ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

CARRERA ELECTRÓNICA Y AUTOMATIZACIÓN

"PROTOTIPO IOT DE GEOLOCALIZACIÓN Y MONITOREO MEDIANTE TECNOLOGÍA LORA, PARA LA PREVENCIÓN DEL ABIGEATO EN EL CANTÓN EL TAMBO".

Trabajo de Titulación

Tipo: Dispositivo Tecnológico

Presentado para optar al grado académico de:

INGENIERO EN ELECTRÓNICA Y AUTOMATIZACIÓN

AUTOR: OSCAR BENJAMÍN ACERO GUAMÁN

DIRECTOR: Ing. EDWIN VINICIO ALTAMIRANO SANTILLÁN

Riobamba – Ecuador

2023

© 2023, Oscar Benjamín Acero Guamán

Se autoriza la reproducción total o parcial, con fines académicos, por cualquier medio o procedimiento, incluyendo cita bibliográfica del documento, siempre y cuando se reconozca el Derecho de Autor.

Yo, OSCAR BENJAMÍN ACERO GUAMÁN, declaro que el presente Trabajo de Titulación es de mi autoría y los resultados del mismo son auténticos. Los textos en el documento que provienen de otras fuentes están debidamente citados y referenciados.

Como autor asumo la responsabilidad legal y académica de los contenidos de este Trabajo de Titulación; el patrimonio intelectual pertenece a la Escuela Superior Politécnica de Chimborazo.

Riobamba, 21 de noviembre de 2023

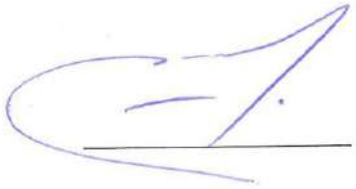
Oscar Benjamin Acero Guamán
030240595-6

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

CARRERA ELECTRÓNICA Y AUTOMATIZACIÓN

El Tribunal del Trabajo de Titulación certifica que: El Trabajo de Titulación; Tipo: Dispositivo Tecnológico**, "PROTOTIPO IOT DE GEOLOCALIZACIÓN Y MONITOREO MEDIANTE TECNOLOGÍA LORA, PARA LA PREVENCIÓN DEL ABIGEATO EN EL CANTÓN EL TAMBO"**, realizado por el señor: **OSCAR BENJAMÍN ACERO GUAMÁN**, ha sido minuciosamente revisado por los Miembros del Tribunal del Trabajo de Titulación, el mismo que cumple con los requisitos científicos, técnicos, legales, en tal virtud el Tribunal Autoriza su presentación.

	FIRMA	**FECHA**
Ing. Marco Antonio Viteri Barrera **PRESIDENTE DEL TRIBUNAL**		2023-11-21
Ing. Edwin Altamirano Santillán **DIRECTOR DEL TRABAJO DE TITULACIÓN**		2023-11-21
Ing. Verónica Elizabeth Mora Chunllo **ASESOR DEL TRABAJO DE TITULACIÓN**		2023-11-21

DEDICATORIA

Este trabajo de Integración Curricular se lo dedico a Dios, por darme conocimiento e inteligencia durante mi formación académica, a mis padres Manuel Acero e Isabel Guamán por todo su apoyo, por sus enseñanzas y valores inculcados en mí. A mi hermana, a mis tíos que siempre me han brindado su apoyo.

Oscar

AGRADECIMIENTO

Primero agradezco a Dios, por darme salud y vida. A mis padres por apoyarme en cada etapa de mi vida, por darme ánimos y fuerza para seguir luchando cada día a pesar de las adversidades. A la Escuela Superior Politécnica de Chimborazo por permitirme formar parte de su familia estudiantil, y forjar en mí conocimientos y valores para un ámbito profesional. De igual manera, un especial agradecimiento al Ing. Pablo Lozada, Ing. Edwin Altamirano y a la Ing. Verónica Mora por su guía durante el tiempo de desarrollo del trabajo de integración. A los docentes de la Facultad de Electrónica y Automatización por la formación académica brindada a lo largo de la carrera.

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE GRÁFICOS

ÍNDICE DE ECUACIONES

ÍNDICE DE ANEXOS

RESUMEN

En este trabajo de titulación se implementó un Prototipo IoT mediante Tecnología Lora, para la prevención del abigeato, el cual permite la geolocalización y monitoreo del ganado en tiempo real mediante una aplicación móvil. Con el desarrollo de este prototipo se buscó guardar la integridad del ganado en caso de hurto, uno de los delitos más comunes que se mantiene en auge hasta la presente fecha. Posee dos módulos desarrollados en la plataforma Arduino, la comunicación entre los módulos es mediante radio frecuencia utilizando el Módulo Transceptor LoRa SX1278 Ra-02 433 Mhz con una distancia máxima de 5 Km con línea de vista de transición y recepción. El primer módulo denominado nodo recolector, posee un Arduino nano encargado de gestionar a los componentes de recepción y trasmisión del prototipo. Para la obtención de los datos se utilizó un Módulo GPS LC86L, estos datos se envían de forma ordenada hacia el segundo nodo llamando Gateway, la información recibida se almacena en una base de datos en la nube para posteriormente ser visualizada mediante una apk que se encuentra desarrollada para ser utilizada en un dispositivo móvil por medio de un mensaje de texto cuando el ganado se encuentra fuera de la zona de monitoreo. De las pruebas realizadas se comprobó que prototipo posee un error de monitoreo mínimo que es despreciable en cada uno de los nodos que lo componen. Se concluye que el prototipo permite la geolocalización y monitoreo en tiempo real del ganado mediante tecnología Lora. Se recomienda disponer de un servicio de internet permanente y con buena cobertura para que el mensaje de alerta llegue oportuno al dueño del ganado.

Palabras clave: <INTERNET DE LAS COSAS (IOT)>, <COMUNICACIÓN INALÁMBRICA>, <SISTEMA DE POSICIONAMIENTO GLOBAL (GPS)>, <TECNOLOGÍA LORA>, <ABIGEATO>, < TARJETAS EMBEBIDAS>, <ENTORNO DE DESARROLLO INTEGRADO (IDE)>, <PYTHON (SOFTWARE)>.

1797-DBRA-UPT-2023

SUMMARY

The objective of this graduate research was to implement an loT Prototype using LoRa Technology to prevent cattle theft, which allows for the real-time geolocation and monitoring of livestock through a mobile application. The development of this prototype aimed to protect the integrity of the livestock in the event of theft, one of the most common crimes that continues to be prevalent to the present date. It has two modules developed on the Arduino platform, and communication between these modules is achieved through radio frequency using the LoRa Transceiver Module SX1278 Ra-02 433 MHz with a maximum range of 5 km under line-of-sight transmission and reception. The first module, called the collector node, has an Arduino nano responsible for managing the reception and transmission components of the prototype. To obtain the data, an LC86L GPS module was used; these data are sent in an organized manner to the second node called Gateway; the received information is stored in a cloud-based database and can be later visualized through an APK developed to be used on a mobile device Notifications are sent via text message when the livestock is detected outside the monitoring zone. From the conducted tests, it was confirmed that the prototype has a minimum monitoring error, which is negligible in each of its constituent nodes. It is concluded that the prototype allows real-time geolocation and monitoring of livestock using Lora technology. It is recommended to have a permanent internet service with reasonable coverage so that the alert message reaches the livestock owner in a timely manner.

Keywords: <INTERNET OF THINGS (IOT)>, <WIRELESS COMUNICATION, <GLOBAL POSITIONING SYSTEM (GPS)>, <LORA TEGHNOLOGY >, <CATTLE THEFT>, <EMBEDDED CARDS>, <INTEGRATED DEVELOPMET ENVIRONMENT (IDE)>, <PYTHON (SOFTWARE>.

1797-DBRA-UPT-2023

Dr. Lenin Iván Lara Olivo
0602546103

INTRODUCCIÓN

El abigeato en el presente año 2023 se mantiene en un punto de auge en las zonas rurales del Ecuador, dando preocupación a los pueblos indígenas debido a que representan su medio de sustento. Muchas autoridades indígenas representantes de las comunidades campesinas han gestionado un plan de trabajo para aplicar el sistema popular conocido como la justicia indígena, con el apoyo de varias instancias como la Gobernación, Policía Nacional, Judicatura, entre otros (EL HERALDO, 2023).

En la actualidad el hurto de ganado es un problema creciente no solo en Latinoamérica sino en todo el mundo, generalmente en países subdesarrollados debiéndose a muchos factores como el desempleo, la corrupción, la inseguridad. Los países africanos están gestionando un plan de trabajo para colocar dispositivos en el ganado, de esta forma tratar de disminuir estos actos ilícitos. Sin embargo, estos elementos no están a disposición de todos los ganaderos debido al precio de los mismos, por tal motivo los gobiernos aspiran cubrir la demanda reduciendo el precio en el mercado (Otieno, 2023).

En el presente trabajo de titulación se desarrolló e implementó un sistema IoT de geolocalización y monitoreo mediante tecnología Lora para la prevención del abigeato en el Cantón el Tambo provincia de Cañar.

Con este prototipo se busca monitorear por medio de geolocalización en tiempo real el ganado dentro de un perímetro de pastoreo específico. El mismo que está programado para enviar una alerta inmediata al propietario al momento de que el ganado monitoreado intente abandonar el área específica. Para nuestro caso se realizó un prototipo experimental, el cual tiene un dispositivo ubicado en el animal.

Los requerimientos hardware para diseño del sistema electrónico del prototipo de geolocalización y monitoreo consisten en dos nodos, el uno llamado recolector que comprende de una tarjeta embebida, un Módulo Lora y Módulo GPS, que actúa como emisor tomando la ubicación del ganado mediante coordenadas geográficas de latitud y longitud. Estos datos son enviados por radiofrecuencia hacia el otro nodo llamado Gateway que utiliza una tarjeta embebida y un Módulo Lora, actuando como receptor.

Los elementos para la implementación del sistema IoT de geolocalización y monitoreo comprende de un sistema de red inalámbrica – WIFI, el cual permite enviar datos a la nube y almacenarlo

tanto del nodo recolector como del Gateway. El prototipo cuenta con una aplicación móvil, con la cual el propietario puede visualizar y monitorear en tiempo real de su ganado.

CAPÍTULO I

En este aparatado se presenta el planteamiento del problema y los antecedentes que conllevan a la realización de este trabajo de titulación referente a la geolocalización mediante el monitoreo utilizando tecnología Lora.

1 DIAGNOSTICO DEL PROBLEMA

1.1 PLANTEAMIENTO DEL PROBLEMA

1.1.1 Antecedentes

En el pasado los moradores que tenían como fuente principal de ingreso a la agricultura y ganadería, se consideró que el atraco de los animales que se usaban para estas actividades productivas, se merecía una considerable sanción. Como consecuencia, la apropiación de bovino exento de la autorización de su dueño se denominó ABIGEATO, siendo los romanos los primeros en tildarlos con este nombre.

.

De acuerdo con el periódico del campo El productor, las dos primeras semanas de 2022, el robo de ganado tiene en angustia a ganaderos de la frontera norte del Ecuador. Tanto en Carchi (Ecuador) como en Nariño (Colombia) cada vez se reportan más casos. Algunas reses fueron localizadas en camales clandestinos. En esta zona fronteriza el hurto de este tipo se ha mantenido en auge de acuerdo con organismo policiales de ambos países. Incluso, se habla de la existencia de bandas organizadas dedicadas al abigeato, con integrantes tanto colombianos como ecuatorianos (EL PRODUCTOR, 2022).

Durante la investigación se coincidió con un proyecto similar, el cual se realizó en febrero del año 2020, dando como resultado un prototipo GPS con sistema de geolocalización de ganado bovino para las llanuras de Cotopaxi, el mismo que fue elaborado por estudiantes de la Universidad Técnica de Cotopaxi (Molina Molina, 2020).

En vista de no existir un sistema orientado al monitoreo de ganado vacuno por medio de geolocalización en tiempo real en el Cantón El Tambo provincia del Cañar y en vista que en los últimos meses se ha incrementado el hurto de ganado, se ha planteado la realización de este trabajo de titulación.

1.2 Justificación

1.2.1 Justificación teórica

En la figura 1-1, se muestra el esquema de Abigeato registrado en Ecuador en los últimos 3 años.

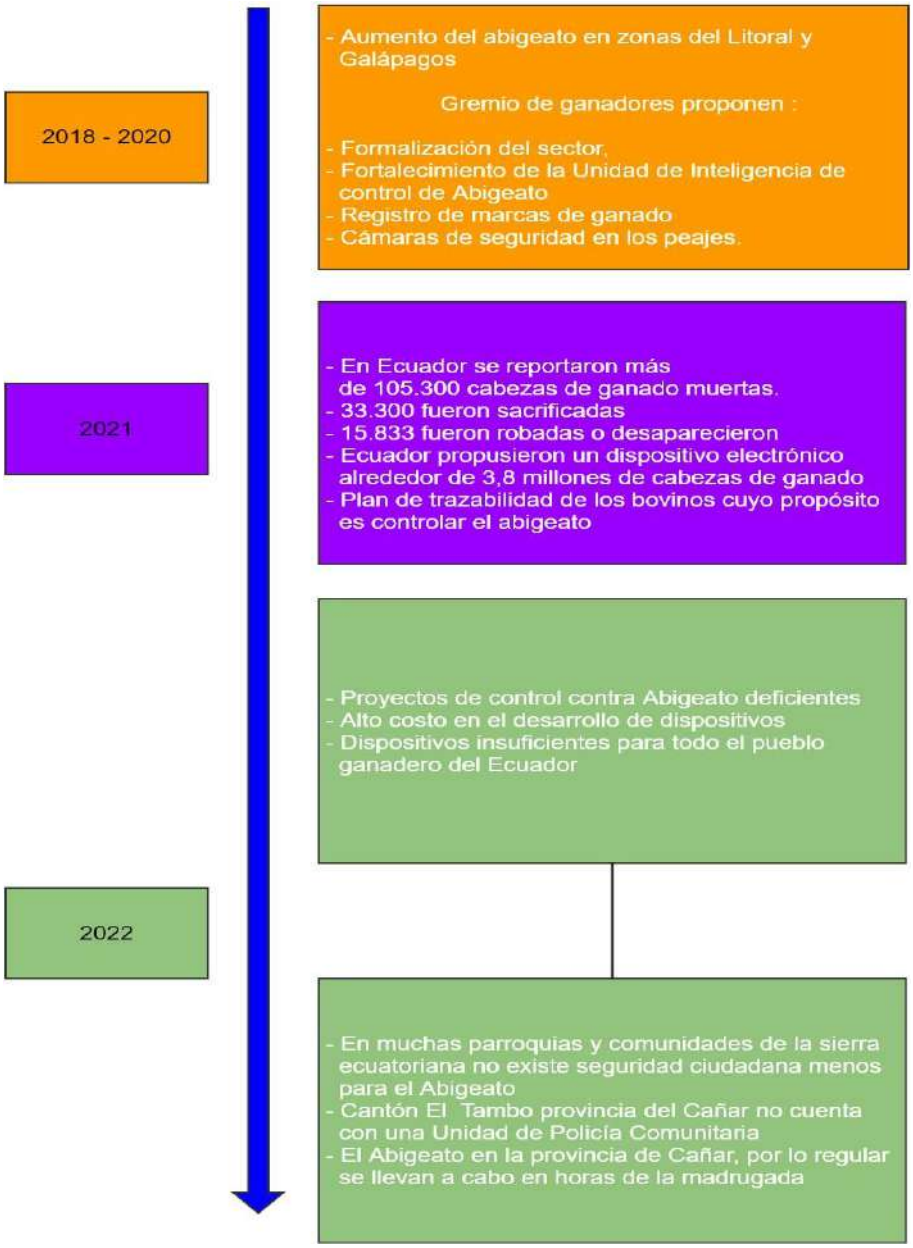


Figura 1-1. Esquema periodo con mayor índice de Abigeato en Ecuador.

Fuente: El Comercio, 2018; Yalilé , 2021; Martinez, 2022; NOTICIASDELCANAR, 2022.

Realizado por: Acero O, 2023.

Para el botín y extracción de ganado en Ecuador los delincuentes utilizan diversas formas para ejecutar la acción fraudulenta, estas se presentan en la figura 2-1.

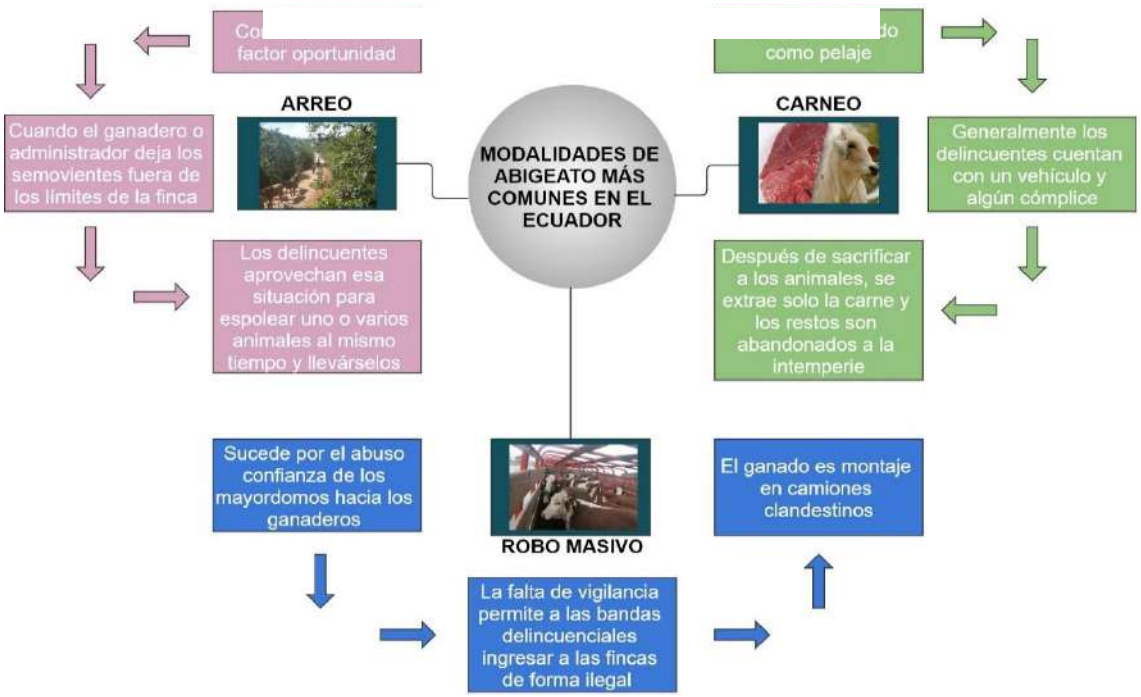


Figura 2-1. Modalidades de Abigeato más comunes en el Ecuador.

Fuente: Contexto Ganadero, 2016.

Realizado por: Acero O, 2023.

Este trabajo de investigación se realizará con la finalidad de monitorear por medio de geolocalización en tiempo real el ganado dentro de un perímetro de pastoreo específico. El mismo que estará programado para enviar una alerta inmediata al momento de que el individuo monitoreado intente abandonar el área específica. Para nuestro caso se realizará un prototipo experimental, el cual tendrá un dispositivo ubicado en el animal a prueba.

1.2.2 Justificación aplicativa

En la figura 3-1, se indica el diagrama previo estudio al desarrollo del prototipo IoT. El cual consistirá en tres partes. En la primera parte se hará un estudio de las tecnologías IoT y Lora. En la segunda parte se asignarán los requerimientos hardware y software para la implementación del prototipo IoT. Finalmente se diseñará un sistema electrónico del dispositivo IoT que permita la geolocalización y monitoreo.

En la figura 4-1, se observa el diagrama de desarrollo para el sistema de monitoreo del con geolocalización.

El sistema empezará con la implementación de dos dispositivos llamados nodo recolector el cual tendrá una tarjeta embebida, un Módulo Lora y Módulo GPS, en cambio para el nodo Gateway

se utilizará una tarjeta embebida y un Módulo Lora. Mediante la programación se realizará la lectura y envió de los datos recolectados y finalmente se desarrollará una aplicación móvil.

En la figura 5-1, se muestra cómo se ejecutará el diagrama de ejecución en tiempo real, que consiste en la adquisición de datos mediante el dispositivo nodo recolector, el cual se coloca en el ganado a prueba y hará a la vez de emisor tomando la ubicación de este mediante coordenadas geográficas latitud y longitud para luego ser enviadas al dispositivo nodo Gateway mediante tecnología Lora.

Figura 3-1. Diagrama en base a un estudio previo al desarrollo del prototipo IoT.

Realizado por: Acero O, 2023.

Figura 4-1. Diagrama de desarrollo del prototipo IoT.

Realizado por: Acero O. 2023.

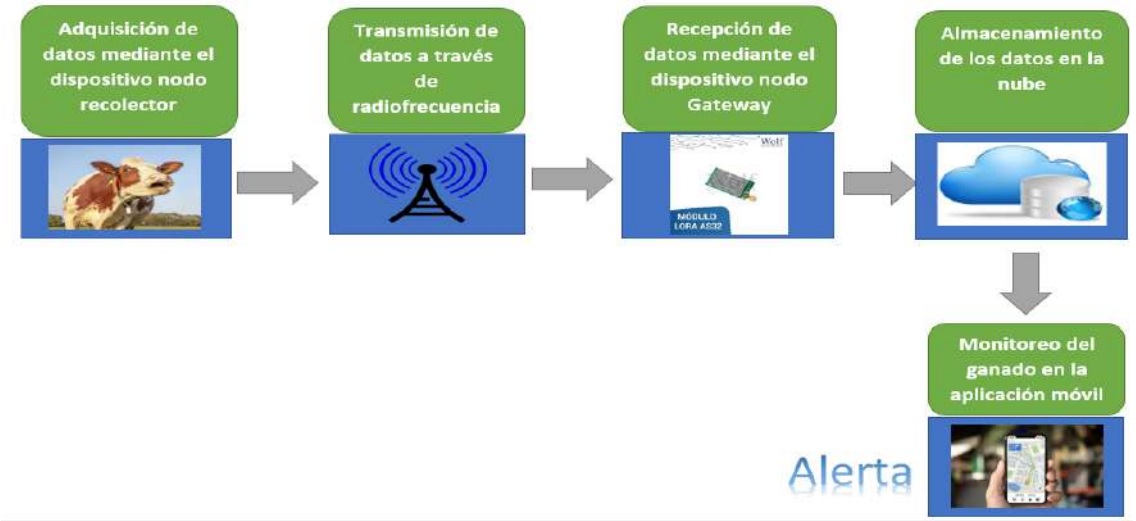


Figura 5-1. Diagrama de ejecución en tiempo real

Realizado por: Acero O. 2023

1.3 Objetivos

1.3.1 Objetivo general

Implementar un prototipo IoT de geolocalización y monitoreo mediante tecnología lora, para la prevención del abigeato en el Cantón El Tambo.

1.3.2 Objetivos específicos.

- Estudiar las tecnologías IoT y Lora para desarrollar una alternativa de monitoreo para prevención del abigeato.
- Determinar los requerimientos de hardware y software necesarios para la implementación del prototipo IoT de geolocalización y monitoreo.
- Diseñar e implementar el sistema electrónico del dispositivo IoT que permita la geolocalización y monitoreo.
- Investigar alternativas para el desarrollo de aplicaciones apk.
- Desarrollar una aplicación móvil que permita visualizar la ubicación del ganado.
- Evaluar el prototipo de geolocalización y monitoreo para precautelar la integridad del ganado en las zonas rurales más vulnerables como es en el Cantón El Tambo.

CAPÍTULO II

En este apartado se presenta la información concerniente de todos los componentes eléctricos y electrónicos que se van a tomar como referencia para la implementación del prototipo.

2 MARCO TEÓRICO

El estudio de los prototipos ¨IoT¨ en aplicaciones de seguridad como es el abigeato, ha buscado comprenderse desde distintas teorías. Ya sea en función proyectos, prototipos experimentales, entre otros. No obstante, es importante definir algunos conceptos claves en el tema de estudio. Entre los cuales se encuentran: IoT, prototipos, sistema de geolocalización y monitoreo, comunicación Inalámbrica, tecnología Lora y tarjetas embebidas. Se determina conceptos de abigeato y prevención.

2.1 Abigeato

El abigeato es una falta que interrumpe con la integridad de un ganado ya sea de mayor o menor incautación sin la autorización del ganadero. Generalmente los implicados son ladrones clandestinos o bandas criminales organizadas que se lucran por este medio, sin medir las consecuencias que conlleva este acto ilícito desde pérdidas económicas hasta la salud emocional a los propietarios.

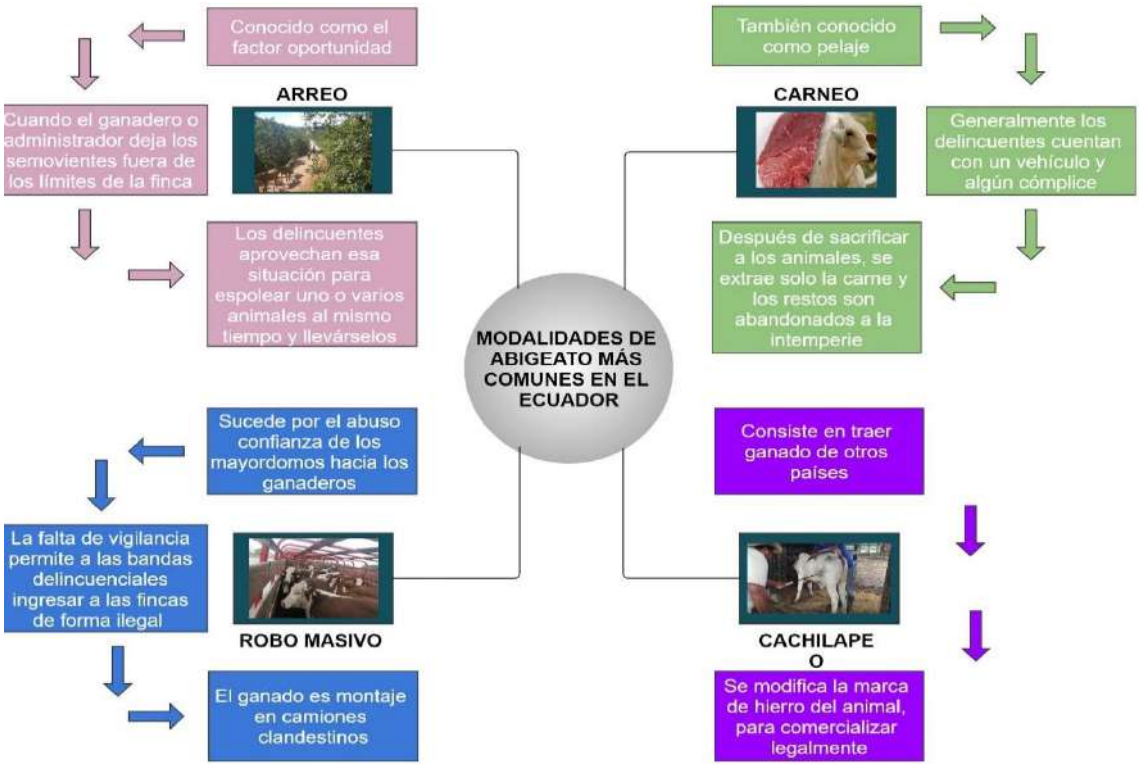
2.1.1 Antecedentes y origen del abigeato

En el pasado los moradores que tenían dependencia absoluta de la ganadería, se consideró que el atraco de estos mamíferos que se usaban para estas actividades productivas, se merecía una considerable sanción. Como consecuencia, la apropiación de bovino exento de la aprobación de su dueño se denominó ABIGEATO, siendo los romanos los primeros en tildarlos con este nombre.

En Latinoamérica, el término Abigeato se fundamenta a raíz del Código Rural en Buenos Aires, Argentina en 1865, sin embargo, las normativas para establecer dicho código empezaron en 1810 (Guerra Iriarte et al, 2014 pág. 22).

Para identificar las estrategias de hurto de vacunos que han desarrollado los maleantes a lo largo del tiempo, en la figura 1-2, se presenta las distintas modalidades de Abigeato usadas en Ecuador.

Figura 1-2. Modalidades de Abigeato más usadas en Ecuador



Fuente: Martinez, 2021.

Realizado por: Acero O. 2023.

2.2 Internet de las cosas ó IoT

Hoy en día IoT está influyendo en nuestro diario vivir, debido a la forma que actuamos y nos comportamos. Siendo así una red gigante con dispositivos conectados que se juntan para compartir datos de su funcionalidad y en el entorno que operan. Proporciona una plataforma y lenguaje común donde los dispositivos pueden vaciar sus datos y comunicarse entre sí. En la tabla 1-2, se muestra la evolución histórica IoT.

Tabla 1-2: Evolución histórica IoT.

Fecha	Hitos
1832	Barón Schilling de Rusia creó un telégrafo electromagnético.
1844	Primer código Morse por Samuel Morse.
1926	Nikola Tesla revista Colliers: Con una conexión inalámbrica perfecta la tierra será como un cerebro.
1964	Marshall McLuhan: Sistema de información a partir de una dinámica a través de medios eléctricos.
1969	Desarrollo de ARPANET.
1974	Inicio de TCP/IP
1989	Tim Berners-Lee: Propuso WWW (World Wide Web)
1990	John Romkey 1: Creo primer Tostadora.
1991	Primera Página Web.

1995	Primer servicio de Comercio Electrónico tales como Amazon, eBay.
1998	Se oficializó Google.
1998	Mark Weiser: La computación omnipresente es lo opuesto a la realidad virtual.
1999	Kevin Ashton: Introdujo el concepto “Internet de las Cosas” ó IoT.

Fuente: Bathla et al, 2019.

Realizado por: Acero O, 2023.

2.2.1 Aplicaciones IoT en el mundo real

Tienen el propósito de establecer un mundo inteligente y conectado con todos los lugares accesibles a la red, mediante aparatos de detección que brindan información del entorno; con máquinas y dispositivos más inteligentes que se ajustan a las necesidades de los humanos. Entre las principales aplicaciones que se desarrollan con IoT tenemos:

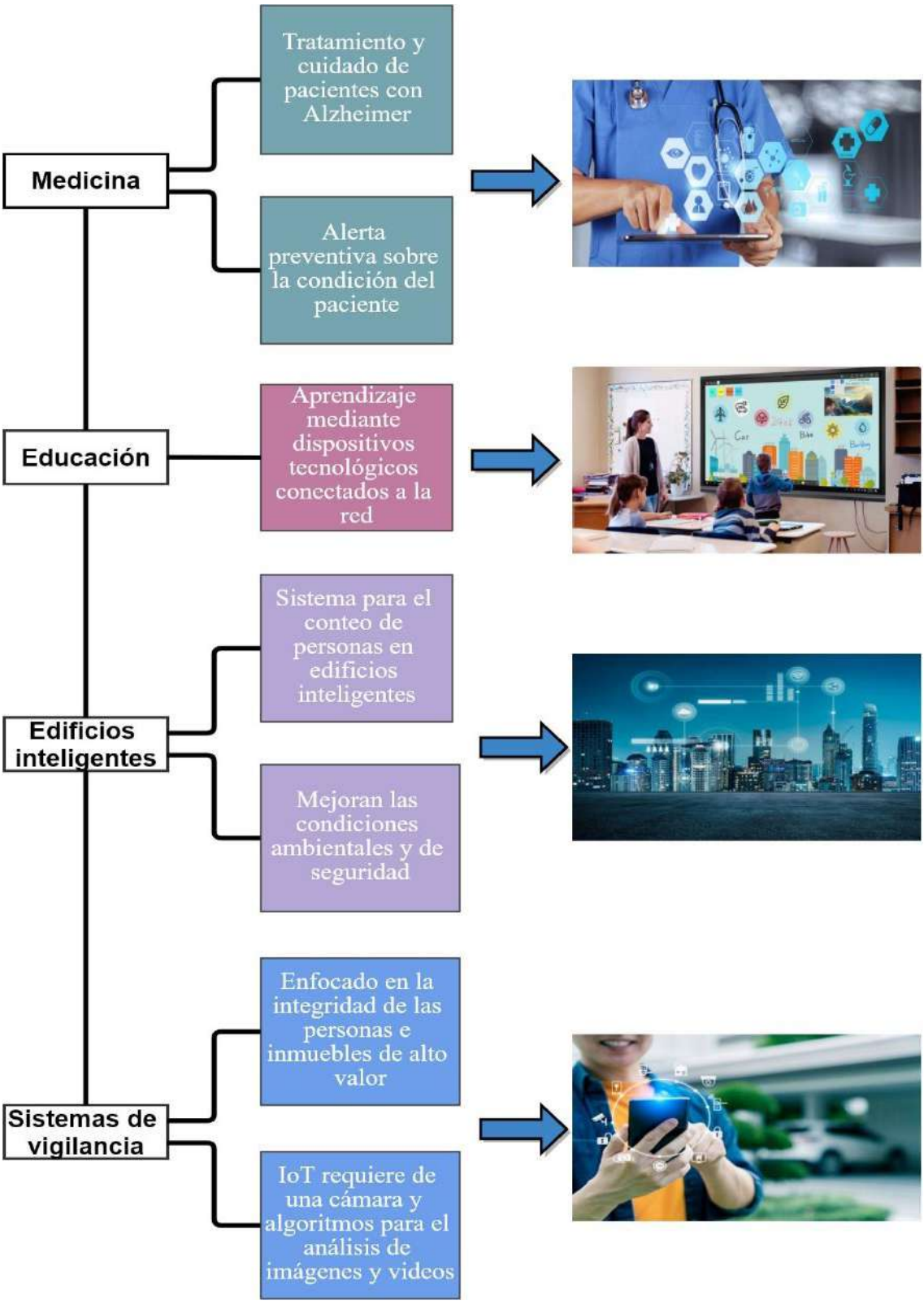


Figura 2-2. Esquema de aplicaciones con IoT.

Fuente: Berte, 2018.

Realizado por: Acero O, 2023.

2.2.2 Sensores y actuadores

Todas las aplicaciones IoT necesitan sensores y actuadores para realizar una tarea especifica, los cuales permiten soluciones en todos los campos verticales de IoT, desde ciudades inteligentes hasta agricultura inteligente, y desde salud personal hasta transporte inteligente.
(Bkheet & Agbinya, 2021).

2.2.2.1 Sensores

Son dispositivos capaces de distinguir señales externas y convertirlas en impulsos eléctricos para ser interpretados por una tarjeta embebida. Los encontramos en todas partes en la vida cotidiana, en el hogar y en el trabajo. Hoy en día tenemos sensores que asimilan las sensaciones de los humanos como sentir, oír, oler e incluso saborear.

2.2.2.2 Actuadores

Son instrumentos que transforman las señales de un sensor en diferentes acciones, es decir son capaces de ejecutar una acción sobre el entorno, son muy útiles para visualizar en el mundo físico algo que queremos. Los actuadores se pueden conectar a un microcontrolador para construir dispositivos.

2.2.3 Comunicación

La comunicación en IOT es preciso para la vinculación entre dispositivos, pero es restringida por las características del objeto, como la duración de la batería o el alcance durante la transmisión de datos. Las tecnologías más comunes utilizadas para IOT, son: Wi-Fi, Bluetooth, RFID, NFC, IEEE 802.15.4, Z-wave, zigbee y Tecnología 4G (LTE) (Bkheet & Agbinya, 2021).

En el gráfico 1-2, se muestra el alcance de los diferentes protocolos que se pueden implementar en IoT, desde los de menor alcance hasta los que cubren varios kilómetros.

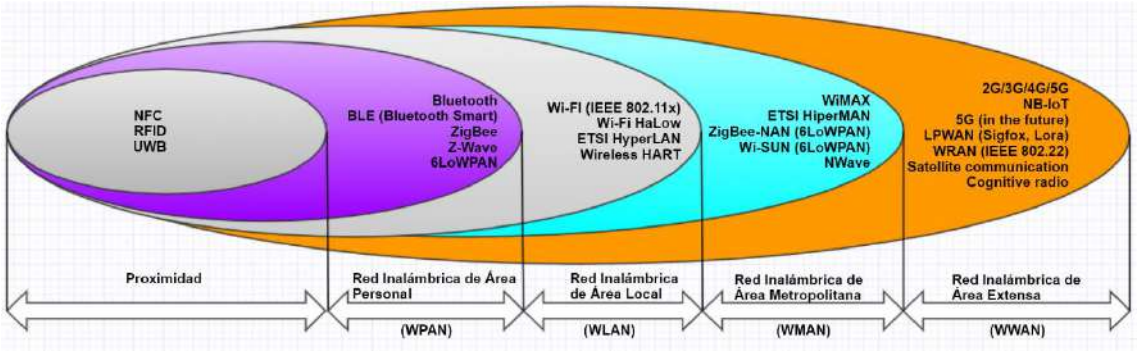


Gráfico 1-2. Alcance de los protocolos que se pueden implementar en IoT.

Fuente: Čolaković & Hadžialić, 2018.

Realizado por: Acero O, 2023.

2.2.4 Modelos de Comunicación en IoT

Los modelos de comunicación varían en cuanto al campo de aplicación que se desee implementar. En la figura 3-2, se observa un esquema con los modelos de comunicación más comunes usados en IoT.

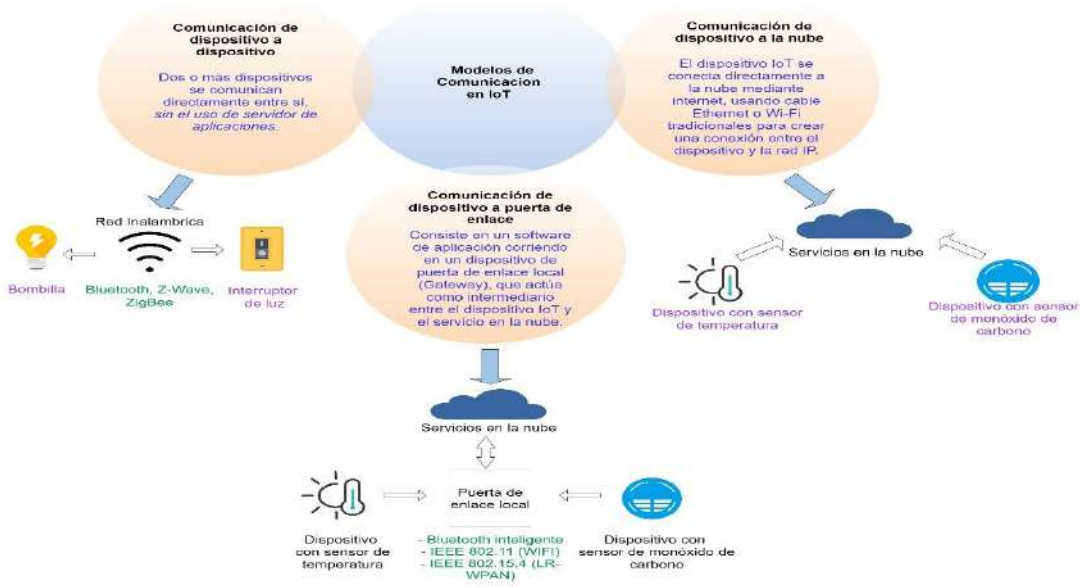


Figura 3-2. Modelos de Comunicación IoT.

Fuente: A Review of Identity Methods of Internet of Things (IOT), 2021.

Realizado por: Acero O, 2023.

El modelo de comunicación más destacado en el desarrollo de prototipos y proyectos IoT es la comunicación entre un dispositivo y el servicio en la nube mediante un intermediario o puerta de enlace, ya que ésta permite enlazar nodos de entrada y salida a distancias considerables.

2.3 Sistema de Monitoreo

Es un grupo de componentes que supervisan un valor en tiempo real, el cuál será transmitido al usuario final para constatar el funcionamiento de una variable, esta puede ser la temperatura ambiente en una edificación, siendo una variable no controlable, pero sirve como referencia para informar la condición actual del mismo.

Hoy en día la seguridad es un tema primordial en cuanto a la protección ya sea personal o de bienes materiales, a lo largo del tiempo diversas compañías han ido desarrollando tecnologías con el fin dar solución a este problema. Gracias al desarrollo del internet, todos los dispositivos y aparatos pueden conectarse entre sí e interactuar con el usuario. Cada vez es más latente la presencia del Iot en el desarrollo de la ciencia. Uno de los campos donde ha tenido gran influencia es en el ámbito de la seguridad basado en sistemas de monitoreo para industrias, agricultura, medicina, entre otros.

2.4 Sistema de posicionamiento global (GPS)

Consiste en un método de navegación basado en señales satelitales para mostrar la ubicación de un objeto. Para el funcionamiento del GPS se requiere 20 satélites, sin embargo, hay alrededor de 30 satélites dando vueltas por el globo. Un receptor GPS necesita señales de tres satélites para posicionar la latitud y la longitud.

Al grupo de satélites que giran alrededor del nuestro planeta se lo denomina Constelación de Satélites, así como se observa en la figura 4-2.

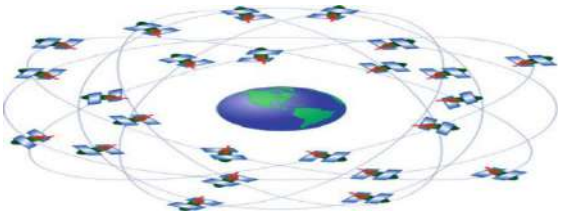


Figura 4-2. Constelación de satélites GPS.

Fuente: Tamazin, 2015.

2.4.1 Evolución de la tecnología del sistema de posicionamiento global (GPS)

A continuación, se recopila una breve reseña acerca de la evolución histórica de la tecnología del sistema de posicionamiento global (GPS), en la tabla 2-2, se muestra la evolución histórica del GPS.

Tabla 2-2: Evolución de la tecnología del sistema de posicionamiento global (GPS).

Fechas	Hitos
1950	Se descubrió un método para rastrear cuerpos fijos en la superficie de la Tierra, gracias a la disposición de los satélites.
1973	Programa de constelación de satélites NAVSTAR GPS.
1978	Se lanzo sus primeros cuatro satélites a orbita.
1995	Sistema de Posicionamiento Global (GPS).

Fuente: Kumar & Moore, 2002.

Realizado por: Acero O, 2023.

2.5 Redes Inalámbricas (Wireless Networks)

Son redes sin cable que tienen la capacidad de comunicarse por medios no guiados a través de ondas electromagnéticas. La trasmisión y recepción se realiza a través de antenas. Se puede trabajar con antenas intermedias para distancias de pocos metros o antenas repetidoras para alcanzar decenas de kilómetros. En la figura 5-2, se observa el esquema de una red inalámbrica.

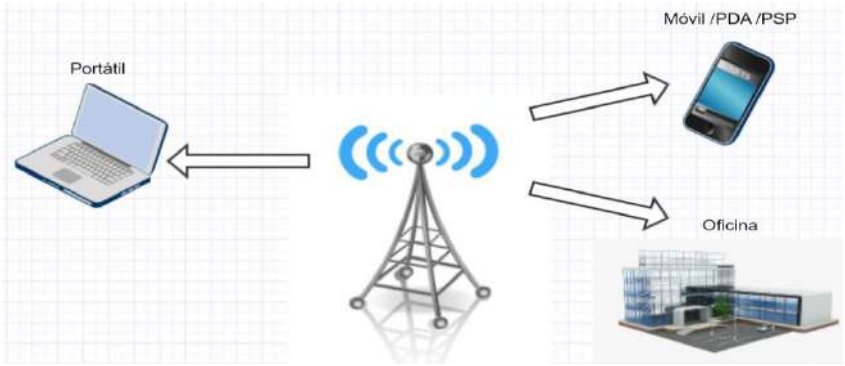


Figura 5-2: Red Inalámbrica.

Realizado por: Acero O, 2023.

Las redes inalámbricas tienen ciertas ventajas y desventajas, como se aprecia en la tabla 3-2.

Tabla 3-2: Ventajas y desventajas de las redes inalámbricas.

Ventajas	Desventajas
Rápida instalación de la red (no hay necesidad de cableado)	Sensible a cambios atmosféricos (lluvia, vientos, entre otros)
Movilidad (el medio de transmisión no está sujeto a ningún cable)	Seguridad vulnerable (requiere aumentar la seguridad y la encriptación)
Menos costes de mantenimiento (no existen cables)	Velocidad de navegación limitada
Compatibilidad con todos los dispositivos móviles.	Interferencias externas.
Productividad en entornos virtuales	Menor ancho de banda
Mayores alcances para zonas remotas	Altos costos iniciales: Dispositivos, antenas, entre otros

Fuente: Gómez, 2010.

Realizado por: Acero O, 2023.

2.6 Tecnología Lora

Hoy en día Lora juega un papel muy importante en la comunidad de IoT porque esta es una tecnología emergente, útil para transferir datos de un punto a otro, y estos puntos pueden estar separados por varios de kilómetros, sin la necesidad de redes móviles ni tecnología Wifi.

El protocolo Lora es propiedad de la empresa Semtech, con abreviatura compuesta en dos palabras Long (LO) y Range (RA) que significa de largo alance y un ancho de banda pequeño. Este sistema es inversamente proporcional, es decir a más ancho de banda menos será el rango.

Existen diferentes chips LoRa de la serie SX127x en el mercado y en tiendas virtuales, los más notables se muestran en la tabla 4-2. Los mismos que tienen gran demanda debido a la optimización que ofrecen al combinarse con tecnologías inalámbricas, además de su tamaño menor, que facilita el desarrollo de prototipos a escalas reducidas. Sin embargo, el factor de dispersión (SF) toma un papel primordial para la distinción de los módulos, donde el rango será más alto si el SF es mayor.

Tabla 4-2: Diferencias entre los chips LoRa Serie SX127x.

Transceptor	Rango de frecuencia	Factor de Dispersión	Ancho de banda	Tasa de bits	Sensibilidad
SX1276	137 a 1020 MHz	6 a 12	7.8 a 500kHz	0.018 a 37.5 kbps	-111 a -148 dBm
SX1277	137 a 1020 MHz	6 a 9	7.8 a 500kHz	0.11 a 37.5 kbps	-111 a -139 dBm
SX1278	137 a 525 MHz	6 a 12	7.8 a 500kHz	0.018 a 37.5 kbps	-111 a -148 dBm
SX1279	137 a 960 MHz	6 a 12	7.8 a 500kHz	0.018 a 37.5 kbps	-111 a -148 dBm

Fuente: Malik, 2020.

Realizado por: Acero O, 2023.

2.7 Lenguajes de Programación

Es una forma u orden que le da un humano a una computadora acorde a una tarea especifica. Está conformado grupo de símbolos que representan una instrucción precisa que combinados crean un programa para diferentes entornos virtuales de trabajo como páginas web, aplicaciones y distintos tipos de software. Ciertos lenguajes facilitan la vida del usuario.

2.7.1 Lenguajes de programación para el IOT

Los lenguajes de programación se han desarrollado y tenido grandes cambios y mejoras para distintas áreas de estudio e investigación. Sin embargo, ciertos lenguajes acontecen crucialmente en el desarrollo de IoT como se muestra en la figura 6-2.

Figura 6-2: Lenguajes de programación más comunes en IoT.

Fuente: Alvarado Quiñonez, 2017.

Realizado por: Acero O, 2023.

2.8 Protocolos dominantes para el envío de mensajes

Cabe mencionar que la actualidad la interacción entre personas o dispositivos se realiza mediante entornos virtuales basados en protocolos de comunicación, en la tabla 5-2, se muestra los protocolos dominantes en IoT.

Tabla 5-2: Protocolos en IoT.

HTTP	MQTT	CoAP
Fácil implementación con respecto a sistemas antiguos en IoT	Se diseñó para Internet de las cosas	Destinado a la comunicación entre dispositivos de bajo consumo
Creado para que los documentos estuvieran disponibles en Internet.	Conexiones TCP	Protocolos enfocados a las operaciones
Conexiones TCP	Arquitectura cliente-servidor y viceversa	Baja sobrecarga
Arquitectura cliente-servidor	Es liviano	Soporte de UDP

Fuente: Alvarado Quiñonez, 2017.

Realizado por: Acero O, 2023.

2.9 Aplicación Móvil

Es un programa destinado para unidades móviles o portátiles, disponibles para distintos sistemas operativos con acceso libre o pagado. Con una interfaz interactiva y de fácil manejo para usuarios con o sin habilidad tecnológica. En la figura 7-2, se muestra un celular con aplicación móviles activas.

Figura 7-2. Aplicación móvil.

Realizado por: Acero O, 2023.

El monitoreo del ganado es un aspecto vital de la gestión del ganado, ya que permite a los agricultores realizar un seguimiento de la salud, el comportamiento y la ubicación de sus animales. Muchas empresas se dedican al desarrollo de aplicaciones móviles para estas tareas, tal es el caso de la empresa Modisar que se encarga de desarrollar software para la Administración de ganado de precisión con dispositivos y módulos que se ubican en el animal, dando solución al seguimiento de ganado IoT (MODISAR, 2014). En la figura 8-2, se muestra una aplicación móvil para el seguimiento de ganado.

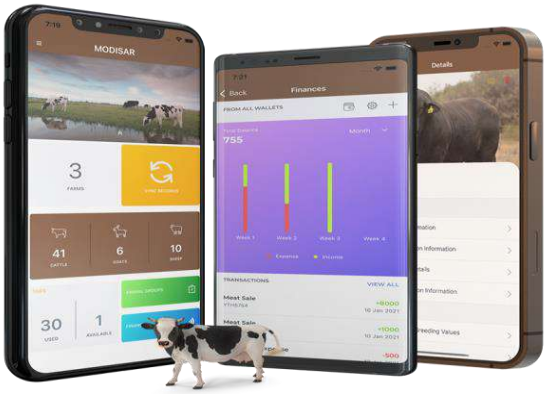


Figura 8-2. Aplicación móvil Modisar.

Fuente: MODISAR, 2014.

Este 2023, una de las plataformas más conocidas para la administración de granjas es Farmbrite, que está disponible para Android iOS y Android y fue desarrollada por la empresa que lleve el mismo nombre. Este software ayuda a las empresas a organizar, optimizar y administrar su negocio agrícola (Jindal, 2023). En la figura 9-2, se muestra la interfaz de la plataforma Farmbrite

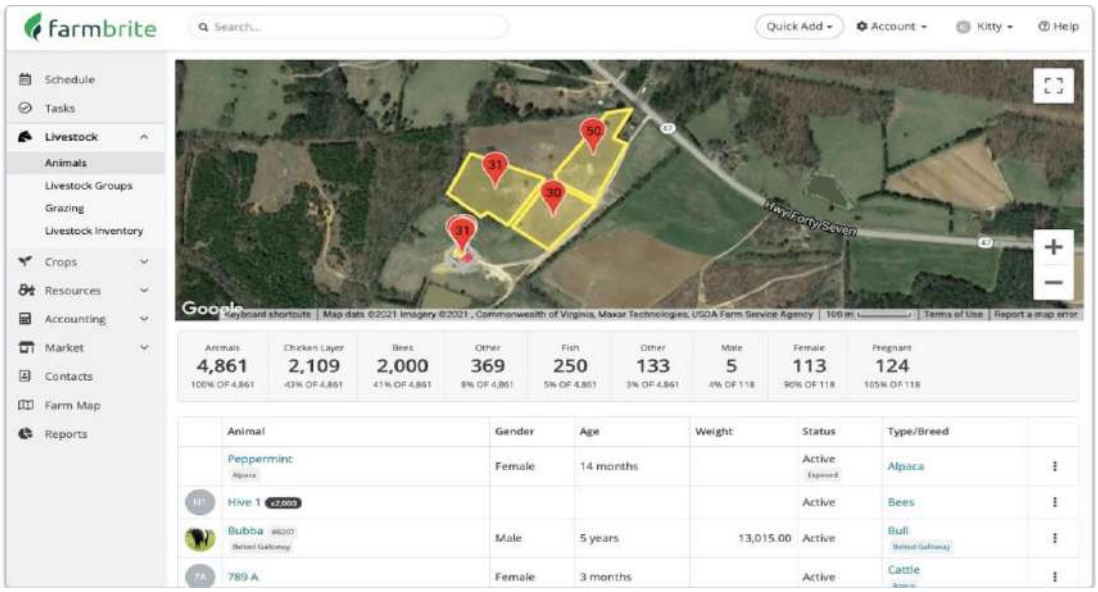


Figura 9-2. Plataforma Farmbrite.

Fuente: MODISAR, 2014.

2.10 Tarjetas de Control Embebido

En la actualidad las tarjetas embebidas tienen un gran crecimiento en todas aplicaciones de la electrónica debido al bajo coste y eficiencia que estos ofrecen, de aquí la importancia dentro de las áreas de la robótica, investigación e incluso incursionando con la tecnología IoT. De la gran variedad de tarjetas de hardware libre disponibles en el mercado, en la última década llevan liderando en el mercado nacional: Arduino, Rapsberry Pi y BeagleBoard. En la figura 10-2, se observa las tarjetas embebidas más comunes en IoT.

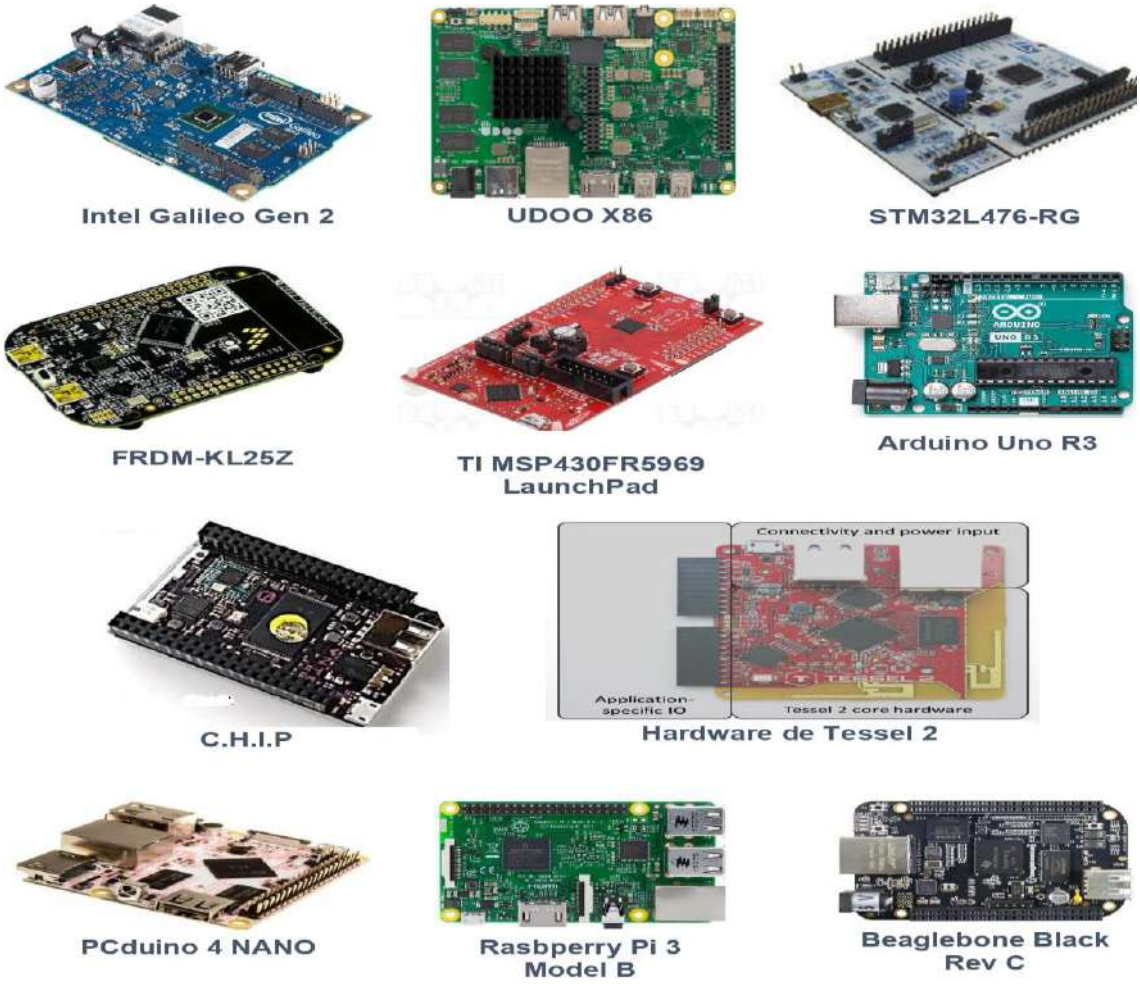


Figura 10-2. Tarjetas Embebidas más comunes en IoT.

Fuente: Roa Prieto & Rojas Scarpetta, 2019.

Realizado por: Acero O, 2023.

2.11 Fuente de alimentación o Batería

Hoy en día todos los dispositivos tecnológicos necesitan una fuente de alimentación para su funcionamiento, sin embargo, gracias a la electrónica se han desarrollado circuitos eléctricos y electrónicos sofisticados que demandan un bajo voltaje, de esta forma existen un sin número de baterías con diferentes características de acuerdo con los requerimientos del equipo o dispositivo.

En cuanto a los dispositivos IoT se necesita realizar una selección adecuada de la fuente de alimentación, tomando en consideración su tamaño, la corriente de consumo y el tiempo de operación. En la tabla 6-2, se muestra los diferentes medios de la alimentación usados en dispositivos IoT.

Tabla 6-2: Fuentes de alimentación en IoT.

Fuente de alimentación en sistemas IoT	Detalle
Baterías de plomo ácido	Usado en vehículos eléctricos, grandes infraestructuras y máquinas de IoT.
Baterias alkalinas	Usado comúnmente en la vida diaria como son controles remotos de TV, relojes, muchos dispositivos inalámbricos, entre otros.
Baterías de iones de litio	Se utilizan mundialmente en todas las aplicaciones modernas, como computadoras portátiles, teléfonos celulares, entre otros.
Baterías de polímero de litio	Están en teléfonos inteligentes, dispositivos portátiles, drones , UAV , equipos médicos, rastreadores GPS, dispositivos IoT , dispositivos LED portátiles, baterías UPS, farolas inteligentes y otros dispositivos.
Paneles solares	Usan el efecto fotovoltaico en semiconductores para generar energía para dispositivos IoT.
Baterías atómicas	Se usa en programas espaciales para alimentar satélites.

Fuente: Rojer, 2023.

Realizado por: Acero O, 2023.

CAPÍTULO III

3 MARCO METODOLÓGICO

En este aparatado se presenta la propuesta y diseño del prototipo con los procesos, materiales, enfoque, alcance, métodos, técnicas e instrumentos que conllevan a la realización de este trabajo de titulación, referente al monitoreo del ganado en tiempo real mediante geolocalización utilizando tecnología Lora.

En el gráfico 1-3, se muestra el diagrama metodológico propuesto del prototipo.

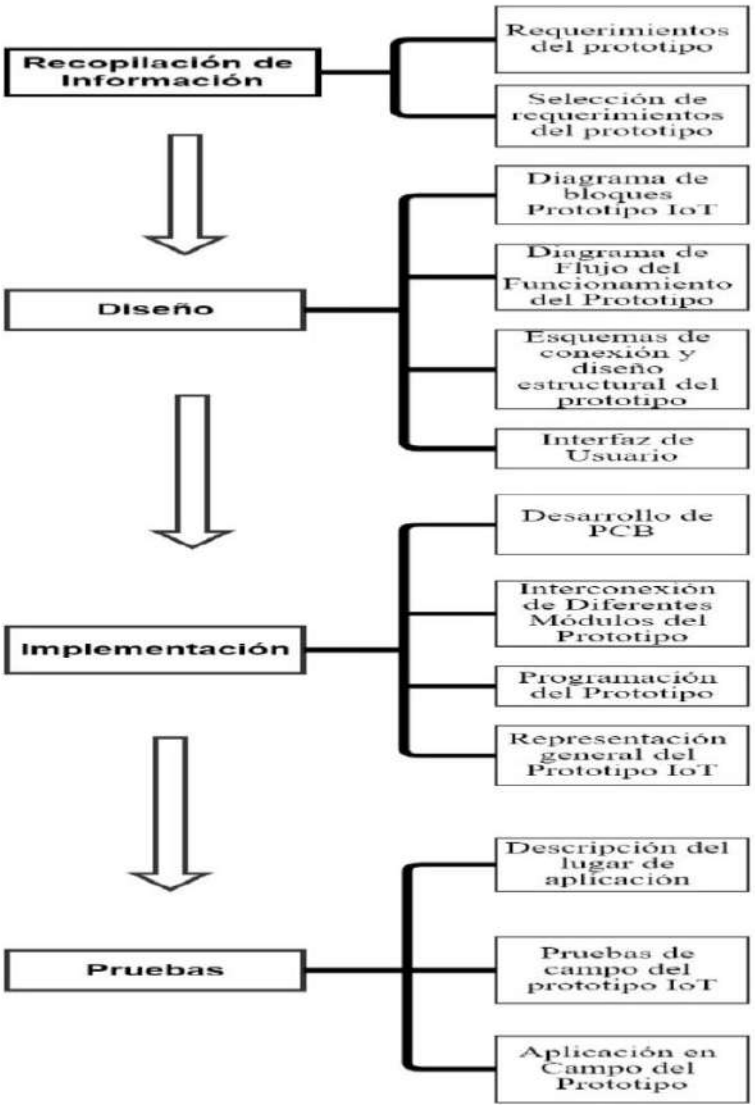


Gráfico 1-3. Diagrama metodológico propuesto del prototipo IoT.

Realizado por: Acero O, 2023.

3.1 Recopilación de Información

Para la recopilación de la información en este trabajo de investigación se basa en fuentes confiables, bases de datos, sitios web y artículos científicos con el fin obtener fuentes confiables de consulta para la implementación y obtención de resultados de las pruebas a realizarse.

3.1.1 Requerimientos del prototipo

La selección de los requisitos para el diseño y fabricación de un prototipo IoT eficiente, conllevan un previo estudio y el análisis de disponibilidad de los materiales en el mercado local. De esta podemos delimitar los requerimientos de hardware y software a involucrarse en el diseño del prototipo. A continuación, se describe cada uno de ellos.

3.1.1.1 Requerimientos de Hardware

- El funcionamiento del prototipo será de 38.7 horas aproximadamente.
- Dispondrá de dos nodos recolectores y un nodo Gateway para el envío y recepción de datos.
- Cada dispositivo nodo recolector incluirá un módulo GPS, un módulo Lora y microcontrolador.
- El nodo Gateway para la recepción y almacenamiento de datos en la nube, estará compuesto por un módulo Lora, Arduino nano y una tarjeta Raspberry PI.
- La alimentación del prototipo será mediante baterías recargables, cuyo tiempo de descarga dependerá del consumo de los diferentes componentes que lo conforman.

3.1.1.2 Requerimientos de software

La programación del prototipo debe presumir diversas funciones, tales como:

- El prototipo recolectará los datos en coordenadas geográficas de la posición de cada ganado objeto de prueba.
- La base de datos alojada en la nube podrá almacenar la información recibida desde el nodo recolector. Envió y recepción de datos recolectados desde los nodos recolectores hacia el nodo Gateway para ser almacenadas para posterior visualización mediante la aplicación móvil.

- El ganado será monitoreado mediante una aplicación móvil la cual indica al usuario la ubicación de cada dispositivo nodo recolector, el mismo que está implementado en un bovino para poder obtener una alerta oportuna en caso de abigeato.

3.1.2 Elementos que conforman el prototipo IoT

En este apartado se exponen todos los elementos que conforman el prototipo, estos son seleccionados y adquiridos de acuerdo con las características y necesidades del dispositivo a implementar, las mismas se describen a continuación.

3.1.2.1 Módulo LC86LICMD

Las características de este módulo nos permiten obtener la posición de cualquier objeto estático o en movimiento, siendo así seleccionado por los bajos niveles de señal, consumo de energía mínimo y por su alta eficiencia y sensibilidad al cambio que este componente tiene para ofrecer. En la figura 1-3, se muestra el Módulo GPS LC86L.

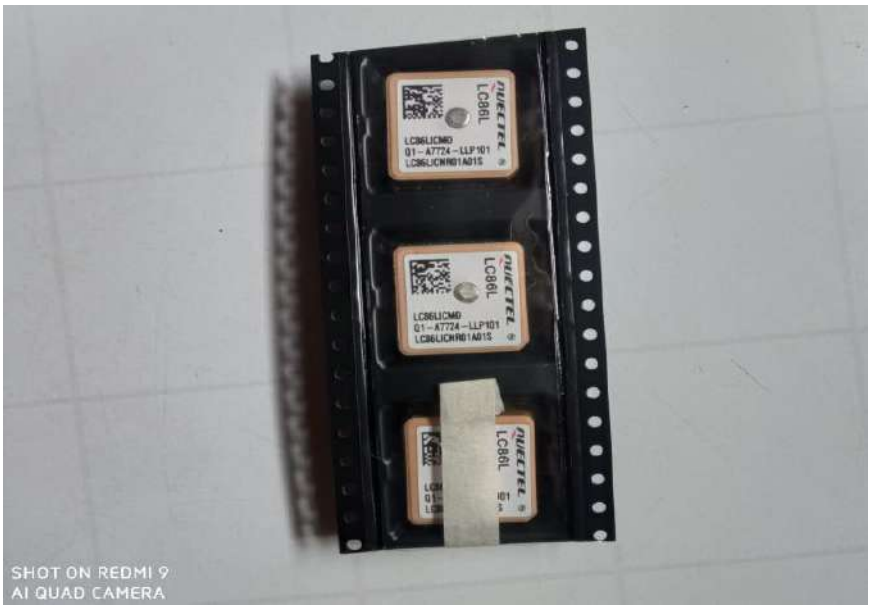


Figura 1-3. Módulo LC86L.

Realizado por: Acero O, 2023.

En la tabla 1-3, se detalla las características del Módulo GPS LC86L.

Tabla 1-3: Especificaciones del Módulo GPS LC86L.

Característica del producto	Valor
Categoría	Módulos GNSS / GPS
Precisión de la posición horizontal	2.5 m

Voltaje de alimentación - Mín.	2.8 V
Voltaje de alimentación-Máx.	4.3 V
Interfaz	UART
Temperatura de trabajo máxima	+ 85 °C
Temperatura de trabajo mínima	- 40 °C
Sensibles a la humedad	Si
Voltaje de alimentación operativo	2.8 V a 4.3 V

Fuente: (QUECTEL, 2022)**.**

Realizado por: Acero O, 2023.

3.1.2.2 AI Thinker LoRa Serie Ra-02

Este módulo Lora es ideal para trabajar con transmisiones a largas distancias, ya que proporciona un radio aproximado de 10 km a la redonda, siendo suficiente para el perímetro que se desea experimentar, sin descartar un inferior gasto de energía que este requiere. En la figura 2-3, se muestra el Módulo Transceptor LoRa SX1278 Ra-02 433 Mhz.

Figura 2-3. Módulo Transceptor LoRa SX1278 Ra-02 433 Mhz.

Realizado por: Acero O, 2023.

En la tabla 2-3, se detalla las características del Módulo Transceptor LoRa SX1278 Ra-02.

Tabla 2-3: Especificaciones del Módulo Transceptor LoRa SX1278 Ra-02.

Característica del producto	Valor
Antena (Ra-02)	Externa IPEX
Estándar inalámbrico	433MHz
Rango de frecuencia	420 ~ 450MHz
Interfaz	SPI GPIO
Voltaje	1,8 ~ 3,7 V, valor típico es 3,3 V
Temperatura de trabajo	-40 a + 85 ° C

Fuente: Satyavathi et al, 2022

Realizado por: Acero O, 2023.

3.1.2.3 TARJETA RASPBERRY PI 3 Modelo B

Es una computadora única montada en una placa, con características de transmisión similares a un ordenador, cuenta con tecnologías wifi y Bluetooth, como se muestra en la figura 3-3. Por tal razón se escogió esta tarjeta de desarrollo por múltiples funciones. En la tabla 3-3, se detalla las características del Módulo Raspberry Pi 3B.

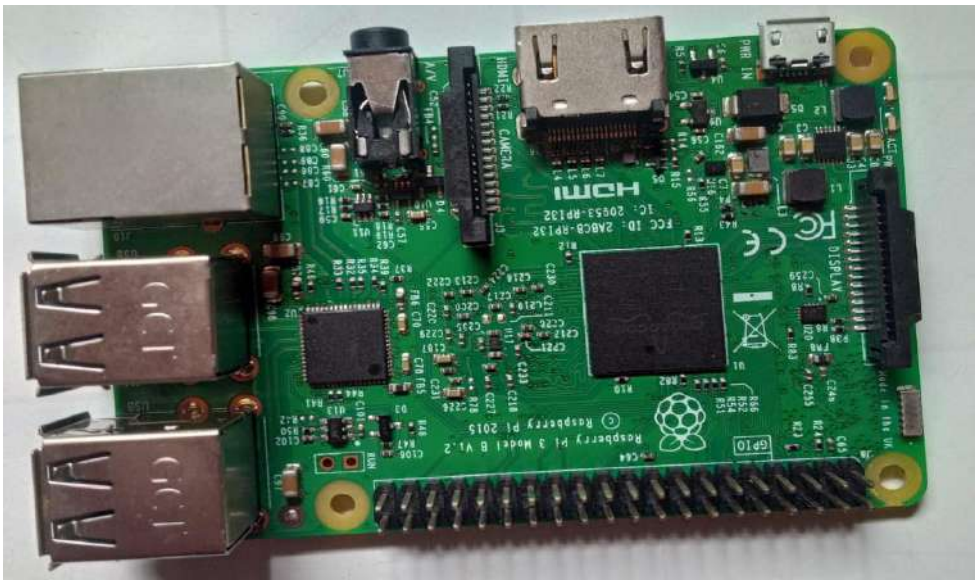


Figura 3-3. Raspberry PI 3 B+.

Realizado por: Acero O, 2023.

Tabla 3-3: Especificaciones del Módulo Raspberry PI 3 B+.

Característica del producto	Valor
Procesador	ARM de cuatro núcleos de 1.2 GHz de 64 bits.
Red	LAN inalámbrica 802.11n.
Versión Bluetooth	4.1
Memoria	1 GB de RAM
Puertos USB	4
Biblioteca de soporte hardware	Python, C / C ++ y bash.
Sistema Operativo	Raspbian y/o NOOBS (sistemas operativos basados en Linux)
Pines	GPIO extendido de 40 pines
Puerto Micro SD	Si
Fuentes de alimentación	3.3 V y 5 V.

Fuente: RASPBERRY PI, 2014.

Realizado por: Acero O, 2023.

3.1.2.4 Arduino Nano V3.0

Para el proyecto se utilizó el Arduino Nano 3.0, el cual está basado en un MCU Atmega328p, como se observa en la figura 4-3, siendo una de las tarjetas de desarrollo más comunes en el campo de la electrónica incluso para trabajar con sistemas de IoT. Gracias a su tamaño, capacidad

y sobre todo a su software libre permite una gestión más sencilla en todo su ámbito. En la tabla 4-3, se detalla las características de placa de desarrollo Arduino Nano 3.0.

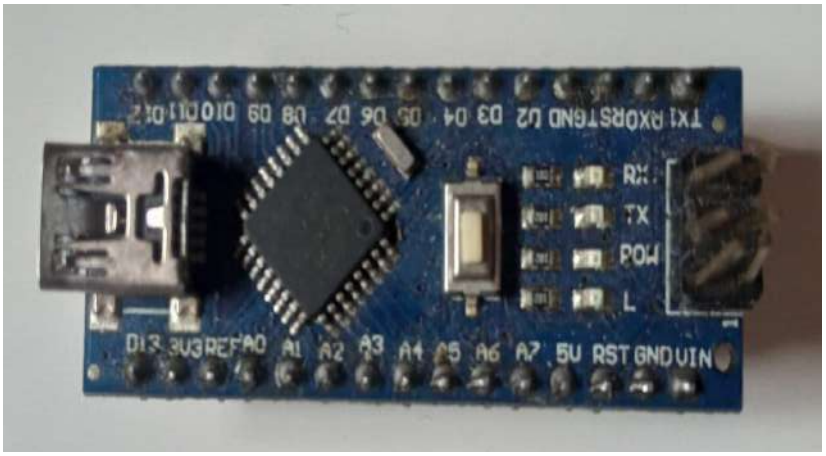


Figura 4-3. Arduino Nano V3.0.

Realizado por: Acero O, 2023.

Tabla 4-3: Especificaciones de placa de desarrollo Arduino Nano 3.0.

Característica del producto	Valor
Microcontrolador	Atmel ATmega328p
Voltaje de funcionamiento	5 V
Voltaje de entrada (recomendado)	7 a 12 V
Voltaje de entrada (máximo)	6 a 20 V
Pines de E/S digitales	14 (6 pines con salida PWM)
Pines de entrada analógica	8
Corriente CC por pin de E/S	40 mA
Memoria flash	32 K (2 KB para gestor de arranque)
SRAM	2 KB
EEPROM	1 KB
Frecuencia	16 MHz

Fuente: NANO, 2018.

Realizado por: Acero O, 2023.

3.1.2.5 Arduino Uno R3

Es una de las placas más populares dentro de la familia Arduino, conformado de un microcontrolador Atmega 328p. Por su precio asequible y fácil configuración fue usado en el nodo Gateway, a pesar de su tamaño cumple con las funcionalidades de otra tarjeta de desarrollo similar, junto con un módulo Lora y una antena recibirán toda la información proveniente del nodo recolector. En la figura 5-3, se muestra una tarjeta de desarrollo Arduino Uno R3.

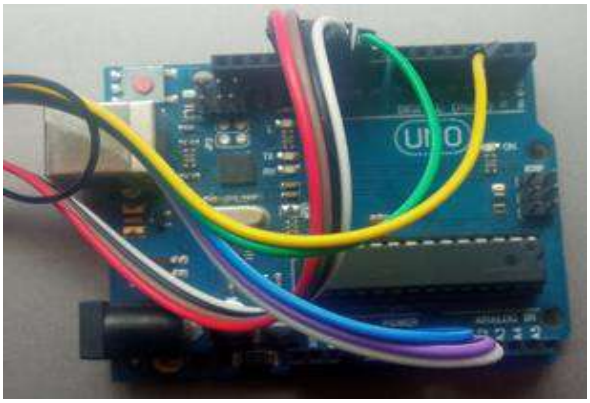


Figura 5-3. Tarjeta de desarrollo Arduino Uno R3.

Realizado por: Acero O, 2023.

En la tabla 5-3, se detalla las características del Arduino Uno R3.

Tabla 5-3: Especificaciones de la tarjeta Arduino Uno R3.

Característica del producto	Valor
Fabricante	Arduino LLC
Voltaje de operación	5 VDC
Voltaje de entrada (recomendado)	7 a 12V
Voltaje de entrada (máximo)	6 a 20V
Memoria flash	32 KB
SRAM	2 KB
EEPROM	1 KB
Frecuencia	16 Mhz
Pines de E/S digitales	14 (6 pines con salida PWM)
Pines de entrada analógica	6
Corriente CC por pin de E/S	20 mA

Fuente: ARDUINO, 2017.

Realizado por: Acero O, 2023.

3.1.2.6 Antena Lora IPX y Antena de resorte

La antena es uno de los accesorios indispensables para los módulos transceptores inalámbricos, como Lora, ya que permite tener un buen rango de señal y una mejor cobertura. De esta forma los módulos pueden emitir y recibir con mayor eficiencia la información de un punto a otro sin perder la comunicación debidos factores ambientales.

En este proyecto se utilizó dos tipos de antenas, la primera es una Antena de resorte de cobre 433 MHz transmisor de receptor RF como se muestra en la figura 6-3 y el segundo es una Antena Lora ipx 433MHz Hotspot rp-SMA visto en la figura 7-3, siendo compatibles con el Módulo Transceptor LoRa SX1278 Ra-02.

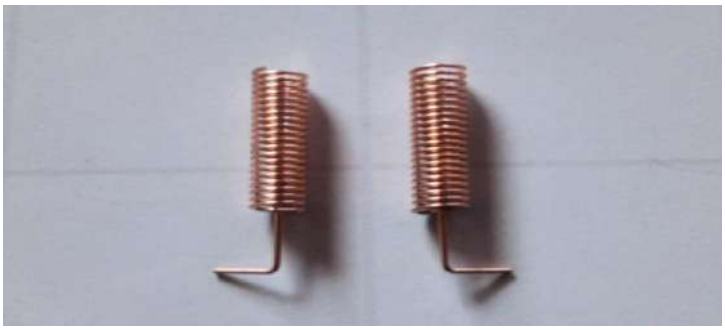


Figura 6-3. Antena de resorte de cobre 433 MHz transmisor de receptor RF.

Realizado por: Acero O, 2023.

En la tabla 6-3, se detalla las características de la Antena de resorte de cobre 433 MHz transmisor de receptor RF.

Tabla 6-3: Especificaciones de la Antena de resorte de cobre 433 MHz transmisor de receptor RF.

Característica del producto	Valor
Rango de frecuencia	433 MHz
Relación de onda estacional	1,5
Impedancia de entrada	50 Ω
Potencia máxima	10 W
Ganancia	2.15 a 4 dBi
Peso	0.07 oz.
Color de la antena	Cobre
Forma de interfaz	Soldadura directa

Fuente: SHENZHEN TAIDA CENTURY TECHNOLOGY CO,LTD, 2013

Realizado por: Acero O, 2023.

Figura 7-3. Antena Lora ipx 433MHz Hotspot rp-SMA.

Realizado por: Acero O, 2023.

En la tabla 7-3, se detalla las características de la Antena Lora ipx 433MHz Hotspot rp-SMA.

Tabla 7-3: **Especificaciones de la Antena Lora ipx 433MHz Hotspot rp-SMA**

Característica del producto	Valor
Frecuencia	433 MHz
Ganancia	3 dB
VSWR	<1.5
Impedancia	50 Ω
Conector de antena	rp-SMA macho
Longitud de la antena	4.3 in
Tipo de cable	rp-Sma hembra cable WiFi Pigtail a Ufl./ipx
Tipo de cable	RF1.13
Longitud del cable	5.9 in

Fuente: SHENZHEN TAIDA CENTURY TECHNOLOGY CO,LTD, 2013.

Realizado por: Acero O, 2023.

3.1.2.7 Batería de iones de litio.

Para la autonomía del dispositivo nodo recolector se utilizaron dos baterías de iones de litio recargables, las cuales destacan en el mercado por su capacidad y calidad al momento de almacenar por más tiempo, sin excluir su tamaño, forma de construcción y tienden a ser menos defectuosas en comparación a las baterías Li-Po. Siendo ideal para este tipo de proyectos donde se requiere mayor eficiencia.

En la figura 8-3, se observa la batería que se usó como fuente de alimentación para el nodo recolector. Es una batería recargable iones de litio de 2500mAh del fabricante Samsung SDI (Samsung SDI, 2020), las características se muestran a continuación:

- Tipo: Batería iones de litio.
- Voltaje nominal: 3.7 V.
- Capacidad: 2500mAh.
- Corriente de carga: Estándar 1.25A y carga rápida Máx. 4A.

Figura 8-3. Batería Samsung de iones de litio 2500 mAh, 3.7 V.

Realizado por: Acero O, 2023.

3.1.2.8 Lenguajes de programación del prototipo

Para efectuar el desarrollo del prototipo, en la parte del nodo recolector es necesario configurar los parámetros del módulo Lora, obtener datos desde el módulo GPS, enviarlos y gestionar el funcionamiento de cada componente. En el caso del nodo Gateway es necesario gestionar los datos recibidos mediante el módulo Lora para almacenarlo en la nube y mostrarlos en la interfaz de usuario. A continuación, se describe los lenguajes de programación usados en usados en el Prototipo.

Arduino IDE: Las diferentes tarjetas Lora son compatibles con la programación en Arduino IDE, de igual forma se dispone de una amplia variedad de librerías para el control de GPS y otros módulos.

Python: Es un lenguaje de alto nivel con mayor crecimiento en el mundo, usado por profesionales de diferentes áreas, desde estudiantes hasta científicos y matemáticos. Gracias a ser un lenguaje multipropósito contempla diversas tareas como la creación aplicaciones web, aplicaciones móviles y de escritorio. Además, viene preinstalada en la Raspberry lo que permite controlar los pines de propósito general e interactuar con el exterior utilizando módulos y sensores.

Firebase: Es un almacén de datos centrado en Front-End de una aplicación móvil, es decir en el diseño. Por su parte el Back-End permite a los clientes la facilidad de almacenar datos en Firebase y obtenerlos directamente del Firebase sin la necesidad de un servidor, todo esto en tiempo real.

Una de las herramientas más destacadas y esenciales de Firebase son las bases de datos en tiempo real. Estas se almacenan en la nube, permiten alojar y disponer de los datos e información de la

aplicación en tiempo real, manteniéndolos actualizados, aunque el usuario no realice ninguna acción. Tiene un paquete gratuito con ciertos límites en cuanto al tamaño de la base de datos y número de interacciones por día, ofrecen más paquetes a diferentes precios.

Android Studio: Es un IDE que se compone de varias herramientas de desarrollo para la creación de aplicaciones móviles, simplificándose así el trabajo de codificación en mucho menos tiempo. Con este software se puede visualizar en vivo la creación de nuestro diseño y posee un emulador en tiempo real que simula ser un dispositivo móvil.

3.2 Diseño del prototipo

Para el diseño del prototipo se requirió la implementación y conexión de los diferentes elementos que conforman trato el nodo receptor como el nodo Gateway, para el monitoreo mediante tecnología Lora que permite mantener una interfaz de comunicación con el dispositivo móvil para el monitoreo en tiempo real de los animales.

3.2.1 Diagrama de bloques del prototipo IoT

Para el desarrollo del prototipo se describen cuatro bloques: Recolección y procesamiento, recepción, almacenamiento y ejecución, los cuales consideran los aspectos más importantes de su funcionamiento. En el grafico 2-3, se muestra el diagrama de bloques del prototipo IoT.

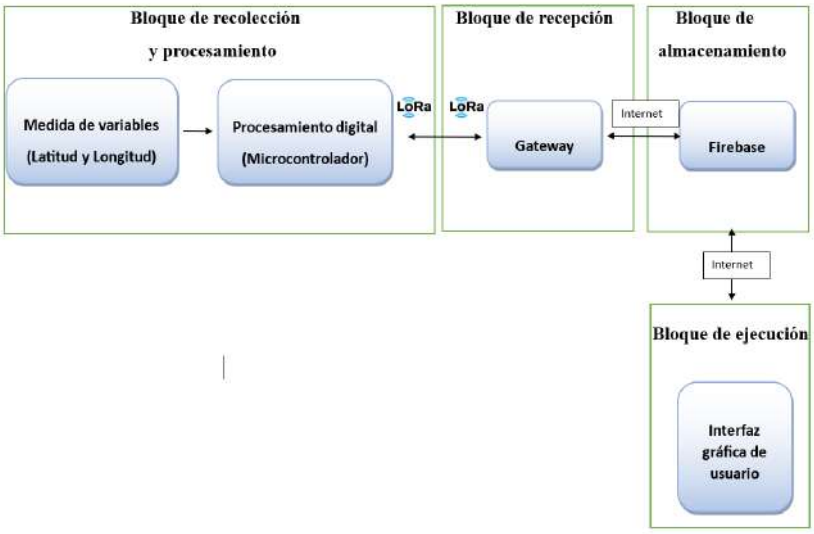


Gráfico 2-3. Diagrama de bloques del prototipo IoT.
Realizado por: Acero O, 2023.

A continuación, se describe cada uno de los bloques que conforman el prototipo de monitoreo.

3.2.1.1 Bloque de recolección

Este bloque recolecta los datos de la ubicación del ganado en coordenadas geográficas mediante el módulo GPS, los datos son procesados por un microcontrolador y enviados mediante el módulo Lora hacia el bloque de recepción.

3.2.1.2 Bloque de recepción

El bloque de recepción consiste en recibir los datos provenientes de los nodos recolectores, mediante el uso de la tecnología Lora.

3.2.1.3 Bloque de almacenamiento

Recibe la información del bloque de recepción mediante internet y lo almacena en una base datos en la nube llamada “Firebase”.

3.2.1.4 Bloque de monitoreo

En este bloque la aplicación móvil recoge los datos almacenados en la nube y visualiza en tiempo real la posición del ganado, de igual forma alerta al usuario en caso de abigeato.

3.2.2 Diagrama de Flujo del Funcionamiento del Prototipo

La lógica empleada en el prototipo es integral para evitar errores. El prototipo se divide en tres diagramas, en el gráfico 3-3, se muestra el diagrama de flujo del nodo recolector. En el gráfico 4 -3, se muestra el diagrama de flujo del nodo Gateway y en el gráfico 5-3, se muestra el diagrama de flujo de la aplicación móvil.

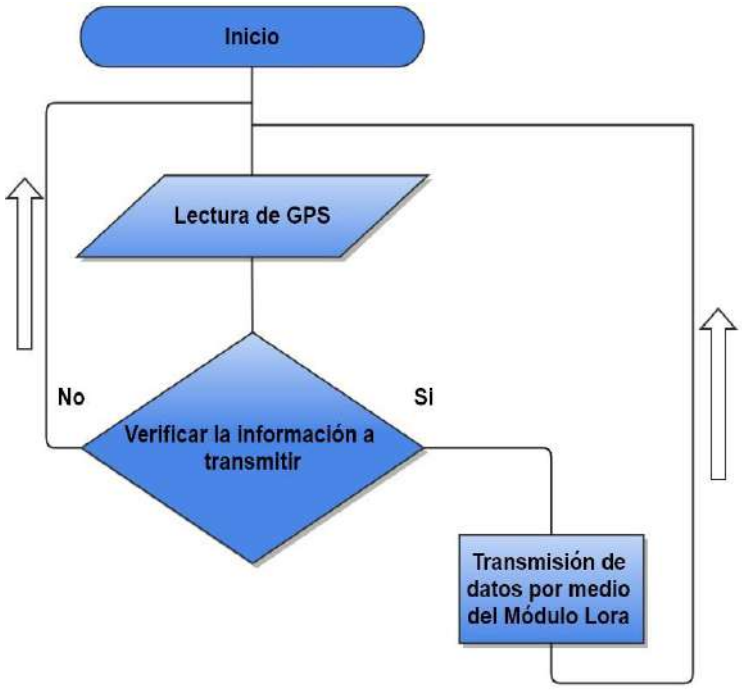


Gráfico 3-3. Diagrama de flujo del nodo recolector

Realizado por: Acero O, 2023.

Este diagrama hace referencia a la recolección de datos mediante el módulo GPS en base a coordenadas geográficas, de esta forma se transmiten los datos mediante un Módulo Lora hacia el nodo Gateway. Al terminar el proceso, continua con la lectura del módulo GPS para seguir validando las coordenadas de los animales objeto de estudio.

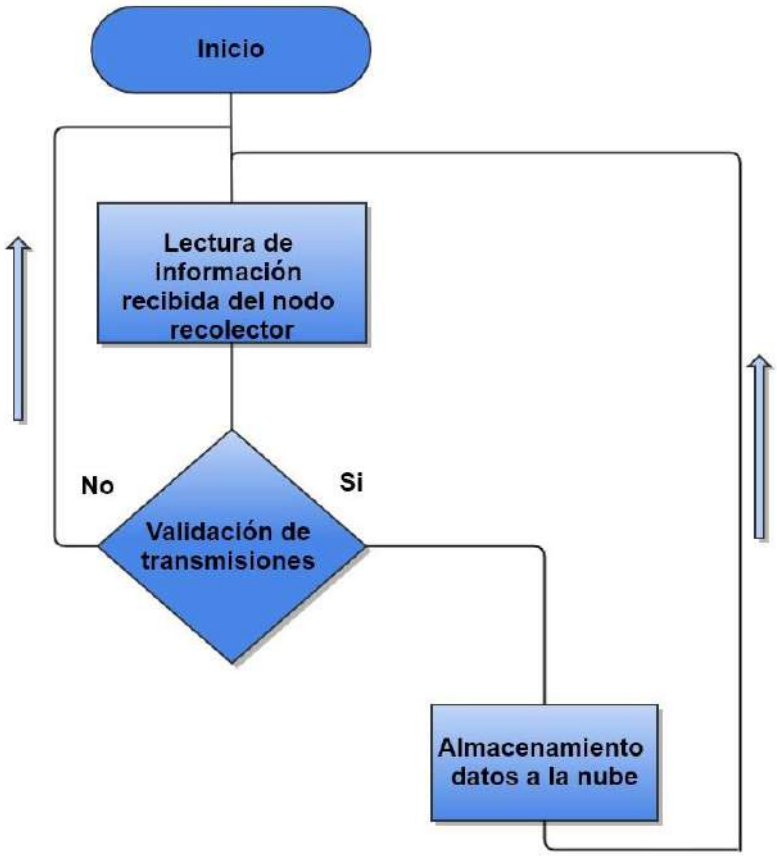


Gráfico 4-3. Diagrama de flujo del nodo Gateway.

Realizado por: Acero O, 2023.

El diagrama se enfoca en validar las transmisiones de información del nodo Recolector para ser almacenados en el DATA SET que se encuentra en la nube, caso contrario esperará la trasmisión de datos desde nodo recolector.

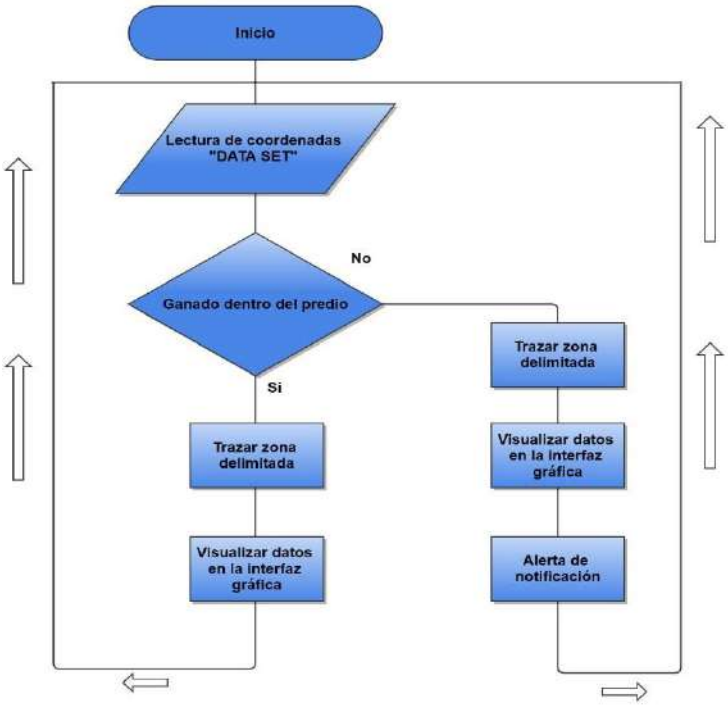


Gráfico 5-3. Diagrama de flujo de la aplicación móvil.

Realizado por: Acero O, 2023.

Este diagrama muestra el procedimiento de la ubicación en tiempo real del animal objeto de estudio, mediante la lectura del DATA SET almacenado en la nube. De esta forma si las coordenadas enviadas por el nodo recolector se encuentran dentro del perímetro delimitado se procede a graficar en la pantalla del dispositivo móvil.

3.2.3 Esquema de conexiones y estructura de los nodos.

Para la construcción del prototipo se empleó la distribución de pines y circuitos de conexión tanto para el nodo recolector como para el nodo Gateway, considerando los diferentes aspectos y funciones que cada una uno conlleva.

3.2.3.1 Nodo recolector

El circuito de conexión para el nodo recolector se realizó con la herramienta Proteus, como se muestra en la figura 9-3.

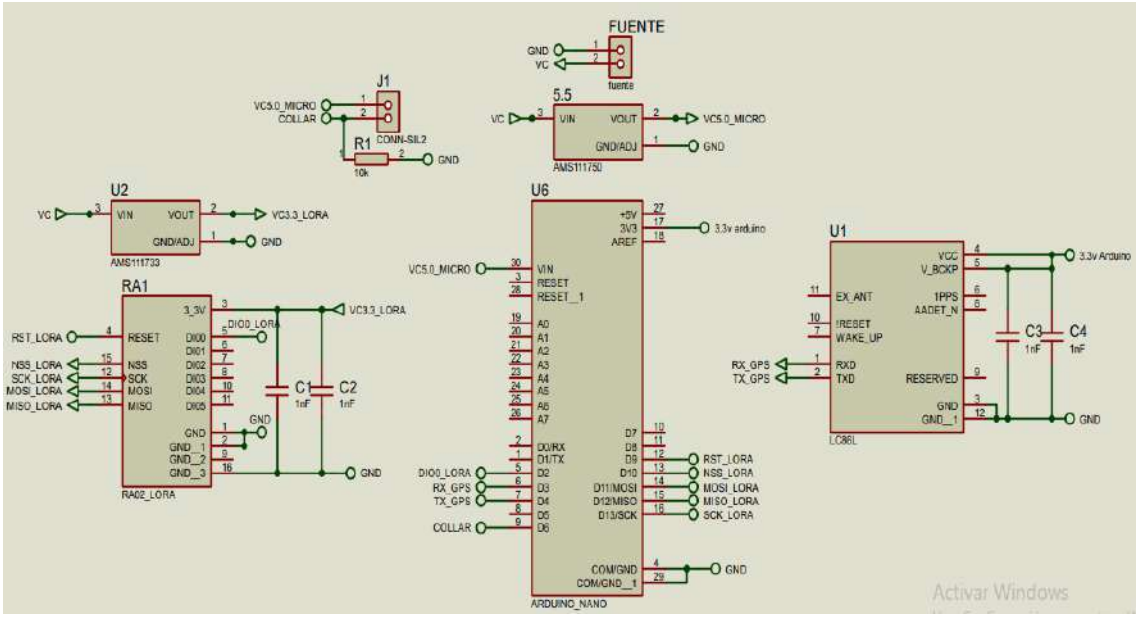


Figura 9-3. Circuito del nodo recolector.

Realizado por: Acero O, 2023.

En el Anexo A, se observa la configuración de pines del Arduino nano V3.0, conformado por un MCU Atmega328p. En la tabla 8-3, se muestra los pines para la conexión de los diferentes componentes.

Tabla 8-3: Asignación de pines del Arduino nano V3.0 al Módulo GPS y Módulo LoRa.

Componente	Puerto Componente	Puerto Arduino nano V3.0
Módulo GPS	RX	D2
	TX	D3
	VCC	3V3
	GND	GND
Módulo Transceptor LoRa SX1278 Ra-02 encapsulado	SCK	D13
	MISO	D12
	MOSI	D11
	NSS	D10
	RST	D9
J1	VC5.0 V	COLLAR
	COLLAR	GND, D6

Realizado por: Acero O, 2023.

La alimentación del nodo recolector se realizó con dos reguladores de 5 V y 3.3 V, en la tabla 9-3, se muestra los pines de conexión para la alimentación de los componentes.

Tabla 9-3: Asignación de pines de la batería de iones de litio al nodo recolector.

Fuente	Puerto componente	Componente	Puerto Arduino nano V3.0	Puerto Módulo Lora Ra-02
Batería Samsung de iones de litio 2500 mAh, 3.7 V	VC	Regulador VC 5.0V	VIN	Puerto no disponible
		Regulador VC 3.3V	Puerto no disponible	3_3v
	GND	GND	GND	GND, GND_1

Realizado por: Acero O, 2023.

3.2.3.2 Nodo Gateway

El circuito de conexión para el nodo Gateway se realizó con la herramienta Proteus, como se muestra en la figura 10-3.

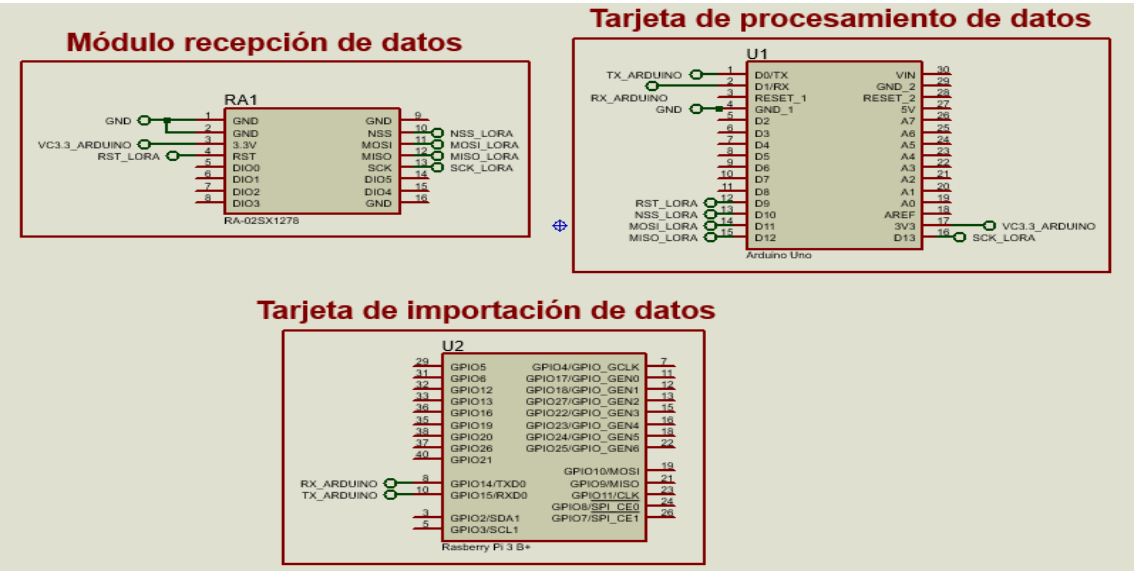


Figura 10-3. Circuito del nodo Gateway.

Realizado por: Acero O, 2023.

En la tabla 10-3, se observa los pines para la conexión de los diferentes componentes con el Arduino UNO R3.

Tabla 10-3: Asignación de pines del Arduino UNO R3 al Módulo LoRa SX1278.

Componente	Puerto Componente	Puerto Arduino UNO R3.
	SCK	D13
	MISO	D12
	MOSI	D11
	NSS	D10

Módulo Transceptor LoRa SX1278 Ra-02	RST	D9
	3.3V	3V3
	GND	GND

Realizado por: Acero O, 2023.

Para la conexión de la tarjeta Raspberry PI 3 B+, en el Anexo D se muestra la configuración de pines, donde esta tarjeta consta de un protocolo de comunicación conformado por los puertos GPIO14 (TXD) y GPIO15 (RXD) los cuales están conectados a los puertos TX(J5) y RX(J6) del Arduino nano respectivamente. En la tabla 11-3, se observa los pines para la conexión de la tarjeta Raspberry PI 3 B+.

Tabla 11-3: Asignación de pines del Arduino UNO R3 a la tarjeta Rasberry PI 3 B+.

Componente	Pin Componente	Puerto Arduino UNO R3
Raspberry PI 3 B+	GPIO14 (TXD)	RX
	GPIO15 (RXD)	TX

Realizado por: Acero O, 2023.

3.2.4 Diseño PCB del prototipo

Se desea un esquemático sencillo pero eficaz que ocupe el menor espacio posible, para que los dispositivos tanto del nodo Recolector y Gateway sean bastante prácticos. Por tal razón se optó un esquema embebido a doble cara en el PCB.

En la figura 11-3, se muestra el diseño en PCB para el nodo Recolector. Este diseño se realizó usando el software Proteus. Además del microcontrolador, módulo Lora, módulo GPS y una antena, cuenta con un circuito regulador de voltaje, el cual permite que se alimente tanto los módulos como el MCU, haciendo que la placa no se dañe en caso de que un componente llegara a fallar consumiendo demasiada corriente.

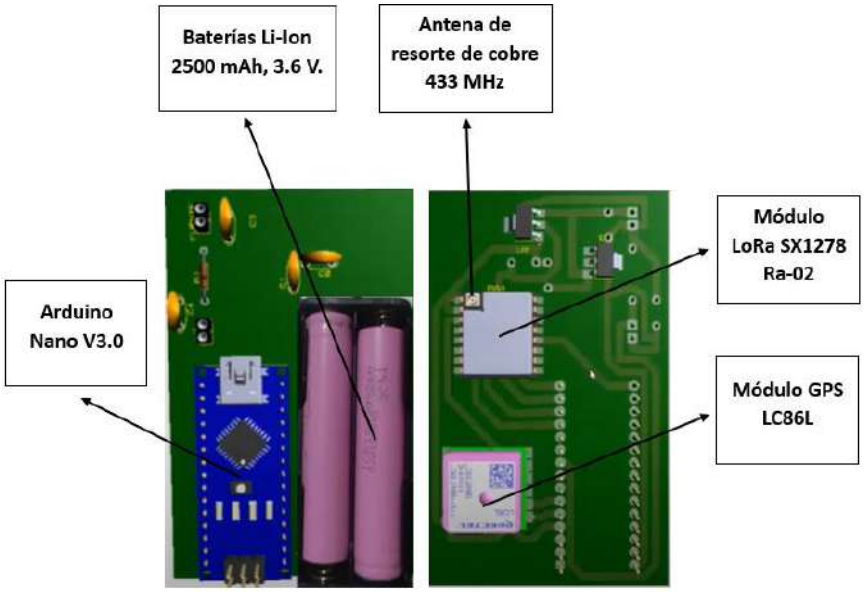


Figura 11-3. Diseño PCB doble cara para nodo el recolector.

Realizado por: Acero O, 2023.

En la figura 12-3, se muestra el diseño en PCB de la placa electrónica para nodo Gateway que se colocara en la Rasberry PI 3. La placa está conformada por un módulo Lora, un Arduino nano y una Antena RF.

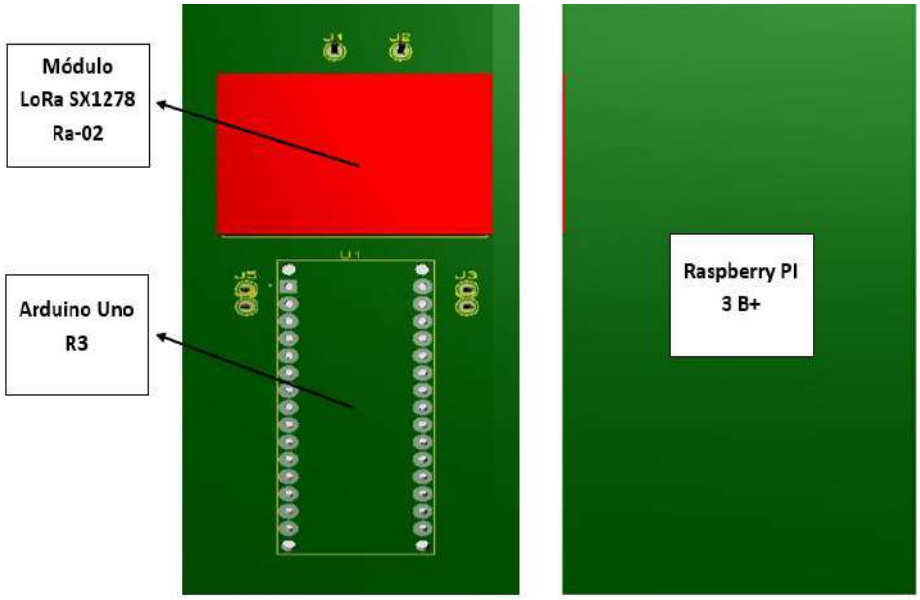


Figura 12-3. Diseño PCB de la placa electrónica para nodo Gateway.

Realizado por: Acero O, 2023.

3.2.4.1 Diseño estructural de la caja nodo recolector.

Para el diseño de la caja contenedora del dispositivo nodo recolector, se utilizó el software SolidWorks, el cual permite modelar piezas en tres dimensiones, para posteriormente imprimir el modelo terminado en una impresora 3D, de esta forma primero se diseñó la caja con un agujero lateral para la salida de la antena, como tal considerando las medidas de la PCB y sujeta de una correa que servirá de collar para el animal a prueba. En la figura 13-3, se observa el diseño de la caja del nodo recolector con su correa.

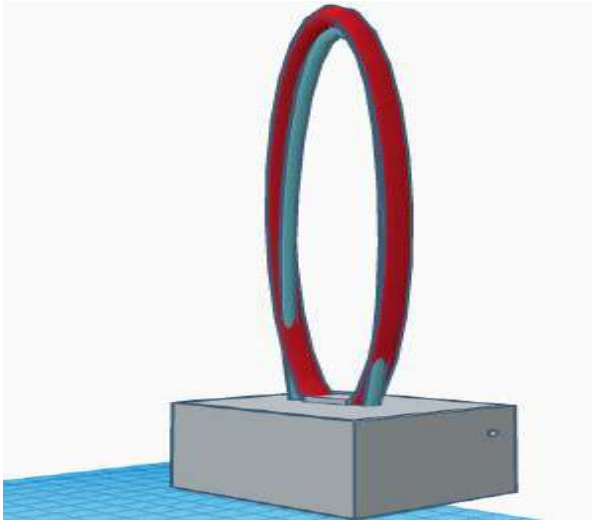


Figura 13-3. Diseño de caja contenedora del nodo recolector.

Realizado por: Acero O, 2023.

En la tabla 12-3, se muestran las dimensiones de la estructura de caja contenedora nodo recolector.

Tabla 12-3: Dimensiones de la caja nodo recolector.

Magnitud	Dimensión (cm)
Largo	10
Ancho	6.5
Altura	3.5

Realizado por: Acero O, 2023

Para alojar los componentes del nodo Gateway se diseñó una caja rectangular, con un orificio para la antena de esta forma evitando bloquear la señal de recepción que llega desde el nodo recolector. En la figura 14-3, se muestra el diseño de la caja contenedora del nodo Gateway.

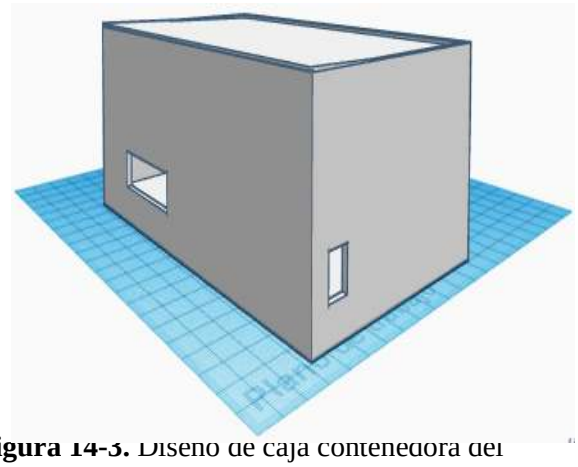


Figura 14-3. Diseño de caja contenedora del nodo Gateway.

Realizado por: Acero O, 2023.

En la tabla 13-3, se muestran las dimensiones de la estructura de caja contenedora nodo Gateway.

Tabla 13-3: Dimensiones de la caja nodo Gateway.

Magnitud	Dimensión (cm)
Largo	16
Ancho	10
Altura	5

Realizado por: Acero O, 2023.

3.2.5 Interfaz de Usuario

La pantalla inicial, mostrada en la figura 15-3, consiste en un ambiente interactivo en el dispositivo móvil, al cual se ingresa mediante un correo electrónico y contraseña del usuario. Por otro lado, se tiene la pantalla principal como se muestra en la figura 16-3, donde se encuentra un mapa de la zona para el monitoreo en tiempo real, de esta forma se determina la ubicación de los nodos recolectores y el lindero virtual delimitado.

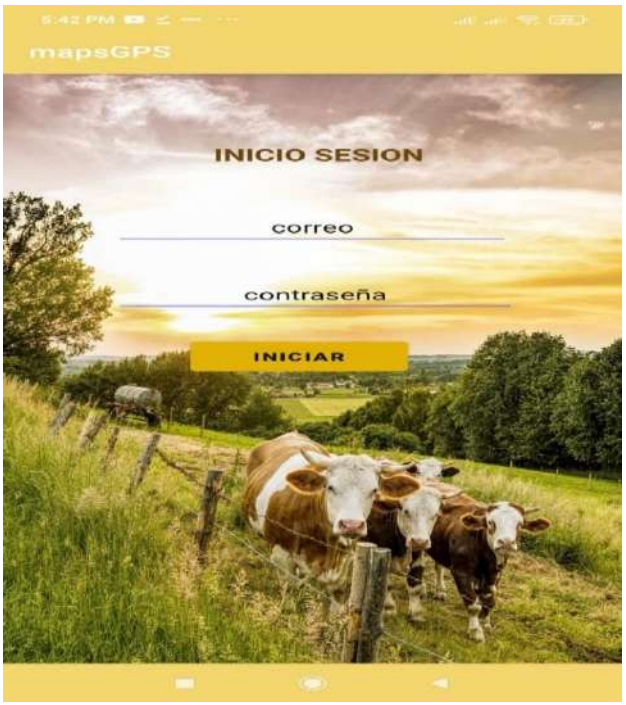


Figura 15-3. Pantalla inicial de la interfaz de usuario.

Realizado por: Acero O, 2023.

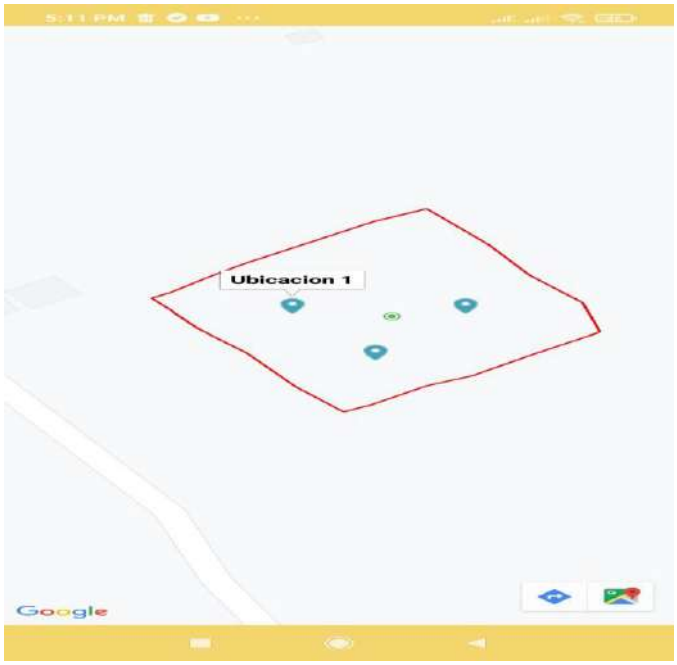


Figura 16-3. Pantalla principal de la interfaz de usuario.

Realizado por: Acero O, 2023.

CAPÍTULO IV

4 PROPUESTA Y DISEÑO DE PROTOTIPO

En este capítulo se realiza la implementación del prototipo en base a los diseños y esquemas de conexión descritos en el capítulo anterior. De esta forma se tiene una idea clara del funcionamiento del prototipo con respeto a los parámetros planteados.

En los últimos años la delincuencia ha mostrado su máximo nivel, tanto en las zonas urbanas como rurales, siendo está ultima la escogida por los bandidos para cometer abigeato, debido a la facilidad y la poca vigilancia que existe por parte de los ganaderos. Por este motivo fue necesario realizar un prototipo IoT experimental para el monitoreo en tiempo real de estos animales, de esta forma el ganadero podrá garantizar el bienestar de los mismos. En la figura 1-4, se observa el esquema general del funcionamiento del prototipo.

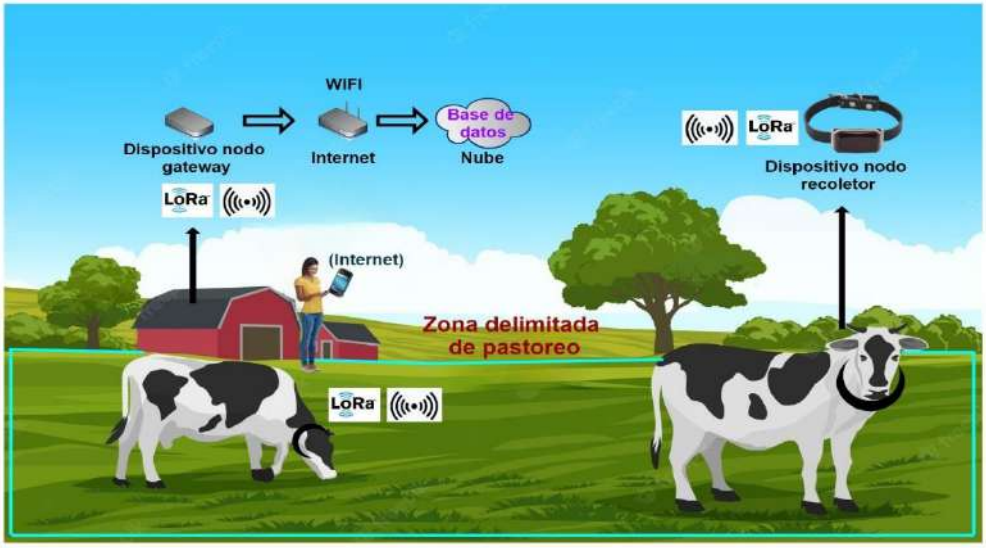


Figura 1-4. Representación del funcionamiento general del Prototipo IoT.

Realizado por: Acero O, 2023.

4.1 Implementación del prototipo de monitoreo

Al tener implementada la placas PCB, se procede a colocar los elementos. En la figura 2-4, se observa el resultado final para el nodo recolector.

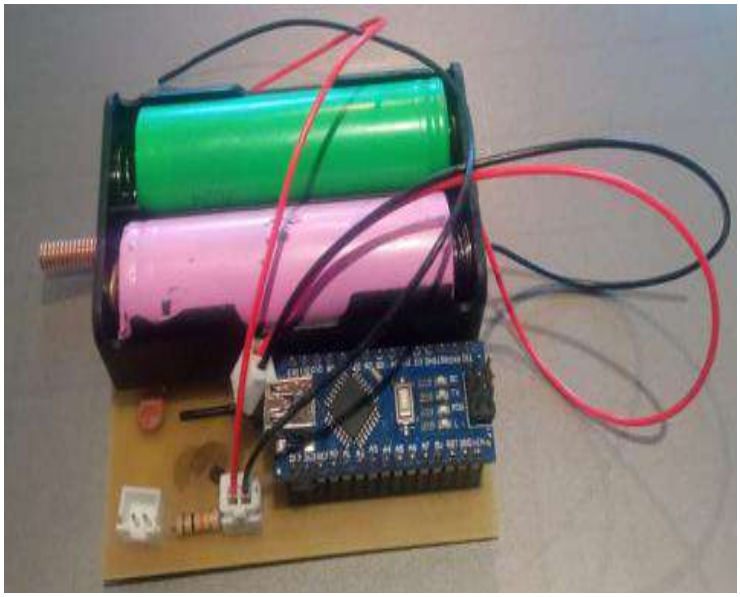


Figura 2-4. PCB implementada para el nodo recolector.

Realizado por: Acero O, 2023.

En la figura 3-4, se muestra el resultado final del nodo Gateway en el interior de la vivienda con los elementos ubicados en la placa.

Figura 3-4. PCB implementada para el nodo Gateway en el interior de la vivienda.

Realizado por: Acero O, 2023.

En la figura 4-4, se muestra un cajetín adicional para la antena Lora del nodo Gateway, el cual se ubicará en el exterior de la vivienda con línea de vista para toda el área delimitada.

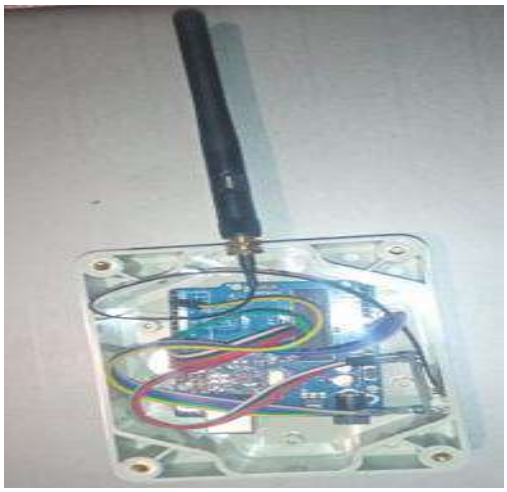


Figura 4-4. Cajetín para la antena Lora.
Realizado por: Acero O, 2023.

4.2 Programación del Prototipo

Una vez diseñados e implementados tanto el nodo recolector como el nodo Gateway, es necesario la programación con distintas instrucciones que se encarguen del monitoreo del ganado objeto de estudio. Es así como en esta sección se abordará la programación del nodo recolector y el nodo Gateway.

4.2.1.1 Diagrama de flujo del Nodo recolector

Es el dispositivo encargado de obtener y enviar los datos del GPS mediante el módulo Lora. En el gráfico 1-4, se observa el diagrama de flujo del nodo recolector.

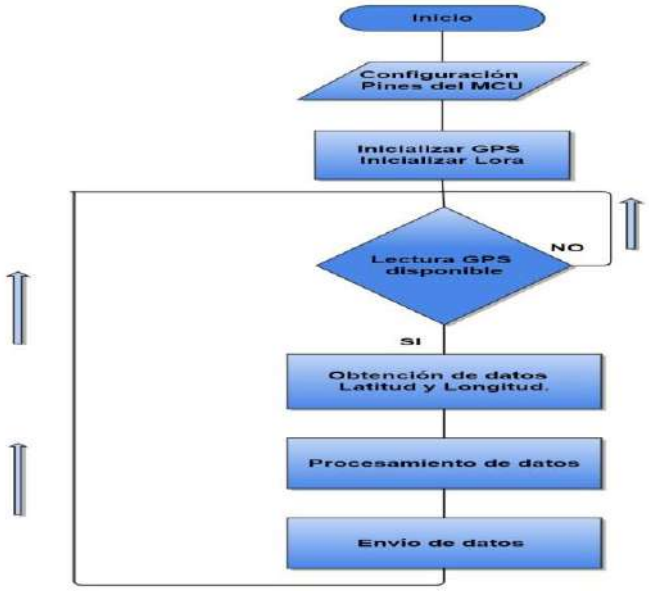


Gráfico 1-4. Diagrama de flujo del programa principal del nodo recolector.

Realizado por: Acero O, 2023.

Las principales librerías utilizadas en la programación del prototipo son:

- **TinyGPS.h**: Es una librería diseñada por Mikal Hart, con una gran funcionalidad GPS NMEA, convierte los datos de posicionamiento en variables fáciles de usar para latitud, longitud, tiempo, entre otros; minimizando de esta forma grandes códigos que pudieran requerir.
- **LoRa.h :** Es una librería que facilita trabajar con chips LoRa Serie SX1278.
- **SPI.h:** Es un protocolo de datos que ayuda a comunicarse en distancias cortas a los dispositivos periféricos y microcontroladores como dispositivo maestro.

4.2.1.2 Diagrama de flujo del Nodo Gateway para envió y recepción de datos

El programa encargado de gestionar la información recibida desde el nodo recolector a través del módulo Lora conectado al Arduino nano, se desarrolla de igual forma que el nodo recolector en Arduino IDE, haciendo el uso de librearías ya mencionadas anteriormente. En el gráfico 2-4, se observa el diagrama de flujo recepción y envió de datos.

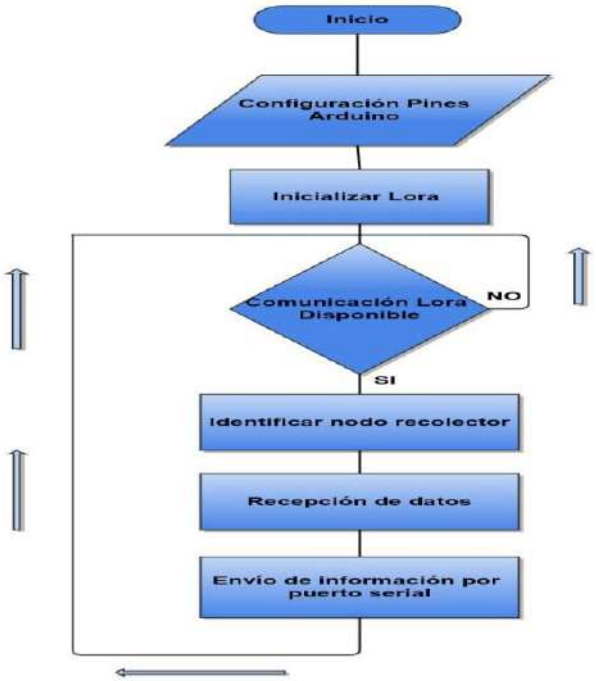


Gráfico 2-4. Diagrama de flujo envío de datos nodo Gateway.

Realizado por: Acero O, 2023.

El programa se desarrolló en lenguaje Python para la tarjeta de desarrollo Raspberry Pi. El cual obtiene la información desde el Arduino Uno y se encarga de subirla a la nube, de igual manera de atender y realizar las peticiones de la interfaz de usuario. En el gráfico 3-4, se observa el diagrama de flujo comunicación serial y almacenamiento de datos del nodo Gateway.

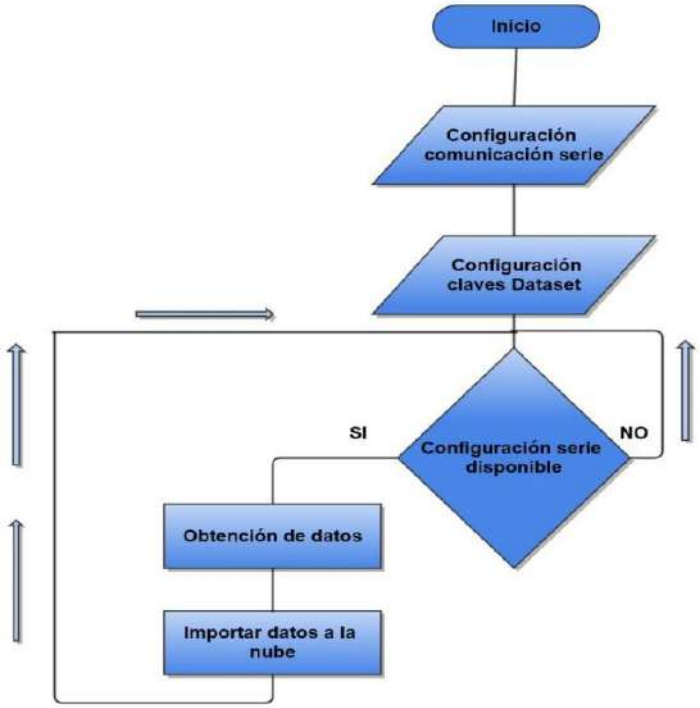


Gráfico 3-4. Diagrama de flujo comunicación serial y almacenamiento de datos del nodo Gateway.

Realizado por: Acero O, 2023.

Las principales librerías utilizadas en la programación del prototipo son:

- **import serial:** Es una sintaxis para la comunicación serial entre el Arduino nano y la tarjeta Raspberry Pi.
- **import firebase_admin:** Es una sintaxis para la comunicación con la base de datos, en este caso firebase el cual se encuentra en la nube y está trabajando en tiempo real.

4.2.1.3 Interfaz de usuario

En este apartado se menciona los métodos y permisos más importantes en la programación de la aplicación móvil.

- **Implementation:** Es una dependencia que se usa para comunicar la aplicación móvil con la base de datos que se encuentra en la nube (ANDROID DEVELOPERS, 2023). Una parte importante de la interfaz es el área delimitada donde el dispositivo debe permanecer, en el caso de que el dispositivo abandone la zona restringida se desplegará un mensaje de alerta.

- **nMap.addPolyline():** Con este método, Android Studio puede dibujar líneas, de esta forma unir todos los puntos y darle forma al lindero como se muestra en la figura 5-4, (ANDROID DEVELOPERS, 2023).

Figura 5-4. Representación Lindero virtual.

Realizado por: Acero O, 2023.

Una vez definido el lindero dentro de la interfaz de usuario, se procede a delimitar el área de pastoreo para el dispositivo nodo recolector ubicado en el ganado, para este proceso se considera un punto de referencia más cercano al centro de la zona de pastoreo, trazando líneas perpendiculares a los dispositivos nodos recolectores ubicados dentro y fuera de la zona delimitada como se representa en la figura 6-4.

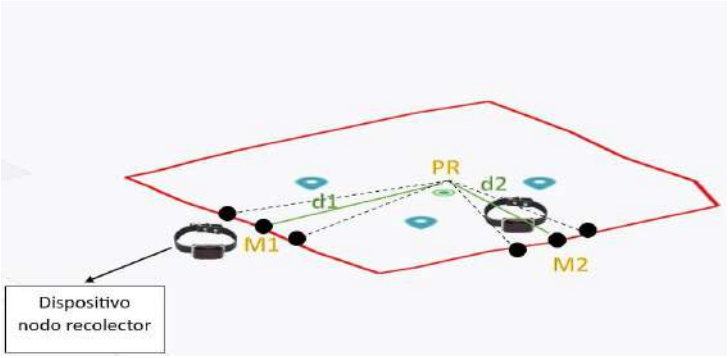


Figura 6-4. Geometría para delimitar el lindero virtual.

Realizado por: Acero O, 2023.

Tomando así los puntos más cercanos para calcular los puntos medios correspondientes **M1 y M2** de cada nodo, usando la ecuación 1-4 y ecuación 2-4. Tomando la ecuación 3-4 se obtiene dos distancias **d1 y d2**, las cuales se comparan para determinar la ubicación del dispositivo de prueba. Entonces si **d1>d2** el dispositivo se encuentra fuera del área de pastoreo, por el contrario, si **d1<d2** el dispositivo se encuentra dentro del área de pastoreo.

Las coordenadas del punto medio de un segmento se calculan en base a dos puntos conocidos, en este caso coordenadas de latitud y longitud (LEHMANN, 1989 págs. 11-12).

$$x = \frac{x_1 + x_2}{2} \quad \textbf{\textit{Ecuación 1-4}}$$

$$y = \frac{y_1 + y_2}{2} \quad \textbf{\textit{Ecuación 2-4}}$$

De acuerdo con la fórmula de la distancia entre dos puntos, se puede calcular las distancias **d1 y d2** en base a los puntos de latitud y longitud (LEHMANN, 1989 págs. 11-12).

$$\boldsymbol{d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}} \quad \textbf{\textit{Ecuación 3-4}}$$

Finalmente se debe considerar los permisos necesarios para acceder a la localización del terreno de estudio, acceder al internet y a las notificaciones que se ejecuten en la aplicación móvil (ANDROID DEVELOPERS, 2023). A continuación, los permisos necesarios:

<uses-permission android:name= “android.permission.INTERNET”/>
<uses-permission android:name= “android.permission.ACCESS_COARSE_LOCATION”/>
<uses-permission android:name= “android.permission.ACCESS_FINE_LOCATION”/>
<uses-permission android:name= “android.permission.POST_NOTIFICATIONS”/>

CAPÍTULO V

5 VALIDACIÓN DEL PROTOTIPO

Para demostrar la validación del prototipo se realizaron algunas pruebas como: la precisión de longitud y latitud del GPS LC86L implementado en el prototipo, consumo de corriente del nodo recolector, latencia del prototipo, pruebas de autonomía y tiempo de carga de las baterías.

5.1 Descripción del lugar de aplicación

El prototipo de monitoreo mediante geolocalización está concebido para instalarse en un terreno ubicado en una Comunidad del Cantón El Tambo, en la Comunidad Chuichun. El solar se encuentra ubicado en las coordenadas 2°30’16’’S, 78°57’8’’W. La parcela cuenta con un servicio de internet 4G. En la figura 1-5, se muestran las medidas que delimitan el terreno.

Figura 1-5. Dimensiones aproximadas del terrero objeto de estudio.

Realizado por: Acero O, 2023.

El terreno delimitado tiene un área aproximada de 3511.795 m2 y un perímetro de 245.609m. Estas medidas también son proporcionadas por la herramienta de medición de Google Earth. En la Figura 2-5, se observa la vivienda en la que se encuentra el modem para el acceso a internet.

Figura 2-5. Vivienda con el acceso a internet para subir los datos a la nube.

Realizado por: Acero O, 2023.

En la figura 3-5, se observa el potrero en el cual se encuentran los bovinos, el terreno presenta poca asimetría en todo su perímetro, lo cual es favorable para la visualización y las pruebas posteriores.

Figura 3-5. Zona delimitada para las pruebas de campo.

Realizado por: Acero O, 2023.

En la figura 4-5, se muestra el ganado a las que se les hará el monitoreo en tiempo real.

Figura 4-5. Ganado bovino para las pruebas de monitoreo.

Realizado por: Acero O, 2023.

5.2 Pruebas del prototipo Iot

Posteriormente al realizar el diseño, construcción y programación del prototipo se procede a verificar su desempeño, se analizan diferentes variables, como la precisión del geoposicionamiento y el rango de cobertura del dispositivo nodo recolector. Las pruebas se realizan en la comunidad de Chuichun en las cercanías del Cantón El Tambo.

5.2.1 Pruebas de Latitud y Longitud del Prototipo

El primer criterio por considerar es la precisión de los datos obtenidos del GPS LC86L implementado en el Prototipo, para esta prueba se tomaron 30 muestras en diferentes puntos, mediante un dispositivo inteligente (Redmi 9 con APP de localización GPS) y el prototipo IoT, en cada punto se obtuvo el valor de referencia de latitud y longitud.

Figura 5-5. Punto de muestreo en la aplicación GPS.

Realizado por: Acero O, 2023.

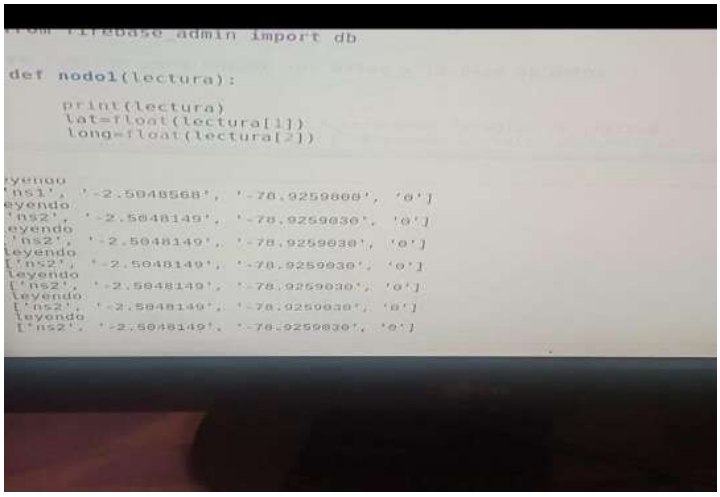


Figura 6-5. Punto de muestreo en el monitor del Prototipo IoT

Realizado por: Acero O, 2023.

A partir de la tabla 1-5 se indica los datos obtenidos en latitud y longitud con el dispositivo inteligente y el prototipo IoT.

Tabla 1-5: Resultado de las pruebas de Latitud y Longitud del Prototipo.

Muestras	Prototipo IoT		Dispositivo inteligente (APP GPS)	
	Latitud	**Longitud**	**Latitud**	**Longitud**
1	-2,5096471	-78,9281271	-2,509497	-78,927963
2	-2,5094298	-78,9279528	-2,509445	-78,927873
3	-2,5094730	-78,9278449	-2,509380	-78,927822
4	-2,5092530	-78,9279961	-2,509392	-78,927943
5	-2,5092630	-78,9279974	-2,509220	-78,927987
6	-2,5091173	-78,9280797	-2,509105	-78,928077
7	-2,5088882	-78,9279320	-2,508993	-78,927942
8	-2,5091173	-78,9280797	-2,508993	-78,928038
9	-2,5090312	-78,9281024	-2,508923	-78,928183
10	-2,5084727	-78,9279235	-2,508795	-78,928120
11	-2,5091173	-78,9280797	-2,508790	-78,928142
12	-2,5097400	-78,9281034	-2,509725	-78,928168
13	-2,5096562	-78,9281842	-2,509663	-78,928220
14	-2,5096958	-78,9280484	-2,509752	-78,928112
15	-2,5094978	-78,9281211	-2,509643	-78,928077
16	-2,5096471	-78,9281271	-2,509657	-78,928038
17	-2,5094801	-78,9278895	-2,509630	-78,927993
18	-2,5096185	-78,9279857	-2,509527	-78,927987
19	-2,5095892	-78,9280155	-2,509512	-78,927940
20	-2,5096385	-78,9276405	-2,509442	-78,927823
21	-2.5096650	-78.9279490	-2.509560	-78.928032
22	-2.5095360	-78.9279590	-2.509483	-78.928012
23	-2.5094730	-78.9279410	-2.509522	-78.927897
24	-2.5095460	-78.9277830	-2.509535	-78.927798
25	-2.5094760	-78.9277530	-2.509500	-78.927755
26	-2.5094220	-78.9277810	-2.509407	-78.927782
27	-2.5093670	-78.9278730	-2.509373	-78.927870
28	-2.5093770	-78.9278580	-2.509392	-78.927792
29	-2.5093590	-78.9278580	-2.509238	-78.927872
30	-2.5093580	-78.9280050	-2.509368	-78.928017

Realizado por: Acero O, 2023.

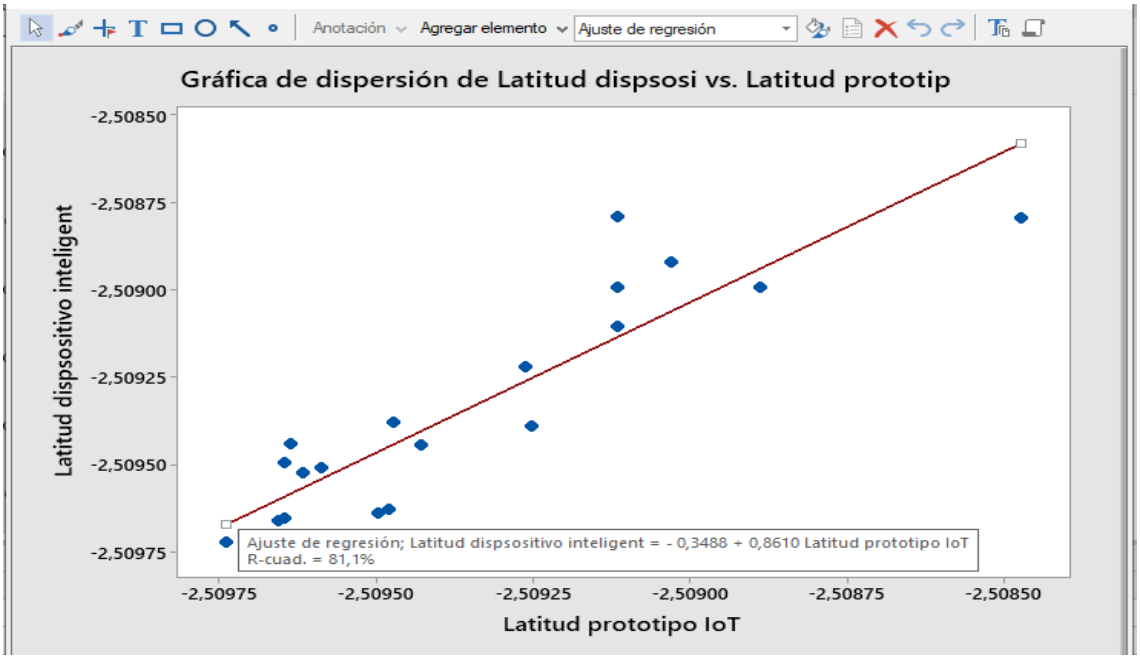
En la tabla 2-5, se puede observar que el error relativo promedio porcentual para la latitud es de 1 y para la longitud 1. Según el autor cumplen con un criterio de validación “muy bueno”, ya que se mantiene debajo del rango del 1% (SANTO, 2005).

Tabla 2-5: Error absoluto y Error relativo en las pruebas de Latitud y Longitud del Prototipo.

Muestra	Error Absoluto		Error relativo	
	Latitud	**Longitud**	**Latitud**	**Longitud**
1	0.00015010	0.0001641	1.37221739727	2.37121595260
2	0.00001520	0.0000798	0.13895872377	1.15309587458
3	0.00009300	0.0000229	0.85020798098	0.33090094646
4	0.00013900	0.0000531	1.27074096083	0.76728560075
5	0.00004300	0.0000104	0.39310691594	0.15027815909
6	0.00001230	0.0000027	0.11244686200	0.03901452207
7	0.00010480	0.0000100	0.95808383234	0.14449822990
8	0.00012430	0.0000417	1.13635324770	0.60255761867
9	0.00010820	0.0000806	0.98916670476	1.16465573297
10	0.00032230	0.0001965	2.94647346528	2.83939021747
11	0.00032730	0.0000623	2.99218357179	0.90022397226
12	0.00001500	0.0000646	0.13713031951	0.93345856513
13	0.00000680	0.0000358	0.06216574485	0.51730366303
14	0.00005620	0.0000636	0.51378159711	0.91900874214
15	0.00014520	0.0000441	1.32742149289	0.63723719384
16	0.00000990	0.0000891	0.09050601088	1.28747922838
17	0.00014990	0.0001035	1.37038899301	1.49555667943
18	0.00009150	0.0000013	0.83649494903	0.01878476989
19	0.00007720	0.0000755	0.70576404443	1.09096163572
20	0.0001965	0.0001825	1.79640718563	2.63709269561
21	0.00013900	0.0000531	1.27074096083	0.76728560075
22	0.00010480	0.0000100	0.95808383234	0.14449822990
23	0.00010820	0.0000806	0.98916670476	1.16465573297
24	0.0001965	0.0000531	1.27074096083	1.09096163572
25	0.00009150	0.0000013	0.83649494903	0.01878476989
26	0.00001230	0.0000027	0.11244686200	0.03901452207
27	0.00014520	0.0000441	1.32742149289	0.63723719384
28	0.00004300	0.0000104	0.39310691594	0.15027815909
29	0.00010820	0.0000806	0.98916670476	1.16465573297
30	0.00000680	0.0000358	0.06216574485	0.51730366303
Media	0.0001093	0.000069	1	1
Desviación Estándar	0.0000684601	0.0000854601	0.756309211	1.171414227

Realizado por: Acero O, 2023.

Para una comparación de la relación lineal existente entre los datos obtenidos del dispositivo inteligente y el prototipo IoT se utiliza la herramienta estadística “Minitab”. En el gráfico 1-5, se muestra la representación en dispersión de la Latitud, con un ajuste de regresión del 81.1 % que es considerado aceptable.



En el gráfico 2-5, se muestra la representación en dispersión de la Longitud en base a los datos obtenidos en la tabla, con un ajuste de regresión del 52.3 % que es considerado aceptable.

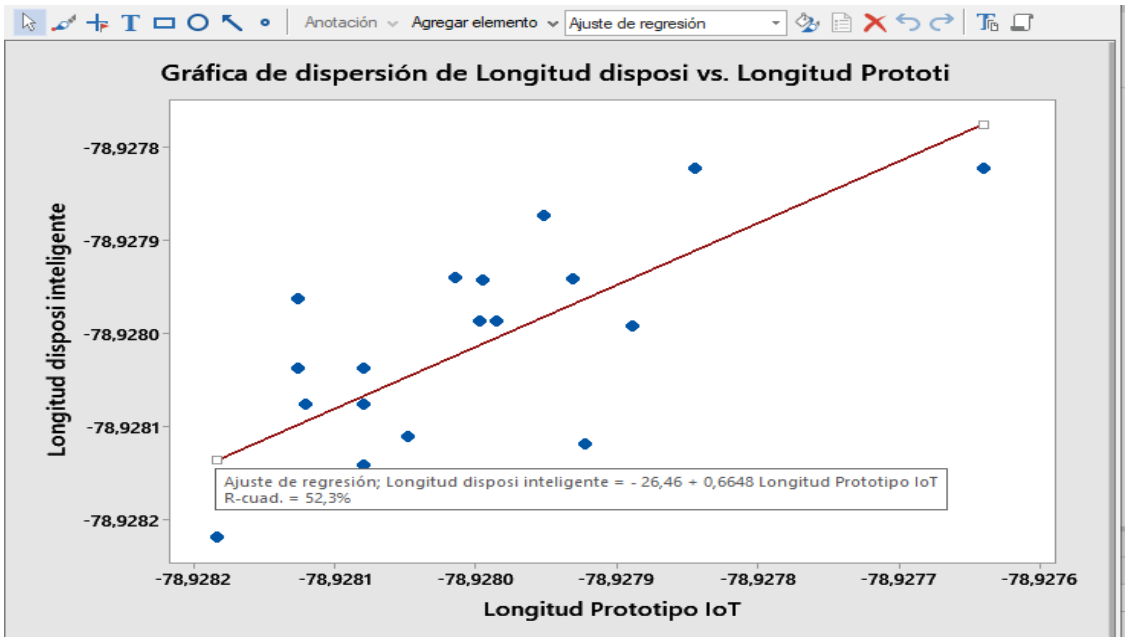


Gráfico 2-5. Datos estadísticos de Longitud del Prototipo.

Realizado por: Acero O, 2023.

5.2.2 Pruebas de latencia

En este apartado se analizó el tiempo de respuesta que presenta el nodo Gateway y la aplicación móvil en recibir la información a partir del nodo recolector. Asegurando que la información llegue

de manera efectiva al nodo Gateway y a la vez a la base de datos en la nube. En la figura 7-5, se muestra la recepción de datos en el entorno del sistema operativo Rasberry Pi.

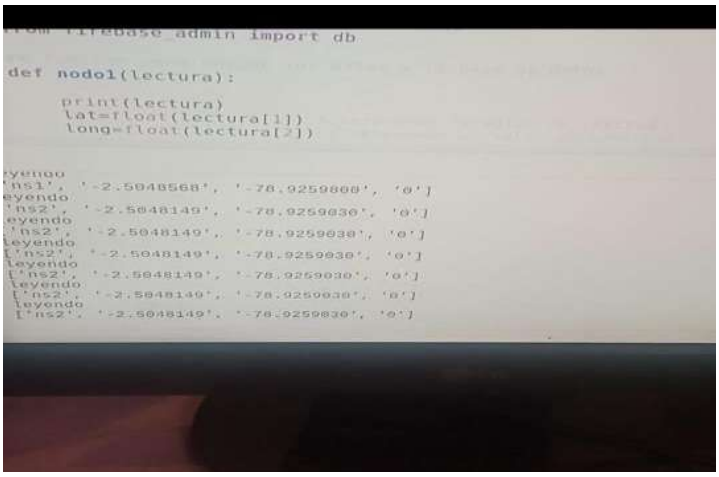


Figura 7-5. Datos obtenidos desde el nodo recolector

Realizado por: Acero O, 2023.

La recepción de datos en el nodo Gateway es en tiempo real, los datos se pueden visualizar mediante el monitor en el entorno de la tarjeta Rasberry Pi y los datos se actualizan continuamente en la base de datos alojada en la nube, como se muestra en la figura 8-5.

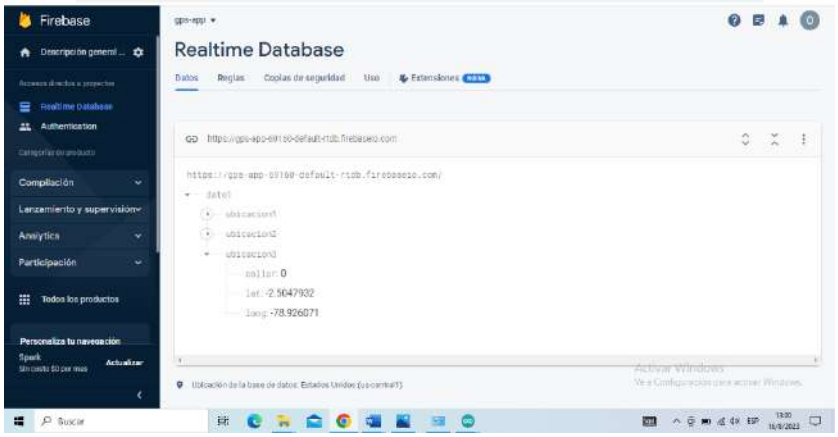


Figura 8-5. Datos obtenidos en la base de datos del dispositivo IoT.

Realizado por: Acero O, 2023.

El tiempo de respuesta confirma una transferencia de información exitosa desde el nodo recolector hasta el nodo Gateway. Para este análisis se obtiene la tabla 3-5 con 30 muestras recogidas en diferentes distancias.

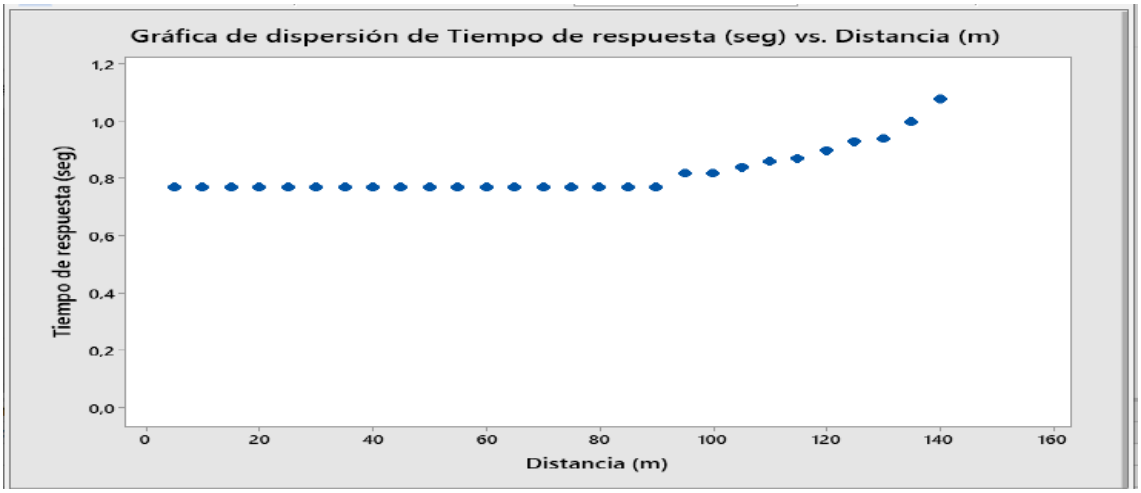
Tabla 3-5: Resultado de las pruebas de Latencia en el Prototipo.

Muestra	Distancia (metros) Nodo recolector -Nodo Gateway	Tiempo máximo de respuesta (segundos)
1	5	0.77
2	10	0.77
3	15	0.77
4	20	0.77
5	25	0.77
6	30	0.77
7	35	0.77
8	40	0.77
9	45	0.77
10	50	0.77
11	55	0.77
12	60	0.77
13	65	0.77
14	70	0.77
15	75	0.77
16	80	0.77
17	85	0.77
18	90	0.77
19	95	0.82
20	100	0.82
21	105	0.84
22	110	0.86
23	115	0.87
24	120	0.90
25	125	0.93
26	130	0.94
27	135	1.00
28	140	1.08
29	145	No trasmite
30	150	No trasmite
Promedio		**0.8**

Realizado por: Acero O, 2023.

A partir de los datos obtenidos de las pruebas realizadas, el tiempo de respuesta desde los 5 hasta 135 metros de distancia es similar con una mínima variación, sin embargo, pasado los 140 metros ya no existe comunicación entre los nodos. En el grafico 3-5, se puede observar que el nodo Gateway cuenta con tiempo máximo de respuesta promedio de 0.8 segundos.

Gráfico 3-5. Gráfica de dispersión para el tiempo de respuesta.



Realizado por: Acero O, 2023.

5.2.3 Prueba de consumo de corriente del nodo recolector

Para determinar el consumo de corriente que requiere el nodo recolector, es necesario conocer la corriente de todos los componentes que lo conforman, por lo tanto, en la tabla 4-5, se muestra la corriente de consumo de cada elemento del nodo.

Tabla 4-5: Recopilación de las corrientes de consumo de los componentes incluidos en el nodo recolector.

Componentes	**Corriente de consumo**
Módulo LC86LICMD	33 mA
Módulo Transceptor LoRa SX1278	Transmisión: 93 mA
Microcontrolador ATmega328p	3.2 mA
Consumo Total del nodo recolector	**129.20 mA**

Realizado por: Acero O, 2023.

5.2.4 Autonomía de las baterías

En base a una prueba experimental, conociendo la corriente de consumo del nodo recolector y la capacidad de la batería, se obtuvo una autonomía estimada de 2 días de funcionamiento del dispositivo, mediante la siguiente ecuación (METAYE, 2022).

$$T = \frac{C}{X} \quad \textit{Ecuación 1-5}$$

C= Capacidad de la batería.

X=Corriente de consumo del dispositivo (nodo recolector).

T= Tiempo de funcionamiento del dispositivo (nodo recolector) (T=horas).

$$T = \frac{2500\ \text{mAh}}{129.20\ \text{mA}}$$
$$T = 19.35\ \text{horas}$$

Teniendo en consideración el uso de dos baterías para la alimentación del nodo recolector, la autonomía total se estima a 38.7 horas. En la figura 9-5, se muestra la evidencia de la autonomía del batería conectado al nodo recolector.

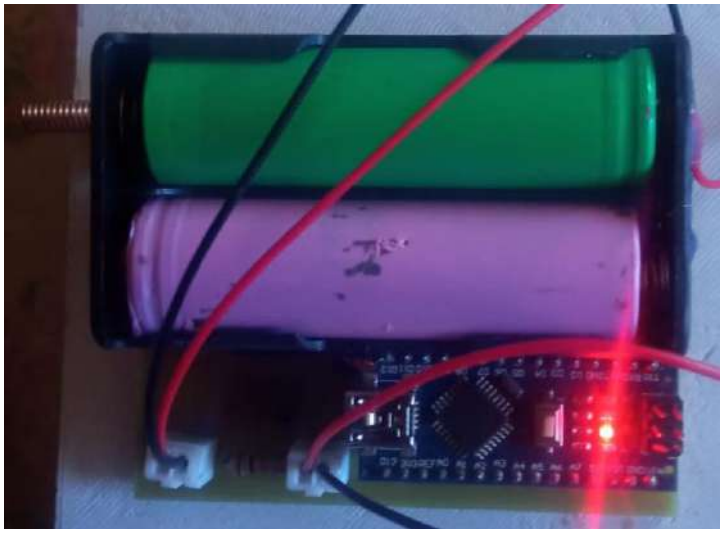


Figura 9-5. Evidencia de la autonomía del batería conectado al nodo recolector.

Realizado por: Acero O, 2023.

5.2.5 Tiempo de carga de las baterías

Para cargar la batería se utiliza una fuente de alimentación que suministra 1000 mA, mediante una regla de tres simples inversas se procede a calcular el tiempo de carga en base a la capacidad de la batería.

$$\textbf{\textit{Capacidad de la bateria}}: \quad 2500\ mA \quad \rightarrow \quad 1\ hora$$
$$\textbf{\textit{Fuente de suminsitro}}: \quad xxxxx\ mA \quad \rightarrow \quad \textbf{\textit{t hora}}$$

$$\boldsymbol{t = \frac{Capacidad\ de\ la\ batería\ (mA) * 1\ hora}{Fuente\ de\ suministro\ (mA)}} \quad \textbf{\textit{Ecuación2-5}}$$

$$t = \frac{2500\ mA * 1\ hora}{1000\ (mA)}$$
$$t = 2.5\ horas$$

5.2.6 Integridad de comunicación entre los nodos

Para la verificación de la integridad de los datos entre los nodos se realizó a una distancia de 30 m, durante un tiempo de 3 horas con intervalos de 0.8 segundos, los cuales son mostrados en el entorno de la tarjeta de desarrollo Rasberry Pi 3 B+, como se muestra en la figura 10-5.

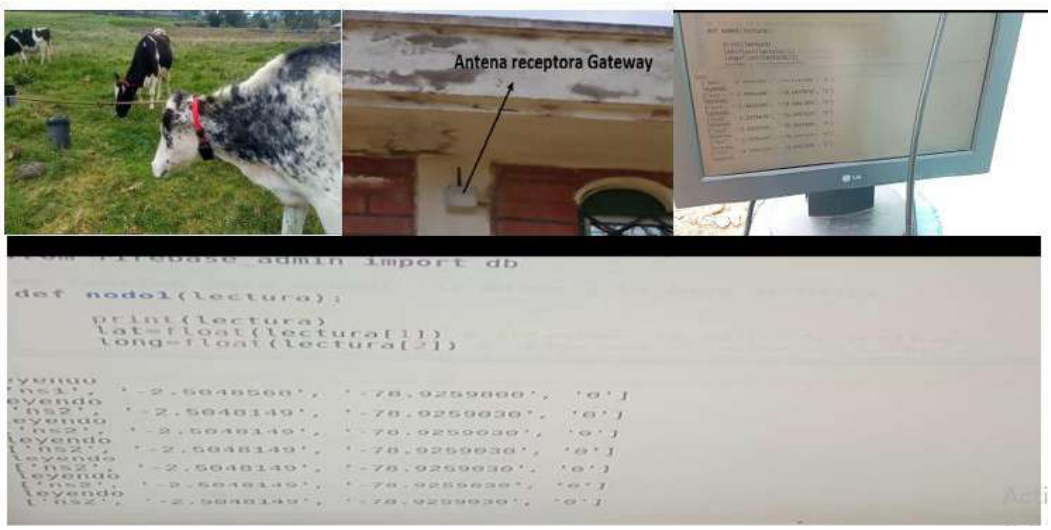


Figura 10-5. Integridad de comunicación entre el nodo Recolector y el nodo Gateway.

Realizado por: Acero O, 2023.

5.3 Aplicación en Campo del Prototipo

En esta prueba se verifica el funcionamiento del Prototipo IoT en tiempo real colocando el dispositivo en el animal a prueba, donde el bovino presenta dos escenarios. El primero cuando se mantiene dentro del perímetro delimitado y el segundo cuando éste lo abandona. En la figura 11-5, se muestra el dispositivo nodo recolector colocado en el ganado.

Figura 11-5. Dispositivo nodo recolector en el ganado.

Realizado por: Acero O, 2023.

En la figura 12-5, se muestra el ganado de prueba dentro del perímetro de pastoreo.

Figura 12-5. Ganado dentro del área de pastoreo.

Realizado por: Acero O, 2023.

Mediante la aplicación móvil se visualiza la ubicación del ganado en tiempo real dentro de la zona delimitada, como se observa en la figura 13-5.

Figura 13-5. Ganado dentro de la zona delimitada, mediante la aplicación móvil.

Realizado por: Acero O, 2023.

En la figura 14-5, se observa al animal fuera del límite de pastoreo, por tal razón se envía una notificación a la aplicación móvil del ganadero.

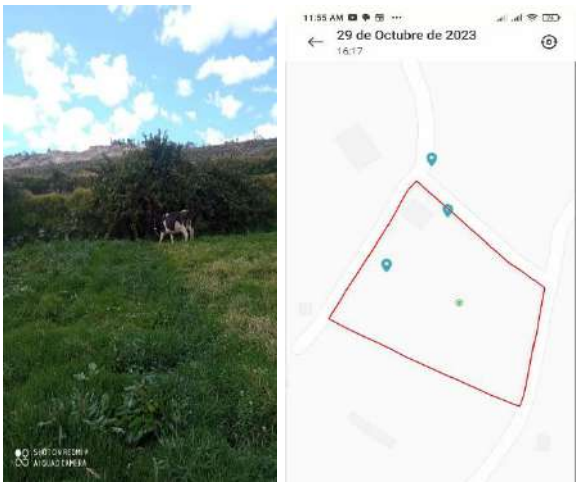


Figura 14-5. Ganado fuera del área de pastoreo.

Realizado por: Acero O, 2023.

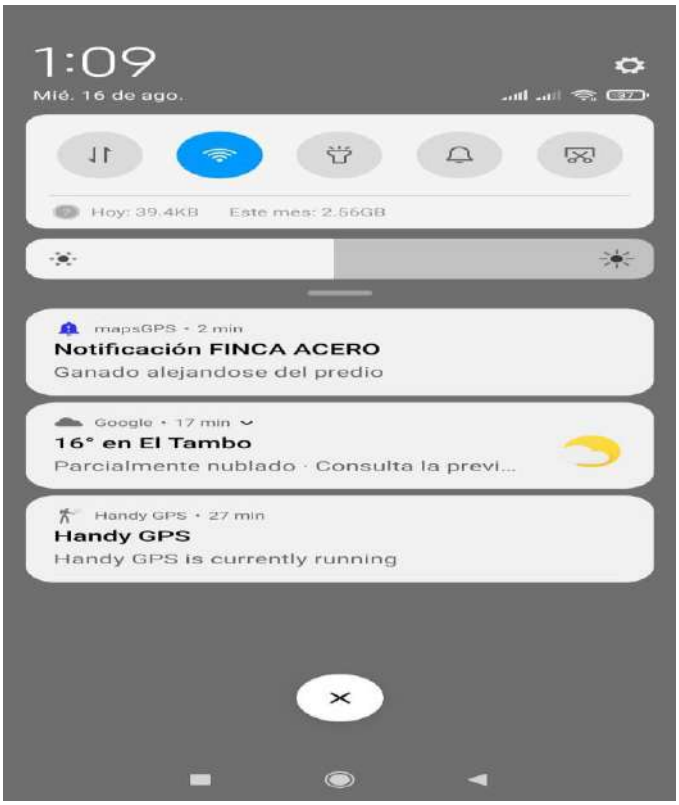


Figura 15-5. Alerta de notificación del ganado, mediante la aplicación móvil.

Realizado por: Acero O, 2023.

De esta forma se verifica el correcto funcionamiento del prototipo IoT, dando una mayor integridad a los bovinos, siendo monitoreados en tiempo real en caso de Abigeato.

5.4 Análisis económico del prototipo IoT.

En la tabla 5-5, se muestra los costos generados en desarrollo e implementación del Prototipo IoT, considerando los requerimientos hardware y software.

Tabla 5-5: Costo de desarrollo e implementación del Prototipo IoT.

Bloque	Descripción	Cantidad	Costo por unidad	Costo total
Tarjetas de desarrollo	Raspberry PI 3 B+	1	$170	$170
	Arduino Nano V3.0	2	$13	$26
	Arduino Uno R3	1	$30	$30
Módulos de comunicación	Módulo GPS LC86L	5	$13	$65
	Módulo Transceptor LoRa SX1278 Ra-02 433 Mhz encapsulado	4	$6	$24
	Módulo Transceptor LoRa SX1278 Ra-02 433 Mhz estándar	1	$15	$15
	Antena de resorte de cobre 433 MHz transmisor de receptor RF	4	$2	$8
	Antena Lora ipx 433MHz Hotspot rp-SMA	2	$3.50	$7
Sistema de alimentación	Batería Samsung de iones de litio 2500 mAh, 3.7 V	6	$10	$60
Software de funcionamiento	Programación del Prototipo	1	$450	$450
Elementos estructurales	Impresión 3D de caja contenedora para nodo recolector y nodo Gateway	2	$5	$10
	Material estructural nodo recolector	2	$10	$20
	Material estructural nodo Gateway	1	$5	$10
	Cinturón	3	$2	$6
Elementos eléctricos - electrónicos	Material Eléctrico y electrónico	1	$70	$70
Otros Gastos	Internet	1	$70	$70
	Mano de obra	1	$80	$80

	Transporte	1	$30	$50
	COSTO TOTAL			$1171

Realizado por: Acero O, 2023.

En Ecuador no existe un mercado con la venta de sistemas IoT o prototipos para monitorear ganado, sin embargo, en el mercado internacional existen prototipos similares destinados al cuidado de ganado tal es el caso de la empresa LPWAN SPAC situado en Shenzhen, China que provee de un collar para ganado basado en la tecnología lorawan a un precio que ronda los $150.

CAPÍTULO VI

6 CONCLUSIONES Y RECOMENDACIONES

En este capítulo son presentadas las conclusiones y recomendaciones en cuanto a este proyecto de titulación se refiere, haciendo énfasis en el diseño, implementación, pruebas y correcciones realizadas en los módulos de geoposición y comunicación, además del hardware y software del prototipo.

6.1 CONCLUSIONES

- Se implementó un prototipo de bajo costo analizando los diferentes elementos, en función a las características de cada dispositivo utilizado en los dos módulos que lo conforman.

- El prototipo dispone de dos nodos el recolector y Gateway, los cuales permiten el monitoreo en tiempo real, mostrando la ubicación actual del ganado mediante una aplicación móvil con tecnología Lora y enviando una notificación de alerta cuando este se aleja del área delimitada de pastoreo o posible caso de abigeato.

- La programación desarrollada en Arduino IDE se realizó mediante funciones o módulos que permiten establecer con claridad los procesos de funcionamiento del prototipo.

- Se incorporó un sistema de comunicación de RF mediante dos módulos Lora, el primero ubicado en el nodo recolector para la trasmisión de datos y el segundo colocado en el nodo Gateway para la recepción. Se incorporó también una tarjeta embebida, la cual se conecta a través internet a la base de datos en la nube y almacena la información recibida, de esta forma permite la comunicación con una aplicación vía WIFI, informándole al personal encargado la ubicación actual del ganado.

- El desarrollo de la aplicación móvil se realizó en Android Studio, donde permite mostrar la ubicación actual del ganado, visualizando sus coordenadas de latitud y longitud cuando el ganado abandona la zona delimitada y se envía una notificación de alerta al dueño del ganado.

- De las pruebas de geoposición realizadas en campo, se logró determinar que el prototipo presenta un error del 1% que es considerado aceptable en cuanto a las pruebas experimentales. De las pruebas de latencia el tiempo de respuesta presenta un promedio de 0.8 segundos, es decir la transferencia de información es inmediata entre los nodos. La autonomía de la batería se mantuvo en funcionamiento por 38.7 horas sin presentar inconvenientes y con un tiempo de carga de 2.5 horas.

6.2 RECOMENDACIONES

- Para usos de tiempo prolongados en el prototipo se recomienda usar componentes con menos consumo energético como celdas fotovoltaicas.

- Mejoramiento de la estructura física del prototipo para soportar las condiciones climáticas y evitar daños a los diferentes módulos.

- Para una comunicación mayores a 140 m en RF, desde un nodo hasta el punto base se debe utilizar un módulo de comunicación con mejores prestaciones y alcance con nuevas tecnologías de comunicación.

- Para incrementar la línea de vista se recomienda colocar la antena del nodo receptor en un punto más alto mediante una estructura metálica.

- Hacer un control periódico preventivo a los dispositivos del prototipo para un mejor funcionamiento del mismo.

GLOSARIO

Prototipo: Es un ejemplar temprano o previo de un producto que se pretende visualizar, probar y multiplicar.

IoT (Internet of Things): Son las siglas en ingles de internet de las cosas.

Sistema de dispositivos interconectados que transfieren e intercambian datos.

Geolocalización: Es la ubicación geográfica real de un objeto.

Geoposición: Permite ubicar una persona o cosa, sobre la superficie terrestre usando coordenadas de Latitud y Longitud.

Latitud: Es la distancia medida en grados que existe entre los paralelos con respecto al Ecuador, con un valor de 0 grados hasta 90 grados, hacia el Norte y hacia el Sur.

Longitud: Es la distancia medida que existe entre los meridianos con respecto al meridiano de Greenwich, con un valor de 0 grados hasta 180 grados, hacia el Este y hasta el Oeste.

Tecnología Lora: Es una tecnología de red de área amplia y bajo consumo para IoT.

M2M (Machine-to-Machine): Son las siglas en ingles de máquina a máquina.

Es la tecnología que permite a dos dispositivos inteligentes intercambiar datos sin la intervención humana.

Abigeato: Robo o hurto de ganado.

Hardware: Partes tangibles de un sistema informático o electrónico.

Software: Partes intangibles de un sistema informático o electrónico.

Aplicación Móvil: Desarrollada para ejecutarse en un dispositivo móvil.

Tarjeta de control embebido: Son tarjetas de hardware libre comúnmente usadas en el campo de la electrónica.

MCU: Por sus siglas en inglés hacen referencia a un microcontrolador.
Es una computadora en un chip, con todos los componentes integrados.

GPS(Global Positioning System): Son las siglas en ingles de sistema de posicionamiento global.

Es una red de satélites y dispositivos receptores utilizados para determinar la ubicación de algo en la Tierra.

Satélite: Cualquier cosa que orbita alrededor de un planeta o una estrella.

Protocolo: Son los pasos que utilizan los dispositivos para comunicarse entre sí.

IP(Internet Protocol): Son las siglas en ingles de protocolo de internet.

Dirección IP: Es un identificador único asignado a cada dispositivo conectado a una red informática.

IEEE (Institute of Electrical and Electronics Engineer): Son las siglas en ingles de instituto de ingeniería eléctrica y electrónica.

Es la asociación profesional más grande del mundo que promueve la innovación y la excelencia tecnológica en beneficio de la humanidad.

IDE (integrated development environment): Son las siglas en ingles de instituto de entorno de desarrollo integrado.

Es una combinación de herramientas como editor de texto, depurador y compilador.

Nube en internet: Hace referencia a los servidores que son accesibles a través de internet.

Fierebase: Es un almacén de datos predeterminado en tiempo real que existe en la nube de internet con acceso directo para el usuario.

Dataset: Colección organizada de datos.

PCB(Printed Circuit Board): Son las siglas en ingles de placa de Circuito Impreso.

Son placas delgadas construidas de un material aislante, con una superficie cubierta de metal.

Interfaz: Es la interacción de una persona con un producto digital tales como un televisor, consola de video juegos, celular, sitios web, etc.

Latencia: Es el tiempo que tardan los datos de viajar de un lugar a otro.

BIBLIOGRAFÍA:

BKHEET, S. A; & AGBINYA, J. I. "A Review of Identity Methods of Internet of Things (IOT)". Advances in Internet of Things [en línea], 2021, (Australia) 11(4), pp. 153-174. [Consulta: 3 noviembre 2022]. ISSN 2161-6825. Disponible en: https://www.scirp.org/journal/paperinformation.aspx?paperid=112421.

ALVARADO QUIÑONEZ, Saúl. Uso de la tecnología Java en el Internet Of Things [En línea] (Trabajo de titulación). (Ingeniería) UNIVERSIDAD POLITÉCNICA DE SINALOA, Sinaloa, Mexico. 2017. pp. 20- 21. [Consulta: 2022-12-25]. Disponible en: http://repositorio.upsin.edu.mx/formatos/UsodeJavaenelInternetOfThings3088.pdf.

BERTE, D.-R. "Defining the IoT". Proceedings of the International Conference on Business Excellence [en línea], 2018, (USA) 12(1), pp. 118 - 128. [Consulta: 3 noviembre 2022]. ISSN 2558-9652. Disponible en: https://sciendo.com/it/article/10.2478/picbe-2018-0013?mbstx=isywy.

BATHLA, R.K; et al. "An Innovative Approach of Big Data and Internet of Things Using 5G Network". International Journal of Computer Applications Technology and Research [en línea], 2019, (India) 8(4), pp. 58-64. [Consulta: 28 octubre 2022]. ISSN 2319–7560. Disponible en: https://www.researchgate.net/publication/331360050_An_Innovative_Approach_of_Big_Data_and_Internet_of_Things_Using_5G_Network.

ANDROID DEVELOPERS. Configure the app module [blog]. [Consulta: 1 junio 2023]. Disponible en: https://developer.android.com/build/configure-app-module.

NANO, Arduino. Arduino Nano [blog]. [Consulta: diciembre 18, 2022]. Disponible en: http://www.mantech.co.za/datasheets/products/a000005-6s.pdf.

ARDUINO. Arduino Uno Rev3. Arduino Uno Rev3 [blog]. [Consulta: 18 de diciembre de 2022]. Disponible en: https://www.arduino.cc/en/hardware.

CONTEXTO GANADERO. Modalidades de abigeato que todo ganadero debe conocer [blog]. [Consulta: febrero 22, 2022]. Disponible en: https://www.contextoganadero.com/ganaderia-sostenible/5-modalidades-de-abigeato-que-todo-ganadero-debe-conocer.

EL COMERCIO. Ganaderos reclaman mayor acción contra el abigeato [blog]. [Consulta: 23 febrero 2022]. Disponible en: https://www.elcomercio.com/actualidad/ecuador/ganaderos-reclamos-abigeato-robo-ganado.html.

EL HERALDO. Preocupa abigeato en zona rural [blog]. [Consulta: 1 mayo 2023]. Disponible en: https://www.elheraldo.com.ec/preocupa-abigeato-en-zona-rural/.

EL PRODUCTOR. El robo de ganado tiene en zozobra a ganaderos de la frontera norte [blog]. [Consulta: 2 febrero 2022]. Disponible en: https://elproductor.com/2022/01/ecuador-el-robo-de-ganado-tiene-en-zozobra-a-ganaderos-de-la-frontera-norte/.

GÓMEZ, J. Servicios en red [en línea]. Madrid-España: Editorial Editex 2010. [Consulta: 2 diciembre 2022]. Disponible en: https://books.google.com.ec/books/about/Servicios_en_Red.html?hl=es&id=vhit3ZmGQPsC&redir_esc=y.

GUERRA IRIARTE, G; et al. Reformas jurídicas a la figura de abigeato respecto a la tipificación y penalización de la sustracción de ganado vacuno caballar y lanar [En línea] (Trabajo de titulación). (Ingeniería) Universidad Nacional de Loja, Loja, Ecuador. 2014. pp. 15- 17. [Consulta: 2022-11-9]. Disponible en: https://dspace.unl.edu.ec/jspui/handle/123456789/14864?mbstx=isywy.

ČOLAKOVIĆ, A; & HADŽIALIĆ, M. "Internet of Things (IoT): A review of enabling technologies, challenges, and open research issues". Computer Networks [en línea], 2018, (United States) 144(C), pp. 17-39. [Consulta: 11 noviembre 2022]. ISSN 2161-6825. Disponible en: https://www.sciencedirect.com/science/article/abs/pii/S1389128618305243.

JINDAL, Juhi. Top 5 Best Agriculture Applications In The World - 2023 [blog]. [Consulta: 10 marzo 2023]. Disponible en: https://tractornews.in/articles/top-5-best-agriculture-applications-in-the-world-2023/.

OTIENO, Julius. Cattle rustlers face life imprisonment in new Senate bill. Cattle rustlers face life imprisonment in new Senate bill [blog]. [Consulta: 8 abril 2023]. Disponible en: https://www.the-star.co.ke/news/2023-04-06-cattle-rustlers-face-life-imprisonment-in-new senate-bill/.

LEHMANN, C. H. Geometría Analítica. Mexico: LIMUSA/NORIEGA EDITORES, 1984, pp. 11-12.

MARTINEZ. Comunidades se organizan para evitar el robo de ganado en Guano [blog]. [Consulta: 3 marzo 2022]. Disponible en: https://www.diariolosandes.com.ec/comunidades-se-organizan-para-evitar-el-robo-de-ganado-en-guano/.

MARTINEZ, Valentina. MODALIDADES DE ABIGEATO [blog]. [Consulta: 9 noviembre 2022]. Disponible en: https://bazarganadero.com/modalidades-de-abigeato/.

METAYE, Romain. A Guide To Understanding Battery Capacity [blog]. [Consulta: 15 agosto 2023]. Disponible en: https://climatebiz.com/battery-capacity/.

MODISAR. Modisar Livestock Management & Tracking Solution [blog]. [Consulta: 10 enero 2023]. Disponible en: https://www.modisar.com/.

MOLINA MOLINA, Danilo Paul. Desarrollo de un sistema para geolocalización de ganado bovino en los páramos de Cotopaxi utilizando software y hardware libre [En línea] (Trabajo de titulación). (Ingeniería) Universidad Técnica de Cotopaxi, Latacunga, Ecuador. 2020. pp. 15- 17. [Consulta: 2022-02-8]. Disponible en: https://repositorio.utc.edu.ec/handle/27000/6682.

NOTICIASDELCANAR. Biblián a merced de la delicuencia [blog]. [Consulta: 1 febrero 2022]. Disponible en:
https://www.noticiasdelcanar.com/2022/02/01/biblian-a-merced-de-la-delincuencia/.

QUECTEL. Quectel LC86LICMD [blog]. [Consulta: 1 diciembre 2022]. Disponible en:
https://www.quectel.com/.

ROJER, Patrick. Powering IoT Devices [blog]. [Consulta: 4 marzo 2023]. Disponible en: https://www.iotforall.com/powering-iot-devices?mbstx=isywy.

RASPBERRY PI. Raspberry Pi 3 Modelo B+ [blog]. [Consulta: 10 diciembre 2022]. Disponible en: https://www.raspberrypi.com/products/raspberry-pi-3-model-b-plus/?mbstx=isywy.

SATYAVATHI, D; et al. "Real-Time Hidden Data Transmission Using Lora. SATYAVATHI". International Journal of Advanced Science Computing and Engineering [en línea], 2022,

(http://ijasce.org/index.php/IJASCE/article/view/88/122) 4(2), pp. 130-137. [Consulta: 3 noviembre 2022]. ISSN 2714-7533. Disponible en: http://ijasce.org/index.php/IJASCE/article/view/88/122.

SAMSUNG SDI. INR18650-25R Datasheet – 2,500mAh, 3.6V, Battery – Samsung. INR18650-25R Datasheet – 2,500mAh, 3.6V, Battery – Samsung [blog]. [Consulta: 10 abril 2023]. Disponible en: https://www.datasheetcafe.com/inr18650-25r-datasheet-lithium-battery/.

ROA PRIETO, Kevin Sebastián & ROJAS SCARPETTA, Luis Carlos. SISTEMA ELECTRÓNICO PARA EL MONITOREO DE BOVINOS HEMBRA EN SU CICLO ESTRAL [En línea] (Trabajo de titulación). (Ingeniería) UNIVERSIDAD CATÓLICA DE COLOMBIA, Bogotá, Colombia. 2019. pp. 43-72. [Consulta: 2022-11-29]. Disponible en: https://repository.ucatolica.edu.co/entities/publication/9ff36b3d-7542-498b-b155-bea72ce13f74.

SHENZHEN TAIDA CENTURY TECHNOLOGY CO,LTD. 2013. 433 MHz Antena 3dbi Hotspot rp-SMA [blog]. [Consulta: 18 diciembre 2022]. Disponible en: http://www.taidacentury.com/en/product.asp?ClassID=003.

SANTO, M. El proceso de medición: análisis y comunicación de datos experimentales [en línea]. Río Cuarto - Argentina: Universidad Nacional de Río Cuarto. [Consulta: 8 agosto 2023]. Disponible en: http://biblioteca.usfa.edu.bo/cgi-bin/koha/opac-detail.pl?biblionumber=4436.

TAMAZIN, M. "High Resolution Signal Processing Techniques for Enhancing GPS Receiver Performance". ResearchGate [en línea], 2015, (Canada) 11(4), pp. 1-154. [Consulta: 11 noviembre 2022]. Disponible en: https://www.researchgate.net/publication/304025457_High_Resolution_Signal_Processing_Techniques_for_Enhancing_GPS_Receiver_Performance.

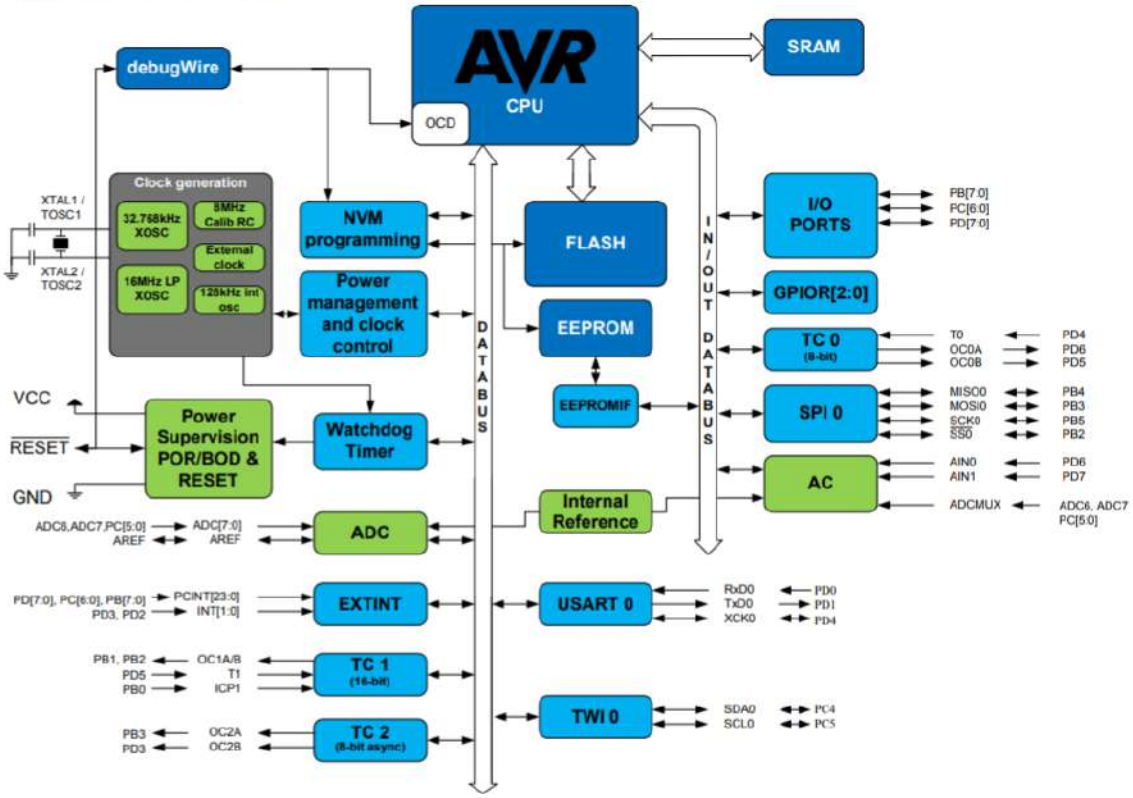
KUMAR , S; & MOORE , K. B. "The Evolution of Global Positioning System (GPS) Technology". Journal of Science Education and Technology [en línea], 2002, (USA) 11(1), pp. 59–80. [Consulta: 8 diciembre 2022]. ISSN 1059-0145. Disponible en: https://link.springer.com/article/10.1023/A:1013999415003.

ANEXOS

ANEXO A: Hoja de datos de las placas Arduino nano V3.0 y Arduino UNO R3.

Block Diagram

Figure 4-1. Block Diagram



14.2. Sleep Modes

The following Table shows the different sleep modes, BOD disable ability and their wake-up sources.

Table 14-1. Active Clock Domains and Wake-up Sources in the Different Sleep Modes.

	Active Clock Domains					Oscillators		Wake-up Sources							Software BOD Disable
Sleep Mode	clkCPU	clkFLASH	clkIO	clkADC	clkASY	Main Clock Source Enabled	Timer Oscillator Enabled	INT and PCINT	TWI Address Match	Timer2	SPM/EEPROM Ready	ADC	WDT	Other I/O	
Idle			Yes	Yes	Yes	Yes	Yes(2)	Yes	Yes	Yes	Yes	Yes	Yes	Yes	
ADC Noise Reduction				Yes	Yes	Yes	Yes(2)	Yes(3)	Yes	Yes(2)	Yes	Yes	Yes		
Power-down								Yes(3)	Yes				Yes		Yes
Power-save					Yes		Yes(2)	Yes(3)	Yes	Yes			Yes		Yes
Standby(1)						Yes		Yes(3)	Yes				Yes		Yes
Extended Standby					Yes(2)	Yes	Yes(2)	Yes(3)	Yes	Yes			Yes		Yes

Note:

1. Only recommended with external crystal or resonator selected as clock source.
2. If Timer/Counter2 is running in asynchronous mode.
3. For INT1 and INT0, only level interrupt.

16. Interrupts

This section describes the specifics of the interrupt handling of the device. For a general explanation of the AVR interrupt handling, refer to the description of *Reset and Interrupt Handling*.

- Each Interrupt Vector occupies two instruction words .
- Reset Vector is affected by the BOOTRST fuse, and the Interrupt Vector start address is affected by the IVSEL bit in MCUCR

16.1. Interrupt Vectors in ATmega328/P

Table 16-1. Reset and Interrupt Vectors in ATmega328/P

Vector No	Program Address(2)	Source	Interrupts definition
1	0x0000(1)	RESET	External Pin, Power-on Reset, Brown-out Reset and Watchdog System Reset
2	0x0002	INT0	External Interrupt Request 0
3	0x0004	INT1	External Interrupt Request 0
4	0x0006	PCINT0	Pin Change Interrupt Request 0
5	0x0008	PCINT1	Pin Change Interrupt Request 1
6	0x000A	PCINT2	Pin Change Interrupt Request 2
7	0x000C	WDT	Watchdog Time-out Interrupt
8	0x000E	TIMER2_COMPA	Timer/Counter2 Compare Match A
9	0x0010	TIMER2_COMPB	Timer/Coutner2 Compare Match B
10	0x0012	TIMER2_OVF	Timer/Counter2 Overflow
11	0x0014	TIMER1_CAPT	Timer/Counter1 Capture Event
12	0x0016	TIMER1_COMPA	Timer/Counter1 Compare Match A
13	0x0018	TIMER1_COMPB	Timer/Coutner1 Compare Match B
14	0x001A	TIMER1_OVF	Timer/Counter1 Overflow
15	0x001C	TIMER0_COMPA	Timer/Counter0 Compare Match A
16	0x001E	TIMER0_COMPB	Timer/Coutner0 Compare Match B
17	0x0020	TIMER0_OVF	Timer/Counter0 Overflow
18	0x0022	SPI STC	SPI Serial Transfer Complete
19	0x0024	USART_RX	USART Rx Complete
20	0x0026	USART_UDRE	USART Data Register Empty
21	0x0028	USART_TX	USART Tx Complete
22	0x002A	ADC	ADC Conversion Complete
23	0x002C	EE READY	EEPROM Ready
24	0x002E	ANALOG COMP	Analog Comparator

ANEXO B: Hoja de datos DEL Módulo Transceptor LoRa SX1278 Ra-02.

1.1. Simplified Block Diagram

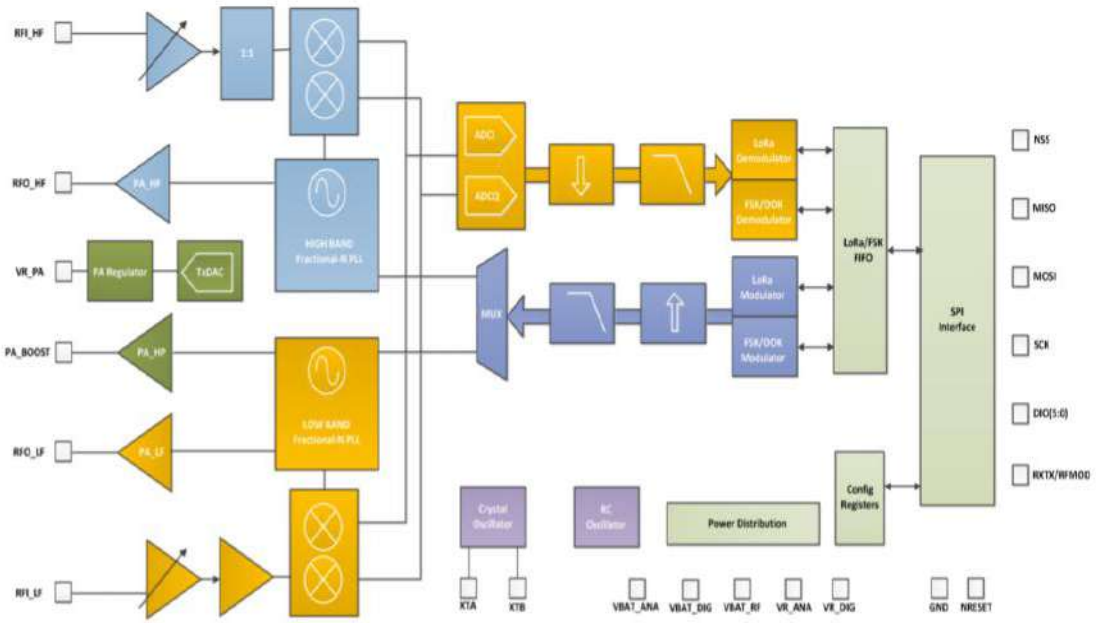


Figure 1. Block Diagram

1.3. Pin Diagram

The following diagram shows the pin arrangement of the QFN package, top view.

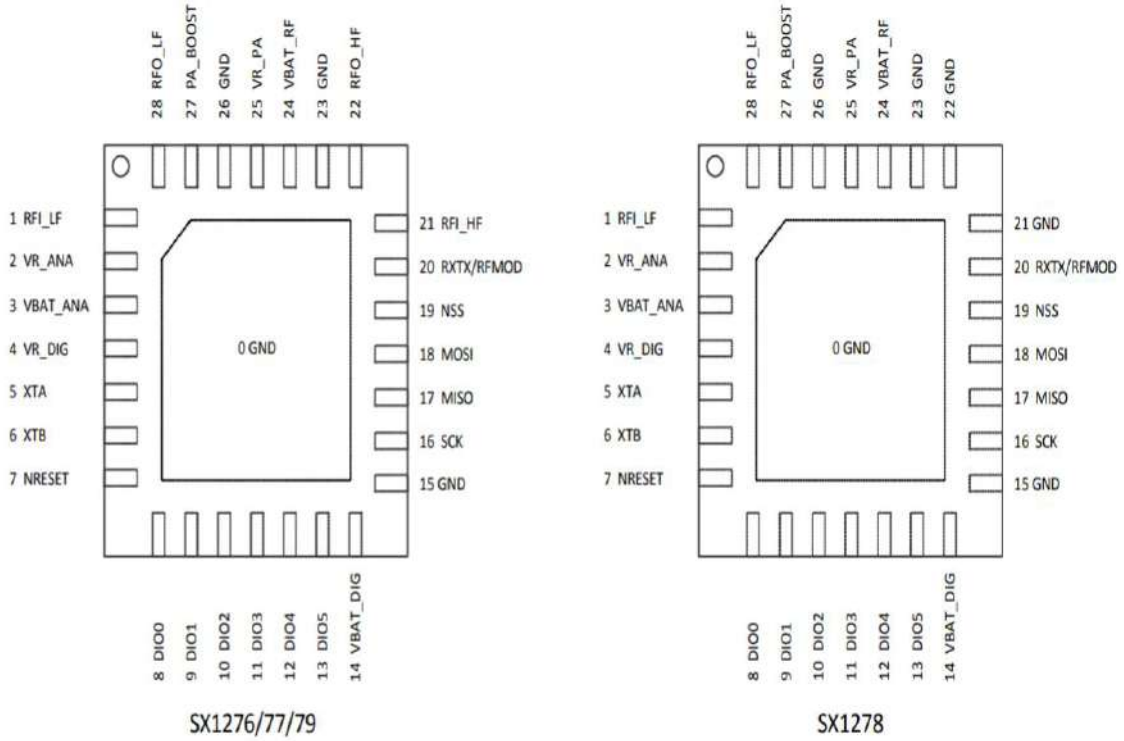
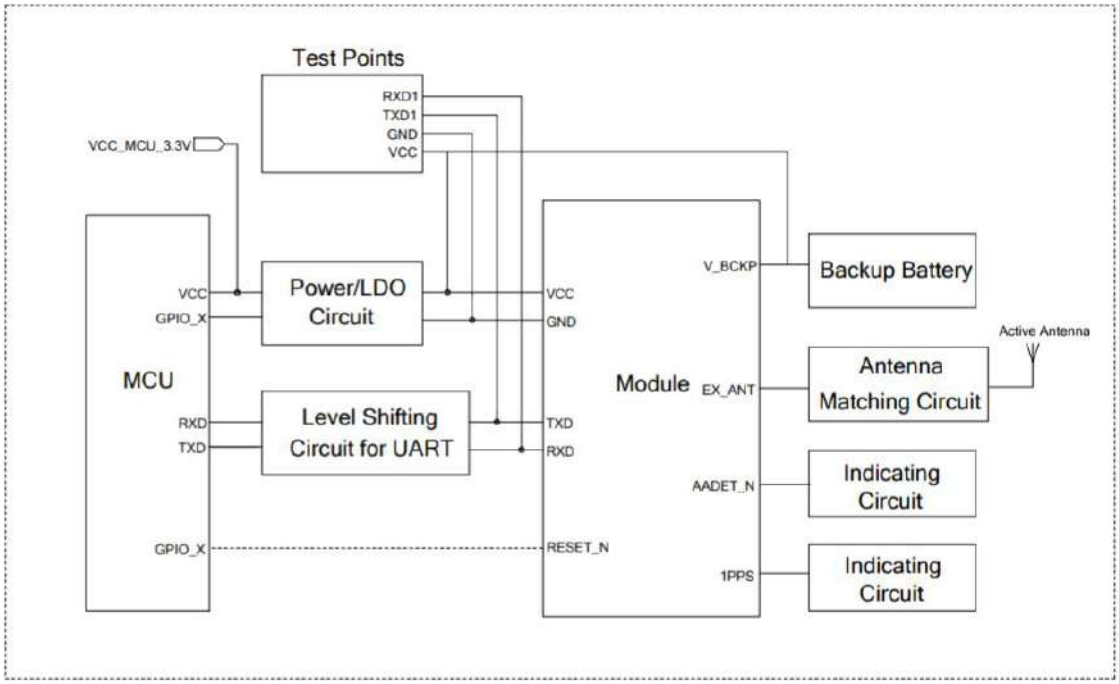


Table 2 Pin Description

Number	Name	Type	Description
	SX1276/77/79/(78)	SX1276/77/79/(78)	SX1276/77/79/(78)
0	GROUND	-	Exposed ground pad
1	RFI_LF	I	RF input for bands 2&3
2	VR_ANA	-	Regulated supply voltage for analogue circuitry
3	VBAT_ANA	-	Supply voltage for analogue circuitry
4	VR_DIG	-	Regulated supply voltage for digital blocks
5	XTA	I/O	XTAL connection or TCXO input
6	XTB	I/O	XTAL connection
7	NRESET	I/O	Reset trigger input
8	DIO0	I/O	Digital I/O, software configured
9	DIO1/DCLK	I/O	Digital I/O, software configured
10	DIO2/DATA	I/O	Digital I/O, software configured
11	DIO3	I/O	Digital I/O, software configured
12	DIO4	I/O	Digital I/O, software configured
13	DIO5	I/O	Digital I/O, software configured
14	VBAT_DIG	-	Supply voltage for digital blocks
15	GND	-	Ground
16	SCK	I	SPI Clock input
17	MISO	O	SPI Data output
18	MOSI	I	SPI Data input
19	NSS	I	SPI Chip select input
20	RXTX/RF_MOD	O	Rx/Tx switch control: high in Tx
21	RFI_HF (GND)	I (-)	RF input for band 1 (Ground)
22	RFO_HF (GND)	O (-)	RF output for band 1 (Ground)
23	GND	-	Ground
24	VBAT_RF	-	Supply voltage for RF blocks
25	VR_PA	-	Regulated supply for the PA
26	GND	-	Ground
27	PA_BOOST	O	Optional high-power PA output, all frequency bands
28	RFO_LF	O	RF output for bands 2&3

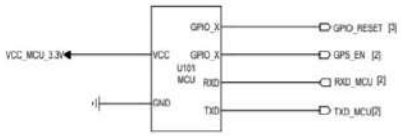
ANEXO C: Hoja de datos del Módulo GPS LC86L.

Block Diagram

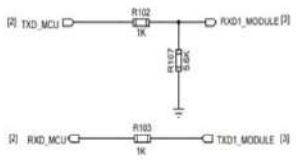


Power Suppply and UART Circuit

Customer's MCU



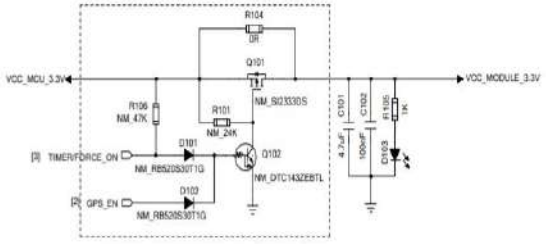
UART Circuit



Note:

The above level shifting circuits realize the voltage-level shifting between VCC_MCU_3.3V and 2.8 V (power domain of UART), and block the leakage current from the power-on devices to power-off devices.

Power Management Circuit (Optional)



Note:

Please ensure the output voltage (VCC_MODULE_3.3V) drop time is greater than 100 ms (from 2.7 to 0.5 V).

	R101	R104	R106	D101	D102	Q101	Q102
L80	24 kΩ	NM	47 kΩ	RB520S30T1G	RB520S30T1G	SI2333DS	DTC143ZEBTL
L86/LC86L	NM	0R	NM	0R	0R	NM	NM

(NM: Not Mounted)

FORCE_ON in L86/LC86L: Logic high of FORCE_ON will force the GNSS module to be woken up from backup mode.

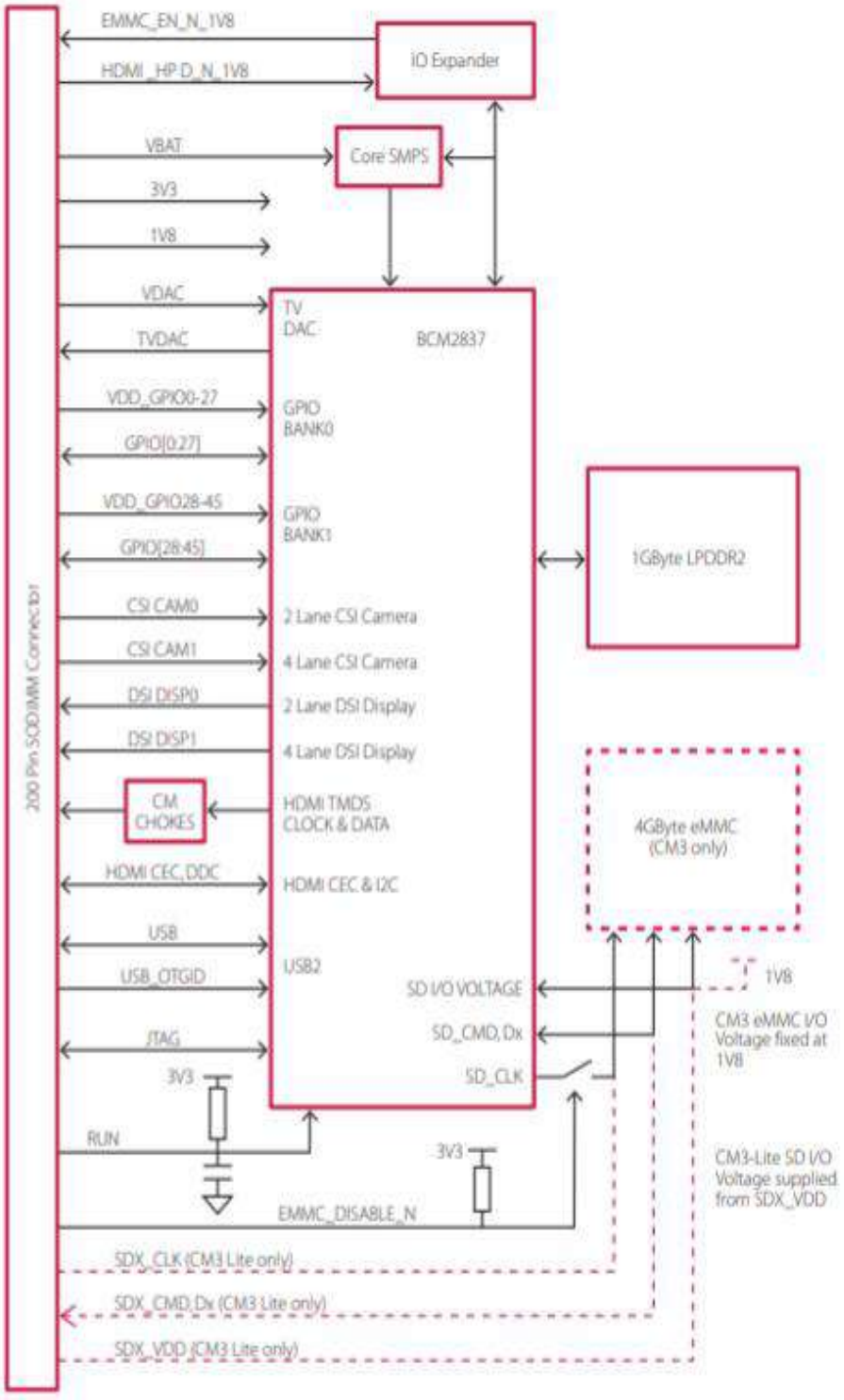
TIMER in L80 module: An open drain output signal used for power on/off control of the GPS module.

When TIMER function is used in L80 module, please ensure V_BCKP is powered all the time.

For more details about TIMER & FORCE_ON, please refer to the corresponding hardware design manuals.

ANEXO D: Hoja de datos del Módulo Raspberry PI 3 B+.

BLOCK DIAGRAM



PIN ASSIGNMENTS

Pin Name	DIR	Voltage Ref	PDN[a] State	If Unused	Description/Notes
RUN and Boot Control (see text for usage guide)					
RUN	I	3V3[b]	Pull High	Leave open	Has internal 10k pull up
EMMC_DISABLE_N	I	3V3[b]	Pull High	Leave open	Has internal 10k pull up
EMMC_EN_N_1V8	O	1V8	Pull High	Leave open	Has internal 2k2 pull up
GPIO					
GPIO[27:0]	I/O	GPIO0-27_VDD	Pull or Hi-Z[c]	Leave open	GPIO Bank 0
GPIO[45:28]	I/O	GPIO28-45_VDD	Pull or Hi-Z[c]	Leave open	GPIO Bank 1
Primary SD Interface[d,r]					
SDX_CLK	O	SDX_VDD	Pull High	Leave open	Primary SD interface CLK
SDX_CMD	I/O	SDX_VDD	Pull High	Leave open	Primary SD interface CMD
SDX_Dx	I/O	SDX_VDD	Pull High	Leave open	Primary SD interface DATA
USB Interface					
USB_Dx	I/O	-	Z	Leave open	Serial interface
USB_OTGID	I	3V3		Tie to GND	OTG pin detect
HDMI Interface					
HDMI_SCL	I/O	3V3[b]	Z[f]	Leave open	DDC Clock (5.5V tolerant)
HDMI_SDA	I/O	3V3[b]	Z[f]	Leave open	DDC Data (5.5V tolerant)
HDMI_CEC	I/O	3V3	Z	Leave open	CEC (has internal 27k pull up)
HDMI_CLKx	O	-	Z	Leave open	HDMI serial clock
HDMI_Dx	O	-	Z	Leave open	HDMI serial data
HDMI_HPD_N_1V8	I	1V8	Pull High	Leave open	HDMI hotplug detect
CAM0 (CSI0) 2-lane Interface					
CAM0_Cx	I	-	Z	Leave open	Serial clock
CAM0_Dx	I	-	Z	Leave open	Serial data
CAM1 (CSI1) 4-lane Interface					
CAM1_Cx	I	-	Z	Leave open	Serial clock
CAM1_Dx	I	-	Z	Leave open	Serial data
DSI0 (Display 0) 2-lane Interface					
DSI0_Cx	O	-	Z	Leave open	Serial clock
DSI0_Dx	O	-	Z	Leave open	Serial data
DSI1 (Display 1) 4-lane Interface					
DSI1_Cx	O	-	Z	Leave open	Serial clock
DSI1_Dx	O	-	Z	Leave open	Serial data
TV Out					
TVDAC	O	-	Z	Leave open	Composite video DAC output
JTAG Interface					
TMS	I	3V3	Z	Leave open	Has internal 50k pull up
TRST_N	I	3V3	Z	Leave open	Has internal 50k pull up
TCK	I	3V3	Z	Leave open	Has internal 50k pull up
TDI	I	3V3	Z	Leave open	Has internal 50k pull up
TDO	O	3V3	O	Leave open	Has internal 50k pull up

ANEXO E: Código de programación del nodo recolector.

```
#include <TinyGPS++.h>
#include <SoftwareSerial.h>

#include <SPI.h>
#include <LoRa.h>

#define rxGPS 4
#define txGPS 3

#define collar 6 // pin cable collar


SoftwareSerial gpsSerial(rxGPS, txGPS);
TinyGPSPlus gps;

// variables para almacenar los valored del GPS
float latitud = 0;
String lati = "";
float longitud = 0;
String lon = "";
String est_collar = "";
// constante identificado del dispocito ns1=>nodo sensor 1
const String codigo_nodo = "ns3/";

void setup()
{
  Serial.begin(9600); // connect serial
  gpsSerial.begin(9600); // connect gps sensor

   pinMode(collar, INPUT);

   while (!Serial);
  Serial.println("LoRa Sender");
  if (!LoRa.begin(433E6)) {
    Serial.println("Starting LoRa failed!");
    while (1);
  }
  LoRa.setTxPower(20); // Configurar la potencia de transmisión a 20 dBm
  LoRa.setSpreadingFactor(9); // Configurar Spreading Factor a 9
  LoRa.setSignalBandwidth(125000); // Configurar el ancho de banda a 125 kHz
  LoRa.setCodingRate4(5); // Configurar el índice de codificación a 4/5
}

void loop()
{

  read_gps();
}
```

```
void read_gps(){

  while (gpsSerial.available())     // check for gps data
  {
    //Serial.println("Intentando leer");
    if (gps.encode(gpsSerial.read()))   // encode gps data
    {

      est_collar= String(digitalRead(collar));

      Serial.print("SATS: ");
      Serial.println(gps.satellites.value());
      Serial.print("LAT: ");
      Serial.println(gps.location.lat(), 6);
      latitud = gps.location.lat(); // almacenamos en una variable el dato de latitud
      lati = String(latitud, 7); // convertimos a cadena de caracteres el dato de latitud

      Serial.print("LONG: ");
      Serial.println(gps.location.lng(), 6);
      longitud = gps.location.lng(); // almacenamos el valor de longitud
      lon = String(longitud, 7); // convertimos a cadena de caracteres el valor de longitud

      String dato = codigo_nodo + lati + "/" + lon+"/"+est_collar; // concatenamos en una variable el codigo del nodo, latitud y longitud
      Serial.println(dato);

      Serial.print("ALT: ");
      Serial.println(gps.altitude.meters());
      Serial.print("SPEED: ");
      Serial.println(gps.speed.mps());
      Serial.print("Hour: ");
      Serial.print(gps.time.hour()); Serial.print(":");
      Serial.print(gps.time.minute()); Serial.print(":");
      Serial.println(gps.time.second());
      Serial.println("----------------------------");

      enviar_lora(dato); // enviamos el dato para ser transmitido por el modulo lora

      delay(2000);
    }
  }
}
// funcion para transmitir los datos por medio de lora
void enviar_lora(String dataToSend) {
  //dataToSend="nsl/100/100/1";
  LoRa.beginPacket();
  LoRa.print(dataToSend);
  LoRa.endPacket(); // <- No se pasan argumentos aquí
  Serial.println("Enviando Paquete " + dataToSend);
  delay(1000); // Puedes ajustar el tiempo entre envíos aquí

}
```

ANEXO F: Código de programación Receptor Lora del nodo Gateway.

```
#include <SPI.h> // comunicacion SPI
#include <LoRa.h> // COMUICACION Lora

String inString = "";

void setup() {
  Serial.begin(9600); // inicimos puerto serie

  while (!Serial); //verificamos comunicacion serial
  Serial.println("LoRa Receiver");
  if (!LoRa.begin(433E6)) { // verificamos comunicacion con el modulo lora
    Serial.println("Starting LoRa failed!");
    while (1);
  }
  //Serial.println("Fin setup");

}

void loop() {

  // intentando capturar un paquete
  int packetSize = LoRa.parsePacket();
  if (packetSize) {
    // received a packet
    //Serial.print("Recibiendo paquete: ");

   // Leyendo paquete
   while (LoRa.available()) { // verificamos paquetes disponibles
     int inChar = LoRa.read(); // leemos el paquete
     inString += (char)inChar; // decodificamos el paquete
   }
   Serial.println(inString); // imprimimos el paquete en el puerto serie
   LoRa.packetRssi();
 }
 inString = "";// vaciamos la variable de almacenamiento
 delay(10);
}
```

ANEXO G: Código de programación almacenamiento en la base de datos Firebase.

```
import serial # libreria comunicacion serial
import firebase_admin # libreria comunicacion con firebase
## importamos dependencias de firebase_admin
from firebase_admin import credentials
from firebase_admin import db

## funcion para enviar los datos a la base de Datos
def nodo1(lectura):

    print(lectura)
    lat=float(lectura[1]) # obtenemos le valor de latitud
    long=float(lectura[2]) # obtenemos el valor de longitud
    estado = int(lectura[3])
    ref = db.reference("dato1") ## hscemos referencia a la identificador de la base de datos
    dato_ref =ref.child("ubicacion1") ## referenciamos el nodo hijo de la base de datos
    dato_ref.update({"lat":lat}) ## actualizamos el valor del objeto
    dato_ref.update({"long":long})## actualizamos el valor del objeto
    dato_ref.update({"collar":estado})## actualizamos el valor del objeto

## funcion para enviar los datos a la base de Datos
def nodo2(lectura):

    print(lectura)
    lat=float(lectura[1]) # obtenemos le valor de latitud
    long=float(lectura[2]) # obtenemos el valor de longitud
    estado = int(lectura[3])
    ref = db.reference("dato1") ## hscemos referencia a la identificador de la base de datos
    dato_ref =ref.child("ubicacion2") ## referenciamos el nodo hijo de la base de datos
    dato_ref.update({"lat":lat}) ## actualizamos el valor del objeto
    dato_ref.update({"long":long})## actualizamos el valor del objeto
    dato_ref.update({"collar":estado})## actualizamos el valor del objeto

## funcion para enviar los datos a la base de Datos
def nodo3(lectura):

    print(lectura)
    lat=float(lectura[1]) # obtenemos le valor de latitud
    long=float(lectura[2]) # obtenemos el valor de longitud
    estado = int(lectura[3])
    ref = db.reference("dato1") ## hscemos referencia a la identificador de la base de datos
    dato_ref =ref.child("ubicacion3") ## referenciamos el nodo hijo de la base de datos
    dato_ref.update({"lat":lat}) ## actualizamos el valor del objeto
    dato_ref.update({"long":long})## actualizamos el valor del objeto
    dato_ref.update({"collar":estado})## actualizamos el valor del objeto

##########################################################
##      Configuraciones

## credenciales de la base de datos
cred = credentials.Certificate("./gps-app-69160-firebase-adminsdk.json")
## direccion de la base de datos
firebase_admin.initialize_app(cred, {
    "databaseURL":"https://gps-app-69160-default-rtdb.firebaseio.com/"})

#ref = db.reference("/nodos1")
#ref.push({"Lat":0, "Long":0})

## iniciamos la comunicacion serial
ser = serial.Serial("COM5",9600)
```

```
#############################################################
##      Bucle infinito
while(True):
    ## leemos el puerto serie
    print("leyendo")
    try:

        lectura = (ser.readline().decode("utf-8")).strip()
        ## dividimos el dato leido utilisando / como separador
        lectura = lectura.split("/")
        ## verificamos la integridad de los datos
        if len(lectura)>=3:
            ## identificaos el nodo sensor
            if lectura[0]=="ns1":
                nodo1(lectura)

            elif lectura[0]=="ns2":
                nodo2(lectura)

            elif lectura[0]=="ns3":
                nodo3(lectura)

            else:
                continue
    except:

        print("Error... Reintentando")
        continue
```

ANEXO H: Código de programación de la aplicación móvil.

```
package com.ejemplo.mapsgps;

import androidx.annotation.NonNull;
import androidx.core.app.NotificationCompat;
import androidx.core.app.NotificationManagerCompat;
import androidx.fragment.app.FragmentActivity;
import android.annotation.SuppressLint;
import android.app.Notification;
import android.app.NotificationChannel;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.graphics.Color;
import android.graphics.drawable.ColorDrawable;
import android.os.Build;
import android.os.Bundle;

import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.BitmapDescriptorFactory;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;
import com.ejemplo.mapsgps.databinding.ActivityMapsBinding;
import com.google.android.gms.maps.model.PolylineOptions;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import java.util.ArrayList;

public class MapsActivity extends FragmentActivity implements OnMapReadyCallback {

    private GoogleMap mMap;
    private ActivityMapsBinding binding;

    //traemos la referencia de la base de datos
    private DatabaseReference mDatabase;
    private ArrayList<Marker> tmpRealTimeMarker = new ArrayList<>();
    private ArrayList<Marker> realTimeMarkers = new ArrayList<>();
    public double lat_ubicacion1;
    public double lon_ubicacion1;
    public double lat_ubicacion2;
    public double lon_ubicacion2;
    public double lat_ubicacion3;
    public double lon_ubicacion3;

    public String collar1;
    public String collar2;
    public String collar3;

    public double puntos_lat[] = new double[70];
    public double puntos_long[] = new double[70];
    public double distancias[] = new double[70];

    private PendingIntent pendingIntent;
    private final static String CHANNEL_ID = "NOTIFICACIÓN";
    private final static int NOTIFICACION_ID = 0;
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    //Colores para los bordes de la pantalla de la aplicación
    String primaryDark="#F3D66E";
    String primary="#F3D66E";
    String background="#e6f3f7";
    cambiarColor(primaryDark, primary,background);

    binding = ActivityMapsBinding.inflate(getLayoutInflater());
    setContentView(binding.getRoot());

    // Obtain the SupportMapFragment and get notified when the map is ready to be used.
    SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
            .findFragmentById(R.id.map);
    mapFragment.getMapAsync(this);

    mDatabase = FirebaseDatabase.getInstance().getReference();
}

/**
 * Manipulates the map once available.
 * This callback is triggered when the map is ready to be used.
 * This is where we can add markers or lines, add listeners or move the camera. In this case,
 * we just add a marker near Sydney, Australia.
 * If Google Play services is not installed on the device, the user will be prompted to install
 * it inside the SupportMapFragment. This method will only be triggered once the user has
 * installed Google Play services and returned to the app.
 */
@SuppressLint("SuspiciousIndentation")
@Override
public void onMapReady(GoogleMap googleMap) {
    mMap = googleMap;


    LatLng punto1 = new LatLng(-2.505387235079453, -78.94986379891634);
    puntos_lat[0] = punto1.latitude;
    puntos_long[0] = punto1.longitude;


    LatLng punto2 = new LatLng(-2.505400298349406, -78.9498758688568);
    puntos_lat[1] = punto2.latitude;
    puntos_long[1] = punto2.longitude;


    LatLng punto3 = new LatLng(-2.5054133616192473, -78.94988793879747);
    puntos_lat[2] = punto3.latitude;
    puntos_long[2] = punto3.longitude;

    LatLng punto4 = new LatLng(-2.505436138602241, -78.94990134984255);
    puntos_lat[3] = punto4.latitude;
    puntos_long[3] = punto4.longitude;

    LatLng punto5 = new LatLng(-2.5054582456735766, -78.94991308450699);
    puntos_lat[4] = punto5.latitude;
    puntos_long[4] = punto5.longitude;

    LatLng punto6 = new LatLng(-2.5054885591568965, -78.94992867484652);
    puntos_lat[5] = punto6.latitude;
    puntos_long[5] = punto6.longitude;
```

```
LatLng punto7 = new LatLng(-2.5055188726400317, -78.94994426518679);
puntos_lat[6] = punto7.latitude;
puntos_long[6] = punto7.longitude;

LatLng punto8 = new LatLng(-2.5055379651094585, -78.94995482638463);
puntos_lat[7] = punto8.latitude;
puntos_long[7] = punto8.longitude;


LatLng punto9 = new LatLng(-2.505557057578801, -78.94996538758278);
puntos_lat[8] = punto9.latitude;
puntos_long[8] = punto9.longitude;


LatLng punto10 = new LatLng(-2.5055835190706266, -78.94998131319848);
puntos_lat[9] = punto10.latitude;
puntos_long[9] = punto10.longitude;

LatLng punto11 = new LatLng(-2.5056099805622605, -78.94999723881483);
puntos_lat[10] = punto11.latitude;
puntos_long[10] = punto11.longitude;


LatLng punto12 = new LatLng(-2.505635102230725, -78.9500124938783);
puntos_lat[11] = punto12.latitude;
puntos_long[11] = punto12.longitude;


LatLng punto13 = new LatLng(-2.505686350433192, -78.9500431716439);
puntos_lat[12] = punto13.latitude;
puntos_long[12] = punto13.longitude;
```

```
puntos_lat[13] = punto14.latitude;
puntos_long[13] = punto14.longitude;


LatLng punto15 = new LatLng(-2.505737431155915, -78.95007451996177);
puntos_lat[14] = punto15.latitude;
puntos_long[14] = punto15.longitude;


LatLng punto21 = new LatLng(-2.5058802896912464, -78.95015850663185);
puntos_lat[20] = punto21.latitude;
puntos_long[20] = punto21.longitude;

LatLng punto22 = new LatLng(-2.5058876587126133, -78.95014543086297);
puntos_lat[21] = punto22.latitude;
puntos_long[21] = punto22.longitude;

LatLng punto23 = new LatLng(-2.5058950277338505, -78.95013235509396);
puntos_lat[22] = punto23.latitude;
puntos_long[22] = punto23.longitude;

LatLng punto24 = new LatLng(-2.5059090958654386, -78.95010720938471);
puntos_lat[23] = punto24.latitude;
puntos_long[23] = punto24.longitude;


LatLng punto25 = new LatLng(-2.5059231639965445, -78.95008206367493);
puntos_lat[24] = punto25.latitude;
puntos_long[24] = punto25.longitude;


LatLng punto26 = new LatLng(-2.5059457734935395, -78.95003562793217);
puntos_lat[25] = punto26.latitude;
puntos_long[25] = punto26.longitude;


LatLng punto27 = new LatLng(-2.5059231639965445, -78.95008206367493);
puntos_lat[26] = punto27.latitude;
puntos_long[26] = punto27.longitude;
```

```
LatLng punto28 = new LatLng(-2.505933212661757, -78.95006194710747);
puntos_lat[27] = punto28.latitude;
puntos_long[27] = punto28.longitude;

LatLng punto29 = new LatLng(-2.50594326132666, -78.9500418305397);
puntos_lat[28] = punto29.latitude;
puntos_long[28] = punto29.longitude;

LatLng punto30 = new LatLng(-2.5059558221580396, -78.950015511364);
puntos_lat[29] = punto30.latitude;
puntos_long[29] = punto30.longitude;

LatLng punto31 = new LatLng(-2.5059683829888915, -78.94998919218779);
puntos_lat[30] = punto31.latitude;
puntos_long[30] = punto31.longitude;

LatLng punto32 = new LatLng(-2.505988145362835, -78.94995130598545);
puntos_lat[31] = punto32.latitude;
puntos_long[31] = punto32.longitude;

LatLng punto33 = new LatLng(-2.5060027159266496, -78.94992565736203);
puntos_lat[32] = punto33.latitude;
puntos_long[32] = punto33.longitude;

LatLng punto34 = new LatLng(-2.506017286489962, -78.94990000873804);
puntos_lat[33] = punto34.latitude;
puntos_long[33] = punto34.longitude;

LatLng punto35 = new LatLng(-2.5060290099319085, -78.94987536594293);
puntos_lat[34] = punto35.latitude;
puntos_long[34] = punto35.longitude;

LatLng punto36 = new LatLng(-2.5060407333733923, -78.94985072314739);
puntos_lat[35] = punto36.latitude;
```

```
LatLng punto44 = new LatLng(-2.5061733757345896, -78.94956976175308);
puntos_lat[43] = punto44.latitude;
puntos_long[43] = punto44.longitude;

LatLng punto45 = new LatLng(-2.506184261786566, -78.94954428076765);
puntos_lat[44] = punto45.latitude;
puntos_long[44] = punto45.longitude;

LatLng punto46 = new LatLng(-2.5061951478380484, -78.9495187997818);
puntos_lat[45] = punto46.latitude;
puntos_long[45] = punto46.longitude;

LatLng punto47 = new LatLng(-2.5061767252888734, -78.9495124295353);
puntos_lat[46] = punto47.latitude;
puntos_long[46] = punto47.longitude;

LatLng punto48 = new LatLng(-2.506158302739668, -78.94950605928898);
puntos_lat[47] = punto48.latitude;
puntos_long[47] = punto48.longitude;

LatLng punto49 = new LatLng(-2.5061383728904256, -78.94950170069926);
puntos_lat[48] = punto49.latitude;
puntos_long[48] = punto49.longitude;

LatLng punto50 = new LatLng(-2.5061184430411685, -78.94949734210968);
puntos_lat[49] = punto50.latitude;
puntos_long[49] = punto50.longitude;

LatLng punto51 = new LatLng(-2.5060762386525433, -78.94948979839658);
puntos_lat[50] = punto51.latitude;
puntos_long[50] = punto51.longitude;

LatLng punto58 = new LatLng(-2.505887658712574, -78.94945543259382);
puntos_lat[57] = punto58.latitude;
puntos_long[57] = punto58.longitude;

LatLng punto59 = new LatLng(-2.505858852537774, -78.94945040345182);
puntos_lat[58] = punto59.latitude;
puntos_long[58] = punto59.longitude;

LatLng punto60 = new LatLng(-2.505830046362955, -78.94944537431003);
puntos_lat[59] = punto60.latitude;
puntos_long[59] = punto60.longitude;

LatLng punto61 = new LatLng(-2.5057997328869326, -78.94944000989186);
puntos_lat[60] = punto61.latitude;
puntos_long[60] = punto61.longitude;

LatLng punto62 = new LatLng(-2.505769419410888, -78.94943464547396);
puntos_lat[61] = punto62.latitude;
puntos_long[61] = punto62.longitude;

LatLng punto63 = new LatLng(-2.5057561886668775, -78.9494517445566);
puntos_lat[62] = punto63.latitude;
puntos_long[62] = punto63.longitude;

LatLng punto64 = new LatLng(-2.505742957922644, -78.94946884436389);
puntos_lat[63] = punto64.latitude;
puntos_long[63] = punto64.longitude;

LatLng punto65 = new LatLng(-2.5057255402339726, -78.94949147477776);
puntos_lat[64] = punto65.latitude;
puntos_long[64] = punto65.longitude;

LatLng punto66 = new LatLng(-2.505708122544912, -78.94951410591602);
puntos_lat[65] = punto66.latitude;
puntos_long[65] = punto66.longitude;
```

```
LatLng punto_centro = new LatLng(-2.505837, -78.949736);
mMap.addMarker(new MarkerOptions().icon(BitmapDescriptorFactory.fromResource(R.mipmap.ic_puntomedio)).anchor(0.0f, 1.0f)
        .position(punto_centro).title("Marcador central"));
mMap.moveCamera(CameraUpdateFactory.newLatLng(punto_centro));
double lat_puntoCentro = punto_centro.latitude;
double lon_puntoCentro = punto_centro.longitude;


mMap.addPolyline(new PolylineOptions().add(punto1, punto2).width(5).color(Color.RED));
mMap.addPolyline(new PolylineOptions().add(punto2, punto3).width(5).color(Color.RED));
mMap.addPolyline(new PolylineOptions().add(punto3, punto4).width(5).color(Color.RED));
mMap.addPolyline(new PolylineOptions().add(punto4, punto5).width(5).color(Color.RED));
mMap.addPolyline(new PolylineOptions().add(punto5, punto6).width(5).color(Color.RED));
mMap.addPolyline(new PolylineOptions().add(punto6, punto7).width(5).color(Color.RED));
mMap.addPolyline(new PolylineOptions().add(punto7, punto8).width(5).color(Color.RED));
mMap.addPolyline(new PolylineOptions().add(punto8, punto9).width(5).color(Color.RED));
mMap.addPolyline(new PolylineOptions().add(punto9, punto10).width(5).color(Color.RED));
mMap.addPolyline(new PolylineOptions().add(punto10, punto11).width(5).color(Color.RED));
mMap.addPolyline(new PolylineOptions().add(punto11, punto12).width(5).color(Color.RED));
mMap.addPolyline(new PolylineOptions().add(punto12, punto13).width(5).color(Color.RED));
mMap.addPolyline(new PolylineOptions().add(punto13, punto14).width(5).color(Color.RED));
mMap.addPolyline(new PolylineOptions().add(punto14, punto15).width(5).color(Color.RED));
mMap.addPolyline(new PolylineOptions().add(punto15, punto16).width(5).color(Color.RED));
mMap.addPolyline(new PolylineOptions().add(punto16, punto17).width(5).color(Color.RED));
mMap.addPolyline(new PolylineOptions().add(punto17, punto18).width(5).color(Color.RED));
mMap.addPolyline(new PolylineOptions().add(punto18, punto19).width(5).color(Color.RED));
mMap.addPolyline(new PolylineOptions().add(punto19, punto20).width(5).color(Color.RED));
mMap.addPolyline(new PolylineOptions().add(punto20, punto21).width(5).color(Color.RED));
mMap.addPolyline(new PolylineOptions().add(punto21, punto22).width(5).color(Color.RED));
mMap.addPolyline(new PolylineOptions().add(punto22, punto23).width(5).color(Color.RED));
mMap.addPolyline(new PolylineOptions().add(punto23, punto24).width(5).color(Color.RED));
mMap.addPolyline(new PolylineOptions().add(punto24, punto25).width(5).color(Color.RED));
mMap.addPolyline(new PolylineOptions().add(punto25, punto26).width(5).color(Color.RED));

//Ingresamos a la base FIREBASE


mDatabase.child("dato1").addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

        for (Marker marker : realTimeMarkers) {
            marker.remove();
        }
        if (dataSnapshot.exists()) {
            String ubi1 = dataSnapshot.child("ubicacion1").getValue().toString();

            String[] ubicacion1 = ubi1.split(",");
            lat_ubicacion1 = Double.parseDouble(ubicacion1[1].substring(5));
            lon_ubicacion1 = Double.parseDouble(ubicacion1[2].substring(6, ubicacion1[2].length() - 1));
            System.out.println(lat_ubicacion1 + "*****************************latitud ubicacion1");
            System.out.println(lon_ubicacion1 + "*****************************longitud ubicacion1");


            //  lat_ubicacion1 = Double.parseDouble(ubicacion1[2].substring(5,ubicacion1[2].length()-1));
            //System.out.println(lat_ubicacion1 + "*****************************latitud ubicacion1");
            //lon_ubicacion1 = Double.parseDouble(ubicacion1[1].substring(6));
            //System.out.println(lon_ubicacion1 + "*****************************longitud ubicacion1");


            String ubi2=dataSnapshot.child("ubicacion2").getValue().toString();
            String [] ubicacion2= ubi2.split(",");
            lat_ubicacion2 = Double.parseDouble(ubicacion2[1].substring(5));
            lon_ubicacion2 = Double.parseDouble(ubicacion2[2].substring(6,ubicacion2[2].length()-1));
            System.out.println(lat_ubicacion2 + "*****************************latitud ubicacion2");
            System.out.println(lon_ubicacion2 + "*****************************longitud ubicacion2");
```

```
String ubi2=dataSnapshot.child("ubicacion2").getValue().toString();
String [] ubicacion2= ubi2.split(",");
lat_ubicacion2 = Double.parseDouble(ubicacion2[1].substring(5));
lon_ubicacion2 = Double.parseDouble(ubicacion2[2].substring(6,ubicacion2[2].length()-1));
System.out.println(lat_ubicacion2 + "*******************************latitud ubicacion2");
System.out.println(lon_ubicacion2 + "*******************************longitud ubicacion2");


// lat_ubicacion2 = Double.parseDouble(ubicacion2[2].substring(5,ubicacion2[2].length()-1));
 //System.out.println(lat_ubicacion2+"*******************************longitud ubicacion2");
//lon_ubicacion2 = Double.parseDouble(ubicacion2[1].substring(6));
//System.out.println(lon_ubicacion2+"*******************************longitud ubicacion2");

String ubi3=dataSnapshot.child("ubicacion3").getValue().toString();
String [] ubicacion3= ubi3.split(",");
lat_ubicacion3 = Double.parseDouble(ubicacion3[1].substring(5));
lon_ubicacion3 = Double.parseDouble(ubicacion3[2].substring(6,ubicacion3[2].length()-1));
System.out.println(lat_ubicacion3 + "*******************************latitud ubicacion3");
System.out.println(lon_ubicacion3 + "*******************************longitud ubicacion3");


// lat_ubicacion3 = Double.parseDouble(ubicacion3[2].substring(5,ubicacion3[2].length()-1));
//System.out.println(lat_ubicacion3+"*******************************longitud ubicacion3");
//lon_ubicacion3 = Double.parseDouble(ubicacion3[1].substring(6));
//System.out.println(lon_ubicacion3+"*******************************longitud ubicacion3");
```

```
Maps map= new Maps();

//*************************************************************************************
//********************************** UBICACION 1 **************************************
//*************************************************************************************
for(int i=0; i <70; i=i+1){

    //distancias[i]=map.calcularDistancias(lat_ub1,lon_ub1, lat,lon);
    //double pos =map.calcularDistancias(0.59436,-77.81676, 0.594595,-77.816578);
    distancias[i] =map.calcularDistancias(lat_ubicacion1,lon_ubicacion1, puntos_lat[i],puntos_long[i]);
    System.out.print("Origen : "+ lat_ubicacion1+" "+lon_ubicacion1);
    System.out.print("  Punto "+ i+": "+ puntos_lat[i]+"  " +puntos_long[i]);
    System.out.println("  Distancia :"+distancias[i]);
}

double dist_center_ubicacion1 = map.calcularDistancias(lat_ubicacion1,lon_ubicacion1, lat_puntoCentro,lon_puntoCentro);
System.out.println("distancia desde el centro hasta ubicacion1: "+dist_center_ubicacion1);

double min_dist_ubicacion1 =10000000000.0;
int pos_min_dist_ubicacion1 =0;
for(int i=0; i<distancias.length;i=i+1){
    if(distancias[i]< min_dist_ubicacion1){
        min_dist_ubicacion1=distancias[i];
        pos_min_dist_ubicacion1=i;
    }
}
System.out.println("Distancia menor:"+min_dist_ubicacion1+" marcador: "+(pos_min_dist_ubicacion1+1));
double dist_center_min_ubicacion_1 =map.calcularDistancias(lat_puntoCentro,lon_puntoCentro, puntos_lat[pos_min_dist_ubicacion1],puntos_long[pos_min_
System.out.println("distancia desde el centro hasta la posicion menor: "+dist_center_min_ubicacion_1);
 double min_dist_ubicacion1_2 =10000000000.0;
 int pos_min_dist_ubicacion1_2 =0;
 for(int i=0; i<distancias.length;i=i+1){
     if((distancias[i]< min_dist_ubicacion1_2) & (distancias[i]!=min_dist_ubicacion1)){
         min_dist_ubicacion1_2=distancias[i];
         pos_min_dist_ubicacion1_2=i;
     }
 }

 System.out.println("Segunda Distancia menor:"+min_dist_ubicacion1_2+" marcador: "+(pos_min_dist_ubicacion1_2+1));
 double dist_center_min_ubicacion1_2 =map.calcularDistancias(lat_puntoCentro,lon_puntoCentro, puntos_lat[pos_min_dist_ubicacion1_2],puntos_long[pos_m
 System.out.println("distancia desde el centro hasta la posicion menor 2: "+dist_center_min_ubicacion1_2);

//  int distancia_centro_ubicacion1=(int) dist_center_ubicacion1;
 // int distancia_menor1_centro_ubi1=(int) min_dist_ubicacion1;
 // int distancia_menor_centro_ubi2=(int)min_dist_ubicacion1_2;

 //ALERTA NODO 1
 if((dist_center_ubicacion1 > dist_center_min_ubicacion_1) && (dist_center_ubicacion1 > dist_center_min_ubicacion1_2)){
     createNotificationChannel();
     crearNotificacion();
             MarkerOptions markerOptions_ubicacion1=new MarkerOptions();
     markerOptions_ubicacion1.position(new LatLng(lat_ubicacion1,lon_ubicacion1)).icon(BitmapDescriptorFactory.fromResource(R.mipmap.ic_ubi_fuera))
             .title("Vaca 1 Latitud: "+lat_ubicacion1+" Longitud: "+lon_ubicacion1);
     tmpRealTimeMarker.add(mMap.addMarker(markerOptions_ubicacion1));
     System.out.println("NOTIFICACION ubicacion 1 ******************************************");
 }
 else{

     MarkerOptions markerOptions_ubicacion1=new MarkerOptions();
     markerOptions_ubicacion1.position(new LatLng(lat_ubicacion1,lon_ubicacion1)).icon(BitmapDescriptorFactory.fromResource(R.mipmap.ic_ubi_dentro))
             .title("Vaca 1 Latitud: "+lat_ubicacion1+" Longitud: "+lon_ubicacion1);
     tmpRealTimeMarker.add(mMap.addMarker(markerOptions_ubicacion1));
 }
```

Activar Windows

```
//*********************************************************************************
//********************************** UBICACION 2 ***********************************
//*********************************************************************************
for(int i=0; i <70; i=i+1){

    //distancias[i]=map.calcularDistancias(lat_ub1,lon_ub1, lat,lon);
    //double pos =map.calcularDistancias(0.59436,-77.81676, 0.594595,-77.816578);
    distancias[i] =map.calcularDistancias(lat_ubicacion2,lon_ubicacion2, puntos_lat[i],puntos_long[i]);
    System.out.print("Origen : "+ lat_ubicacion2+" "+lon_ubicacion2);
    System.out.print("  Punto "+ i+": "+ puntos_lat[i]+"  " +puntos_long[i]);
    System.out.println("  Distancia :"+distancias[i]);
}

double dist_center_ubicacion2 = map.calcularDistancias(lat_ubicacion2,lon_ubicacion2, lat_puntoCentro,lon_puntoCentro);
System.out.println("distancia desde el centro hasta ubicacion1: "+dist_center_ubicacion2);

double min_dist_ubicacion2 =10000000000.0;
int pos_min_dist_ubicacion2 =0;
for(int i=0; i<distancias.length;i=i+1){
    if(distancias[i]< min_dist_ubicacion2){
        min_dist_ubicacion2=distancias[i];
        pos_min_dist_ubicacion2=i;
    }
}
System.out.println("Distancia menor:"+min_dist_ubicacion2+" marcador: "+(pos_min_dist_ubicacion2+1));
double dist_center_min_ubicacion2_1 =map.calcularDistancias(lat_puntoCentro,lon_puntoCentro, puntos_lat[pos_min_dist_ubicacion2],puntos_long[pos_mir
System.out.println("distancia desde el centro hasta la posicion menor: "+dist_center_min_ubicacion2_1);

double min_dist_ubicacion2_2 =10000000000.0;
int pos_min_dist_ubicacion2_2 =0;
for(int i=0; i<distancias.length;i=i+1){
    if((distancias[i]< min_dist_ubicacion2_2) & (distancias[i]!=min_dist_ubicacion2)){
        min_dist_ubicacion2_2=distancias[i];
        pos_min_dist_ubicacion2_2=i;
    }
```

Activar Windows

```
 //ALERTA NODO 2


 if((dist_center_ubicacion2 > dist_center_min_ubicacion2_1) & (dist_center_ubicacion2 > dist_center_min_ubicacion2_2) ){
    createNotificationChannel();
    crearNotificacion();

    MarkerOptions markerOptions_ubicacion2=new MarkerOptions();
    markerOptions_ubicacion2.position(new LatLng(lat_ubicacion2,lon_ubicacion2)).icon(BitmapDescriptorFactory.fromResource(R.mipmap.ic_ubi_fuera))
            .title("Vaca 2 Latitud: "+lat_ubicacion2+" Longitud: "+lon_ubicacion2);
    tmpRealTimeMarker.add(mMap.addMarker(markerOptions_ubicacion2));
    System.out.println("NOTIFICACION*****ubicacion 2**************************************");
 }
 else{
    MarkerOptions markerOptions_ubicacion2=new MarkerOptions();
    markerOptions_ubicacion2.position(new LatLng(lat_ubicacion2,lon_ubicacion2)).icon(BitmapDescriptorFactory.fromResource(R.mipmap.ic_ubi_dentro))
            .title("Vaca 2 Latitud: "+lat_ubicacion2+" Longitud: "+lon_ubicacion2);
    tmpRealTimeMarker.add(mMap.addMarker(markerOptions_ubicacion2));
 }

 private void crearNotificacion() {

     NotificationCompat.Builder builder = new NotificationCompat.Builder(getApplicationContext(), CHANNEL_ID);
     builder.setSmallIcon(R.drawable.ic_alerta);
     builder.setContentTitle("Notificación FINCA ACERO");
     builder.setContentText("Ganado alejandose del predio");
     builder.setColor(Color.BLUE);
     builder.setPriority(NotificationCompat.PRIORITY_DEFAULT);
     //color de pantalla
     builder.setLights(Color.MAGENTA, 1000, 1000);
     //vibración
     builder.setVibrate(new long[]{1000, 1000, 1000, 1000, 1000});
     //sonido
     builder.setDefaults(Notification.DEFAULT_SOUND);
     NotificationManagerCompat notificationManagerCompat = NotificationManagerCompat.from(getApplicationContext());
     notificationManagerCompat.notify(NOTIFICACION_ID, builder.build());
 }

 private void createNotificationChannel() {
     if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
         CharSequence name = "Notificación";
         NotificationChannel notificationChannel = new NotificationChannel(CHANNEL_ID, name, NotificationManager.IMPORTANCE_DEFAULT);
         NotificationManager notificationManager = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
         notificationManager.createNotificationChannel(notificationChannel);
     }

 }

 // Colores de los bordes de la pantalla de la aplicación
 private void cambiarColor(String primaryDark, String primary, String background){
     getWindow().setStatusBarColor(Color.parseColor(primaryDark));
     //getSupportActionBar().setBackgroundDrawable(new ColorDrawable(Color.parseColor(primary)));
     getWindow().setBackgroundDrawable(new ColorDrawable(Color.parseColor(background)));
     getWindow().setNavigationBarColor(Color.parseColor(primary));
 }
```

Activar
Ve a Confi

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

DIRECCIÓN DE BIBLIOTECAS Y RECURSOS DEL APRENDIZAJE

UNIDAD DE PROCESOS TÉCNICOS

REVISIÓN DE NORMAS TÉCNICAS, RESUMEN Y BIBLIOGRAFÍA

Fecha de entrega: 23/ 11 / 2023

INFORMACIÓN DEL AUTOR
Nombres – Apellidos: Oscar Benjamín Acero Guamán
INFORMACIÓN INSTITUCIONAL
Facultad: Informática y Electrónica
Carrera: Electrónica y Automatización
Título a optar: Ingeniero en Electrónica y Automatización
f. Analista de Biblioteca responsable: Ing. Fernanda Arévalo M.

UNIDAD DE PROCESOS TÉCNICOS
DBRA
DIRECCION DE BIBLIOTECAS Y RECURSOS DEL APRENDIZAJE
ESPOCH

DIRECCION DE BIBLIOTECAS Y RECURSOS PARA EL APRENDIZAJE Y LA INVESTIGACION
Ing. Jhonatan Parreño Urquillos MBA
ANALISTA DE BIBLIOTECA 1

1797-DBRA-UPT-2023