ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA ELECTRÓNICA EN REDES Y COMUNICACIÓN DE DATOS

TRABAJO DE TITULACIÓN, PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO ELECTRÓNICO EN REDES Y COMUNICACIÓN DE DATOS

TEMA: DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA CLIENTE-SERVIDOR PARA EL ENVÍO DE POSICIÓN Y SIGNOS VITALES DE MASCOTAS SOBRE DISPOSITIVOS MÓVILES EN LA PLATAFORMA ANDROID

AUTOR: ANDRADE PARREÑO, CARLOS ANDRÉS

DIRECTOR: MSC. ING. ALULEMA FLORES, DARWIN OMAR

SANGOLQUÍ

2016

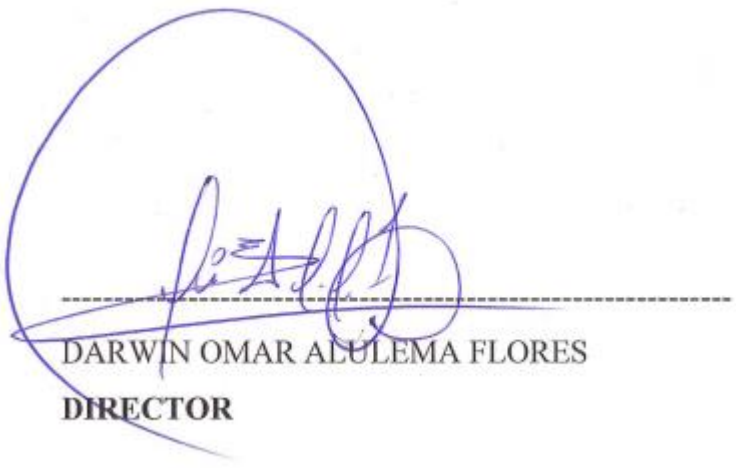
DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERIA ELECTRÓNICA EN REDES Y COMUNICACIÓN DE DATOS

CERTIFICACIÓN

Certifico que el trabajo de titulación, "***DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA CLIENTE-SERVIDOR PARA EL ENVÍO DE POSICIÓN Y SIGNOS VITALES DE MASCOTAS SOBRE DISPOSITIVOS MÓVILES EN LA PLATAFORMA ANDROID*"**, realizado por el señor ***ANDRADE PARREÑO CARLOS ANDRÉS***, ha sido revisado en su totalidad y analizado por el software anti-plagio, el mismo cumple con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de Fuerzas Armadas ESPE, por lo tanto me permito acreditarlo y autorizar al señor ***ANDRADE PARREÑO CARLOS ANDRÉS*** para que lo sustente públicamente.

Sangolquí, Agosto del 2016



DARWIN OMAR ALULEMA FLORES

DIRECTOR

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERIA ELECTRÓNICA EN REDES Y COMUNICACIÓN DE DATOS

AUTORÍA DE RESPONSABILIDAD

Yo, ***ANDRADE PARREÑO CARLOS ANDRÉS***, con cédula de identidad N° 1715068761, declaro que este trabajo de titulación, ***"DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA CLIENTE-SERVIDOR PARA EL ENVÍO DE POSICIÓN Y SIGNOS VITALES DE MASCOTAS SOBRE DISPOSITIVOS MÓVILES EN LA PLATAFORMA ANDROID"*** ha sido desarrollado considerando los métodos de investigación existentes, así como también se ha respetado los derechos intelectuales de terceros considerándose en las citas bibliográficas.

Consecuentemente declaro que este trabajo es de mi autoría, en virtud de ello me declaro responsable del contenido, veracidad y alcance de la investigación mencionada.

Sangolquí, 22 de Agosto del 2016

CARLOS ANDRÉS ANDRADE PARREÑO

C.C 1715068761

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERIA ELECTRÓNICA EN REDES Y COMUNICACIÓN DE DATOS

AUTORIZACIÓN

Yo, ***ANDRADE PARREÑO CARLOS ANDRÉS***, autorizo a la Universidad de las Fuerzas Armadas ESPE publicar en la biblioteca Virtual de la institución el presente trabajo de titulación **"DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA CLIENTE-SERVIDOR PARA EL ENVÍO DE POSICIÓN Y SIGNOS VITALES DE MASCOTAS SOBRE DISPOSITIVOS MÓVILES EN LA PLATAFORMA ANDROID"** cuyo contenido, ideas y criterios son de mi autoría y responsabilidad.

Sangolquí, 22 de Agosto del 2016

CARLOS ANDRÉS ANDRADE PARREÑO

C.C 1715068761

DEDICATORIA

A mi madre, que a pesar de que no se encuentra conmigo en estos momentos, es la persona que me impulso en gran parte de mi carrera, inculcándome valores, consejos y responsabilidad.

A mi hermano quien ha estado conmigo en los momentos más difíciles de mi vida.

A mi abuela, quien me ha brindado comprensión y me ha apoyado en distintas situaciones y momentos.

A toda mi familia quienes de una u otra manera me han enseñado a seguir adelante, a pesar de las dificultades presentadas en el camino.

Carlos Andrés Andrade Parreño

AGRADECIMIENTO

Agradezco a mi madre, quien fue la persona que me impulso de distintas maneras para la culminación de mi carrera profesional además de inculcarme varios valores los cuales me sirvieron en mi vida personal y profesional.

A mi tío Marcelo y mi primo Pablo para la obtención de materiales y realización del diseño visual del arnés.

A mi hermano por su apoyo incondicional.

Al Ingeniero Darwin Alulema por su guía, consejos y acertadas correcciones en el transcurso de la elaboración del proyecto.

A mi prima Gabriela y mi tía política Zoily por ayudarme con las mascotas para la realización de las pruebas.

A mi familia en general, por el apoyo y disposición prestada.

Carlos Andrés Andrade Parreño

ÍNDICE DE CONTENIDO

CERTIFICACIÓN **I**

AUTORÍA DE RESPONSABILIDAD **II**

AUTORIZACIÓN **III**

DEDICATORIA **IV**

AGRADECIMIENTO **V**

ÍNDICE DE CONTENIDO **VI**

INDICE DE TABLAS **IX**

ÍNDICE DE FIGURAS **XI**

RESÚMEN **XV**

ABSTRACT **XVI**

CAPÍTULO 1 **1**
1. DEFINICIÓN DEL PROYECTO 1
1.1. Antecedentes. 1
1.2. Justificación e importancia 2
1.3. Alcance del proyecto 3
1.4. Objetivos 4
1.4.1. Objetivo general de la investigación 4
1.4.2. Objetivos Específicos 4
1.5. Estado del Arte. 4

CAPÍTULO II **9**
2. MARCO TEÓRICO 9
2.1. Marco Legal 9
2.1.1. Reglamento Nacional de Tenencia de Perros 9
2.1.2. Código Orgánico Integral Penal 10
2.1.3. Código Civil Libro IV 10
2.1.4. Ordenanzas Municipales del Distrito metropolitano de Quito 10
2.1.5. Proyecto de Ley “Ley Orgánica de Bienestar Animal” 13
2.2. Sistema Operativo Android 13
2.2.1. ¿Qué es Android? 13
2.2.2. Importancia del desarrollo en Android 14
2.2.3. El código fuente de Android 15
2.2.4. Versiones del Sistema Operativo Android 17

2.2.5. Desarrollo en Android 18
2.3. Plataforma Programable Arduino 19
2.3.1. ¿Qué es Arduino? 19
2.3.2. Importancia de la tarjeta Programable Arduino 19
2.3.3. Hardware 21
2.4. Sistema de Posicionamiento Global (GPS) 29
2.5. Servicio General de Paquetes Vía Radio (GPRS) 29
2.6. Módulo SIM 908C 29
2.6.1. Características Generales 29
2.6.2. Hardware 31
2.6.3. Modos de Operación del Módulo SIM 908C 32
2.6.4. Diagrama Funcional del Módulo SIM 908 35
2.6.5. Distribución de pines del módulo SIM 908 36
2.7. Módulo Transductor Acelerómetro MMA7361 44
2.7.1. Funcionamiento 45
2.7.2. Descripción 46
2.8. Sensor de Temperatura Infrarrojo MLX90614 49
2.8.1. Características 49
2.8.2. Funcionamiento interno del sensor MLX90614. 50
2.8.3. Descripción de pines. 51
2.8.4. Especificaciones eléctricas. 52
2.9. Arquitectura Cliente-Servidor 53
2.9.1. Servidor Web 54

CAPÍTULO III **55**
3. IMPLEMENTACIÓN DEL PROYECTO 55
3.1. Diseño de Hardware 55
3.1.1. Recepción de señal de posicionamiento geográfico GPS 57
3.1.2. Recepción de información de signos vitales 61
3.1.3. Unidad de Control 64
3.1.4. Procesamiento de información GPS. 66
3.1.5. Transmisión de información 68
3.1.6. Alimentación. 70
3.2. Implementación de Hardware 72
3.2.1. Implementación del módulo SIM 908 73
3.2.2. Implementación del sensor acelerómetro MMA7361 82
3.2.3. Implementación del sensor MLX90614 86
3.2.4. Implementación Final y Conexión General 87
3.3. Diseño de Software. 90
3.3.1. Representación de interacción de los componentes del sistema de software 98
3.3.3. Diseño e implementación de servicios para el sistema de hardware.
104
3.3.4. Interfaz Web 111
3.3.5. Diseño Web 111
3.4. Diseño e implementación de la aplicación en Android. 144
3.4.1. Diseño de la aplicación en Android. 144

3.4.2. Implementación de servicios para la aplicación en Android. 147

CAPÍTULO IV **175**
4. ESCENARIOS DE PRUEBA 175
4.1. Sujetos de prueba 178
4.2. Consumo de datos. 179
4.3. Escenarios de Prueba 181
4.3.1. Escenario 1: Parque. 182
4.3.2. Escenario 2: Interior de Casa 187
4.3.3. Escenario 3: Ciudad. 191
4.3.4. Escenario 4: Auto. 195
4.3.5. Escenario 5: Ambiente Variable. 197
4.4. Consumo de batería. 199
4.5. Costos. 199
4.6. Análisis de resultados. 200

CAPÍTULO V **204**
5. CONCLUSIONES Y RECOMENDACIONES 204
5.1. Conclusiones 204
5.2. Recomendaciones 209
5.3. BIBLIOGRAFIA 211

INDICE DE TABLAS

TABLA 1 CARACTERÍSTICAS DE DISPOSITIVOS PARA GEOLOCALIZACIÓN DE MASCOTAS EN EL MERCADO. 5
TABLA 2 VERSIONES PUBLICADAS DEL SISTEMA OPERATIVO ANDROID 18
TABLA 3 VERSIONES Y DISTRIBUCIONES DE PLACAS ARDUINO Y GENUINO ACTUALMENTE EN EL MERCADO 22
TABLA 4 RESUMEN DE ESPECIFICACIONES TÉCNICAS DE LA PLATAFORMA ARDUINO UNO 25
TABLA 5 CARACTERÍSTICAS DE HARDWARE MODULO SIM 908..... 31
TABLA 6 RESUMEN DE LOS MODOS DE OPERACIÓN DEL MÓDULO SIM 908..... 33
TABLA 7 DESCRIPCIÓN DE PINES DEL MÓDULO SIM 908 38
TABLA 8 AJUSTE DE SENSIBILIDAD ACELERÓMETRO MMA7361..... 48
TABLA 9 DESCRIPCIÓN DE PINES MLX90614..... 52
TABLA 10 ESPECIFICACIONES ELÉCTRICAS MLX90614 52
TABLA 11 RESUMEN DE VOLTAJE Y CORRIENTE DE DISPOSITIVOS INVOLUCRADOS EN EL PROYECTO..... 70
TABLA 12 VARIABLES GLOBALES USADAS EN EL DESARROLLO DEL HARDWARE..... 75
TABLA 13 CONEXIÓN ARDUINO UNO A MÓDULO ACELERÓMETRO MMA7361..... 84
TABLA 14 COMPONENTES Y FUNCIONES INICIALES DE LA PÁGINA WEB..... 112
TABLA 15 CARACTERÍSTICAS FÍSICAS DE CAJA PARA EL ARNÉS.... 176
TABLA 16 SUJETOS DE PRUEBA..... 178
TABLA 17 PRUEBA DE CONSUMO DE DATOS..... 180
TABLA 18 CONSUMO DE DATOS..... 180
TABLA 19 RESUMEN DE PLANES DE DATOS RECOMENDADOS POR EL CONSUMO DEL PROYECTO..... 181
TABLA 20 PRECISIÓN DE UBICACIÓN EN ESCENARIO 1 183
TABLA 21 COORDENADAS DE UBICACIÓN EN ESCENARIO 1..... 183
TABLA 22 VARIACIÓN ACELERÓMETRO CON MASCOTA EN MOVIMIENTO 185
TABLA 23 VARIACIÓN ACELERÓMETRO EN REPOSO..... 185
TABLA 24 TEMPERATURA CORPORAL EN ESCENARIO 1..... 186
TABLA 25 PRECISIÓN DE UBICACIÓN EN ESCENARIO 2 188
TABLA 26 COORDENADAS GEOGRÁFICAS ESCENARIO 2..... 189
TABLA 27 VARIACIÓN ACELERÓMETRO CON MASCOTA SENTADA 190
TABLA 28 TEMPERATURA CORPORAL EN ESCENARIO 2..... 191
TABLA 29 PRECISIÓN GPS EN ESCENARIO 3..... 192
TABLA 30 COORDENADAS GEOGRÁFICAS ESCENARIO 3..... 193
TABLA 31 VARIACIÓN ACELERÓMETRO EN ESCENARIO 3..... 194
TABLA 32 TEMPERATURA CORPORAL EN ESCENARIO 3..... 195
TABLA 33 DATOS ESCENARIO 4..... 196
TABLA 34 LUGARES RECORRIDOS..... 196
TABLA 35 RECUPERACIÓN DE UBICACIÓN EN ESCENARIO 5 197

TABLA 36 RECUPERACIÓN DE MOVIMIENTO EN ESCENARIO 5 198
TABLA 37 COSTOS DE IMPLEMENTACIÓN..... 200

ÍNDICE DE FIGURAS

FIGURA 1 COLLARES INTELIGENTES EN EL MERCADO, 1. PETPACE, 2. HEYREX, 3. TRACTIVE, 4. WHISTLE 6
FIGURA 2 LOGO ANDROID 14
FIGURA 3 PILA ANDROID 15
FIGURA 4 ESTRUCTURA DEL CÓDIGO AOSP EN ANDROID 17
FIGURA 5 PLACA ARDUINO UNO..... 24
FIGURA 6 ESQUEMA DE PINES DEL MICROCONTROLADOR ATMEGA328..... 26
FIGURA 7 COMPONENTES DE LA PLACA ARDUINO UNO. 27
FIGURA 8 PINES DE LA PLACA ARDUINO UNO. 27
FIGURA 9 ESQUEMA ELÉCTRICO DE ARDUINO UNO. 28
FIGURA 10 VISTA FRONTAL SIM908 30
FIGURA 11 DIAGRAMA FUNCIONAL DEL SIM 908..... 35
FIGURA 12 DIAGRAMA DE PINES DEL MÓDULO SIM 908 VISTA SUPERIOR. 36
FIGURA 13 MÓDULO SIM908 PARA ARDUINO. 41
FIGURA 14 ANTENA 42
FIGURA 15 MODULO SIM 908 Y CIRCUITO DE ACOPLAMIENTO..... 42
FIGURA 16 ANTENA 43
FIGURA 17 MODULO SIM 908 Y CIRCUITO DE ACOPLAMIENTO PARA ANTENA GSM..... 44
FIGURA 18 MÓDULO ACELERÓMETRO MMA7361. 44
FIGURA 19 DIAGRAMA REPRESENTATIVO DEL FUNCIONAMIENTO DEL MÓDULO ACELERÓMETRO MMA 7361. 45
FIGURA 20 DIAGRAMA REPRESENTATIVO DE LA ACELERACIÓN DINÁMICA. 46
FIGURA 21 DIAGRAMA REPRESENTATIVO DE LA ACELERACIÓN ESTÁTICA. 46
FIGURA 22 PINES ENTRADA DE VOLTAJE Y TIERRA. 47
FIGURA 23 REGULADOR DE 5V. 47
FIGURA 24 PINES ANÁLOGOS DE SALIDA. 47
FIGURA 25 PINES DE CONTROL Y SALIDA DIGITAL. 48
FIGURA 26 DIAGRAMA DE BLOQUES DEL FUNCIONAMIENTO DEL SENSOR MLX90614. 50
FIGURA 27 VISTA INFERIOR MLX90614. 51
FIGURA 28 DIAGRAMA REPRESENTATIVO DE FUNCIONAMIENTO DEL SISTEMA A NIVEL DE HARDWARE. 56
FIGURA 29 REPRESENTACIÓN DE HARDWARE UTILIZADO EN LA ADQUISICIÓN DE SEÑAL GPS. 57
FIGURA 30 REPRESENTACIÓN DE TRAMA GPS. 58
FIGURA 31 EJEMPLO DE TRAMA GPS. 58
FIGURA 32 DIAGRAMA DE FLUJO RECEPCIÓN GPS. 60
FIGURA 33 REPRESENTACIÓN DE HARDWARE INVOLUCRADO EN LA ADQUISICIÓN DE INFORMACIÓN DE MOVIMIENTO. 61

FIGURA 34 DIAGRAMA DE FLUJO RECEPCIÓN MOVIMIENTOS 62
FIGURA 35 REPRESENTACIÓN DE HARDWARE UTILIZADO PARA LA ADQUISICIÓN DE INFORMACIÓN DE TEMPERATURA. 63
FIGURA 36 DIAGRAMA DE FLUJO RECEPCIÓN TEMPERATURA 63
FIGURA 37 DIAGRAMA DE FLUJO DEL PROCESO DE CONTROL DE INFORMACIÓN EN EL SISTEMA. 65
FIGURA 38 DIAGRAMA DE FLUJO DEL ESTABLECIMIENTO DE FORMATO GPS. 67
FIGURA 39 REPRESENTACIÓN DE HARDWARE INVOLUCRADO EN EL ENVÍO DE INFORMACIÓN. 68
FIGURA 40 DIAGRAMA DE FLUJO DEL PROCESO DE ENVÍO DE INFORMACIÓN. 69
FIGURA 41 DIAGRAMA DE BLOQUES DE LA ALIMENTACIÓN DEL HARDWARE 72
FIGURA 42 ESQUEMA DE CONEXIÓN ARDUINO UNO – SIM 908. 74
FIGURA 43 DIAGRAMA DE CLASES GENERAL DEL SISTEMA DE HARDWARE 76
FIGURA 44 DIAGRAMA DE CLASES DEL MÉTODO “SETUP”. 77
FIGURA 45 DIAGRAMA DE CLASES DEL MÉTODO “LOOP”. 80
FIGURA 46 ESQUEMA DE CONEXIÓN MÓDULO ACELEROMETROMMA7361 CON ARDUINO UNO. 83
FIGURA 47 DIAGRAMA DE CLASES MÉTODO “SETUP “Y LIBRERÍA “ACCELEROMMA7361” 84
FIGURA 48 DIAGRAMA DE CLASES GENERAL MÉTODO “LOOP” CON LA LIBRERÍA “ACCELEROMMA7361”. 85
FIGURA 49 ESQUEMA DE CONEXIÓN SENSOR MLX90614 CON ARDUINO UNO 87
FIGURA 50 ESQUEMA DE CONEXIÓN GENERAL. (FRITZING SOFTWARE) 88
FIGURA 51 VISTA CAJA ABIERTA 89
FIGURA 52 VISTA CAJA CERRADA 89
FIGURA 53 DIAGRAMA DE CASO DE USO GENERAL DEL SISTEMA 92
FIGURA 54 DIAGRAMA DE CASO DE USO “AÑADIR ID" 94
FIGURA 55 DIAGRAMA DE CASO DE USO “VER UBICACIÓN” 95
FIGURA 56 DIAGRAMA DE CASO DE USO “VER SIGNOS VITALES” 97
FIGURA 57 DIAGRAMA REPRESENTATIVO DE LA CONEXIÓN DEL SISTEMA 99
FIGURA 58 DIAGRAMA REPRESENTATIVO DEL SISTEMA DE BASE DE DATOS 100
FIGURA 59 REPRESENTACIÓN DE ENTIDADES DE LA BASE DE DATOS 100
FIGURA 60 REPRESENTACIÓN DE ENTIDADES Y ATRIBUTOS DE LA BASE DE DATOS 101
FIGURA 61 DIAGRAMA LÓGICO DE BASE DE DATOS 104
FIGURA 62 DIAGRAMA GENERAL DE DISEÑO DE SERVICIOS PARA EL HARDWARE DEL SISTEMA 105
FIGURA 63 DIAGRAMA DE FLUJO DEL SCRIPT DE CONFIGURACIÓN “CARGAR2.PHP” 106

FIGURA 64 DIAGRAMA DE ARQUITECTURA INICIAL DE LA PÁGINA WEB 112
FIGURA 65 ELEMENTOS DE LA PÁGINA WEB 113
FIGURA 66 ASPECTO VISUAL DE LA PÁGINA WEB SECCIÓN “QUIENES SOMOS” 114
FIGURA 67 DIAGRAMA DE ARQUITECTURA DE LA SECCIÓN 3 REGISTRO 114
FIGURA 68 ASPECTO VISUAL DE LA SECCIÓN REGISTRO 115
FIGURA 69 DIAGRAMA DE FLUJO DE LA SECCIÓN REGISTRO 116
FIGURA 70 DIAGRAMA DE ARQUITECTURA DE LA SECCIÓN 4 INICIO DE SESIÓN 116
FIGURA 71 DIAGRAMA DE FLUJO DE LA SECCIÓN INICIAR SESIÓN 117
FIGURA 72 ASPECTO VISUAL DE LA SECCIÓN INICIAR SESIÓN 117
FIGURA 73 DIAGRAMA REPRESENTATIVO DE FUNCIONAMIENTO DE INICIO DE SESIÓN Y REGISTRO 119
FIGURA 74 PROCESO DE SCRIPT DE CONFIGURACIÓN REGISTER.PHP . 121
FIGURA 75 PROCESO DE SCRIPT DE CONFIGURACIÓN 125
FIGURA 76 DIAGRAMA REPRESENTATIVO DE INGRESO AL PERFIL DE USUARIO 127
FIGURA 77 VISTA WEB DE OPCIONES EN PERFIL DE USUARIO 128
FIGURA 78 DIAGRAMA REPRESENTATIVO DE FUNCIONAMIENTO DE "AGREGARCOLLAR.HTML" 129
FIGURA 79 VISTA WEB DE "AGREGARCOLLAR.HTML" 130
FIGURA 80 DIAGRAMA DE FLUJO DE ARCHIVO DE CONFIGURACIÓN “REGISTROCOLLAR.PHP” 131
FIGURA 81 DIAGRAMA REPRESENTATIVO DE CONEXIÓN DEL SERVICIO “VER UBICACIÓN” 134
FIGURA 82 DIAGRAMA DE FLUJO DEL SERVICIO WEB “MOSTRAR UBICACIÓN” 135
FIGURA 83 MAPA EN PÁGINA WEB 139
FIGURA 84 ALERTA DE SIGNOS VITALES EN PÁGINA WEB 144
FIGURA 85 DIAGRAMA DE FLUJO GENERAL DEL USO EN LA APLICACIÓN EN ANDROID 145
FIGURA 86 DIAGRAMA DE ARQUITECTURA DE CONTENIDOS DE LA APLICACIÓN EN ANDROID 146
FIGURA 87 DISEÑO VISUAL DE LA APLICACIÓN EN ANDROID 147
FIGURA 88 LAYOUT “START_ACTIVITY.XML” 148
FIGURA 89 DIAGRAMA DE CLASES “INICIO.CLASS” 148
FIGURA 90 DIAGRAMA DE FLUJO DEL SERVICIO DE REGISTRO EN ANDROID 149
FIGURA 91 DIAGRAMA DE CLASES “REGISTRAR.CLASS” 150
FIGURA 92 LAYOUT “REGISTER_ACTIVITY.XML” 150
FIGURA 93 DIAGRAMA DE FLUJO SCRIPT “REGISTER.PHP” 152
FIGURA 94 DIAGRAMA DE FLUJO INICIO DE SESIÓN EN ANDROID 154
FIGURA 95 DIAGRAMA DE CLASES DE “INGRESAR.CLASS” 154
FIGURA 96 LAYOUT “LOGIN_ACTIVITY.XML” 155
FIGURA 97 DIAGRAMA DE FLUJO DE SCRIPT DE CONFIGURACIÓN “LOGIN.PHP” 157

FIGURA 98 LAYOUT “USER_PROFILE.XML” 157
FIGURA 99 LAYOUT “ANADIR_ACTIVITY.XML”..... 158
FIGURA 100 DIAGRAMA DE CLASES “ANADIRID.CLASS”..... 159
FIGURA 101 DIAGRAMA DE FLUJO VER UBICACIÓN ANDROID..... 160
FIGURA 102 DIAGRAMA DE CLASES RELACIONADAS AL SERVICIO DE UBICACIÓN EN ANDROID..... 161
FIGURA 103 LAYOUT “READING_JSON.XML” 161
FIGURA 104 UBICACIÓN EN ANDROID..... 165
FIGURA 105 DIAGRAMA DE FLUJO VER SIGNO VITAL ANDROID..... 166
FIGURA 106 DIAGRAMA DE CLASES VER SIGNO VITAL ANDROID..... 167
FIGURA 107 “LAYOUT ACTIVITY_FORMSIGNO.XML” Y “ACTIVITY_VER_SIGNOS.XML”..... 168
FIGURA 108 DIAGRAMA DE FLUJO SCRIPT “PERROARNES.PHP”..... 170
FIGURA 109 DISEÑO VISUAL DE CAJA Y ARNÉS 175
FIGURA 110 MEDIDAS DE LA CAJA CONTENEDORA DE LOS COMPONENTES DE HARDWARE..... 176
FIGURA 111 ESTABLECIMIENTO DEL ARNÉS EN MASCOTAS..... 177
FIGURA 112 ARNÉS EN SUJETO DE PRUEBA “M3”..... 178
FIGURA 113 ARNÉS EN SUJETO DE PRUEBA “M2”..... 179
FIGURA 114 ARNÉS EN SUJETO DE PRUEBA “M1”..... 179
FIGURA 115 ESCENARIO 1 182
FIGURA 116 ESCENARIO 2 188
FIGURA 117 ESCENARIO 3 192

RESÚMEN

El proyecto establece un prototipo para evitar la pérdida de mascotas. Se realizó un estudio del estado del arte de los sistemas de monitoreo de mascotas para dispositivos móviles en el mercado actual, con lo que se determinó las características, técnicas y condiciones para la implementación del proyecto. Se implementó un arnés, capaz de obtener las coordenadas geográficas de ubicación y los signos vitales de la mascota. Como signo vital, el arnés detecta el movimiento y la temperatura del animal. Los sensores GPS y de signos vitales, fueron sometidos a una fase de pruebas para determinar la precisión y obtener la configuración con mayor eficacia para el proyecto. La información obtenida por los sensores es procesada a través de la plataforma programable Arduino y enviada a través de un módulo GPS/GSM/GPRS para Arduino a un Servidor Web el cual obtiene, almacena y procesa la información. Además se implementó una interfaz web y una aplicación en Android para que el usuario pueda acceder a los datos almacenados en el servidor desde un computador o un dispositivo móvil. El ingreso a los servicios ofrecidos por los terminales fue configurado para que se lo realice por medio de autenticación, añadiendo el nombre de usuario y la contraseña, lo cual el usuario establece la información mencionada por medio de un apartado disponible para el registro. Se realizó la fase de pruebas para conocer la precisión del dispositivo en diferentes situaciones y escenarios.

PALABRAS CLAVE

- **MASCOTAS**
- **ANDROID**
- **GPS**
- **SIGNOS VITALES**
- **SERVIDOR WEB**

ABSTRACT

The project proposes a prototype to avoid the loss of pets. A state of the art study about pet monitoring system with mobile devices in the global market was made to determine the characteristics, techniques and conditions that are necessary to develop and implement the project. A harness was designed to get the geographic position and the vitals of the pet, the movement and body temperature were chosen as vitals. The GPS and body temperature sensors were tested to determine the precision and get reliable settings for the project. The information gathered by the sensors is processed through an Arduino programmable platform and sent through a GPS/GSM/GPRS module to a web server that records, saves and processes the information. A web interface and an Android app were also implemented, so the user could access to the data stored on the server from any computer or Android mobile device. the access to the services offered was configured to ask for an username and a password as an authentication method to retrieve the information. The testing phase to determine the accuracy of the device in different situations and scenarios was performed.

KEYWORDS:

- **PETS**
- **ANDROID**
- **GPS**
- **VITAL SIGNS**
- **WEB SERVER**

CAPÍTULO 1
1. DEFINICIÓN DEL PROYECTO

1.1. Antecedentes.

En el Ecuador no existe una preocupación y responsabilidad del estado con la protección animal, a pesar del "Reglamento de Tenencia y Manejo Responsable de perros" dado por el Acuerdo Ministerial 116, publicado en el registro oficial 532, el 19 de Febrero del 2009. Actualmente no existen leyes y penalizaciones claras relacionadas con el incumplimiento de normas en la tenencia de animales, por tal motivo, existen varias fundaciones sin fines de lucro encargadas del control de mascotas y perros callejeros. Actualmente se encuentra en proceso de aprobación la Ley Orgánica de Bienestar Animal (L.O.B.A.), presentado a la Asamblea Nacional Constituyente en el mes de Octubre del año 2014, la cual, si es que es aprobada, será la encargada de poner lineamientos claros al trato con los animales y definir penas por su maltrato.

Hoy en día, la tecnología es algo que se encuentra disponible para muchas personas en el mundo, los teléfonos inteligentes sobre el sistema operativo Android, tienen muchos adeptos, y se encaminan para en un futuro reemplazar los computadores personales en ciertas aplicaciones, por su comodidad, portabilidad y la capacidad de procesamiento que están alcanzando.

El Sistema Operativo Android, está basado en un núcleo Linux, fue diseñada para dispositivos móviles con pantalla táctil y se ha convertido en el principal sistema para aplicaciones móviles que existe actualmente junto con IOS de la empresa Apple Inc. La ventaja de la programación en Android es su facilidad de acceso al lenguaje y a las herramientas para el desarrollo de una aplicación además de su amplio contenido de librerías.

En Ecuador, existe poca difusión de las personas con respecto a las aplicaciones móviles relacionadas con el control de mascotas, que involucre un sistema de rastreo y recuperación de información del animal, por lo que los dueños de animales domésticos optan por procedimientos ineficaces para su protección, como collares

con cuerda o con su encierro, lo que puede llevar a diferentes problemas como, daños físicos y de comportamiento animal.

1.2. Justificación e importancia

Con la lucha de organizaciones sociales defensoras del bienestar animal, el Ecuador se encuentra en proceso de cambio en sus leyes y penas relacionadas con la tenencia de animales, por lo que la tecnología se la puede utilizar como alternativa en una situación determinada que afecta la vida de un animal, como es su extravío. La ley L.O.B.A actualmente en vías de aprobación sostiene que el bienestar de un animal tiene relación con "El Plan del Buen Vivir" planteado por el actual régimen, y en consecuencia afecta a la vida de las personas.

Un pequeño porcentaje de la inversión municipal está destinado a la esterilización de animales callejeros, lo cual ayuda a controlar su proliferación. En algunos lugares, las administraciones zonales se encargan de la esterilización, las cuales se realizan en lugares donde hay mayor cantidad de animales callejeros. La inversión actual del gobierno en turismo hace que los municipios pongan mayor atención a la restauración de lugares históricos y en algunas ocasiones los animales callejeros no ayudan a dar una buena imagen a la ciudad, que en la actualidad se encuentra promocionada turísticamente a nivel mundial de muchas maneras.

Hoy en día, imprimir folletos y papeles informando sobre la pérdida de un animal doméstico es algo que no genera resultados, ya que considero que es muy poco probable que una persona se detenga a observar y a intentar identificar al animal perdido. Además, no es necesario que la mascota este perdida para que el dueño quiera conocer su ubicación y si se encuentra bien, por ejemplo al enviar a la mascota a la veterinaria o al salir de viaje, mucha gente tiene dudas de su trato y a veces no hay certeza de su bienestar, por lo que el envío de la ubicación y de los signos vitales darían tranquilidad al dueño en diferentes situaciones. Considerando que gran cantidad de personas actualmente tienen acceso a la tecnología, y con el surgimiento de nuevos avances en las tecnologías de comunicación, lo cual hace que las operadoras telefónicas se concentren en dar mayor cobertura y que el recurso de Internet se lo pueda utilizar en cualquier lugar.

El proyecto se justifica plenamente ya que mediante una aplicación capaz de informar sobre la ubicación y signos vitales de una mascota, se daría tranquilidad a su dueño. Muchas personas se encuentran en busca de diferentes soluciones para proteger a su mascota. En la actualidad, las personas de clase media tienen la capacidad de cubrir diferentes gastos relacionados con sus animales domésticos, como es la alimentación, salud o la limpieza del animal, por lo que no se dudaría en optar por una alternativa para su recuperación. La creación de fundaciones encargadas del rescate animal, son un indicador de la preocupación de mucha gente que se unen a organizaciones como P.A.E. las cuales no tienen fines de lucro y sólo se impulsan por voluntariados y la preocupación de las personas por los animales.

1.3. Alcance del proyecto

El proyecto ejecuta el monitoreo de ubicación de las mascotas, el sistema de localización que se integra en el arnés del animal, contiene una antena GPS para la obtención de coordenadas de ubicación geográfica. Además de las coordenadas de ubicación, el proyecto recupera el movimiento y la temperatura del animal, las medidas mencionadas son tomadas como signos vitales, que sirve de referencia para conocer si el animal se encuentra en movimiento y con temperatura corporal externa normal. Mediante una serie de muestras y validaciones se conoce si una mascota se encuentra con signos vitales. Por medio de tecnología GSM/GPRS se envía la información GPS y de signos vitales recuperada hacia el servidor web, en donde se procesa y almacena la información en la tabla de una base de datos. Por medio de los terminales se integra la capacidad de consulta del usuario de la información de ubicación y signos vitales recuperada por el arnés. El sistema aloja un registro de usuarios, con la capacidad de alojar la cantidad de usuarios que se deseen registrar en el sistema, cada usuario consta de un perfil, en donde puede realizar la consulta de la información enviada por el arnés correspondiente a su mascota. Por medio de una etapa de pruebas se estableció la precisión de la información enviada por el arnés, sus limitaciones y desventajas.

1.4. Objetivos

1.4.1. Objetivo general de la investigación

Diseñar e implementar un sistema de control de mascotas en el Sistema Operativo Android, basado en un sistema cliente/servidor para el registro de la ubicación del animal, y de sus signos vitales.

1.4.2. Objetivos Específicos

a. Establecer el estado del arte de los sistemas de monitoreo de mascotas actuales sobre dispositivos móviles.
b. Diseñar e implementar la tarjeta Arduino con el sensor de signos vitales y ubicación.
c. Diseñar el servidor web y la base de datos para el almacenamiento de la ubicación y los signos vitales del animal.
d. Diseñar e implementar una interfaz amigable con el usuario para su acceso mediante el S.O. Android y vía web.
e. Establecer escenarios de pruebas para determinar la funcionalidad del sistema.

1.5. Estado del Arte.

La tecnología ayuda a cubrir varios problemas en diferentes áreas, ya sea social, empresarial, familiar, etc. El avance tecnológico y la facilidad de acceso a herramientas para el desarrollo, ayudan a utilizar la tecnología en todo lo que el hombre se propone, hoy en día es muy difícil encontrar algo que no se encuentre desarrollado.

A nivel mundial existen varios dispositivos de monitoreo inteligentes para mascotas disponibles en el mercado, los cuales mediante una investigación se ha identificado los más populares y de fácil accesibilidad para el usuario promedio (Ver Tabla 1).

Tabla 1

Características de dispositivos para geolocalización de mascotas en el mercado

Nombre Comercial	Monitoreo	Datos Técnicos	Costo	Peso	Año
PetPace	Temperatura Pulso Respiración Calorías	Indicadores LED. Batería recargable LiPo 250mAh. Ethernet 10/100 BASE-T.	$149.95 14.95/mes	43gr	2012
HeyrexVet	Descanso Ejercicio Alimentación diaria. Actividad diaria.		$149.95 $9/mes	-	2013
Whistle	Ubicación Actividades				2013
FitBark	Actividades Descanso. Ubicación	Bluetooth 4.0 Compatibilidad IOS Compatibilidad Android Duración 14 días de carga		8gr	2013
WÜF	Actividades Ubicación	Simulador web de la app	$129		2014
Otto Petcare System	Actividad física Comparación de datos.	Acelerómetros y giroscopios integrados. Dispensador de alimentos Wifi			2015
Tractive	Ubicación en tiempo real. Actividades.	Entrenador de perros. Vibración en el collar para comunicación			2015
Voyce	Frecuencia cardiaca y respiratoria, Descanso Actividad física. Calorías Peso perdido Salud	Informe de distancia recorrida. Informe de chequeos médicos			2015

La Figura 1 muestra un ejemplo gráfico, para que el lector tenga una idea visual de algunos tipos de dispositivos mencionados en la Tabla **1**. Todos los collares tienen muchas características en común y su eficacia en su funcionamiento depende de las pruebas realizadas por los usuarios.

Figura 1 Collares inteligentes en el mercado, 1. PetPace, 2. HeyRex, 3. Tractive, 4. Whistle

Por lo general los collares mencionados en la Tabla 1, son no invasivos, por ejemplo, el collar PetPace monitorea la salud de la mascota mediante el seguimiento de forma inalámbrica de los signos vitales, y una serie de parámetros físicos y de comportamiento, y envía dicha información a un servidor en la nube para futuros reportes y análisis de datos. No realiza la medición de la temperatura vía rectal, forma habitual de medición de temperatura de los animales y la cual es usada por los veterinarios, y genera mayor precisión, por lo que el collar muy probablemente use un termómetro auricular infrarrojo para la medición de la temperatura (NexGen Dog, 2014), la cual según estudios y comparaciones no tiene una exactitud con la temperatura real en comparación con el termómetro rectal, lo que puede ocasionar una falla en la medición real y por ende el verdadero estado de salud del animal. (Greer, 2007)

En los collares inteligentes para mascotas, el pulso cardiaco y la frecuencia respiratoria pueden ser una información muy valiosa para analizar el estado de salud de la mascota, pero se debe tomar en cuenta que la diferencia de pulsaciones por minuto varía según la raza del perro, es decir si se tiene los datos exactos de la frecuencia cardiaca normal por animal se puede realizar una comparación realmente precisa y un análisis del estado de salud real del animal. (NexGen Dog, 2014)

Se puede considerar que PetPace y Voyce, según los datos que envía, son collares para uso veterinario, ya que una persona promedio no tiene la capacidad técnica de analizar datos como las calorías quemadas, si el pulso cardiaco es normal, o si las actividades promedio en el día son normales, ya que esto depende de distintos factores como la raza del perro y la alimentación del mismo, la cual debe ser la recomendada por un profesional de la salud en animales domésticos. Además un punto importante a recalcar es que no se puede reemplazar la opinión de un veterinario con datos y análisis realizados de forma sistemática. (NexGen Dog, 2014)

La forma no invasiva de los collares enumerados en la Tabla 1 es una característica importante de los dispositivos en mención, cuando una mascota tiene problemas cardiacos se debe realizar un monitoreo continuo, para esto los veterinarios utilizan un instrumento conocido como “grabadora Holter”, este tipo de dispositivos requieren llevar varios cables conectados al animal, se requiere afeitarlo , además de un vendaje para proteger al medidor, por lo que un collar del tipo no invasivo, puede ser muy útil realizando este tipo de mediciones y comparándolas con un estudio previo realizado por un veterinario para conocer las pulsaciones normales en el animal, con esto, se puede obtener una ventaja importante en el mismo ya que con un estudio previo dicho collar puede ser muy útil para un monitoreo constante en una determinada situación y riesgo de salud en una mascota. (Petpace, 2015)

El collar Voyce, utiliza una tecnología propia y patentada para el envío de ritmo cardiaco, un sensor que utiliza ondas de radio de baja frecuencia para medir la frecuencia cardíaca, y la frecuencia respiratoria (seguimiento de los movimientos musculares en el pecho del perrito), Para comercializar el collar, Voyce tuvo que conseguir la aprobación de la FCC, y muestra algo interesante, la exportación de los datos de ritmo cardiaco y frecuencia respiratoria en un archivo PDF, el cual se lo puede imprimir o enviar a un profesional para su respectivo análisis. (Philips, 2015)

Tomando en cuenta los puntos antes analizados, el pulso cardiaco de un perro requiere especialización por raza, lo que requiere de un profesional que guíe e indique las pulsaciones normales por cada raza de perro y tamaño, lo que nos indica que la automatización de este tipo de medidas biométricas no siempre es lo más adecuado.

El sitio Web CanineJournal realiza una comparación entre los collares que se encuentran en el mercado y son considerados líderes por la revista en el envío de la ubicación del animal y en la medición de actividad del perro. El sitio web realiza un análisis de las características de cada collar que consideran importantes y llegan a la conclusión que el collar Fitbark es el más fácil de usar, el collar Tagg es el más exacto, y Whistle es el mejor para realizar un seguimiento cercano de la salud de la mascota. (Wilson, 2015). Se debe considerar que para realizar un análisis verdadero de los collares mencionados en el artículo, la persona los debe usar en su mascota por un tiempo determinado y definir cuál le parece el mejor en diferentes ocasiones y circunstancias, el sitio web toma opiniones de diferentes personas y comentarios de los mismos.

CAPÍTULO II
2. MARCO TEÓRICO

2.1. Marco Legal

2.1.1. Reglamento Nacional de Tenencia de Perros

Dado por El Acuerdo Ministerial 116, publicado en el Registro Oficial 532 el 19 de Febrero del 2009, se elaboró el Acuerdo Interministerial para la Tenencia Responsable de Perros, el cual entró en vigencia en Agosto del mismo año. Según el Reglamento, las obligaciones que las personas deben tener con los animales son:

- Otorgar condiciones de vida adecuadas a las características del animal. (PAE)
- Educar, socializar e interactuar con el perro en la comunidad.
- Mantener únicamente el número de perros que las normas de bienestar animal permiten.
- Recoger y disponer sanitariamente los desechos del animal.
- Cuidar que los perros no causen molestias a los vecinos.
- Entre otras. (PAE)

Entre las prohibiciones que las personas deben tener con los animales según El Acuerdo Ministerial son:

- Maltratar, golpear o someter al animal.
- Abandonar o mantener en estado de aislamiento.
- Encadenar, enjaular o confinar permanentemente en terrazas, patios, balcones o similares.
- Envenenar masivamente perros propios o ajenos.
- Usar la imagen de perros para simbolizar maldad, agresividad o peligro.
- Entre otras. (PAE)

2.1.2. Código Orgánico Integral Penal

En el Código Orgánico Integral Penal, se establece en el Artículo 249, “Maltrato o muerte de mascotas o animales de compañía”, que la persona que por acción u omisión cause daño, produzca lesiones, deterioro a la integridad física de una mascota o animal de compañía, será sancionada con pena de cincuenta a cien horas de servicio comunitario. Si se causa la muerte del animal será sancionada con pena privativa de libertad de tres a siete días. Se exceptúan de esta disposición, las acciones tendientes a poner fin a sufrimientos ocasionados por accidentes graves, enfermedades o por motivos de fuerza mayor, bajo la supervisión de un especialista en la materia.

El Artículo 250, establecido en el Código Orgánico Integral Penal establece sanciones relacionadas con las peleas entre perros. (Ministerio de Justicia, Derechos Humanos y Cultos, 2014)

2.1.3. Código Civil Libro IV

El Art. 2226 establecido en el Código Civil, establece que el dueño de un animal es responsable de los daños causados por éste, aún después que se haya suelto o extraviado; salvo que la soltura, extravío o daño no puedan imputarse a culpa del dueño o del dependiente encargado de la guarda o servicio del animal. Lo que se dice del dueño se aplica a toda persona que se sirva de un animal ajeno; salva su acción contra el dueño, si el daño ha sobrevenido por una calidad o vicio del animal, que el dueño con mediano cuidado o prudencia debió conocer o prever, y de que no le dio conocimiento. (Código Civil Ecuatoriano Art.2226, 2005)

2.1.4. Ordenanzas Municipales del Distrito metropolitano de Quito

Según la investigación realizada acerca de las ordenanzas municipales de la ciudad de Quito, no consta en el registro de ordenanzas, ninguna que se encuentre relacionada con animales domésticos y mascotas, y que haya sido aprobada en años actuales. La última resolución con respecto a Ordenanzas Municipales de animales domésticos es la N° 128 expedida en el año 2004.

La Ordenanza Municipal N° 128 considera distintos puntos para la resolución y aprobación de la Ordenanza (Ordenanza Metrolpolitana N°128, 2004), los cuales son:

- En el numeral 1 del Art. 12 de la Ley Orgánica de Régimen Municipal al determinar los fines esenciales del Municipio se establece "Procurar el bienestar material y social de la colectividad…"; (Ordenanza Metrolpolitana N°128, 2004)
- La letra i) del Art. 164 de la Ley Orgánica de Régimen Municipal al referirse a la Higiene y Asistencia Social, establece que a la administración le compete determinar las condiciones en que se han de mantener los animales domésticos e impedir su vagancia en las calles y demás lugares públicos. (Ordenanza Metrolpolitana N°128, 2004).
- En el Registro Oficial N° 203 del 4 de Noviembre del 2003, se publicó el Reglamento sobre la tenencia de perros y gatos. (Ordenanza Metrolpolitana N°128, 2004).
- Actualmente, en la ciudad de Quito, se evidencia un preocupante descuido de algunos propietarios de animales domésticos, en especial de perros, los cuales han llegado inclusive a agredir a seres humanos, poniendo en riesgo la vida de las personas. (Ordenanza Metrolpolitana N°128, 2004).
- Es urgente que el Municipio de Quito, tome acciones encaminadas a solucionar esta problemática. (Ordenanza Metrolpolitana N°128, 2004).

Considerando los puntos antes mencionados. Se pone en conocimiento los Artículos considerados importantes y relacionados con el proyecto. El Art 1. de la Ordenanza Municipal N° 128 establece que el objetivo de dicha Ordenanza, en la cual fija las normas básicas para el debido control y las obligaciones que deben cumplir los propietarios y responsables del cuidado de mascotas, a fin de evitar accidentes, transmisión de enfermedades y establecer sanciones por su incumplimiento. (Ordenanza Metrolpolitana N°128, 2004)

El Art. 2 de la Ordenanza Municipal N°128 establece los responsables de la aplicación de la normativa, la cual se encarga de su cumplimiento La Municipalidad del Distrito Metropolitano de Quito mediante las Administraciones Zonales,

Comisarios Metropolitanos, Policía Metropolitana y Veedores Cívicos Ad_Honorem. (Ordenanza Metrolpolitana N°128, 2004)

En el Art. 3 numeral 1, se establece las condiciones para la tenencia de animales domésticos, indica que los dueños o en poder de quien se encuentren los perros, son los responsables de la manutención y condiciones de vida. El numeral 2 establece que los perros y otros animales domésticos, deben permanecer en el domicilio de su propietario, o en lugares adecuadamente cerrados que impidan su evasión, con las seguridades necesarias. Los animales podrán circular por las vías y espacios públicos, únicamente en compañía de sus propietarios con el correspondiente collar en el que conste el nombre y la dirección del propietario, sujetos de tal manera que impida su fuga. (Ordenanza Metrolpolitana N°128, 2004). Además se debe cumplir con lo que establece la letra e) de la sección III de la Ordenanza N° 100, publicada en el Registro Oficial N° 194 del 21 de Octubre del 2003, que establece las responsabilidad de los propietarios con respecto a la recolección de los residuos sólidos urbanos, domésticos, comerciales, industriales y biológicos (Ordenanza Metrolpolitana N°128, 2004)

El Art.5, establece que los perros que sean de temperamento agresivo o comportamiento impredecible, capaces de provocar a las personas lesiones sumamente graves, deben mantenerse dentro del domicilio en condiciones muy seguras, cuando estos deban salir de sus domicilios lo deberán hacer acompañados de su propietario (Ordenanza Metrolpolitana N°128, 2004). El Art. 6 establece que las distintas molestias que puedan ocasionar un can o animal doméstico a sus vecinos es la responsabilidad de su dueño (Ordenanza Metrolpolitana N°128, 2004).

Se realizó un análisis de la Ordenanza Municipal N° 128, la cual según la investigación realizada es la más actual con respecto a ordenanzas relacionadas con los animales domésticos, se tomó en consideración los artículos relacionados con el proyecto realizado.

El Art. 3 de La Ordenanza Municipal N° 128 nos orienta en la importancia legal de conocer la ubicación de nuestra mascota, por lo que es importante conocer los riesgos legales actuales con respecto a la tenencia de animales.

2.1.5. Proyecto de Ley “Ley Orgánica de Bienestar Animal”

Debido al vacío legal en que actualmente se encuentran las leyes ecuatorianas con respecto a los animales domésticos en nuestro país, diversas organizaciones animalistas, se unieron para realizar una propuesta a la Asamblea Nacional Constituyente, la cual plantea un Proyecto de Ley completo con respecto al bienestar animal, dicho proyecto se denomina L.O.B.A. (Ley Orgánica de Bienestar Animal). El proyecto plantea 70 artículos, los cuales se basan en fundamentos como la violencia interpersonal, salud pública, derechos de la naturaleza, bienestar animal y el buen vivir. (El Comercio, 2014).

Dicho proyecto expone diversas causas de la necesidad de aprobación de la misma, los motivos y el proyecto de ley completo se encuentran en la página web del proyecto LOBA. El proyecto de ley fue presentado a la Asamblea Nacional Constituyente el 30 de Octubre del 2014, y se realizó la primera sesión de audiencias sobre el proyecto de Ley Orgánica de Bienestar Animal LOBA el 15 de Marzo del 2015, en la Comisión de Biodiversidad de la Asamblea Nacional, encargada de su tratamiento luego de haber sido presentada, por organizaciones de la sociedad civil conjuntamente con asambleístas. Actualmente se encuentra en vías de aprobación.

2.2. Sistema Operativo Android

2.2.1. ¿Qué es Android?

Android es un sistema operativo y una plataforma de código libre, basado en un kernel Linux, para teléfonos móviles. Además, este sistema operativo es usado (aunque no es muy habitual), por tablets, netbooks, reproductores de música e incluso PC’s. Android. El sistema operativo permite su desarrollo en un entorno de trabajo (framework) de Java, aplicaciones sobre una máquina virtual Dalvik (una variación de la máquina de Java con compilación en tiempo de ejecución). Además, lo que lo diferencia de otros sistemas operativos, es que cualquier persona que conozca un poco de programación, puede crear nuevas aplicaciones, widgets, o incluso, modificar el propio sistema operativo, dado que Android es de código libre,

por lo que conociendo y teniendo bases sobre el desarrollo en el lenguaje Java, va a ser muy fácil comenzar a programar en esta plataforma. (Baez, y otros)

El sistema operativo Android es una plataforma de código abierto, significa que no se encuentra ligado a un fabricante de hardware en particular. La apertura de Android es la característica que le permitió ganar mercado rápidamente. Cualquier fabricante de hardware puede diseñar y vender dispositivos con el sistema Android integrado, el código fuente de Android está disponible en "http://source.android.com" en el cual se puede verlo y modificarlo, y permite a los fabricantes crear interfaces de usuario personalizadas y añadir características incorporadas a los dispositivos, también permite analizar el código para ver el funcionamiento de una tarea determinada. (Felker & Dobbs, 2011)

Figura 2 Logo Android

Fuente: (García, 2012)

2.2.2. Importancia del desarrollo en Android

Con Android se puede desarrollar aplicaciones, que se encuentren disponibles para millones de usuarios en todo el mundo, publicarlas y comercializarlas si fuera el caso. La programación está basada en los lenguajes de programación Java y XML, con muchas facilidades para su desarrollo, compilación del programa y pruebas, contiene incluso un simulador capaz de imitar el comportamiento de ciertas aplicaciones, el cual se encuentra incorporado en el IDE de desarrollo nativo del sistema, Android Studio IDE.

El Android Market (Google Play) pone la aplicación a la disposición de los usuarios fácilmente. Los usuarios no tienen que buscar en Internet una aplicación específica para instalar, sino que simplemente ingresan desde su móvil al Google Play, el cual se encuentra preinstalado en el dispositivo Android, y en él, se tiene

acceso a todas las aplicaciones disponibles y seguras de Android, esto además de beneficiar al usuario final, beneficia al desarrollador por la facilidad de acceso a las aplicaciones desarrolladas (Felker & Dobbs, 2011), las aplicaciones desarrolladas, pueden ejecutarse en muchos dispositivos con diferentes tamaños de pantalla y resoluciones.

2.2.3. El código fuente de Android

Android es una pila de software de código abierto creado para una amplia gama de dispositivos con diferentes factores de forma. Los propósitos principales de Android son la creación de una plataforma de software abierto, disponible para los operadores, OEMs y desarrolladores y hacer que sus ideas innovadoras sean una realidad, para introducir un producto de éxito, en el mundo real que mejora la experiencia móvil de los usuarios. (Android, 2014).

El proyecto también quiso asegurar que no exista ningún punto central de fracaso, donde una industria podría restringir o controlar las innovaciones de cualquier otra. El resultado es una producción de calidad, con código fuente abierto para la personalización y portabilidad. (Android, 2014).

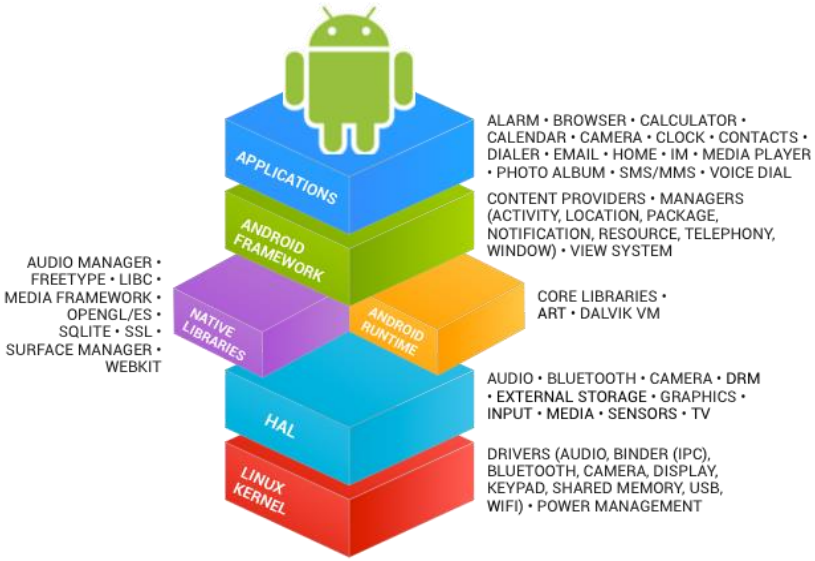


Figura 3 Pila Android

Fuente: (Android, 2014)

El proyecto de código abierto Android (AOSP) mantiene una pila de software completa (Ver Figura 3), para ser portado por los fabricantes de equipos originales y otros ejecutores del dispositivo, se ejecuta en su propio hardware. Para mantener la calidad de Android, Google ha invertido en ingenieros, a tiempo completo, gerentes de producto, diseñadores de interfaz de usuario, probadores de control de calidad, y otras funciones necesarias para llevar Android a los dispositivos modernos de mercado. (Android, 2014)

De acuerdo con ello, se mantiene una serie de "líneas de código" separando la versión estable actual de Android del trabajo experimental inestable.

La Figura 4 muestra a nivel conceptual cómo se estructura el código AOSP. En la cual son referidos como "líneas de código" en lugar de "ramas", simplemente porque en un momento dado puede haber más de una rama en una "línea" código dado. Por ejemplo, cuando se corta una autorización, que puede o no puede llegar a ser una nueva rama en base a las necesidades del momento. A continuación enumeramos algunas características del código AOSP de Android (Android, 2014):

- En cualquier momento dado, hay una última versión actual de la plataforma Android. Esto normalmente toma la forma de una rama en el árbol (Android, 2014).
- Constructores de dispositivos y colaboradores trabajan con la actual versión más reciente, corrigiendo errores, antes del lanzamiento de nuevos dispositivos, experimentando con nuevas características, y así sucesivamente (Android, 2014).
- Paralelamente, Google trabaja internamente en la próxima versión de la plataforma Android y un marco de acuerdo a las necesidades y objetivos del producto. Google desarrolla la próxima versión de Android para trabajar con un socio del dispositivo en un dispositivo insignia cuyas especificaciones son elegidos para impulsar Android en la dirección que Google cree que debería ir (Android, 2014).
- Cuando la "n + 1" th versión está lista, se publica en al árbol de origen público y se convierte en la nueva versión más reciente (Android, 2014).

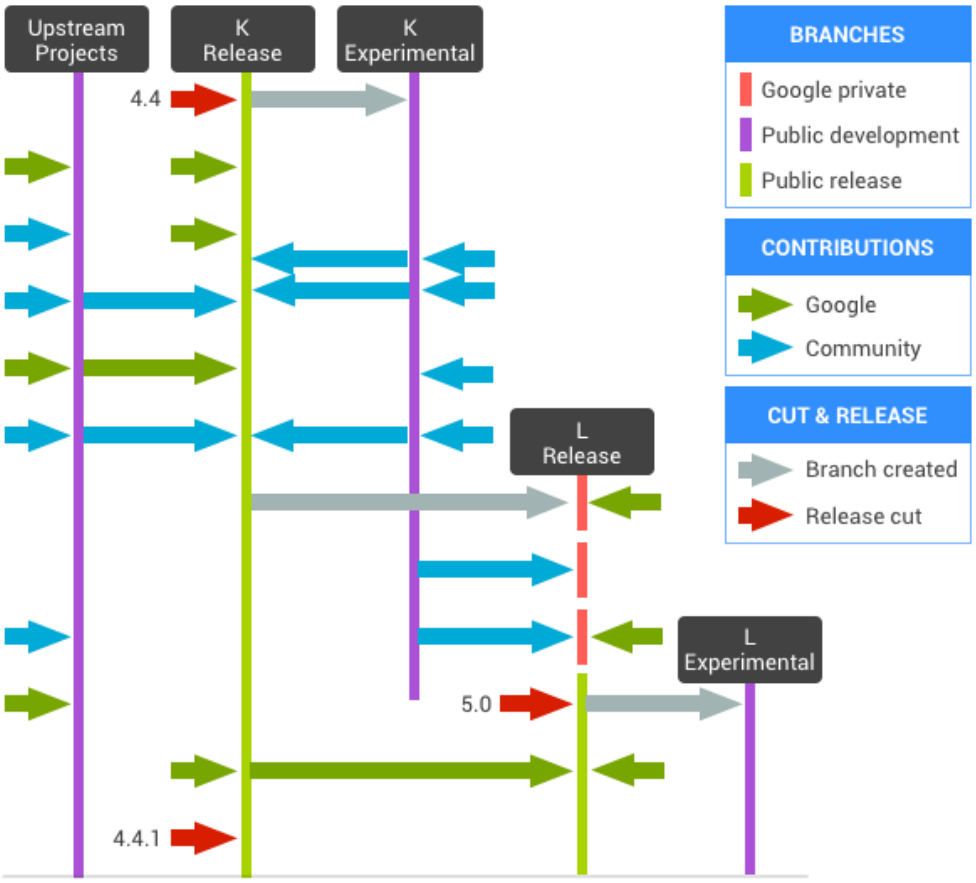


Figura 4 Estructura del Código AOSP en Android

Fuente: (Android, 2014)

2.2.4. Versiones del Sistema Operativo Android

Android ha puesto en el mercado diversas versiones de su Sistema Operativo, el nombre de cada versión corresponde a un dulce, en la Tabla 2 se muestra las versiones de Android que se han publicado hasta la fecha.

Tabla 2

Versiones Publicadas del Sistema Operativo Android

Nombre del código	Versión	Nivel API
Sin nombre de código	1.0	API level 1
Sin nombre de código	1.1	API level 2
Cupcake	1.5	API level 3, NDK 1
Donut	1.6	API level 4, NDK 2
Eclair	2.0	API level 5
Eclair	2.0.1	API level 6
Eclair	2.1	API level 7, NDK 3
Froyo	2.2.x	API level 8, NDK 4
Gingerbread	2.3.- 2.3.2	API level 9, NDK 5
Gingerbread	2.3.3 - 2.3.7	API level 10
Honeycomb	3.0	API level 11
Honeycomb	3.1	API level 12, NDK 6
Honeycomb	3.2.x	API level 13
Ice Cream Sandwich	4.0.1 - 4.0.2	API level 14, NDK 7
Ice Cream Sandwich	4.0.3 - 4.0.4	API level 15, NDK 8
Jelly Bean	4.1.x	API level 16
Jelly Bean	4.2.x	API level 17
Jelly Bean	4.3.x	API level 18
KitKat	4.4 - 4.4.4	API level 19
Lollipop	5.0	API level 21
Lollipop	5.1	API level 22
Marshmallow	6.0	API level 23

Fuente: (Android, 2014)

2.2.5. Desarrollo en Android

El desarrollo de aplicaciones en Android se realiza mediante el lenguaje Java en su mayor parte y pequeñas partes del espacio de trabajo son programadas en XML, como por ejemplo el Android Manifiest de Android Studio IDE. Existen varias herramientas disponibles para el desarrollo en Android, por ejemplo el IDE oficial Android Studio y el ADT (Android Development Tools) de Eclipse, entre los más populares, existe además herramientas web como App Inventor, las cuales sirven

sólo para el desarrollo de ciertas aplicaciones básicas y no cubren en su desarrollo herramientas importantes en Android como por ejemplo la conexión a una base de datos remota, lo que limita su uso para un desarrollo completo. La instalación y análisis de la estructura de proyectos en Android Studio IDE, el cual es el IDE utilizado en el proyecto se encuentra especificado en el Anexo 1.

2.3. Plataforma Programable Arduino

2.3.1. ¿Qué es Arduino?

Arduino es una plataforma electrónica de código abierto basado en hardware y software fácil de usar, utilizado para la creación de prototipos. Arduino es capaz de detectar el medio ambiente mediante la recepción de las aportaciones de diversos sensores, por ejemplo medir la temperatura del medio ambiente, un dedo sobre un botón, etc. y pueden convertirlos en una salida, como la activación de un motor, el encendido de un LED, publicar algo en línea, entre otros. Para el procesamiento de dichos sensores, Arduino usa su propio lenguaje de programación el cual envía un conjunto de instrucciones al hardware Arduino para que actúe según el requerimiento del desarrollador. (Arduino, 2016).

A través de los años Arduino ha sido el cerebro de miles de proyectos, desde objetos cotidianos a instrumentos científicos complejos. Una comunidad mundial de fabricantes, estudiantes, aficionados, artistas, programadores y profesionales, han reunido sus contribuciones en torno a esta plataforma de código abierto, también han añadido una gran cantidad de conocimiento accesible que puede ser de gran ayuda tanto para principiantes como para expertos. (Arduino, 2016).

2.3.2. Importancia de la tarjeta Programable Arduino

Gracias a su sencilla y accesible experiencia de usuario, Arduino se ha utilizado en miles de diferentes proyectos y aplicaciones. El software de Arduino es fácil de usar para los principiantes, pero lo suficientemente flexible para los usuarios avanzados. Se ejecuta en Mac, Windows y Linux. Los maestros y los estudiantes lo

utilizan para construir instrumentos científicos de bajo coste, demostrar los principios de química y física, o para empezar con la programación y la robótica. Los diseñadores y arquitectos construyen prototipos interactivos, músicos y artistas lo utilizan para instalaciones y experimentar con nuevos instrumentos musicales. Los responsables, por supuesto, lo utilizan para construir muchos de los proyectos expuestos en la Maker Fair. Arduino es una herramienta clave para aprender cosas nuevas. Cualquier persona, niños, aficionados, artistas, programadores, pueden comenzar a probar y desarrollar sus ideas, simplemente siguiendo la instrucciones paso a paso de un kit, o compartir ideas en línea con otros miembros de la comunidad Arduino. (Arduino, 2016)

Hay muchos microcontroladores y plataformas de microcontroladores disponibles para computación física, por ejemplo, Parallax Basic Stamp, de Netmedia BX-24, Phidgets, Handyboard del MIT, y muchos otros ofrecen una funcionalidad similar. Todas estas herramientas toman los detalles de la programación de microcontroladores y se envuelve en un paquete fácil de usar. Arduino también simplifica el proceso de trabajar con microcontroladores, pero ofrece algunas ventajas para los profesores, estudiantes y aficionados interesados sobre otros sistemas, entre las principales ventajas de la plataforma podemos enumerar:

- Fácil acceso, las placas Arduino son relativamente de bajo costo en comparación con otras plataformas de microcontroladores. La versión menos cara del módulo Arduino puede ser ensamblado a nano, e incluso los módulos pre montados cuestan menos de $ 50 (Arduino, 2016).
- Multiplataforma - El software de Arduino (IDE) se ejecuta en Windows, Macintosh OS X, y Linux. La mayoría de los sistemas de microcontroladores se limitan a Windows (Arduino, 2016).
- El software de Arduino (IDE) es fácil de usar para los principiantes, pero lo suficientemente flexible para los usuarios avanzados. Para los profesores, se basa convenientemente en el entorno de programación Processing, para que los estudiantes puedan aprender a programar en ese entorno y estén familiarizados con cómo funciona el IDE de Arduino (Arduino, 2016).

- El software de Arduino se publica como herramientas de código abierto, disponible para la extensión por los programadores experimentados. El idioma se puede ampliar a través de bibliotecas de C++ (Arduino, 2016).
- Los planes de las placas Arduino se publican bajo una licencia de Creative Commons, por lo que los diseñadores de circuitos experimentados pueden hacer su propia versión del módulo, ampliándolo y mejorándolo. Incluso los usuarios con poca experiencia pueden construir la versión de tablero del módulo con el fin de entender cómo funciona y ahorrar dinero (Arduino, 2016).

2.3.3. Hardware

Arduino posee muchas distribuciones, cada edición es pensada para un público en concreto o para una tarea específica, existe una gran variedad de placas oficiales, así como no oficiales de la distribución Arduino, además de diferentes módulos, dispositivos y sensores propios de Arduino, los cuales simplifican la comunicación de ciertos aplicativos y brindan a la placa una extensión de la capacidad de la misma. En la Tabla 3 se muestra una tabla demostrativa de las características de hardware que existe entre las diferentes distribuciones de la placa Arduino y Genuino (Marca utilizada para productos vendidos fuera de EEUU) que se encuentran en el mercado actualmente.

Tabla 3

Versiones y distribuciones de placas Arduino y Genuino actualmente en el mercado

Nombre	Procesador	Voltaje de Operación /Entrada	Velocidad CPU	Entradas / Salidas Análogas	Entradas y salidas digitales/PWM	USB
101	Intel Curie	3.3 V / 7-12 V	32 MHz	6/0	14/4	Regular
Due	ATSAM3X8E	3.3 V / 7-12 V	84 MHz	12/2	54/12	2 Micro
Gemma	ATtiny85	3.3 V / 4-16 V	8 MHz	1/0	3/2	Micro
LilyPad	ATmega168V ATmega328P	2.7-5.5 V / 2.7-5.5 V	8MHz	6/0	14/6	-
LilyPad SimpleSnap	ATmega328P	2.7-5.5 V / 2.7-5.5 V	8 MHz	4/0	9/4	
LilyPad USB	ATmega32U4	3.3 V / 3.8-5 V	8 MHz	4/0	9/4	Micro
Mega 2560	ATmega2560	5 V / 7-12 V	16 MHz	16/0	54/15	Regular
Mega ADK	ATmega2560	5 V / 7-12 V	16 MHz	16/0	54/15	Regular
Micro	ATmega32U4	5 V / 7-12 V	16 MHz	12/0	20/7	Micro
MKR1000	SAMD21 Cortex-M0+	3.3 V/ 5V	48MHz	7/1	8/4	Micro
Nano	ATmega168 ATmega328P	5 V / 7-9 V	16 MHz	8/0	14/6	Mini
Pro	ATmega168 ATmega328P	3.3 V / 3.35-12 V 5 V / 5-12 V	8 MHz 16 MHz	6/0	14/6	-
Pro Mini	ATmega328P	3.3 V / 3.35-12 V 5 V / 5-12 V	8 MHz 16 MHz	6/0	14/6	
Uno	ATmega328P	5 V / 7-12 V	16 MHz	6/0	14/6	Regular
Yun	ATmega32U4 AR9331 Linux	5 V	16 MHz 400MHz	12/0	20/7	Micro
Zero	ATSAMD21G18	3.3 V / 7-12 V	48 MHz	6/1	14/10	2 Micro

Fuente: (Arduino)

Las distribuciones de Arduino son muy similares, y por lo general existen diferencias con respecto a características de tamaño y funcionamiento, por lo que el desarrollador debe elegir la distribución que más se acopla a las necesidades del proyecto a realizar. Entre las características de hardware que condicionan la elección de la placa Arduino, según el tipo de proyecto en que se lo va a usar, tenemos, por ejemplo el número de pines analógicos y digitales (normales y de tipo PWM o modulados por ancho de pulso para simular una salida analógica) que se va a usar en el proyecto, con este pequeño análisis se puede descartar algunas placas más simples que no tengan suficientes pines o, al contrario, descartar las de mayor número de ellos para reducir los costes. Otra consideración a tomar en cuenta antes de adquirir una tarjeta programable es el tamaño del código que se va a generar para los "scketchs", un código muy largo, con muchas constantes y variables demanda una mayor cantidad de memoria flash para su almacenamiento. (PE, 2014)

La memoria RAM es la encargada de cargar los datos para su procesamiento inmediato, la memoria RAM va ligada al microcontrolador y afecta al procesamiento de la tarjeta Arduino. En los Arduinos oficiales podemos diferenciar entre dos tipos fundamentales de microcontroladores, los de 8 y 32 bits basados en ATmega AVR y los SMART basados en ARM de 32 bits y con un rendimiento superior, ambos creados por la compañía Atmel, cabe recalcar que la mayoría de proyectos se implementan en un procesador de 8 bits. (PE, 2014)

En la realización del proyecto se usó la placa Arduino UNO (Ver Figura 5), la cual es la más vendida y aconsejable sobre todo si se está comenzando y no se tiene mucha experiencia en el desarrollo en Arduino. Por lo que se va a realizar un análisis de los pines y conexiones de entrada de dicha tarjeta. (PE, 2014)

2.3.3.1. Arduino UNO

La plataforma Arduino UNO es una placa electrónica basada en el microcontrolador ATmega328P. La placa se encuentra conformada por 14 pines digitales de entrada y salida (de los cuales 6 se pueden usar como salidas PWM), 2 pines para transmisión y recepción serial, 6 entradas analógicas, un cristal de cuarzo de 16 Mhz, una entrada USB y un conector de alimentación (Arduino, 2016)

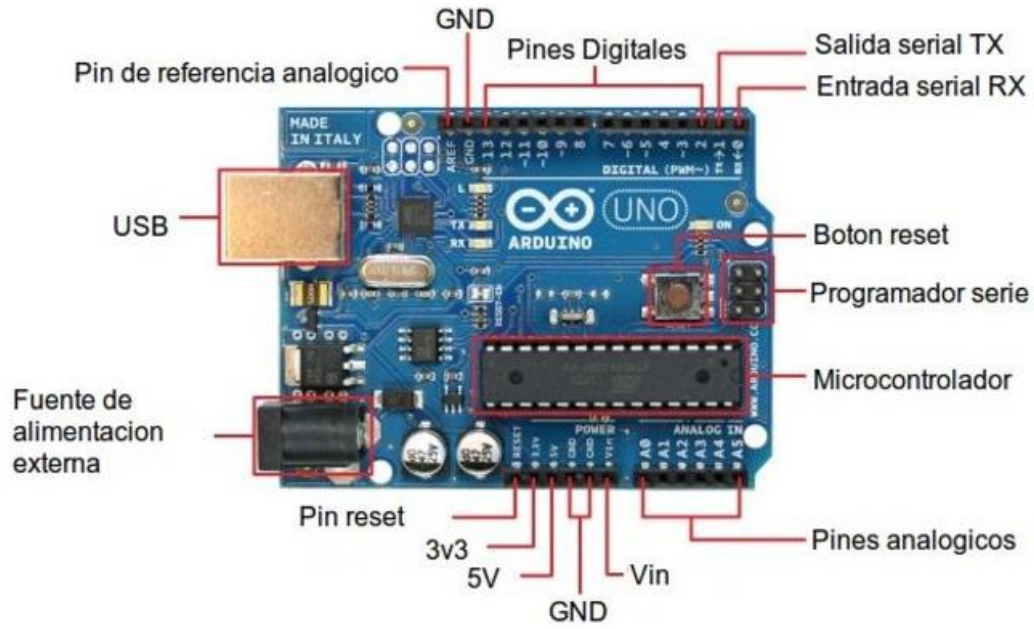


Figura 5 Placa Arduino UNO

Fuente: (PE, 2014)

Por medio de la Figura 5 se indica de una forma dinámica la distribución de los pines y conexiones de la placa Arduino UNO. En la Tabla 4 se muestra un resumen de las características técnicas de la tarjeta programable.

Tabla 4

Resumen de Especificaciones Técnicas de la plataforma Arduino UNO

Microcontrolador	**ATmega328P**
Tensión de funcionamiento	5V
Voltaje de entrada (recomendado)	7-12V
Voltaje de entrada (límite)	6-20V
E / S digitales	14 (de los cuales 6 proporcionan salida PWM)
PWM digital pines I / O	6
Pines de entrada analógica	6
Corriente continua para Pin I / O	20 mA
Corriente CC para Pin 3.3V	50 mA
Memoria flash	32 KB (ATmega328P) de los cuales 0,5 KB son utilizados por el gestor de arranque.
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Velocidad de reloj	16 MHz
Longitud	68,6 mm
Anchura	53,4 mm
Peso	25 g

Fuente: (Arduino, 2016)

La plataforma Arduino UNO está basada en el microcontrolador ATmega328P y es la versión de Arduino usada principalmente por principiantes y desarrolladores novatos, por su facilidad de conexión y accesibilidad a ejemplos.

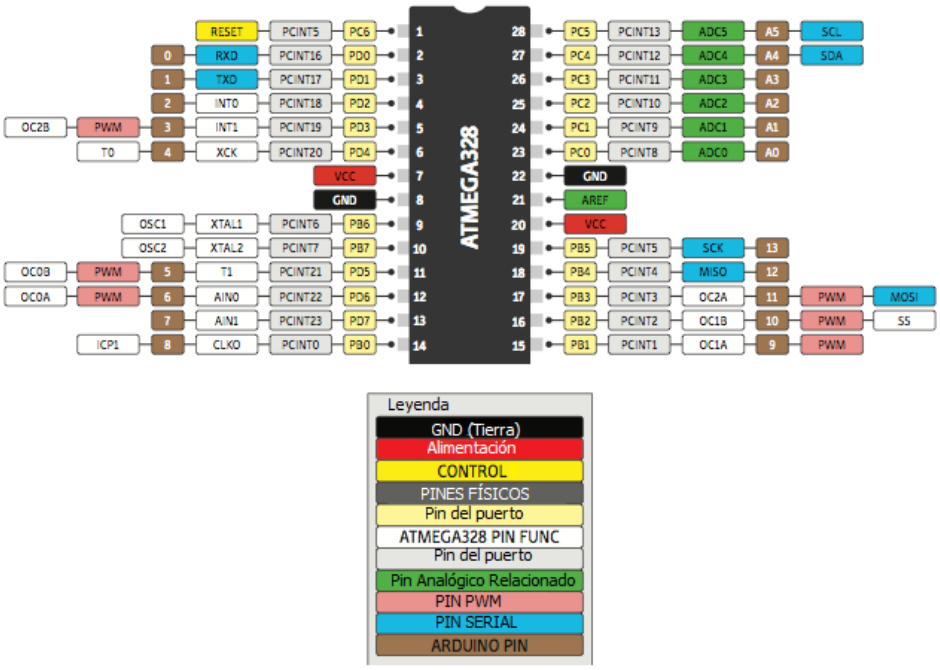


Figura 6 Esquema de pines del microcontrolador ATMEGA328

Fuente: (Joan, 2014)

El microcontrolador ATMEGA328P cuenta con 32KB de memoria flash, 2KB de memoria RAM y 1KB de memoria EEPROM. En la Figura 6 se muestra un esquema del microcontrolador ATMEGA328 y los pines que la conforman con su respectiva descripción. El microcontrolador se encuentra embebido en la tarjeta programable Arduino UNO, y se encarga del control general del sistema, la placa es un acoplamiento para facilitar la conexión y manipulación del microcontrolador. (Joan, 2014)

La Figura 7 muestra los componentes de la tarjeta Arduino UNO, es un acoplamiento para facilitar la conexión de interfaces hacia el microcontrolador, tiene 14 entradas/salidas digitales, 6 entradas analógicas, un oscilador de cristal de 16 Mhz, un conector USB, un conector de alimentación, una cabecera ICSP, y un botón de reset. Contiene todo lo necesario para apoyar el microcontrolador, basta con conectarlo a un ordenador con un cable USB, alimentarlo a un adaptador AC-DC o a una bateria.

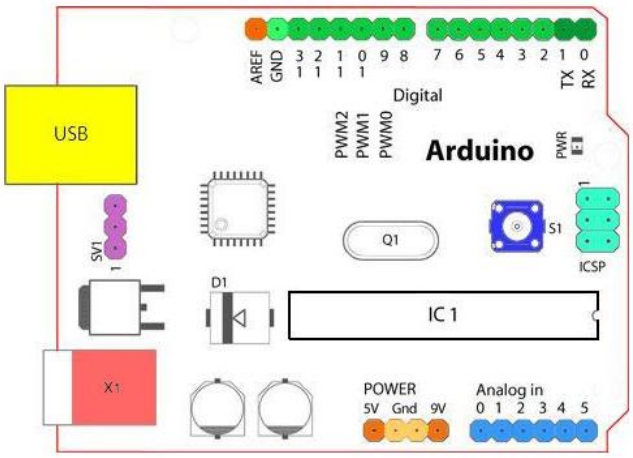


Figura 7 Componentes de la placa Arduino Uno.

Fuente: (Joan, 2014)

En la Figura 8 se muestra un diagrama de la tarjeta Arduino UNO con la distribución de pines y el microcontrolador ATMEGA 328 enbebido, en el transcurso del proyecto se va a especificar la función de los pines utilizados en el desarrollo del sistema, y el porque de su utilización., el software que se usa para cargar los comandos y sentencias es gratuito y se lo puede descargar desde la página oficial de Arduino. (Joan, 2014)

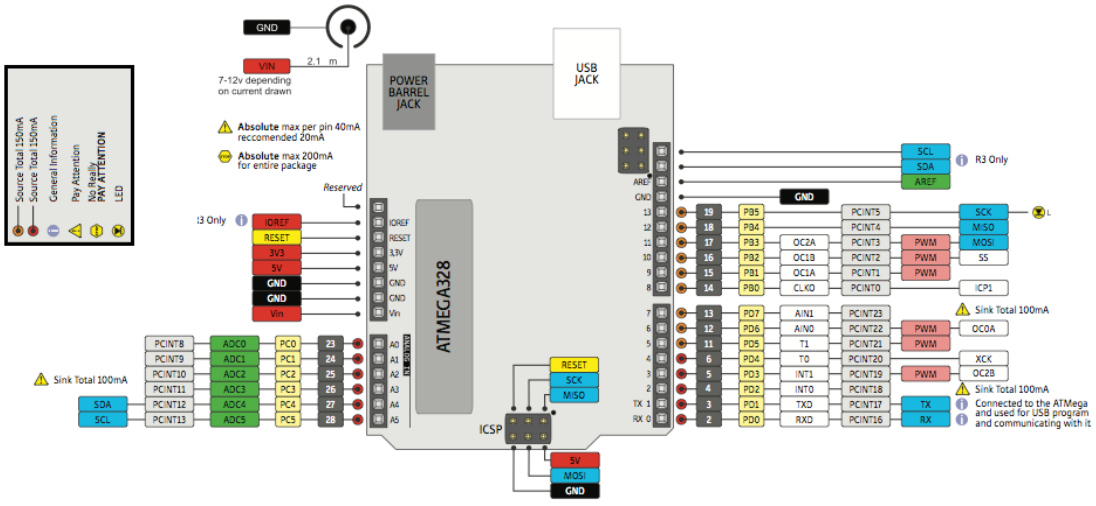


Figura 8 Pines de la placa Arduino Uno.

Fuente: (Joan, 2014)

Es importante que el lector conozca que se puede realizar la confección de la placa de forma manual con los componentes mostrados en la Figura 9. Realizar la tarjeta de forma manual, no fue considerada como una opción factible para la realización del proyecto por temas de tiempo y confiabilidad, ya que la tarjeta Arduino da cierta garantía de su funcionamiento por su reputación, además el costo no es muy extenso por lo que no fue necesario la confección manual y se optó por la adquisición de una tarjeta Arduino ya ensamblada.

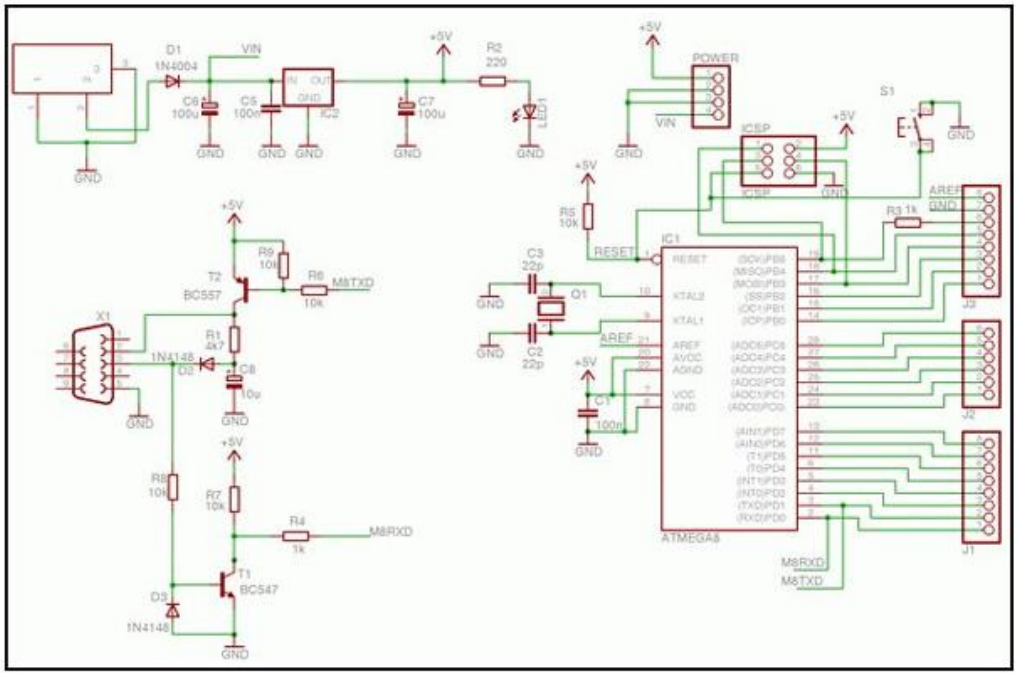


Figura 9 Esquema eléctrico de Arduino Uno.

Fuente: (Joan, 2014)

Por medio del software Arduino (IDE) es posible programar la tarjeta Arduino UNO, los pasos para su instalación se encuentran adjuntas en el Anexo 2, en la plataforma de programación para Arduino (Arduino IDE) en el menú de herramientas se puede elegir la distribución de Arduino de la placa que se está utilizando. (Arduino, 2016)

Una de las facilidades que brinda la tarjeta programable Arduino es la capacidad de conectar diferentes módulos, para distintos propósitos, por ejemplo, un módulo de bluetooth, un módulo de wifi, un módulo GPS, entre otros (Arduino, 2016). Para la realización del proyecto, se utilizó el módulo SIM 908C, el cual es un módulo

GPS/GPRS/GSM, utilizado para la obtención de ubicación y envío de datos vía GPRS, en el apartado 2.6, se realiza un análisis más completo de este módulo, el cual, es fundamental para la obtención de la ubicación y el envío de las coordenadas obtenidas al Servidor Web, para su procesamiento.

2.4. Sistema de Posicionamiento Global (GPS)

GPS es un sistema que permite determinar la posición de un objeto en la tierra, con una precisión de pocos metros de error. El sistema fue desarrollado, instalado y empleado por el Departamento de Defensa de los Estados Unidos, para determinar las posiciones en el globo, el sistema GPS está constituido por 24 satélites y utiliza la trilateración. Para poder utilizar este sistema, se requiere de un receptor el cual se encarga de localizar los satélites de la red, y así la ubicación, cuya precisión depende de su alcance y tecnología. (Vergara Merino, Hernández Correas, Virúes Ortega, Ramos Campo, & García Cabañas Bueno, 2016)

2.5. Servicio General de Paquetes Vía Radio (GPRS)

GPRS por sus siglas en inglés (General Packet Radio Service), es una extensión del “Sistema Global para Comunicaciones Móviles” o GSM, para la transmisión de datos mediante conmutación de paquetes. Una conexión GPRS está establecida por la referencia a su nombre de punto de acceso (APN). Con GPRS se pueden utilizar servicios como Wireless Application Protocol (WAP), servicio de mensajes cortos (SMS), Multimedia Messaging System (MMS), Internet y para los servicios de comunicación, como el correo electrónico y la World Wide Web (WWW).

2.6. Módulo SIM 908C

2.6.1. Características Generales

Diseñado por Global Market, el módulo SIM 908C está integrado con un módulo de alto rendimiento GSM/GPRS, y un mòdulo GPS. El sistema GSM/GPRS está

conformado por un módulo cuatribanda que funciona en las frecuencias de 850 Mhz GSM, 900Mhz EGSM, 1800Mhz DCS y 1900 Mhz PCS. El SIM 908, soporta diferentes esquemas de codificación GPRS, CS-1, CS-3 y CS-4. Sus dimensiones son de 30x30x3.2 mm, por lo que su tamaño lo permite cubrir la mayor parte de requerimientos con respecto a espacio en las aplicaciones, tal como M2M, teléfonos inteligentes, PDA y otros dispositivos móviles. En la Figura 10 se puede visualizar la vista frontal del módulo, el cual cuenta con 80 contactos y proporciona todas las interfaces de hardware entre el módulo y tableros de los clientes (SIMCom, 2011). Entre las principales ventajas con respecto al hardware están:

- El puerto serial y el puerto de depuración pueden ayudar para que el usuario desarrolle aplicaciones fácilmente.
- Puerto Serial GPS.
- Dos canales de audio, el cual están incluidas dos entradas y dos salidas, y son configuradas mediante comandos AT.
- Interfaz de carga.
- Programable para propósito general de entrada y salida.
- El teclado e interfaces de pantalla SPI brinda a los usuarios la flexibilidad para desarrollar aplicaciones personalizadas.

El módulo SIM908 está diseñado de tal forma que al entrar en modo de suspensión (sleep), el consumo de corriente es aproximadamente de 1,0 mA, en este modo el GPS se encuentra apagado, además integra el protocolo TCP/IP y comandos AT (Ver Anexo 3), los cuales son útiles para la transferencia y procesamiento de información. En la Tabla 5, se detalla un resumen de las características técnicas principales del módulo SIM 908. (SIMCom, 2011).

Figura 10 Vista frontal SIM908

Fuente: (SIMCom, 2011)

2.6.2. Hardware

En la Tabla 5 se especifica las características de hardware más importantes para el correcto funcionamiento del módulo, además de diferentes características importantes relacionadas con el hardware.

Tabla 5

Características de hardware Modulo SIM 908

Característica	Implementación
Rango de alimentación	3.2V ~ 4.8V
Ahorro de energía	Consumo de energía típico en modo de reposo es 1.0mA
Carga	Soporta control de carga para batería tipo Li-on
Bandas de frecuencia	Cuatribanda: GSM 850, EGSM 900, DCS 1800, PCS 1900. El módulo puede buscar la frecuencia de forma automática, también se puede establecer la frecuencia mediante el comando AT "AT+CBAND", para mayor detalles sobre comandos AT (Revisar Manual de Comandos AT)
Potencia de Transmisión	Clase 4 (2W) para las frecuencias: GSM 850 y 900 EGSM Clase 1 (1W) para las frecuencias 1800DCS y 1900 PCS
Conectividad GPRS	Multi-slot clase 10 (Por defeto) Multi-slot clase 8 (Opcional)
Rango de temperatura	Operación normal: -30°C ~+80°C Operación Restringida: -40°C ~ -30°C y +80 °C ~ +85°C Temperatura de almacenamiento: -45°C ~ +90 °C
GPRS	- Velocidad máxima de transferencia de datos descendente: 85,6 Kbps. - Velocidad máxima de transferencia de datos ascendente: 42.8 Kbps. - Esquema de codificación: CS-1, CS-2, CS-3 y CS-4. - Integración del protocolo TCP/IP.
CSD	Soporta transmisión CSD
USSD	Soporta Datos de Servicios no estructurados suplementarios.

Continúa

Característica	Implementación
SMS	Modos MT, MO, CB, Text y PDU. Almacenamiento de SMS en la tarjeta SIM.
Interfaz SIM	Soporta una tarjeta SIM de 1,8V y 3V.
Antena Externa	Contacto de antena
Características de Audio	Modos de códec de voz: - Media velocidad de transmisión. - Velocidad completa de transmisión. - Velocidad completa mejorada. - Multi-velocidad adaptativa. - Cancelación de eco. - Supresión de ruido.
Puerto Serial y Puerto de Depuración	**Puerto serial** - Interfaz de modem completa con estado y líneas de control desbalanceado y asíncrono. - Soporta velocidad de 1200 bps a 115200 bps - Puede ser usado por comandos AT y flujo de datos. - Apoyo RTS/CTS de intercomunicación de hardware y software ON/OFF para el control de flujo. **Puerto de Depuración** - Interfaz de modem nulo, GPS/DBG_TXD y GPS/DBG_RXD - Puede ser usado para depurar y actualizar el Firmware.
Características físicas	Tamaño: 30*30*3.2 mm Peso: 5.2 g.
Actualización de Firmware	Actualización de Firmware por medio del puerto de depuración.

Fuente: (SIMCom, 2011)

2.6.3. Modos de Operación del Módulo SIM 908C

El módulo SIM 908 tiene la capacidad de funcionar con diferentes modos de operación, en la Tabla 6 se detalla un resumen.

Tabla 6

Resumen de los modos de operación del módulo SIM 908

Modo		Función
Operación Normal	GSM/GPRS SLEEP	El módulo entra automáticamente en modo Sleep, si las condiciones de Sleep se encuentran activas y no existe ninguna interrupción de hardware impidiéndolo, por ejemplo GPIO o de datos en el puerto serie. En este modo el consumo de corriente se reduce al mínimo nivel.
	GSM IDLE	El software está activo, el módulo se encuentra registrado en la red GSM, y se encuentra listo para comunicarse.
	GSM TALK	La conexión y comunicación del módulo se encuentra establecida, en este caso el poder de consumo depende de la configuración de la red, tales como DTX encendido/apagado, FR/EFR/HR secuencias de salto de la antena.
	GPRS STANDBY	El módulo está listo para la transferencia de datos GPRS, pero no se envían datos, en este caso el consumo de energía depende de la configuración de la red y configuración GPRS.

Continúa

Modo		Función
	GPRS DATA	En este modo hay transferencia de datos GPRS (PPP, TCP o UDP), en este modo el consumo de energía está relacionado con la configuración de red (por ejemplo, control de nivel de potencia); velocidades de datos de enlace ascendente/descendente y la configuración GPRS.
Power Down	Mediante el comando AT "AT+CPOWD=1" (Véase SIM908 AT Command Manual_V1.01), o mediante el botón PWRKEY del módulo se puede enviar potencia normal al mismo. La unidad de administración de energía apaga el suplemento de energía para la banda base del módulo, y sólo la fuente de alimentación para el RTC se mantiene. El software no se encuentra activo.	
Modo de Funcionalidad Mínima.	El comando AT "AT + CFUN" (Véase SIM908 AT Command Manual_V1.01) se puede utilizar para fijar el módulo a un modo de funcionalidad mínima sin la eliminación de la fuente de alimentación. En este modo, la parte de radiofrecuencia del módulo no funciona o la tarjeta SIM no será accesible. El nivel de energía de este módulo es más bajo que en el modo normal.	

Continúa

Modo	Función
Modo sólo-carga	El módulo entrará en modo solo-carga de forma automática, cuando un cargador y la batería están conectados a un módulo SIM908 apagado. En este modo, el módulo no busca la red y limita el acceso a los comandos AT disponibles.
Modo de carga durante el funcionamiento normal	El módulo pasará automáticamente a este modo cuando un cargador está conectado a un módulo en modo de operación normal cuando el voltaje de la batería no es inferior a 3.2 V.

Fuente: (SIMCom, 2011)

2.6.4. Diagrama Funcional del Módulo SIM 908

Es importante conocer el funcionamiento interno del módulo, en la Figura 11 se puede observar los bloques de funcionamiento relacionado con el sistema, es importante conocer los componentes para realizar una distribución y conexión acertada con la tarjeta Arduino UNO en el montaje final.

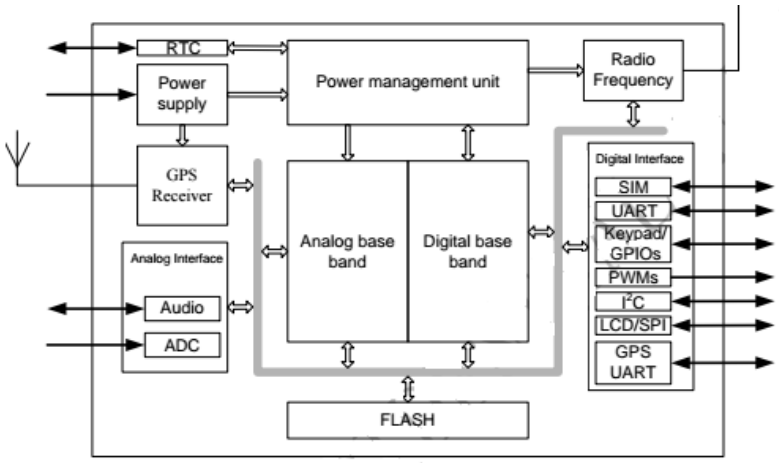


Figura 11 Diagrama funcional del SIM 908

Fuente: (SIMCom, 2011)

Los bloques representados en la Figura 11 se describen a continuación:

- El bloque de interfaces digitales, se encuentra integrado por interfaces de entrada/salida como: SIM, UART, Keypad/GPIOs, I^2C, LCD/SPI, GPS UART y una interfaz de salida PWMs.
- Dos bloques que corresponden a entradas/salidas de señal inalámbrica, el receptor GPS se encarga de obtener las coordenadas de posicionamiento, mientras que el bloque de Radio Frecuencia es el encargado de registrar el módulo a la red de telefonía celular, así como el envío de información hacia Internet mediante tecnología GPRS.
- Dos interfaces análogas, una de entrada/salida correspondiente a la interfaz de audio, y una entrada correspondiente al convertidor análogo/digital.
- El bloque RTC corresponde al reloj en tiempo real.
- Alimentación de poder, el cual se encarga de energizar al sistema.
- La unidad de procesamiento consta de una unidad de distribución de energía, una banda base análoga y otra digital, además de una memoria Flash.

2.6.5. Distribución de pines del módulo SIM 908

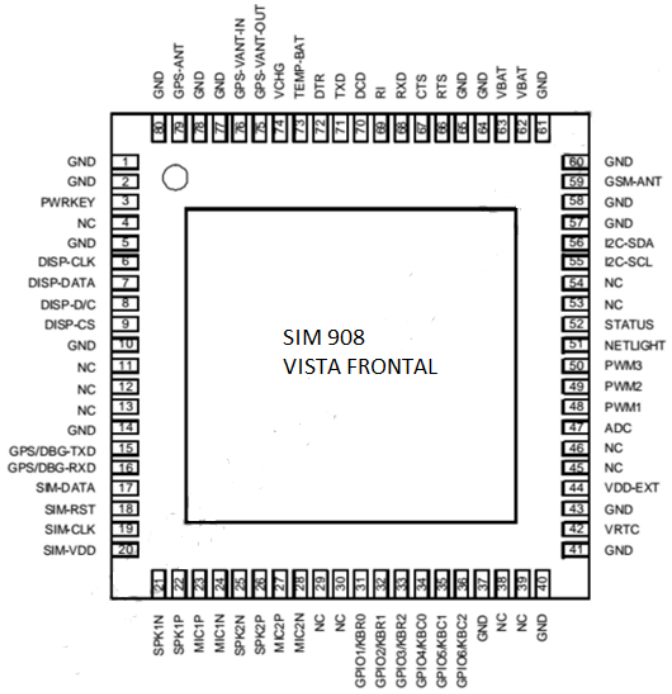


Figura 12 Diagrama de pines del módulo SIM 908 Vista Superior.

Fuente: (SIMCom, 2011)

El módulo es un encapsulado que consta con varios pines de entrada o salida En la Tabla 7 se detalla una descripción de cada pin correspondiente al módulo SIM 908, se puede verificar gráficamente la ubicación de los pines por medio de la revisión de la Figura 12.

Tabla 7

Descripción de pines del módulo SIM 908

Nombre de Pin	Número de Pin	Entrada/Salida	Descripción
Fuente de alimentación			
VBAT	62, 63	Entrada	Fuente de alimentación
VRTC	42	Entrada/Salida	Fuente de alimentación para RTC
VDD-EXT	44	Salida	Fuente de alimentación de salida de 2,8 V
GPS-VANT-OUT	75	Salida	2.8 V de salida para la antena GPS
GPS-VANT-IN	76	Entrada	Fuente de alimentación de la antena GPS
GND	1, 2, 5, 10, 14, 37, 40, 41, 43, 57, 58, 60, 61, 64, 65, 77, 78, 80		Tierra
Interfaz de carga			
VCHG	74	Entrada	Entrada de carga
TEMP_BAT	73	Entrada	Sensor de temperatura de batería
Encendido/Apagado			
PWRKEY	3	Entrada	PWRKEY se apaga por menos de un segundo y luego se libera para encender/apagar el módulo.
Interfaces de audio			
MIC1P	23	Entrada	Entrada diferencial de audio
MIC1N	24		
SPK1P	22	Salida	Salida diferencial de audio
SPK1N	21		
MIC2P	27	Entrada	Entrada diferencial de audio
MIC2N	28		

Continúa

Nombre de Pin	Número de Pin	Entrada/Salida	Descripción
SPK2N	25	Salida	Salida diferencial de audio
SPK2P	26		
Estado			
STATUS	52	Salida	Estado de encendido
NETLIGHT	51	Salida	Estado de red
Interface LCD			
DISP-CLK	6	Salida	Interfaz de pantalla
DISP-DATA	7	Entrada/Salida	
DISP -D/C	8	Salida	
DISP -CS	9	Salida	
Interface I^2C			
I2C-SDA	56	Salida	I^2C bus de datos serial
I2C-SCL	55	Entrada/Salida	I^2C bus de reloj serial
Interfaz de teclado / GPIOs			
GPIO1/KBR0	31	Entrada/Salida	GPIO1/ Teclado fila 0
GPIO2/KBR1	32		GPIO2/ Teclado fila 1
GPIO3/KBR2	33		GPIO3/ Teclado fila 2
GPIO4/KBC0	34		GPIO4/ Teclado columna 0
GPIO5/KBC1	35		GPIO5/ Teclado columna 1
GPIO6/KBC2	36		GPIO6/ Teclado columna 2
Puerto Serial			
RXD	68	Entrada	Recepción de datos.
TXD	71	Salida	Transmisión de datos.
RTS	66	Salida	Solicitud de envío.
CTS	67	Entrada	Borrado para envío.
DCD	70	Salida	Soporte de datos a detectar.
RI	69	Salida	Indicador de timbre.
DTR	72	Entrada	Terminal de datos listos.
Interfaz GPS/Depuración			
GPS/DBG-TXD	15	Salida	Para la salida de información del GPS NMEA, depuración y actualización de firmware.
GPS/DBG-RXD	16	Entrada	

Continúa

Nombre de Pin	Número de Pin	Entrada/Salida	Descripción
Interfaz SIM			
SIM-VDD	20	Salida	Alimentación de voltaje para la tarjeta SIM soporta 1.8 V o 3 V.
SIM-DATA	17	Entrada/Salida	Datos SIM Entrada/Salida
SIM-CLK	19	Salida	Reloj SIM
SIM-RST	18	Salida	Reinicio SIM
ADC			
ADC	47	Entrada	Entrada de voltaje rango 0V-2.8V
Pulso con modulación (PWM)			
PWM1	48	Salida	PWM
PWM2	49	Salida	PWM
PWM3	50	Salida	PWM
Interfaz GSM/GPS RF			
GSM-ANT	59	Entrada/Salida	Conexión de la antena de radio GSM
GPS-ANT	79	Entrada	Conexión de la antena de radio GPS
Sin conexión			
NC	4,11,12,13,29,30, 38,39,45,46,53,54	-	

Fuente: (SIMCom, 2011)

La tarjeta Arduino UNO tiene a su disposición varios módulos, los cuales son fabricados para la integración y control por medio de la tarjeta programable, el chip SIM908 dispone de un módulo (Véase Figura 13) para facilitar su uso con la tarjeta programable Arduino.

Figura 13 Módulo SIM908 para Arduino.

Fuente: (NIPLE SOFTWARE, 2016)

Para la adquisición de señales GPS, el módulo requiere la conexión de una antena. Según el datasheet del SIM908 se debe utilizar una antena de parche cerámica RHCP con un plano de tierra del tamaño adecuado y orientado hacia el cielo, las características de la antena GPS son:

- Conector RG174.
- Frecuencia central 1575.42 ±1Mhz
- Ancho de banda CF ± 5Mhz.
- Polarización RHCP
- Ganancia 5dBic
- Impedancia 50 Ω
- Radio Axial 3dB
- Dimensión 25*25*4 mm.
- Tensión de alimentación 2.2 ~ 5V DC
- Consumo 5 ~ 15 mA.
- Método de montaje Imán / Adhesivo Ambiental.
- Temperatura de funcionamiento -40 ℃ ~ + 85 ℃
- Humedad relativa hasta el 95%
- Vibración 10 a 55 Hz con 1,5 mm de amplitud (SIMCom, 2011).

Figura 14 Antena GPS para el módulo SIM 908.

Fuente: (NIPLE SOFTWARE, 2016)

Para la conexión de la antena GPS se requiere confeccionar un circuito de acoplamiento para evitar pérdidas. Cabe mencionar que el módulo SIM908 para Arduino (Véase Figura 13) tiene el circuito de acoplamiento integrado (SIMCom, 2011), lo que evita el armado y montado de dicho circuito, una de las razones por lo que se realizó la adquisición del módulo, además de la confianza y el ahorro del tiempo que genera.

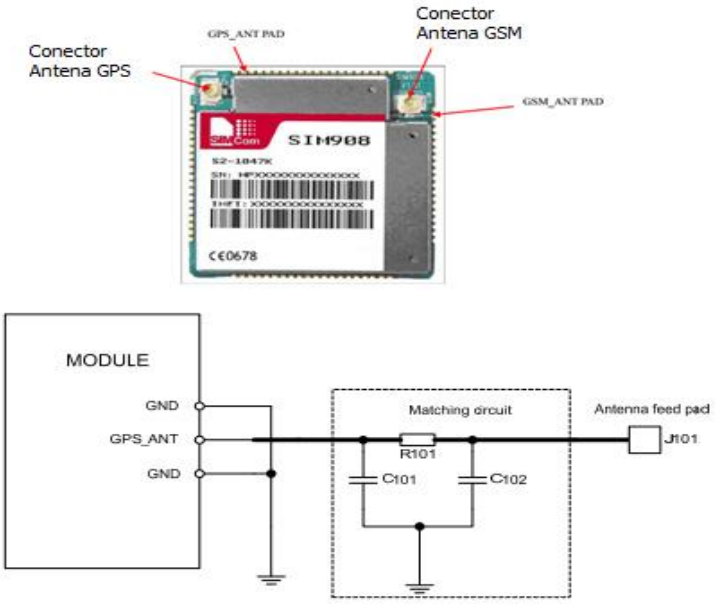


Figura 15 Modulo SIM 908 y circuito de acoplamiento para antena GPS.

Fuente: (SIMCom, 2011)

Para la adquisición de la antena GPS el usuario debe tener en cuenta los siguientes factores

- Elegir una antena lineal con un patrón de ganancia semiesférica razonablemente uniforme de > -4dBi.

- El uso de una antena con una ganancia inferior a continuación, esto le dará resultados menos que deseables. Tenga en cuenta que una antena RHCP con una ganancia de 3 dBi, equivale a una antena de polarización lineal de 0 dBi.
- El correcto dimensionamiento plano de tierra es una consideración crítica para las antenas GPS pequeñas.
- La correcta colocación de la antena GPS debe ser siempre la primera consideración en la integración de la SIM18. (SIMCom, 2011)

Además de la antena GPS, el módulo SIM908 tiene incluido un conector RG174, en dónde es posible montar una antena GSM, para el registro del chip en la red celular y la posterior transmisión de información mediante tecnología GPRS (Véase Figura 16), por lo general la antena viene incluida al momento de la adquisición del módulo (SIMCom, 2011).

Figura 16 Antena GSM.

Fuente: (NIPLE SOFTWARE, 2016)

La interfaz de RF del módulo (Ver Figura 11) tiene una impedancia de 50Ω para adaptarse al diseño físico de las aplicaciones individuales. Para reducir al mínimo la pérdida en el cable de RF, se tiene que ser muy cuidadoso en el momento de elegir el cable de RF. SIMCom, la empresa desarrolladora del módulo recomienda los siguientes valores de pérdida de inserción:

- GSM900 <1dB
- DCS1800 <1,5 dB

Para facilitar la sintonización de la antena y la prueba de certificación, un conector de RF y un circuito de adaptación de la antena deben ser añadidos (SIMCom, 2011). La Figura 17 muestra el circuito de adaptación añadido, cabe resaltar que el módulo SIM908 para Arduino tiene el circuito de acoplamiento integrado, lo que evita el armado y montado de dicho circuito, una de las razones por lo que se realizó la adquisición del módulo, además de la confianza y el ahorro del tiempo que genera.

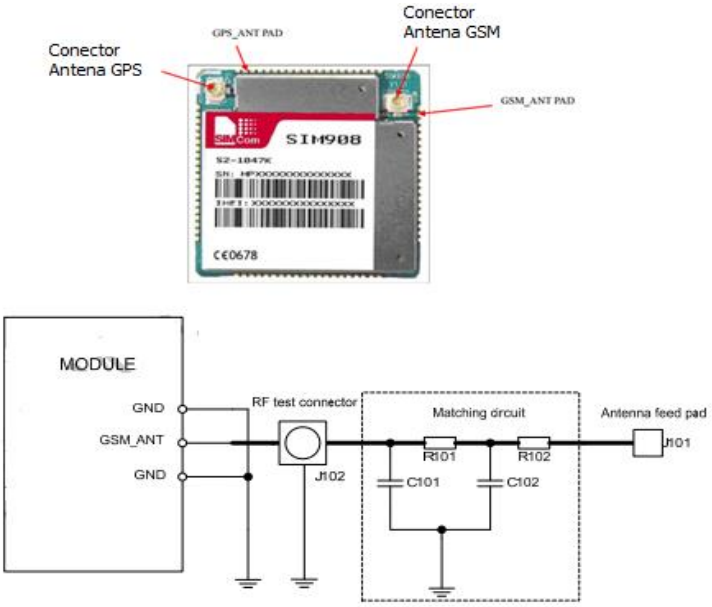


Figura 17 Modulo SIM 908 y circuito de acoplamiento para antena GSM.

Fuente: (SIMCom, 2011)

2.7. Módulo Transductor Acelerómetro MMA7361

Los acelerómetros son dispositivos que miden la tasa de aceleración de un objeto, la unidad de medida de la aceleración es m/s^2 o en las fuerzas G (g), en el planeta Tierra, la sola fuerza de gravedad es equivalente a 9,8 m/s^2. Los acelerómetros son útiles para detectar las vibraciones en los sistemas y para las aplicaciones de orientación. El acelerómetro MMA7361, es el utilizado en la realización del proyecto, por lo que se va a realizar un análisis de este tipo de transductor. En la Figura 18 se observa la distribución del encapsulado en donde se dispone el acelerómetro. (Guarnizo)

Figura 18 Módulo Acelerómetro MMA7361.

Fuente: (Guarnizo)

2.7.1. Funcionamiento

El acelerómetro MMA7361 tiene la capacidad de medir la aceleración en 3 dimensiones, conocidos como ejes, eje X, eje Y y eje Z. Todo cuerpo posee un centro de masa en donde se equilibran todas sus fuerzas. Si se aplica una fuerza en un centro de masa el cuerpo no va a rotar, pero si se aplica desplazando de su centro experimenta un momento de fuerza que lo hará rotar. (Circelli, 2015)

El sensor MMA7361 es de baja energía, con acondicionamiento de señal, filtro de paso bajo de 1 polo, compensación de temperatura, prueba automática (Self test). Detección de gravedad 0- caída libre (0g-Detect), y g-Select que permite la selección de dos sensibilidades. El desplazamiento de gravedad cero y la sensibilidad se ajustan de fábrica y no requieren dispositivos externos, además posee un modo de reposo (sleep). En la Figura 19 se muestra un diagrama del funcionamiento interno del circuito, y de los pines que posee. (Guarnizo)

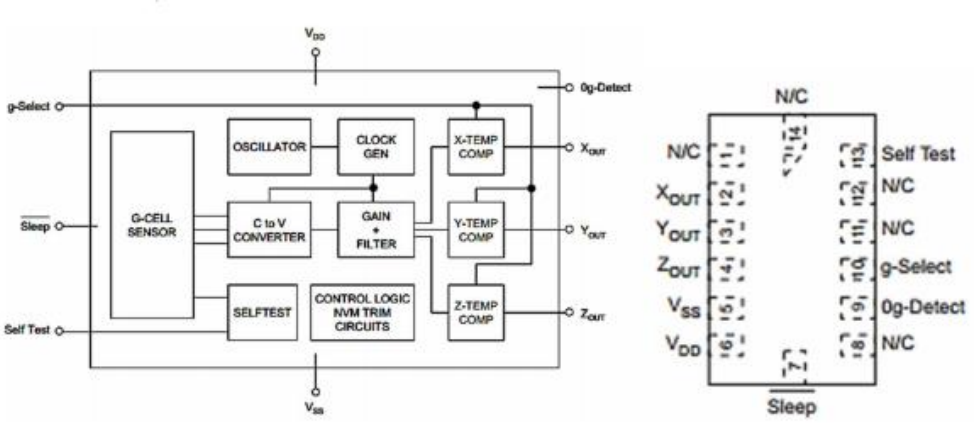


Figura 19 Diagrama Representativo del Funcionamiento del módulo acelerómetro MMA 7361.

Fuente: (Guarnizo)

En la Figura 20 y Figura 21, se representan las aceleraciones dinámicas y estáticas correspondientes al transductor.

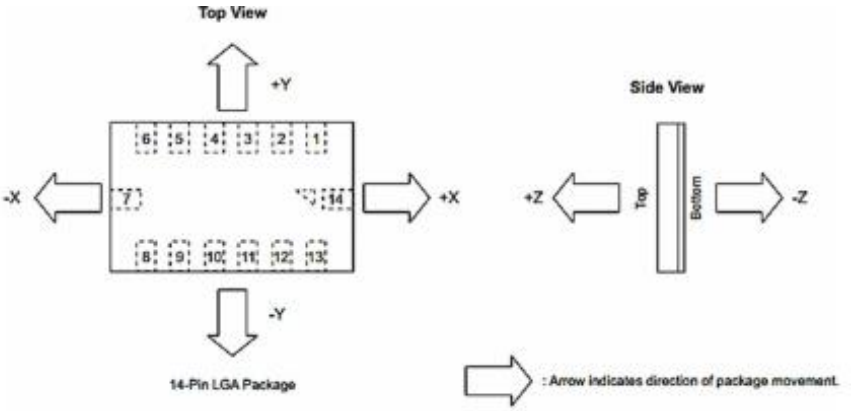


Figura 20 Diagrama Representativo de la aceleración dinámica.

Fuente: (Guarnizo)

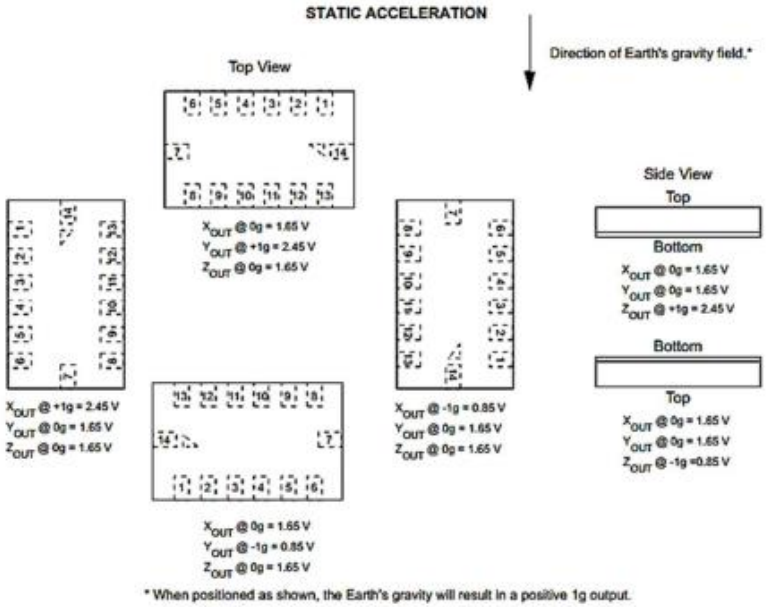


Figura 21 Diagrama Representativo de la aceleración estática.

Fuente**:** (Guarnizo)

2.7.2. Descripción

Se realiza un detalle de los pines disponibles en el módulo:

- Entrada de voltaje de 5V, 3.3V y GND, en la Figura 22 se observa la distribución de pines correspondiente a voltaje y tierra del módulo que contiene el sensor acelerómetro, la tarjeta cuenta con dos diferentes valores de entrada de voltaje, y con un pin a tierra. El sensor acelerómetro trabaja a 3.3V. lo que

lo hace muy económico en el consumo. Si se alimenta con 5V, se activa un regulador de 3.3V para energizar el sensor. (Guarnizo).

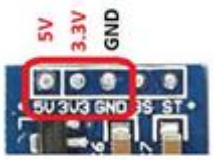


Figura 22 Pines entrada de Voltaje y tierra.

Fuente: (Guarnizo)

- Regulador a 3.3V, en la Figura 23 se muestra la distribución del regulador de voltaje dentro de la tarjeta, el cual permite tomar el voltaje de entrada (entre 5V – 7V), y arrojar 3.3V al pin 6 (Ver Figura 19) en dónde se ingresa el voltaje de funcionamiento para el sensor. (Guarnizo)

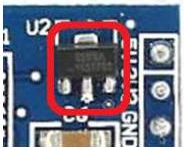


Figura 23 Regulador de 5V.

Fuente: (Guarnizo)

- Pines analógicos de salida, en la Figura 24 se observa la distribución de los tres pines analógicos con que cuenta la tarjeta, los cuales, expresan las coordenadas X, Y, y Z, y representan un voltaje en movimiento del sensor en el eje correspondiente. (Guarnizo)

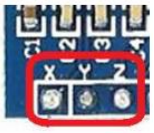


Figura 24 Pines análogos de salida.

Fuente: (Guarnizo)

- Pines de control y salida digital, en la Figura 25 se observa la distribución de los pines de control (ST, GS y SL), y el pin de salida digital (0G). A continuación se realiza un pequeño análisis los pines mencionados. (Guarnizo)

Figura 25 Pines de control y salida digital.

Fuente: (Guarnizo)

ST: Self-Test.- El sensor proporciona una función de auto verificación que permite verificar la integridad mecánica y eléctrica del acelerómetro (Ver Figura 25). (Guarnizo).

0G: 0g-Detect.- El sensor ofrece una función que proporciona una señal lógica alta cuando los tres ejes están en 0g, es decir detecta cuando el sensor está en caída libre (Ver Figura 25). (Guarnizo).

GS: g-Select.- Esta característica permite la selección entre dos sensibilidades, ya sea de 1.5g o 6g (Ver Tabla 8). (Guarnizo)

Tabla 8

Ajuste de Sensibilidad Acelerómetro MMA7361

g-Select	g-Range	Sensibilidad
0	1.5g	800mV/g
1	6g	206mV/g

Fuente: (Guarnizo)

SL: Sleep Mode.- El sensor acelerómetro proporciona un modo de reposo, lo cual proporciona una reducción significativa de la corriente de funcionamiento cuando el modo Sleep se encuentra activado. Una señal de entrada a bajo nivel

en el pin 7 (Modo Sleep), Ver Figura 19, si se coloca el dispositivo en este modo se reduce la corriente a 3uA. (Guarnizo)

2.8. Sensor de Temperatura Infrarrojo MLX90614

El sensor MLX90614 desarrollado por Melexis, es un sensor infrarrojo para medir temperaturas sin necesidad de contacto con el objeto a medir. Este sensor cuenta con un amplificador de bajo ruido, un convertidor analógico digital de alta resolución de 17 bits, una unidad de procesamiento digital de la señal. El sensor viene calibrado con una salida PWM y SMBus. El usuario puede configurar la salida digital para modulación por ancho de pulso (PWM). Como estándar, el PWM 10 bits está configurado para transmitir de forma continua la temperatura medida en intervalo de -20 a 120 ° C, con una resolución de salida de 0,14 ° C. (Melexis, s.f.)

2.8.1. Características

El sensor de temperatura consta de las siguientes características:

- Bajo en costo y pequeño en tamaño. (Melexis, s.f.)
- Fácil de integrar. (Melexis, s.f.)
- Calibrado de fábrica en el rango de temperaturas: -40 a 125 ° C para la temperatura del sensor y -70 380 ° C para la temperatura del objeto. (Melexis, s.f.)
- Alta precisión de 0,5 ° C por encima del rango de temperaturas (0 + 50 C tanto para Ta y A). (Melexis, s.f.)
- Precisión médica de 0,1 ° C en un rango de temperatura limitado disponible bajo petición. (Melexis, s.f.)
- Resolución de la medición de 0,02 ° C. (Melexis, s.f.)
- Interfaz digital SMBus compatible para las lecturas de temperatura rápidos y redes de sensores edificio. (Melexis, s.f.)
- Salida PWM, adaptable para lectura continua. (Melexis, s.f.)
- Modo de ahorro de energía. (Melexis, s.f.)

2.8.2. Funcionamiento interno del sensor MLX90614.

El funcionamiento del sensor MLX90614 es controlado por una máquina de estado interna (Ver Figura 26), que controla las mediciones, los cálculos del objeto y la temperatura ambiente, hace el post-tratamiento de las temperaturas a la salida a través de una salida de PWM o la interfaz compatible SMBus. (Microelectronic Integrated Systems , 2015)

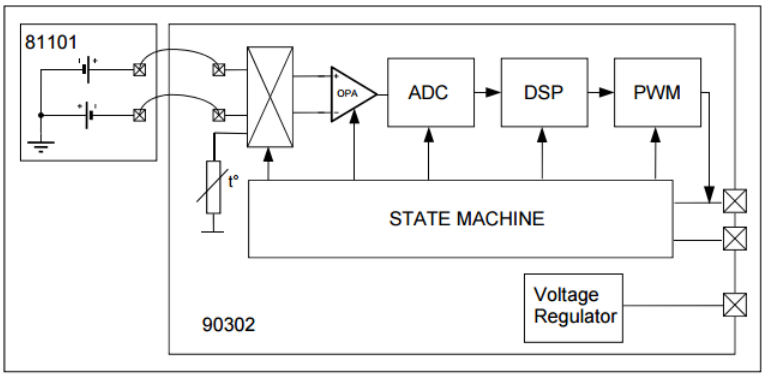


Figura 26 Diagrama de Bloques del funcionamiento del sensor MLX90614.

Fuente: (Microelectronic Integrated Systems , 2015)

El ASSP es compatible con 2 sensores IR. La salida de los sensores de infrarrojo es amplificada por un filtro de bajo nivel de ruido y bajo desplazamiento por medio de un amplificador de pulsos con ganancia programable, convertido por un modulador Delta Sigma a un solo flujo de bits y se alimenta a un potente DSP para procesamiento adicional (Ver Figura 26). La señal es tratada (por medio de la EEPROM contender) filtros de paso bajo FIR e IIR para reducir el ancho de banda de la señal de entrada y conseguir el rendimiento deseado de ruido y frecuencia de actualización. La salida del filtro IIR es el resultado de la medición y está disponible en la RAM interna. (Microelectronic Integrated Systems , 2015)

Basándose en los resultados de las mediciones anteriores, la correspondiente temperatura ambiente Ta y temperatura objeto A se calculan. Ambas temperaturas calculadas tienen una resolución de 0.01C. Los datos de Ta y A pueden leerse de dos maneras: las células de memoria RAM de lectura dedicados a este fin a través de la interfaz de 2 hilos (0,02 ° C Resolución, intervalos fijos), o a través de la salida

digital PWM (resolución de 10 bits, rango configurable). (Microelectronic Integrated Systems , 2015)

En el último paso del ciclo de medición, la medida de asistencia técnica se reajustará a la resolución de salida deseada del PWM y los datos recalculados se cargan en los registros de la máquina de estado del PWM, lo que crea una frecuencia constante con un ciclo de trabajo que representa los datos medidos. (Microelectronic Integrated Systems , 2015)

2.8.3. Descripción de pines.

La Figura 27 muestra la vista inferior del sensor MLX90614.

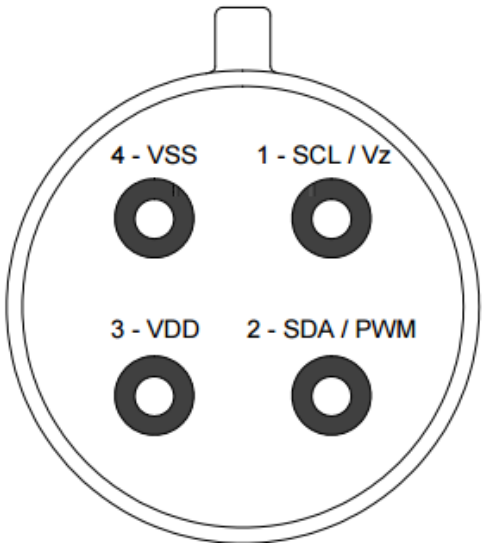


Figura 27 Vista Inferior MLX90614.

Fuente: (Microelectronic Integrated Systems , 2015)

La descripción de los pines del sensor se muestra en la Tabla 9.

Tabla 9

Descripción de pines MLX90614

Pin	Función
SCL/Vz	Entrada de reloj de serie para el protocolo de comunicaciones de 2 hilos. Está compuesto por un Zener de 5.7V y está disponible en este pin para la conexión del transistor bipolar externo a MLX90614Axx y suministrar el dispositivo desde la fuente de 8 ... 16V externa.
SDA/PWM	Entrada / salida digital. En el modo normal de la temperatura del objeto medido está disponible en esta entrada de ancho de pulso modulado. En el modo de compatibilidad SMBus el pin se configura automáticamente como drenaje abierto NMOS.
Vdd	Tensión de alimentación externa.
Vss	Tierra.

Fuente: (Microelectronic Integrated Systems , 2015)

2.8.4. Especificaciones eléctricas.

La Tabla 10 muestra las especificaciones eléctricas del sensor MLX90614.

Tabla 10

Especificaciones Eléctricas MLX90614

Parámetro	Símbolo	Min	Type	Max	Unidades
Voltaje Externo	VDD	4.5	5	5.5	V
Corriente de suministro	IDD		1.3	2	mA

Para mayor detalle sobre las especificaciones del sensor consultar el documento anexo “MLX90614-Datasheet-Melexis”. (Microelectronic Integrated Systems , 2015)

2.9. Arquitectura Cliente-Servidor

Diversas aplicaciones en la actualidad se ejecutan en un entorno Cliente-Servidor, los equipos clientes siempre forman parte de una red, y contactan a un servidor, el cual tiene características especiales y potentes con respecto a procesamiento de información, y que proporcionan servicios a los equipos clientes, por ejemplo servicios web, servicio de correo, servicio FTP, etc. (CCM, 2016)

Los servicios son utilizados por programas denominados programas clientes que se ejecutan en equipos clientes. Por eso se utiliza el término "cliente" (cliente FTP, cliente de correo electrónico, etc.) cuando un programa que se ha diseñado para ejecutarse en un equipo cliente, capaz de procesar los datos recibidos de un servidor. (CCM, 2016).

La arquitectura Cliente-Servidor, se caracteriza por dividir el trabajo en dos partes; el servidor que centraliza el servicio, y el cliente, que controla la interacción con el usuario. Esta arquitectura proporciona al usuario final el acceso transparente a las aplicaciones, datos, servicios de cómputo o cualquier otro recurso. El modelo soporta un medio ambiente distribuido en el cual los requerimientos de servicio hechos por estaciones de trabajo inteligentes o "clientes", resultan en un trabajo realizado por otros computadores llamados “servidores". (UCOL, 2016)

Todos los servidores disponibles en la actualidad trabajan mediante la arquitectura cliente-servidor, pero esto no significa que todos los servidores son usados para un sólo propósito, la principal característica de un servidor es la de brindar servicios a los clientes que se encuentran conectados en la misma red, los servicios varían según la necesidad que el servidor se encuentre dando solución. Por ejemplo, un servidor de base de datos, es un servicio que se encarga de almacenar y organizar información enviada por otro servidor, como puede ser un servidor web, mediante un lenguaje de programación para el envío automático o de forma manual de la información a almacenar por la base de datos.

2.9.1. Servidor Web

El servidor Web se encarga de procesar una aplicación del lado del servidor, recibiendo diversas peticiones de conexión por parte de los clientes, y generando una respuesta en cualquier lenguaje o para cualquier aplicación cliente, por lo general el cliente compila y ejecuta el servicio a partir de un navegador web. Por ejemplo, en el momento que un usuario (cliente) escribe una URL en el navegador, a continuación y sin que el usuario se dé cuenta, el navegador solicita dicha página al servidor que alberga el sitio solicitado, a continuación el servidor envía los datos a través de Internet, finalmente el navegador interpreta la información y muestra el resultado en el navegador. (Krall & Sierra, 2016)

Existen diversos lenguajes de programación que se ejecutan en el lado del servidor para procesar información o responder un requerimiento y en el lado del cliente para mostrar una información requerida. Por ejemplo al ingresar a una página Web, el servidor que procesa el requerimiento de visualización y manipulación de dicha página es un servidor Web, por lo general con el lenguaje de programación HTML, o PHP si se requiere una tarea especializada como puede ser una conexión con la base de datos y almacenamiento de información. Para la transmisión de datos se suele usar el protocolo HTTP, perteneciente a la capa de aplicación del modelo OSI TCP/IP. (Krall & Sierra, 2016)

Existen diversas formas para la instalación y uso de un servidor web, se necesita un equipo con requerimientos de hardware especiales y con mayor potencia para su instalación, entre las alternativas de instalación y configuración de un servidor web local, podemos nombrar XAMPP, LAMP o WAMP entre los más conocidos. Además de la instalación de un servidor web local se puede adquirir hosting web gratuitos o de costos muy bajos para almacenar nuestra página web, en los cuales únicamente se debe mover los archivos con extensión html, php, js, etc. (Krall & Sierra, 2016)

CAPÍTULO III
3. IMPLEMENTACIÓN DEL PROYECTO

Para que el proyecto funcione de la forma deseada, es necesario realizar el diseño de hardware y software del sistema. Por lo que se adquirió las herramientas que el proyecto necesita para su funcionamiento. A continuación se especifica los requerimientos que debe cumplir de forma general el sistema, a partir de la perspectiva del usuario final:

- Ver ubicación de la mascota a través de un dispositivo móvil o vía web.
- Ver si el animal se encuentra con signos vitales (Movimiento y temperatura) a través de un dispositivo móvil o vía web.

Una de las partes primordiales del proyecto es el diseño y la implementación de hardware y software, el arnés ubicado en el animal va a enviar la ubicación y signos vitales, para su posterior procesamiento. La información transmitida por el sistema de hardware (Arnés ubicado en el pecho del animal) es procesada y almacenada por el sistema de software (Servidor web y base de datos), para su posterior presentación en un dispositivo móvil o a una interfaz web.

3.1. Diseño de Hardware

El proyecto requiere de la implementación de varios instrumentos a nivel de hardware, los cuales, cumplen un propósito y se encargan de realizar una tarea específica para el funcionamiento general del sistema. En la Figura 28 se representa el diagrama general del funcionamiento del proyecto a nivel de hardware.

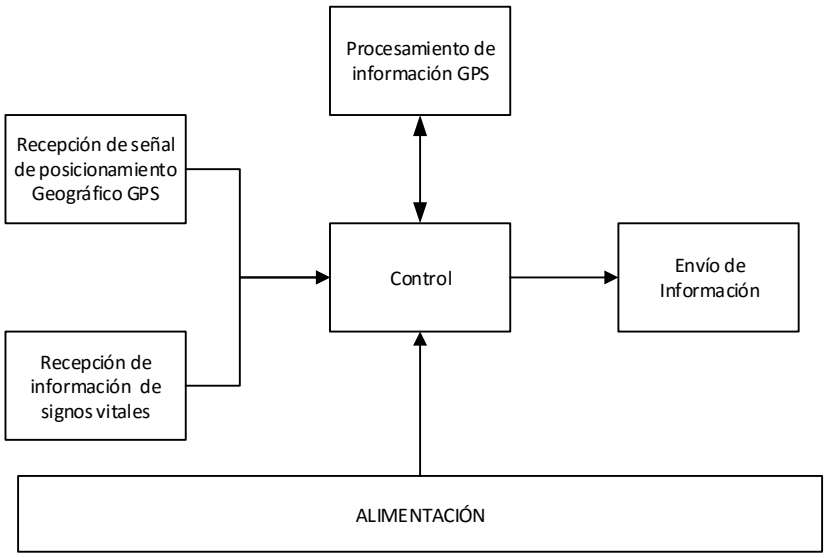


Figura 28 Diagrama Representativo de Funcionamiento del Sistema a nivel de hardware.

En el diagrama general de funcionamiento del sistema a nivel de hardware (Ver Figura 28), se representa las funciones que realiza el arnés de mascotas ubicado en el animal. Las funciones se representan mediante bloques, la interacción y orden de ejecución de los componentes se encuentra señalizado. A continuación se realiza un análisis de los bloques representados en la Figura 28:

- Recepción de señal de posicionamiento geográfico GPS, representa el ingreso de señales correspondientes a la ubicación geográfica GPS obtenidos por la antena GPS.
- Recepción de información de signos vitales, representa el ingreso de información correspondiente a los signos vitales de la mascota, obtenido por el sensor de movimiento y de temperatura. El acelerómetro MMA7361 y el sensor MLX90614 fueron utilizados para este propósito.
- Control, representa la base funcional del sistema, realiza el control de los módulos y sistemas complementarios, además de tareas relacionadas con el procesamiento de información.
- Procesamiento de información, representa una parte de las tareas realizadas por la unidad de control, realiza las tareas de procesamiento aritmético para el establecimiento de información al formato requerido para su implementación, el acoplamiento de una señal para el envío de datos reales para su

manipulación, además del análisis y la toma de decisiones de la información receptada.

- Transmisión de datos, representa el envío de información de los datos obtenidos y procesados hacia el servidor web.
- Alimentación, representa el bloque de alimentación, encargado de energizar el dispositivo y sus componentes.

3.1.1. Recepción de señal de posicionamiento geográfico GPS

La recepción de la señal GPS se establece por medio de la antena GPS conectada al módulo SIM 908. El control del módulo se lo realiza por medio de la tarjeta programable Arduino UNO, a partir de comandos AT (Ver Anexo 3). La Figura 29 muestra el diagrama representativo de los componentes de hardware involucrados en la obtención de señal GPS.

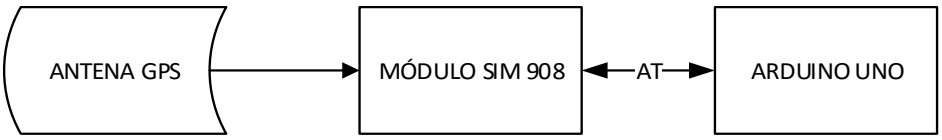


Figura 29 Representación de hardware utilizado en la adquisición de señal GPS

Se recibe la información de posicionamiento geográfico, latitud y longitud, por medio de la antena GPS, dicha antena se conecta al módulo SIM 908, el cual es controlado por medio de comandos AT mediante la tarjeta programable Arduino UNO.

La antena GPS recepta la información de posicionamiento global recibida por medio de triangulación satelital, dicha información es procesada a una trama de 77 bytes. La trama está conformada por varios valores de posicionamiento enviado por el satélite. La Figura 30 muestra una representación de la trama recibida por la antena GPS.

Figura 30 Representación de Trama GPS.

La trama es procesada a formato de carácter, la información se encuentra dividida por ",". Los valores importantes para la implementación del proyecto son la latitud y longitud. Los cuales son recibidos en formato NMEA por lo que es necesaria su conversión a grados decimales para su interpretación en el mapa. La Figura 31 muestra un ejemplo de una trama recibida por el módulo SIM 908 a través de la antena GPS, con la respectiva información a la que representa.

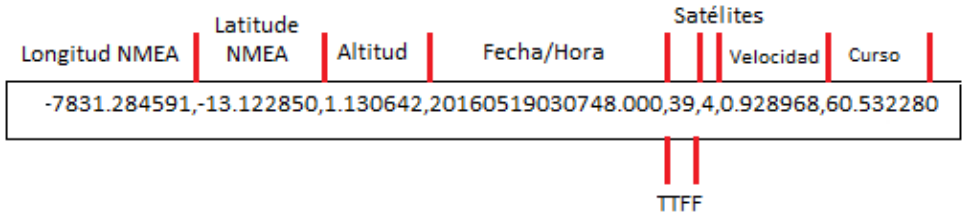


Figura 31 Ejemplo de Trama GPS.

La información de la trama se encuentra separada por "," (Ver Figura 31). Para conocer lo que representa cada valor se realiza una breve descripción de los valores que conforman la trama.

- Longitud NMEA: Corresponde el valor de longitud geográfica recibido por la antena GPS en formato NMEA.
- Latitud NMEA: Representa el valor de latitud geográfica recibido por la antena GPS en formato NMEA.
- Altitud: Representa el valor de la altitud geográfica medida a nivel del mar.
- Fecha/Hora: Corresponde al año, mes, día, hora, minuto, segundos y milisegundos (Formato UTC).
- TTFF: Tiempo que demoró en establecerse la primera construcción GPS.

- Satélites: Representa el número de satélites que la antena identificó para interpretar la señal, mientras mayor es el número, los valores de posicionamiento son más exactos.
- Velocidad OTG: Representa la velocidad de la antena en tierra.
- Curso: Representa la orientación en que se encuentra la antena en tierra.

Las tareas que realiza la unidad de control (Tarjeta programable Arduino UNO), para la recepción de señales GPS, se representa por medio del diagrama de flujo de la Figura 32.

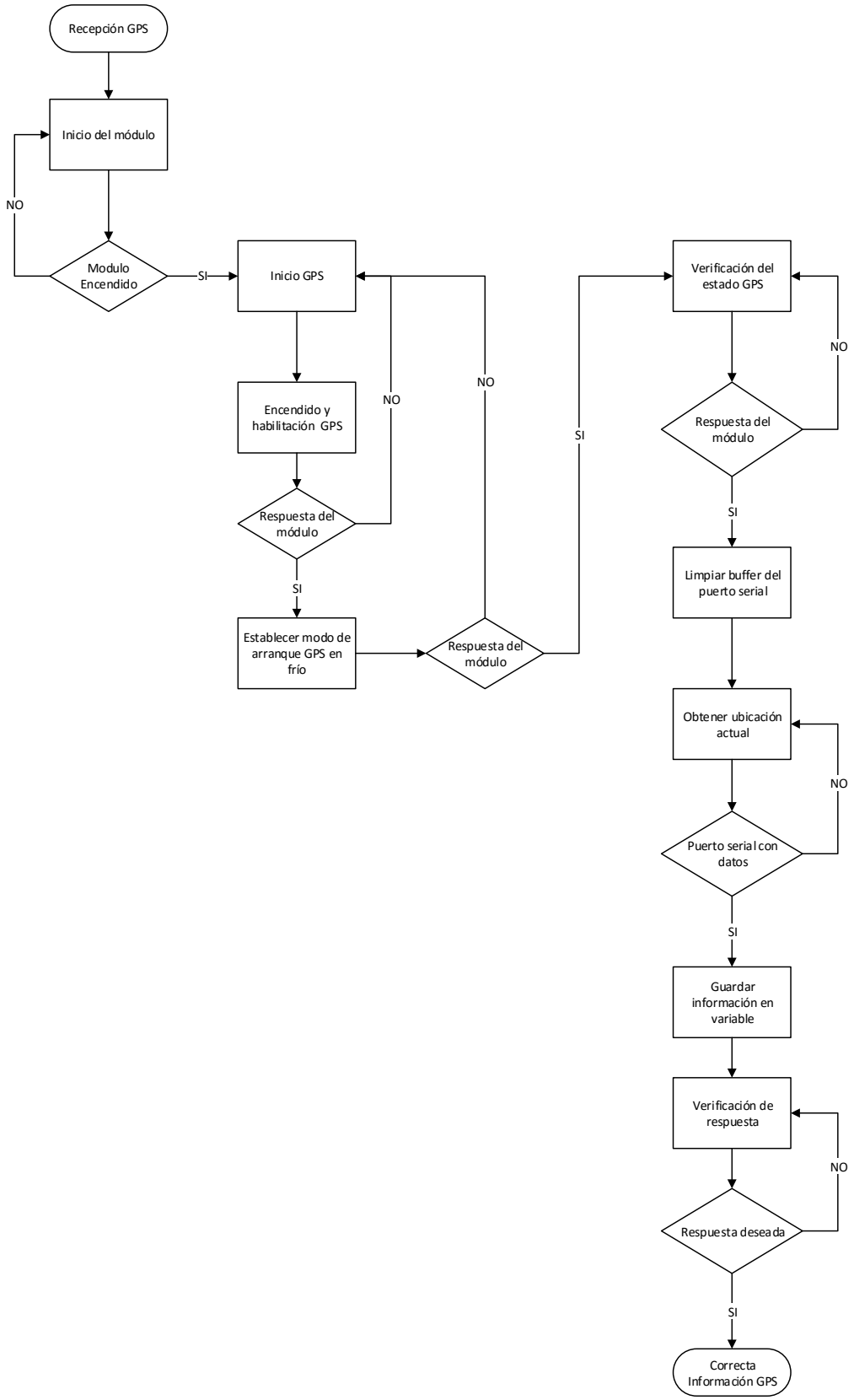


Figura 32 Diagrama de flujo Recepción GPS.

La Figura 32 muestra la representación por medio de un diagrama de flujo, de las tareas que realiza la unidad de control (Tarjeta programable Arduino UNO), la cual controla el módulo SIM 908 por medio de comandos AT para obtener la información de posicionamiento global GPS, cada vez que un comando AT es enviado, la tarjeta requiere de una respuesta por parte del módulo (Ver Figura 32). Al terminar el proceso, la correcta información GPS, representa una trama similar a la de la Figura 31, dicha información es almacenada en la tarjeta Arduino UNO, para su posterior procesamiento y envío.

3.1.2. Recepción de información de signos vitales

3.1.2.1. Movimiento

El movimiento de la mascota, es una de las medidas de signos vitales usada para el proyecto, se la obtiene mediante el módulo acelerómetro MMA7361, está detección de movimiento es la que informa al dueño si la mascota se encuentra con signos vitales. El acelerómetro MMA7361 es un transductor acelerómetro que mide el movimiento y las vibraciones. El sensor interactúa con la unidad de control (tarjeta programable Arduino UNO), la cual por medio de comandos nativos del lenguaje de programación del software Arduino UNO IDE (Ver Anexo 2), realiza la calibración e inicio del sensor para el posterior procesamiento y envío de información. La Figura 33 muestra el diagrama representativo de los componentes de hardware involucrados en la obtención de información de signos vitales.

Figura 33 Representación de hardware involucrado en la adquisición de información de movimiento.

Los valores detectados por el acelerómetro MMA 7361 corresponden a un valor de variación de tensión proporcional al valor de la aceleración en cada uno de sus

ejes X, Y o Z. Los pines X, Y y Z, representan las vibraciones y giros por cada eje. Mediante sentencias y librerías (Ver Anexo 2), se acopla la señal que ingresa, a un valor medible para la implementación y necesidades que requiere el proyecto. Por medio del diagrama de flujo de la Figura 34, se representan las tareas que realiza el sistema de detección de signos vitales mediante el acelerómetro MMA7361.

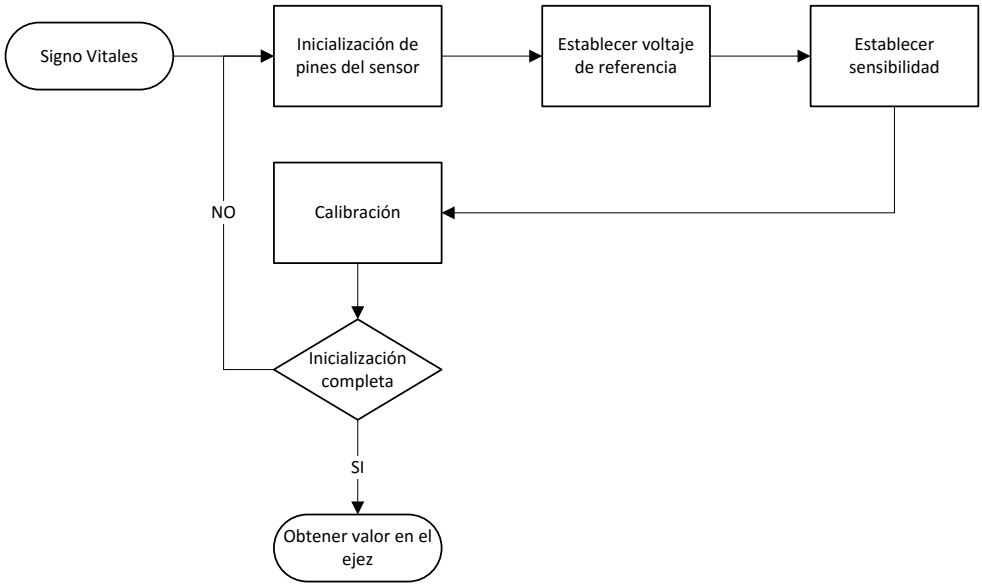


Figura 34 Diagrama de flujo Recepción Movimientos

En la Figura 34 se muestra el diagrama de flujo de las tareas que realiza la unidad de control (Tarjeta programable Arduino UNO) para la inicialización del sensor acelerómetro MMA7361. Únicamente se obtiene el valor de tensión en el eje Z, porque se requiere saber el movimiento de la mascota, y sólo se requiere conocer las vibraciones en un eje, por lo que se realizó la implementación con ese PIN únicamente.

3.1.2.2. Temperatura.

La temperatura de la mascota se la obtiene mediante el sensor de temperatura MLX90614, la temperatura externa del animal es una de las medidas usadas para informar al dueño si la mascota se encuentra con signos vitales. El sensor de

temperatura MLX90614 utiliza tecnología infrarroja, por lo que no se requiere el contacto directo del animal con el sensor. El sensor interactúa con la unidad de control (tarjeta programable Arduino UNO), la cual por medio de comandos nativos del lenguaje de programación del software Arduino UNO IDE (Ver Anexo 2), realiza configuración e inicio del sensor para el posterior procesamiento y envío de información. La Figura 35 muestra el diagrama representativo de los componentes de hardware involucrados en la obtención de información de temperatura.

Figura 35 Representación de hardware utilizado para la adquisición de información de temperatura.

El MLX90614 provee dos métodos de salida: PWM y SMBus (TWI, I^2C). la salida PWM de 10 bits provee una resolución de 0.14°C, mientras que la interfaz TWI tiene una resolución de 0.02°C. El MLX90614 viene calibrado de fábrica en un rango amplio de temperaturas: de -40 a 85°C para la temperatura ambiente y de -70 a 382.2°C para la temperatura del objeto. El valor medido es la temperatura promedio de todos los objetos en el Campo de Vista del sensor. El MLX90614 ofrece una precisión estándar de 0.5°C en temperaturas de habitación. Es importante adquirir la librería de Arduino para salida I^2C, por medio del diagrama de flujo de la Figura 36 se muestra los procesos que se debe realizar para el funcionamiento del sensor con la tarjeta programable Arduino UNO.

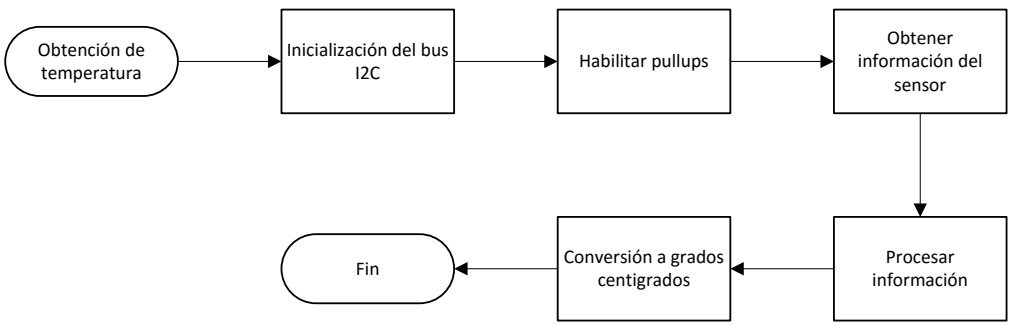


Figura 36 Diagrama de flujo recepción temperatura

3.1.3. Unidad de Control

La Unidad de Control realiza la administración del sistema, y realiza las siguientes tareas:

- Encendido y control del módulo SIM 908 por medio de comandos AT, para la recepción de información de posicionamiento geográfico, latitud y longitud.
- Adquisición de los datos correspondientes al movimiento de la mascota. Inicialización, calibración y adquisición de información obtenida por el sensor acelerómetro MMA7361.
- Adquisición de los datos correspondientes a la temperatura externa de la mascota. Procesamiento de la información correspondiente a la temperatura obtenida por el sensor MLX90614.
- Procesamiento de información, esta parte del sistema de control realiza varias tareas como: la conversión de los valores de latitud y longitud al formato GPS establecido para la ubicación en el mapa de Google. El almacenamiento y preparación de información para su posterior envío.
- Conexión con la red celular para el envío de información.
- Realizar el envío de la información adquirida y procesada hacia el servidor web.

La Unidad de control está conformado por la tarjeta programable Arduino UNO, se encarga de la activación de los componentes a nivel de hardware y la realización de todos los procesos, cada proceso específico, como la recepción, el procesamiento, y el envío está analizado en distintos apartados en este capítulo. Las tareas a nivel general del proyecto, se indican en la Figura 37.

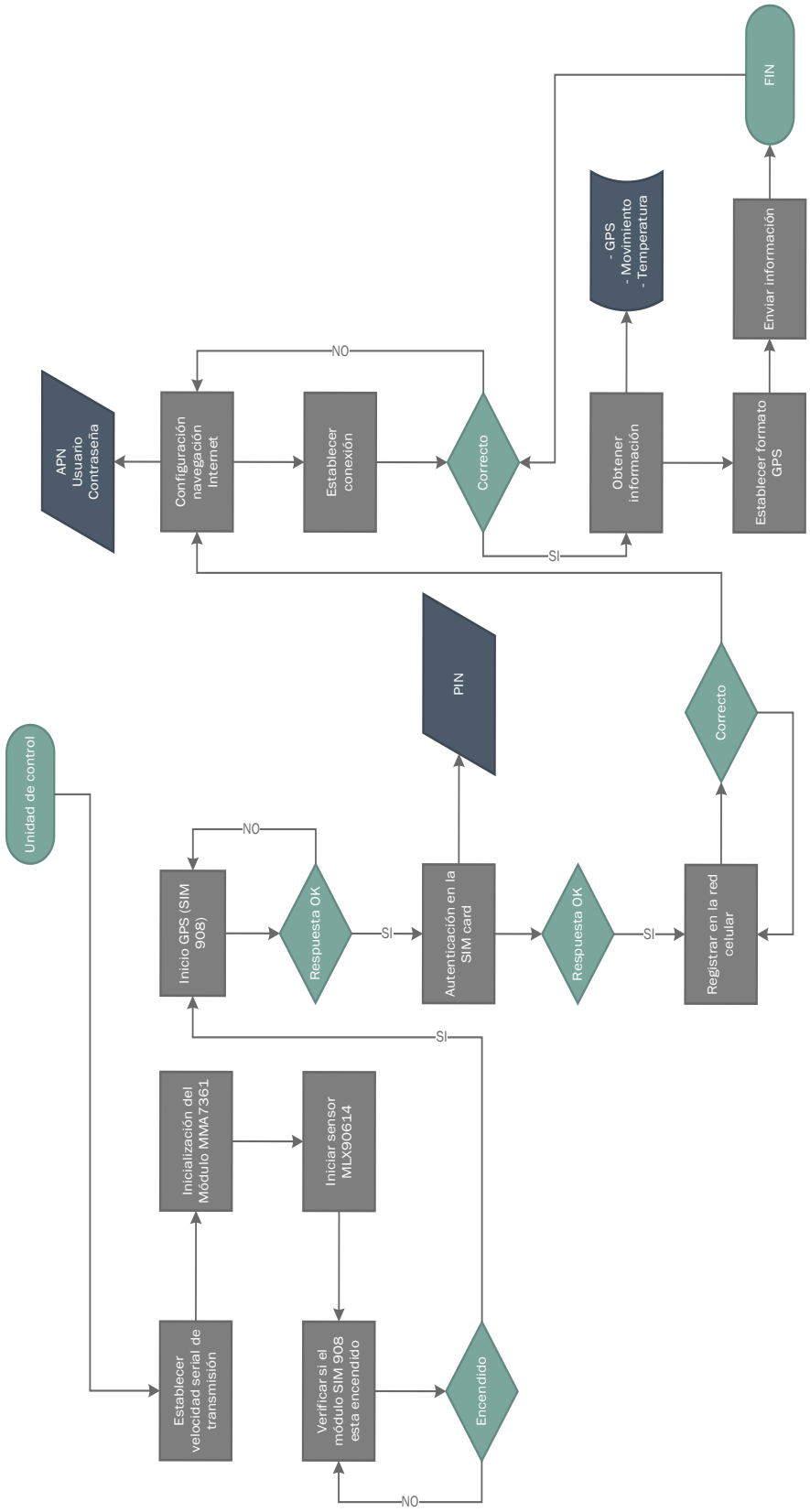


Figura 37 Diagrama de flujo del proceso de control de información en el sistema.

En la Figura 37 se establece el proceso general de flujo de información, la tarjeta programable realiza acondicionamiento y control del sistema, es el cerebro del sistema de hardware y procesa todas las tareas, los sensores y módulos son conectados a la tarjeta, y esta se encarga de procesar la información recibida mediante sentencias cargadas al hardware por medio del software Arduino IDE (Ver Anexo 2).

3.1.4. Procesamiento de información GPS.

Es importante realizar el procesamiento de los datos recibidos por la antena GPS al formato que se requiere para la ubicación en el mapa. Los datos de latitud y longitud ingresan en formato NMEA. Por lo que se debe realizar la conversión al formato grados decimales, el cual es el formato de ubicación en el que se representan los marcadores en el mapa de Google. La fórmula de conversión se representa en la Formula (1).

$$ddmm.mmmm = (d)dd + (mm.mmmm/60) \quad (1)$$

Por medio de un ejemplo se representa la conversión mencionada en la Formula (1).

-32 18.0489 = 32 grados + 18.0489 / 60 = -32.300815
64 47.5086 = 64 grados + 47.5086 / 60 = 64.79181

Los valores -3218.0489 y 6447.5086 representan la información de latitud y longitud recibida en formato NMEA. El signo positivo o negativo de los valores representan el punto cardinal en que se encuentra ubicada la posición ingresada, el signo “+”, representa los puntos Norte y Este, mientras que el signo “-“, representa el Sur y Oeste. El diagrama de flujo de la conversión NMEA a grados decimales se establece en la Figura 38.

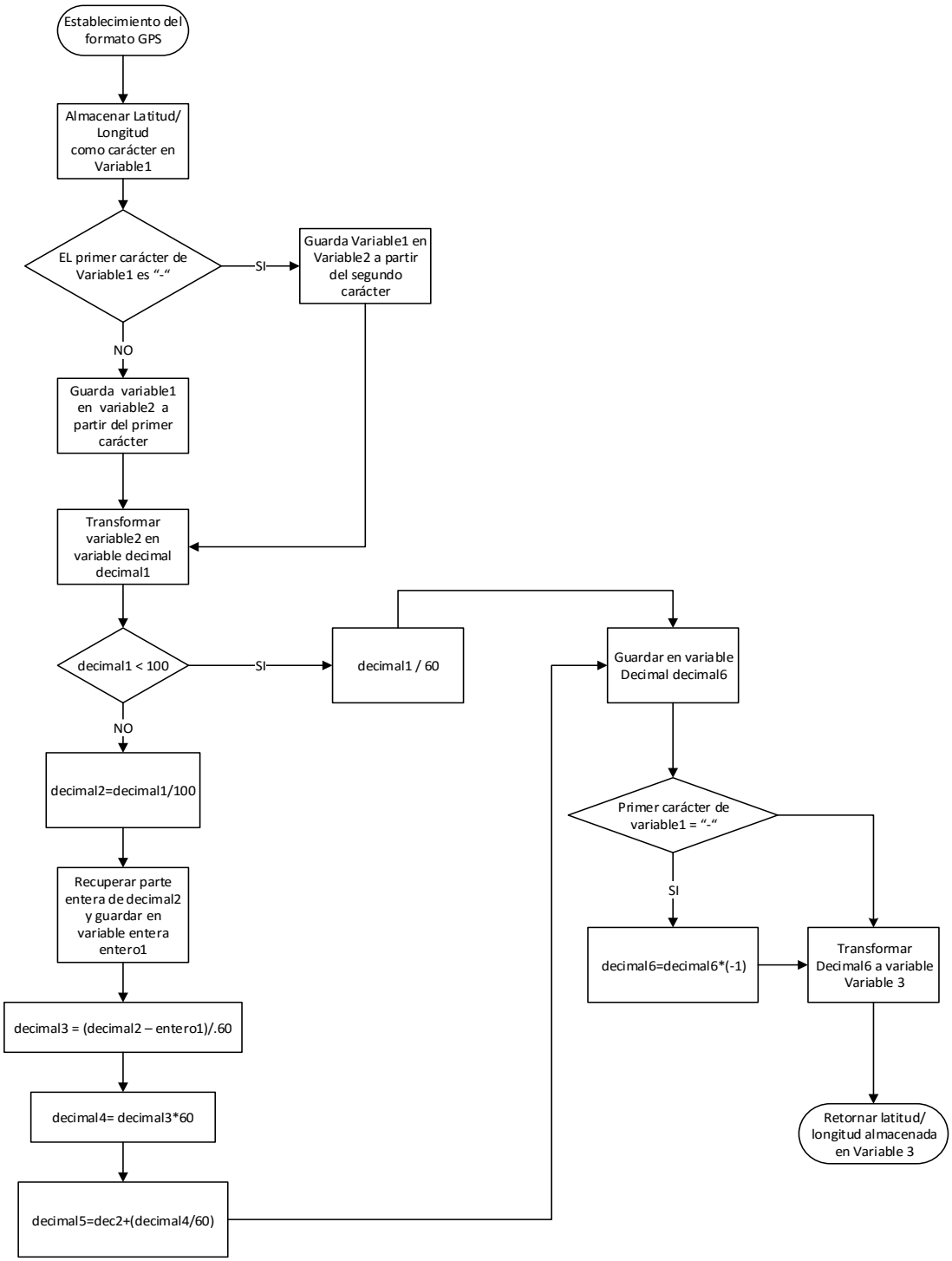


Figura 38 Diagrama de flujo del establecimiento de formato GPS.

El procesamiento de información GPS, se establece por medio de programación en la tarjeta programable Arduino UNO, una vez que la información de

posicionamiento geográfico (latitud/longitud) ingresa al sistema, es almacenado en la memoria de la tarjeta Arduino UNO, y realiza las operaciones indicadas en la Figura 38, para su conversión al formato grados decimales, formato establecido para la ubicación de un punto en el mapa, mediante el sistema de software.

3.1.5. Transmisión de información

Una vez obtenida y procesada la información, ésta es enviada hacia el servidor web para su tratamiento y almacenamiento en una base de datos. El módulo SIM 908 realiza el envío mediante tecnología GSM/GPRS, por lo que se requiere la adquisición de un chip de telefonía celular con datos incluidos, para conocer con detalle la manera de montar el proyecto (Ver **Anexo 5**). El envío mediante tecnología GSM/GPRS se realiza por medio del módulo SIM 908 controlado por la tarjeta Arduino UNO, mediante comandos AT (Ver **Anexo 3**). En la Figura 39 se representa los componentes de hardware involucrados en el envío de información por medio de la red celular.

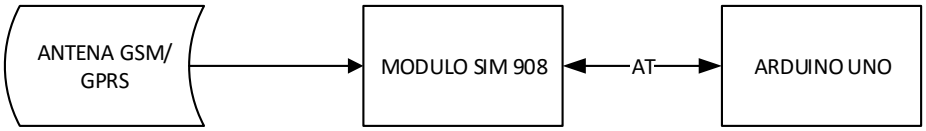


Figura 39 Representación de hardware involucrado en el envío de información

Para enviar la información por medio de la red celular, se debe autenticar en el chip de telefonía celular por medio del código PIN. Además se debe autorizar el envío de información por medio de la red celular, por lo que se debe disponer de las credenciales de la red: la APN, el nombré de usuario y la contraseña. En la Figura 40 se muestra por medio de un diagrama de flujo los procesos que realiza el sistema para la transmisión de información.

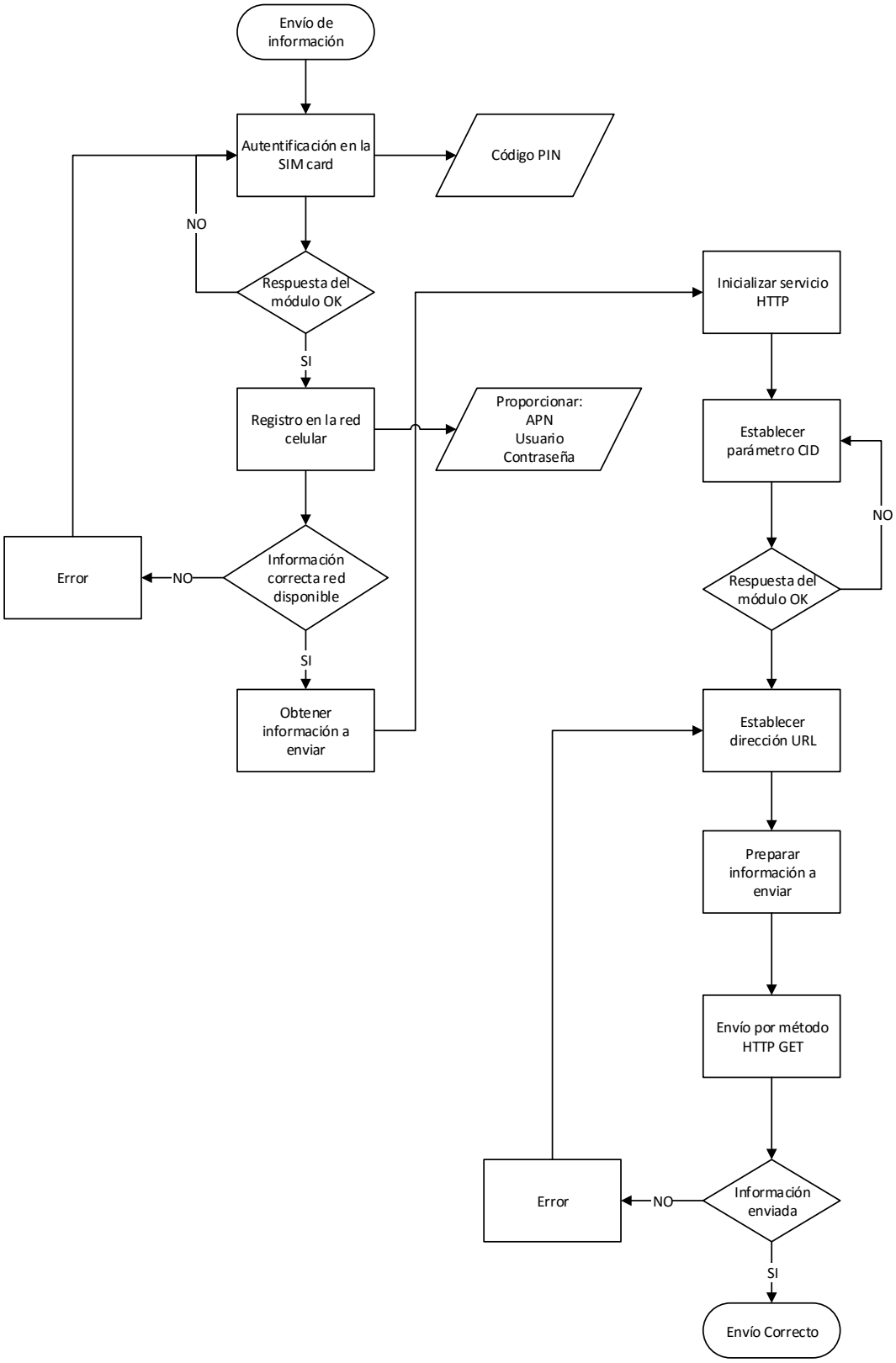


Figura 40 Diagrama de flujo del proceso de envío de información.

Las sentencias condicionales de la Figura 40, esperan respuestas positivas o negativas por parte del módulo SIM 908, para continuar con el proceso. Mientras una tarea no se cumpla en su totalidad, el programa se va a ejecutar de forma indefinida hasta recibir una respuesta favorable por parte del módulo.

3.1.6. Alimentación.

La implementación de la bateria de alimentación depende de diversos aspectos, la facilidad de adquisición, la corriente de suministro y el voltaje, para que la batería sea capaz de alimentar a todos los componentes involucrados en el proyecto. En la Tabla 11 se establece un resumen del consumo de corriente y el voltaje de funcionamiento que requiere cada componente.

Tabla 11

Resumen de voltaje y corriente de dispositivos involucrados en el proyecto

Dispositivo	Voltaje de alimentación	Consumo de corriente
Arduino UNO	7-12 V (Recomendado)	50mA (máximo)
Módulo SIM 908	3.2-4.8V (Recomendado)	2A (Rafagas de transmisión) 500 mA (Operación normal)
Acelerometro MMA7361	3.3V	40uA
MLX90614	5V	25mA

Se debe poner en consideración que el módulo SIM 908 requiere un alto consumo de corriente para su correcta operación, la batería que se tuvo a disposición posee las siguientes características:

Bateria Litio: 3.7V - 2500 mA

Salida USB: 5V

Para conocer teóricamente el tiempo de descarga de la batería, con los componentes que necesita el proyecto para funcionar, se utilizó la Formula (2).

$$Tiempo\ de\ descarga = \frac{carga\ electrica\ bateria}{consumo\ electrico\ del\ dispositivo} \quad (2)$$

El módulo SIM 908, requiere distinto suministro de corriente, según su modo de operación, cuando el módulo se encuentra transmitiendo o recibiendo información GSM/GPRS, en estado de operación del módulo, el tiempo de descarga de la batería se especifica en la Formula (3).

$$Td = \frac{2500mAH}{2000mA} = 1.25H \quad (3)$$

Para el modo de operación normal, el tiempo en horas de la duración de bateria, se establece en la Formula (4).

$$Td = \frac{2500mAH}{500mA} = 2.5H \quad (4)$$

El consumo de la tarjeta programable Arduino UNO, el acelerómetro y el sensor de temperatura es mínimo. Los valores calculados en la Formula (3) y (4), son valores teóricos, en la etapa de pruebas se realiza un análisis real del tiempo de descarga de la batería.

Para que el sistema obtenga una autonomia y portabilidad se añadió una etapa para carga del circuito de forma portable, el diagrama de bloques de la Figura 41 muestra la forma de implementación del circuito mencionado. Para la implementación se usó dos cargadores universales, con los cuales se carga la batería y alimenta a los componentes de hardware de manera simultanea.

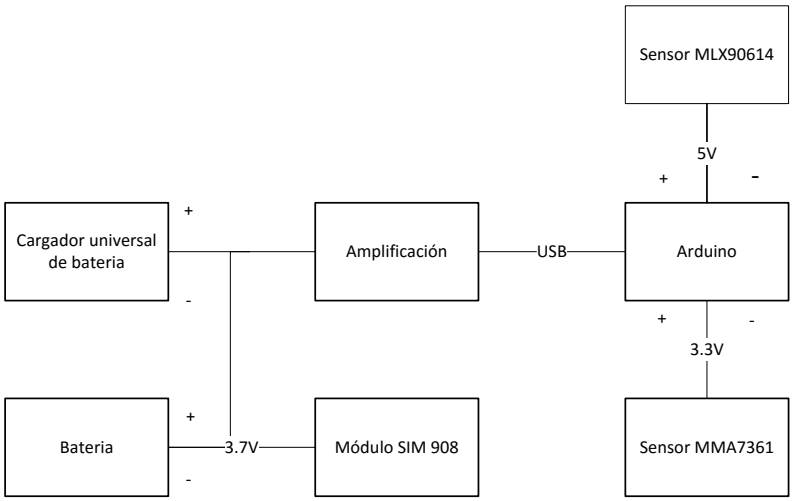


Figura 41 Diagrama de bloques de la alimentación del hardware.

Los bloques representados en la Figura 41, se describen:

- Cargador Universal de batería: Se adquirió un cargador universal de batería con una entrada de poder para conectar el sistema a la toma de corriente, y cargar la batería dispuesta.
- Amplificación: Se dispone un segundo circuito recuperado de un cargador universal de baterías, con un conector USB integrado, para la alimentación de la tarjeta programable Arduino UNO.
- Batería: Se conecta a la salida del "cargador Universal", al módulo SIM 908, y a la etapa de amplificación para la conexión de la tarjeta programable Arduino UNO.
- El Módulo SIM 908 y la tarjeta programable consumen la energía proporcionada por la batería, el sensor MMA7361 se alimenta de la tarjeta programable a través del pin de 3.3V, y el sensor MLX90614 a través del pin 5V de la tarjeta programable Arduino.

3.2. Implementación de Hardware

La implementación del hardware corresponde a la conexión y configuración de los dispositivos que conforman el sistema, los instrumentos de hardware involucrados son:

- Tarjeta Programable Arduino UNO.
- Módulo SIM 908 para Arduino.
- Sensor Acelerómetro MMA7361.
- Sensor de temperatura infrarrojo MLX 90614.

En el Anexo 5 se dispone el manual de ensamble del proyecto, en la cual se especifica la implementación de las antenas GSM/GPRS y GPS en el módulo SIM 908, además de la implementación del chip de telefónica celular. El Anexo 2 contiene la guía de inicio del software de programación de la tarjeta programable Arduino UNO, se establece la conexión de la tarjeta a la computadora y una visión general del software Arduino IDE mediante el cual se carga en la tarjeta programable Arduino UNO el software establecido para el funcionamiento del sistema de Hardware. La codificación del software está basado en el código que se encuentra en el Anexo 6, realizando variantes según la necesidad de funcionalidad que se requirió para el proyecto.

3.2.1. Implementación del módulo SIM 908

Para la adquisición de las coordenadas geográficas se conecta la tarjeta Arduino UNO con el módulo SIM 908 con la antena GPS conectada. El control del módulo se realiza mediante comandos AT a través de la tarjeta programable. El envío y recepción de información GPS y GSM/GPRS se realiza por medio de los pines RXD y TXD de la tarjeta programable Arduino UNO, se los representa mediante la numeración 0 y 1 respectivamente, pero presentan una dificultad al realizar el envío y recepción de forma portátil, ya que su funcionamiento depende de la conexión de la tarjeta al puerto USB del computador, con el implemento monitor serial del software Arduino UNO IDE ejecutándose (Véase Anexo 2). Para la implementación de un sistema portable se requiere añadir una librería la cual convierta dos pines digitales de la tarjeta en configuración serial RX y TX, la librería que realiza la acción mencionada es: “SoftwareSerial.h” y la conversión se la realiza inicializando los pines 3 y 4 de la siguiente manera:

```
SoftwareSerial mySerial(3, 4);
```

En donde el pin 3 se lo establece con configuración de recepción RX, y el pin 4 de transmisión TX. La conexión de la tarjeta programable Arduino UNO con el módulo SIM 908 se representa en la Figura 42.

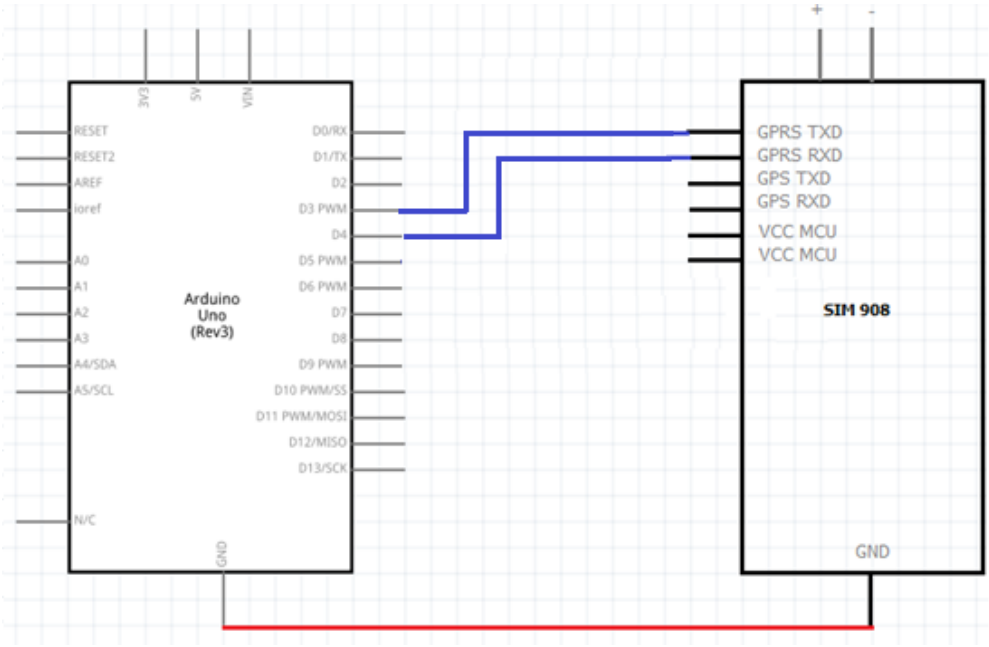


Figura 42 Esquema de conexión Arduino UNO – SIM 908.

Es importante iniciar el valor de la velocidad del puerto serial, para la transmisión y recepción de información. Mediante la realización de pruebas e investigación se dispuso la velocidad de transmisión en 57600 bps, dicha velocidad trabaja de manera adecuada con el módulo SIM 908.

```
mySerial.begin(57600);
```

Además de la inicialización de los pines de transmisión y recepción, se debe considerar las variables que se encargan de almacenar la información para la conexión del módulo SIM 908 a la red celular y las variables encargadas del almacenamiento de información general del sistema. En la Tabla 12 se establece un resumen de las variables globales inicializadas, con su descripción y valor de inicialización.

Tabla 12

Variables globales usadas en el desarrollo del hardware

Variable	Tipo	Valor Inicial	Descripción
pin	char	Vacío o “ ”	Código PIN
apn	char	"internet.claro.com.ec"	Almacena el nombre del punto de acceso correspondiente a la empresa de telefonía celular a la que corresponde el chip.
user_name	char	Vacío o “ ”	Almacena el nombre de usuario para la autentificación en el Punto de Acceso.
password	char	Vacío o “ ”	Almacena la contraseña para la autentificación en el Punto de Acceso.
longitud	char	Sin inicialización.	Almacena el valor de la longitud geográfica recibida por el sistema GPS.
latitud	char	Sin inicialización.	Almacena el valor de la latitud geográfica recibida por el sistema GPS.
eje_z	int	Sin inicialización.	Almacena el valor de la vibración del sensor en el eje z.
aux_str	char	Sin inicialización.	Almacena los comandos AT.
frame	char	Sin inicialización.	Almacena la trama GPS.
url	char	“http://locatesystem.freevar.com/locate/cargar.php”	Variable que almacena la dirección URL del archivo de configuración que procesa y almacena la información enviada por el sistema de hardware.

Los valores de inicialización de la variables, APN, user_name y password, son facilitadas por la empresa de telefonía a la que corresponde el chip dispuesto en el módulo SIM 908 (Ver Anexo 5), por facilidad de acceso, se realizó la configuración con la empresa de telefonía celular Claro. Además de la declaración de las variables correspondientes a la información de conexión a la red celular, se debe inicializar las

variables que van a almacenar la información de latitud, longitud y la dirección en donde se encuentra alojado el archivo de configuración en el servidor web (Ver Tabla 12). Además se realizó la declaración de diversas variables usadas en operaciones varias. La estructura básica del lenguaje de programación en Arduino se puede verificar en el Anexo 2.

La recepción y envío de la información GPS y GSM/GPRS se la realiza mediante comandos AT (Ver Anexo 3), para obtener facilidad al momento del envío de un comando AT al módulo, se estableció el siguiente método:

```
sendATcommand(AT, "OK", 2000);
```

El método para el envío de comandos AT se lo denominó sendATcommand, consta de 3 atributos, en donde el primer atributo corresponde al comando AT (Ver Anexo 3), el segundo corresponde a la respuesta esperada, la cual es retornada por el módulo SIM 908 y el tercer atributo el tiempo de espera. La función general del método es retornar el valor de 1 cuando existe una respuesta positiva por parte del módulo SIM 908, y el valor de 0 cuando no hay respuesta, ya sea por un error o porque el módulo no se ha iniciado correctamente.

El software Arduino UNO IDE consta de dos métodos principales “setup( )” y “loop ( )”, en el método “setup” se establece la configuración inicial del proyecto en Arduino, mientras que en el método “loop” contiene el programa que se ejecuta cíclicamente Ambas funciones son necesarias para que el programa trabaje. El diagrama de clases de la Figura 43muestra la representación general del programa del Sistema de hardware en Arduino UNO.

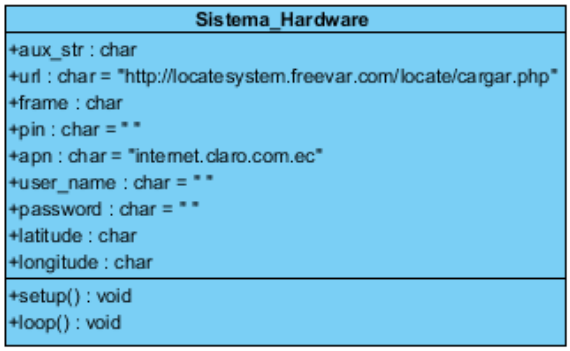


Figura 43 Diagrama de clases general del sistema de hardware

El módulo SIM 908 realiza la recepción de la información de posicionamiento global GPS y del envío general de la información por medio de tecnología GSM/GPRS a un script de configuración alojado en el servidor web, como se indica en la Figura 43 el código general del programa se encuentra conformado por dos métodos, en cada método se establece la configuración de inicialización y obtención de información GPS y de conexión con la red de datos GSM/GPRS para el posterior envío. Los métodos “setup ( ) y “loop ( )” son nativos del software Arduino UNO IDE (Ver Anexo 2). En la Tabla 12 se muestra las variables usadas a nivel general en el programa, las cuales son usadas en métodos dispuestos en los métodos principales, el diagrama de clases UML de la Figura 41 representa las variables y los sub-métodos que utilizan dichas variables. En primera instancia en la Figura 44 se especifica las variables y funciones incluidas en el método “setup ( )”.

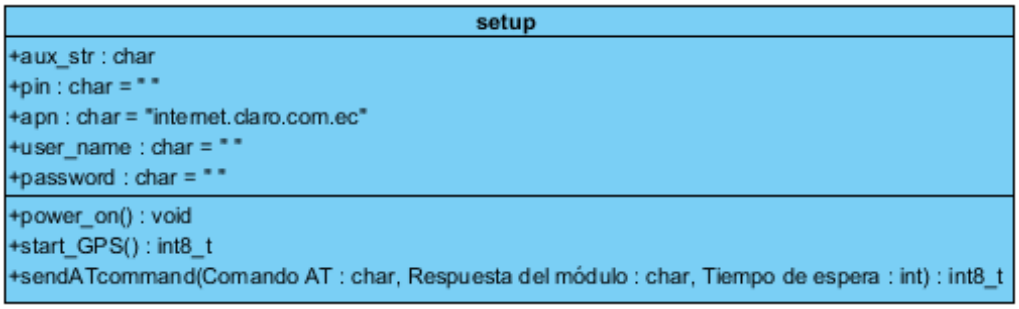


Figura 44 Diagrama de clases del método “setup”

En el apartado de diseño de hardware por medio de diagramas de flujo se estableció el funcionamiento general del sistema, en el método “setup ( )” se establece la configuración e inicialización inicial del programa. La primera operación a realizar para la inicialización y puesta en marcha del módulo SIM 908 es la verificación de encendido del módulo por parte de la tarjeta Arduino. El método “power_on” (Véase Figura 44) realiza la verificación de inicio del módulo, por medio del método “sendATcommand”, se envía el comando AT: “AT” (Ver Anexo 3), dicho comando requiere de una respuesta por parte del módulo, la respuesta esperada es “OK”, si el módulo no retorna dicha respuesta a la tarjeta programable, significa que no se encuentra iniciado.

```
sendATcommand("AT", "OK", 2000);
```

El método “sendATcommand” devuelve un valor de 1 si existe respuesta por parte del módulo, caso contrario el valor es de 0, mientras el módulo no se encuentre iniciado, el programa no sigue, por lo que se almacena el valor devuelto por el método antes mencionado en la variable “answer”, y mientras la variable no es igual a uno se sigue enviando el comando por medio de un bucle de repetición.

```
while(answer == 0){
            answer = sendATcommand("AT", "OK", 2000);
        }
```

Una vez que se constató el inicio del módulo, se procede a realizar la autenticación en el chip de telefonía celular mediante el código PIN, por medio del método “sendATcommand”, se establece el comando AT para el establecimiento del código PIN en el módulo.

```
snprintf(aux_str, sizeof(aux_str), "AT+CPIN=%s", pin);
sendATcommand(aux_str, "OK", 2000);
```

El método “snprintf” almacena en la variable “aux_str” el comando AT: “AT+CPIN = %s”, en donde %s se almacena el valor de la variable pin. En la Tabla 12 se puede verificar el valor de inicialización de la variable. Una vez realizada la autenticación, el método “start_GPS ( )” inicia el servicio GPS.

```
sendATcommand("AT+CGPSPWR=1", "OK", 2000);
sendATcommand("AT+CGPSRST=0", "OK", 2000);
```

Por medio del comando AT: “AT+CGPSPWR = 1” (Ver Anexo 3), se enciende el módulo. Por medio del comando AT: “AT+CGPSRST = 0” (Ver Anexo 3), se establece el modo de inicialización GPS, el valor 0 indica el inicio en modo “COLD”, valor recomendado para el inicio GPS por primera vez. Por medio del comando AT: “AT + GPSSTATUS?” se espera el retorno de la afirmación de obtención de la señal GPS por parte del módulo, la respuesta una vez que se recuperó una posición puede ser “2D FIX “o “3D FIX” los valores indican la construcción en 2 dimensiones y 3 dimensiones respectivamente, cualquiera de las dos respuestas

indican que el módulo recibió la información geográfica correcta, por lo que el método devuelve el valor de 1 si se recibe la señal esperada.

Una vez obtenida la información GPS se realiza el registro en la red celular y la posterior autentificación para la utilización de Internet. El registro y autenticación se la realiza de la siguiente manera:

```
// Espera un correcto registro en la red celular
while (sendATcommand("AT+CREG?", "+CREG: 0,1", 2000) == 0);

// Se establece las variables de autentificación para el uso de
// Internet
sendATcommand("AT+SAPBR=3,1,\"Contype\",\"GPRS\"", "OK", 2000);
snprintf(aux_str,      sizeof(aux_str),"AT+SAPBR=3,1,\"APN\",\"%s\"",
apn);
sendATcommand(aux_str, "OK", 2000);

snprintf(aux_str,  sizeof(aux_str),  "AT+SAPBR=3,1,\"USER\",\"%s\"",
user_name);
sendATcommand(aux_str, "OK", 2000);

snprintf(aux_str,  sizeof(aux_str),  "AT+SAPBR=3,1,\"PWD\",\"%s\"",
password);
sendATcommand(aux_str, "OK", 2000);
```

Para conocer los valores de las variables de autentificación para el envío de información por medio de Internet, verifique la Tabla 12. En el método “loop” se establece el programa de la tarjeta programable Arduino UNO que realiza de forma cíclica, es decir las funciones que se van a realizar de manera indefinida en el programa, en la Figura 45 se muestran los métodos que se ejecutan de manera cíclica, y corresponden a la lectura de la trama GPS procesada por el módulo, establecimiento de formato de dicha trama y al envío de la información.

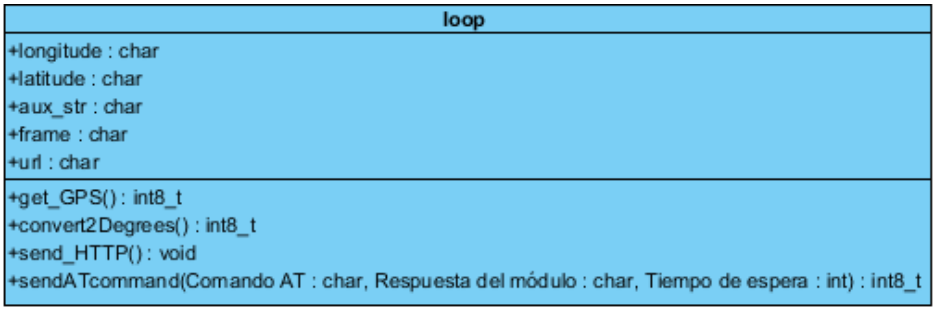


Figura 45 Diagrama de clases del método "loop".

El sistema debe obtener la ubicación de forma continua, lo cual se realiza por medio del método "get_GPS", en dicho método, se obtiene la ubicación actual mediante comandos AT (Ver Anexo 3), por medio del método "sendATcommand" se realiza el envío del comando:

```
sendATcommand("AT+CGPSINF=0", "AT+CGPSINF=0\r\n\r\n", 2000);
```

El programa espera la respuesta de los valores receptados por la antena y procesados por el módulo en el formato GPS establecido, si es que existe respuesta por parte del módulo, los valores procesados por el módulo SIM 908 se almacenan en la variable "frame",

```
frame[counter] = Serial.read();
```

Caso seguido se divide y almacenan la información recibida en variables "String", para su posterior establecimiento de formato y envío.

```
strtok(frame, ","); //Divide el frame por ,
// Obtiene longitud y la almacena en la variable longitude
strcpy(longitude,strtok(NULL, ","));
// Obtiene latitud y la almacena en la variable latitude
strcpy(latitude,strtok(NULL, ","));
```

Una vez obtenidos de la trama los valores de latitud y longitud para almacenarlos en variables "String", se procede al establecimiento del formato que se necesita para ubicar un marcador en el mapa de Google, mediante los métodos:

```
convert2Degrees(latitude);
convert2Degrees(longitude);
```

Los métodos retornan los valores de longitud y latitud en el formato que requieren los mapas de Google, usados tanto en la aplicación Android, como en la página Web, para ubicar un punto marcador en el mapa, los métodos realizan la operación analizadas con anterioridad en la etapa de diseño.

Además de la adquisición de los valores de posicionamiento global de manera continua, se realiza el envío de dichos valores, lo cual se lo realiza por medio del método "sendHTTP ( )", tanto la obtención y envió se lo realiza de manera cíclica, ya que se debe considerar que el valor de posicionamiento cambia según la ubicación a nivel geográfico de la antena. Para iniciar con el envío, primero se verifica la inicialización del servicio HTTP mediante el comando AT:

```
answer = sendATcommand("AT+HTTPINIT", "OK", 10000);
```

La variable "answer", es del tipo entero, y recibe el valor de 1 devuelto por el método "sendATcommand" si es que se obtiene una respuesta del comando AT favorable o caso contario el valor es 0. Una vez que se verifica si el servicio HTTP se encuentra inicializado, se procede con establecer los valores de los parámetros HTTP, correspondiente al identificador de perfil de portador.

```
answer = sendATcommand("AT+HTTPPARA=\"CID\",1", "OK", 5000);
```

El método "sendATcomand" retorna el valor de 1, si es que la respuesta del comando AT enviado por medio de la tarjeta Arduino es "OK". Una vez fijados los parámetros CID, se establece la dirección URL en donde se aloja el archivo de configuración en el servidor web a donde se realiza el envío. El valor de la variable "url" correspondiente a la dirección URL, el valor de la variable se puede verificar en la Tabla 12.

```
sprintf(aux_str, "AT+HTTPPARA=\"URL\",\"%s", url);
```

```
Serial.print(aux_str);
```

En la variable “aux_str” se almacena el comando AT para el establecimiento de la dirección URL a donde se va a enviar, caso seguido se imprime el comando en el puerto serial.

```
sprintf(frame,"?visor=false&latitude=%s&longitude=%s&altitude=%s
&usuario=%s  &eje_z=%d&collarid=%s", latitude, longitude, usuario,
eje_z, collarid);
Serial.print(frame);
```

En la variable “frame” se almacena los valores de las variables obtenidas y las cuales van a ser enviadas para su procesamiento en el servidor web, mediante la variable “answer”, se verifica si el modulo recibió la dirección y las variables de manera adecuada por medio de la respuesta recibida en la variable “answer”.

```
answer = sendATcommand("\"", "OK", 5000);
```

Si la respuesta es la esperada, se devuelve el valor de 1 por medio del método “sendATcommand”, por lo que se realiza el envío por medio del método HTTP “GET” por medio del siguiente comando AT (Ver Anexo 3).

```
sendATcommand("AT+HTTPACTION=0", "+HTTPACTION:0,200", 30000);
```

Tanto la recepción, el establecimiento del formato GPS y el envío de la información se la realiza de manera cíclica cada cierto tiempo.

3.2.2. Implementación del sensor acelerómetro MMA7361

La conexión entre el sensor MMA7361 y la tarjeta Arduino se la realiza de forma directa, su uso con la tarjeta es común y Arduino IDE (Software oficial de la plataforma Arduino), incluye la librería “AcceleroMMA7361.h” (Ver Anexo 2) para la adquisición de los datos medidos por el transductor, el esquema de conexión del módulo con la tarjeta se representa en la Figura 46.

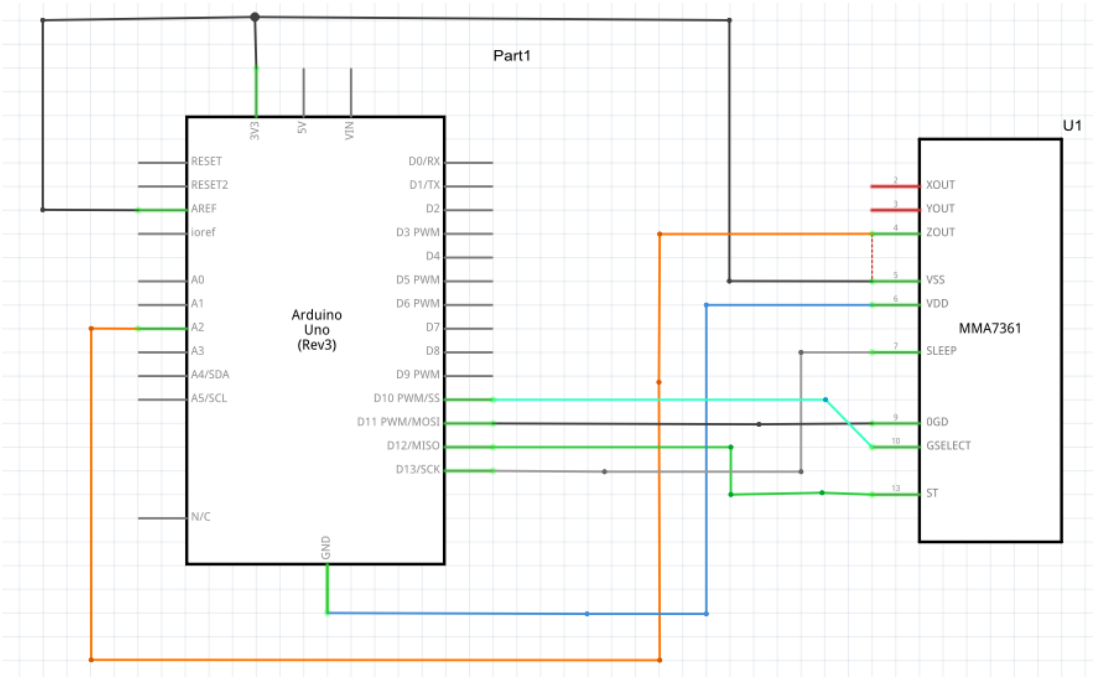


Figura 46 Esquema de conexión Módulo AcelerometroMMA7361 con Arduino Uno

En la Tabla 13 se muestra un resumen de los pines de conexión, los pines XOUT, YOUT y ZOUT proporcionan los valores de vibración e inclinación del movimiento del acelerómetro en los ejes X, Y y Z respectivamente, el sistema a diseñar sólo requiere el uso del eje z, porque únicamente se va a medir el movimiento en dicho eje, dicho valor ingresa al pin A2 de la tarjeta Arduino UNO (Ver Figura 46). Los pines analógicos reciben la información del sensor, para su posterior implementación por medio de la librería “AcceleroMMA7361.h”. La alimentación del acelerómetro se la realiza mediante la tarjeta programable Arduino UNO, el transductor tiene dos pines de alimentación, 5 V y 3.3 V, su funcionamiento es con 3.3V, pero por motivo de las baterías que existen actualmente en el mercado, tiene un pin de 5V, el cual posee un regulador de tensión a 3.3 V. Como la tarjeta Arduino UNO consta de una salida de 3.3 V, no es necesario la utilización del pin de 5V, la polarización negativa (tierra) se conecta de forma común entre el sensor y la tarjeta programable.

Tabla 13

Conexión Arduino UNO a módulo acelerómetro MMA7361

Arduino UNO	Acelerómetro MMA7361
13	SL (Sleep)
12	ST (SelfTest)
11	0G (Zero G)
10	GS (gSelect)
A0	X
A1	Y
A2	Z
3.3V	3.3V
GND	GND

Los signos vitales son representados a través del acelerómetro MMA7361, su configuración en la tarjeta programable Arduino UNO se lo realiza por medio de la librería "AcceleroMMA7361.h". La inicialización del acelerómetro la realiza el método "setup", el diagrama de clases de la detalla un esquema general del método "setup" relacionado con los métodos heredados por la librería correspondiente al sensor acelerómetro.

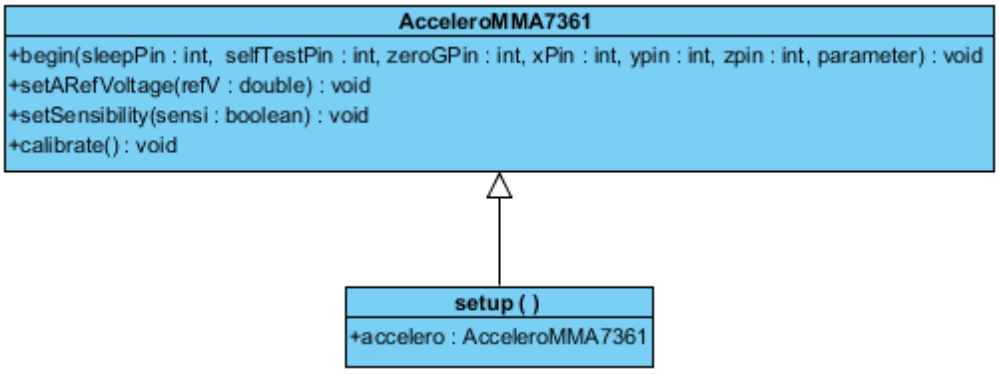


Figura 47 Diagrama de clases método "Setup "y librería "AcceleroMMA7361"

El método “Setup” hereda los métodos de la librería “AcceleroMMA7361.h”, el llamado a los métodos se los realiza por medio del objeto “accelero” de tipo “AcceleroMMA7361”, la inicialización del módulo se la realiza de la siguiente manera:

```
//inicialización de pines (SL, ST, 0G, GS A0, A1, A2)
accelero.begin(13, 12, 11, 10, A0, A1, A2);
//Voltaje de referencia con que se va a alimentar el sensor
accelero.setARefVoltage(3.3);
//Establecer la Sensibilidad
accelero.setSensitivity(LOW);
//Calibración
accelero.calibrate();
```

La variable “accelero” llama a la ejecución de los métodos incluidos en la librería. El método “begin” inicializa el acelerómetro, el método “setARefVoltage” dispone el voltaje en 3.3V, voltaje de funcionamiento del sensor. En el método “setSensitivity” se especifica la sensibilidad del acelerómetro y la calibración del mismo se la realiza mediante el método “calíbrate”.

Una vez inicializado el sensor acelerómetro, se procede con la adquisición de la información detectada por el sensor, correspondiente a las vibraciones en el eje Z, el diagrama de clases de la muestra la interacción del método “loop” con la librería “AcceleroMMA7361”.

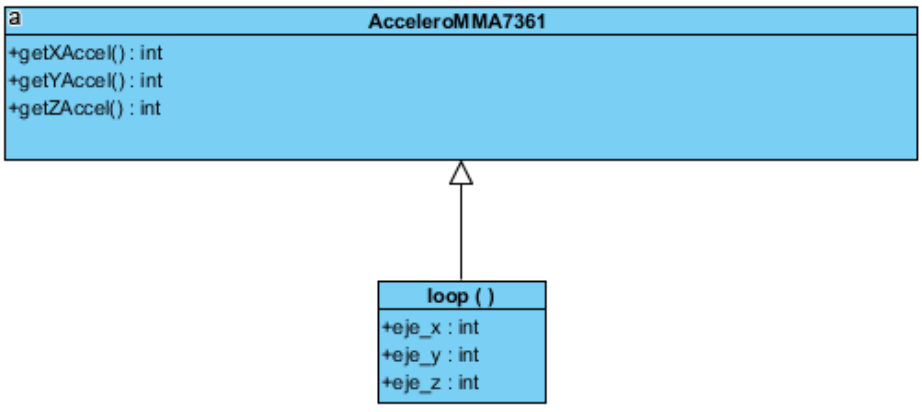


Figura 48 Diagrama de clases general método “loop” con la librería “AcceleroMMA7361”.

El método “loop” hereda los métodos de la librería “AcceleroMMA7361.h”, el llamado a los métodos se los realiza por medio del objeto “accelero” del tipo AcceleroMMA7361, la obtención de información en cada eje se la realiza de la siguiente manera:

```
eje_x = accelero.getXAccel();
eje_y = accelero.getYAccel();
eje_z = accelero.getZAccel();
```

Como se ha mencionado, únicamente se va a considerar los valores obtenidos en el eje Z, aunque el valor no va a ser una medida de signos vitales real, es importante para conocer las vibraciones en el eje e informar si el animal se encuentra en movimiento**.**

3.2.3. Implementación del sensor MLX90614

La conexión entre el sensor MLX90614 y la tarjeta Arduino se la realiza de forma directa, el sensor tiene un acoplamiento para la conexión con la plataforma de forma directa, se debe incluir la librería “i2cmaster.h” (Véase Anexo 2), para la comunicación por medio del protocolo I2C de la plataforma con el sensor. El esquema de conexión se puede verificar en la Figura 49.

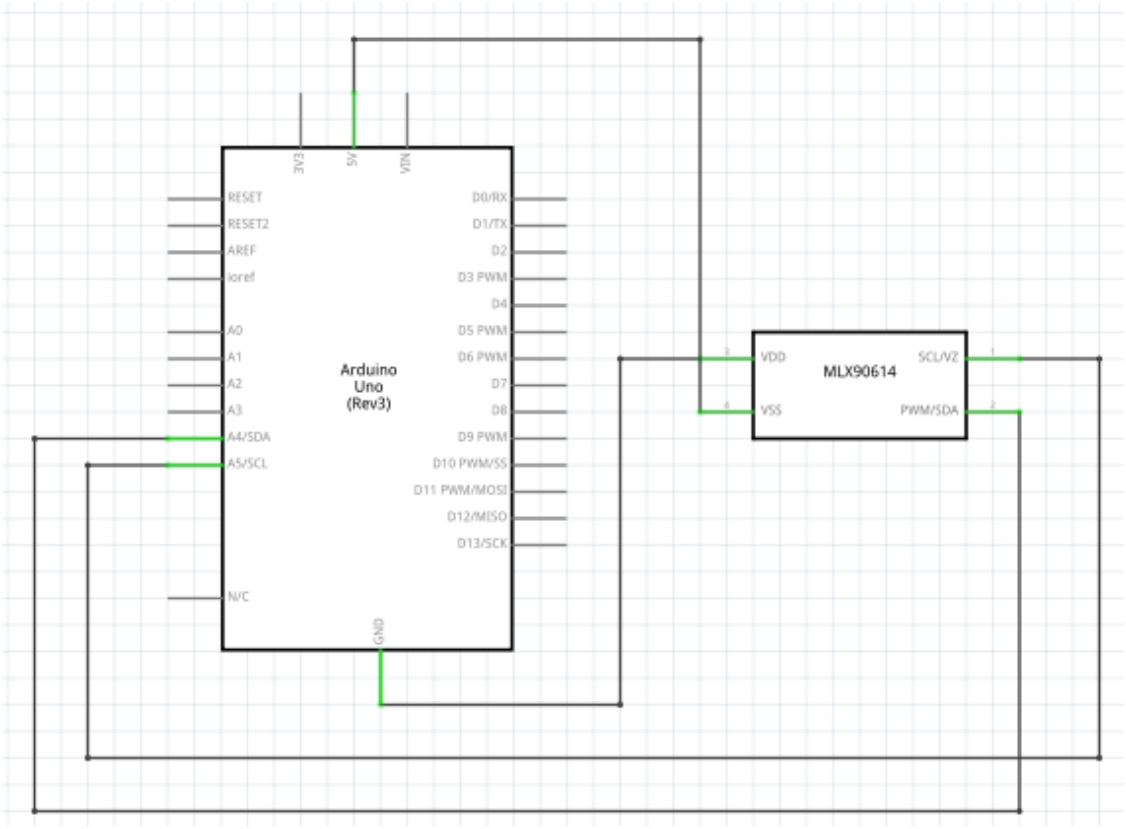


Figura 49 Esquema de conexión sensor MLX90614 con Arduino Uno

El código implementado para el funcionamiento del sensor de temperatura con la tarjeta programable Arduino UNO fue recuperado de un blog en línea, el autor corporativo del blog es Arduinadas, y el enlace de donde se recuperó el código es: "http://arduinadas.blogspot.com/2012/06/sensor-de-temperatura-mlx90614.html".
Se realizó pruebas de funcionamiento para conocer y establecer la precisión del sensor de temperatura MLX90614, y corroborar el correcto funcionamiento del programa recuperado.

3.2.4. Implementación Final y Conexión General

Una vez analizada la implementación del circuito por partes, se procedió a implementar la conexión general del circuito y la disposición de todos los componentes en una caja, para su posterior uso en mascotas. La Figura 50 muestra la conexión a nivel de hardware general de las placas, sensores y módulos usados en la implementación del proyecto.

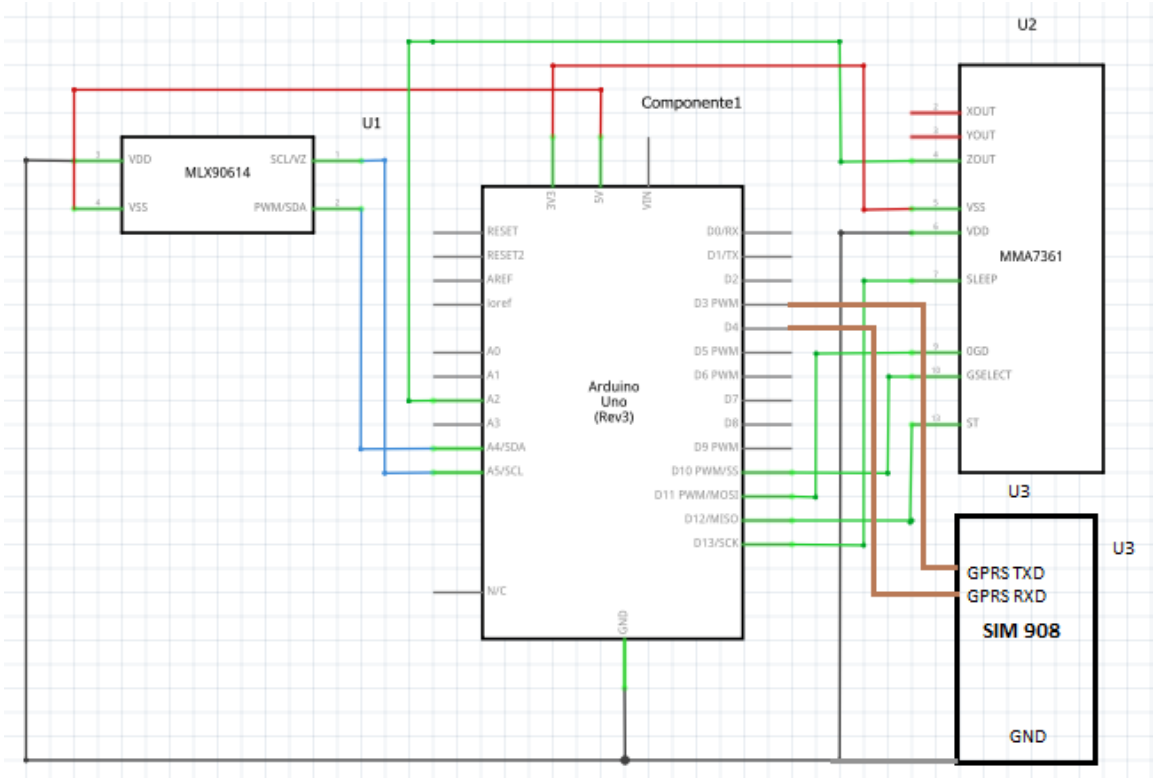


Figura 50 Esquema de conexión general.

En la Figura 50 se muestra todos los componentes analizados con anterioridad. Los componentes son energizados mediante una batería de litio, cuyas especificaciones ya fueron analizadas, todos los componentes son establecidos en una caja capaz de almacenar el hardware, para su posterior establecimiento en un arnés para el uso en una mascota. Los componentes analizados y representados en el esquema de conexión general (Ver Figura 50) se disponen en una caja común como se muestra en la Figura 51.

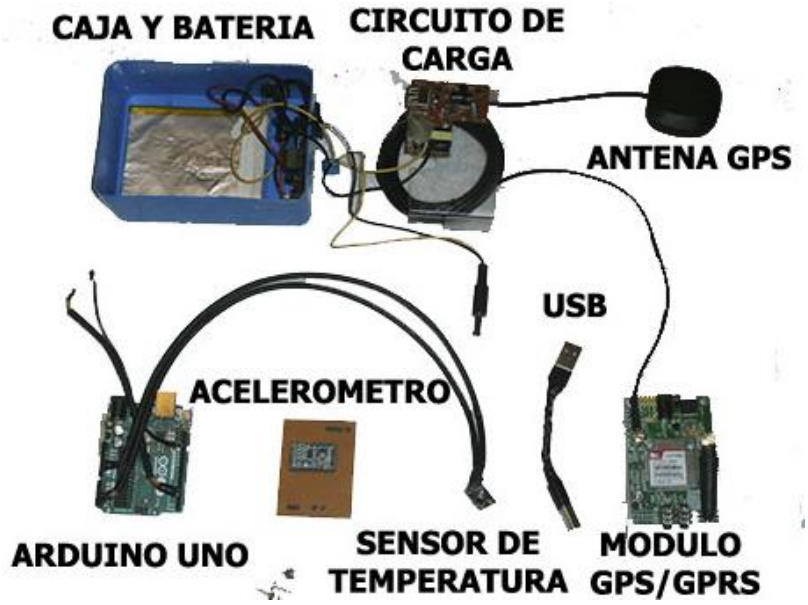


Figura 51 Vista caja abierta

En la Figura 51 se muestra todos los componentes analizados con la caja la cual contiene a todos los componentes mostrados, y los cuales se encuentran afuera para una mayor visualización y entendimiento. En la Figura 52 se muestra la caja cerrada la cual es dispuesta en el arnés para mascotas.

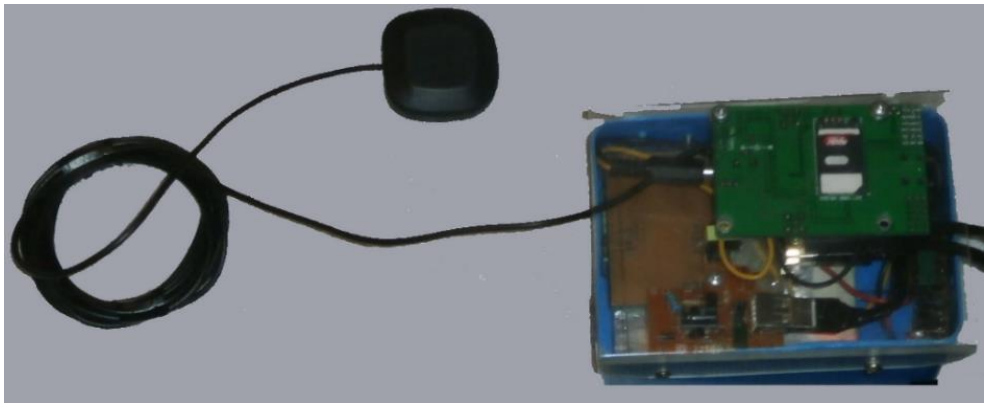


Figura 52 Vista caja cerrada

Una vez analizada y establecida la implementación de hardware se procede con el diseño e implementación del sistema de software.

3.3. Diseño de Software.

El sistema despliega la ubicación y los signos vitales de la mascota a un usuario afiliado a través de un dispositivo móvil o una interfaz web. La funcionalidad del sistema es la siguiente: registrarse mediante un método electrónico, adquirir el hardware, el cual corresponde al arnés que va ubicado en el animal y se encarga de enviar la información de ubicación y signos vitales para su almacenamiento en la base de datos. Una vez que el sistema tiene constancia del usuario registrado y el arnés. El usuario debe iniciar sesión y verificar la información de signos vitales y ubicación correspondiente a la mascota a través de un terminal. El sistema accede a la siguiente información del arnés y el usuario:

- Nombre
- Usuario
- Password
- Correo
- Nombre de la mascota
- Ubicación de la mascota
- Signos vitales de la mascota.
- Id del arnés

El usuario tiene acceso a la siguiente información almacenada por el sistema:

- Nombre de la mascota
- Ubicación
- Signos Vitales.
- Id del arnés.

Considerando la información de acceso, el sistema se diseñó a nivel de hardware y software para llevar a cabo los siguientes procesos:

- Consulta y manipulación de la información añadida por los usuarios.
- Brindar la facilidad a los usuarios de registrarse, iniciar sesión, añadir las mascotas que desee dentro del perfil mediante el id del arnés, ver ubicación y signos vitales de la mascota.

- El envío de información desde y hacia el sistema, desde el dispositivo de hardware del animal hacia el sistema, y del sistema hacia el dispositivo móvil o terminal web manipulada por el usuario.

Al tener conocimiento previo de la información a manejar, se diseñó una estructura de datos que permite representar e interactuar con la información que fluye a través del sistema. El diseño de software está conformado por las herramientas de programación utilizadas para la elaboración de la aplicación móvil y la interfaz web a nivel de usuario, el servidor web y la base de datos a nivel del sistema. Para la elaboración del aplicativo se utilizó diversos lenguajes de programación, los cuales fueron necesarios para el diseño de los diferentes componentes de software. Para el desarrollo de la aplicación en Android se usó Java, para el diseño del servidor web se usó PHP, JavaScript, HTML y CSS. El usuario puede acceder a la información de ubicación y signos vitales de su mascota a partir de dos plataformas, una aplicación instalada en un teléfono móvil con sistema operativo Android y una interfaz web a través de un navegador.

El diagrama de casos de uso (Ver Figura 53), representa la manera de cómo un Usuario (Actor) opera con el sistema desarrollado, además de la forma, tipo y orden en como los elementos interactúan. En la Figura 53 se muestra el diagrama de Casos de Uso correspondiente al sistema general del aplicativo.

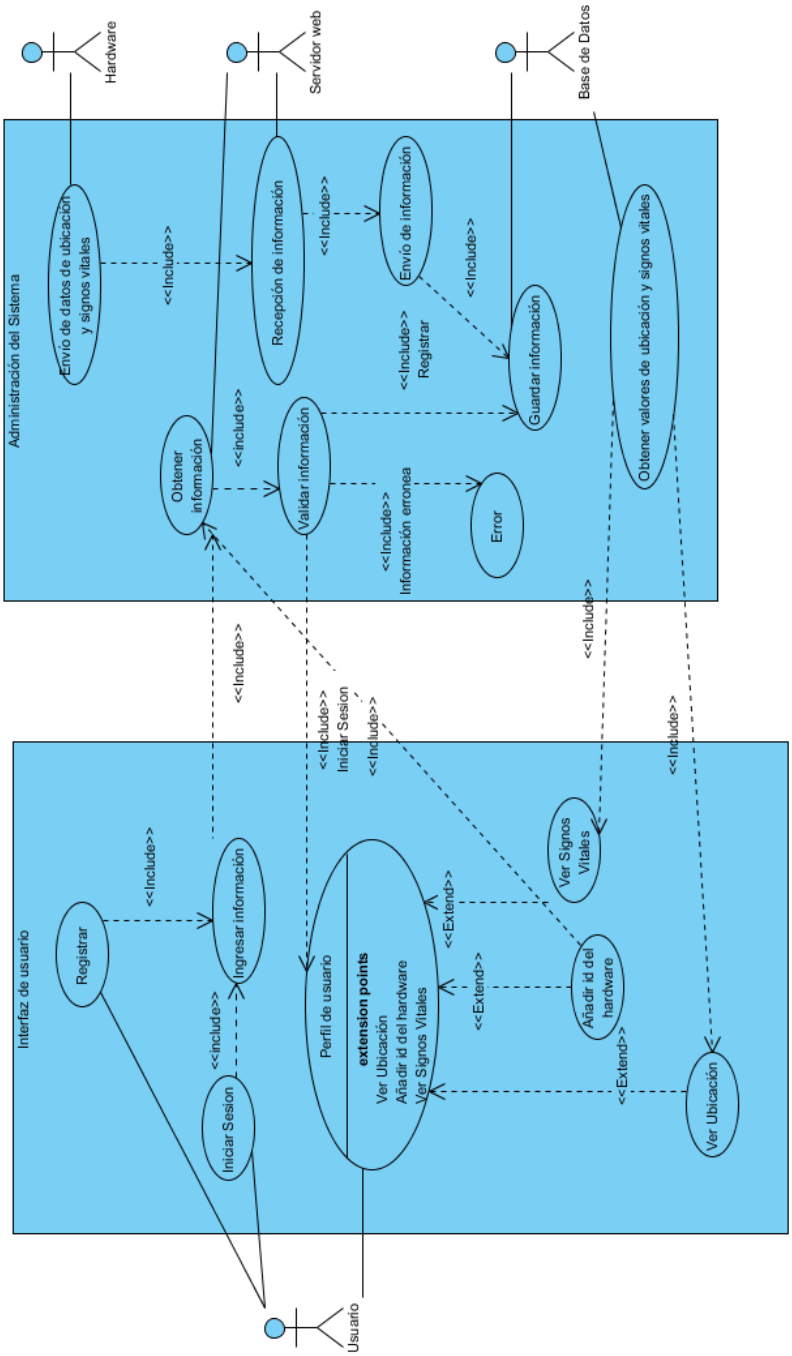


Figura 53 Diagrama de Caso de Uso general del sistema

El Diagrama de Casos de Uso de la Figura 53consta de dos ambientes, el sistema de la interfaz de usuario, el cual representa los terminales a los que el usuario final tiene acceso, como fuere la aplicación Android o la interfaz web; y el ambiente de la

administración del sistema, lo que corresponde a la adquisición, procesado y almacenamiento de la información enviada o requerida por el usuario.

El actor “Usuario” puede realizar las acciones: Registrar, Iniciar Sesión o Ingresar al Perfil de Usuario en donde accede a la opciones de servicio que el sistema ofrece (Ver Figura 53), dichas acciones se encuentran en el sistema “Interfaz de Usuario”. El usuario interactúa con ellas de forma directa, y éstas conllevan a nuevas acciones intermedias. La primera acción a la cual el usuario accede es “Registrar”, dicha acción no depende de otras, sólo del actor “Usuario” (Ver Figura 53). La acción “Registrar” conlleva a la acción “Ingresar Información” (Ver Figura 53), en la cual el actor “Usuario” llena la información que el terminal solicita para un nuevo registro. Al ingresar la información, ésta, es enviada al ambiente “Administración del Sistema” (Ver Figura **53**), la cual consta de tres actores, el Servidor Web, la base de datos y el sistema de hardware. El “Servidor Web” representa el servicio que recibe, valida y envía la información ingresada por el usuario, el actor “Base de Datos” representa el servicio de almacenamiento de la información enviada por el Servidor Web (Ver Figura **53**), el actor “hardware” representa el sistema de hardware es decir el arnés ubicado en la mascota, que envía la información de ubicación y signos vitales para su almacenamiento en la base de datos mediante el “Servidor web” para la posterior consulta de la información por medio de los terminales de usuario, La acción “Validar Información”, puede entregar una respuesta errónea, lo que lleva a la acción “Error” (Ver Figura **53**), e indica un error en los datos insertados por el actor “Usuario", si los datos ingresados son correctos, estos se almacenan en la base de datos, realizando un registro exitoso.

Al igual que la acción “Registrar”, el actor “Usuario” puede acceder a la acción “Iniciar Sesión” (Ver Figura 53). Se debe estar registrado previamente en el sistema para tener un inicio de sesión exitoso. Si el “Usuario” elije la opción “Iniciar Sesión” (Ver Figura **53**), el sistema requiere el ingreso de información mediante la acción “Ingresar Información”, al insertar los datos solicitados por el sistema, dicha información es enviada a la “Administración del Sistema”, la cual por medio del “Servidor web”, se encarga de recibir la información y validarla. Si la información ingresada por el actor “Usuario” es correcta la “Administración del Sistema”

direcciona al usuario a la acción “Perfil de Usuario” en la cual el actor “Usuario” dispone de las funciones que el aplicativo ofrece. Las acciones son: “Añadir ID del hardware”, “Ver Ubicación” o “Ver Signos Vitales” (Ver Figura **53**), dichas acciones se analizan en un diagrama de casos de uso independiente. Si la información ingresada por el actor “Usuario” es incorrecta o no se encuentra registrada en la base de datos, el sistema envía un error mediante la acción “Error” (Ver Figura **53**). Las acciones “Ver Ubicación” y “Ver Signos Vitales” interactúan con la información enviada por el actor “hardware” y almacenada en la base de datos a través del “servidor web”.

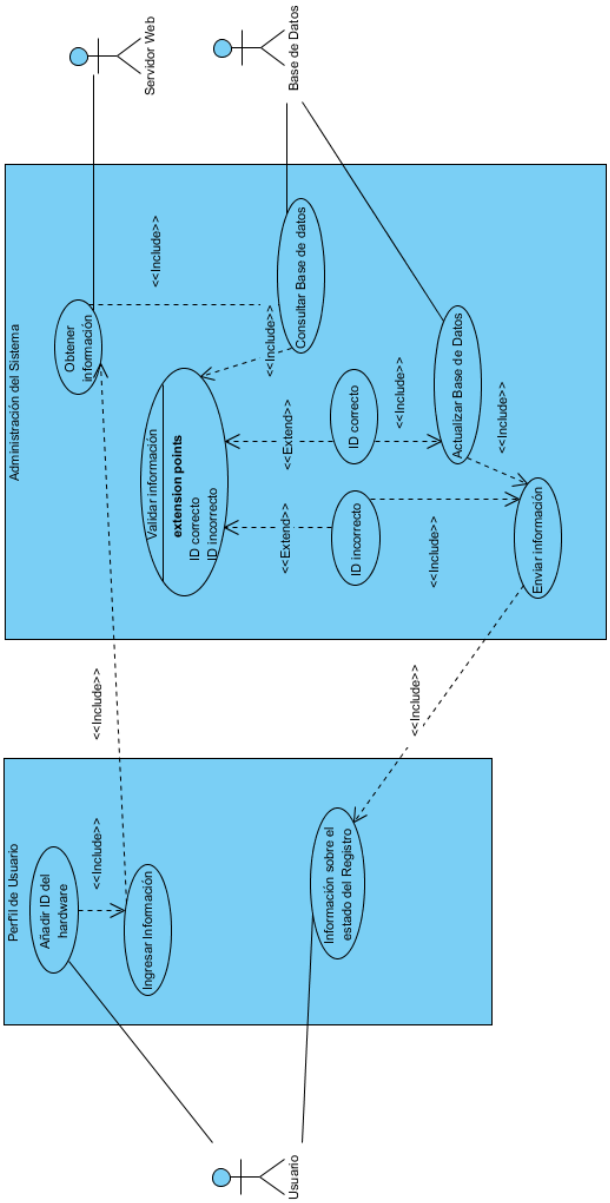


Figura 54 Diagrama de Caso de Uso “Añadir ID"

Al realizar un correcto inicio de sesión, el usuario puede acceder a varias acciones dentro de su perfil de usuario. La acción que registra el nombre de la mascota en el sistema es “Añadir ID del Hardware”, es importante para que los usuarios puedan identificar a su mascota, en especial cuando una persona tiene varios arneses, en distintos animales. El funcionamiento de la acción “Añadir ID del hardware” se representa en la Figura 54, tiene relación directa con el actor “Usuario”, al ingresar al aplicativo, dicha acción permite al usuario registrar el nombre o identificación de la mascota asociada al arnés. El sistema requiere el ingreso de información por parte del usuario mediante la acción “Ingresar Información” (Ver Figura 54), dicha información corresponde al ID del arnés y al nombre de la mascota. Los datos ingresados son enviados a la “Administración del Sistema”, la acción “Obtener Información” (Ver Figura 54) recepta los datos ingresados por el usuario, acto seguido el sistema realiza una consulta a la base de datos por medio de la acción “Consultar Base de Datos” (Ver Figura 54), la acción “Validar Información”, mediante la información consultada en la base de datos valida si la información correspondiente al identificador del arnés ingresada por el usuario es correcta, si lo es, el sistema actualiza la tabla en la base de datos con el nombre de la mascota ingresada por el usuario, e indica el estado del registro al usuario final mediante la acción “Información sobre el estado del registro” (Ver Figura 54).

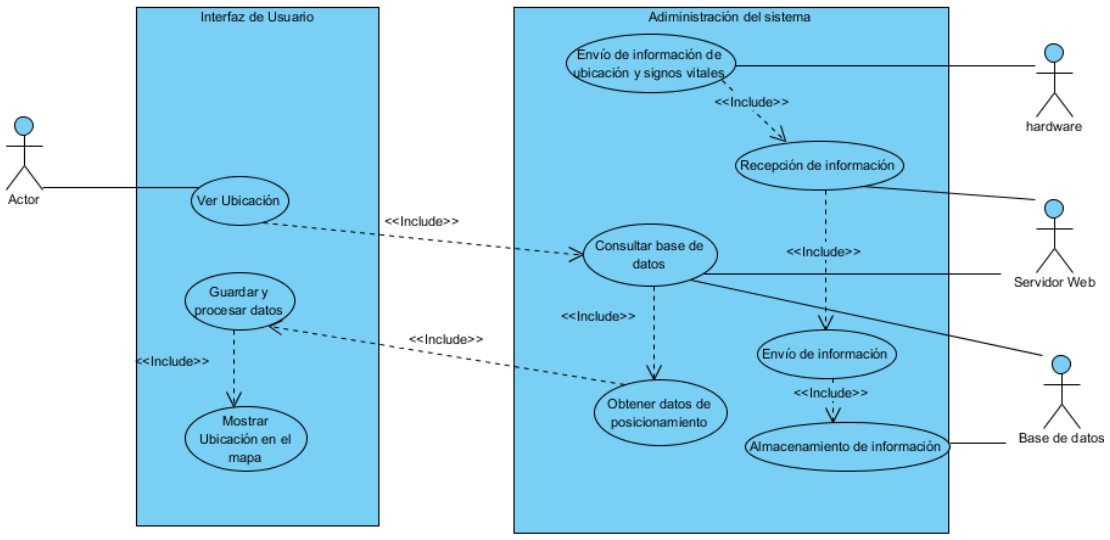


Figura 55 Diagrama de Caso de Uso “Ver Ubicación”

En la Figura 55 se muestra el diagrama de casos de uso que corresponde a la acción de verificación de ubicación, dicha acción interactúa de forma directa con el actor “Usuario”. Al acceder a la opción “Ver Ubicación” en la “Interfaz de Usuario” (Ver Figura 55), se envía una sentencia a la “Administración del Sistema”, la cual realiza una consulta a la base de datos. Mediante la consulta realizada se obtienen los valores de posicionamiento geográficos almacenados en la base de datos, latitud y longitud, dicha información es enviada nuevamente a la “Interfaz de Usuario” y el aplicativo procesa la información mediante la acción “Guardar y procesar datos” (Ver Figura 55), dicha acción se encarga de obtener los datos consultados por el script de configuración alojado en el Servidor Web, para procesar y ubicar la información en el mapa. De forma paralela el hardware, es decir el arnés ubicado en la mascota debe estar encendido, porque es el encargado del envío de la ubicación y signos vitales en tiempo real al servidor web para su posterior almacenamiento en la base de datos.

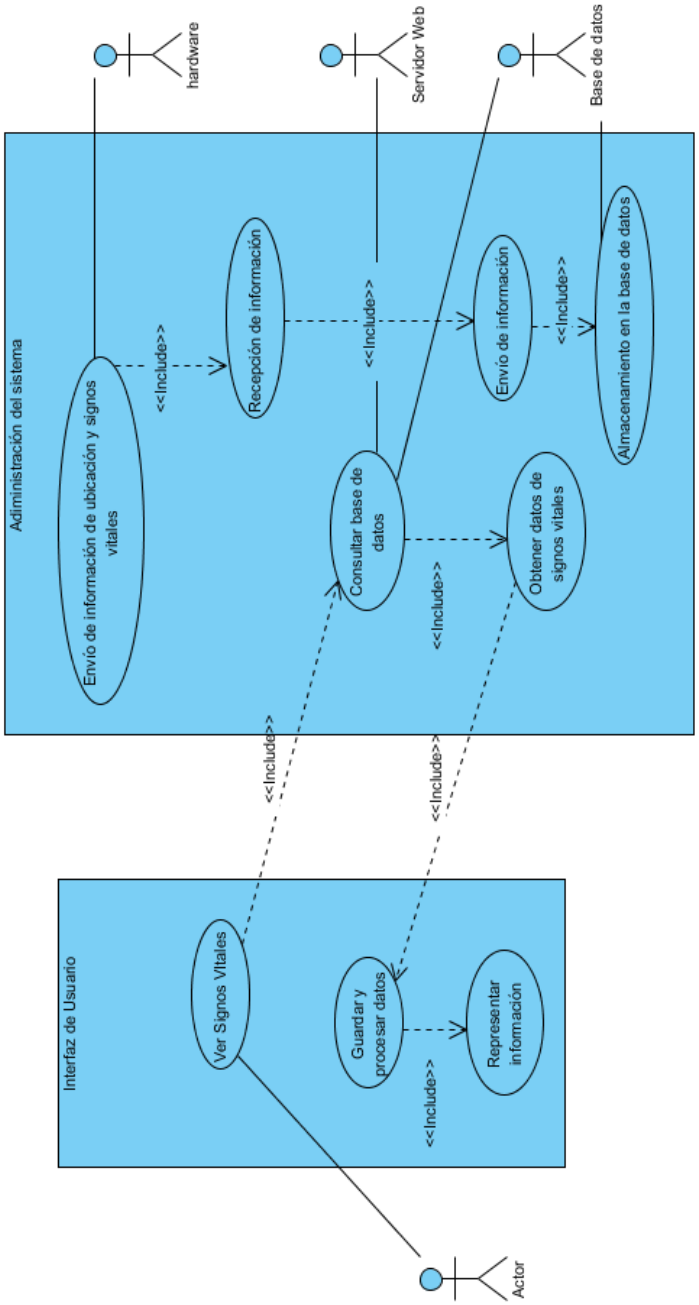


Figura 56 Diagrama de Caso de Uso "Ver Signos Vitales"

En la Figura 56 se muestra el diagrama de casos de uso que corresponde a la acción de verificación de signos vitales, el procedimiento que realiza la tarea es similar a la acción "Ver Ubicación". Al acceder a la opción "Ver Signos Vitales" en la "Interfaz de Usuario" (Ver Figura 56), se envía una sentencia a la "Administración del Sistema", la cual realiza una consulta a la base de datos, con la cual se obtienen los valores de signos vitales almacenados en la tabla, dicha información es enviada

nuevamente a la “Interfaz de Usuario” y el aplicativo procesa la información mediante la acción “Guardar y procesar datos” (Ver Figura 56), la acción se encarga de obtener datos consultados por el script de configuración alojado en el Servidor Web, para procesar y mostrar la información procesada al usuario final mediante la acción “Representar Información” (Ver Figura 56). Es importante tener en cuenta que el sistema de hardware debe estar encendido para obtener valores a tiempo real, el proceso que realiza el sistema de hardware se representa por medio del actor “hardware” (Ver Figura 56).

3.3.1. Representación de interacción de los componentes del sistema de software

Mediante el análisis de los diagramas de casos de uso, se establece los componentes que el sistema requiere para su funcionamiento a nivel general:

- Interfaz de usuario, corresponde a la interfaz web y a la aplicación en Android en la que el usuario final tiene acceso al aplicativo.
- Sistema de hardware, arnés ubicado en la mascota, el cual obtiene los datos de posicionamiento global GPS e información obtenida por los sensores de signos vitales.
- Web Hosting, conformado por el Servidor Web, contiene los scripts de configuración PHP encargados de recibir y procesar la información enviada tanto por el sistema de hardware como por las interfaces de usuario, además almacena las archivos de configuración correspondientes a la interfaz web. El Web Hosting también integra el servidor de Base de datos, para almacenar la información. El alojamiento web se lo realiza en “www.freewebhostingarea.com”, para un mayor detalle acerca del hosting. (Ver Anexo 4).

El diagrama de la Figura 57 representa un diagrama general de los componentes del sistema a nivel de hardware y software.

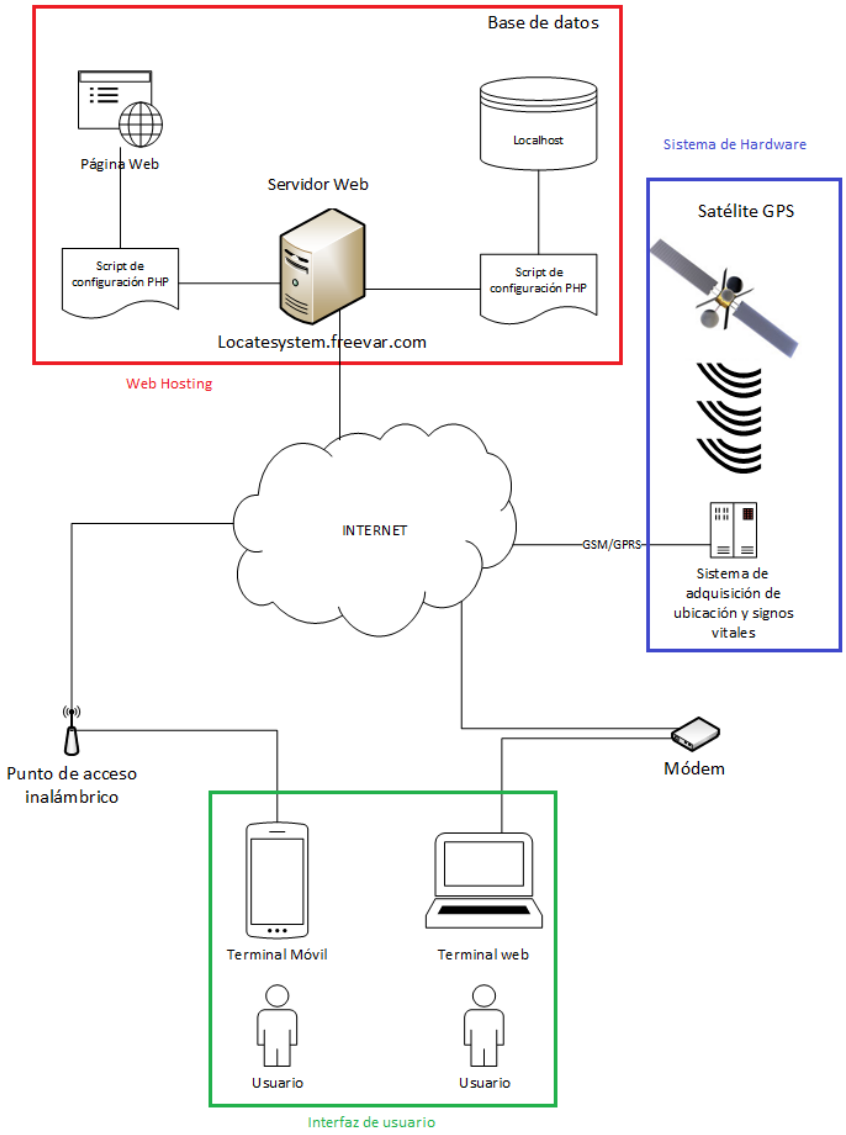


Figura 57 Diagrama representativo de la conexión del sistema

3.3.2. Diseño de Base de datos

El gestor de base de datos se incluye en el hosting web registrado (Ver Anexo 4). El administrador de base de datos incluye "phpMyAdmin" la cual administra la gestión MySQL. El objetivo principal de una base de datos es el almacenamiento de información. El proyecto requiere del almacenamiento de distintos tipos de información manejados por el sistema:

- Almacenamiento de información de los datos obtenidos por el sistema de hardware (posición geográfica y signos vitales).

- Almacenamiento de la información ingresada en el sistema de software correspondiente a los usuarios registrados en el sistema.
- Almacenamiento del identificador de hardware (Arnés para mascotas).
- Además de una tabla adicional, en donde se almacena la validación de la información de signos vitales adquiridos por el sistema de hardware.

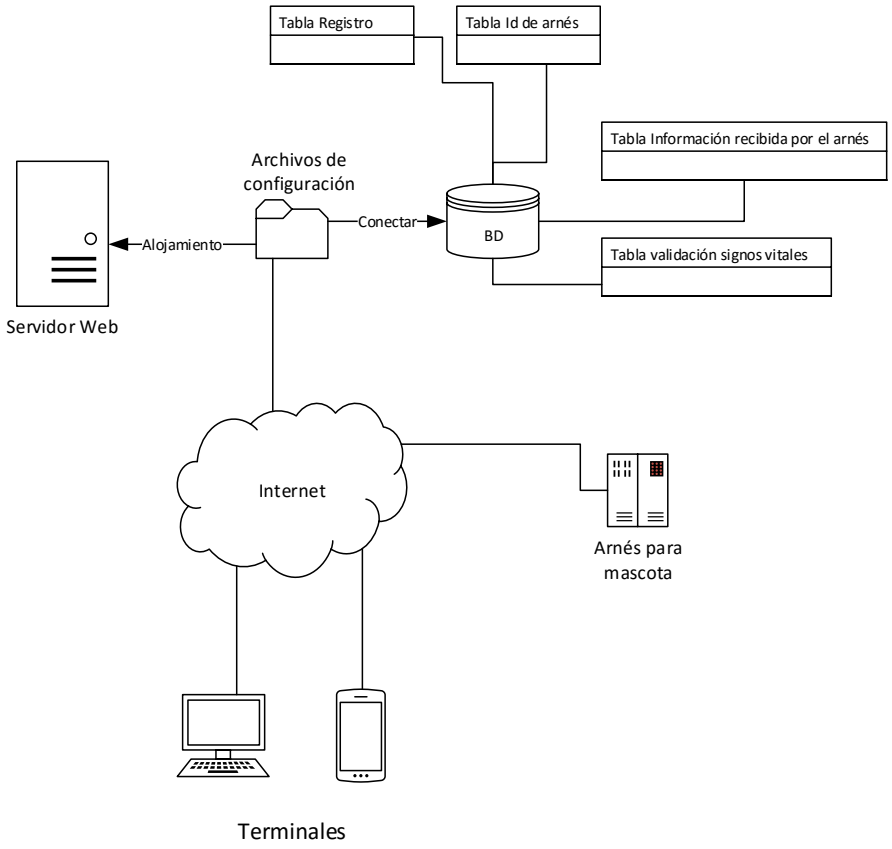


Figura 58 Diagrama representativo del sistema de base de datos

Una vez analizada la información a almacenar se establece las entidades que se integran en el sistema de base de datos, la Figura 59 representa por medio de un rectángulo las entidades establecidas para la base de datos.

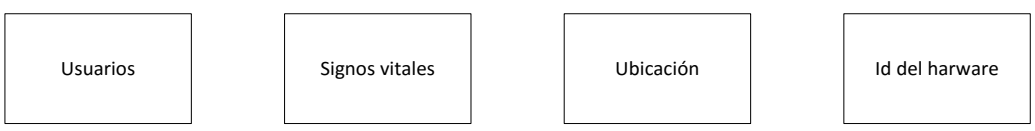


Figura 59 Representación de entidades de la base de datos

Los atributos definen o identifican las características de entidad (es el contenido de esta entidad). Cada entidad contiene distintos atributos, que proveen información sobre la misma. Estos atributos pueden ser de distintos tipos (numéricos, texto, fecha...). La Figura 60 representa las entidades con sus atributos correspondientes.

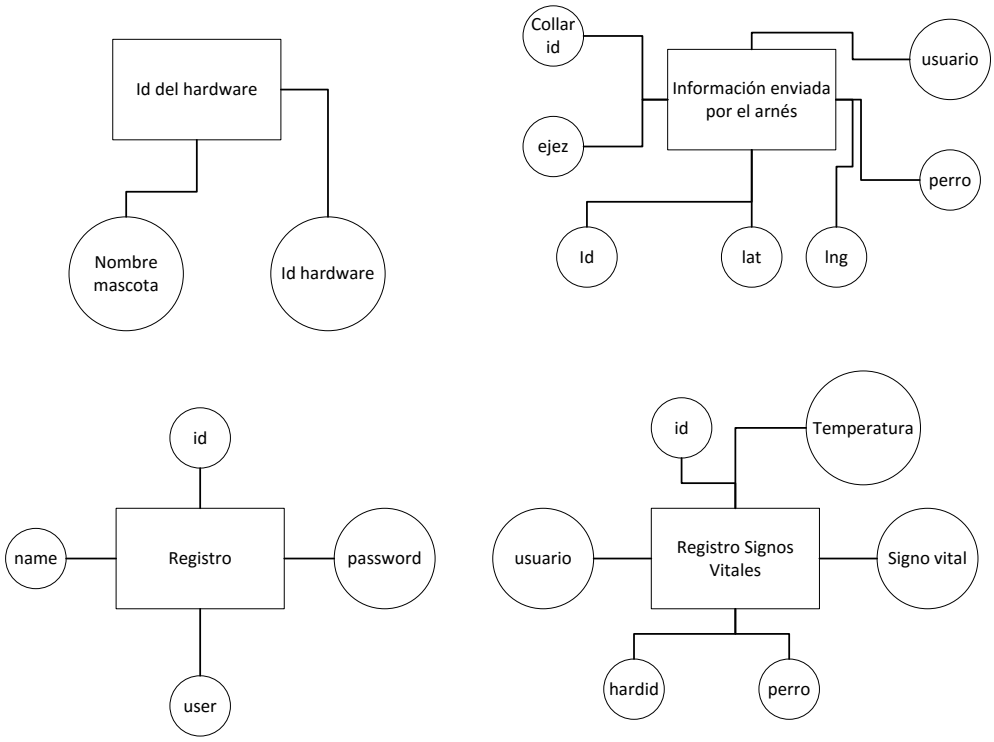


Figura 60 Representación de entidades y atributos de la base de datos

Los atributos son representados mediante circunferencias (Ver Figura 60). Con la información analizada, el sistema requiere de la creación de cuatro tablas, cada tabla contiene diversos atributos, los cuales representan con su definición y componentes.

3.3.2.1. Tabla de almacenamiento de información enviada por el sistema de hardware (Tabla reg)

La tabla "reg", almacena la información procesada y enviada por el sistema de hardware (arnés para mascota). La información a almacenar por la tabla es:

- Id: Identificador de la fila, corresponde a un valor numérico único y auto incrementable.

- Lat: Almacena la información de la variable de posicionamiento geográfico latitud.
- Lng: Almacena la información de la variable de posicionamiento geográfico longitud.
- Usuario: Representa el nombre de usuario con que la persona se registró.
- Ejez: Almacena el valor de vibración en el eje z adquirido por el sensor acelerómetro MMA7361.
- Perro: Representa el nombre de la mascota correspondiente al arnés.
- Collarid: Representa el identificador de 6 digitos único para cada arnés disponible.

3.3.2.2. Tabla de usuarios registrados en el sistema (Tabla registroandroid)

La tabla “registroandroid” almacena la información ingresada por un usuario para registrarse por medio de un método electrónico en el sistema. Posee la información de los usuarios registrados, los atributos de la tabla son:

- Id: Identificador de la fila, corresponde a un valor numérico único y auto incrementable.
- Name: Nombre del usuario registrado en el sistema.
- User: Nombre de usuario del usuario registrado en el sistema, representa un identificador único del usuario registrado, y se lo utiliza para el inicio de sesión en el mismo.
- Password: Contraseña de acceso al sistema, su importancia es guardar integridad de los datos almacenados en el perfil de usuario, y brindar seguridad de acceso de un usuario a la aplicación.
- Email: Correo electrónico del usuario, importante para conocer un medio de comunicación por parte del administrador con el usuario.

3.3.2.3. Tabla de almacenamiento de id del hardware (collarid)

La tabla “collarid” guarda los identificadores del arnés para mascotas, el identificador es un número definido por el administrador del sistema, y el cual se encuentra almacenado previamente en la tabla, los atributos incluidos en la tabla son:

- Id: Identificador del arnés, corresponde a un número de 6 dígitos establecido por el administrador, es único y usado para agregar el nombre de la mascota en el sistema, para su posterior identificación.
- Perro: Corresponde al nombre de la mascota agregada por el usuario, es importante para identificar al animal por medio de su nombre, ya sea en los signos vitales o en la ubicación en el mapa.

3.3.2.4. Tabla de almacenamiento de validación de signos vitales (Tabla signos)

El valor representativo de los signos vitales de la mascota, no corresponde a una unidad medible, sino al movimiento del pecho del animal, por lo que un solo valor no dará una respuesta acertada con respecto a que si la mascota se encuentra viva, por lo que en la tabla “signos” se almacena una validación de 10 muestras enviadas por el hardware, se realizará el análisis de la configuración en los siguiente apartados. Los atributos asociados a la tabla son:

- Id: Identificador de la fila, corresponde a un valor numérico único y auto incrementable.
- Usuario: Nombre de usuario de la persona registrada.
- Signo vital: Resultado de la validación de signos vitales, valor importante para la validación de existencia de movimiento de la mascota.
- Hardid: Identificador único del arnés para mascotas.
- Perro: Nombre de la mascota perteneciente al usuario registrado.
- Temp: Temperatura externa del animal.

La Figura 61 muestra el diagrama lógico de las tablas de la base de datos, la base de datos es usada en su totalidad para almacenamiento y consultas, a pesar de que las tablas incorporadas contienen atributos iguales, su funcionamiento y consulta por parte del aplicativo, no hay una relación entre las tablas, ya que cada una se usa para una operación especifica.

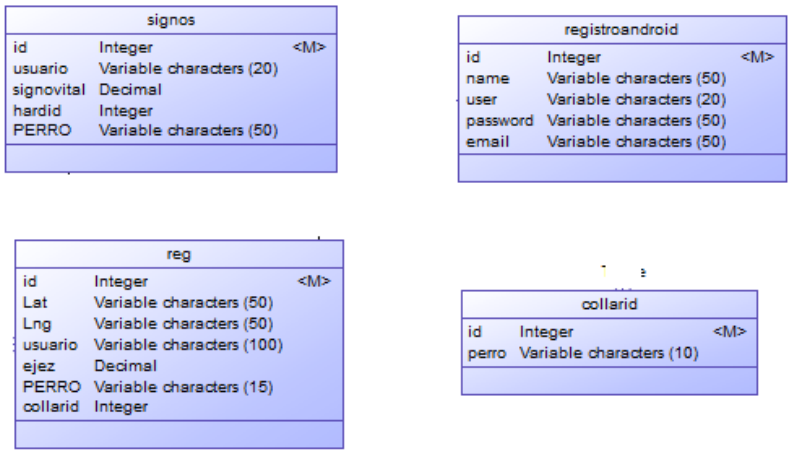


Figura 61 Diagrama lógico de base de datos

3.3.3. Diseño e implementación de servicios para el sistema de hardware.

3.3.3.1. Diseño de servicios para el sistema de hardware

El sistema de hardware ubicado en el arnés para mascotas se encarga de obtener, procesar y enviar la información. La información enviada por el sistema es recibida por un script de configuración PHP, almacenado en el servidor web en la siguiente dirección:

“http://locatesystem.freevar.com/cargar2.php”

El diagrama de la Figura 62 se representa la conexión general relacionado con el sistema de hardware implementado en el arnés para mascotas. Por medio del script de configuración PHP mediante el método HTTP “GET” recibe la información enviada por el sistema de hardware usando el mismo método.

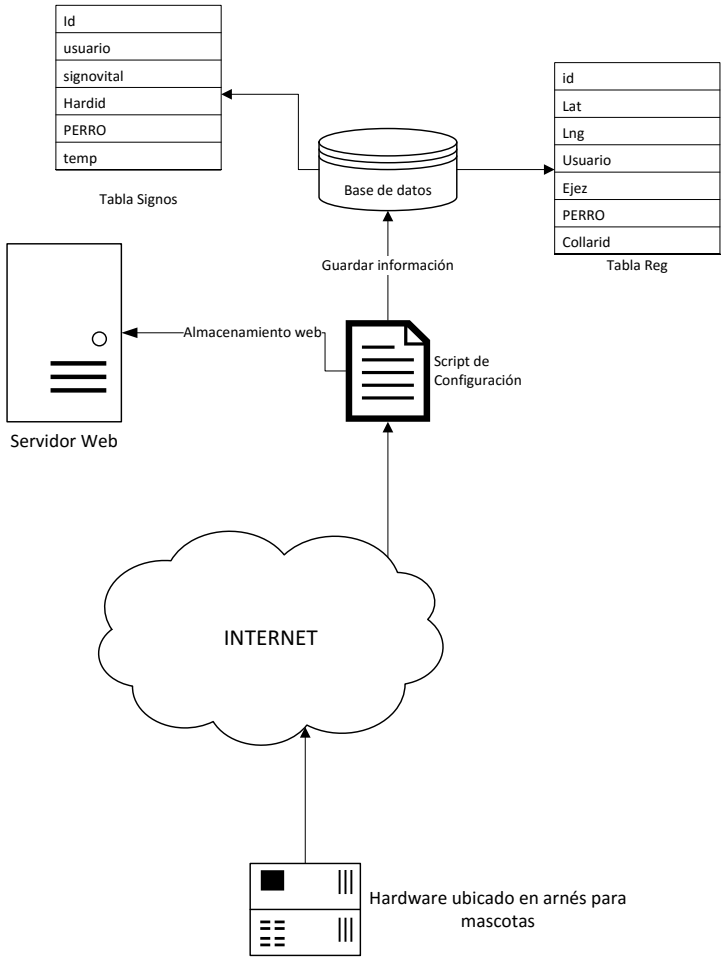


Figura 62 Diagrama general de diseño de servicios para el hardware del sistema

El sistema de hardware (Arnés para mascotas) envía los siguientes parámetros por medio del método GET hacia el archivo de configuración “cargar2.php” alojado en el servidor web:

- Latitud
- Longitud
- Usuario
- Eje z
- Id del arnés
- Temperatura.

Las tareas de funcionamiento del sistema de hardware, se encuentra representado en el diagrama de flujo de la Figura 63.

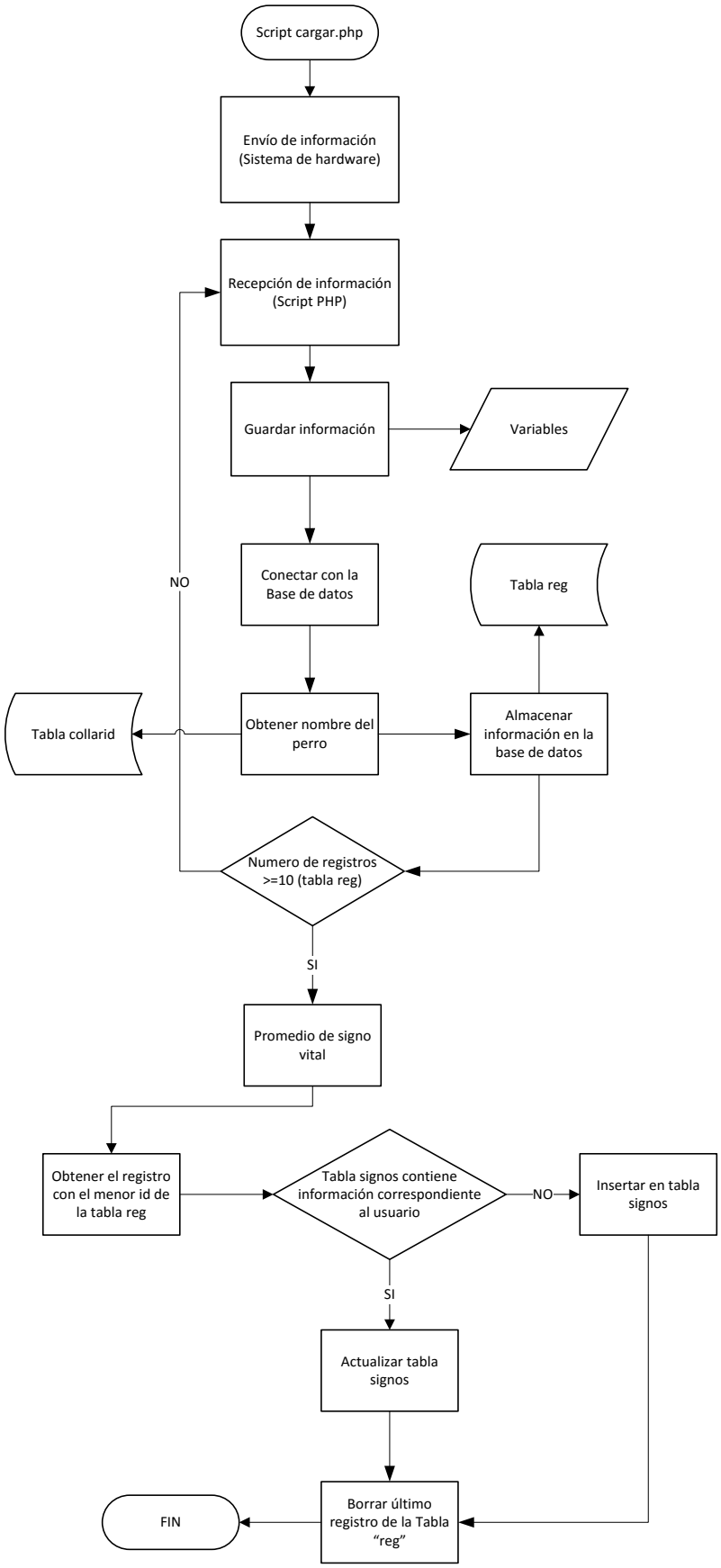


Figura 63 Diagrama de flujo del script de configuración "cargar2.php"

El script de configuración “cargar2.php” está representado en la Figura 63, en primera instancia el sistema de hardware envía los parámetros obtenidos y procesados hacia el script de configuración alojado en el servidor web. El script por medio del método HTTP “GET” recibe la información y la almacena en variables. Dicha información es almacenada en la tabla “reg” de la base de datos. El nombre de la mascota es añadida por el usuario mediante el terminal, el cual es almacenado en la tabla “collarid”, el script recibe el nombre de la mascota a través de la tabla mencionada e inserta los valores recibidos en la tabla “reg”, la cual va a almacenar la información relacionada con la información enviada por el arnés y en la que los usuarios finales acceden para verificar la información de ubicación y signos vitales proporcionada por el arnés.

La validación de los signos vitales se la realiza por medio de los valores de movimiento enviado por el arnés, se realiza la resta del mayor registro de signos vitales de la tabla “reg” con el menor valor, ya que el sensor acelerómetro con un movimiento mínimo recepta una variación importante debido a la sensibilidad del sensor, la resta de las dos variables obtenidas a partir de la tabla “reg”, debe ser mayor a un número establecido en la etapa de pruebas, para indicar movimiento en la mascota e informar el estado del mismo.

Si es la primera vez que el sistema obtiene y valida la información, se inserta los valores de “usuario”, “signovital”, “hardid”, “temperatura” y “perro” en la tabla “signos” de la base de datos, caso contrario se actualiza dicha tabla. Por último se elimina el registro correspondiente al menor id del registro en la tabla “reg” para evitar el llenado innecesario de la base de datos.

3.3.3.2. Implementación de servicios para el sistema de hardware.

En primera instancia el archivo de configuración “cargar2.php” recibe la información enviada por el arnés ubicado en la mascota mediante el método HTTP “GET” o “POST”, la recepción se la realiza mediante la función PHP “getParameter”.

```
function getParameter($par, $default = null){
```

```
            if (isset($_GET[$par]) && strlen($_GET[$par])) return
            $_GET[$par];
            elseif (isset($_POST[$par]) && strlen($_POST[$par]))
                return $_POST[$par];
            else return $default;
        }
```

El método “getParameter” verifica si existe una variable enviada por el arnés, una vez que el script tiene constancia de la información recibida, almacena la información en variables para su posterior inserción en la tabla “reg” de la base de datos.

```
                $lat = getParameter("latitude");
                $lng = getParameter("longitude");
                $usuario= getParameter("usuario");
                $ejez=getParameter("eje_z");
                $collarid=getParameter("collarid");
                $temp=getParameter("grados");
```

Las variables: “$lat”, “$lng”, “$usuario”, “$ejez” “$collarid” y “$temp”, almacenan la información correspondiente a la latitud y longitud (Posicionamiento geográfico), el nombre del usuario registrado, y correspondiente al dueño del arnés, el valor del signo vital, el cual será procesado por medio de varias muestras, para validar si la mascota se encuentra con signos vitales y si la mascota está en movimiento, el id del arnés, el cual identifica por medio de un número único el arnés de la mascota y la temperatura procesada por el sensor infrarrojo integrado, respectivamente. Una vez almacenada la información se procede con la conexión a la base de datos, para el almacenamiento de la información para la posterior consulta de las terminales (Aplicación Android y Página Web).

```
$conexion = mysql_connect($host_db, $user_db, $pass_db);
```

En dónde “$host_db” almacena la dirección del host, las variables “$user_db” y “$pass_db” el nombre de usuario y contraseña para autenticación en el servicio de

base de datos. Otro valor a almacenar en la tabla “reg” es el nombre de la mascota, el nombre es añadido por el usuario por medio del terminal web o Android, el nombre se almacena en la tabla “collarid”, el script de configuración “cargar2.php”, obtiene el nombre de la mascota y la almacena en la variable “$fila”, verificando si existe algún cambio del nombre de la mascota registrado por el usuario.

```
$registroid = mysql_query("SELECT Perro FROM collarid WHERE id = '$collarid'");
$fila = mysql_fetch_row($registroid);
```

Una vez obtenido el nombre de la mascota a través de la tabla “collarid”, se inserta la información en la tabla “reg”, dicha tabla es consultada por el aplicativo final para obtener la ubicación y la variación de movimiento del animal para su posterior procesamiento e identificar la mascota, la tabla “reg” es el centro en donde se almacena la información.

Para validar si el animal se encuentra con signos vitales, se toman 10 registros del movimiento del sensor acelerómetro, los registros son almacenados en la tabla “reg”, una vez que hay 10 registros, se obtiene el mayor y el menor valor de los registros correspondientes al movimiento del sensor acelerómetro (ejez). Se resta los dos registros, y se verifica la diferencia, si la diferencia es mayor a un número establecido en la fase de pruebas, quiere decir que si existe movimiento de la mascota, caso contrario, el animal se encuentra sin signos vitales, la validación se realiza mediante cualquiera de las dos opciones de verificación de información disponibles para el usuario final. Otra medida de signo vital es la temperatura, a pesar que no se puede obtener la temperatura médica real, ya que la única forma precisa de obtenerla es mediante una medición vía rectal, el sensor mide la temperatura externa de la mascota, con la que el usuario puede asumir si la mascota se encuentra con vida en determinada situación.

```
////// Obtener valor mayor de movimiento
$rowSQL = mysql_query( "SELECT max( ejez ) AS max, collarid FROM reg where collarid ='$collarid' GROUP BY collarid " );
$row = mysql_fetch_array( $rowSQL );
$mayor = $row['max'];
```

```
////// Obtener valor menor de movimiento
$rowSQL = mysql_query( "SELECT min( ejez ) as min, collarid FROM reg
where collarid ='$collarid' GROUP BY collarid " );

$row = mysql_fetch_array( $rowSQL );
$menor = $row['min'];


////// Resta el mayor menos el menor valor
$promedio = $mayor-$menor;
```

La variable “$promedio” es almacenada en la tabla “signos”, dicha tabla es consultada por los terminales manipulados por el usuario final (aplicación en Android y página web), para la validación de la información almacenada, correspondiente a los signos vitales, también se realiza la inserción de la variable “$temp” correspondiente a la temperatura recuperada por el arnés.

```
mysql_query("insert into signos (usuario,signovital,hardid,PERRO)
values ('$usuario','$promedio','$collarid','$fila[0]')",$conexion);
mysql_query("insert into signos (temp) values ('$temp')",$conexion);
```

Una vez almacenada la información en las diferentes tablas, el sistema elimina el valor del registro con el menor id, del usuario, para que no exista información innecesaria ocupando la base de datos.

```
$rowSQL = mysql_query( "SELECT MIN( Id ) AS id FROM `reg`;" );
$row = mysql_fetch_array($rowSQL);
$id = $row['id'];
mysql_query("DELETE FROM reg where Id = '$id' AND usuario =
'usuario'" ,$conexion);
```

Por medio de la metodología mencionada, se establece el comportamiento en el sistema de software, la recepción, procesado y almacenamiento de información enviado por el sistema embebido en el arnés ubicado en la mascota.

3.3.4. Interfaz Web

La interfaz web, es una de las plataformas disponibles para que el usuario final manipule el aplicativo, el usuario tiene la capacidad de realizar todas las acciones que el sistema pone a su disposición mediante un navegador web. Para la configuración de una página web es importante el registro en un hosting online (Véase Anexo 4), el cual aloja los archivos web a los cuales accede el usuario.

3.3.5. Diseño Web

Para la implementación de la plataforma web se utilizó varios lenguajes de programación como: HTML, PHP, JavaScript y CSS, dichos lenguajes interactúan para dar el aspecto y funcionamiento deseado a la interfaz. La página web dispone de varios contenidos. Según la experiencia previa en navegación web, y teniendo en cuenta las funcionalidades que requiere la aplicación, se dispuso los siguientes elementos principales:

- Inicio
- Quienes Somos
- Contactos
- Registro
- Inicio de Sesión.

Cada componente realiza una función y tiene un objetivo en la página principal, en la Tabla 14 se detalla los distintos componentes con sus respectivas funciones en el terminal web:

Tabla 14

Componentes y funciones iniciales de la página web

Componente	Función
Inicio	Direcciona a la página principal.
Quienes Somos	Información general y objetivo de la página web.
Contactos	Número de teléfono y correos del administrador del proyecto.
Registro	Registro de nuevos usuarios en el sistema.
Iniciar Sesión	Ingreso a los servicios proporcionados por el sistema.

Cada componente direcciona al usuario a la función que requiere ingresar, en la Figura 64 se muestra el diagrama de la arquitectura inicial de la página web.

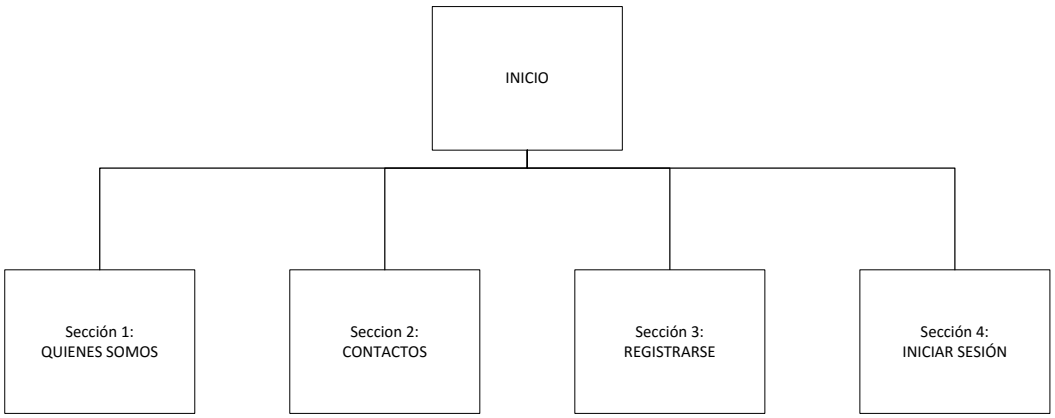


Figura 64 Diagrama de arquitectura inicial de la página web

El componente Inicio es la primera página a la que el usuario ingresa. Todas las secciones de la interfaz web, tienen un diseño común, es decir la distribución de sus componentes son similares, para brindar una experiencia de navegación atractiva

visualmente al usuario. El diseño general del sitio web (Ver Figura 65), consta de los siguientes elementos:

- Título General, identificador del sistema correspondiente al título del proyecto, dicho elemento no cambia en las secciones de la página web.
- Título de Sección, identifica la sección al que el usuario ingresa, la Figura 64 muestra las secciones iniciales disponibles en la página web.
- Menú desplegable, contiene las opciones de las secciones a las que el usuario puede acceder.
- Información/Formularios, contiene la información principal de la página, como por ejemplo: formularios, anuncios e información general.
- Publicidad/Pie de Página, reservado para el ingreso a publicidad en Facebook, Twitter, etc. Además, se integra el nombre del propietario de la página web como pie de página, es un elemento común para todas las secciones.

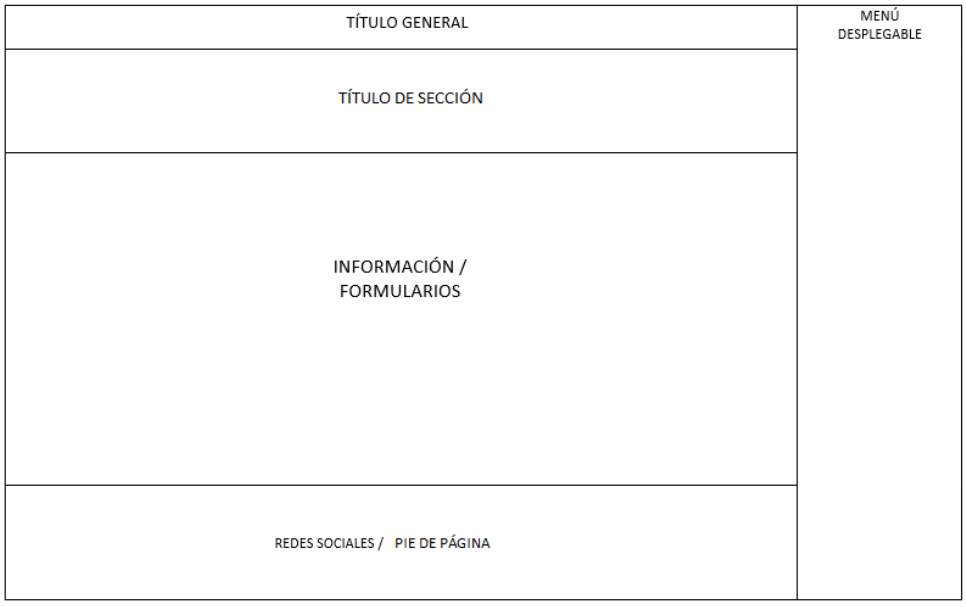


Figura 65 Elementos de la página web

Para que la página web sea visualmente atractiva para el usuario, se utilizó una plantilla, cuya configuración HTML y CSS se encuentra predefinida y únicamente se realizó cambios en las imágenes e información integrada para dar un aspecto de acuerdo al sitio web y a la temática deseada, la Figura 66 muestra el aspecto general

de la página web, con su menú el cual está relacionado con las secciones indicadas en la Figura 64, y a la cual el usuario va a tener acceso ingresando al link: “http://locatesystem.freevar.com/locate/index.html”.

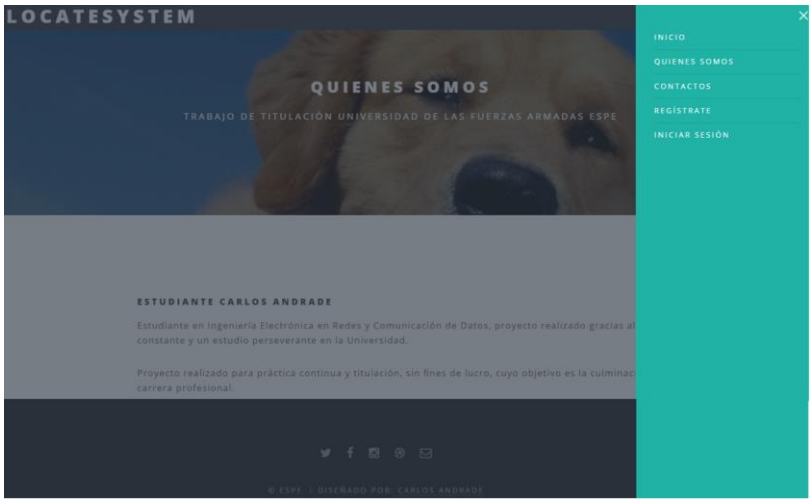


Figura 66 Aspecto visual de la página web sección “Quienes Somos”

Las secciones iniciales (Ver Figura 64) direccionan al usuario hacia los componentes de la página web que van a brindar la función de visualización de Signos Vitales y Ubicación. Estas secciones mediante formularios HTML solicitan al usuario el ingreso de datos para su posterior registro e inicio de sesión respectivamente.

Las secciones 3 y 4 (Ver Figura 64) brindan la funcionalidad principal del sistema, la sección 3 representa la opción de registro en el sistema, al elegir dicha opción en el menú de la página web principal, se despliega un formulario en donde el usuario va a ingresar los datos requeridos por el sistema para su registro.

Figura 67 Diagrama de arquitectura de la Sección 3 Registro

Usando el formato gráfico de la página principal, se diseñó la sección “Registro”. Se dispuso un Formulario HTML para el ingreso de datos por parte del usuario, dicha información será almacenada en una base de datos, además se estableció dos botones: “Registrarme” y “Borrar”. El botón Registrarme envía los datos para guardarlos en la base de datos, mientras que el botón Borrar, borra todo el contenido del Formulario.

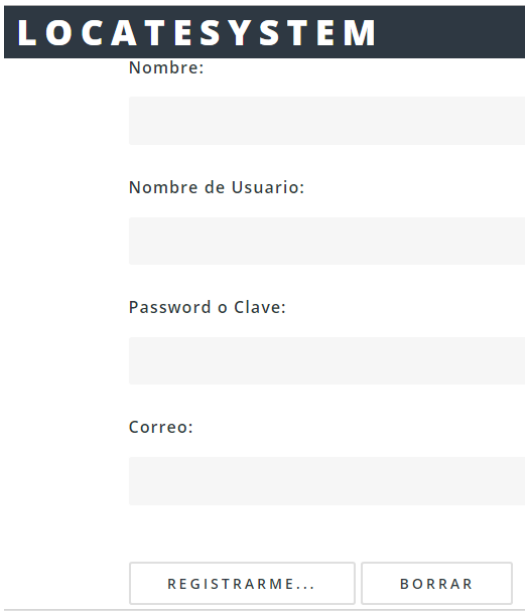


Figura 68 Aspecto Visual de la Sección Registro

Se realiza una validación de datos, para que el Nombre de Usuario y el Correo no sean igual al de un usuario registrado con anterioridad, la validación se realiza mediante el archivo de configuración PHP, “Register.php”, el cual recibe la información ingresada en el formulario para almacenarla en la base de datos y realizar la validación indicada. El diagrama de Flujo de la Figura 69 muestra el funcionamiento general de la sección Registro.

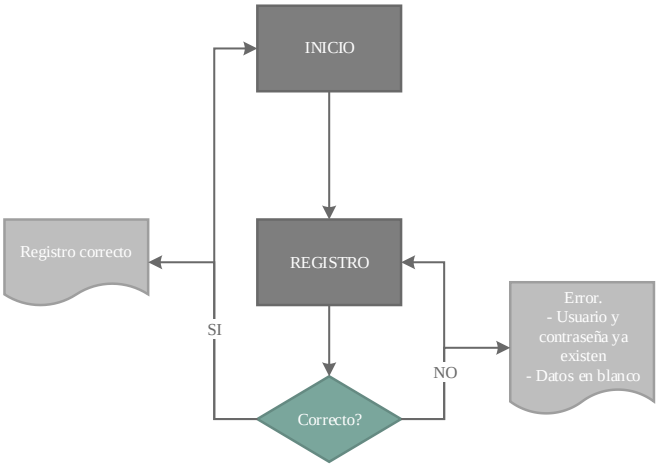


Figura 69 Diagrama de flujo de la Sección Registro

La sección 4 brinda al usuario la funcionalidad de Inicio de Sesión en el sistema (Ver Figura 64), al elegir dicha opción en el menú de la página web principal, se despliega un formulario en donde el usuario va a ingresar los datos requeridos por el sistema para su inicio de sesión, si el inicio de sesión es correcto, se muestra una nueva página en donde el usuario va a poder acceder a cualquiera de las opciones representadas en la Figura 70, cada opción direcciona al usuario a una página distinta.

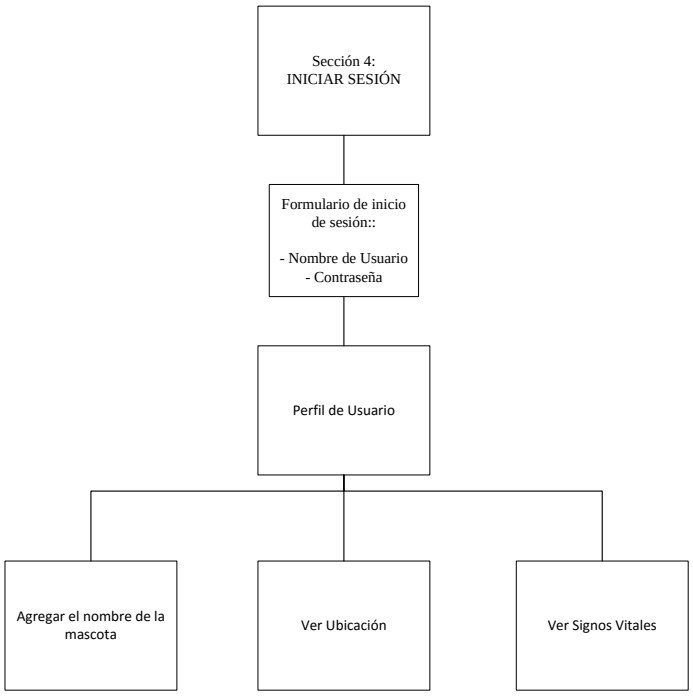


Figura 70 Diagrama de arquitectura de la Sección 4 Inicio de sesión

La Sección 4, Iniciar Sesión mediante un formulario envía los datos de autenticación ingresados por el usuario, y mediante el archivo de configuración PHP "validar_usuario.php", realiza la validación de los datos previamente registrados en la base de datos para el correcto o fallido inicio de sesión en el sistema. El diagrama de Flujo de la Figura 71 muestra el funcionamiento general de la sección Iniciar Sesión.

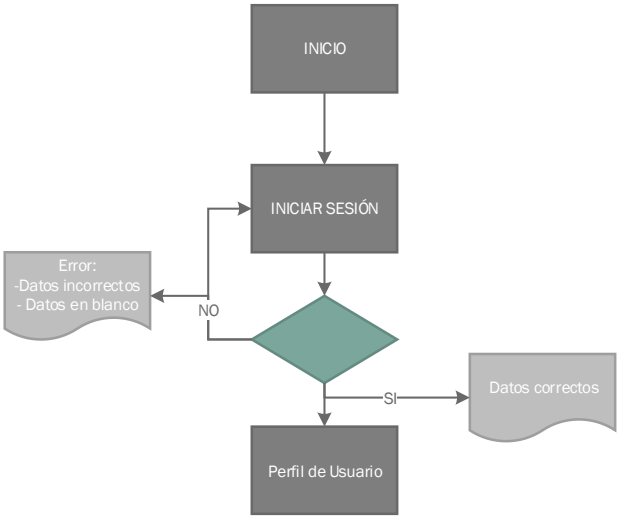


Figura 71 Diagrama de flujo de la Sección Iniciar Sesión

Usando el formato gráfico de la página principal, el diseño de la sección "INICIAR SESIÓN" se muestra mediante la Figura 72. Se dispuso un Formulario HTML para el ingreso de datos por el usuario, dicha información es comparada con la base de datos, además se dispuso un botón: Iniciar Sesión. El cual envía la información para validarla con la base de datos. Si el usuario realiza un inicio de sesión exitoso, se lo direcciona al Perfil de usuario.

Figura 72 Aspecto Visual de la Sección Iniciar Sesión

3.3.5.1. Implementación de servicios para la interfaz web

El punto central del sistema son los servicios configurados en el servidor web, el cual realiza la adquisición de datos, envío de información para su almacenamiento en la base de datos y la disposición de información para su manipulación por los aplicativos. Tanto la aplicación Android como la aplicación web requieren conectarse a los archivos de configuración alojados en el servidor web. El terminal web utiliza los scripts de configuración PHP para realizar las siguientes tareas:

- Registro de usuario.
- Inicio de sesión.
- Obtener datos de ubicación y signos vitales.

La Figura 73 muestra el diagrama representativo del Registro e Inicio de Sesión de un usuario en el sistema a través de la página web, representa los módulos del terminal web correspondiente a inicio de sesión y registro; y los scripts de configuración a los que se conectan dichos módulos para realizar una tarea específica.

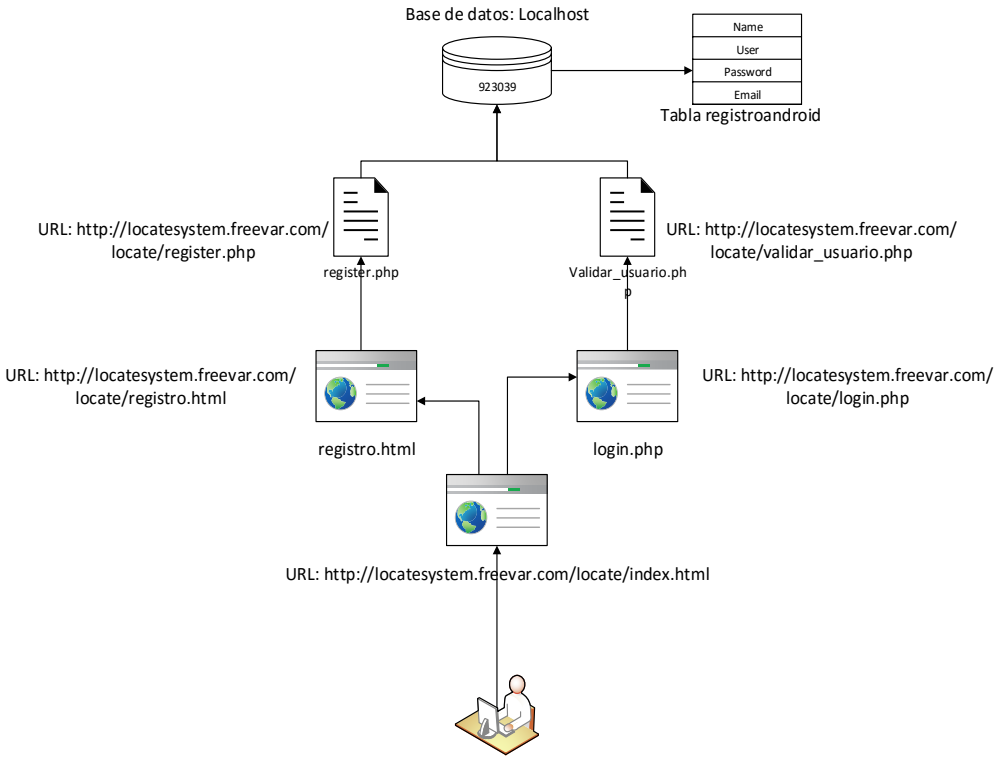


Figura 73 Diagrama representativo de funcionamiento de Inicio de Sesión y registro

En la Figura 73 se representa a un Usuario, el cual por medio de un computador ingresa a la dirección URL de la página principal del proyecto, en donde se encuentra alojado el aplicativo web. Representa gráficamente las opciones de ingreso de un usuario que accede al sistema, con las direcciones URL correspondientes.

3.3.5.2. Implementación de servicio Registro Web

Si un usuario selecciona la opción de Registrarse, se lo direcciona al documento HTML alojado en la dirección: “locatesystem.freevar.com/locate/registro.html” al desplegarse la nueva página web, el sistema solicita el ingreso de los datos para el registro.

Mediante el formulario HTML se solicita al usuario llenar los campos; nombre, nombre de usuario, password y correo electrónico. En la etiqueta “<form>” se establece las acciones que realiza el formulario, el usuario envía los datos mediante

los atributos “action”, y “method”, en los cuales se ingresa la dirección a dónde se envía la información ingresada y el método HTTP con que se envía la información.

```
<form action="register.php" method="post">
        <h3>Crea una cuenta</h3>
       <!--Nombre                                                    -->
       <label for="nombre">Nombre:</label>
       <input type="text" name="nombre" maxlength="16" />
       <br/>

        <!--Nombre Usuario-->
        <label for="nombre">Nombre de Usuario:</label>
        <input type="text" name="nombreUsuario" maxlength="16" />
        <br/>

         <!--Password-->
         <label for="pass">Password o Clave:</label>
         <input type="password" name="password" maxlength="8" />
         <br/>


          <!--Correo-->
          <label for="nombre">Correo:</label>
          <input type="text" name="correo" maxlength="16" />
          <br/><br/>

<input   type="submit"   name="submit"   value="Registrarme..."   />
<input type="reset" name="clear" value="Borrar"/>

</form>
```

Los datos a ingresar se disponen en la etiqueta input de tipo “text” y “password”, y el envío se lo realiza pulsando el botón “Registrarme” de tipo “submit”, con el que se envía la información agregada al script “register.php” el cual es un archivo de configuración en formato PHP y se encarga de recibir, validar y guardar la información ingresada en el formulario. Los procesos que realiza el script de configuración “Register.php” se detallan en la Figura 74.

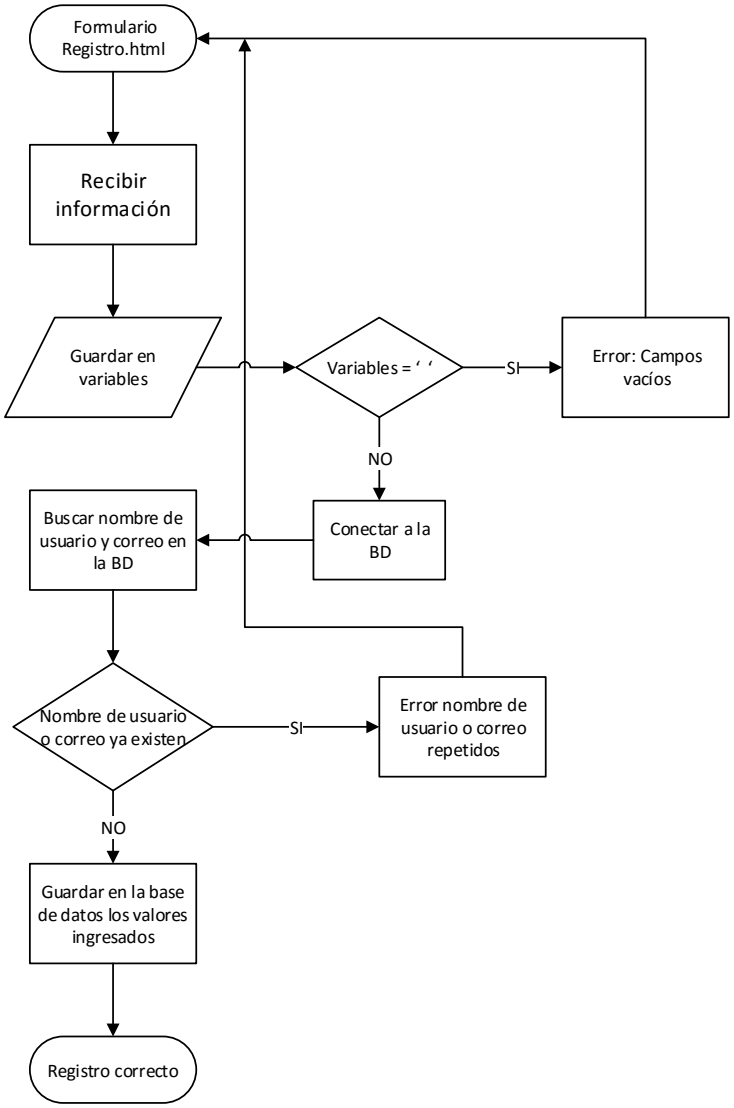


Figura 74 Proceso de script de configuración Register.php

La Figura 74 detalla los pasos que realiza el script de configuración "Register.php", en primera instancia, se ingresa y envía los datos mediante el formulario "Registro.html", el archivo de configuración PHP recibe y guarda la información en variables, mediante el método HTTP "POST" se recibe la información, dicho proceso se lo realiza de la siguiente manera:

```
$name = $_POST['nombre'];
$username = $_POST['nombreUsuario'];
$password = $_POST['password'];
$email = $_POST['correo'];
```

Las variables $name, $username, $password y $email guardan la información enviada por el formulario de registro, dichas variables son validadas para que el usuario no pueda enviar el formulario con valores vacios. La validación se la realiza de la siguiente manera:

```
if($name == '' || $username == '' || $password == '' || $email == ''){
echo "<br />". "Por favor llena todos los campos" . "<br />";

echo "<a href='registro.html'>Intente nuevamente</a>";
exit;
```

Por medio de una sentencia condicional IF, las variables son comparadas con valores vacios, si una variable contiene un valor vacio o ‘ ’ ingresa a la sentencia condicional en donde se direcciona al usuario al formulario de registro para que realice nuevamente el ingreso de información.

Si la condición no se cumple, el script de configuración se conecta con la base de datos, mediante el método mysql_connect de la siguiente manera:

```
$conexion = mysql_connect($host_db, $user_db, $pass_db);
```

En donde “$host_db” es la variable que guarda la dirección de la base de datos. Las variables “$user_db” y “$pass_db” contienen los valores de nombre de usuario y contraseña de conexión a la base de datos respectivamente. Cuando el formulario envía información añadida por un usuario que desea registrarse en el sistema, La validación de la información ingresada se la realiza comparando la información con la tabla “registroandroid” de la base de datos, en la cual se almacenan los usuarios registrados en el sistema.

```
$buscarUsuario = "SELECT * FROM registroandroid WHERE user='$_POST[nombreUsuario]' ";
 $result = mysql_query($buscarUsuario);
 $count = mysql_num_rows($result);
```

```
$buscarCorreo = "SELECT * FROM registroandroid WHERE email =
'$_POST[correo]' ";
$resultcorreo = mysql_query($buscarCorreo);
$countcorreo = mysql_num_rows($resultcorreo);

if ($count == 1||$countcorreo==1){
echo "<br />". "El Nombre de Usuario o correo ya existen!"."<br />";
echo "<a href='registro.html'>Por favor intente nuevamente</a>";
exit; }
```

Por medio de las variables “$buscarUsuario” y “$buscarCorreo” se guarda la información de los nombres de usuario y correos almacenados en la base de datos. Se realiza el conteo de la información recibida y mediante la sentencia condicional IF se verifica si el valor es igual a 1, lo cual indicaría que el nombre de usuario o el correo ya se encuentran registrados en el sistema y direcciona al usuario al formulario “registro.html” para que realice un nuevo intento. Si ninguna de las validaciones se cumplen, el usuario inserta los datos enviados por el formulario “registro.html” en la tabla “registroandroid”.

```
$query = "INSERT INTO registroandroid (name, user, password, email)
VALUES
('$_POST[nombre]','$_POST[nombreUsuario]','$_POST[password]','$_POST
[correo]')";
```

3.3.5.3. Implementación de servicios para Inicio de Sesión Web

Si un usuario se encuentra registrado, puede iniciar sesión para acceder a los servicios que el sistema ofrece. Si selecciona la opción de Iniciar Sesión, se lo direcciona al archivo de configuración PHP alojado en la dirección: “locatesystem.freevar.com/locate/login.php” (Ver **Figura 73**) al desplegarse la nueva página web, mediante un formulario, el sistema solicita el ingreso de los datos necesarios para el inicio de sesión.

Mediante un formulario HTML el sistema solicita al usuario llenar los campos nombre de usuario y contraseña, la etiqueta “<form>” especifica los valores y las acciones que realiza el formulario. El usuario es capaz de enviar información

ingresada mediante los atributos “action”, y “method”, en los cuales se ingresa la dirección en donde esta alojado el archivo de configuración y a dónde va la información ingresada y el método HTTP con que se va a enviar la información, en este caso se utilizó el método POST. A continuación se muestra el formulario “login.php” en formato HTML.

```
<form action="validar_usuario.php" method="post">
<table>
<tr>
<td>Usuario:</td>
<td><input name="admin" required="required" type="text" /></td>
</tr>

<tr>
<td>Password:</td>
<td><input name="password_usuario" required="required"
type="password" /></td>
</tr>

<tr>
<td colspan="2"><input name="iniciar" type="submit" value="Iniciar
Sesion" /></td>
</tr>
</table>
</form>
```

El envío se realiza pulsando el botón “Iniciar Sesión” de tipo “submit”, el cual envía la información al archivo “validar_usuario.php”, es un archivo de configuración en formato PHP que recibe y valida la información ingresada en el formulario. Los procesos que realiza el script de configuración validar_usuario.php se detallan en la Figura 75.

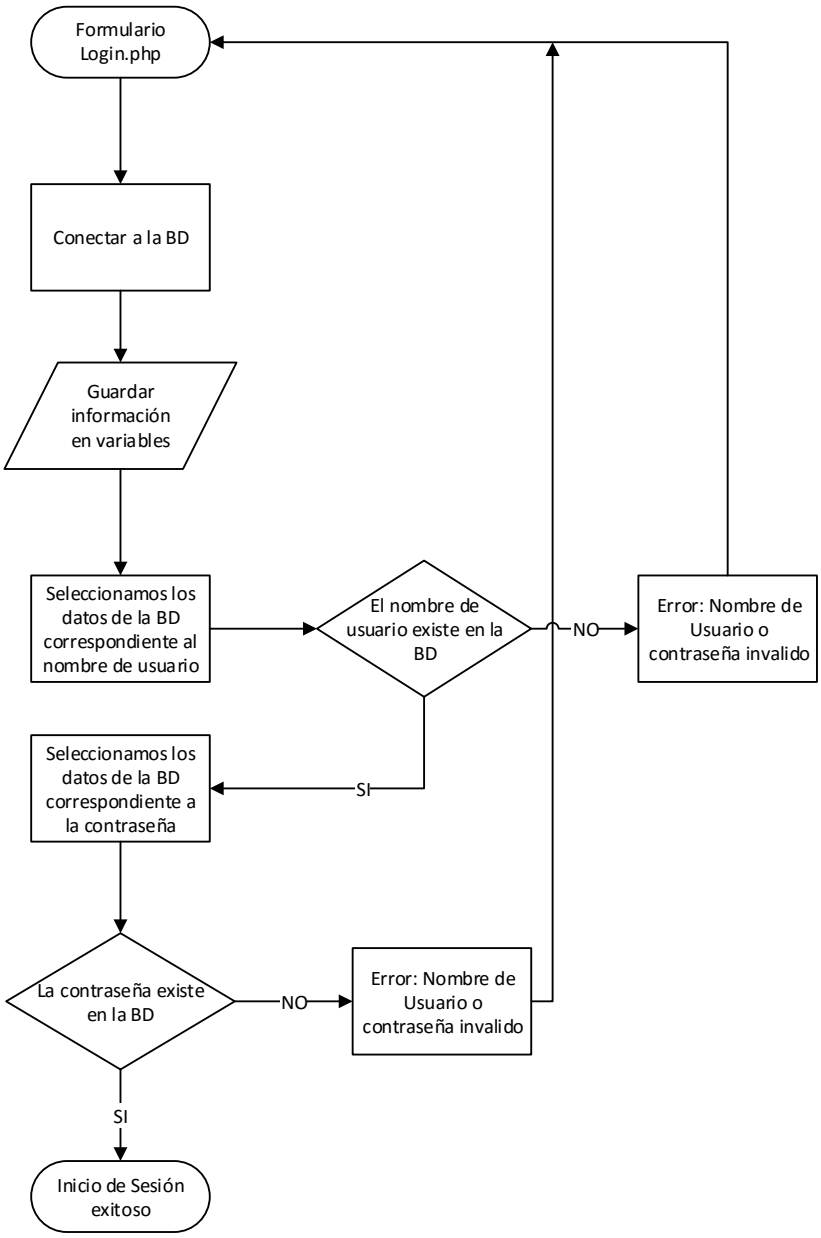


Figura 75 Proceso de script de configuración

La Figura 75 muestra el diagrama general del proceso que realiza el script de configuración “validar_usuario.php”, el cual recibe la información enviada por el formulario “login.php”, y valida dicha información para el correcto o fallido inicio de sesión de un usuario previamente registrado en el sistema. En primera instancia el script se conecta con la base de datos.

```
mysql_connect('Direccion_BD','Usuario','Contrasena')
```

Se captura la información enviada por el formulario “login.php” mediante el método HTTP “POST”, la información obtenida se almacena en las variables “$usuario” y “$password”.

```
$usuario = $_POST["admin"];
$password = $_POST["password_usuario"];
```

Se realiza la consulta a la base de datos, con la que se indica que seleccione de la tabla “registroandroid” solo los campos que tenga como valor de la columna “user” el valor de nombre de usuario recibido del formulario.

```
$result = mysql_query("SELECT * FROM registroandroid WHERE user =
'$usuario'");
```

Mediante el método condicional IF se valida si el nombre de usuario existe en la base de datos.

```
if($row = mysql_fetch_array($result))
```

En la sentencia IF del código mostrado, se integra un nuevo método condicional IF, el cual se ejecuta si el usuario es correcto, y realiza la validación de la contraseña.

```
if($row["password"] == $password)
```

Si el nombre de usuario y la contraseña son correctos, se crea una sesión, en la cual se almacena el nombre de usuario en una variable de sesión “usuario” con la que se realiza la validación de información en los servicios y se direcciona al usuario a la página web “PerfilUsuario.php”, almacenado en la dirección: “http://locatesystem.freevar.com/locate/PerfilUsuario.php”.

```
session_start();
$_SESSION['usuario'] = $usuario;
echo "<script>window.location.href = 'PerfilUsuario.php';</script>";
```

3.3.5.4. Implementación de servicios de “Agregar nombre de mascota”, “Ver Ubicación” y “Ver signos vitales” para la interfaz web.

Si un usuario realiza un inicio de sesión correcto, puede elegir entre los servicios que el sistema ofrece:

- Agregar el nombre de la mascota, ofrece al usuario la opción de agregar el nombre de la mascota mediante el id del hardware, para su identificación en el sistema.
- Ver Ubicación, ofrece al usuario la opción de verificar la posición de la mascota en el mapa.
- Ver signos vitales, ofrece al usuario la opción de verificar si existen signos vitales de la mascota por medio de un navegador web, las medidas que demuestran si la mascota se encuentra con vida son: el movimiento y la temperatura externa.

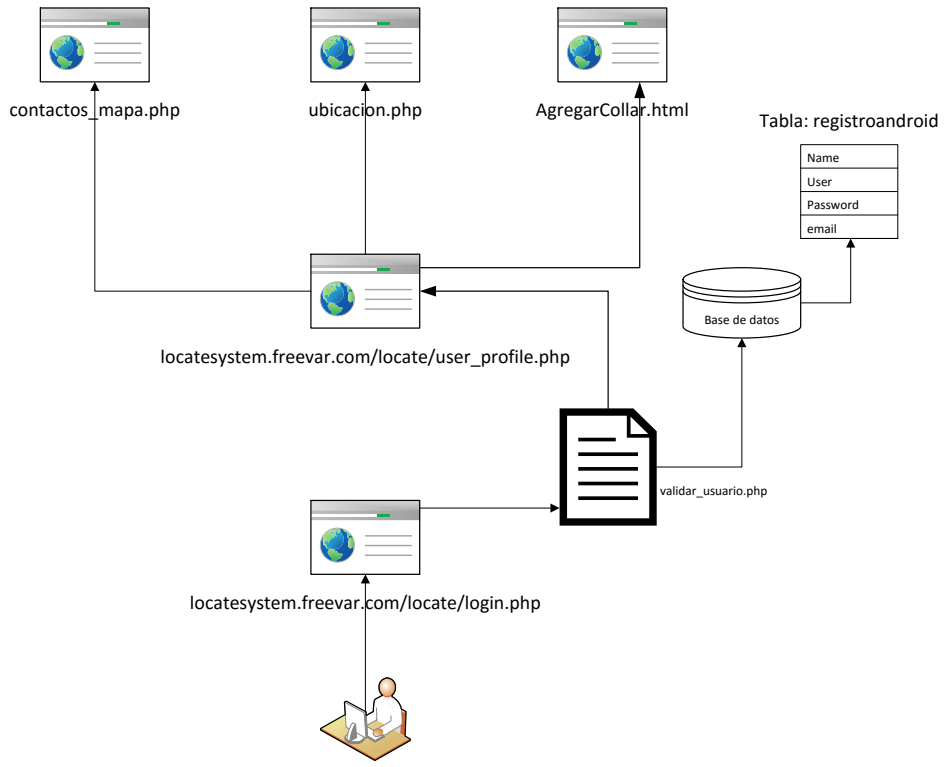


Figura 76 Diagrama representativo de ingreso al perfil de usuario

La Figura 76, muestra el diagrama representativo de ingreso de un usuario a su perfil. Mediante el formulario se ingresa el nombre de usuario y contraseña, dicho formulario envía los datos de autenticación al archivo de configuración "validar_usuario.php", el archivo compara con la tabla "registroandroid" en la base de datos, si el usuario y contraseña son correctos, el sistema reenvía al usuario a la página web "perfil_usuario.php" ubicado en la dirección:
"http://www.locatesystem.freevar.com/locate/perfil_usuario.php"
En la página web "perfil_usuario.php", el sistema muestra las opciones de registro que ofrece, mediante el código HTML que se muestra a continuación, se establece las opciones de acceso de un usuario afiliado, mediante tres formularios HTML, cada formulario contiene un botón, mediante el método "action", cada botón direcciona a la página que contiene el servicio al que requiere ingresar el usuario.

```
<form action="AgregarCollar.html">
<input type="submit" value="Agrega el nombre de tu mascota">
</form>
<form action="contactosmapa.php">
<input type="submit" value="Ver Ubicacion">
</form>
<form action="formulario_signos.php">
<input type="submit" value="Ver Signos Vitales">
</form>
```

El perfil de usuario consta de tres botones: Ver Ubicación, Ver Signos Vitales y Agregar el nombre de tu mascota, cada opción direcciona a las páginas "contactosmapa.php", "ubicación.php" y "AgregarCollar.html" respectivamente, el diseño gráfico de los botones de ingreso a las distintas opciones que ofrece la página web se observa en la Figura 77.

Figura 77 Vista web de opciones en perfil de usuario

- **Servicio Web “Agregar el nombre de tu mascota”**

La opción a la que debe ingresar un usuario que accede por primera vez es “Agregar el nombre de tu mascota”, dicha opción es importante para que el usuario agregue el nombre de la mascota en el sistema, y lo pueda identificar mediante el nombre. El usuario puede contener múltiples mascotas, por lo que dicha opción es importante para identificar la mascota en los servicios ofrecidos por el sistema. En la Figura 78 se representa el funcionamiento general de la opción “Agrega el nombre de tu mascota”.

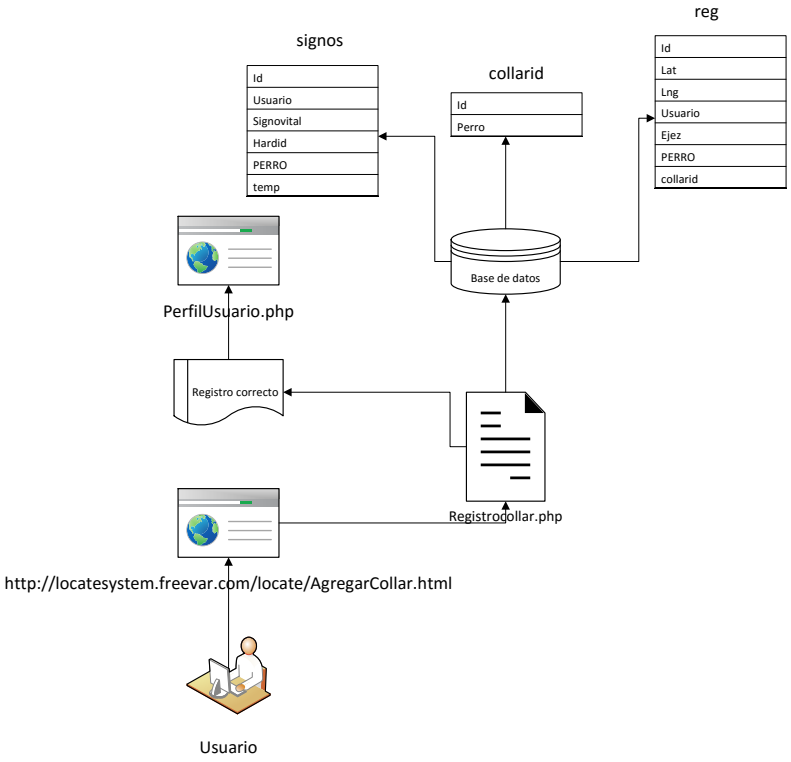


Figura 78 Diagrama representativo de funcionamiento de "AgregarCollar.html"

El usuario al elegir la opción “Agregar el nombre de tu mascota” (Ver Figura 77), se realiza el proceso indicado en la Figura 78, primero se direcciona a la archivo “AgregarCollar.html” alojado en la dirección:
“http://locatesystem.freevar.com/locate/AgregarCollar.html”.

Mediante un formulario el sistema solicita al usuario el ingreso del nombre de la mascota y el id del hardware, en este caso el arnés ubicado en el animal. La codificación en lenguaje HTML del formulario se muestra a continuación. Los métodos “action” y “post” representan la dirección del archivo de configuración a donde se envían los datos y el método HTTP con que se envía la información respectivamente.

```
<form action="registrocollar.php" method="post">
 <h3>Registra tu id:</h3>
<!--Id-->
<label for="CollarId">Id:</label>
<input type="text" name="CollarId" maxlength="16" />
<br/>
<!--Nombre Perro-->
<label for="perro">Nombre del perro:</label>
<input type="text" name="perro" maxlength="16" />
<br/>
<input type="submit" name="submit" value="Agregar Arnes" />
<input type="reset" name="clear" value="Borrar"/>
</form>
```

En las entradas de tipo “text” con los nombres “CollarId” y “perro” se ingresa la información del identificador del hardware y el nombre de la mascota respectivamente, mediante el botón de tipo “submit” se envía la información ingresada al archivo de configuración “registrocollar.php”, el diseño de vista web de la sección agregar ID se muestra en la Figura 79.

Figura 79 Vista web de "AgregarCollar.html"

El archivo de configuración “registrocollar.php” recibe la información ingresada por el usuario, realiza la validación de la información ingresada y el registro en la base de datos, la Figura 80 muestra el diagrama de flujo del archivo de configuración “registrocollar.php”.

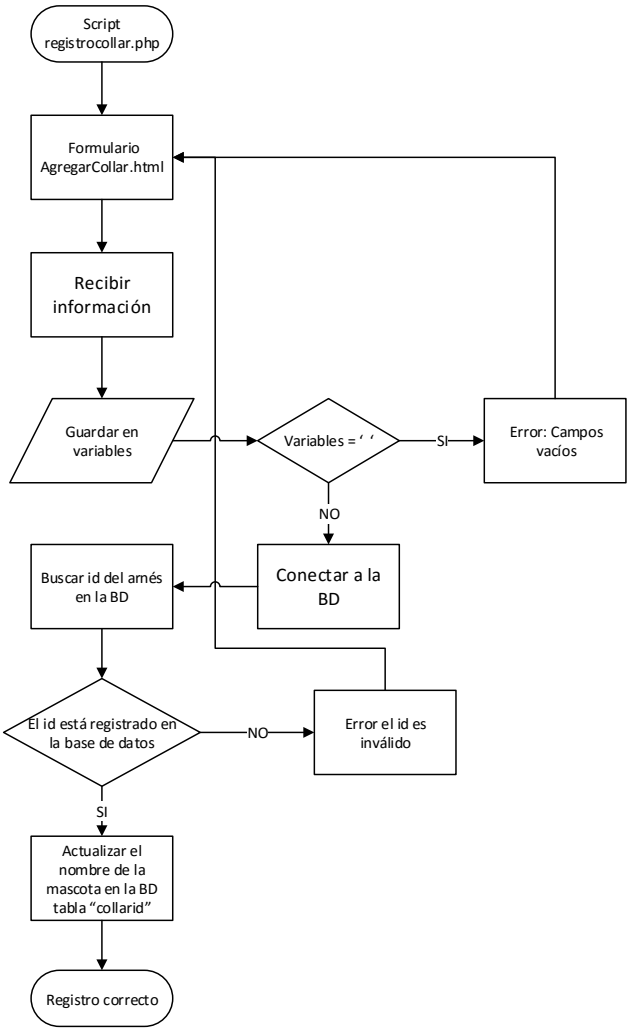


Figura 80 Diagrama de Flujo de archivo de configuración “registrocollar.php”

La Figura 80 muestra el diagrama general del proceso que realiza el script de configuración “registrocollar.php”, el cual recibe la información enviada por el formulario “registro.html”, y valida dicha información para el correcto o fallido registro del nombre de la mascota en el sistema. En primera instancia el script recibe la información enviada por el formulario mediante el método POST, y la guarda en las variables “$collarid” y “$perro”.

```
$collarid= $_POST['CollarId'];
$perro= $_POST['perro'];
```

Mediante la sentencia IF el script compara las variables que contienen la información ingresada en el formulario con valores vacíos, si una de las variables está vacía el script lo direcciona al formulario “AgregarCollar.html” para un nuevo intento de registro de información, la validación se realiza:

```
if($collarid== '' || $perro== ''){
    echo "<br />". "Por favor llena todos los campos" . "<br />";
    echo "<a href='AgregarCollar.html'>Intente nuevamente</a>";
    exit;}
```

Realiza la conexión con la base de datos para establecer la validación de la información ingresada en el formulario. Las variables “$host_db”, “$user_db” y “$pass_db” contienen la dirección del host, el usuario y la contraseña de autenticación con la base de datos respectivamente, los datos de conexión se almacenan en la variable “$conexion”.

```
$conexion = mysql_connect($host_db, $user_db, $pass_db);
```

Por medio de la variable “$conexión” se selecciona la base de datos a donde se conecta el sistema para realizar la validación de información.

```
mysql_select_db(923039, $conexion) or die("No se puede seleccionar
la base de datos.");
```

Se realiza la búsqueda en la base de datos del identificador ingresado por el usuario. El id tiene que estar registrado en la tabla “collarid” en el sistema para que el usuario pueda agregar el nombre de la mascota.

```
$buscarid = "SELECT * FROM collarid WHERE id = '$_POST[CollarId]'";
 $resultid = mysql_query($buscarid);
 $countid = mysql_num_rows($resultid);
```

En primera instancia se selecciona la tabla de la base de datos a donde el sistema va a comparar el identificador, el nombre de la tabla es “collarid”, mediante la variable “$resultid” se busca el identificador en la tabla, y mediante la variable “$countid” se guarda el valor numérico del número de valores correspondientes al conteo mencionado. Si el valor del conteo almacenado en la variable “$countid” es igual a 1, se establece que el valor ingresado es válido y se realiza la actualización de las tablas “reg”, “collarid” y “signos”, correspondientes a la tabla en donde se almacenan la ubicación, los identificadores de arnés y signos vitales, Dicha actualización de tablas se realiza:

```
if ($countid == 1){
$query = "UPDATE collarid, reg, signos SET
collarid.perro='$_POST[perro]', reg.PERRO='$_POST[perro]',
signos.perro ='$_POST[perro]' WHERE collarid.id = '$_POST[CollarId]'
AND reg.collarid='$_POST[CollarId]'AND
signos.hardid='$_POST[CollarId]' ";}
```

Los comandos indicados actualiza tres tablas de manera simultánea, ya que en la Tabla “reg” se almacenan los valores de ubicación, en la tabla “signos”, los signos vitales, y en la tabla “collarid” se almacenan los identificadores añadidos previamente por el administrador del sistema, cada hardware o arnés contiene un identificador único.

Si el valor del identificador ingresado en el formulario no existe en la tabla “collarid”, el sistema indica un error en el id ingresado, y direcciona al usuario al formulario “AgregarCollar.html” para que realice un nuevo intento de registro del nombre de la mascota. Mediante la sentencia ELSE:

```
else{
        echo "<br />". "El id del collar no es válido". "<br />";
        echo "<a href='AgregarCollar.html'>Intente nuevamente</a>";
        exit; }
```

- **Servicio Web "Ver Ubicación"**

La verificación de la ubicación de la mascota se la realiza pulsando el botón "Ver Ubicación" (Ver Figura 77), en la cual se despliega un mapa con marcadores que indican en donde se encuentra ubicado el animal en el mapa. El archivo de configuración de la opción mencionada se encuentra en la siguiente ruta: "http://locatesystem.freevar.com/locate/contactosmapa.php", para realizar el comportamiento deseado en el servicio, se agregó varios scripts de configuración. Para la implementación del servicio se requiere que el sistema de hardware envíe la ubicación de forma periódica. El diagrama de la Figura 81 representa el funcionamiento del servicio "Ver Ubicación".

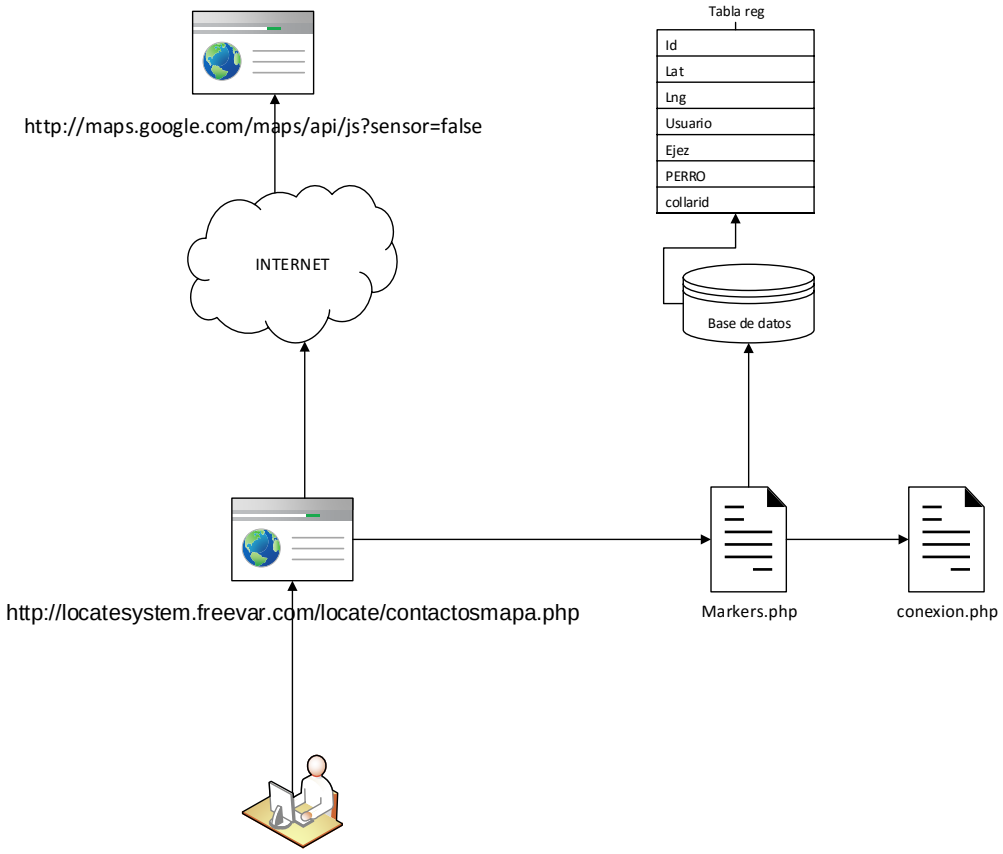


Figura 81 Diagrama representativo de conexión del servicio "Ver Ubicación"

Al realizar el inicio de sesión por medio de la variable "$_SESSION["usuario"]" se envía el valor del nombre del usuario con el que se realiza la validación con la

información obtenida en la base de datos. La variable de sesión es almacenada en la variable "$usuario" en el script PHP "contactosmapa.php". El proceso que realiza el servicio se representa mediante el diagrama de flujo de la Figura 82.

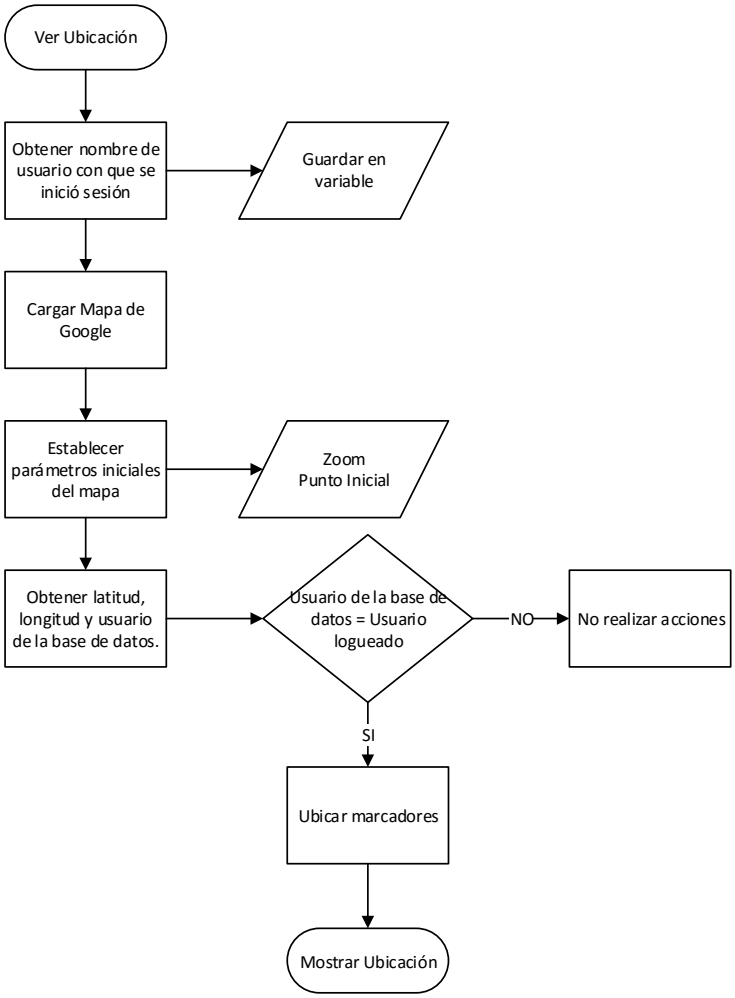


Figura 82 Diagrama de Flujo del servicio web "Mostrar Ubicación"

Primero se recibe el nombre de usuario ingresado para el inicio de sesión, y se lo almacena en la variable $usuario.

```
<?php
session_start();
$usuario= $_SESSION["usuario"]; ?>
```

Por medio del lenguaje de programación JavaScript, se carga el mapa de Google en la página web.

```
<script type="text/javascript"
src="http://maps.google.com/maps/api/js?sensor=false"></script>
```

Se especifica los parámetros de configuración iniciales, por medio de la función "load".

```
function load() {
var map = new google.maps.Map(document.getElementById("map"), {
//Se especifica el punto en donde se va a cargar la vista del mapa
center: new google.maps.LatLng(-0.19609929185424455,-
78.47674743679204),
//Se establece el zoom en el mapa
zoom: 12,
//Se establece el tipo de mapa
mapTypeId: 'roadmap'});}
```

Una vez establecidos los parámetros de inicialización del mapa, se procede a ubicar los marcadores de ubicación, para que el usuario conozca el lugar en donde se encuentra la mascota. Para establecer la ubicación, se procede a consultar los valores de la tabla "reg", en la cual se almacena la latitud y longitud enviada por el arnés. El script de configuración "markers.php" genera un archivo de tipo XML. Dicho archivo primero establece la conexión con la base de datos, mediante el archivo de configuración "conexión.php". Se incluye el archivo por medio del método "include".

```
include ("conexion.php");
```

Se selecciona la tabla contenedora de los valores latitud y longitud enviada por el arnés.

```
$sql=mysqli_query($con,"select * from reg ORDER BY Id");
```

Se obtienen e imprimen los datos correspondientes a la latitud, longitud, nombre de usuario y nombre de la mascota en formato XML. A continuación se indica por medio de un ejemplo el formato XML mencionado.

```
<markers>
<marker id="467" lat="-0.233916" lng="-78.524117"
usuario="CarlosAndres" perro="pepe" </markers>
```

El formato mostrado se genera de la siguiente forma:

```
while($row=mysqli_fetch_array($sql))
{
    echo "<marker id ='".$row['Id']."' lat='".$row['Lat']."'
lng='".$row['Lng']."' usuario='".$row['usuario']."'
perro='".$row['PERRO']."' ejez ='".$row['ejez']."'>\n";
    echo "</marker>\n";
}
```

En el archivo de configuración “contactosmapa.php” se accede al formato XML generado por el script “markers.php”. La variable “$usuario”, contiene el valor del nombre de usuario que ingresó al sistema, se realiza la conversión al tipo de variable String, la conversión se almacena en la variable “usuario1”.

```
var usuario= "<?php echo $usuario; ?>" ;
var usuario1 = String(usuario);
```

Se accede al formato XML establecido por el archivo de configuración “markers.php”. En específico se identifica el id “marker”.

```
var markers = xml.documentElement.getElementsByTagName("marker");
```

Mediante el método “for”, se ingresa a cada uno de las etiquetas XML dentro del id “marker”, la cual integra los valores almacenados en la base de datos. Se requiere el nombre de usuario y el nombre de la mascota almacenado en la tabla “reg”.

```
var txt= String(markers[i].getAttribute("usuario"));
var perro= String(markers[i].getAttribute("perro"));
```

El método “getAttribute “almacena en las variables “txt” y “perro” los valores de el nombre de usuario y nombre de la mascota almacenadas en la tabla “reg”. La variable “txt” se compara con la variable “$usuario”, si el nombre de usuario es igual al nombre de usuario almacenado en la base de datos, se ubican los marcadores en el mapa correspondiente a las mascotas del usuario ingresado, la forma es la siguiente:

```
if(txt==usuario1){
          var point = new google.maps.LatLng(
          parseFloat(markers[i].getAttribute("lat")),
          parseFloat(markers[i].getAttribute("lng")));
          var icon = 'marker.png';
          var marker = new google.maps.Marker({
          map: map,
          position: point,
          icon: icon,
          title: perro
          })}
```

La variable “point” ubica en el mapa dos atributos, latitud y longitud, recuperado del archivo XML mediante las variables “lat” y “lng”, por medio de la variable “icon” se establece la imagen “marker.png” la cual corresponde al puntero de ubicación en los mapas, en el título del marcador se establece el nombre de la mascota por medio de la variable “title”, en dicho método se ubica el valor del nombre de la mascota recuperado del archivo XML generado por el script de configuración “markers.php”. La Figura 83 muestra un ejemplo del mapa en la página web en donde el usuario inicio sesión.

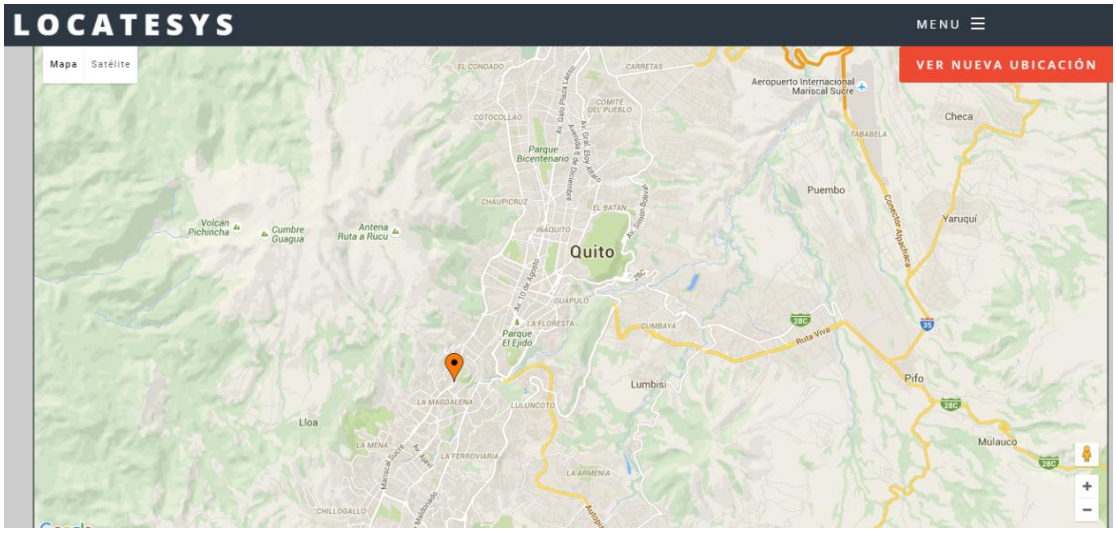


Figura 83 Mapa en página web

El botón “Ver Nueva Ubicación” recarga la página para que se recupere y ubique la ubicación actual en el mapa.

- **Servicio Web “Ver Signos Vitales”**

Otra opción que el sistema pone a disposición de usuario es “Ver Signos Vitales”, la opción en forma general muestra al usuario si la mascota se encuentra con signos vitales. Al elegir la opción de “Ver Signos Vitales” dentro del perfil de usuario se direcciona al archivo web ubicado en la siguiente dirección:
“http://locatesystem.freevar.com/locate/formulario_signos.php”
El archivo de configuración a donde se direcciona es “formulario_signos.php”. Se despliega un formulario en donde el sistema muestra las mascotas relacionadas al usuario ingresado. El formulario despliega botones del tipo “radio” para que el usuario tenga la facilidad de elección de la mascota que desea verificar los signos. En primera instancia, el script se conecta con la base de datos, y retorna los valores correspondientes al nombre de la mascota del usuario ingresado.

```
<?php

    $host_db = "localhost";
    $user_db = "923039";
```

```
    $pass_db = "12Carlos189Ex";
    $conexion = mysql_connect($host_db, $user_db, $pass_db);
    mysql_select_db(923039, $conexion) or die("No se puede
    seleccionar la base de datos.");
    $signos= mysql_query("SELECT PERRO FROM signos WHERE usuario =
   '$usuario'",$conexion);
?>
```

Mediante la información recuperada, se generan los botones tipo “radio”, y se los dispone en el formulario para el envío de información al script “validar_signos.php”, para la validación y despliegue de información correspondiente a la mascota seleccionada.

```
<h2> Elige tu mascota: </h2>
<form action="validar_signos.php" method="post" >
<?php
while($data = mysql_fetch_assoc($signos)){
echo "<h5><input type = 'radio' name ='id_mascota'
value='{$data['PERRO']}'>". $data['PERRO'].'</h5></br>';
}
mysqli_close($conn); ?>
<td colspan="2"><input name="iniciar" type="submit" value="Ver
signos vitales" /></td>
</form>
```

El formulario requiere la selección de la mascota, mediante el botón “Ver signos vitales” se envía la información ingresada por medio del método HTTP “POST” al archivo de configuración “validar_signos.php”. El archivo de configuración “validar_signos.php” recibe la información de nombre de mascota. Una vez recibida la información por medio del método HTTP “POST”.

```
$idmascota= $_POST["id_mascota"];
```

Se realiza la conexión con la base de datos en específico con la tabla “signos”.

```
$result = mysql_query("SELECT * FROM signos WHERE PERRO =
'$idmascota' OR hardid='$idmascota'");
```

Si el nombre de la mascota se encuentra registrado en la base de datos se genera una sesión con el nombre del arnés o la mascota y se direcciona al archivo de configuración “signosvitales.php”.

```
if($row = mysql_fetch_array($result))
{ session_start();
  $_SESSION['usuario'] = $usuario;
  $_SESSION['mascota'] = $idmascota;
 echo "<script>window.location.href ='signosvitales.php';</script>";
//Redireccionamos a la pagina: signosvitales.php
header("Location: signosvitales.php");
```

Según el nombre de mascota seleccionado, el sistema direcciona al archivo de configuración “signosvitales.php”. Primero el script recibe los valores correspondientes al usuario y al nombre de la mascota.

```
<?php
session_start();
$usuario= $_SESSION["usuario"];
$mascota= $_SESSION["mascota"]; ?>
```

Por medio de un estilo CSS, se establece la animación representativa correspondientes a los signos vitales, si la mascota tiene movimiento, se establece en la animación el color del corazón en rojo y con movimiento, en caso de que no exista movimiento, el color del corazón se establece en azul sin movimiento. El estilo CSS se configuró de la siguiente manera:

```
<style type="text/css">
@keyframes latidos {
    from { transform: none; }
    50% { transform: scale(1.4); }
    to { transform: none; }
}
```

```
.corazon {
    display: inline-block;
    font-size: 150px;
    text-shadow: 0 0 10px #222,1px 1px  0 #450505;
    color: red;
    animation: latidos .5s infinite;
    transform-origin: center;
}
.corazon_detenido {
    display: inline-block;
    font-size: 150px;
    text-shadow: 0 0 10px #222,1px 1px  0 #450505;
    color: blue;
    transform-origin: center;
} </style>
```

La etiqueta CSS ".corazon" corresponde al estilo si es que se detecta movimiento en el animal, el estilo ".corazon_detenido" corresponde al estilo si es que no se detectó movimiento. Se obtiene los valores de signos vitales almacenados en la tabla "signos" de la base de datos, si el valor es menor a un valor especificado en la etapa de pruebas, la mascota no se está movimiento, caso contrario se puede asumir que la mascota se encuentra con vida mediante las pruebas realizadas.

```
<?php
if ($valor_signo > 10){
echo "Si existen signos vitales \n" ;
?>
    <html>
    <div class="corazon">&#x2665;</div>
    </html>
    <?php
    echo "<br>";
    echo "Temperatura: \n";
    echo $temp; echo "°C \n";
    echo "<br>"; echo "<br>";
    echo "Visto por ultima vez: \n"; echo $tiempo
```

```
    ?>
    <?php
    }
 else {
    echo "Alerta no se detecto signo vital \n" ;?>
    <html>
    <div class="corazon_detenido">&#x2665;</div>
    </html>
    <?php
    echo "<br>"; echo "Temperatura: \n";
    echo $temp; echo "<br>"; echo "<br>";
    echo "Visto por ultima vez: \n";
     echo $tiempo
     ?> <?php  }  ?>
```

La variable “$valor_signo” corresponde a la resta del mayor valor con el menor valor de la información de diez muestras obtenidas por el sensor acelerómetro, lo que indica si existe vibración de movimiento en la mascota o no, es decir realizando la resta y comparándola con un valor, se va a poder determinar si la mascota se encuentra con movimiento. El valor establecido es “10”, ya que las mínimas vibraciones hacen variar la información obtenida por el sensor acelerómetro. La resta establece si existe variación en los valores censados. La segunda medición de signos vitales recuperada por el arnés es la temperatura externa de la mascota, a pesar que no se puede conocer el estado médico del animal mediante esta medida, ya que para conocer la temperatura médica real se debe realizar la medición vía rectal, es importante para que el usuario pueda conocer la temperatura externa del animal, y saber si se encuentra con vida. La medida de temperatura se recupera de la tabla “signos”, y se muestra para que el usuario pueda conocer la temperatura de la mascota.

Por medio de la Figura 84 se muestra un ejemplo de las opciones de imágenes de signos vitales que se muestra en el sistema según el valor recuperado de la tabla “signos” almacenada en la base de datos.

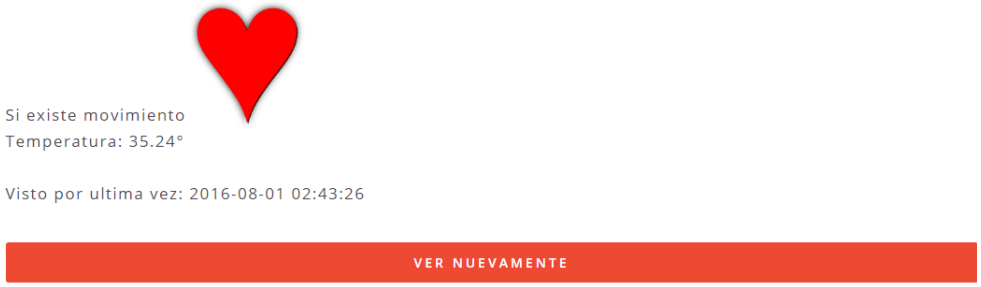


Figura 84 Alerta de Signos vitales en página web

Mediante el botón "Ver Nuevamente" se recarga la página para que el sistema ofrezca una información actualizada al usuario.

3.4. Diseño e implementación de la aplicación en Android.

La aplicación móvil fue desarrollada para el Sistema Operativo Android, la ventaja de Android es la facilidad de accesibilidad al software de desarrollo, porque es de código libre, además incluye diversas librerías. Android Studio IDE es el software oficial para desarrollo en Android, su adquisición es libre, por lo que nos brinda facilidad al momento de desarrollar. El Anexo 1 muestra de forma detallada la instalación, configuración y estructura de un proyecto en Android Studio.

3.4.1. Diseño de la aplicación en Android.

Mediante el diagrama de flujo de la Figura 85, se establece los pasos que un usuario por medio de un terminal (Teléfono móvil Android), debe seguir para usar el aplicativo, como se ha mencionado, las opciones que el sistema propone son principalmente la verificación de la ubicación y signos vitales de la mascota con el sistema de hardware ubicado en la misma.

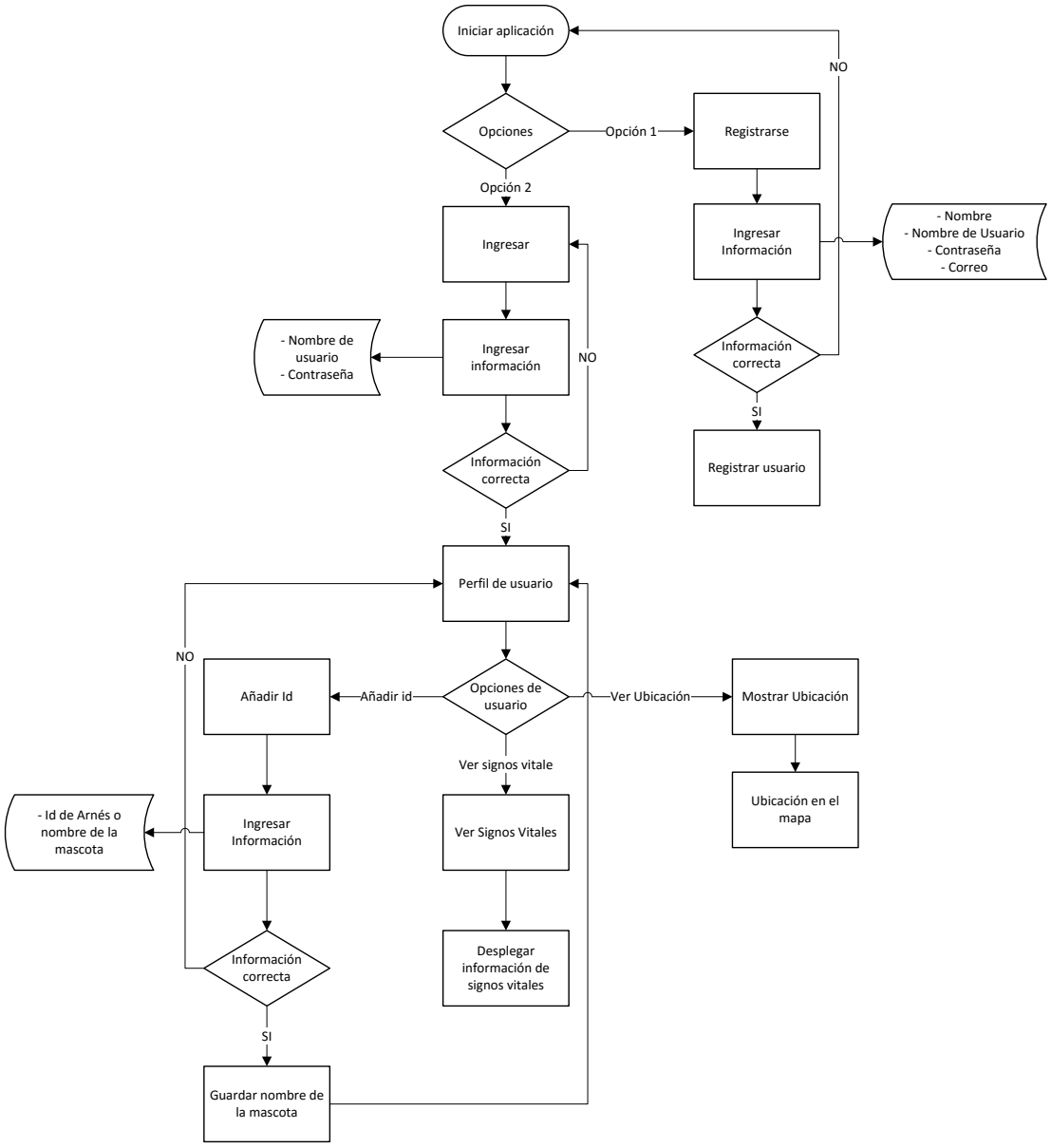


Figura 85 Diagrama de Flujo general del uso en la aplicación en Android

Al iniciar la aplicación (Ver Figura 85), el sistema ofrece dos opciones a elección del usuario, “Iniciar Sesión” y “Registrarse”, para acceder a los servicios que el sistema brinda, un usuario tiene que registrase, y adquirir el arnés para mascotas. Una vez cumplidos ambos requisitos, se accede al sistema por medio de la opción “Iniciar Sesión”, se ingresa la información de “Nombre de Usuario” y “Contraseña”, se valida si la información es correcta, si es así, el sistema direcciona al “Perfil de Usuario” (Ver Figura 85). En el perfil, el usuario puede elegir entre las opciones: “Añadir Mascota”, “Ver Ubicación” y “Ver Signos Vitales”. La opción “Añadir

Mascota", ofrece la posibilidad de agregar el nombre de la mascota para la identificación en el sistema, las opciones de "Ver Signos Vitales" y "Ver Ubicación", brindan la funcionalidad principal en del proyecto, la capacidad de verificar si el animal se encuentra con vida (Movimiento y temperatura) en una situación y la ubicación de la mascota en el mapa.

El diagrama de arquitectura de contenidos (Ver Figura 86), representa las pantallas y contenidos de la aplicación, las pantallas están representadas como rectángulos, cuyo título se sitúa en la parte superior fuera del rectángulo, y su contenido es representado dentro, además mediante líneas horizontales de color rojo se realiza una separación del nivel de acceso a las pantallas en la aplicación, en este caso se identificó 5 niveles.

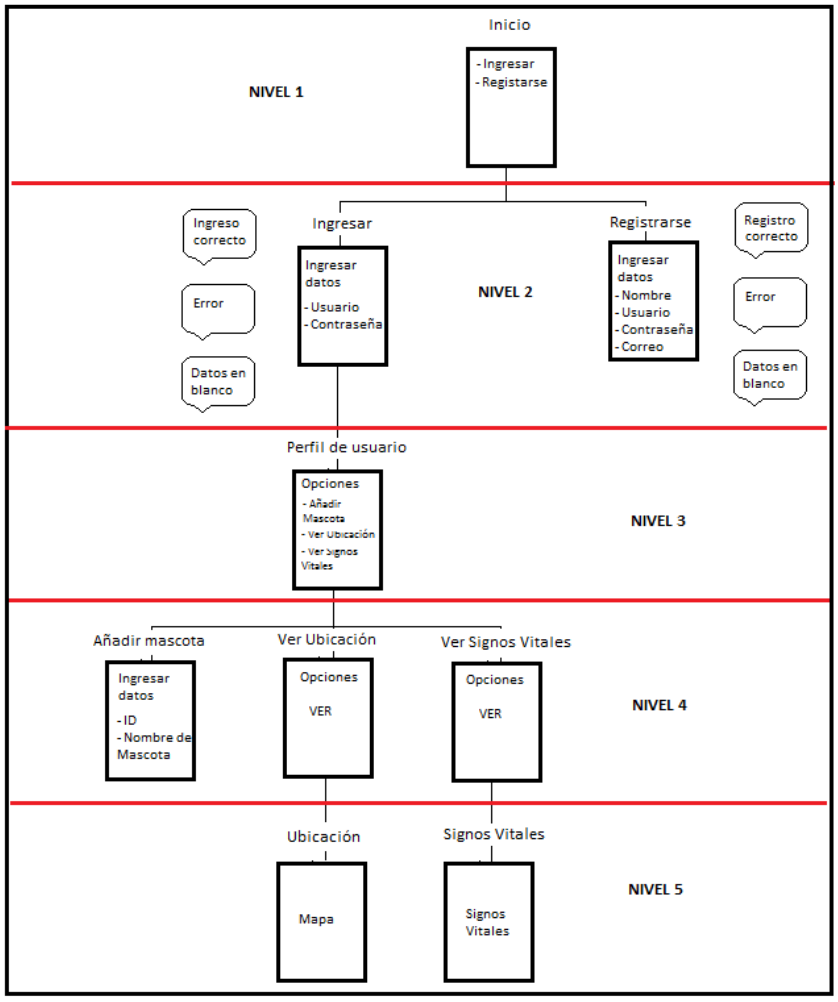


Figura 86 Diagrama de Arquitectura de contenidos de la aplicación en Android

Se tiene que establecer un diseño visualmente atractivo al usuario, la forma básica de una pantalla en la aplicación para dispositivo móvil, tiene la estructura establecida en la Figura 87.

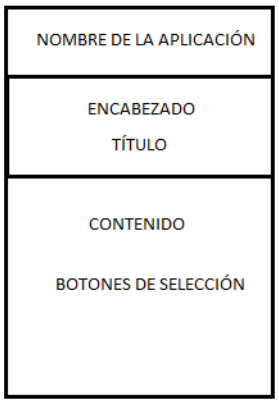


Figura 87 Diseño Visual de la aplicación en Android

3.4.2. Implementación de servicios para la aplicación en Android.

La aplicación móvil interactúa con el servidor web para establecer los siguientes servicios:

- Registrarse
- Iniciar Sesión
- Añadir nombre de la mascota.
- Ver Ubicación
- Ver Signos Vitales (Movimiento y temperatura).

La primera pantalla al que el usuario tiene acceso, contiene dos botones, las cuales ponen a disposición del usuario dos opciones iniciales, “Registrarse” e “Iniciar Sesión”. Para conocer de una manera detallada la estructura de una aplicación desarrollada en Android Studio IDE, revisar Anexo 1. La Figura 88 muestra el “layout” asociado a la clase “Inicio.class”, que corresponde a la pantalla inicial a la que el usuario ingresa. El “layout” mencionado es “start_activity.xml”

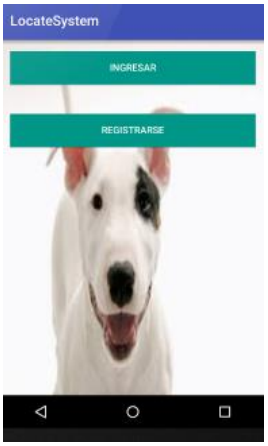


Figura 88 Layout "start_activity.xml"

La clase "Inicio.class" es la clase que proporciona los servicios de ingreso a las opciones "Iniciar Sesión" y "Registrarse", por medio de la Figura 89 se representa el diagrama de clases correspondiente a la clase asociada con la pantalla inicial de la aplicación.

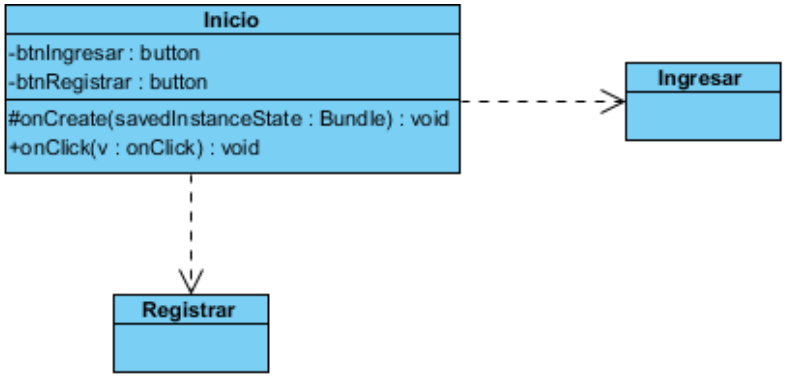


Figura 89 Diagrama de clases "Inicio.class"

La Figura 89, muestra la representación de las variables y métodos establecidos en la clase "Inicio.class", las variables "btnIngresar" y "btnRegistrar", direccionan al usuario a la opción Ingresar o Registrarse respectivamente. El método "onCreate", inicializa la información a cargar en la pantalla, mientras que el método onClic direcciona a las clases, correspondientes al botón pulsado por el usuario.

```
if(v == btnIngresar){
    startActivity(new Intent(this, Ingresar.class));}
```

```
if(v == btnRegistrar){
    startActivity(new Intent(this, Registrar.class));}
```

3.4.2.1. Servicio de Registro en Android.

El diagrama de flujo representado en la Figura 90 muestra las tareas que realiza la aplicación en Android para que un usuario se registre en el sistema.

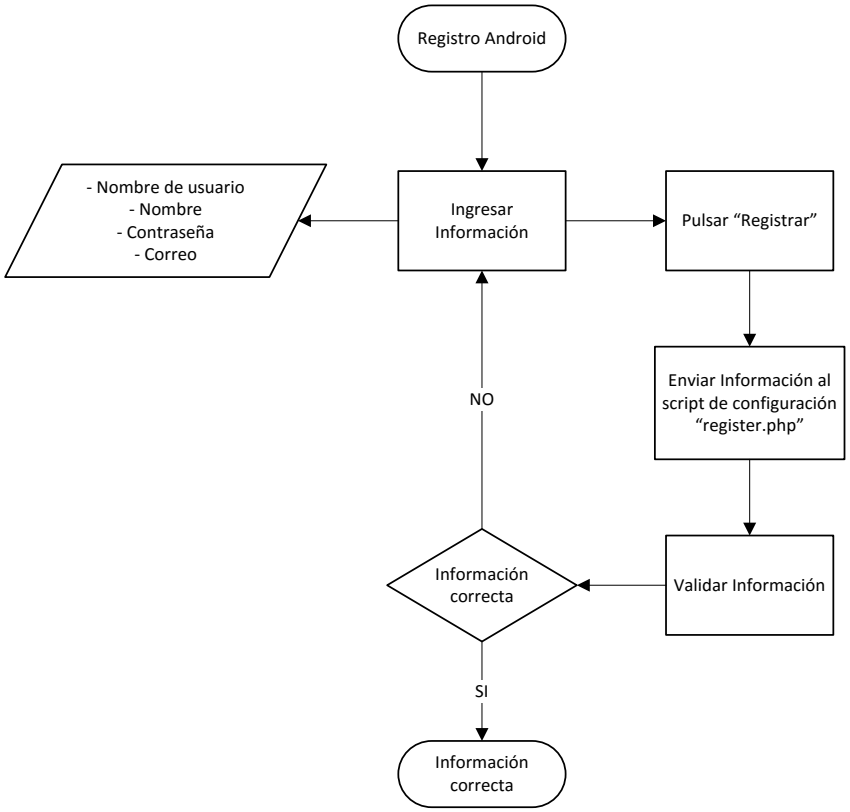


Figura 90 Diagrama de flujo del servicio de registro en Android

El diagrama de clases de la Figura 91 representa la interacción de las clases, variables y métodos incluidos en la clase Registrar.class, la clase mencionada se encuentra relacionada con el layout "register_activity.xml".

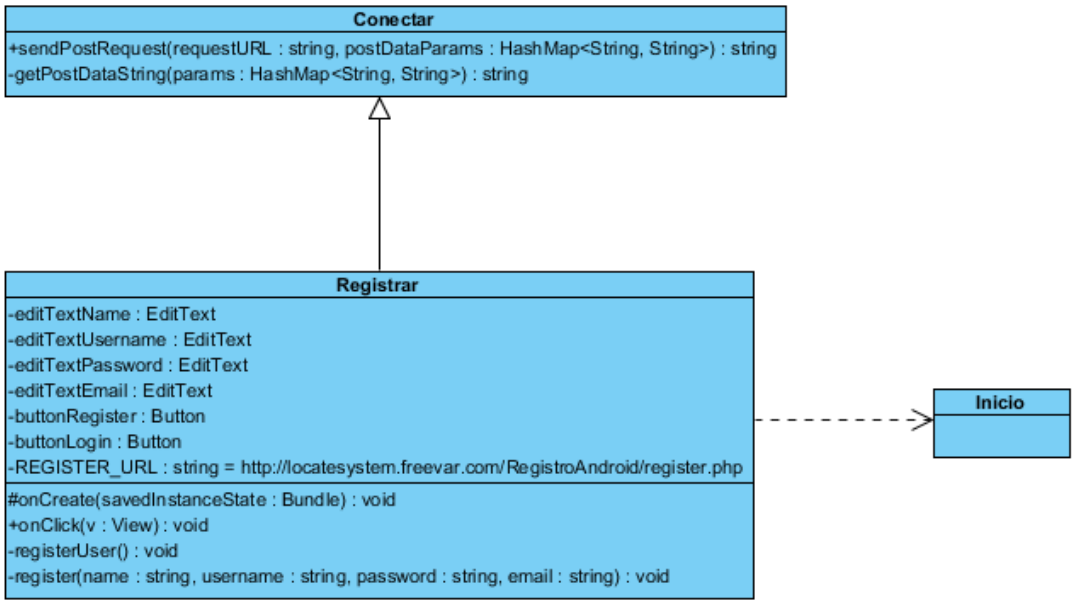


Figura 91 Diagrama de clases "Registrar.class"

La Figura 91 representa la iteración de las clases relacionadas al registro en la aplicación en Android, la clase "Registrar.class" hereda el método "sendPostRequest" de la clase "Conectar.class", la cual realiza el envío de la información ingresada por el usuario en el layout "register_activity.xml" asociado a la clase "Registrar.class" (Ver Figura 92), mediante el método HTTP "POST".

Figura 92 Layout "register_activity.xml"

El método "register" se establece mediante hilos asíncronos, es decir realiza varias tareas en distintos tiempos. En primera instancia se realiza la declaración de objetos.

```
ProgressDialog loading;
Conectar ruc = new Conectar();
```

El objeto “loading” del tipo “ProgressDialog”, incluye la función para el llamado de la barra indicadora de carga o de espera de la aplicación. El llamado del método “sendPostRequest” se lo ejecuta en el método “register”, por medio del objeto “ruc” del tipo “conectar”. El primer hilo de la clase “register” es “onPreExecute”, en el cual se genera la gráfica indicadora de espera cargada por medio del objeto “loading”.

```
loading = ProgressDialog.show(Registrar.this, "Please Wait",null,
true, true);
```

El segundo hilo incluido en la clase “register” es “doInBackground”, en este se disponen las funciones usadas para el envío de la información ingresada por el usuario a través del layout “register_activity.xml” (Ver Figura 92).

```
HashMap<String, String> data = new HashMap<String,String>();
                data.put("name",params[0]);
                data.put("username",params[1]);
                data.put("password",params[2]);
                data.put("email",params[3]);
String result = ruc.sendPostRequest(REGISTER_URL,data);
return  result;
```

El método “HashMap”, dispone los parámetros recibidos por el método “register”, name, username, password y email, y los representa por medio de una clave, con la que se identifica el valor de los atributos ingresados, una vez representada la información se procede a realizar el envío por medio de la variable “data” al archivo de configuración “register.php” alojado en el servidor web, el cual está almacenado en la variable REGISTER_URL, y cuyo valor se aprecia en la Figura 91. El envío se lo realiza por medio del objeto ruc, del tipo Conectar, el cual hereda el método “sendPostRequest”, y se encarga de realizar el envío de la información ingresada. Por medio del método “onClick” (Ver Figura 91), el sistema llama al método de

envío de información al script alojado en el servidor web, y lo direcciona a la clase "inicio.class", mostrando un mensaje informativo del estado del registro.

El script de configuración "register.php", está almacenado en la siguiente dirección URL: "http://locatesystem.freevar.com/RegistroAndroid/register.php", las tareas que realiza el script de configuración son representadas mediante el diagrama de flujo de la Figura 93.

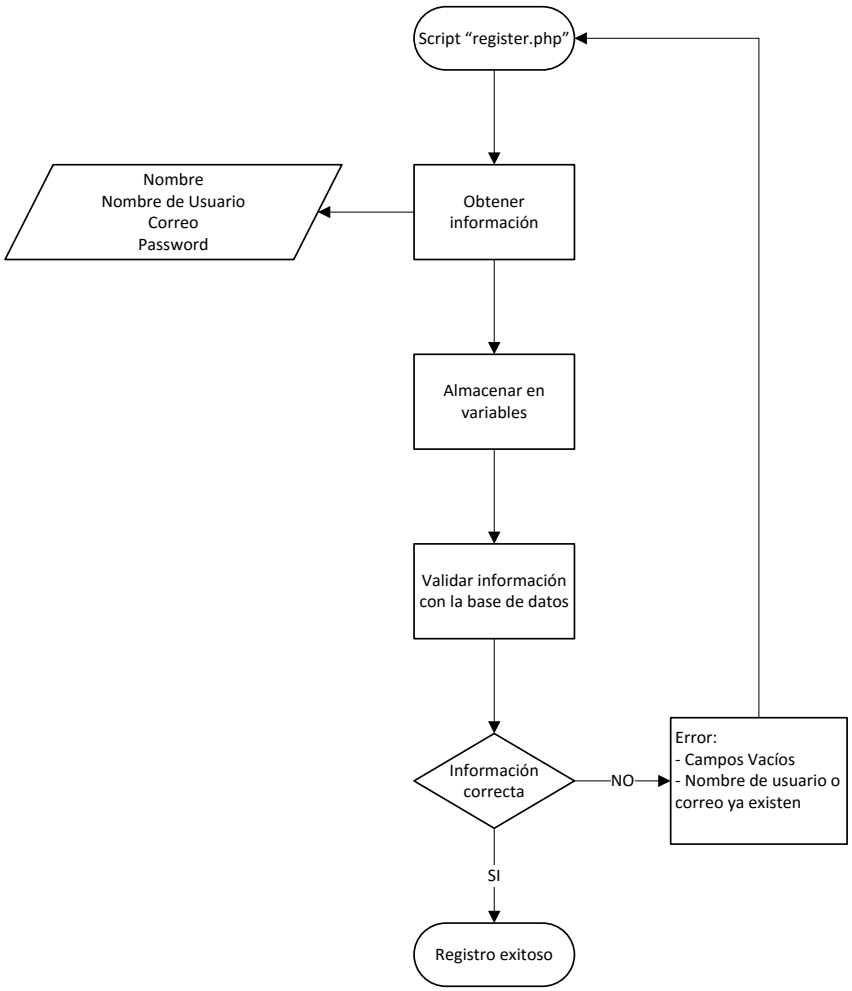


Figura 93 Diagrama de flujo script "register.php"

La implementación del script representado en la Figura 93, es similar a los script configurados para la página web analizados previamente, primero recibe la información, y la almacena en variables.

```
$name = $_POST['nombre'];
```

```
$username = $_POST['nombreUsuario'];
$password = $_POST['password'];
$email = $_POST['correo'];
```

Se valida si no existen campos vacíos y si el nombre de usuario y correo no se encuentra registrado en la base de datos previamente.

```
if($name == '' || $username == '' || $password == '' || $email == ''){ echo 'Por favor llena todos los campos';
}else{
require_once('dbConnect.php');
$sql = "SELECT * FROM registroandroid WHERE user='$username' OR email='$email'";
$check = mysqli_fetch_array(mysqli_query($con,$sql));
if(isset($check)){ echo 'Nombre de usuario o email ya existen ';}
```

Si no se completó la información requerida, o si el nombre de usuario o correo ya existen, el script devuelve un mensaje de error, para informar al usuario el inconveniente que no le permite registrar. Si ninguna de ambas validaciones se cumple, el sistema registra la información ingresada en la tabla “registroandroid”.

```
$sql = "INSERT INTO registroandroid(name,user,password,email) VALUES('$name','$username','$password','$email')";
```

Con lo que se retorna al usuario un mensaje informando el registro exitoso en el sistema.

3.4.2.2. Servicio de Inicio de Sesión en Android

Mediante el diagrama de flujo de la Figura 94 se muestra las tareas que el sistema realiza para que un usuario registrado inicie sesión en el sistema.

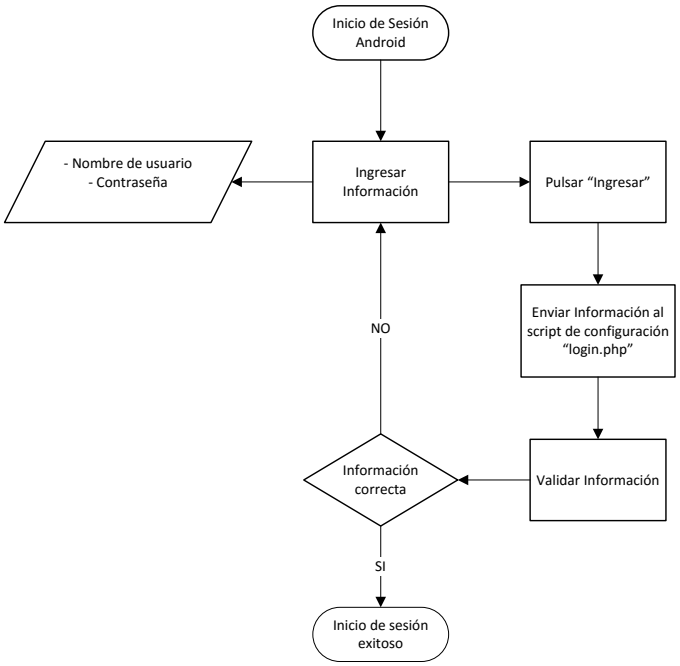


Figura 94 Diagrama de flujo inicio de sesión en Android

El diagrama de clases de la Figura 95, representa la iteración de las clases que conforman el servicio de "Inicio de Sesión" en la aplicación Android. La clase "Ingresar.class" hereda el método "sendPostRequest" de la clase "Conectar.class", el cual es usado para el envío de la información al archivo de configuración alojado en el servidor web.

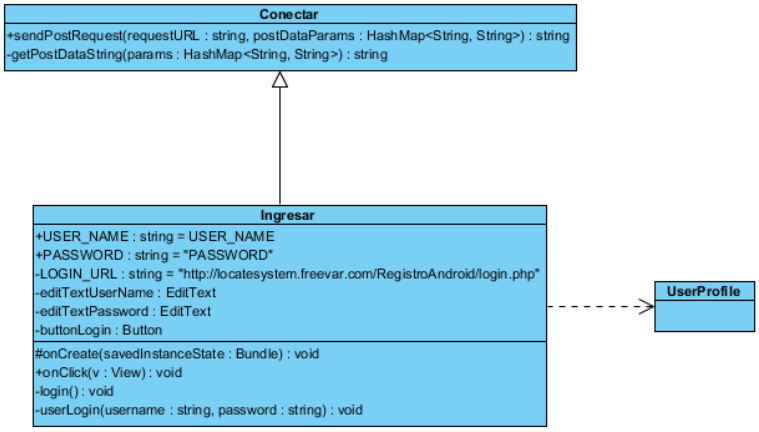


Figura 95 Diagrama de clases de "Ingresar.class"

La clase "Ingresar.class" se asocia con el layout "login_activity.xml" (Ver Figura 96), el cual contiene el diseño de la pantalla que corresponde al ingreso en la

aplicación, el usuario ingresa el nombre de usuario y contraseña y pulsa el botón "Ingresar", para que el sistema realice la validación de la información ingresada.

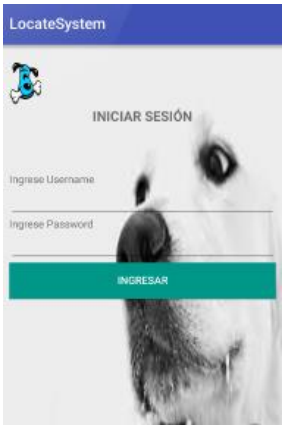


Figura 96 Layout "login_activity.xml"

El método "onCreate" (Ver Figura 95), inicializa las variables del tipo "button" y "editText" definidos en el layout, además de asociar el layout a la clase. El método "login_user" (Ver Figura 95) por medio de los atributos "username" y "password" envía la información ingresada al archivo de configuración "login.php", la dirección del script de configuración está almacenada en la variable "LOGIN_URL". El método "userLogin" actúa mediante hilos asíncronos, por lo que contiene varios métodos integrados, el método "onPostExecute" carga un mensaje de espera por medio del objeto "loading" de tipo "Progress Dialog".

```
loading = ProgressDialog.show(Ingresar.this,"Please
Wait",null,true,true);
```

El método "doInBackground", en este se disponen las funciones usadas para el envío de la información ingresada por el usuario a través del layout "login_activity.xml" (Ver Figura 96).

```
HashMap<String,String> data = new HashMap<>();
data.put("username",params[0]);
data.put("password",params[1]);
```

El método “HashMap”, dispone los atributos recibidos por el método “user_login”, username y password, y los representa por medio de una clave, con la cual se identifica el valor correspondiente a los atributos recibidos, una vez representada la información se procede a realizar el envío por medio del objeto “data” al archivo de configuración register.php alojado en el servidor web, su dirección está almacenada en la variable “LOGIN_URL”, y su valor se aprecia en la Figura 95.

```
Conectar ruc = new Conectar();
String result = ruc.sendPostRequest(LOGIN_URL,data); return result;
```

Para realizar el envío se usó el objeto ruc, perteneciente a la clase “Conectar.class”, de la cual hereda el método “sendPostRequest”, y realiza el envío de la información correspondiente. El script de configuración “login.php” retorna el valor “success” si es que el nombre de usuario y contraseña son correctos. Si es retornado el valor “success”, la clase direcciona al perfil de usuario de la aplicación, que se encuentra configurado en la clase “UserProfile.class”.

```
if(s.equalsIgnoreCase("success")){
    Intent intent = new Intent(Ingresar.this,UserProfile.class);}
```

Mediante el método “onClick” (Ver Figura 95) se detecta si el usuario pulsa el botón “Ingresar”, con lo cual realiza toda la metodología mencionada para el ingreso al perfil. Por medio del diagrama de flujo de la Figura 97, se muestra la metodología de funcionamiento del script de configuración “login.php”.

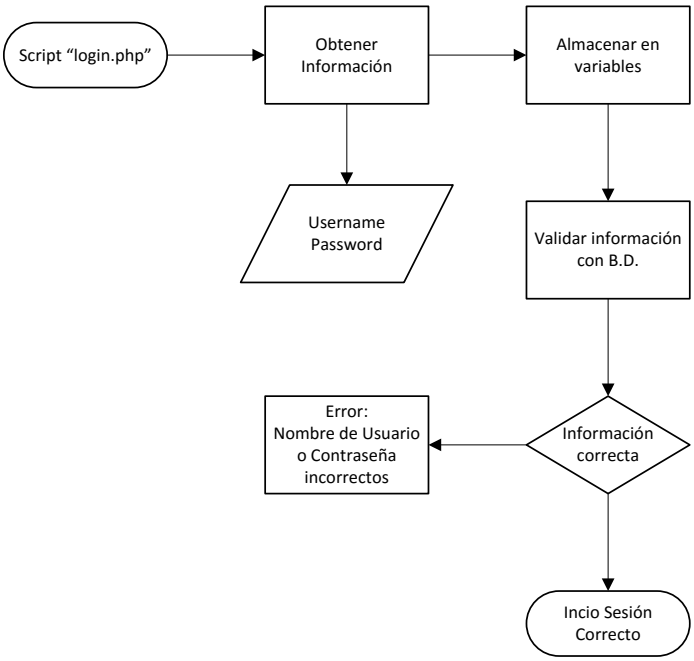


Figura 97 Diagrama de flujo de script de configuración “login.php”

La implementación del script maneja la misma metodología de los archivos de configuración antes mencionados, por lo que una nueva explicación no es necesaria. Una vez que el sistema compara y valida la información ingresada por el usuario con la tabla “registroandroid”, la aplicación dirige al usuario a la clase “UserProfile.class”, la cual está asociada con el layout “user_profile.xml” (Ver Figura 98).

Figura 98 Layout “user_profile.xml”

La clase UserProfile, muestra al usuario las opciones de servicios disponibles por el sistema. Para que el nombre de la mascota se almacene en el sistema, el usuario

debe pulsar la opción “Añadir Mascota”, dicha opción direcciona a la clase “AnadirId.class”, (Ver Figura 99), la clase hace referencia al layout “Anadir_activity.xml”.

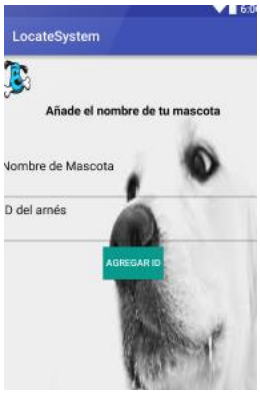


Figura 99 Layout “Anadir_activity.xml”

El diagrama de clases de la Figura 100 representa el funcionamiento y la asociación de las clases relacionadas con la función de añadir el nombre de la mascota. La clase contiene dos atributos del tipo “EditText”: “editTextIdArnes” y “editTextPerro”, en los cuales el usuario ingresa el identificador del arnés proporcionado por el administrador mediante el hardware y el nombre del perro. El atributo “btnAgregarId” mediante los métodos “AnadirCollar”, “Anadir” y “onClick” realiza la consulta al archivo de configuración alojado en el servidor web, la localización del script está representado por el atributo “REGISTER_URL” la dirección URL en donde está almacenado el script:
“http://locatesystem.freevar.com/RegistroAndroid/AnadirCollar.php”
Y corresponde a la dirección en donde está alojado el archivo de configuración en el servidor web y realiza las funciones de validación del Id y registro del nombre de la mascota en la base de datos. La clase “AnadirId”, depende de la clase “Conectar”, ya que utiliza el método “sendPostRequest” de dicha clase (Ver Figura 100), para enviar los datos al servidor web para su posterior validación de información.

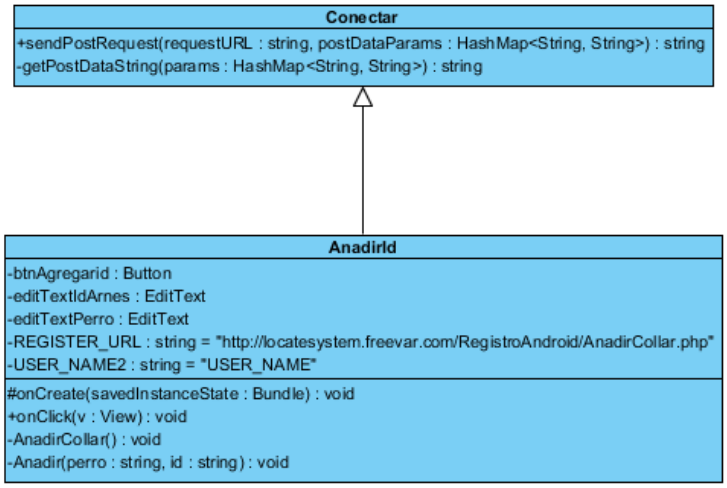


Figura 100 Diagrama de clases "AnadirId.class"

3.4.2.3. Servicio de Ver Ubicación en Android.

Una de las tareas más importantes del proyecto es la verificación de la ubicación de la mascota a través de la aplicación Android, una vez ingresada a la opción en el perfil de usuario, el sistema provee del servicio de verificación de ubicación de las mascotas correspondientes únicamente al usuario ingresado. El diagrama de flujo de la Figura 101, muestra las tareas que se realiza para brindar el servicio.

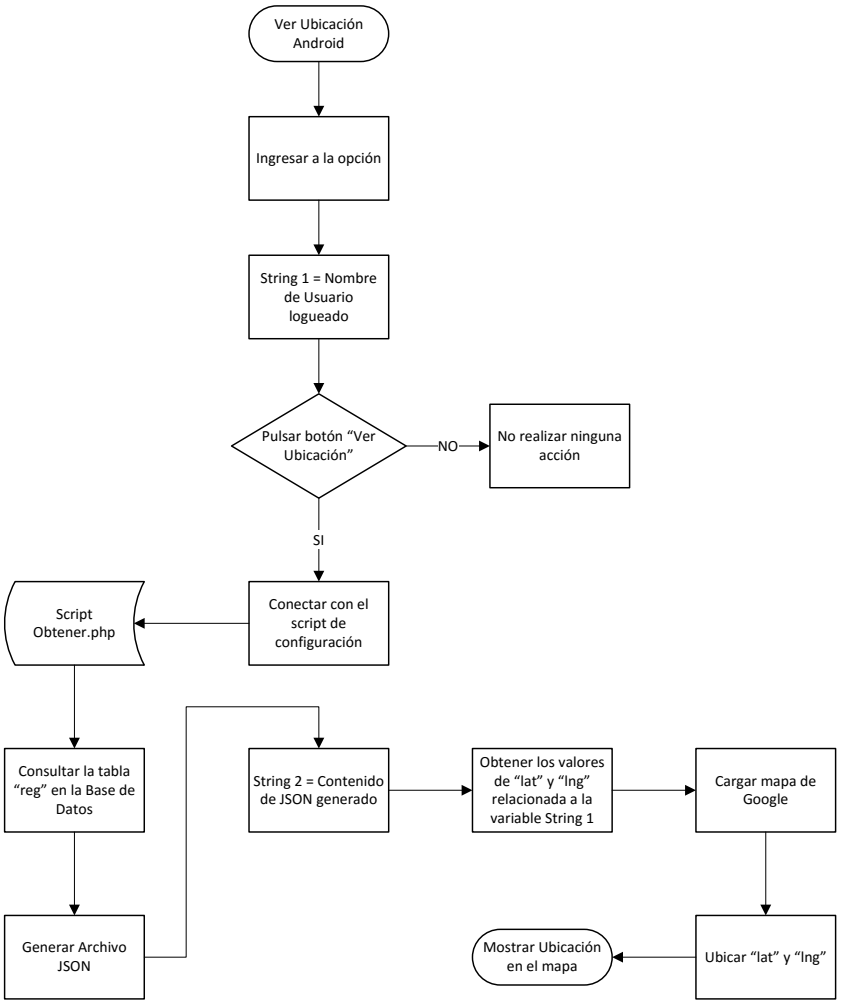


Figura 101 Diagrama de flujo Ver Ubicación Android

El diagrama de clases de la Figura 102 muestra el comportamiento general de las clases relacionadas con el servicio de ubicación en la aplicación en Android. Las clases que realizan la funcionalidad, son "ConvJsonTxt.class" y "MapsActivity.class", obtienen la información de los valores de posicionamiento geográfico almacenados en la base de datos y los ubican en el mapa.

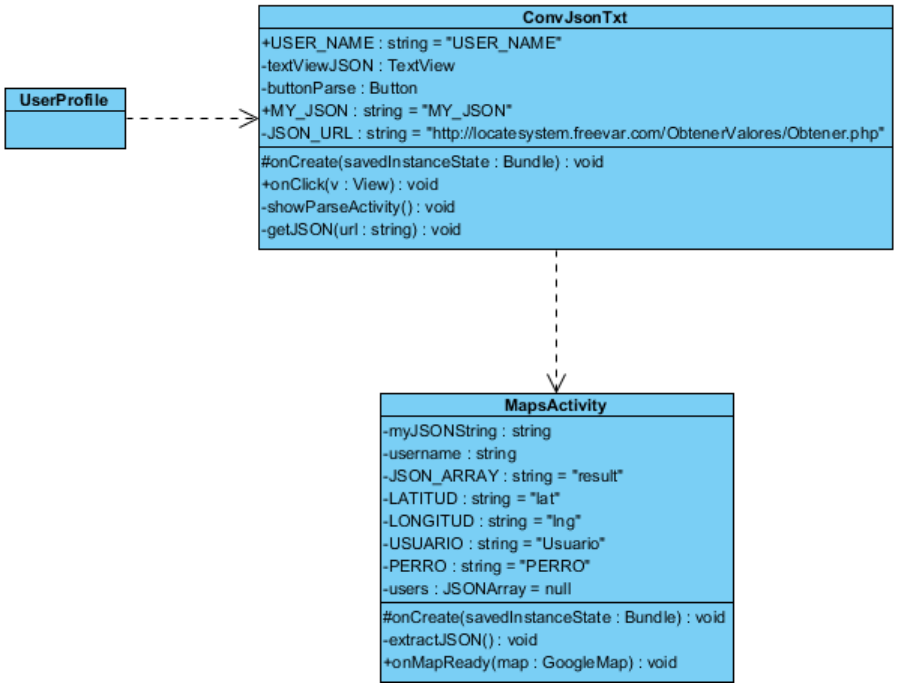


Figura 102 Diagrama de clases relacionadas al servicio de ubicación en Android

En el perfil de usuario se elige la opción de "Ver Ubicación", una vez que el usuario elije la opción, la aplicación lo direcciona a la clase "ConvJsonTxt.class" la cual está asociada con el layout "reading_json.xml" (Ver Figura 103).

Figura 103 Layout "reading_json.xml"

La clase "ConvJsonTxt.class" recibe la información de posicionamiento geográfico, latitud y longitud almacenada en la base de datos, para su posterior ubicación en el mapa de Google. En el método "OnCreate" (Ver Figura 102) se inicializan las variables tipo "Button" y "EditText", además de asociar la clase con el layout "reading_json.xml" (Ver Figura 103). En el método "onClick" se verifica la

pulsación del botón “Mostrar Ubicación”, representado por la variable de tipo “button” “buttonParse”. La clase “getJson” recibe el parámetro que corresponde a la dirección URL en donde está alojado el script de configuración “Obtener.php”, la variable que contiene la dirección mencionada es “JSON_URL”. El script de configuración realiza una consulta a la base de datos y genera un archivo del tipo JSON, el cual tiene la siguiente forma:


```
{"result":[{"id":"471","Usuario":"CarlosAndres","lat":"-
0.233916","lng":"-78.524117","ejez":"74","PERRO":"pepe"},
{"id":"472","Usuario":"CarlosAndres","lat":"-0.233916","lng":"-
78.524117","ejez":"73","PERRO":"pepe"},
```


El método “getJSON” hace el llamado de tres hilos asíncronos, el primer hilo es el método “onPreExecute”, únicamente carga una vista de espera para el usuario.

```
loading = ProgressDialog.show(ConvJsonTxt.this, "Please
Wait...",null,true,true);
```

El segundo hilo es “doInBackground”, en este se realiza la consulta a la información generada por el script de configuración “Obtener.php”, y se lo almacena en una variable de tipo String.

```
String json;
while((json = bufferedReader.readLine())!= null){
    sb.append(json+"\n");} return sb.toString().trim();
```

Una vez realizada la consulta, ejecuta el método “onPostExecute”, en donde se dispone en la variable textViewJSON de tipo “TextView” el JSON generado por el script.

```
textViewJSON.setText(s);
```

En el hilo “onPostExecute” una vez almacenado el JSON recuperado del script de configuración, se ejecuta el método “showParseActivity” (Ver Figura 102). El método “showParseActivity por medio del atributo “USER_NAME” recibe el

nombre de usuario enviado por la clase “UserProfile.class”, para la comparación de dicha variable con el json añadido en el atributo “textViewJSON” y mostrar los datos correspondientes únicamente del usuario logueado, la validación mencionada se la realiza en la clase “MapsActivity.class”. Y el envío se establece por medio de las variables “MY_JSON” y “USER_NAME”.

```
intent.putExtra(MY_JSON,textViewJSON.getText().toString());
intent.putExtra(USER_NAME,username);
```

La clase “MapsActivity.class” recibe la información de la tabla “reg” almacenada en la base de datos y el valor del nombre de usuario ingresado en el método “onCreate” a través de las variables “myJSONString” y “username” (Ver Figura 102).

```
myJSONString = intent.getStringExtra(ConvJsonTxt.MY_JSON);
username = intent.getStringExtra(ConvJsonTxt.USER_NAME);
```

En el método “onCreate” se carga el mapa de Google, es importante añadir el código de verificación en la consola de administración de Google, para la autenticación de la aplicación y así permitir el uso de mapas de Google en la App. (Ver Anexo 1).

```
SupportMapFragment mapFragment = (SupportMapFragment)
getSupportFragmentManager().findFragmentById(R.id.map);
mapFragment.getMapAsync(this);
```

El método “extractJSON” (Ver Figura 102), genera por medio de la variable “jsonObject” del tipo “JSONObject” un objeto en formato “JSON” de la variable “MyJSONString” recuperada, mediante la variable “users” de tipo “JSON_ARRAY” se genera un arreglo con los valores recuperados por la variable “jsonObject”.

```
JSONObject jsonObject = new JSONObject(myJSONString);
users = jsonObject.getJSONArray(JSON_ARRAY);
```

El método “onMapReady” se ejecuta una vez cargado el mapa, y realiza la especificación de los atributos, tales como el tipo de mapa, el zoom, además de la ubicación de los marcadores, en primera instancia se realiza la llamada al método “extractJSON”, para obtener la información del JSON creado por el script de configuración “Obtener.php”. Por medio de una sentencia “FOR”, se accede a cada uno de los elementos en el JSON recuperado y almacenado en la variable “users”.

```
for(int i=0;i<users.length();i++) {
JSONObject jsonObject = users.getJSONObject(i);}
```

Por medio de las variables de tipo String “usuario” y “perro”, se accede a los elementos del JSON a través del objeto “jsonObject”.

```
String usuario = jsonObject.getString(USUARIO);
String perro = jsonObject.getString(PERRO);
```

Mediante la sentencia “IF”, se compara que el nombre de usuario que inicio la sesión sea igual con el objeto “JSON” almacenado en la variable usuario, ya que dicho objeto obtiene la información correspondientes a todos los nombres de usuario almacenados en la tabla “reg” de la base de datos. Si las variables comparadas son iguales, se obtiene por medio del objeto “jsonObject” los valores correspondientes a la latitud y longitud correspondiente al usuario ingresado. Conforme se realiza el barrido en el archivo de tipo JSON, se recupera la información correspondiente a posicionamiento geográfico.

```
if (username.equals(usuario)){
                    double x = jsonObject.getDouble(LATITUD);
                    double y = jsonObject.getDouble(LONGITUD);}
```

Una vez comparada la información, si el algoritmo encuentra una coincidencia de usuario, se procede con el establecimiento de los puntos para el marcador informativo de posición y el título del marcador que corresponde al nombre de la mascota almacenado en la variable de tipo String “perro” y recuperado del Json

alojado en el servidor Web. Se establece la ubicación en el mapa de la posición de la ubicación mediante un marcador, con el zoom predeterminado por el desarrollador.

```
LatLng sydney = new LatLng(x, y);
map.addMarker(new MarkerOptions().position(sydney).title(perro));
float zoomLevel = 10;
map.moveCamera(CameraUpdateFactory.newLatLngZoom(sydney,zoomLevel));
```

La Figura 104 muestra el resultado de un ejemplo de verificación de ubicación a través de un mapa, y que corresponde al layout asociado con la clase.

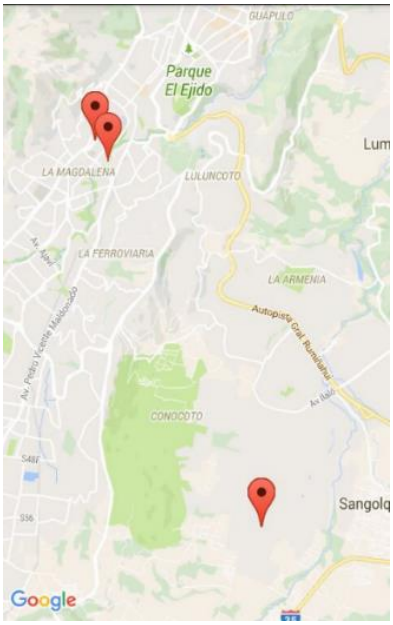


Figura 104 Ubicación en Android

3.4.2.4. Servicio Ver signos Vitales en Android.

El aplicativo dispone de la verificación de signos vitales de las mascotas correspondientes al usuario. El diagrama de flujo de la Figura 105 muestra de forma general el funcionamiento del servicio de verificación de signos vitales.

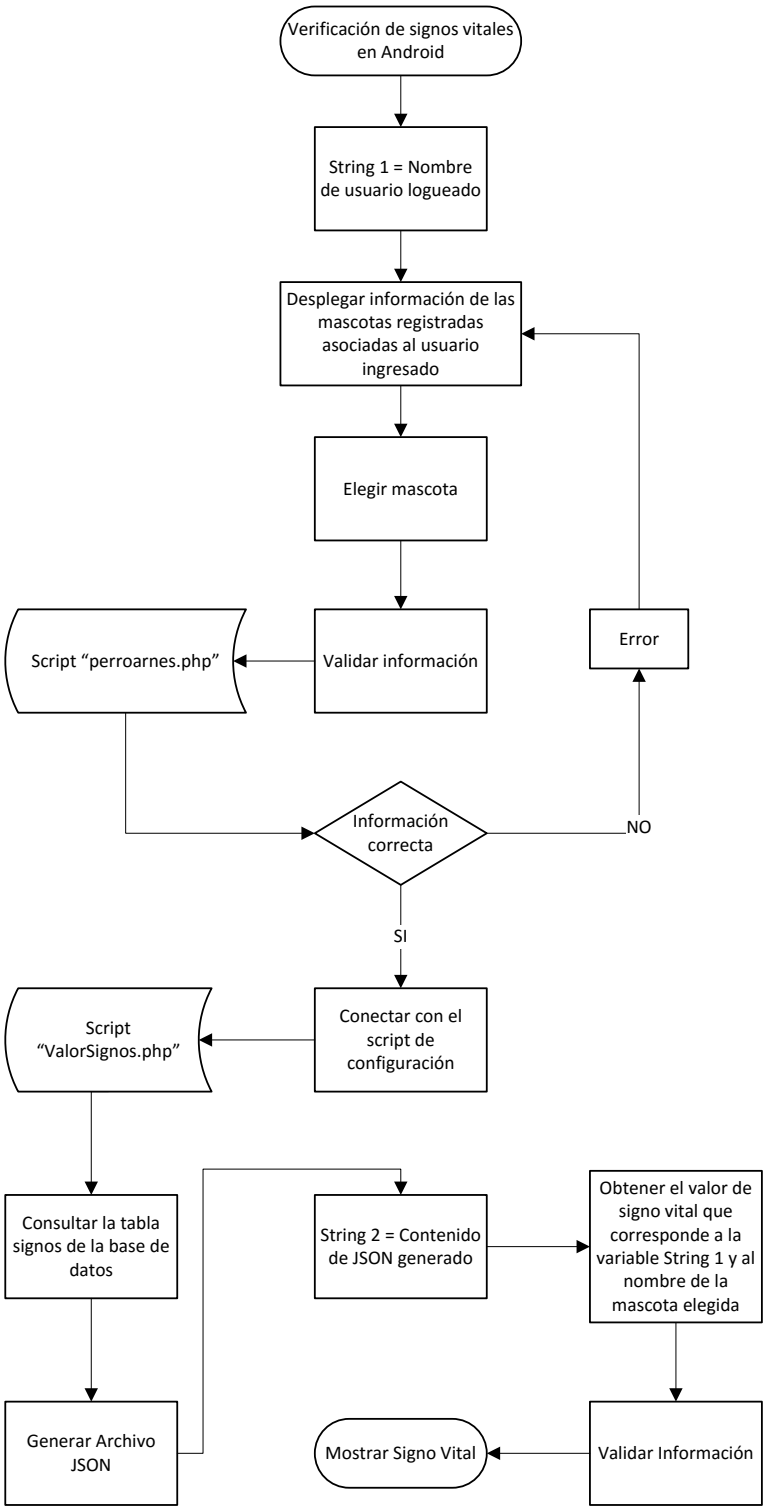


Figura 105 Diagrama de flujo Ver Signo Vital Android

En el perfil de usuario, el usuario elije la opción de verificación de signos vitales por medio de la plataforma Android. La clase "UserProfile.class" envía la información correspondiente al nombre de usuario ingresado, y al JSON de la tabla

"signos", recuperado por el script "ValorSignos.php". Para establecer el funcionamiento deseado se necesitó de la creación de dos clases "FormularioSignos" y "VerSignos" (Ver Figura 106).

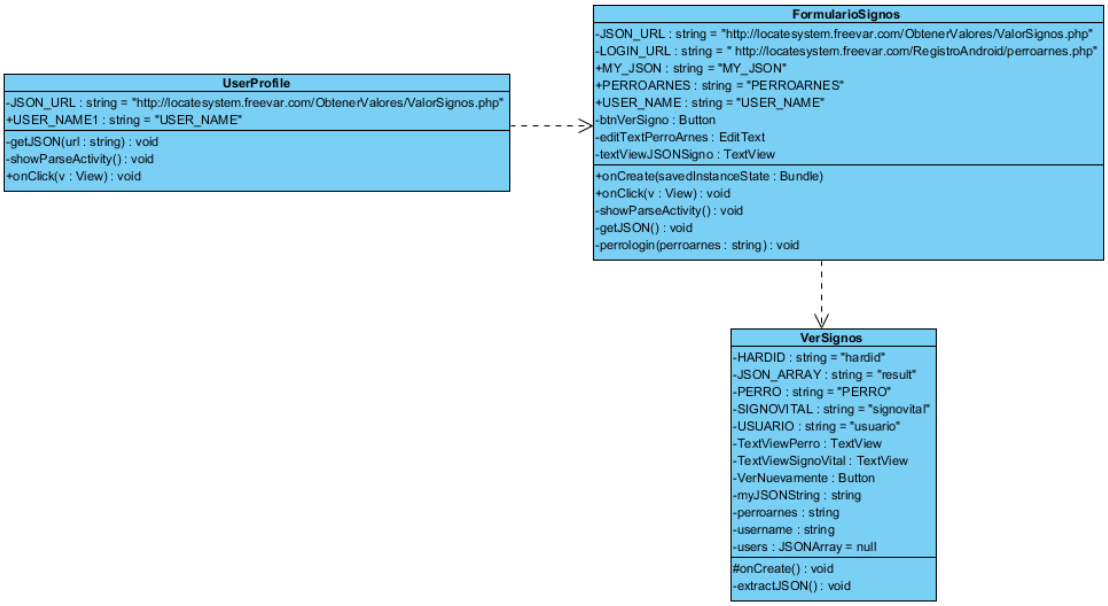


Figura 106 Diagrama de clases Ver Signo Vital Android

La clase "UserProfile.class" envía la información recuperada de la tabla "signos" mediante la variable "MY_JSON_SIGNOS" a la clase "FormularioSignos.class", la variable "MY_JSON_SIGNOS" recupera la información mediante el método "getJSON", el cual consulta al script de configuración "ValorSignos.php" encargado de generar el JSON con los valores recuperados de la tabla "signos".

La clase "FormularioSignos.class", está relacionada con el layout "activity_formsigno.xml" (Ver Figura 107), en donde se dispone la estructura gráfica de la clase. Mientras que la clase "VerSignos.class" se relaciona con el layout "activity_ver_signos.xml" (Ver Figura 107). La clase "FormularioSignos.class", recibe la información del JSON generado por la clase "UserProfile.class", por medio de una validación entre variables se dispone la información correspondiente al nombre de la mascota perteneciente al usuario en botones tipo "radio", para que el usuario visualice las mascotas añadidas en su perfil, y elija la mascota a verificar la información de signos vitales. Los layout mencionados se muestran en la Figura 107.

Figura 107 "Layout activity_formsigno.xml" y "activity_ver_signos.xml"

En la clase "FormularioSignos.class" se integra la funcionalidad de autenticación del nombre de mascota elegida, y de la consulta de la tabla "signos" en la base de datos por medio del script de configuración "ValorSignos.php" en donde se realiza la consulta con la base de datos y se genera el archivo en formato JSON. El método "onCreate" de la clase "FormularioSignos.class" establece la comunicación con el layout relacionado, además de relacionar los componentes tipo "TextView" y "Button" con variables dentro de la clase.

```
btnVerSigno = (Button) findViewById(R.id.btnVerSigno);
textViewJSONSigno = (TextView) findViewById(R.id.textViewJsonSigno);
textViewJSONSigno.setMovementMethod(new ScrollingMovementMethod());
```

En el método "onCreate" también se almacena la información correspondiente al JSON generado en la clase "UserProfile.class". El JSON se genera con los datos almacenados en la tabla "signos" para generar automáticamente los botones tipo "radio" en el layout relacionado con la clase, de acuerdo a las mascotas registradas por el usuario.

El método "perrologin", está conformado por hilos asíncronos, cada hilo realiza una tarea antes, durante, después, para su ejecución se usan los métodos: "onPreExecute", "doInBackground" y "onPostExecute" respectivamente. En el

método “onPreExecute” se muestra al usuario un indicador de “Cargando”, e indica que el sistema se encuentra procesando la solicitud.

```
loading = ProgressDialog.show(FormularioSignos.this,"Please
Wait",null,true,true);
```

En el método “doInBackground” se establece la ejecución de las tareas principales del método, el método “perrologin” valida la opción elegida por el usuario en el layout “activity_form_signo.xml”.

```
HashMap<String,String> data = new HashMap<>();
      data.put("perroarnes",params[0]);
Conectar ruc = new Conectar();
String result = ruc.sendPostRequest(LOGIN_URL,data);
return result;
```

El método “HashMap”, dispone los atributos recibidos por el método “perrologin”, almacenado en la vartiable “perroarnes”, correspondiente al nombre de la mascota que el usuario selecciona en el botón tipo “radio” desplegado (Ver Figura 107), y los representa por medio de una clave, con la cual se identifica el valor de los atributos recibidos, una vez representada la información se procede a realizar el envío por medio del objeto “data” al archivo de configuración perroarnes.php alojado en el servidor web, el cual está almacenado en la variable “LOGIN_URL”, y cuyo valor se aprecia en la Figura 106.

En el método “onPostExecute” se establece las tareas que se realizan posterior a la ejecución del método “doInBackground”, de acuerdo a la respuesta obtenida por el método. La ejecución del método se realiza de acuerdo a la respuesta del script “perroarnes.php”. En la Figura 108 se establece el diagrama de flujo del script de configuración “perroarnes.php”.

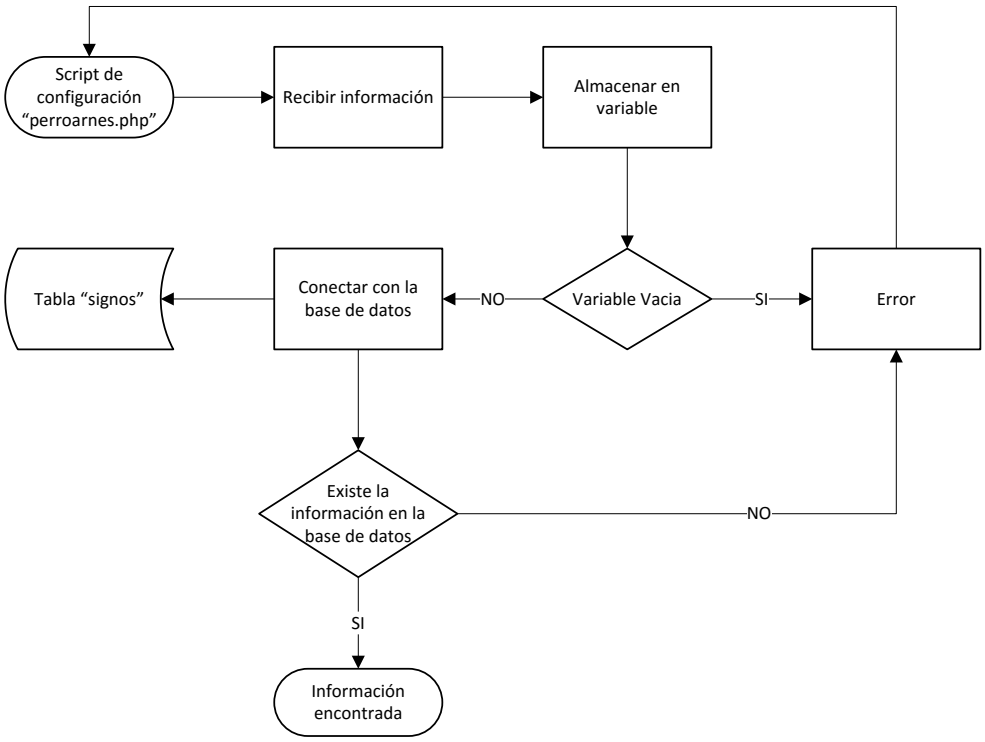


Figura 108 Diagrama de flujo script "perroarnes.php"

El usuario, dentro de la opción de verificación de signos vitales, selecciona el nombre de la mascota registrada, esta información es enviada al script de configuración perroarnes.php almacenada en el servidor web, cuya dirección URL es:

"http://locatesystem.freevar.com/RegistroAndroid/perroarnes.php"

El script recibe la información por medio del método HTTP "POST", y lo almacena en la variable $perroarnes.

```
$perroarnes = $_POST['perroarnes'];
```

Si la información almacenada se encuentra vacía, el script devuelve un mensaje de error, el cual es mostrado en la aplicación Android.

```
if($perroarnes == ''){ echo 'Por favor llena todos los campos';}
```

Si no se cumple la condicional, y existe información en la variable $perroarnes. El script se conecta con la tabla "signos" almacenado en la base de datos, y realiza la

verificación de si la información ingresada existe en la base de datos. La verificación es comparada con los campo “PERRO” de la tabla “signos”.

```
$sql = "select * from signos where hardid='$perroarnes' OR
PERRO='$perroarnes'";
$check = mysqli_fetch_array(mysqli_query($con,$sql));
```

Una vez realizada la validación, por medio de la sentencia “IF”, el script devuelve a la aplicación en Android un mensaje de éxito o error, según si la información fue o no encontrada.

```
if(isset($check)){
echo "success";
}else{ echo "Nombre de mascota o arnes no encontrado";}
```

Si la información ingresada es correcta, el mensaje retornado por el script es “success”, caso contrario el sistema muestra un mensaje de error al usuario. Si el mensaje es “success”, la aplicación ejecuta el método “getJSON”, en el cual se extrae la información de los signos vitales correspondiente a la mascota que el usuario requiere verificar la información.

```
if(s.equalsIgnoreCase("success")){
    getJSON(JSON_URL);     }
else{
    Toast.makeText(FormularioSignos.this, s,
    Toast.LENGTH_LONG).show();}
```

El método “getJSON” recibe la información de la tabla “signos” en la base de datos, la cual por medio del archivo de configuración “ValorSignos.php”, genera un archivo en formato JSON, para la consulta de la aplicación en Android, la estructura del archivo generado por el script “ValorSignos.php” es la siguiente:

```
{"result":[{"id":"7","usuario":"CarlosAndres","temp":"35.24","signov
ital":"22","hardid":"456072","PERRO":"candy","tiempo":"2016-08-01
02:43:26"},
```

{"id":"2","usuario":"CarlosAndres","temp":"35.24","signovital":"40",
"hardid":"586478","PERRO":"sam","tiempo":"2016-07-28 11:33:08"}]}

El método “getJSON”, está conformado por hilos asíncronos, en el cual, el método “onPreExecute”, muestra un mensaje de espera para el usuario, el método “doInBackground” se conecta con el script de configuración “ValorSignos.php” , cuya dirección URL se encuentra almacenada en la variable “LOGIN_URL” (Ver Figura 106).

```
String json;
while((json = bufferedReader.readLine())!= null){
    sb.append(json+"\n");}
```

La información generada por el script en formato JSON, se almacena en una variable del tipo “TextView”, en el método “onPostExecute.

```
textViewJSONSigno.setText(s);
```

Una vez almacenado el valor generado por el script de configuración, se realiza el llamado al método “showParseActivity”, y se envía la información del JSON almacenado, al nombre de usuario ingresado, y el nombre de la mascota seleccionado por el usuario, la información mencionada es usada para realizar la validación de usuario y de los signos vitales, la validación es realizada por la clase “VerSignos.class” (Ver Figura 106).

```
intent.putExtra(MY_JSON,textViewJSONSigno.getText().toString());
intent.putExtra(USER_NAME,username);
intent.putExtra(PERROARNES,perroarnes);
```

La información en formato JSON se almacena en la variable de tipo String “myJSONString”, la cual es reenviada por la clase “FormularioSigno.class”.

```
myJSONString = intent.getStringExtra(FormularioSignos.MY_JSON);
```

El método “extractJSON” ingresa a cada uno de los componentes del JSON almacenado en la variable, por medio del objeto “users” de tipo “JSONArray”, en específico a la etiqueta almacenada en la variable “JSON_ARRAY” de tipo String, cuyo valor se puede verificar en la Figura 106.

```
JSONObject jsonObject = new JSONObject(myJSONString);
users = jsonObject.getJSONArray(JSON_ARRAY);
```

Se recupera los valores consultados por la clase “FormularioSignos.class”, correspondiente a el nombre de usuario, y el nombre de la mascota.

```
username = intent.getStringExtra(Ingresar.USER_NAME);
perroarnes = intent.getStringExtra(FormularioSignos.PERROARNES);
```

Para realizar la validación de los signos vitales, el algoritmo requiere recuperar ciertos elementos del JSON generado por el script de configuración, por medio de una sentencia “FOR”, se ingresa a cada uno de los elementos almacenados en el JSON, y se recupera específicamente lo correspondiente al nombre de usuario, al valor del signo vital, identificador del arnés y nombre de la mascota almacenada en la tabla “signos” en la base de datos.

```
JSONObject jsonObject = users.getJSONObject(i);
String usuario = jsonObject.getString(USUARIO);
String perro = jsonObject.getString(PERRO);
String hardid = jsonObject.getString(HARDID);
```

Una vez recuperada la información, se procede a realizar la validación, por medio de la sentencia “IF”, se compara los valores ingresados por el usuario, correspondiente al nombre de usuario y al nombre de la mascota, con los valores recuperados de la base de datos, y generados en el archivo en formato JSON. Únicamente se recupera los valores del usuario que inicio sesión, y del nombre de la mascota del usuario correspondiente para la verificación, si se cumple la condición, el sistema valida si la mascota se encuentra con signos vitales, de acuerdo al valor almacenado en la base de datos, además proporciona un mensaje de en el

“TextView” y una imagen de tipo “ImageView” distinta de acuerdo al resultado de la validación.

```
if (username.equals(usuario) &&
(perroarnes.equals(perro)||perroarnes.equals(hardid))){

        int j = jsonObject.getInt("signovital");

        TextViewPerro.setText(perro);

        if (j<11){
      TextViewSignoVital.setText("No hay movimiento  "+tiempo+
      "  Temperatura:"+temperatura+ "°C");
          img.setImageResource(R.drawable.corazonazul);

        }

        else {
      TextViewSignoVital.setText("Si existe movimiento   "+tiempo+"
      Temperatura:"+temperatura+ "°C");
          img.setImageResource(R.drawable.corazones11);

           }}
```

Toda la información mostrada en los terminales y enviada por medio del hardware (Arnés para perros), se validó su funcionamiento, la etapa de pruebas establece la precisión, ventajas y limitaciones que posee el proyecto.

CAPÍTULO IV

4. ESCENARIOS DE PRUEBA

La etapa de pruebas de funcionamiento, establece por medio de escenarios la precisión de la información recuperada por el arnés ubicado en el animal, mediante los valores obtenidos, se establece las ventajas y limitaciones del proyecto. El arnés para mascotas se desarrolló de una sola medida, y por el tamaño establecido se propone el uso del arnés únicamente para razas de perro medianas y grandes, se diseñó una caja con los componentes que el proyecto requiere para su funcionamiento, el diseño visual de la caja que va ubicada en el arnés se puede verificar en la Figura 109.

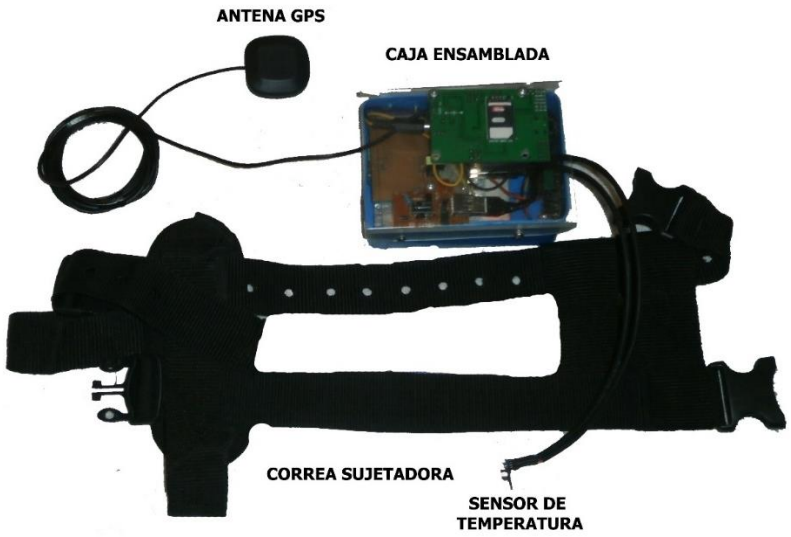


Figura 109 Diseño Visual de caja y arnés

En la Figura 109 se muestra los componentes del sistema de hardware diseñado, la descripción de cada parte son:

- Caja ensamblada, contiene las placas que conforman los circuitos que se encargan de censar el movimiento de la mascota, del envío de información mediante tecnología GSM/GPRS, recepción de señal GPS, circuito de carga y alimentación de las placas.
- Sensor de temperatura, el sistema contiene un sensor de temperatura infrarrojo, el cual mide la temperatura externa del animal, y su disposición en

el arnés debe ser de tal manera que se mantenga fija hacia el animal, y mida su temperatura de forma constante, sin importar su movimiento, su conexión es por medio de la plataforma programable Arduino UNO.

- Antena GPS, obtiene las señales de posicionamiento geográfico, latitud y longitud, su disposición en el arnés debe estar en la parte externa superior de la mascota para no tener dificultades con obstáculos y obtener las señales de posición de manera correcta, su conexión es mediante el módulo SIM 908.
- Correa sujetadora, mediante la correa sujetadora, se sostienen la caja y los componentes en la mascota.

Las características de la caja se disponen según la necesidad del proyecto, se lo estableció de la manera más compacta posible, de manera que sea lo menos incómodo para la mascota. Las medidas de la caja se puede observar en la Figura 110.

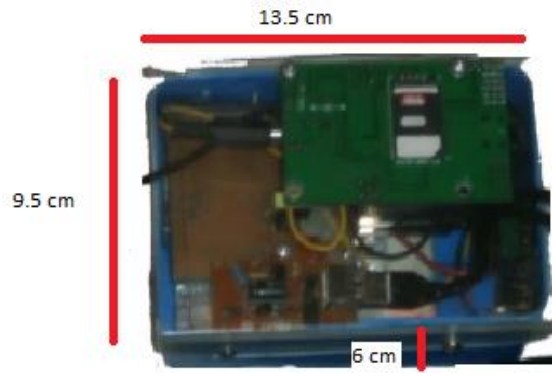


Figura 110 Medidas de la caja contenedora de los componentes de hardware

La Tabla 15 muestra un resumen de las características físicas de longitud, alto, ancho y peso de la caja.

Tabla 15

Características físicas de caja para el arnés

Medida	Valor
Peso/Masa	482 gramos
Longitud	13,5 cm
Alto	6 cm
Ancho	9,5 cm

En la Tabla 15 se resume las medidas de la caja que va ubicada en el arnés para mascotas, mediante las medidas recuperadas y el diseño realizado, se puede establecer el tipo de mascota, y raza de perro que la puede utilizar, además del diseño del arnés capaz de sostener los componentes que se encargan de realizar las tareas establecidas por el proyecto. La caja diseñada se la ubica en un arnés de acuerdo al animal que se desea monitorear, se puede diseñar un arnés para un animal distinto a un perro, como por ejemplo un caballo, pero el acceso a una mascota de las magnitudes y tamaño mencionado es limitado, también por las medidas de la caja es imposible ubicar en una mascota muy pequeña como un hámster, por lo que el arnés se diseñó para su uso en animales con mayor accesibilidad, y los más comunes en hogares, los perros. Se tiene que tomar en cuenta que existen diversas razas de perro, por lo que se realizó pruebas de ubicación y comodidad en perros de distintas razas, y tamaño. Mediante la etapa de pruebas se puede establecer que el arnés puede ser usado en razas de perro mediano y grande (Ver Figura 111).

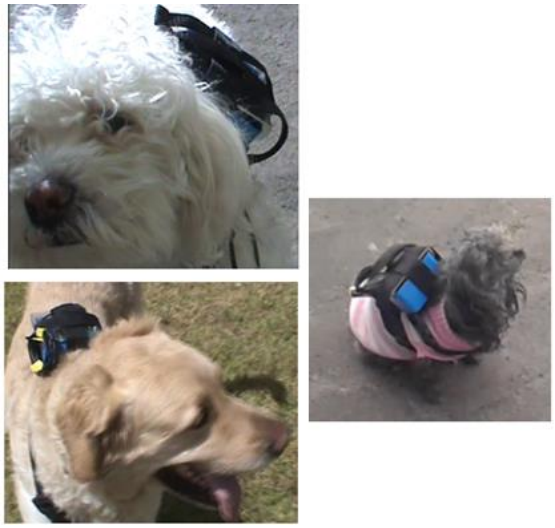


Figura 111 Establecimiento del arnés en mascotas

En la Figura 111 se muestra los tipos de razas de perro que se usaron para establecer el arnés y realizar las pruebas, como se observa en la Figura 111, el arnés se establece de manera cómoda en animales medianos y grandes, es posible establecerlo en animales pequeños, pero es incómodo y limita la movilidad del animal.

4.1. Sujetos de prueba

La etapa de pruebas se la realizó en 3 perros de distintos tamaños y razas. En la Tabla 16 se muestra las características de los perros utilizados en la etapa de pruebas.

Tabla 16

Sujetos de Prueba

ID	Nombre	Tamaño	Raza
M1	Sam	Grande	Golden
M2	Pelusa	Mediano	Castellano
M3	Candy	Pequeño	French P71oodle

El arnés es posible ubicarlo en perros de distintas razas, pero su tamaño lo limita en comodidad en las razas pequeñas. La macota con ID “M3” (Ver Tabla 16), es una raza de perro pequeña, fue posible su implementación, pero se notó una incomodidad marcada en el animal. Además el peso de la caja limita a la mascota en su movilidad. La Figura 112 muestra la implementación del arnés en la mascota mencionada.

Figura 112 Arnés en sujeto de prueba “M3”

El perro con ID “M2”, es un tipo de raza de perro mediana, el arnés se lo ubicó en este tipo de animales de manera acertada, demostrando que se puede realizar la implementación en razas de perro de tamaño mediano. La Figura 113 muestra la ubicación del arnés en razas de perro mediano.

Figura 113 Arnés en sujeto de prueba “M2”

La mascota con ID “M1” es una raza de perro grande, la implementación del arnés en este tipo de mascota no presentó inconvenientes, y el animal no mostro signos de incomodidad en el transcurso de su uso. La Figura 114 muestra el uso del arnés en el tipo de mascota mencionado.

Figura 114 Arnés en sujeto de prueba “M1”

4.2. Consumo de datos.

El módulo SIM 908 por medio de la tarjeta programable Arduino UNO realiza el envío de información usando tecnología GSM/GPRS, por lo que requiere la implementación de un chip de telefonía celular con datos incluidos, los planes disponibles para la implementación son varios, en este caso por motivo de accesibilidad se usó un plan de datos de la empresa de telefonía celular Claro, la cual es esencial para el envío de la información recuperada por el arnés. Para conocer el consumo de datos del proyecto se realizó los siguientes pasos:

1. Se realiza una consulta de saldo de datos en el chip por medio de un dispositivo móvil.
2. Se instala el chip en el arnés para mascotas diseñado.
3. Se mantiene el dispositivo funcionando.
4. Después de un tiempo prudente, se apaga el dispositivo, para recuperar el chip.
5. Se realiza una nueva consulta de saldo desde un teléfono móvil.

Los valores recuperados se muestran en la Tabla 17.

Tabla 17

Prueba de consumo de datos

Hora Inicial de consulta	Hora Final de consulta	Datos consumidos	Tiempo de Encendido
20h50	23h50		3 horas
998.17 Mb	996,9 Mb	1.27 Mb	

Por medio del análisis realizado, se establece el consumo de datos diario, semanal y mensual, el funcionamiento constante del arnés por un tiempo máximo de un mes, va a consumir los valores mostrados en la Tabla 18.

Tabla 18

Consumo de datos

Tiempo de uso	Datos consumidos
3 horas	1.27 Mb
Un día (24 horas)	10.16 Mb
Una semana (168 horas)	71.12 Mb
Un mes (720 horas)	304.18 Mb

Por medio de los valores recuperados en la Tabla 18, se puede conocer los planes mínimos de datos disponibles en el mercado que cumplen con los requisitos mínimos para no tener problemas con los datos disponibles en el arnés y con el envío de información. En la Tabla 19 se muestra un resumen de los planes disponibles en el mercado de las principales telefónicas que dan servicio de telefonía en el Ecuador.

Tabla 19

Resumen de planes de datos recomendados por el consumo del proyecto

Empresa de Telefonía	Megas incluidos	Costo Mensual
CNT	500 Mb/mes	$9.99 + IVA
CLARO	500 Mb/mes	$15 + IVA
Movistar	500 Mb/mes	$15 + IVA

Con la información correspondiente al consumo de datos recuperada por medio del chip establecido en el arnés, se puede conocer si los planes mostrados en la Tabla 19 cumplen con los requisitos para que el proyecto no tenga problemas con la disponibilidad de datos. Mediante el análisis realizado, el consumo de datos máximo en un mes es de 304,18 Mb, en la Tabla 19 se muestra que los planes que ofrecen las operadoras dan un mínimo de 500 Mb/mes, por lo que cualquiera de los planes mostrados en la Tabla 19 pueden cubrir con lo que el sistema necesita para un funcionamiento sin problemas con respecto al consumo de datos.

4.3. Escenarios de Prueba

Mediante diversos escenarios de prueba, es posible recolectar la información obtenida y verificar la operación del arnés en diferentes ambientes. Los escenarios se establecieron de acuerdo a la accesibilidad que se tuvo a los mismos. Los ambientes en donde se realizaron las pruebas fueron:

- Parque.
- Casa.
- Calle.
- Auto.
- Ambiente Variable.

Es importante conocer el funcionamiento del arnés en diversos ambientes, cada escenario tiene su particularidad, difiere en los obstáculos que pueden influir en la recuperación de información de posicionamiento. La temperatura ambiente también es un factor que varía de acuerdo al escenario, y lo cual puede llegar a influir en el

valor de temperatura externa recuperada del animal. Los escenarios para pruebas fueron establecidos en distintas situaciones climáticas y tiempos, para conocer la influencia del estado climático en la recuperación de cierta información.

4.3.1. Escenario 1: Parque.

Descripción: El Escenario 1, corresponde a un parque, las características del parque muestra un ambiente abierto, el suelo del lugar en donde se realizó la prueba en su mayoría es cementado con pequeños lugares de césped, el parque consta de poca vegetación, con pequeños árboles en los alrededores, postes y canchas de varios deportes, en si el escenario no muestra mayores obstáculos que dificulten en la recuperación de información de posicionamiento. La Figura 115 muestra una imagen del escenario descrito.

Figura 115 Escenario 1

El lugar correspondiente al Escenario 1 es el parque Santa Ana 1 de la ciudad de Quito, las condiciones climáticas y de tiempo, para la realización de la prueba fueron:

- Cielo despejado.
- Hora 11 am – 12 pm.
- Mascota en movimiento.
- Temperatura: 23 °C

Ubicación: Los factores que pueden llegar a influir en los valores de posicionamiento recuperados, son los obstáculos que el escenario presenta, las condiciones climáticas, las cuales pueden influir en la información que la antena GPS

recupera. Las condiciones que el escenario presenta se puede asumir que no influyen en la recuperación de información de posicionamiento, ya que la prueba fue realizada en un escenario con poca vegetación en los alrededores, y con un cielo despejado. La precisión recuperada en el Escenario 1 se muestra en la Tabla 20.

Tabla 20

Precisión de Ubicación en Escenario 1

N° de prueba	Escenario	Sujeto de prueba	Precisión	Tiempo de uso
1	Parque	M2	2 metros	1 min
2	Parque	M2	3 metros	4 min
3	Parque	M2	1 metro	7 min
4	Parque	M2	2 metros	9 min
5	Parque	M2	2 metros	11 min

Como se muestra en la Tabla 20, los obstáculos presentados en el Escenario 1 no influyen en la obtención de posicionamiento geográfico recuperado por el arnés. Otro factor a tomar en cuenta es la movilidad de la mascota, por medio de la información recuperada (Ver Tabla 20) se establece que el movimiento normal de la mascota no presenta dificultades para que el arnés recupere con una precisión aceptable la información de posicionamiento geográfico. Los datos de longitud y latitud recuperada en el Escenario 1 se muestran en la Tabla 21.

Tabla 21

Coordenadas de ubicación en Escenario 1

N° de Prueba	**Sujeto de prueba**	**Latitud**	**Longitud**
1	M2	-0.239177	-78.520325
2	M2	-0.239209	-78.520244
3	M2	-0.239262	-78.520267
4	M2	-0.239285	-78.520183
5	M2	-0.239392	-78.520204

La Tabla 21 muestra las coordenadas geográficas recuperadas en la etapa de pruebas en el Escenario 1, la precisión en las condiciones establecidas son bastante precisas, ya que para que la antena GPS recupere una buena precisión se requiere que

la mascota se encuentre al aire libre. Como se muestra en la Tabla 20, la precisión recuperada en este escenario es de 1m – 3m, precisión bastante acertada para el propósito del arnés, por lo que los obstáculos presentados en este tipo de escenarios no influyen en obtener la ubicación de forma precisa. El tiempo de recuperación de información GPS varía de 30 segundos a 1 minuto, no representó un problema al momento de realizar las pruebas, ya que el tiempo de recuperación de información es bajo y el sistema retorna lo antes posible la información recuperada.

Movimiento: El movimiento del arnés fue la medida recuperada para informar al dueño si su mascota se encuentra moviéndose, el acelerómetro ubicado en el arnés retorna los cambios que el animal muestra respecto a su movimiento, el sensor al percibir movimiento retorna un valor variable y diferente al que el acelerómetro envía en reposo, el cual es un valor estable. La validación de información se la realiza mediante la resta del mayor número con el menor número recuperado por el acelerómetro, la cantidad de muestras analizadas y con las que se realiza la validación es 10. Mediante las pruebas realizadas se establece que la diferencia del mayor número con el menor número retornado por el sensor es la adecuada para validar el movimiento del arnés. La tabla “signos” en la base de datos aloja la información de signos vitales correspondiente al movimiento, la diferencia entre los valores recuperados debe ser mayor a 10, ya que la sensibilidad del sensor es muy alta, y si no se pone un valor base un poco elevado, se puede informar de forma errónea al usuario. La Tabla 22 muestra los valores recuperados con la mascota en movimiento.

Tabla 22

Variación acelerómetro con mascota en movimiento

Variación Acelerómetro	Estado de mascota	Sujeto de Prueba
120	Caminando	M2
90	Caminando	M2
98	Caminando	M2
110	Caminando	M2
135	Caminando	M2
89	Caminando	M2
105	Caminando	M2
114	Caminando	M2
92	Caminando	M2

El arnés retorna valores que corresponden a la vibración del acelerómetro, cuando una mascota se mueve se recibe valores aleatorios, y con una marcada diferencia entre ellos (Ver Tabla 22). Los valores censados se almacenan en la tabla "reg" de la base de datos, mediante esta tabla se realiza la validación de la información, por medio de la diferencia entre los valores almacenados en la Tabla "reg". En la Tabla 20 se muestra una parte de la información recuperada en el Escenario 1. El mayor valor almacenado en la Tabla 22 es 135, y el menor es 89, la resta entre el mayor con el menor valor es 46. Para conocer el comportamiento y los valores retornados por el arnés sin movimiento se lo ubica en un objeto estático y se obtiene los valores que se muestra en la Tabla 23.

Tabla 23

Variación acelerómetro en reposo

Variación Acelerómetro	Estado de mascota	Sujeto de Prueba
100	Sin movimiento	M2
98	Sin movimiento	M2
99	Sin movimiento	M2
101	Sin movimiento	M2
100	Sin movimiento	M2
100	Sin movimiento	M2
98	Sin movimiento	M2
99	Sin movimiento	M2
101	Sin movimiento	M2
100	Sin movimiento	M2

En la Tabla 22 y Tabla 23 se muestra los valores recuperados por el sensor acelerómetro ubicado en el arnés en el Escenario 1, se muestra los valores del arnés en una mascota y en un objeto en reposo para simular la falta de movimiento del animal respectivamente, la diferencia entre el mayor valor y el menor valor recuperado en la Tabla 23 es 3, el acelerómetro tiene una alta sensibilidad por lo que a pesar que el arnés este completamente en reposo recupera una variación de movimiento mínima, por lo que en las terminales se establece que la diferencia entre los valores mínimos y máximos recuperados sea mayor a 10 para establecer un movimiento en el arnés.

Temperatura: Otra medida importante que informa al usuario si la mascota se encuentra con vida en una situación determinada es la temperatura. La temperatura se recupera por medio de un sensor de temperatura infrarrojo, el cual mide únicamente la temperatura externa en el animal, aunque la medida recuperada no equivale a una medición médica, es decir que no nos permite conocer si el animal se encuentra con fiebre o enfermo, ya que la única forma de conocerlo es por medio de medición de temperatura vía rectal, la información recuperada informa al usuario si la mascota se encuentra con temperatura corporal, lo cual es un signo vital que informa al usuario si el animal se encuentra con signos vitales.

La temperatura externa de una mascota depende de diversos factores, como la temperatura ambiental, la raza del perro, el ejercicio físico de la mascota, etc. Por lo que conocer la temperatura médica real de una mascota requiere de métodos invasivos con el animal. La Tabla 24 muestra los valores de temperatura recuperados en el Escenario 1, junto con la temperatura ambiental.

Tabla 24

Temperatura corporal en Escenario 1

Temperatura Ambiente	Temperatura corporal externa	Sujeto de prueba
23 °C	34.6 °C	M2
23 °C	33.8 °C	M2
23 °C	33.9 °C	M2
23 °C	34.3°C	M2
23 °C	34.5 °C	M2

A pesar que la temperatura no es una medida médica real, puede ayudar al usuario a identificar fácilmente si el animal emana temperatura corporal, lo que demostraría que la mascota se encuentra con signos vitales. La temperatura corporal del animal junto con el movimiento de la mascota son medidas que juntas informan al dueño de la mascota si ésta se encuentra bien, en una situación determinada.

4.3.2. Escenario 2: Interior de Casa

Es importante conocer la precisión del arnés en ambientes cerrados, el interior de una casa cumple los requisitos para realizar este tipo de pruebas.

Descripción: El ambiente en donde se realizaron las pruebas tiene las siguientes características:

- Paredes cementadas.
- Techo de hormigón.
- Ancho de la pared de 15 cm – 20 cm.
- Ventanas y puertas en los alrededores.
- Una parte de la casa va hacia un ambiente abierto en este caso un patio.
- La segunda parte de la casa va hacia un ambiente cerrado en este caso otro cuarto.

El lugar correspondiente al Escenario 2, es el interior de una casa ubicada en el sector “Los Dos Puentes”, al Sur de Quito. Las condiciones del Escenario 2 fueron:

- Hora: 18h00 – 19h00.
- Cielo nublado.
- Lugar cerrado.
- Temperatura: 21 °C

En la Figura 116 se muestra una imagen del lugar correspondiente al Escenario 2 y en donde se realizaron las pruebas.

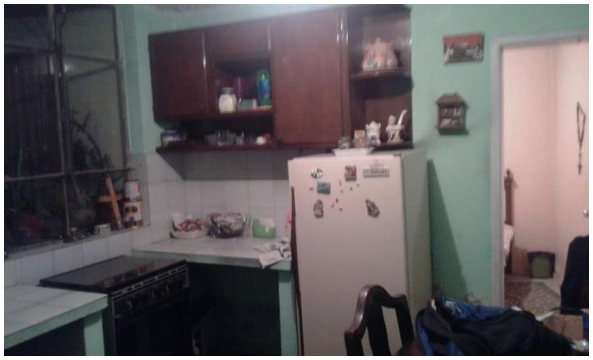


Figura 116 Escenario 2

Ubicación: Se realizó pruebas en el interior de una casa, para conocer la precisión del arnés en un ambiente cerrado. Los datos recuperados se pueden verificar en la Tabla 25.

Tabla 25

Precisión de Ubicación en Escenario 2

N°	Escenario	Sujeto de prueba	Precisión	Tiempo de uso
1	Casa	M3	-	2 min
2	Casa	M3	-	10 min
3	Casa	M3	-	15 min
4	Casa	M3	+ 100 metros	17 min
5	Casa	M3	+ 100 metros	19 min
6	Casa	M3	+100 metros	21 min
7	Casa	M3	+100 metros	23 min

La precisión recuperada en el Escenario 2 demuestra que la precisión obtenida por la antena GPS depende de las condiciones en donde se realiza la prueba, la obtención de información de las coordenadas geográficas tomó un aproximado de 17 minutos en recuperar la información, un tiempo muy alto para el establecimiento de posicionamiento por medio de la antena GPS, el motivo del tiempo de recuperación de información alto es porque el ambiente cerrado dificulta a la antena GPS visualizar o receptar la información enviada por los satélites, además del tiempo de establecimiento de posición alto, la precisión es mayor a 100 metros, lo cual muestra que la precisión no es la adecuada para ubicar de forma correcta a un animal. Las coordenadas de ubicación del lugar en donde se realizaron las pruebas son:

Latitud: -0.233609

Longitud: -78.523616

Las coordenadas recuperadas por el arnés en el Escenario se muestran en la Tabla 26.

Tabla 26

Coordenadas Geográficas Escenario 2

N° de Prueba	Sujeto de prueba	Latitud	Longitud
1	M3	-	-
2	M3	-	-
3	M3	-	-
4	M3	-0.233393	-78.522299
5	M3	-0.233545	-78.522221
6	M3	-0.233609	-78.522093
7	M3	-0.233781	-78.522168

La precisión recuperada en este tipo de escenario, es limitada, ya que la antena no puede encontrar todos los satélites que retornan las coordenadas geográficas mediante triangulación satelital, los obstáculos dificultan obtener una precisión acertada en este tipo de escenarios. Es importante recibir la información de al menos tres satélites de forma simultánea para que mediante triangulación satelital se recupere una posición acertada. Los obstáculos dificultan la recepción de los satélites, y el motivo de porque la precisión no es la acertada es porque el sistema no recupero el número de satélites mínimo para una triangulación satelital adecuada.

Movimiento: La medida de movimiento recuperado en ambientes cerrados es importante para que el dueño de una mascota, sepa si el animal se encuentra con signos vitales, es decir en movimiento y con una temperatura normal. Como una situación de ejemplo, si una persona se va de viaje, puede verificar si el animal se encuentra en movimiento, o si encuentra una anomalía, el dueño tiene la oportunidad de comunicarse con alguna persona para que verifique si el animal se encuentra bien. Las medidas de movimiento recuperado por el sensor acelerómetro establecido en el arnés se muestra en la Tabla 27. El movimiento de la mascota fue leve, ya que permaneció sentado.

Tabla 27

Variación acelerómetro con mascota sentada

Variación Acelerómetro	Estado de mascota	Sujeto de prueba
105	Sentado	M3
92	Sentado	M3
95	Sentado	M3
100	Sentado	M3
90	Sentado	M3
92	Sentado	M3
94	Sentado	M3
102	Sentado	M3
101	Sentado	M3

A pesar que la mascota se encuentra sentada, lo que no es un movimiento representativo (Ver Tabla 27), se puede verificar que el sensor detecta pequeñas variaciones de movimiento normal en un animal, porque a pesar de que se encuentra sentado, existen diversos factores como la respiración, y un cambio de posición repentino que retornan un cambio en los valores receptados por el sensor. La manera que el sistema valida si el animal se encuentra moviéndose o no, es restando el mayor valor retornado por el arnés con el menor valor, en la Tabla 27 el valor mayor es 105 y el menor es 90, la diferencia de los valores mencionados es de 15, el número es un valor mayor a 10 lo que indicaría al usuario que la mascota tiene movimiento. El sistema retorna una vibración mínima en el arnés, lo que a pesar de que el animal está sentado retorna vibraciones que informan al usuario que la mascota se encuentra con movimiento mediante la validación mencionada.

Temperatura: Los valores de temperatura en un ambiente cerrado, son distintas a la de un ambiente abierto, ya que las distintas partes del cuerpo presentan temperaturas distintas, debido, principalmente, a la temperatura ambiental. Por falta de situaciones climáticas externas, como viento y frío, se recupera un valor de temperatura con mayor estabilidad, ya que las situaciones climáticas externas no afectan en la temperatura del animal. La Tabla 28 muestra los valores de temperatura recuperados.

Tabla 28

Temperatura corporal en Escenario 2

Temperatura Ambiente	Temperatura corporal externa	Sujeto de prueba
21 °C	31.3°C	M3
21 °C	31.9 °C	M3
21 °C	31.4°C	M3
21 °C	31.9°C	M3
21 °C	31.4°C	M3

Como se ha venido mencionando, para conocer la temperatura médica de una mascota y su estado de salud es necesario la medición mediante un termómetro especializado y realizarla vía recta, la temperatura recuperada en un ambiente cerrado presenta menores variaciones, ya que el ambiente obstaculiza que las situaciones climáticas externas influyan en la temperatura corporal externa de la mascota.

4.3.3. Escenario 3: Ciudad.

Un escenario habitual para la pérdida de mascotas es la ciudad, este escenario puede imponer distintos peligros en un animal. Una mascota que se encuentre perdida en las calles de una ciudad, se arriesga a: ser atropellado, robado, maltratado por personas desconocidas. Fue importante la realización de un análisis en este tipo de escenarios, ya que posee distintos factores como edificios y árboles, que pueden influir en la precisión de ubicación obtenido por el arnés.

Descripción: El lugar correspondiente al Escenario 3 es una ciudad, las condiciones presentadas en el Escenario, muestra casas de pequeña y mediana medida en los alrededores, edificios pequeños, pocos árboles, postes, cables de luz y de teléfono. Las calles en donde se realizaron las mediciones en su mayoría son cementadas con pequeñas irregularidades y veredas partidas. Las condiciones del Escenario 3 son:

- Cielo semi-despejado.
- Hora: 15h00 – 16h00
- Temperatura. 21°C

- Lugar abierto.

El Escenario 3 corresponde al sector “Santa Ana” en el Sur de la ciudad de Quito, en la Figura 117 se muestra una imagen del lugar correspondiente al Escenario 3.

Figura 117 Escenario 3

Ubicación: Una vez conocidas las condiciones del Escenario 3, se procede con la recuperación de información GPS. En la Tabla 29 se muestra la información correspondiente a la precisión GPS recuperada en el Escenario 3.

Tabla 29

Precisión GPS en Escenario 3

N°	Escenario	Sujeto de prueba	Precisión	Tiempo de uso
1	Ciudad	M2	4 m	2 min
2	Ciudad	M2	2 m	4 min
3	Ciudad	M2	2 m	8 min
4	Ciudad	M2	2 m	10 min
5	Ciudad	M2	3 m	19 min

En la Tabla 29Tabla 29 muestra que la precisión en un ambiente como la ciudad es bastante exacta, lo que establece que el uso del arnés es propicio para este tipo de escenarios, y que los edificios, casas y árboles en los alrededores no afectan en la precisión GPS recuperada por el dispositivo. Las coordenadas de ubicación en donde se realizó la prueba son:

Latitud: -0.2377596

Longitud: -78.521833

En la Tabla 30 se muestra las coordenadas recuperadas en las pruebas realizadas en el Escenario 3.

Tabla 30

Coordenadas Geográficas Escenario 3

N° de Prueba	Sujeto de prueba	Latitud	Longitud
1	M2	- 0.237509	- 78.521757
2	M2	- 0.237831	- 78.521834
3	M2	- 0.238043	- 78.521741
4	M2	- 0.238083	-78.520891
5	M2	- 0.237343	-78.520977

Las coordenadas recuperadas establecen una precisión más que aceptable en este tipo de escenario, los valores de latitud y longitud concuerdan de una forma muy cercana con la posición recuperada por el GPS de un teléfono celular, con el que se recuperó y se comparó la precisión con las medidas obtenidas. Los obstáculos demostrados en este tipo de escenario se demuestra que no son influyentes en la recuperación de información, y se puede verificar que la antena GPS únicamente requiere visibilidad al cielo para recuperar valores de posicionamiento geográfico acertado. Los obstáculos presentados en este Escenario no interrumpen la vista de la antena GPS hacia el cielo, aunque se presentaron varios obstáculos terrestres, estos no presentaron inconvenientes en la visibilidad del arnés hacia los satélites y recuperación de información acertada.

Movimiento: El movimiento de la mascota presentado en este tipo de escenarios fue caminar y en pequeñas ocasiones correr de forma pausada y lenta, las variaciones de movimiento no varía mucho por escenario, pero es importante recuperar la información. En la Tabla 31 se muestra la información de variación del acelerómetro recuperada por el arnés.

Tabla 31

Variación acelerómetro en Escenario 3

Variación Acelerómetro	Estado de mascota	Sujeto de Prueba
80	Caminando	M2
95	Caminando	M2
99	Caminando	M2
130	Corriendo	M2
105	Corriendo	M2
94	Caminando	M2
110	Caminando	M2
92	Caminando	M2
100	Caminando	M2

Como se muestra en la Tabla 31, cuando existe mayor movimiento en el arnés los valores receptados y enviados por el acelerómetro varían en un rango mayor, la variación de los valores recuperados y mostrados en la Tabla 31, es 50, una variación muy amplia y la cual informa al dueño de la mascota que el animal se encuentra en movimiento, se debe tener en cuenta que la primera vez que el arnés se enciende, va a demorar un estimado de tiempo para tener un número de muestras igual a 10, y poder realizar la validación que realiza el terminal. Al obtener mayor vibración el acelerómetro recupera valores mayores o más fuera del rango que en una caminata normal.

Temperatura: Al igual que en los escenarios analizados con anterioridad, la temperatura corporal externa del animal depende de los factores climáticos que el escenario presenta, además de las diferentes variaciones normales: aumenta durante el día y desciende por la noche. El esfuerzo, la excitación y la exposición prolongada a ambientes cálidos o húmedos, también influyen en los cambios de la temperatura. Los valores de temperatura recibida se muestran en la Tabla 32.

Tabla 32

Temperatura corporal en Escenario 3

Temperatura Ambiente	Temperatura corporal externa	Sujeto de prueba
21 °C	32.5°C	M2
21 °C	31.9°C	M2
21 °C	32.3°C	M2
21 °C	32.6°C	M2
21 °C	31.9 °C	M2

Los valores recuperados en la Tabla 32 muestra la variación de temperatura en un ambiente afectado por diversos factores externos, si es que los factores cambian, los mismos influyen en la temperatura corporal del animal, por ejemplo si existe lluvia, muy probablemente la temperatura del perro baje considerablemente. No se realizó pruebas en un escenario lluvioso porque se quiso evitar el daño de los componentes eléctricos dispuestos en la caja, a pesar que la caja fue diseñada para resguardar a los componentes de la forma más impermeable posible. Muy probablemente la temperatura del can en ambientes lluviosos descenderá de forma considerable.

4.3.4. Escenario 4: Auto.

Un escenario ideal para la medición de la precisión del arnés en movimiento es por medio de un vehículo, es importante conocer la precisión de la ubicación en altas velocidades, para conocer las limitaciones y ventajas del arnés. Las causas por las que se realiza la validación en este tipo de escenarios, es principalmente para saber la precisión del arnés con un movimiento en altas velocidades. La velocidad del vehículo vario de 50 Km/h a 60 Km/h y 80 Km/h en vía rápida. La Tabla 33 muestra un resumen de la información recuperada en el Escenario 4.

Tabla 33

Datos escenario 4

N°	Escenario	Sujeto de prueba	Ubicación aproximada	Tiempo de uso
1	Auto	M1	SI	5 min
2	Auto	M1	SI	10 min
3	Auto	M1	SI	15 min
4	Auto	M1	NO	20 min
5	Auto	M1	SI	21 min
6	Auto	M1	SI	25 min

En los datos recuperados mediante la prueba realizada en el Escenario 4, se fue testeando y comparando la posición del puntero en la aplicación Android, con respecto al movimiento del vehículo, la precisión se obtuvo de manera visual, se pudo determinar que en un tramo determinado no se obtuvo la posición marcando una pequeña estabilidad en el sistema, pero en un pequeño tiempo la posición se recuperó de manera exitosa. En la Tabla 34 se muestra el tramo recorrido por el vehículo, no se especificó las coordenadas especificas ya que por el movimiento del vehículo no se pudo recuperar las coordenadas exactas.

Tabla 34

Lugares recorridos

N° de Prueba	Sujeto de prueba	Ubicación	Recuperación de ubicación
1	M1	San Roque	SI
2	M1	Cumanda	SI
3	M1	Autopista General Rumiñaui (Peaje)	SI
4	M1	Puente 7	SI
5	M1	Av. Ciudad del Niño (Valle de los Chillos)	NO
6	M1	Av Abdón Calderón (Valle de los chillos)	SI

Se navegó un tramo mediante un vehículo y se prosiguió a ir recuperando la posición, para simular la recuperación de la mascota. En la Tabla 34, se muestran los

lugares por donde se recorrió, en un momento dado la ubicación no se recibió y no se actualizó la ubicación, pero en el lapso de unos minutos, la posición se comenzó a estabilizar demostrando que el sistema trabaja de una manera óptima en altas velocidades.

4.3.5. Escenario 5: Ambiente Variable.

Ubicación: Se realizó pruebas en escenarios simulados de pérdida del animal, por ejemplo una mascota va con una persona a lugares desconocidos, y mediante las terminales se intenta recuperar a la mascota, en la Tabla 35 se muestra un resumen de los resultados obtenidos en las simulaciones realizadas, y el éxito o no, de la recuperación de la mascota en distintos escenarios. Para tener una mayor facilidad de desplazamiento, el animal se trasladó acompañado de una persona por medio de un vehículo, y la recuperación se la obtuvo movilizándose por medio de un segundo vehículo. La recuperación se la realizó a partir de lugares cercanos y con una lejanía similar al ambiente simulado, por lo que el tiempo de recuperación no se lo realizó a partir de un mismo punto.

Tabla 35

Recuperación de ubicación en Escenario 5

Escenario	Nombre de mascota	Ubicación Exitosa	Tiempo de uso	Tiempo en ubicar
Villaflora	M2	SI	15 min	10 min
Miraflores	M2	SI	20 min	14 min
Universidad Central	M1	SI		8 min
Cinco de Junio (Casa)	M2	NO	20 min	-
Conocoto	M3	SI	15 min	5 min

Las pruebas realizadas en el Escenario 5, establecen que la precisión de la mascota depende de las condiciones en donde se encuentra ubicado el animal, por lo general en lugares cerrados, la ubicación demora un cierto tiempo en ser recuperada.

Además la exactitud de la ubicación recuperada no es muy precisa, por lo que se evidencia varios inconvenientes al utilizar el arnés en este tipo de escenarios.

Movimiento, al igual que en la ubicación, se planteó un ambiente variable y desconocido para la recuperación de información correspondiente al movimiento, es decir se puso el arnés en lugares desconocidos sin visibilidad del sueño de la mascota y se procedió con la recuperación de información mediante las terminales, esto muestra la recuperación en un ambiente más realista. La Tabla 36 muestra la información recuperada en un ambiente variable.

Tabla 36

Recuperación de movimiento en Escenario 5

N° de Prueba	Sujeto de Prueba	Movimiento	Tiempo de uso	Recuperación exitosa
1	M1	SI	10 min	SI
2	M2	SI	15 min	SI
3	M3	NO	12 min	SI
4	M2	NO	20 min	SI
5	M2	SI	15 min	SI

En la Tabla 36 se puede verificar que en lugares desconocidos y variables se recupera de manera correcta en todas las pruebas realizadas correspondientes al movimiento. El Escenario 5 simula una situación real, con lo cual se puede obtener una información más cercana a la realidad. De manera aleatoria se estableció en algunas situaciones el arnés en una mascota y en otras en un objeto estático, ya que no se puede inducir la falta de movimiento en una mascota. Como se muestra en la Tabla 36 la recuperación de la información fue correcta en todas las pruebas realizadas. A pesar de la recuperación correcta de la información de movimiento, el arnés no tiene la capacidad de informar al usuario el tipo de movimiento generado, ya sea si el animal se encuentra corriendo, caminando o sentado, pero para el objetivo del proyecto la información recuperada por el arnés correspondiente al movimiento es la ideal.

4.4. Consumo de batería.

Es importante conocer el tiempo de descarga del dispositivo, ya que mediante esta medida se puede conocer el tiempo que una persona tiene para la recuperación de la mascota,

Tiempo de carga: Cinco horas aproximadamente

Tiempo de descarga: Cuatro horas y media a cinco horas aproximadamente**.**

Por lo general las personas se dan cuenta de la pérdida de una mascota de manera inmediata, por lo que un tiempo prudente de funcionamiento del dispositivo se lo estimaría entre 2 a 3 horas, aunque pueden existir situaciones especiales en las que el dueño no se da cuenta del extravío del animal. Por el consumo de energía de los componentes, la batería establecida es la ideal para energizar al circuito, además de su portabilidad necesaria para la implementación en el arnés. La disposición de una batería es importante para la portabilidad del proyecto, los diferentes inconvenientes en la implementación de una batería en el arnés son:

- Alto consumo de corriente del módulo SIM 908.
- Voltaje de 5V para alimentación de la plataforma Arduino UNO.
- Accesibilidad a una batería para la alimentación simultanea de la tarjeta Arduino UNO y el módulo SIM 908.
- Accesibilidad a una batería cargable para evitar el cambio seguido de las baterías.
- El GPS del módulo SIM 908 deja de funcionar si la batería no suministra suficiente corriente.

4.5. Costos.

El proyecto establece un punto de investigación, para que un investigador pueda mejorarlo y abaratar los costos de implementación, así como mejorar su tamaño, la investigación se enfocó en la parte de software, por lo que se utilizó el hardware disponible en el mercado capaz de cumplir con los requerimientos que el sistema

debe cumplir. La Tabla 37 muestra un resumen del costo de las herramientas utilizadas para la realización del proyecto.

Tabla 37

Costos de implementación

Cantidad	Hardware	Costo Unitario	Costo Mensual	Costo Total
1	Tarjeta programable Arduino UNO original	$40		$40
1	Módulo SIM 908 para Arduino (Antena GPS y GSM/GPRS incluida.)	$95		$95
1	Acelerómetro MMA 7361	$6		$7
1	Sensor MLX906131 infrarrojo	$30		$30
1	Batería de Litio	-		-
2	Cargadores Universales	$5		$10
1	Caja de Plástico	$2		$2
	Tornillos y tuercas	-		-
	Cables	-		-
1	Chip de Telefonía celular (Datos Incluidos)	$5		$5
	Plan de datos		$17,10	17,10
	TOTAL		$17,10	$206.10

Se puede reducir los costos de implementación, diseñando la tarjeta programable Arduino UNO y el módulo SIM 908 para Arduino de forma cacera, ya que los circuitos y componentes en el diseño se encuentran publicados, ya que se trata de hardware y software libres, pero su diseño, equivale a la realización de una nueva investigación.

4.6. Análisis de resultados.

El proyecto marca un punto de investigación para la elaboración de sistemas de geolocalización para mascotas, las condiciones con las que se implementó el sistema es para una colaboración netamente investigativa, en la etapa de pruebas se recuperó

distintos datos, los cuales son el punto de referencia para establecer la precisión del proyecto elaborado. Además de la información recuperada por el arnés para mascotas, es importante analizar los agentes externos involucrados en las muestras analizadas, y verificar la injerencia de los agentes mencionados en la información recuperada, así como la alimentación asociada al sistema, ya que el sistema requiere de una portabilidad para el uso en animales.

La información de posicionamiento recuperada marca una precisión considerable en lugares abiertos, y con vista al cielo, pero se nota una marcada imprecisión en la información recuperada en ambientes cerrados, esto se debe a los obstáculos que dificultan la obtención de una ubicación acertada por parte de la antena GPS. A pesar de que la precisión en lugares cerrados no es la adecuada, el arnés se diseñó para su uso en lugares al aire libre, por lo general las mascotas se pierden en este tipo de escenarios, por lo que la localización y la información recuperada para este tipo de variable es acertada para el propósito para lo que el arnés fue diseñado. La información recuperada por el sistema en condiciones de velocidad también nos dan resultados favorables, a pesar que no se pudo determinar la precisión del arnés en este tipo de escenarios, visualmente se pudo corroborar la precisión del mismo.

En lugares abiertos y despejados, el tiempo de recuperación de información del arnés varía de 30 segundos a 1 minuto, y su precisión se estima de 1 m a 10 m, precisión adecuada para la recuperación de un animal. Mientras que en lugares cerrados, el tiempo de recuperación puede variar de 15 minutos a 30 minutos, en ocasiones el sistema no recupera la información satelital por más tiempo o nunca lo hace, la precisión recuperada varía de 50 m, 100 m a 300 m en algunas ocasiones, la distancia no es la ideal para la recuperación de un animal, y el motivo principal de esta imprecisión son los obstáculos que impiden que la antena pueda visualizar todos los satélites encargados de estimar mediante triangulación satelital y recuperar la información de manera rápida y precisa.

Mediante las pruebas realizadas en simulaciones de situaciones reales, se pudo establecer la eficacia del arnés, ya que este tipo de pruebas fue realizado sin que la persona sepa el lugar en donde se encuentra la mascota con el arnés ubicado. Y en la mayoría de ocasiones la recuperación fue exitosa (Ver Tabla 36). Las ocasiones en

las cuales no se pudo obtener la localización de la mascota fueron escenarios cerrados.

El movimiento fue establecido como medida de recuperación de información para conocer si la mascota se encuentra con vida en una determinada situación, mediante la información recuperada, el sistema establece si la mascota se encuentra con movimiento o no. No es una medida que provea de una forma fiable si el animal se encuentra vivo, pero puede establecer si el arnés dispuesto en la mascota se ha movido, lo que de una forma constante sirve para que el dueño de una mascota establezca si su animal se encuentra moviéndose, ya que el movimiento ocasionado por agentes externos no permanece de manera constante. La medición del movimiento por medio del sensor acelerómetro se realizó obteniendo varias medidas de variación, y calculando si la diferencia de vibración recuperado por cada medida establece que el animal se encuentra moviéndose.

Por medio de la etapa de pruebas, se analizó el movimiento en los distintos estados en que puede estar una mascota, por ejemplo, corriendo, caminando, sentado o sin moverse, mediante las medidas recuperadas, se puede establecer si el animal está moviéndose, lo que indicaría que se encuentra con vida. El sensor acelerómetro ubicado en el sistema de hardware detecta un mínimo movimiento en el arnés, y la información se la muestra por medio de los terminales (Página web o Aplicación en Android). El sensor acelerómetro detecta vibraciones mínimas, por lo que el movimiento de un perro lo va a detectar sin inconvenientes, se debe considerar que el sistema no detecta si el arnés se está moviendo por causa del movimiento del animal o por alguna otra causa externa, pero se considera que un animal se va a estar moviendo de forma constante mientras que un movimiento externo no. Una situación que puede ocasionar el problema mencionado, es si el arnés se encuentra en un vehículo en movimiento, el movimiento de un vehículo puede ser constante por un largo tiempo, por lo que se añadió otra medida la cual ayuda al usuario a determinar si el animal se encuentra con vida, la temperatura. No fue posible la recuperación de información cuando un animal está dormido, por la notable incomodidad que el arnés causa en los animales con lo cual no se permitió llegar a ese estado.

La temperatura se dispuso como una segunda medida para que el usuario pueda determinar si el arnés está ubicado en la mascota, y se encuentra con signos vitales,

una medición de temperatura médica solo es posible determinarla vía rectal, pero mediante un sensor infrarrojo el arnés recupera la temperatura corporal externa del animal, la cual debe ser considerablemente mayor a la temperatura ambiente. Mediante las terminales se informa al usuario la temperatura externa del animal. El sistema pone a consideración del usuario la temperatura corporal como medición de signos vitales, ya que la temperatura corporal de una mascota por lo general es considerablemente mayor a la de la temperatura ambiente. Se realizó una medición de la temperatura del animal y de la temperatura ambiente para determinar la temperatura en distintos escenarios.

Por medio de la información de movimiento y de temperatura, el usuario puede tener la certeza que su mascota se encuentra con signos vitales en ocasiones determinadas. La temperatura se tomó como una medida de signos vitales, como se ha venido mencionando, la única manera de conocer la temperatura médica de un animal es vía rectal, pero esta medida se dispuso para que el usuario pueda asumir o no si el animal está con signos vitales, el sensor es infrarrojo por lo que debe estar dispuesto hacia el animal para que pueda medir la temperatura del mismo.

CAPÍTULO V
5. CONCLUSIONES Y RECOMENDACIONES

Las conclusiones se establecieron de acuerdo a los objetivos propuestos, la manera de realización en el diseño e implementación de hardware y software y los resultados establecidos por medio de los escenarios de prueba.

5.1. Conclusiones

- Se debe considerar la situación legal actual en el Ecuador con respecto a las mascotas, para conocer cómo se debe incurrir en la solución tecnológica planteada.
- El acceso a las herramientas de desarrollo de software en el Sistema Operativo Android, son gratuitas y disponibles para el desarrollador, lo que facilitó y agilizó la realización del proyecto.
- El módulo SIM 908 GPS/GSM/GPRS consume 2000 mA en modo de operación, cuando la batería se encuentra agotada, las funciones realizadas por el módulo no se ejecutan de forma adecuada.
- Se debe disponer de un chip de una operadora telefónica con datos incluidos para el correcto funcionamiento del sistema.
- Los pines de transmisión y recepción de la plataforma Arduino UNO se conectan a los pines de recepción y transmisión del módulo SIM 908 respectivamente, la plataforma Arduino transmite los comando AT programados mediante el pin TX, el módulo recibe el comando mediante el pin RX, y envía la respuesta por medio del pin TX, para que la plataforma Arduino la reciba por medio del pin RX.
- Los pines 0 y 1 corresponden a los pines de recepción (RX) y transmisión (TX) de la plataforma Arduino UNO por defecto, la utilización de estos pines para transmitir y recibir la información de forma portátil se dificulta porque se encuentran conectados al puerto USB de forma interna y requieren de la conexión y actividad de este puerto para el envío y recepción de información. Para la comunicación de la tarjeta Arduino con el módulo SIM 908 de forma

portátil, fue necesaria integrar la librería “Software Serial”, para configurar los pines 3 y 4 en modo recepción y transmisión.

- La plataforma Arduino UNO requiere de al menos 5V para su funcionamiento normal, por lo que se integró un circuito con salida USB de 5 V para el propósito establecido, Se implementó un circuito de carga y alimentación USB de 5V recuperados por medio de cargadores universales.
- El consumo de corriente de la tarjeta programable Arduino UNO, los sensores acelerómetro y de temperatura infrarroja requieren de un suministro de corriente mínimo para un funcionamiento adecuado.
- Para la implementación del sensor acelerómetro MMA7361 y el sensor infrarrojo de temperatura MLX90614 se utilizó la implementación de las librerías “AcceleroMMA7361.h” y “i2cmaster.h” para la comunicación del sensor infrarrojo, librerías integradas para el software Arduino IDE.
- La conexión de la polarización negativa (tierra) se la debe implementar de forma común.
- Para un correcto registro del módulo en la red celular, el escenario debe tener señal y recepción de telefonía.
- El control del módulo SIM 908 para envío de información mediante GSM/GPRS y recepción de información GPS se lo realizó mediante comandos AT.
- Si se dispone información de autenticación incorrecta en el chip establecido en el módulo SIM 908 no es posible el registro y envío de información por medio de la red celular.
- La velocidad de comunicación serial establecida para el correcto funcionamiento del sistema es de 57600 bps.
- El GPS del módulo SIM 908 deja de funcionar si la batería no suministra suficiente corriente.
- El consumo de datos en un día es de 10 Mb aproximadamente, asumiendo el uso constante del arnés durante las 24 horas del día.
- El tiempo de descarga en modo de operación es de cuatro horas y media a cinco.

- La información GPS recuperada ingresa en formato NMEA, por lo que es necesaria su conversión al formato grados decimales, usado por los mapas de Google para ubicar un marcador en el mapa.
- Mediante el hosting gratuito “Freevar” se estableció el sistema cliente-servidor de forma correcta y sin compkicaciones.
- Para almacenar y consultar la información recuperada por el arnés para mascotas, esta es enviada por medio de método HTTP GET hacia el script de configuración PHP, el cual recibe la información por medio del mismo método y la almacena en tablas dispuestas en la base de datos.
- El hosting debe incluir un gestor de base de datos.
- Para consultar la información de posicionamiento geográfico y signos vitales por medio de una terminal, el usuario debe estar registrado y obtener un arnés con su identificador previamente.
- Para el inicio de sesión el usuario usa la información correspondiente al nombre de usuario y contraseña. Un usuario no puede usar un nombre o correo previamente registrado para su registro.
- El diseño de las terminales web y de la aplicación en Android fue establecido por medio de formatos comunes para una visión más atractiva hacia el usuario final.
- Se recupera la información de la base de datos en el terminal web por medio de un script de configuración PHP, que recupera y ordena la información mediante la generación de en un archivo XML y en el terminal por medio de un script de configuración PHP que recupera y ordena la información por medio de la generación de un archivo en formato JSON.
- Un usuario tiene acceso únicamente a la información de las mascotas que lo corresponden.
- Para separar la información correspondiente a cada usuario y mascota, se compara la información de nombre de usuario ingresada y autenticada con los archivos en formatos establecidos, JSON para Android y XML para terminal web, y solo se muestra la información del usuario que inició sesión.
- Se puede verificar la información de posicionamiento y signos vitales recuperada por el arnés, vía web o mediante la aplicación en Android.

- La aplicación en Android consta de hasta cinco niveles de acceso.
- De acuerdo al diseño de hardware, se establecieron los componentes en una caja de 13.5cm x 6cm x 9,5cm en largo, alto y ancho.
- La ubicación recuperada por el sistema se almacena en una tabla de la base de datos, en la cual las terminales disponibles para el usuario final consulta y despliega la información en marcadores establecidos en mapas.
- Se puede mostrar un máximo de 10 marcadores de ubicación en el mapa por mascota en situaciones normales.
- La precisión geográfica recuperada en ambientes cerrados, es inestable e imprecisa. La precisión no es la adecuada para la recuperación correcta de una mascota.
- La precisión GPS en lugares abiertos, con poca vegetación y obstáculos en los alrededores es de 1m a 10 m, puede existir variaciones de posicionamiento y precisión por la ubicación del animal con respecto a los satélites GPS y los obstáculos presentados en un punto establecido.
- El tiempo de recuperación de información GPS es variable, y depende del escenario, la situación energética del dispositivo y la ubicación en la que se encuentra el dispositivo a nivel geográfico.
- El tiempo de recuperación de información GPS en un parque con poca vegetación y obstáculos en los alrededores es de 30 segundos a 1 minuto.
- En un escenario cerrado, como el interior de una casa la información de posicionamiento recuperada es inestable e imprecisa, los obstáculos como paredes de hormigón habituales en hogares dificultan la recuperación de latitud y longitud, y en muchas ocasiones no es posible la recuperación de información de posicionamiento. Usualmente la precisión en este tipo de escenario supera los 100 metros.
- El tiempo de recuperación GPS en lugares cerrados varía de 15 minutos a 30 minutos, en ocasiones el sistema no puede localizar los satélites por más tiempo o nunca.
- El sistema recupera una posición acertada en altas velocidades.

- La temperatura ambiente influye en la información de temperatura externa de la mascota.
- La temperatura externa que recupera el arnés depende de diversos factores, como la raza del perro, situación climática o el estado de la mascota.
- La temperatura externa recuperada no equivale a una medida de temperatura médica, ya que para esto se debe disponer de sensores y termómetros especializados e invasivos con el animal.
- La temperatura obtenida en los diferentes escenarios depende de los factores climáticos externos, la raza o tipo de mascota en que está dispuesto el arnés, el lugar corporal de la mascota en donde se dispuso el arnés, en este caso el lomo de la mascota y el estado de salud y físico de la mascota.
- La temperatura recuperada en ambientes cerrados presenta menos variaciones debido a que las condiciones cerradas no permiten el paso de factores climáticos externos.
- Los valores de signos vitales (movimiento y temperatura) son recuperados para que el dueño de una mascota pueda discernir según la información recibida, si la mascota se encuentra con vida.
- Únicamente fue necesaria la recuperación de los valores correspondientes al eje Z del sensor acelerómetro para conocer la vibración y movimiento en la mascota.
- El sensor acelerómetro recupera números aleatorios positivos, mientras existe mayor movimiento la variación de los números es mayor, mientras hay menos movimiento la variación es mínima.
- Para obtener información acertada de movimiento en la mascota, el arnés debe estar encendido por 1 – 2 minutos aproximadamente.
- La validación de la información correspondiente al movimiento de la mascota recuperada por el acelerómetro, se la estableció restando el mayor valor recuperado con el menor en una muestra de diez elementos, por medio de pruebas se estableció que la resta es decir la variación de la información debe ser mayor al número 10 para asumir que existe movimiento.
- Cuando el arnés se encuentra en reposo el sensor acelerómetro recupera variaciones mínimas lo que equivale a una variación numérica pequeña por

lo que se estableció una variación menor a 10 en un rango de diez muestras para asumir reposo en el arnés.

- Los valores obtenidos por el sensor acelerómetro con el arnés dispuesto en una mascota caminando, tiene una amplia variación, con la toma de muestras establecida, la variación está en un rango mayor a 10.
- Los valores obtenidos por el sensor acelerómetro con el arnés dispuesto en una objeto en reposo, tiene una variación mínima, con la toma de muestras establecida la variación está en un rango menor a 10.
- La recuperación de signos vitales en ambientes cerrados no se dificulta siempre y cuando exista señal telefónica de la telefónica usada para el propósito.
- Con una posición de la mascota en que el movimiento del animal sea pequeño, por ejemplo si la mascota está sentada, se recupera vibraciones con una validación mayor a 10, acertada para el propósito.
- En situaciones variables y desconocidas, el usuario recuperó de forma acertada la información de posicionamiento y signos vitales recuperados por el arnés.

5.2. Recomendaciones

- Se recomienda el uso del arnés en mascotas de tamaño mediano y grande, por el tamaño de la caja diseñada.
- El uso de un arnés para sujetar la caja con los componentes de hardware diseñados en la macota.
- La temperatura recuperada, es una temperatura externa del animal, la cual no corresponde a una medida médica, por lo que si un dueño tiene dudas del estado de salud de su mascota debe acudir a un profesional.
- Usar un chip de telefonía celular con plan incluido de 300 - 500 Mb, para evitar una recarga constante de saldo en el chip y evitar agotar los datos de forma imprevista.

- El uso de la tarjeta programable Arduino UNO y el módulo SIM 908 ensamblado previamente da una cierta garantía de su correcto funcionamiento.
- Sujetar fijamente el arnés en la mascota para evitar daños de hardware.
- Diseñar un arnés de acuerdo al tipo y tamaño de mascota que se desea monitorear.
- Cargar el dispositivo en su totalidad para evitar falta de suministro de energía en el sistema.

5.3. BIBLIOGRAFIA

(s.f.). Obtenido de http://docente.ucol.mx/sadanary/public_html/bd/cs.htm

Android. (2014). *Android Open Source Project*. Obtenido de http://source.android.com/source/code-lines.html#terms-and-caveats

Android. (2014). *Android Open Source Project*. Obtenido de http://source.android.com/source/index.html

Arduino. (2016). *Arduino*. Recuperado el 11 de Marzo de 2016, de Arduino Software (IDE): https://www.arduino.cc/en/Guide/Environment

Arduino. (2016). *Arduino*. Obtenido de https://www.arduino.cc/en/Main/ArduinoBoardUno

Arduino. (2016). *Introduction Arduino*. Obtenido de https://www.arduino.cc/en/Guide/Introduction

Arduino. (s.f.). *Arduino.* Obtenido de https://www.arduino.cc/en/Products/Compare

Baez, M., Borrego, Á., Cordero, J., Cruz, L., González, M., Hernández, F., . . . Zapata, Á. (s.f.). *Introducción a Android.* Madrid: E.M.E.

CCM. (Marzo de 2016). *CCM*. Recuperado el Enero de 2016, de http://es.ccm.net/contents/148-entorno-cliente-servidor

Circelli, G. (12 de Febrero de 2015). *Panama Hitek.* Obtenido de http://panamahitek.com/acelerometros-de-3-ejes-lo-que-necesitas-saber/

Código Civil Ecuatoriano Art.2226. (2005). *Código Civil Libro IV*. Quito.

Deluxe Small Business Sales, Inc. (8 de Abril de 2015). *Heyrex*. Obtenido de http://www.heyrex.com/en/heyrex/

Efrati, A., & Lessin, J. (04 de Diciembre de 2013). *The Information*. Obtenido de https://www.theinformation.com/Google-s-Andy-Rubin-Pursues-Replicant-Robots

El Comercio. (30 de Octubre de 2014). 70 artículos componen el proyecto de ley a favor de los animales. *El Comercio*.

Felker, D., & Dobbs, J. (2011). *Android Application Development for Dummies.* Indianapolis, Indiana: Wiley Publishing, Inc.

García, M. (04 de Octubre de 2012). *Brandemia*. Obtenido de http://www.brandemia.org/la-historia-del-logo-de-android

Gonzalez Gutierrez, E. (2016). *Aprender a programar.com*. Obtenido de http://aprenderaprogramar.com/index.php?option=com_content&view=article&id=429:introduccion-a-html-internet-y-los-flujos-de-informacion-estructura-cliente-servidor-navegador-cu00703b&catid=69:tutorial-basico-programador-web-html-desde-cero&Itemid=192

Greer, R. (15 de Junio de 2007). *Affinity*. Recuperado el 15 de Enero de 2016, de http://www.affinity-petcare.com/veterinary/actualidad-veterinaria/abstracts/1920

Guarnizo, J. (s.f.). *Manual de Usuario Acelerómetro MMA7361*. Recuperado el 16 de Enero de 2016

Hellotechie. (23 de Octubre de 2014). *Sparkfun*. Recuperado el 2016, de https://learn.sparkfun.com/tutorials/sik-experiment-guide-for-arduino---v32

i4C Innovations, Inc. (19 de Mayo de 2015). *Voyce*. Obtenido de http://voyce.com/

Joan. (02 de Enero de 2014). *Microcontroladores y electrónica*. Obtenido de http://mikroe.es/arduino-uno/

Joan. (03 de 01 de 2014). *Microcontroladores y electrónica*. Recuperado el 15 de Febrero de 2016, de http://mikroe.es/arduino-uno-2/

Krall, C., & Sierra, M. (2016). *Aprender a programar.com.* Obtenido de http://aprenderaprogramar.com/index.php?option=com_content&view=article&id=706:ique-es-css-html-conocimiento-previo-para-poder-aprender-css-desde-cero-cu01003d&catid=75:tutorial-basico-programador-web-css-desde-cero&Itemid=203

Lettner, M. (7 de Diciembre de 2015). *Tractive*. Obtenido de https://tractive.com/es?contxp=false

Melexis. (s.f.). *Digital Plug & Play Infrared Thermometer in a TO-Can*. Recuperado el 13 de Julio de 2016, de MLX90614: https://www.melexis.com/en/product/mlx90614/digital-plug-play-infrared-thermometer-to-can

Microelectronic Integrated Systems . (2015). *MLX90614 Family Datasheet.* Junio.

Ministerio de Justicia, Derechos Humanos y Cultos. (2014). *Código Orgánico Integral Penal.* Quito: Graficas Ayerve C.A.

NexGen Dog. (24 de Agosto de 2014). *NextGen Dog.* Obtenido de http://nextgendog.com/quick-petpace-smart-collar-review/

NIPLE SOFTWARE. (22 de Febrero de 2016). *PLACA SIM 908*. Obtenido de Niple Blog: http://www.niplesoft.net/blog/2016/02/22/sim908/

Ordenanza Metrolpolitana N°128. (2004). Quito.

Otto Petcare Systems. (5 de Noviembre de 2015). *Otto Systems Petcare*. Obtenido de http://www.ottopetcaresystems.com/

PAE. (s.f.). *PAE Protección Animal del Ecuador*. Recuperado el 20 de Enero de 2016, de www.pae.ec/derecho-animal/legislación/vigente/

PE, I. (29 de Julio de 2014). *comohacer.eu*. Recuperado el 19 de Abril de 2016, de http://comohacer.eu/analisis-comparativo-placas-arduino-oficiales-compatibles/#Caracteristicas_generales

Petpace. (12 de Agosto de 2015). *Nasdaq.* Obtenido de https://globenewswire.com/news-release/2015/08/12/760207/10145839/en/PetPace-Collar-Continuously-Monitors-Pulse-Rate-in-Cardiac-Patient-at-Home-Provides-Real-Time-Reports.html

Philips, J. (27 de Mayo de 2015). *PC WORLD*. Recuperado el 20 de Noviembre de 2015, de http://www.pcworld.com/article/2923138/review-voyce-is-the-wellness-monitor-for-dogs-that-makes-wearables-relevant-to-humans.html

SIMCom. (2011). *SIM908 Hardware Design.* Shangai: SIM Technology Building.

Torres, C. (6 de Junio de 2014). *Androidsis.* Obtenido de http://www.androidsis.com/la-verdadera-historia-de-android-nacimiento-del-sistema-operativo-2003/

UCOL. (16 de 02 de 2016). *Definición del cliente servidor* . Obtenido de http://docente.ucol.mx/sadanary/public_html/bd/cs.htm#_Indice

Universidad de Colima. (s.f.). Obtenido de http://docente.ucol.mx/sadanary/public_html/bd/cs.htm#_Indice

Vergara Merino, R., Hernández Correas, A., Virúes Ortega, D., Ramos Campo, D., & García Cabañas Bueno, J. A. (2016). *Piloto de dron (RPAS).* Madrid, España: Ediciones Paraninfo.

Wilson. (16 de Octubre de 2015). *Canine Journal*. Obtenido de www.caninejournal.com/dog-activity-monitor/

Zapata, A. (2006). *Diseño del comportamiento: Diagrama de actividades.*