiza todas las fechas de cada paciente y recorre las variables que contienen todos los datos.
5. Compara la fecha actual y la fecha obtenida desde la base de datos.
6. Actualiza el tiempo transcurrido desde la última coordenada recibida en la tabla "Resultado de análisis de las ubicaciones".
7. Verifica el estado inicial del paciente.

8. Si el estado inicial es igual a ‘N/A’ indica que no es el cuidador a cargo, y procede directamente a preguntar si finalizó el análisis.
9. Si el estado inicial es diferente, entonces procede a verificar el tiempo transcurrido. Si el mismo es mayor que cuatro (4) minutos, entonces actualiza a estado ‘Desconocido’ y la respuesta a ‘negativa’ en la tabla “Resultado de análisis de las ubicaciones”.
10. En caso de que el tiempo sea menor que cuatro (4) minutos, verifica si el estado inicial es igual a ‘peligro’, de ser así, procede a verificar si el análisis finalizó.
11. Si es diferente de ‘Peligro’, actualiza el estado a ‘Seguro’ en la tabla “Resultado de análisis de las ubicaciones”.
12. Si finalizó el análisis, espera tres (3) minutos y retorna al inicio del programa.

D.4. Recepción de mensaje.

1. Inicializa el protocolo serial por hardware. Se habilita la comunicación serial a una tasa de 115,200 baudios.
2. Activa el modo lectura de SMS del módulo GSM. Habilita mediante el comando AT correspondiente la lectura de los SMS recibidos.
3. Lee la cantidad de SMS recibidos. Se verifica la cantidad de SMS y se toma el más reciente.
4. Verifica si se ha recibido un nuevo SMS. Comprueba que se haya recibido un nuevo SMS.
5. Decisión: ¿Se ha recibido un nuevo SMS? Si se ha recibido un nuevo SMS el programa lee la longitud de caracteres del SMS, si no se ha recibido retorna al paso 3.
6. Verifica la cantidad de caracteres en el SMS. Se confirma la cantidad de caracteres en el SMS.

7. Decisión: ¿Es menor o igual a 9? Si la cantidad de caracteres es menor o igual a 9. El contenido del mensaje se guardará como parte de un proceso de cuidador, si es mayor a 9 se guardará como parte de un proceso de paciente.
8. Abre programa pulsera / Abre programa cuidador. Se ejecuta un programa dependiendo del contenido del mensaje recibido.
9. Borra el SMS recibido. Se eliminan los SMS.
10. Retardo de tiempo. Se esperan 2 segundos antes de volver a iniciar la confirmación de SMS, y vuelve al paso 3.

Programas modulares de proceso por solicitud

D.5. Mensaje cuidador.

1. El mensaje es obtenido como un parámetro que proviene desde otro programa ("Recibir Mensajes") como una cadena. Para ello, se utiliza la librería del sistema de Python ("os").
2. Separa el mensaje en dos (2) cadenas: *Información del mensaje, ID del cuidador.*
3. Verifica la información del mensaje.
4. Si la información es 'U' o 'P', indica que solicita la ubicación actual del paciente, accede a la base de datos, obtiene el número de teléfono del cuidador y la ubicación actual del paciente. Luego envía la ubicación mediante mensaje de texto.
5. Si la información es 'R', indica que recibió la notificación de 'peligro' del paciente y cambia la respuesta a positiva en la tabla de "Resultado de análisis de las ubicaciones".
6. Cierra el programa.

D.6. Mensaje pulsera.

1. El mensaje es obtenido como un parámetro que proviene desde otro programa (“Recibir Mensajes”) como una cadena. Para ello, se utiliza la librería del sistema de Python (“os”).
2. Separa el mensaje en cuatro (4) cadenas: *Latitud de la coordenada, Longitud de la coordenada, Fecha en la que fue tomada la coordenada, ID de la pulsera.*
3. Establece el formato de la fecha: "*d/m/Y H:M:S*".
4. Sube los datos a la base de datos en la tabla “Registro de Ubicaciones”, la latitud y longitud en el campo “Ubicacion_Actual” y la fecha en el campo “Ultima_Actualizacion”.
5. Cierra el programa.

D.7. Envío de mensaje.

1. Obtiene el número telefónico del cuidador y el contenido del mensaje. Los datos de número telefónico y contenido del mensaje se toman de la base de datos y se guardan en el programa.
2. Inicializa el protocolo serial por hardware. Se prepara el puerto serial para controlar periféricos a una tasa de 115,200 baudios.
3. Envía el SMS al número de teléfono asignado y el mensaje especificado. Se envía el contenido del mensaje a través de SMS.
4. Retardo de tiempo. Se espera un tiempo de 5 segundos para el envío del SMS.
5. Fin. Se finaliza y cierra el programa.

D.8. Realización de llamada.

1. Inicializa el protocolo serial por hardware. Se habilita la comunicación serial a una tasa de 115,200 baudios.
2. Se obtienen los datos del paciente en estado desconocido o de peligro.
3. Realiza una llamada de voz al Sistema Nacional de Atención a Emergencias y Seguridad 9-1-1.
4. Retado de 20 segundos. Tiempo de espera para la realización de la llamada de voz.
5. ¿La llamada fue colgada? Se mantendrá la llamada durante 20 segundos hasta que esta sea colgada, si es colgada, se establecerá como positiva la respuesta del sistema. De lo contrario retorna al paso 3.
6. Actualiza respuesta a "Positivo". Esto significa que el paciente se encuentra bajo el seguimiento del Sistema Nacional de Atención a Emergencias y Seguridad 9-1-1.
7. Fin. Termina la ejecución del programa.

D.9. Diagrama de la pulsera electrónica.

1. Se inicializa el protocolo serial UART por software. Se escogen los pines 2 (como R_X) y 3 (como T_X) de la placa de Arduino y se establece una tasa de 115,200 baudios.
2. Identificación del No. IMEI del módulo GSM, ICCID de la tarjeta SIM y el número telefónico. En esta etapa se verifica que tanto el chip SIM808 como la tarjeta SIM correspondan con su número de identificación.
3. Decisión de la tarjeta SIM. Si alguno de los números de identificación anteriores no coincide con los asignados por el procesador de datos o si no tiene tarjeta SIM, la pulsera electrónica indicará que debe reiniciarse el dispositivo con la tarjeta correcta. De lo contrario el programa continuará su ejecución.

4. Activación del GPS. Se habilita el GPS del módulo SIM808.
5. Retardo de tiempo. Se espera un tiempo de 1.5 segundos para la activación del GPS.
6. Iniciar búsqueda de arreglo satelital. Se realiza un escaneo de las señales de al menos dos satélites GPS que sirvan de referencia para determinar la ubicación geográfica del receptor.
7. Decisión sobre hallazgo de arreglo satelital. Si se ha encontrado un arreglo satelital continúa el flujo del programa. De lo contrario retorna al punto 5.
8. Guardado de los datos de la ubicación geográfica determinada. Se realiza un registro de los datos de la ubicación geográfica.
9. Organización de los datos en el formato establecido. Se ordenan los datos de la ubicación geográfica obtenida en el formato: *latitud, longitud, UTC, ID de la pulsera.*
10. Retardo de tiempo. Se espera un tiempo de cuatro (4) minutos para el envío coordinado de SMS.
11. Mensaje instantáneo. Se envía un SMS con los datos de la ubicación geográfica en el formato establecido al procesador de datos. Luego, el programa retorna al punto 6.

Aplicación móvil

D.10. Opciones de usuario.

1. Verifica la opción seleccionada por el usuario.
2. Si el usuario ha solicitado la ubicación del paciente mediante mensaje de texto, entonces obtiene el ID del cuidador desde la base de datos, establece el formato del mensaje de texto al formato establecido en el procesador de datos y finalmente envía la solicitud.

3. Si el usuario desea insertar una zona segura, procede a verificar si aún tiene zonas seguras vacías. En caso de que tenga alguna vacía procede a ejecutar la clase “Zonas seguras del paciente”, de lo contrario notifica al usuario que no tiene zonas seguras vacías.
4. Si el usuario desea editar una zona segura, procede a verificar si seleccionó alguna. De ser así, ejecuta la clase “Zonas seguras”, de lo contrario notifica al usuario que debe escoger la zona segura que desea editar.
5. Si el usuario desea eliminar una zona segura verifica si seleccionó alguna. Si se ha seleccionado alguna procede a ejecutar la clase “Zonas seguras del paciente”, de lo contrario notifica al usuario que debe escoger la zona segura que desea eliminar.
6. Si el usuario desea cambiar su clave, se ejecuta la clase “Cambio de clave”.
7. Retorna al inicio del programa.

D.11. Zonas seguras del paciente.

1. Verifica la acción seleccionada.
2. Si la acción es “eliminar” una zona segura, elimina la zona segura en la base de datos.
3. En caso de que la opción no sea “eliminar”, significa que es insertar o editar, entonces procede a verificar el lugar seleccionado como zona segura y a solicitar el diámetro de la zona.
4. Si la acción es insertar una zona segura, guarda los datos en la base de datos.
5. Si la acción es editar una zona segura, edita los datos en la base de datos.
6. Al culminar cualquier acción seleccionada, muestra las coordenadas nuevamente para que el usuario visualice los cambios realizados.

D.12. Envío de mensaje al módulo procesador de datos.

1. Solicita al teléfono los permisos necesarios para enviar mensajes de textos, y acceder a los comandos de envío de mensajes.
2. Si no recibió autorización, el programa no se puede ejecutar ya que no tiene acceso a enviar mensajes. Notifica al usuario que los permisos fueron rechazados y que ocurrió un error al enviar el mensaje de texto.
3. Por otra parte, si ha recibido autorización, configura el sistema para enviar mensajes de texto. Verifica el número de teléfono del procesador de datos y el contenido del mensaje, envía el SMS e informa al usuario que la solicitud fue enviada con éxito.

D.13. Mostrar ubicación geográfica.

1. Verifica las coordenadas que va a mostrar en el mapa.
2. Si las coordenadas corresponden a una zona segura, entonces verifica las zonas seguras en la base de datos que hay disponibles, y convierte la ubicación geográfica de formato de cadena de texto a formato de localización.
3. Si las coordenadas corresponden al paciente, obtiene las coordenadas de la ubicación del paciente desde la base de datos, y la convierte de formato de cadena de texto a formato de localización.
4. Si las coordenadas corresponden al cuidador, obtiene las coordenadas mediante el GPS del teléfono (en formato de localización).
5. Una vez obtenida alguna ubicación, se procede a crear la posición en el mapa.
6. Se crea el título de la posición.
7. Se fija el marcador (posición) en el mapa.
8. Regresa al inicio.

D.14. Recepción de mensajes del módulo procesador de datos.

1. Solicita al teléfono los permisos necesarios para recibir y leer mensajes de textos, y también para acceder a los comandos de recibir mensajes.
2. Si no recibió autorización, el programa no se puede ejecutar ya que no tiene acceso a enviar mensajes. Notifica al usuario que los permisos fueron rechazados y que ocurrió un error al enviar el mensaje de texto.
3. Si se ha recibido autorización, accede a la carpeta de mensajes de textos de entrada (recibidos).
4. Verifica el remitente del mensaje de texto.
5. Si el remitente no es el procesador de datos, entonces ignora el mensaje y regresa al inicio del programa.
6. Si el remitente es el procesador de datos, verifica el contenido del mensaje de texto.
7. Si el contenido son las coordenadas actuales del paciente, modifica la ubicación del paciente en la base de datos.
8. Si el contenido es una alerta de peligro de la ubicación del paciente, entonces notifica al usuario que éste se encuentra en peligro. Si recibió respuesta del cuidador entonces envía un mensaje de texto al procesador de datos.
9. Si no recibió una respuesta, espera 30 segundos y vuelve a notificar al cuidador hasta que reciba una.
10. Finalmente, retorna al inicio del programa.

APÉNDICE E: Programas del sistema

E.1. Pulsera electrónica.

```
#include "Adafruit_FONA.h"  // Librería del módulo SIM808.
#include <SoftwareSerial.h> // Librería para utilizar UART por Software.

// Se definen los pines para utilizar UART por Software:
#define FONA_RX 2
#define FONA_TX 3
#define FONA_RST 4
Adafruit_FONA fona = Adafruit_FONA(FONA_RST);


uint8_t readline(char *buff, uint8_t maxbuff, uint16_t timeout = 0);


// Variables globales:
int tiempo = 3*60; // Período de tiempo en segundos para el envío automático de SMS.
char sendto[21] = "8096855221"; //Número del procesador de datos.

int tiempo_c = tiempo;
char message[141];

char replybuffer[255];
uint8_t type;

//Se configura UART por Software:
SoftwareSerial fonaSS = SoftwareSerial(FONA_TX, FONA_RX);
SoftwareSerial *fonaSerial = &fonaSS;

void setup() {

  while (!Serial);

Serial.begin(115200);
Serial.println(F("Prueba funcional de la pulsera Electronica."));
Serial.println(F("Inicializando....(Puede tomarse 3 segundos.)"));

fonaSerial->begin(4800);
if (! fona.begin(*fonaSerial)) {
   Serial.println(F("Couldn't find FONA"));
   while (1);
  }
  type = fona.type();
  Serial.println(F("El dispositivo FONA ha sido detectado."));
  Serial.print(F("Encontrado."));
  switch (type) {
    case FONA800L:
      Serial.println(F("FONA 800L")); break;
    case FONA800H:
```

```
      Serial.println(F("FONA 800H")); break;
    case FONA808_V1:
      Serial.println(F("FONA 808 (v1)")); break;
    case FONA808_V2:
      Serial.println(F("FONA 808 (v2)")); break;
    case FONA3G_A:
      Serial.println(F("FONA 3G (American)")); break;
    case FONA3G_E:
      Serial.println(F("FONA 3G (European)")); break;
    default:
      Serial.println(F("???")); break;
  }

  // Presenta el número de IMEI del dispositivo.
  char imei[15] = {0}; // Asigna tamaño de la variable para guardar los 16 números del IMEI.
  uint8_t imeiLen = fona.getIMEI(imei);
  if (imeiLen > 0) {
    Serial.print("IMEI: "); Serial.println(imei);
  }


//Verificación de la Tarjeta SIM:
   fona.getSIMCCID(replybuffer);
   Serial.print(F("SIM CCID = ")); Serial.println(replybuffer);


if (String(replybuffer) != "00000000000000000000") {

  Serial.print("Esta tarjeta SIM no esta asociada al dispositivo.\n");
  Serial.print("Contacte con el soporte tecnico para resolver este problema.");
  while(1) {
  }

  }


if (String(replybuffer) == "ERROR") {

  Serial.print("No hay tarjeta SIM insertada.");

  while(1) {
  }

  }


  // Encendido del GPS.
    if (!fona.enableGPS(true))
    Serial.println(F("Encendido fallido."));
    delay(1500);
}


void loop(){

loope:
```

```
delay(1000);



// Inicio de búsqueda de un arreglo 2D o 3D:
int8_t stat;
stat = fona.GPSstatus();

      if (stat < 0) {
        Serial.println(F("Ha fallado la lectura de datos."));
      }

      if (stat == 0) {
        Serial.println(F("GPS apagado."));
      }

      if (stat == 1) {
      Serial.println(F("No se ha encontrado un arreglo."));
      goto loope;
      }

      if (stat == 2) {
        Serial.println(F("Arreglo 2D encontrado."));
      }

      if (stat == 3) {
        Serial.println(F("Arreglo 3D encontrado."));
      }

// Revisa la localización GPS.
     char gpsdata[120];
      fona.getGPS(0, gpsdata, 120);

      if (type == FONA808_V1)
        Serial.println(F("Reply in format:
mode,longitude,latitude,altitude,utctime(yyyymmddHHMMSS),ttff,satellites,speed,course"));

      else {

      Serial.println(F("Reply in format:
mode,fixstatus,utctime(yyyymmddHHMMSS),latitude,longitude,altitude,speed,course,fixmode,reserved1,HD
OP,PDOP,VDOP,reserved2,view_satellites,used_satellites,reserved3,C/N0max,HPA,VPA"));
      Serial.println(gpsdata);
      }


int n=0;
for (int i = 2; i <= 25; i++) {
message[n]= gpsdata[i];
n++;
}


n=24;
for (int i = 37; i <= 55; i++) {
message[n]= gpsdata[i];
```

```
n++;
}

//ID de la pulsera asignado.
message[43] = '8';
message[44] = '5';
message[45] = '6';
message[46] = '4';
message[47] = '5';

  while(tiempo_c >= 0) {

  delay(1000);
  Serial.print("Tiempo restante (en segundos): ");
  Serial.println(tiempo_c);
  Serial.print("\r");

  tiempo_c=tiempo_c-1;
  }

// Envío de un SMS:
  Serial.print("Numero del procesador de datos: \r");
  Serial.println(sendto); // Número de móvil del Procesador de datos.

  Serial.print("Contenido del mensaje: \r");
  Serial.println(message);

  if (!fona.sendSMS(sendto, message)) {
     Serial.println(F("No se puedo enviar el mensaje."));
  }

  else{
        Serial.println(F("Mensaje Enviado!"));
  }

tiempo_c = tiempo;

}
```

Programas del módulo procesador de datos

E.2. Programa Alerta estado del paciente.

```
1.  # -*- coding: utf-8 -*-
2.
3.  # DISEÑO DE UN SISTEMA ELECTRÓNICO PARA LOCALIZAR PERSONAS CON ALZHEIMER #
4.  # PROGRAMA: Alerta de estado del paciente.
5.
6.
7.  # Incluye en el proyecto la librería.
8.  import os   # Librería para el sistema operativo.
9.  import sys  # Librería para el acceso a algunas variables utilizadas o mantenidas por el intérprete.
10. import sqlite3 # Librería para la conexion con la base de datos.
11. import time    # Importa la librería de tiempo.
```

```
12.
13.
14.  # Recarga el sistema.
15.  reload(sys)
16.  # Selecciona la codificación del sistema.
17.  sys.setdefaultencoding('utf8')
18.
19.  global Paciente, Telefono, Estado, Respuesta, Proceso
20.
21.  # Función que inicia el proceso de análisis: def Alerta().
22.  def Alerta():
23.      os.system('clear')
24.      print("                PROGRAMA: ALERTA ESTADO")
25.      print("EMITE UNA ALERTA SI EL PACIENTE ESTA EN PELIGRO O SE DESCONOCE SU PARADERO")
26.
27.  # Carga los datos desde la base de datos.
28.      CargarDatos()
29.      Decision()
30.      if ('Peligro' in Estado) == False:
31.          print("")
32.          print("Todos los pacientes están seguros.")
33.      else:
34.          if ('Negativa' in Respuesta) == False:
35.              print("")
36.              print("Alguien conoce el estado de peligro del paciente.")
37.
38.  def CargarDatos():
39.      global Paciente, Telefono, Estado, Respuesta, Proceso
40.      Paciente = []
41.      Telefono = []
42.      Estado = []
43.      Respuesta = []
44.      Proceso = []
45.
46.      # Tabla de la base de datos.
47.      tabla='estado'
48.
49.  # Campos de los cuales se van a obtener lo datos.
50.      campos=" Paciente, Telefono, Estado, Respuesta, Proceso "
51.
52.      # Comando a ejecutar en la tabla de la base de datos.
53.      comando = "select " + campos + " from " + tabla + " where Estado=='Peligro' or Estado=='Desconocido'"
54.
55.      # Establece la conexión con la base de datos.
56.      with sqlite3.connect('Tesis.s3db') as con:
57.          con.text_factory = str
58.          # Crea un cursor para ejecutar los comandos en la tabla de la base de datos.
59.          cursor = con.cursor()
60.          # Ejecuta el comando en la tabla de la base de datos.
61.          cursor.execute(comando)
62.          for art in cursor:
63.              Paciente.append(unicode(str(art[0]),'cp1252'))
64.              Telefono.append(unicode(str(art[1]),'cp1252'))
65.              Estado.append(unicode(str(art[2]),'cp1252'))
66.              Respuesta.append(unicode(str(art[3]),'cp1252'))
67.              Proceso.append(unicode(str(art[4]),'cp1252'))
68.          cursor.close()  # Cierra el cursor creado para ejecutar comandos.
69.      con.close() # Cierra la conexión con la base de datos.
70.
71.  def Decision():
72.      Mensaje = ' '
```

```
73.     CambioProceso = ' '
74.     for Recorrido in range(len(Paciente)):
75.         if Estado[Recorrido] == 'Peligro' and Respuesta[Recorrido] == 'Negativa':
76.             Mensaje = Paciente[Recorrido] + " Fuera de la zona de seguridad."
77.
78.         elif Estado[Recorrido] == 'Desconocido' and Respuesta[Recorrido] == 'Negativa':
79.             Mensaje = "La ubicación de paciente del " + Paciente[Recorrido] + " No ha sido recibida por un largo periodo de tiempo."
80.
81.         if Mensaje != ' ':
82.             if Proceso[Recorrido] == 'N/A':
83.                 print("")
84.                 print("Proceso = 'Aplicacion'. " + Mensaje)
85.                 CambioProceso = 'APLICACION'
86.                 ProcesoCambio(CambioProceso, Recorrido)
87.             elif Proceso[Recorrido] == 'APLICACION':
88.                 print("")
89.                 print("Proceso = 'Mensaje de Texto'. " + Mensaje)
90.                 CambioProceso = 'MENSAJE DE TEXTO'
91.                 ProcesoCambio(CambioProceso, Recorrido)
92.                 os.system('sudo python Enviar_Mensajes.py ' + Telefono[Recorrido].encode('cp1252') + ' ' + Mensaje.replace(" ", "~").encode('cp1252'))
93.             elif Proceso[Recorrido] == 'MENSAJE DE TEXTO':
94.                 print("")
95.                 print("Proceso = '911'. " + Mensaje)
96.                 CambioProceso = '911'
97.                 ProcesoCambio(CambioProceso, Recorrido)
98.                 os.system('sudo python Realizar_Llamada.py ' + Paciente[Recorrido].replace(" ", "~").encode('cp1252'))
99.             elif Proceso[Recorrido] == '911':
100.                print("")
101.                print("Proceso = '911'. " + Mensaje)
102.                os.system('sudo python Realizar_Llamada.py ' + Paciente[Recorrido].replace(" ", "~").encode('cp1252'))
103.            Mensaje = ' '
104.            CambioProceso = ' '
105.
106. def ProcesoCambio(CambioProceso, ID):
107.
108.     # Tabla de la base de datos.
109.     tabla='estado'
110.
111.     # Datos a editar en la tabla.
112.     value = "Proceso='" + CambioProceso + "'"
113.
114.     # Comando a ejecutar en la tabla de la base de datos.
115.     comando = "update " + tabla + " set " + value + " where Paciente='" + Paciente[ID].encode('cp1252') +"' and Estado!='N/A'"
116.
117.     # Establece la conexion con la base de datos.
118.     with sqlite3.connect('Tesis.s3db') as con:
119.         cursor = con.cursor()       # Crea un cursor para ejecutar los comandos en la tabla de la base de datos.
120.         cursor.execute(comando)     # Ejecuta el comando en la tabla de la base de datos.
121.         cursor.close()              # Cierra el cursor creado para ejecutar comandos.
122.     con.close()
123.
124. while True:   # Bucle infinito.
125.     Alerta()  # Llama a la función principal.
126.     time.sleep(180)  # Retardo de 180 segundos (3 minutos).
```

E.3. Programa Análisis de coordenadas.

```
1.  # -*- coding: utf-8 -*-
2.
3.  # DISEÑO DE UN SISTEMA ELECTRÓNICO PARA LOCALIZAR PERSONAS CON ALZHEIMER #
4.  #PROGRAMA: ANALIZAR COORDENADAS.
5.
6.  # Incluye en el proyecto la librería.
7.  import os   # Librería para el sistema operativo.
8.  import sys  # Librería para el acceso a algunas variables utilizadas o mantenidas por el intérprete.
9.  import math # Librería para las funciones matemáticas.
10. import sqlite3  # Librería para la conexión con la base de datos.
11. import time     # Importa la librería de tiempo.
12.
13. reload(sys) # Recarga el sistema.
14. sys.setdefaultencoding('utf8')  # Selecciona la codificación del sistema.
15.
16. # Variable para guardar las distancias máximas que se puede alejar el paciente.
17. global DistanciasZonas
18. global Coordenadas  # Variable para guardar las coordenadas de las zonas del paciente.
19. global EstadoAnterior # Variable para guardar el estado anterior de la ubicación del paciente.
20. global Cuidador
21. global Paciente
22. global ResultadoFinal
23.
24. # Función que inicia el proceso de análisis: def Proceso().
25. def Proceso():
26.     CargarEstados()
27.     Datos = CargarCoordenadas() # Carga los datos desde la base de datos.
28.     global ResultadoFinal
29.     ResultadoFinal = []
30.     os.system("clear")
31.     print("              PROGRAMA: ANALIZAR COORDENADAS")
32.     print("DETERMINA LA DISTANCIA ENTRE LA UBICACION ACTUAL DEL PACIENTE Y LAS COO
    RDENADAS DE LAS ZONAS SEGURAS")
33.
34.     # Realiza el proceso de análisis para cada paciente y zonas.
35.     for Analisis in range(len(Datos[0])):
36.         # Llama la función para evaluar los datos y obtener el resultado final.
37.         Resultado = DecodificarDatos (Datos[0][Analisis], Datos[1][Analisis])
38.         ResultadoFinal.append(Resultado)
39.     SeleccionarCuidador()
40.
41. def CargarEstados():
42.     # Tabla de la base de datos.
43.     tabla='estado'
44.
45.     global EstadoAnterior
46.     EstadoAnterior = []
47.
48.     campos=" Estado "   # Campos de los cuales se van a obtener lo datos.
49.     comando = "select " + campos + " from " + tabla # Comando a ejecutar en la tabla de la base de datos.
50.     with sqlite3.connect('Tesis.s3db') as con:  # Establece la conexión con la base de datos.
51.         con.text_factory = str
52.         cursor = con.cursor()   # Crea un cursor para ejecutar los comandos en la tabla de la base de datos.
53.         cursor.execute(comando) # Ejecuta el comando en la tabla de la base de datos.
54.         for art in cursor:
55.             EstadoAnterior.append(unicode(str(art[0]),'cp1252'))
56.         cursor.close()  # Cierra el cursor creado para ejecutar comandos.
57.     con.close()         # Cierra la conexión con la base de datos.
58.
```

```
59.  def CargarCoordenadas():
60.      tabla='zona'    # Tabla de la base de datos.
61.      CoordZo = []    # Variable para almacenar las coordenadas de las zonas de seguridad.
62.      CoordAct = []   # Variable para almacenar las coordenadas de la ubicación actual.
63.
64.      global Cuidador, Paciente
65.      Cuidador = []
66.      Paciente = []
67.
68.      # Campos de los cuales se van a obtener lo datos.
69.      campos=" Ubicacion_Actual, Zona_Segura_1, Zona_Segura_2, Zona_Segura_3, Zona_Segura_4, Zona_Segu
     ra_5, Cuidador, Paciente "
70.
71.      # Comando a ejecutar en la tabla de la base de datos.
72.      comando = "select " + campos + " from " + tabla  + " order by Paciente"
73.      # Establece la conexión con la base de datos.
74.      with sqlite3.connect('Tesis.s3db') as con:
75.          con.text_factory = str
76.
77.          # Crea un cursor para ejecutar los comandos en la tabla de la base de datos.
78.          cursor = con.cursor()
79.
80.          # Ejecuta el comando en la tabla de la base de datos.
81.          cursor.execute(comando)
82.          for art in cursor:
83.              CoordAct.append(unicode(str(art[0]),'cp1252'))
84.              CoordZo.append([unicode(str(art[1]),'cp1252'), unicode(str(art[2]),'cp1252'),unicode(str(art[3]),'cp1252')
     , unicode(str(art[4]),'cp1252'), unicode(str(art[5]),'cp1252')])
85.              Cuidador.append(unicode(str(art[6]),'cp1252'))
86.              Paciente.append(unicode(str(art[7]),'cp1252'))
87.          cursor.close()  # Cierra el cursor creado para ejecutar comandos.
88.      con.close()         # Cierra la conexión con la base de datos.
89.
90.      for Recorrido in range(len(CoordZo)):
91.          # Recorrer la variable que almacena las coordenadas de las zonas seguras.
92.          while ('Inhabilitada' in CoordZo[Recorrido]) == True:
93.              CoordZo[Recorrido].remove('Inhabilitada')   # Eliminar las zonas de seguridad que no tengan coordenad
     as asignadas.
94.          while (' ' in CoordZo[Recorrido]) == True:      # Recorrer la variable que almacena las coordenadas de las z
     onas seguras.
95.              CoordZo[Recorrido].remove(' ')              # Eliminar las zonas de seguridad que no tengan coordenadas as
     ignadas.
96.          while ('None' in CoordZo[Recorrido]) == True:   # Recorrer la variable que almacena las coordenadas de la
     s zonas seguras.
97.              CoordZo[Recorrido].remove('None')           # Eliminar las zonas de seguridad que no tengan coordenadas
      asignadas.
98.
99.      # Recorrido para reemplaza el polo por el signo correspondiente.
100.     for Recorrido in range(len(CoordZo)):
101.         for RecorridoInterno in range(len(CoordZo[Recorrido])):
102.             CoordZo[Recorrido][RecorridoInterno] = CoordZo[Recorrido][RecorridoInterno].replace('N','')
103.             CoordZo[Recorrido][RecorridoInterno] = CoordZo[Recorrido][RecorridoInterno].replace('S','-')
104.             CoordZo[Recorrido][RecorridoInterno] = CoordZo[Recorrido][RecorridoInterno].replace('E','')
105.             CoordZo[Recorrido][RecorridoInterno] = CoordZo[Recorrido][RecorridoInterno].replace('W','-')
106.
107.     # Recorrido para reemplaza el polo por el signo correspondiente.
108.     for Recorrido in range(len(CoordAct)):
109.         CoordAct[Recorrido] = CoordAct[Recorrido].replace('N','')
110.         CoordAct[Recorrido] = CoordAct[Recorrido].replace('S','-')
111.         CoordAct[Recorrido] = CoordAct[Recorrido].replace('E','')
112.         CoordAct[Recorrido] = CoordAct[Recorrido].replace('W','-')
```

```
113.
114.     # Retorna las coordenadas actuales de los paciente y las coordenadas de sus zonas seguras.
115.     return CoordAct, CoordZo
116.
117. def DecodificarDatos(CoordAct, CoordZo):
118.     # Variable para guardar los valores en la variable tipo lista "Coordenadas".
119.     RecorridoCoordenadas=0
120.     # Determina la cantidad de zonas seguras.
121.     Cantidad = len(CoordZo)
122.
123.     # Obtiene los datos de la ubicación actual.
124.     # Elimina el string "Lat.:".
125.     CoordAct = CoordAct.split('Lat.: ')[1]
126.
127.     # Obtiene la latitud de la ubicación actual del paciente.
128.     CoordActLat = CoordAct.split(' ;')[0]
129.     # Obtiene la latitud de la ubicación actual del paciente.
130.     CoordActLon = CoordAct.split(' ; Long.: ')[1]
131.
132.     # Establece que es la variable de todo el programa.
133.     global DistanciasZonas
134.     # Establece que es la variable de todo el programa.
135.     global Coordenadas
136.
137.     # Declara la variable para guardar los valores de la latitud y la longitud.
138.     Coordenadas=[i for i in range(Cantidad * 2)]
139.     # Declara la variable para guardar los valores de las distancias máximas de las zonas seguras.
140.     DistanciasZonas=[i for i in range(Cantidad)]
141.
142. # Bucle para descomprimir los valores de la zonas seguras (Latitud, Longitud, Distancia).
143.     for RecorridoCoordZo in range(Cantidad):
144.         # Elimina el string "Lat.:".
145.         CoordZo[RecorridoCoordZo] = CoordZo[RecorridoCoordZo].split('Lat.: ')[1]
146.         # Obtiene la latitud de la zona segura.
147.         Coordenadas[RecorridoCoordenadas] = CoordZo[RecorridoCoordZo].split(' ;')[0]
148.         # Elimina el string " ; Long.: ".
149.         CoordZo[RecorridoCoordZo] = CoordZo[RecorridoCoordZo].split(' ; Long.: ')[1]
150.         # Obtiene la longitud de la zona segura.
151.         Coordenadas[RecorridoCoordenadas+1] = CoordZo[RecorridoCoordZo].split(' ;')[0]
152.         # Elimina el string " ; Dist. Máx.: ".
153.         CoordZo[RecorridoCoordZo] = CoordZo[RecorridoCoordZo].split(' ; Dist. Max.: ')[1]
154.         # Obtiene la distancia máxima de la zona segura.
155.         DistanciasZonas[RecorridoCoordZo] = CoordZo[RecorridoCoordZo].split(' m')[0]
156.         # Incrementa la variable para cambiar de una zona a otra.
157.         RecorridoCoordenadas = RecorridoCoordenadas + 2
158. # Llama la función que calcula la distancia entre coordenadas y obtener el estado de la ubicación.
159.     ResultFinal = AnalizarDatos(CoordActLat, CoordActLon)
160.
161.     return ResultFinal  # Retorna el resultado del proceso.
162.
163. def AnalizarDatos(CoordActLat, CoordActLon):
164.     # Verifica si todos los campos fueron llenados.
165.     r=6370  # Radio de la tierra en "km".
166.
167.     # Rutina para buscar las coordenadas.
168.     lat1= (float(CoordActLat)* (math.pi))/180   # Convierte Latitud 1 a radianes.
169.     lon1= (float(CoordActLon)* (math.pi))/180   # Convierte Longitud 1 a radianes.
170.
171. # Variable para realizar el recorrido en Coordenadas para "Latitud" y "Longitud".
172.     RecorridoCoord = 0
173.     # Determina la cantidad de zonas que existen.
```

```
174.    Cantidad = len(DistanciasZonas)
175.    # Declara la variable para la cantidad de zonas existentes.
176.    Resultado = [i for i in range(Cantidad)]
177.
178. # Realiza un recorrido por todas las zonas seguras.
179.    for Recorrido in range(Cantidad):
180.        # Convierte Latitud 2 a radianes.
181.        lat2=(float(Coordenadas[RecorridoCoord])* (math.pi))/180
182.        # Convierte Longitud 2 a radianes.
183.        lon2=(float(Coordenadas[RecorridoCoord+1])* (math.pi))/180
184.        # Incrementa la variable que obtiene la latitud y longitud de las zonas, para cambiar de una a otra.
185.        RecorridoCoord = RecorridoCoord + 2
186.
187.        # Realiza la resta de Latitud 2 - Latitud 1.
188.        diflat = sum([-lat1,lat2])
189.        # Realiza la resta de Longitud 2 - Longitud 1.
190.        diflon = sum([-lon1,lon2])
191.
192.        # Realiza la operación para obtener "a".
193.        a=(math.pow((math.sin(diflat/2)),2)) + (math.cos(lat1)) * (math.cos(lat2)) * (math.pow((math.sin(diflon/2)
    ),2))
194.
195.        # Realiza la operación para obtener "c".
196.        c = 2 * math.atan2(math.sqrt(a),math.sqrt(1-a))
197.
198.        # Realiza la operación para obtener la distancia entre ambas coordenadas.
199.        d = r*c
200.
201.        # Convierte de kilómetro a metro.
202.        d = d*1000
203.
204.        if (float(d) < float(DistanciasZonas[Recorrido])):  # Determina si la distancia esta dentro de la zona segura.
205.            Resultado[Recorrido]="Seguro"   # Si la zona es segura, le asigna su valor.
206.
207. # Declara la variable que almacena el estado de la ubicación final.
208.    ResultFinal = ""
209.
210.    for Prueba in Resultado:
211.        # Pregunta si el paciente está dentro de alguna zona.
212.        if (Prueba == "Seguro"):
213.            # Estable el estado final a "Seguro".
214.            ResultFinal = Prueba
215.
216.        # Verifica el estado final, si no esta seguro entonces procede a ejecutar la acción de alerta.
217.    if (ResultFinal != "Seguro"):
218.        ResultFinal = "Peligro"
219.
220.    return ResultFinal
221.
222. # Función para asignar el cuidador.
223. def SeleccionarCuidador():
224.    Inicio = 0
225.    Final = 0
226.    Conclusion = ""
227.    CuidadorAsignado = ""
228.
229.    for Recorrido in range(len(Paciente)):
230.        if Paciente[Recorrido] != Paciente[Recorrido -1]:
231.            Inicio = Paciente.index(Paciente[Recorrido])
232.
```

```
            Final = Paciente.count(Paciente[Recorrido]) + Inicio

            Conclusion = ""
            if ("Seguro" in ResultadoFinal[Inicio:Final]) == True:
                for Recorrido2 in range(Inicio,Final):
                    if ResultadoFinal[Recorrido2] == "Seguro":
                        Conclusion = "Seguro"
                    else:
                        Conclusion = "N/A"

                    if EstadoAnterior[Recorrido] != Conclusion and EstadoAnterior[Recorrido] !="Desconocido":
                        # Llama la función para editar el estado de ubicación en la base de datos.
                        EditarEstado(Cuidador[Recorrido2], Conclusion)
                print("")
                print(Paciente[Recorrido] + ". Estado: 'Seguro'")

            else:
                CuidadorAsignado = CargarCuidadorACargo(Paciente[Recorrido])
                for Recorrido2 in range(Inicio,Final):
                    if Cuidador[Recorrido2] == CuidadorAsignado:
                        Conclusion = "Peligro"
                    else:
                        Conclusion = "N/A"

                    if EstadoAnterior[Recorrido] != Conclusion and EstadoAnterior[Recorrido] !="Desconocido":
                        # Llama la función para editar el estado de ubicación en la base de datos.
                        EditarEstado(Cuidador[Recorrido2], Conclusion)
                print("")
                print(Paciente[Recorrido] + ". Estado: 'Peligro'")

def CargarCuidadorACargo(Paci):
    tabla='estado'  # Tabla de la base de datos.
    Dato = ""
    campos=" Cuidador " # Campos de los cuales se van a obtener lo datos.
    comando = "select " + campos + " from " + tabla + " where Paciente='" + Paci.encode('cp1252') + "' and A_Cargo_del_Paciente='Si'"                                    # Comando a ejecutar en la tabla de la base de datos
    with sqlite3.connect('Tesis.s3db') as con:  # Establece la conexión con la base de datos.
        con.text_factory = str
        cursor = con.cursor()   # Crea un cursor para ejecutar los comandos en la tabla de la base de datos.
        cursor.execute(comando) # Ejecuta el comando en la tabla de la base de datos.
        for art in cursor:
            Dato = unicode(str(art[0]),'cp1252')
        cursor.close() # Cierra el cursor creado para ejecutar comandos.
    con.close()       # Cierra la conexión con la base de datos.

    return Dato

# Función para ingresar los datos.
def EditarEstado(Cuid, Conclusion):
    Respuesta = ""
    valueadicional = ""
    if Conclusion == "Seguro":  # Verifica el estado final, si no esta seguro entonces procede a ejecutar la acción de alerta.
        Cargo = "Si"
        Respuesta = "N/A"
        valueadicional = ", Proceso='N/A'"

    elif Conclusion == "Peligro":   # Verifica el estado final, si no esta seguro entonces procede a ejecutar la acción de alerta.
        Cargo = "Si"
        Respuesta = "Negativa"
```

```
        valueadicional = ""

    else:
        Respuesta = "N/A"
        Cargo = "No"
        valueadicional = ", Proceso='N/A'"


    tabla='estado'  # Tabla de la base de datos.
    # Datos a editar en la tabla.
    value = "A_Cargo_del_Paciente='" + Cargo + "', Estado='" + Conclusion + "', Respuesta='" + Respuesta + "'" "'" + valueadicional
    # Comando a ejecutar en la tabla de la base de datos.
    comando = "update " + tabla + " set " + value + "where Cuidador='" + Cuid.encode('cp1252') + "'"

# Establece la conexión con la base de datos.
    with sqlite3.connect('Tesis.s3db') as con:
        # Crea un cursor para ejecutar los comandos en la tabla de la base de datos.
        cursor = con.cursor()
        # Ejecuta el comando en la tabla de la base de datos.
        cursor.execute(comando)
        # Cierra el cursor creado para ejecutar comandos.
        cursor.close()
    con.close()


while True: # Bucle infinito.
    Proceso()   # Llama a la función principal.
    time.sleep(5)  # Retardo de 180 segundos (3 minutos).
```

E.4. Programa Actualizar tiempo.

```
 -*- coding: utf-8 -*-

# DISEÑO DE UN SISTEMA ELECTRÓNICO PARA LOCALIZAR PERSONAS CON ALZHEIMER #
# PROGRAMA: ACTUALIZAR TIEMPO.

# Incluye en el proyecto la librería.
import os   # Librería para el sistema operativo.
import sys  # Librería para el acceso a algunas variables utilizadas o mantenidas por el intérprete.
import math # Librería para las funciones matemáticas.
import sqlite3  # Librería para la conexión con la base de datos.
import time as retardo  # Importa la librería de tiempo.
from datetime import datetime, date, time, timedelta

global Fecha1, EstadoAnterior, Paciente

# Función que inicia el proceso de análisis.
def Proceso():
    CargarEstados()
    CargarFecha()
    DeterminarTiempo()

def CargarFecha():
    # Tabla de la base de datos.
    tabla='zona'
    global Fecha1
```

```
26.     Fecha1=[]
27.     # Campos de los cuales se van a obtener lo datos.
28.     campos=" Ultima_Actualizacion "
29.     # Comando a ejecutar en la tabla de la base de datos.
30.     comando = "select " + campos + " from " + tabla
31.     # Establece la conexión con la base de datos.
32.     with sqlite3.connect('Tesis.s3db') as con:
33.         con.text_factory = str
34.         # Crea un cursor para ejecutar los comandos en la tabla de la base de datos.
35.         cursor = con.cursor()
36.         # Ejecuta el comando en la tabla de la base de datos.
37.         cursor.execute(comando)
38.         for art in cursor:
39.             Fecha1.append(unicode(str(art[0]),'cp1252'))
40.             # Cierra el cursor creado para ejecutar comandos.
41.         cursor.close()
42.     con.close()
43.
44. def DeterminarTiempo():
45.     Fecha2 = datetime.now()
46.     Campos = ""
47.     os.system("clear")
48.     print("                PROGRAMA: ACTUALIZAR TIEMPO")
49.     print("DETERMINA EL TIEMPO TRANSCURRIDO DESDE LA ULTIMA COORDENADA RECIBIDA"
    )
50.     for Analisis in range(len(Fecha1)):
51.         Fecha1[Analisis] = datetime.strptime(Fecha1[Analisis], "%d/%m/%Y %H:%M:%S")
52.         Diferencia = Fecha2 - Fecha1[Analisis]
53.
54.         Campos = "Tiempo_Actualizacion=' Dias: " + str(Diferencia.days) + ", Minutos: " + str((Diferencia.second
    s/60)) + "' "
55.         if Diferencia.days == 0 and (Diferencia.seconds/60) <= 4 and EstadoAnterior[Analisis] != "Peligro" and
    EstadoAnterior[Analisis] != "N/A":
56.             #Campos = Campos + ", Estado='Seguro', Respuesta = 'N/A'".
57.             print("")
58.             print(Paciente[Analisis])
59.             print("Dias: " + str(Diferencia.days) + ", Minutos: " + str((Diferencia.seconds/60))) # + ", Estado: Segur
    o")
60.         elif (Diferencia.days > 0 or (Diferencia.seconds/60) > 4) and EstadoAnterior[Analisis] != "Desconocido" a
    nd EstadoAnterior[Analisis] != "N/A":
61.             #Campos = Campos + ", Estado='Desconocido', Respuesta = 'Negativa'".
62.             print("")
63.             print(Paciente[Analisis])
64.             print("Dias: " + str(Diferencia.days) + ", Minutos: " + str((Diferencia.seconds/60)))
65.
66.         EditarEstado(Analisis, Campos)
67.
68. def CargarEstados():
69.     # Tabla de la base de datos.
70.     tabla='estado'
71.
72.     global EstadoAnterior, Paciente
73.     EstadoAnterior = []
74.     Paciente = []
75.
76.     # Campos de los cuales se van a obtener lo datos.
77.     campos=" Estado, Paciente "
78.     # Comando a ejecutar en la tabla de la base de datos.
79.     comando = "select " + campos + " from " + tabla
80.     # Establece la conexión con la base de datos.
81.     with sqlite3.connect('Tesis.s3db') as con:
```

```
82.         con.text_factory = str
83.         # Crea un cursor para ejecutar los comandos en la tabla de la base de datos.
84.         cursor = con.cursor()
85.         # Ejecuta el comando en la tabla de la base de datos.
86.         cursor.execute(comando)
87.         for art in cursor:
88.             EstadoAnterior.append(unicode(str(art[0]),'cp1252'))
89.             Paciente.append(unicode(str(art[1]),'cp1252'))
90.             # Cierra el cursor creado para ejecutar comandos.
91.         cursor.close()
92.         # Cierra la conexión con la base de datos.
93.     con.close()
94.
95. # Funcion para ingresar los datos.
96. def EditarEstado(Analisis, value):
97.     # Tabla de la base de datos.
98.     tabla='estado'
99.     # Comando a ejecutar en la tabla de la base de datos.
100.    comando = "update " + tabla + " set " + value + "where Paciente='" + Paciente[Analisis].encode('cp1252') + "'
    "
101.
102. # Establece la conexión con la base de datos.
103.
104.    with sqlite3.connect('Tesis.s3db') as con:
105.        # Crea un cursor para ejecutar los comandos en la tabla de la base de datos.
106.        cursor = con.cursor()
107. # Ejecuta el comando en la tabla de la base de datos.
108.        cursor.execute(comando)
109.        # Cierra el cursor creado para ejecutar comandos.
110.        cursor.close()
111.    con.close()
112.
113. while True: # Bucle infinito.
114.    Proceso()   # Llama a la función principal.
115.    retardo.sleep(180) # Retardo de 180 segundos (3 minutos).
```

E.5. Programa Recibir mensajes.

```
1.  # -*- coding: utf-8 -*-
2.
3.  # DISEÑO DE UN SISTEMA ELECTRÓNICO PARA LOCALIZAR PERSONAS CON ALZHEIMER #
4.  #PROGRAMA: RECIBIR MENSAJES.
5.
6.  import os   # Librería para el sistema operativo.
7.  import sys  # Librería para el acceso a algunas variables utilizadas o mantenidas por el intérprete.
8.  import time # Importa la librería de tiempo.
9.  import serial #Librería para las funciones de comunicación serial.
10.
11. ser = serial.Serial('/dev/ttyAMA0', 115200)
12. time.sleep(0.5)
13.
14. while True:
15.
```

```
    if ser.isOpen():
        ser.close()
    ser.open()
    ser.isOpen()


    ser.close()
    time.sleep(0.1)
    ser.open()
    time.sleep(0.1)
    ser.write("AT+CMGF=1\r")


    os.system("clear")
    print("                    PROGRAMA: RECIBIR MENSAJES")
    print("RECIBE LOS MENSAJES Y DETERMINA SI PROVIENE DE LA PULSERA O DEL CUIDADOR
")
    print("")

    time.sleep(1)



    time.sleep(0.1)
    #Lectura de la cantidad total de SMS's.
    ser.write("AT+CPMS?\r")
    time.sleep(0.1)
    res = ser.readline() #Linea 1.
    time.sleep(0.1)
    res = ser.readline() #Linea 2.
    time.sleep(0.1)
    res = ser.readline() #Linea 3.
    time.sleep(0.1)
    res = ser.readline() #Linea 4.
    time.sleep(0.1)
    res_cad=res[12:13]
    time.sleep(0.1)
    #res_cad=0
    time.sleep(0.5)
    print(res_cad)
    num=int(res_cad) if res_cad else None
    print(type(res_cad))

    print ('Cantidad de SMSs: '+str(num))

    if num > 0:

        ser.close()
        time.sleep(1)
        ser.open()
        time.sleep(0.1)
        ser.write("AT+CMGR=1"+"\r") #Activa el modo de lectura.
        time.sleep(1)

        response_1 = ser.readline()
        responsecad_1=response_1[0:9]
        time.sleep(1)
        print ('Enviando solicitud para leer SMS...')
        print ('Tomando datos del ultimo SMS...')

        response_2 = ser.readline()
```

```
76.         responsecad_2=response_2[35:52]
77.         print ('Fecha de llegada del SMS a la pulsera (DD/MM/YY): '+ responsecad_2 +'\n')
78.
79.         response_3 = ser.readline() #Guarda los datos de la ubicación geográfica en la variable response_3.
80.         tam = len(response_3)
81.
82.
83.         if tam > 9:
84.
85.             Mensaje = response_3[0:45]
86.             response = ser.readline()
87.
88.             response_4 = ser.readline()
89.             responsecad_4=response_4[0:2]
90.
91.             # Ejecuta el Programa Mensaje_Pulsera.py si se detecta que proviene de la pulsera electrónica.
92.             os.system('sudo python Mensaje_Pulsera.py ' + Mensaje)
93.             time.sleep(5)
94.
95.         if tam <=9:
96.
97.             Mensaje = response_3[0:8]
98.             # Ejecuta el Programa Mensaje_Cuidador.py si se detecta que proviene de la aplicación móvil.
99.             print('Se ha recibido SMS de un cuidador.')
100.            os.system('sudo python Mensaje_Cuidador.py ' + Mensaje)
101.            print ('Borrando SMSs...')
102.
103.        ser.write("AT+CMGD=1"+"\r")
104.        time.sleep(1)
105.        print("")
106.        print ('Lectura de datos finalizada correctamente.')
107.        time.sleep(15)
108.
109.    if num == 0:
110.
111.        print ('No hay mensajes recibidos.')
```

E.6. Mensaje cuidador.

```
1.  # -*- coding: utf-8 -*-
2.
3.  # DISEÑO DE UN SISTEMA ELECTRÓNICO PARA LOCALIZAR PERSONAS CON ALZHEIMER #
4.  #PROGRAMA: MENSAJE CUIDADOR.
5.  import os   # Librería para el sistema operativo.
6.  import sys  # Librería para el acceso a algunas variables utilizadas o mantenidas por el intérprete.
7.  import sqlite3  # Librería para la conexión con la base de datos.
8.
9.  if(len(sys.argv) > 1):  # Comprueba si hay datos provenientes de otro archivo.
10.     Mensaje  = sys.argv[1]  # Le asigna el primer valor proveniente de un archivo externo a la variable.
11.
12. # Decodificar datos.
13. ID=''
14. Texto = ''
15.
16. """
17. X=U, cuando solicito ubicación del paciente.
18. X=S, cuando la ubicación solicitada fue recibida por el cuidador.
```

```
X=P, cuando la ubicación solicitada no fue recibida por el cuidador.
X=R, cuando el cuidador recibe la alerta de peligro
"""

def ModificarEstado():
    comando = "update estado set Respuesta='Positiva' where Cuidador like '%Ref. #" + ID + ": %'"
    # Establece la conexion con la base de datos.
    with sqlite3.connect('Tesis.s3db') as con:
        # Crea un cursor para ejecutar los comandos en la tabla de la base de datos.
        cursor = con.cursor()
        # Ejecuta el comando en la tabla de la base de datos.
        cursor.execute(comando)
        # Cierra el cursor creado para ejecutar comandos.
        cursor.close()
        # Cierra la conexión con la base de datos.
    con.close()


def CargarDatos():
    Ubicacion = ' '
    Telefono = ' '

    tabla="zona"  # Tabla donde se insertarán los datos.
    # Comando a ejecutar en la tabla de la base de datos.
    comando = "select Ubicacion_Actual from " + tabla + " where Cuidador like '%Ref. #" + ID + ": %'"
    # Establece la conexión con la base de datos.
    with sqlite3.connect('Tesis.s3db') as con:
        con.text_factory = str
        cursor = con.cursor()       # Crea un cursor para ejecutar los comandos en la tabla de la base de datos.
        cursor.execute(comando)     # Ejecuta el comando en la tabla de la base de datos.
        for art in cursor:
            Ubicacion = unicode(str(art[0]),'cp1252')
        cursor.close()  # Cierra el cursor creado para ejecutar comandos.
    con.close()         # Cierra la conexión con la base de datos.

    tabla="cuidador"    # Tabla donde se insertarán los datos
    # Comando a ejecutar en la tabla de la base de datos.
    comando = "select Telefono from " + tabla + " where ID ='" + ID + "'"
    # Establece la conexión con la base de datos.
    with sqlite3.connect('Tesis.s3db') as con:
        con.text_factory = str
        # Crea un cursor para ejecutar los comandos en la tabla de la base de datos.
        cursor = con.cursor()
        # Ejecuta el comando en la tabla de la base de datos.
        cursor.execute(comando)
        for art in cursor:
            Telefono = unicode(str(art[0]),'cp1252')
        cursor.close()  # Cierra el cursor creado para ejecutar comandos.
    con.close() # Cierra la conexión con la base de datos.

    return Telefono, Ubicacion

try:
    # Separa los datos del mensaje.
    Texto = Mensaje.split(',')[0]
    ID = Mensaje.split(',')[1]

    if Texto == 'U' or Texto == 'P':
        print("")
        print("Mensaje proveniente del cuidador.")
        print('Enviar/Reenviar ubicación del paciente.')
```

```
80.         Datos = CargarDatos()
81.         Prueba = Datos[1].replace(" ", "~")
82.         Prueba = Prueba.replace(";", "-")
83.         os.system('sudo python Enviar_Mensajes.py ' + Datos[0] + ' ' + Prueba)
84.     elif Texto == 'S':
85.         print("")
86.         print("Mensaje proveniente del cuidador.")
87.         print('Ubicación del paciente recibida con éxito.')
88.     elif Texto == 'R':
89.         print("")
90.         print("Mensaje proveniente del cuidador.")
91.         print('Alerta del estado del paciente recibida con exito, cambiar respuesta a positiva.')
92.         ModificarEstado()
93. except:
94.     pass
```

E.7. Mensaje pulsera.

```
1.  # -*- coding: utf-8 -*-
2.
3.  # DISEÑO DE UN SISTEMA ELECTRÓNICO PARA LOCALIZAR PERSONAS CON ALZHEIMER #
4.  #PROGRAMA: MENSAJE PULSERA.
5.
6.  # Incluye en el proyecto las librerías.
7.  import os   # Librería para el sistema operativo.
8.  import sys  # Librería para el acceso a algunas variables utilizadas o mantenidas por el intérprete.
9.  from Tkinter import *   # Librería para la interfaz gráfica.
10. import ttk as ttk   # Librería que importa todos los módulos ttk.
11. import tkMessageBox as msj # Librería que importa los mensajes.
12. import sqlite3          # Librería para la conexión con la base de datos.
13.
14. # Llama las funciones de tkinter.
15. master = Tk()
16.
17. global table, ID, ComandoBusqueda, tree # Declara las variables globales para toda la ventana.
18. ID='0' # Le asigna el valor "cero" para indicar que ningun dato ha sido seleccionado.
19.
20. NombreTabla = StringVar() # Variable que muestra el nombre de la tabla seleccionada.
21.
22. if(len(sys.argv) > 1):    # Comprueba si hay datos provenientes de otro archivo.
23.     table  = sys.argv[1]  # Le asigna el primer valor proveniente de un archivo externo a la variable.
24.
25. if(len(sys.argv) > 2):    # Comprueba si hay DOS datos provenientes de otro archivo.
26.     # Le asigna el segundo valor proveniente de un archivo externo a la variable, y lo decodifica.
27.     ComandoBusqueda = " " + sys.argv[2].replace("~", " ")
28. else:
29.     ComandoBusqueda = ""  # De lo contrario le asigna un valor inicial.
30.
31. # Función para verificar si se seleccionó algún dato.
32. def ComprobarSeleccion():
33.     if ID == '0':
34.         msj.showwarning ("","Debe seleccionar un dato primero") # Despliega un mensaje de advertencia.
35.         return False  # Retorna el valor false.
36.     else:
37.         return True
```

E.8. Envío de mensaje.

```
1.  # -*- coding: utf-8 -*-
2.
3.  # DISEÑO DE UN SISTEMA ELECTRÓNICO PARA LOCALIZAR PERSONAS CON ALZHEIMER #
4.  # PROGRAMA: ENVIAR MENSAJES.
5.  # Incluye en el proyecto la librería.
6.  import os   # Librería para el sistema operativo.
7.  import sys  # Librería para el acceso a algunas variables utilizadas o mantenidas por el intérprete.
8.  from Tkinter import * # Librería para la interfaz gráfica.
9.
10. # Llama las funciones de tkinter.
11. master = Tk()
12.
13. global Busqueda # Variable global para el comando de búsqueda.
14. Busqueda = ""   # Le asigna un valor inicial a la variable.
15.
16. # Evento "Cerrar" de la ventana/formulario.
17. def Cerrar(event=None):
18.     master.destroy()  # Cierra la ventana.
19.     os.system('sudo python menu.py estado ' + Busqueda) # Abre el formulario "menu".
20.
21. # Funcion del boton Aceptar.
22. def Aceptar(event=None):
23.     Comando = "" # Variable en la que se va a guardar el comando.
24.     if TextId.get() != "":  # Verifica si el cuadro de texto tiene alguna información.
25.         campo = "ID" # Campo que corresponde al cuadro de texto.
26.         Comando = Comando + " or " + campo + "
```

E.9. Realización de llamada.

```
1.  # -*- coding: utf-8 -*-
2.
3.  # DISEÑO DE UN SISTEMA ELECTRÓNICO PARA LOCALIZAR PERSONAS CON ALZHEIMER #
4.  # PROGRAMA: REALIZAR LLAMADA.
5.
6.  import os    # Librería para el sistema operativo.
7.  import sys   # Librería para el acceso a algunas variables utilizadas o mantenidas por el intérprete.
8.  import serial # Libreria del protocolo Serial.
9.  import time  # Librería de tiempo.
10. import sqlite3 #Librería de la base de datos.
11.
12. Paciente = ' '
13.
14. # Comprueba si hay datos provenientes de otro archivo.
15. if(len(sys.argv) > 1):
16.     # Le asigna el primer valor proveniente de un archivo externo a la variable.
17.     Paciente  = unicode(str(sys.argv[1].replace("~", " ")),'cp1252')
18.
19. ser = serial.Serial('/dev/ttyAMA0', 115200)
20.
21. ser.write("XXXXXXXXXX;\r")  # Número Telefónico: 911.
22.
23. time.sleep(15)
24.
25. ser.write("ATH\r")
```

```
26.
27.  def ModificarEstado():
28.      comando = "update estado set Respuesta='Positiva' where Paciente = '" + Paciente.encode('cp1252') + "' and
     Estado != 'N/A'"
29.      with sqlite3.connect('Tesis.s3db') as con:  # Establece la conexión con la base de datos.
30.          cursor = con.cursor()   # Crea un cursor para ejecutar los comandos en la tabla de la base de datos.
31.          cursor.execute(comando)  # Ejecuta el comando en la tabla de la base de datos.
32.          cursor.close()  # Cierra el cursor creado para ejecutar comandos.
33.      con.close()
34.
35.
36.  res = ser.readline()
37.  res = ser.readline()
38.  res = ser.readline()
39.  ser.close()
40.
41.  ModificarEstado()
42.  print("Llamada realizada.")
```

Programas de la aplicación móvil

E.10. Zonas Seguras.

- **Eliminar.**

```
public void Eliminar(String Campo)
   {
      // Crea una variable para acceder a las posibles transacciones que se pueden efectuar en la tabla zonas.
      ZonaAccionesBaseDatos acction = new ZonaAccionesBaseDatos(MapsActivity.this);
      // Crea una variable para acceder a los valores de los campos de la tabla zonas.
      Zonas datos = new Zonas();

      // Verifica la zona que será eliminada
      if (Campo == "zona1"){
         // Asigna el valor para borrar la coordenada
         datos.zona1 = " ";
      }
      else if (Campo == "zona2"){
         // Asigna el valor para borrar la coordenada
         datos.zona2 = " ";
      }
      else if (Campo == "zona3"){
         // Asigna el valor para borrar la coordenada
         datos.zona3 = " ";
      }
      else if (Campo == "zona4"){
         // Asigna el valor para borrar la coordenada
         datos.zona4 = " ";
      }
      else if (Campo == "zona5"){
         // Asigna el valor para borrar la coordenada
         datos.zona5 = " ";
      }

      // Llama la función para modificar la coordenada en la base de datos.
      acction.update(datos);
   }
```

- **Insertar/Editar.**

```
public void InsertarEditar (String Campo)
   {
            String Distancia = "10";
            // Crea una variable para acceder a las posibles transacciones que se pueden efectuar en la tabla zonas.
      ZonaAccionesBaseDatos acction = new ZonaAccionesBaseDatos(MapsActivity.this);
            // Crea una variable para acceder a los valores de los campos de la tabla zonas.
            Zonas datos = new Zonas();
            // Declara las variables que almacenaran los datos extraidos
            String Latitud;
            String Longitud;

            // Utiliza la función split para separar los caracteres de texto de los caracteres numéricos, y obtener la latitud y la
      longitud.
            CoordenadasSeleccionada = CoordenadasSeleccionada.split("lat/lng: (")[1];
            Latitud = CoordenadasSeleccionada.split(",")[0];
            CoordenadasSeleccionada = CoordenadasSeleccionada.split(",")[1];
            Longitud = CoordenadasSeleccionada.split(")")[0];

      // Reemplaza el signo por su respectivo polo.
      if (Float.parseFloat(Latitud) >= 0)
      {
            Latitud = "N" + Latitud;
            }
            else
            {
            Latitud = "S" + Latitud;
            }

      // Reemplaza el signo por su respectivo polo.
            if (Float.parseFloat(Longitud) >= 0)
            {
            Longitud = "E" + Longitud;
            }
            else
            {
            Longitud = "W" + Longitud;
            }

            // Aplica el formato utilizado para guardar las coordenadas en la base de datos
            String Texto = "Lat.: " + Latitud + " ; Long.: " + Longitud + " ; Dist. Max.: " + Distancia + " m";

            // Verifica la zona en la cual se insertara/editara la coordenada
            if (Campo == "zona1"){
               datos.zona1 = Texto;
            }
            else if (Campo == "zona2"){
               datos.zona2 = Texto;
            }
            else if (Campo == "zona3"){
               datos.zona3 = Texto;
            }
            else if (Campo == "zona4"){
               datos.zona4 = Texto;
            }
            else if (Campo == "zona5"){
               datos.zona5 = Texto;
            }
            // Llama la función para modificar la coordenada en la base de datos.
            acction.update(datos);    }
```

E.11. Enviar mensajes.

```
. toPhone: Numero de teléfono del destinatario; SMS: Contenido del mensaje de texto.
   public String sendSMS(String toPhone,String SMS){
      try {
         // Crea una instancia para utilizar la libreria "SmsManager".
         SmsManager smsManager = SmsManager.getDefault();
         // Llama la función para enviar un mensaje texto de la librería "SmsManager".
         smsManager.sendTextMessage(toPhone,null,SMS,null,null);
      }
      catch (Exception e) {
         // Indica que ha ocurrido un error al intentar enviar un mensaje de texto.
         e.printStackTrace();
      }
   }
```

E.12. Mostrar coordenadas.

- **Función que extrae las coordenadas geográficas de la base de datos.**

```
//Función que obtiene la longitud, latitud y distancia máxima de las coordenadas almacenadas en la base de datos.
   private void ObtenerDatos(boolean Zona, String Texto, String Nombre_del_Marcador) {
      // Declara las variables que almacenaran los datos extraidos
      String Latitud;
      String Longitud;
      String Distancia;

      // Reemplaza el polo por su respectivo signo. Nota: el cero (0) significa positivo.
      Texto = Texto.replace('N', '0');
      Texto = Texto.replace('S', '-');
      Texto = Texto.replace('E', '0');
      Texto = Texto.replace('W', '-');

      // Utiliza la función split para separar los caracteres de texto de los caracteres numéricos, y obtener la latitud y la
longitud.
      Texto = Texto.split("Lat.: ")[1];
      Latitud = Texto.split(" ;")[0];
      Texto = Texto.split(" ; Long.: ")[1];
      Longitud = Texto.split(" ;")[0];

      // Función utilizada para colocar el punto geográfico en el mapa.
      MostrarPuntosGeograficos(Zona, Float.parseFloat(Latitud), Float.parseFloat(Longitud), Nombre_del_Marcador);

      // Si las coordenadas pertenecen a una zona segura, entonces también debe extraer el diámetro de la zona.
      if (Zona == true) {
         // Utiliza la función split para separar los caracteres de texto de los caracteres numéricos, y obtener la distancia.
         Texto = Texto.split(" ; Dist. Max.: ")[1];
         Distancia = Texto.split(" m")[0];

         // Función utilizada para colocar el diámetro de la zona segura
         MostrarDiametroZonaSegura(Float.parseFloat(Latitud), Float.parseFloat(Longitud), Integer.parseInt(Distancia));
      }
   }
```

- **Función que muestra el diámetro de la zona segura.**

```
private void MostrarDiametroZonaSegura(float Latitud, float Longitud, int Distancia) {

    // Colores del círculo.
    int ColorBorde = 0xFF0099CC; //darkblue outline
    int ColorRelleno = 0xFF33B5E5; //opaque blue fill

    // Variable para las coordenadas del punto geográfico del centro del circulo en el mapa.
    LatLng Coordenadas = new LatLng(Latitud, Longitud);

    // Crea una instancia para las opciones del diámetro de la zona. center define el punto geográfico en que se colocara el circulo,
    // radius es el radio del círculo, fillColor es el color de relleno de la figura, strokeColor es el color del borde de la figura y
    // strokeWidth es el grosor del borde de la figura.
    CircleOptions circleOptions = new CircleOptions()
            .center(Coordenadas)
            .radius(Distancia/2)
            .fillColor(ColorRelleno)
            .strokeColor(ColorBorde)
            .strokeWidth(8); // In meters

    // Coloca el circulo en el mapa.
    Circle circle = mMap.addCircle(circleOptions);
}
```

- **Función que muestra los puntos geográficos de las zonas seguras y del paciente.**

```
private void MostrarPuntosGeograficos(boolean Zona, float Latitud, float Longitud, String Nombre_del_Marcador) {
    // Variable para las coordenadas del punto geografico en el mapa.
    LatLng Coordenadas = new LatLng(Latitud, Longitud);

    // Variable para almacenar el color del marcador del punto geográfico
    float ColorMarcador;

    // Si el punto geográfico es de alguna zona de seguridad entonces el marcador sera azul y el marcador del paciente amarillo.
    if (Zona == true)
    {
        // Declara el color del marcador (azul)
        ColorMarcador = BitmapDescriptorFactory.HUE_BLUE;
    }
    else
    {
        // Declara el color del marcador (amarillo)
        ColorMarcador = BitmapDescriptorFactory.HUE_YELLOW;
    }

    // Crea un nuevo marcador. icon es el color del marcador, el title es el nombre del marcador y position es el punto geografico del marcador.
    Marcadores.add(mMap.addMarker(new MarkerOptions()
            .icon(BitmapDescriptorFactory.defaultMarker(ColorMarcador))
            .title(Nombre_del_Marcador)
            .position(Coordenadas)));
    // Realiza un acercamiento al punto geográfico.
    mMap.animateCamera(CameraUpdateFactory.newLatLngZoom(Coordenadas, 16));
}
```

- **Función que muestra los puntos geográficos de las zonas seguras y del paciente.**

```
// Función que obtiene las ubicaciones geográficas de las zonas seguras y del paciente.
   private void CoordenadasBaseDatos(int Id_Cuidador){
      // Crea una variable para acceder a las posibles transacciones que se pueden efectuar en la tabla zonas.
      ZonaAccionesBaseDatos acction = new ZonaAccionesBaseDatos(MapsActivity.this);
      // Crea una variable para acceder a los valores de los campos de la tabla zonas.
      Zonas datos;

      // Se obtienen los valores de los campos de la tabla zonas.
      datos = acction.getLogByID(Id_Cuidador);

      // Se verifica la cantidad de caracteres del campo, si es igual o menos que uno entonces indica que no posee
coordenadas.
      if ((datos.zona1.length()) > 1) {
         // Llama la función que obtiene la latitud, longitud y distancia de la zona
         ObtenerDatos(true, datos.zona1, "Zona Segura #1");
      }

      // Se verifica la cantidad de caracteres del campo, si es igual o menos que uno entonces indica que no posee
coordenadas.
      if ((datos.zona2.length()) > 1) {
         // Llama la función que obtiene la latitud, longitud y distancia de la zona
         ObtenerDatos(true, datos.zona2, "Zona Segura #2");
      }

      // Se verifica la cantidad de caracteres del campo, si es igual o menos que uno entonces indica que no posee
coordenadas.
      if ((datos.zona3.length()) > 1) {
         // Llama la función que obtiene la latitud, longitud y distancia de la zona
         ObtenerDatos(true, datos.zona3, "Zona Segura #3");
      }

      // Se verifica la cantidad de caracteres del campo, si es igual o menos que uno entonces indica que no posee
coordenadas.
      if ((datos.zona4.length()) > 1) {
         // Llama la función que obtiene la latitud, longitud y distancia de la zona
         ObtenerDatos(true, datos.zona4, "Zona Segura #4");
      }

      // Se verifica la cantidad de caracteres del campo, si es igual o menos que uno entonces indica que no posee
coordenadas.
      if ((datos.zona5.length()) > 1) {
         // Llama la función que obtiene la latitud, longitud y distancia de la zona
         ObtenerDatos(true, datos.zona5, "Zona Segura #5");
      }

      // Se verifica la cantidad de caracteres del campo, si es igual o menos que uno entonces indica que no posee
coordenadas.
      if ((datos.CoordenadaPaciente.length()) > 1) {
         // Llama la función que obtiene la latitud y la longitud.
         ObtenerDatos(false, datos.CoordenadaPaciente, "Posición del paciente");
      }
   }
```

E.13. Recibir mensajes.

```
public void refreshInbox() {
      ContentResolver contentResolver = getContentResolver();
      Cursor inboxCursor = contentResolver.query(Uri.parse("content://sms/inbox"), null, null, null, null);
      int indexBody = inboxCursor.getColumnIndex("body");
      int indexAddress = inboxCursor.getColumnIndex("address");
      int timeMillis = inboxCursor.getColumnIndex("date");
      Date date = new Date(timeMillis);
      SimpleDateFormat format = new SimpleDateFormat("dd/MM/yy");
      String dateText = format.format(date);
      if (indexBody < 0 || !inboxCursor.moveToFirst()) return;
      arrayAdapter.clear();
      do {
         String str = inboxCursor.getString(indexAddress) + " at " + "\n" + dateText+"\n" +
inboxCursor.getString(indexBody)  + "\n";
         arrayAdapter.add(str);

      } while (inboxCursor.moveToNext());
   }
```

APÉNDICE F: Figuras de la base de datos

SQL Query | Result | Edit Data

Table: estado

Filterfield: Filter:

ID	Cuidador	Paciente	A_Cargo_del_Paciente	Telefono	Tiempo_Actualizacion	Estado	Respuesta	Proceso
1	Ref. #00001: Luis Alejandro Mateo López	Ref. #1: Juan Carlos Nuñez Medina	Si	829-717-2708	Dias: 0, Minutos: 0	Seguro	N/A	N/A
2	Ref. #00002: Julio Antonio Pérez Medina	Ref. #1: Juan Carlos Nuñez Medina	No	829-455-5902	Dias: 0, Minutos: 0	N/A	N/A	N/A
3	Ref. #00003: José María Martinez Pérez	Ref. #2: Carmen María Pérez Báez	Si	809-548-5985	Dias: 0, Minutos: 3	Seguro	N/A	N/A

Figura F.1. Módulo procesador de datos: Tabla de la base de datos que muestra los resultados y análisis del sistema sobre el estado del paciente.

SQL Query | Result | Edit Data

Table: cuidador

Filterfield: Filter:

ID	Nombres	Apellidos	Edad	Cedula	Direccion	Telefono	Correo_Electronico	Ocupacion	Paciente_Asociado	Id_Dispositivo	Usuario
00001	Luis Alejandro	Mateo López	30	402-8284988-6	Avenida Bolivar 233, Santo Domingo	829-717-2708	lmateo@gmail.com	Profesor	Ref. #1: Juan Carlos Nuñez Medina	85985	lmateo
00002	Julio Antonio	Pérez Medina	21	402-2161561-5	Av México 48, Santo Domingo	829-455-5902	jperez@gmail.com	Estudiante	Ref. #1: Juan Carlos Nuñez Medina	85985	jperez
00003	José María	Martinez Pérez	30	001-5625165-4	Dirección Gral de Catastro Nacional,	809-548-5985	jmartinez@gmail.com	Profesor	Ref. #2: Carmen María Pérez Báez	66767	jmartinez

Figura F.2. Módulo procesador de datos: Tabla de la base de datos con la información de cuidadores registrados.

SQL Query | Result | Edit Data

Table: usucuidador

Filterfield: Filter:

ID	Usuario	Clave
00001	lmateo	123456
00002	jperez	123456
00003	jmartinez	123456

Figura F.3. Módulo procesador de datos: Tabla de la base de datos con las credenciales de los custodios. *Estos datos no son visualizados en la interfaz gráfica.*

SQL Query | Result | Edit Data

Table: zona

Filterfield: Filter:

ID	Cuidador	Paciente	Id_Dispositivo	Ubicacion_Actual	Ultima_Actualizacion	Cantidad_de_Zonas	Zona_Segura_1	Zona_Segura_2	Zona_Segura_3	Zona_Segura_4	Zona_Segura_5
1	Ref. #00001: Luis Alejandro	Ref. #1: Juan Carlos	85985	Lat.: N90.0000000	08/11/2016 12:33:20	4	Lat.: N90.00000	Lat.: S18.2730 ;	Lat.: N20.27535	Lat.: S25.27535	Inhabilitada
2	Ref. #00002: Julio Antonio P	Ref. #1: Juan Carlos	85985	Lat.: N90.0000000	08/11/2016 12:33:20	3	Lat.: N23.56265	Lat.: S10.598 ;	Lat.: S45.1888 ;	Inhabilitada	Inhabilitada
3	Ref. #00003: José María Ma	Ref. #2: Carmen Mar	66767	Lat.: N25.123456 ;	08/11/2016 12:30:20	2	Lat.: N25.12345	Lat.: N55.56265	Inhabilitada	Inhabilitada	Inhabilitada

Figura F.4. Módulo procesador de datos: Tabla de la base de datos con la información de zonas seguras y coordenadas geográficas del paciente.

APÉNDICE G: Publicación periódica

Estudio demuestra que dominicanos son proclives a sufrir de Alzheimer

SANTO DOMINGO. Los hallazgos de un estudio sobre el Alzheimer hecho por la Columbia University Medical Center entre hispanos que residen en el barrio Washington Heights, del bajo Manhattan, en Nueva York, revelan que los dominicanos son de los grupos de hispanos con mayor prevalencia de la enfermedad.

De una muestra de 2,100 personas, de las cuales 65% eran hispanas y de éstas 70% dominicanas, le da poder a los investigadores para decir que la incidencia de la enfermedad es mucho mayor en latinos que en blancos y negros.

Rafael Lantigua, uno de los tres investigadores, dijo que se trata de un Estudio Familiar de Influencia Genética en Alzheimer (EFIGA), aleatorio, donde se entrevistó a varias familias, y que inició en el 1989.

Después de los primeros resultados obtenidos en Nueva York, los investigadores expandieron hace 12 años el universo a República Dominicana y Puerto Rico y en la actualidad incluye 700 familias con múltiples personas afectadas por la enfermedad y 9,000 evaluadas.

En el país, el estudio cuenta con el apoyo de la facultad de medicina de la Pontificia Universidad Católica Madre y Maestra (PUCMM), y entre sus objetivos está la oportunidad de desarrollar nuevos diagnósticos, así como terapias para tratar la enfermedad.

Lantigua dijo que las investigaciones continuarán en el país por los próximos cinco años con 25 familias más y con el financiamiento de los Estados Unidos que desde 1989, han aportado US$40 millones.

También, que 30 casos dominicanos pertenecen al pool de 74 mil personas estudiadas a nivel mundial para determinar los factores de riesgos y posibles tratamientos.

Los síntomas comunes de la dolencia son dificultad para recordar informaciones nuevas, cambios de ánimo y comportamiento. La pérdida de memoria afecta las capacidades intelectuales.

Generalidades

Al Alzheimer se le considera como el tipo de demencia más común en el mundo, con más 46 millones de personas afectadas.

Su frecuencia entre hispanos es tres veces mayor que en los anglosajones y cinco veces mayor que en la población en general.

Se estima que los individuos mentalmente activos tienen menos riesgo de padecer la enfermedad, y que el ejercicio físico disminuye las posibilidades de sufrir la afección.

Otros logros

La colección más grande de datos de familias hispanas que se utilizan para conocer la enfermedad y tratarla se registra como uno de los logros más importante de la investigación.

También, la publicación de más de 300 trabajos en revistas médicas científicas, que se utilizan como referentes, así como la identificación de los 30 genes asociados al mal.

Principales factores de riesgo

Los principales factores de riesgo para sufrir de Alzheimer son obesidad, diabetes, enfermedades cardiovasculares y vida sedentaria. Para los investigadores, lo más importante es la falta de educación, por lo que recomiendan a los gobiernos invertir en educación para prevenir esa y otras dolencia prevenibles.

Figura G.1. Artículo sobre estudios de la enfermedad de Alzheimer en dominicanos.[136] *Fuente: Diario Libre.*

APÉNDICE H: Contenido del CD

- ***Trabajo de grado digital.***

- Documento en su versión para Word 2007.
- Documento en formato PDF.

- **Carpeta*:*** *Seccionada para Turnitin.*

- Trabajo de grado (sin los anexos)
- Anexos

- **Video:** *DISEÑO DE UN SISTEMA ELECTRÓNICO PARA LOCALIZAR PERSONAS CON ALZHEIMER.* En este video se presenta una prueba de funcionamiento del sistema y se compone de las siguientes partes:

- Hardware del procesador de datos
- Hardware de la pulsera electrónica
- Ingreso de información en la interfaz gráfica del administrador
- Procesador de datos: Ventanas de la Interfaz gráfica
- Inicio de sesión en la aplicación móvil e ingreso de zona segura
- Inicio de los programas modulares del módulo procesador de datos
- Inicio del programa de la pulsera electrónica
- Pulsera electrónica: Envío de la ubicación
- Procesador de datos: Análisis de las coordenadas geográficas (fuera de la zona segura)
- Solicitud de ubicación y recepción de alerta de peligro en la aplicación móvil
- Programas modulares del procesador de datos: Recepción de respuesta de la aplicación
- Recepción de respuesta de alerta del procesador de datos a la aplicación móvil
- Pulsera electrónica: Envío de la ubicación
- Programas modulares del procesador de datos: Análisis de la ubicación recibida
- Solicitud de ubicación en la aplicación móvil
- Programas modulares del procesador de datos: Recepción de solicitud de la ubicación del paciente