Sistema de trazabilidad para usuarios de Transmilenio

Autor:

Juan Diego Sánchez Parra

Director: Ing. Daniel Jaramillo Ramírez, Ph.D.

Codirector: Ing. Rafael Puerta

Pontificia Universidad Javeriana

Facultad de ingeniería

Departamento de ingeniería electrónica

Bogotá, 2021

Agradecimientos

En primer lugar, quiero agradecer a mi familia por el amor, la confianza y el acompañamiento dado en estos años de formación personal y profesional.

Agradezco a mis directores los ingenieros Daniel Jaramillo y Rafael Puerta, cuyos aportes fueron indispensables en el desarrollo de este trabajo de grado. También quiero agradecer a la universidad Javeriana, en especial al departamento de ingeniería electrónica por brindarme todas las herramientas y los conocimientos adquiridos en esta etapa de mi vida.

Finalmente quiero agradecer a todos mis amigos y compañeros de la universidad por los valiosos momentos y por su conocimiento aportado.

Muchas gracias a todos.

Tabla de contenido

Lista de imágenes

Lista de Tablas

1. Introducción

Los sistemas de transporte masivo tienen como objetivo principal movilizar de una forma eficiente a los habitantes en las ciudades, para cumplir con este objetivo es fundamental que exista una buena planificación del sistema partiendo de la información recopilada referente a las condiciones y necesidades del mismo, como resultado de este ejercicio se conocen los problemas, se proponen soluciones y se toman decisiones en beneficio de la movilidad. Debido a lo anterior es importante que la información adquirida sea fiable, concuerde con la realidad y sea recopilada de forma frecuente para contar con un conocimiento preciso de las exigencias actuales del sistema.

Actualmente, en Bogotá y municipios aledaños se realizan periódicamente dos informes que sirven de sustento para la planificación del sistema de transporte, en primer lugar la Encuesta de movilidad realizada cada cuatro años, siendo la última del 2019 [1], en este año se consultó a aproximadamente 100.000 personas sobre los trayectos realizados, el medio de transporte utilizado, el motivo del viaje, entre otros, tal información permite conocer los patrones de desplazamiento de los ciudadanos y así priorizar el gasto público de forma eficiente en temas de movilidad a largo plazo, principalmente en las obras requeridas. El segundo es el informe Estadísticas de oferta y demanda del sistema de transporte público – SITP [2] realizado bimestralmente por la empresa Transmilenio S.A, en este informe se presentan las cifras de oferta respecto a la infraestructura del sistema (estaciones, buses), personal humano, las cifras de demanda basadas en las validaciones de las entradas en las estaciones y buses, igualmente los índices de eficiencia basados en pasajeros transportados y la velocidad en el sistema.

Los dos informes mencionados anteriormente disponen información precisa para el sistema de transporte tanto en infraestructura como disponibilidad del servicio, sin embargo, en el caso de la Encuesta de movilidad la periodicidad de su realización no se adapta a las condiciones actuales y a posibles variaciones que se presenten, por ejemplo la actual pandemia del Covid-19, en la que la demanda del Transmilenio tuvo una disminución de pasajeros de hasta un 87% [3], debido a medidas tomadas por la administración distrital; caso contrario es el informe bimestral de Transmilenio, que sí presenta información de la demanda del sistema con mayor frecuencia, debido a que su periodicidad es menor, analiza igualmente las validaciones de entradas por los torniquetes de las estaciones, sin embargo, esta información al ser de carácter general no contempla aspectos específicos de importancia como el trayecto que realizan los usuarios, la ruta escogida y los tiempos del usuario en el servicio.

Existe un inconformismo con el sistema de transporte público por parte de sus usuarios, según cifras de Bogotá cómo vamos del año 2018 [4], el 35% de los habitantes realizaba sus viajes en el componente troncal de Transmilenio y el nivel de satisfacción era del 13%, la mayoría de las quejas se centran en la demora al esperar el servicio que necesitan y a la congestión en las estaciones. Esta insatisfacción resultante del desconocimiento de las necesidades de los usuarios ha provocado que cada vez más personas se bajen del sistema y opten por otros modos de transporte como el carro y la moto de uso particular, que generan mayor contaminación y congestión en las vías de la ciudad y que afecta la idea de una movilidad sostenible en la ciudad de Bogotá.

El sistema de transporte masivo de la ciudad de Bogotá requiere una alternativa diferente a los dos informes enunciados, que le permita obtener más información relacionada con las necesidades de sus usuarios y realizar una planificación (a largo plazo) y una operación (a corto plazo) del transporte acorde a la realidad del sistema. Este trabajo de grado presenta un sistema piloto diseñado para adquirir en tiempo real y de forma automática la información que requiere un sistema como Transmilenio, entre ellos, el trayecto realizado por los usuarios conociendo la estación de origen y destino, los tiempos de ingreso y salida, las condiciones actuales del sistema para la operación, los usuarios en estación y el tiempo de espera del

servicio. Además, esta solución propuesta permite obtener la información en sitio, es decir, en contacto con la infraestructura del sistema (estaciones y buses) con una comunicación entre el usuario (su dispositivo celular) y los prototipos en las estaciones, sin el uso de datos o internet en el dispositivo celular. Lo que le permite obtener la información de forma directa y en tiempo real, conociendo el comportamiento del usuario al hacer uso del servicio Transmilenio.

2. Marco teórico

2.1 Geovallado

Un geovallado es una zona delimitada o acotada de un espacio geográfico real, es decir, un área que encierra un lugar determinado, como se puede ver en la imagen 1. Esta área puede representar un lugar de interés para una o varias personas, como el sitio de trabajo, de estudio o incluso de algún almacén de preferencia [5].

Imagen 1. Geovallado

El objetivo de delimitar una ubicación con un geovallado es que se produzca alguna acción cuando se detecte el ingreso, la salida o la permanencia en este lugar de interés, un ejemplo común es el envío de una notificación por correo o al celular para indicar que se está en este lugar.

Los usuarios de Transmilenio tanto del servicio troncal como del zonal tienen que acercarse a una estación o un paradero para hacer uso del sistema, por tal razón, las estaciones del Transmilenio es este trabajo representan zonas de geovallados.

2. 2 Búsqueda de WiFi [6]

La búsqueda de WiFi es el proceso que hacen los dispositivos móviles para encontrar las redes inalámbricas disponibles del área. Existen dos tipos de búsqueda en la que el dispositivo toma diferentes roles, estos son el escaneo pasivo y el escaneo activo.

En el escaneo pasivo, el dispositivo se mueve entre los canales de WiFi esperando recibir las tramas que envían los puntos de acceso, de esta forma obtiene la información de las redes disponibles con un mayor esfuerzo debido a su permanencia como receptor. En la imagen 2 se muestra gráficamente el escaneo pasivo.

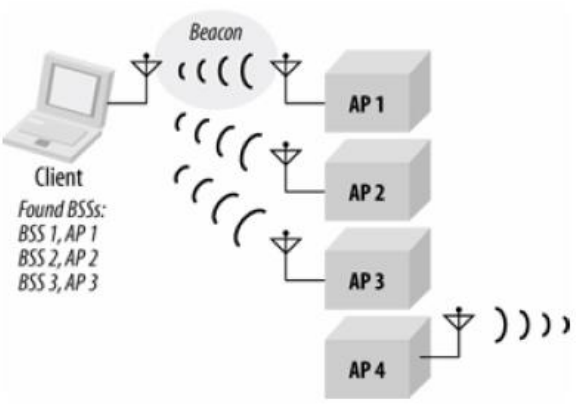


Imagen 2. Búsqueda pasiva de WiFi. [6]

Por el contrario, en el escaneo activo el dispositivo toma la iniciativa y envía tramas o mensajes inalámbricos del protocolo IEEE802.11 solicitando respuesta de las redes WiFi. En cada canal transmite un mensaje tipo "*Probe Request*", a lo que los puntos de acceso responden con un mensaje tipo "*Probe response*" con su información, este proceso se muestra gráficamente en la imagen 3.

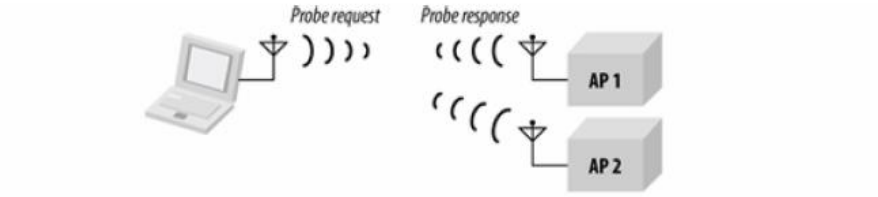


Imagen 3. Búsqueda activa de WiFi. [6]

En la imagen 4 se presenta el procedimiento realizado inalámbricamente considerando el tiempo y dos puntos de acceso en el ambiente.

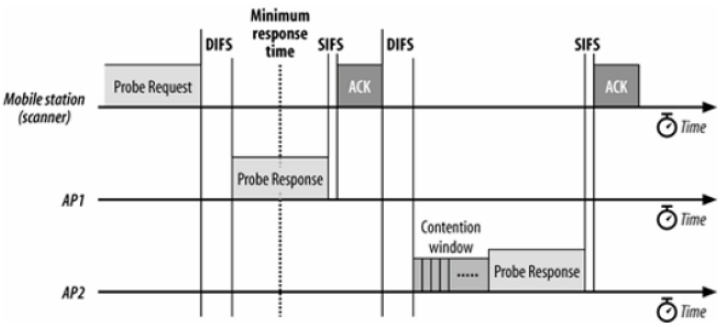


Imagen 4. Representación inalámbrica búsqueda activa de WiFi [6]

Como se mencionó, el escaneo activo envía la trama del mensaje tipo "*Probe Request*" del protocolo IEEE802.11, lo que permite que al hacer una búsqueda de WiFi el dispositivo fuerce una comunicación, enviando su información en el paquete, en especial la dirección física o MAC que permite reconocer el dispositivo.

2.3 Protocolo IEEE802.11 [6]

IEEE802.11 es una familia de estándares que definen las normas para el uso de conectividad inalámbrica, más conocido como WiFi. La siguiente imagen presenta la estructura básica de las tramas o "*frames*" que son enviados de forma inalámbrica con el protocolo IEEE802.11.

Imagen 5. Trama paquete IEEE802.11 [6]

No todos los campos son utilizados en el envío de cada paquete, esto depende del tipo y subtipo que identifica un paquete. Los campos de las direcciones representan direcciones MAC o también llamadas direcciones físicas de un dispositivo de red, es un identificador único que permite distinguir a cada dispositivo, estas direcciones MAC tienen un total de 48 bits separados en 6 bloques de bytes. En la trama del paquete IEEE802.11 de la imagen 5, la dirección 1 representa la MAC de destino, mientras que la dirección 2 es la MAC de origen, importante para identificar el dispositivo que recibe y el que envía.

2.3.1 Paquete *Probe Request*

En el campo de control del *frame* define los tipos y subtipos del paquete enviado con el protocolo IEEE802.11, los campos de este se ven en la imagen 6.

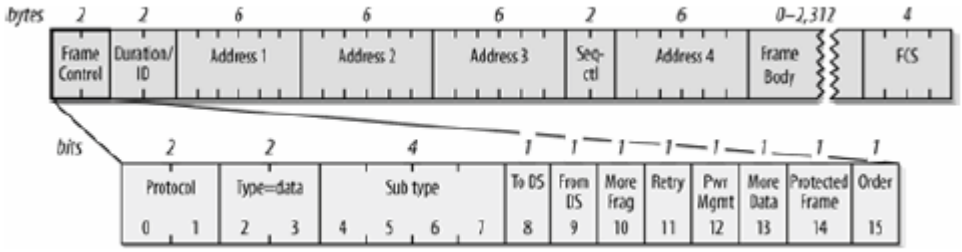


Imagen 6. Frame control [6]

El campo de protocolo se refiere a la versión que contiene el paquete, no existen modificaciones, por lo que este campo es una constante cero.

Los tipos de paquetes existentes son los siguientes.

- Management *frames* (*type* = 00)
- Control *frames* (*type* = 01)
- Data *frames* (*type* = 10)

La combinación (*type* = 11) está reservada.

El tipo de paquete que contiene el mensaje de interés (*Probe Request*) es el *Management frame*, que cuenta con las siguientes combinaciones.

Subtype value	Subtype name
Management frames (type=00)[a]	
0000	Association request
0001	Association response
0010	Reassociation request
0011	Reassociation response
0100	Probe request
0101	Probe response
1000	Beacon
1001	Announcement traffic indication message (ATIM)
1010	Disassociation
1011	Authentication
1100	Deauthentication
1101	Action (for spectrum management with 802.11h, also for QoS)

Imagen 7. Valor paquete Probe Request.[6]

Teniendo conocimiento de lo anterior para identificar un mensaje IEEE802.11 tipo MGMT con subtipo *Probe Request* (enviado en una búsqueda activa de WiFi), basta con filtrar el primer byte del *frame* control de la siguiente forma.

Bits	00000100
Decimal o hexadecimal	4
Representación primer byte *frame* control *Probe Request*	

Tabla 1. Filtro Probe Request

Otro aspecto importante al captar los paquetes es el valor de RSSI, que es el factor que indica el nivel de potencia del paquete captado, tiene unidades de dBm y este valor se representa con numeros negativos. Los paquetes captados con valor mas cercano al cero son los de mayor intensidad o potencia recibida y los valores más negativos indican mayor perdida de señal. Este valor es asignado por el captador de pquetes.

2.4 Aleatorización MAC [7]

En los sistemas operativos Android, con el fin de evitar un reconocimiento de actividad y mantener la privacidad del usuario, fue incluida la opción de aleatorizar la dirección MAC en las búsquedas de WiFi, esto quiere decir que en el campo de la trama con la dirección MAC de origen se asigna una dirección al azar, diferente de la verdadera dirección MAC del dispositivo, esto en cada búsqueda.

Esta opción fue incluida desde el sistema operativo Android Oreo u 8, lo que no permite hacer un seguimiento del dispositivo por medio de la dirección MAC desde esta versión o superiores. Sin embargo, desde Android 10 cuando los dispositivos están conectados a una red permite habilitar la opción de fijar la MAC propia al hacer búsquedas de WiFi [8].

Este trabajo está enfocado en el desarrollo de una aplicación que permita hacer rastreo de dispositivos que no aleatorizan su dirección MAC, por tal motivo se usaron dispositivos con distinta versión de sistema operativo Android que no hicieran aleatorización MAC o que la hicieran y se pudiera configurar, es decir, dispositivos con Android menor a 8 o con Android 10 o superior.

3. Objetivo general y específicos

3.1 Objetivo general

Desarrollar e implementar un sistema piloto que permita conocer en tiempo real el trayecto detallado que realizan los usuarios en Transmilenio.

3.2 Objetivos específicos

1 Diseñar una aplicación móvil para el usuario de Transmilenio que permita distinguirlo de los demás pasajeros dentro del sistema, que haga uso del geovallado y tenga bajo consumo de batería.

2 Hacer uso de los mensajes tipo *Probe Request* del protocolo de comunicaciones IEEE802.11 desde la aplicación móvil, que permita enviarlos periódicamente y en los dispositivos captadores de estos mensajes obtenerlos con precisión para posteriormente obtener los tiempos y calcular la duración del usuario en estación.

3 Transmitir la información recolectada en tiempo real desde una estación emulada del Transmilenio mediante un módulo GPRS a un servidor web de Firebase donde se almacene.

4 Realizar el análisis de la información transmitida y presentar la información agregada de varios usuarios.

4. Desarrollo

En este capítulo se describe el sistema piloto realizado. Este consiste en la integración de distintas partes tanto de hardware como de software que serán explicadas. El diagrama de bloques del sistema piloto se presenta a continuación.

4.1 Diagrama de bloques del sistema piloto

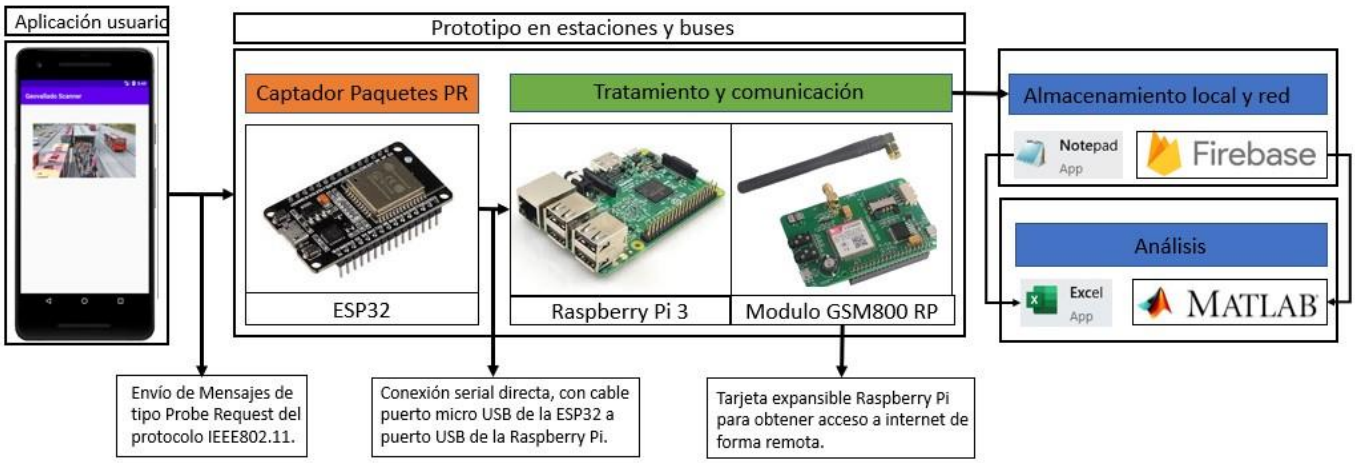


Imagen 8. Diagrama de bloques sistema realizado [Autor]

El sistema consta de las siguientes partes;

- Aplicación móvil: aplicación para dispositivos con sistema operativo Android, diseñada para un usuario común de Transmilenio, utiliza la ubicación, agrega los geovallados y acciona las búsquedas de WiFi activas de forma controlada.

- Captador de paquetes *Probe Request*: uso de tarjeta de desarrollo ESP32 con la capacidad de captar los paquetes inalámbricos del protocolo IEEE802.11, filtrándolos para obtener únicamente los paquetes tipo *Probe Request* con una cobertura determinada.

- Tratamiento y comunicación: con la Raspberry Pi se recibe las capturas hechas por la ESP32, se realiza el procedimiento para cada dirección MAC de asignación de tiempos, cálculos de duración y se almacena la información de estas capturas. Se requiere conexión a internet por lo que se usó la tarjeta TAR SIM800 RPI que usa datos de una tarjeta SIM.

- Almacenamiento: se guarda la información de forma local en un archivo de texto, escribiendo en este para cada captura; tiempo de captura, dirección MAC, valor de RSSI, número de repetición, duración con captura anterior y duración total. En el almacenamiento en el servidor de Firebase, se distingue por dirección MAC, se almacena el tiempo inicial y final de captura, este último se actualiza a la vez que llega cada captura con esa dirección MAC, cuando el dispositivo supera una duración mayor a los 3 minutos se envía al servidor esta duración, para indicar que este usuario ha estado esperando mucho por su servicio.

- Análisis: se observa el comportamiento del sistema con el archivo de texto mediante Excel, con esta información se puede mirar el tiempo entre capturas y el valor de RSSI. Con la información enviada en tiempo real a Firebase, se trató mediante el software de Matlab que presenta la información organizada de forma visual con un tablero de gráficas.

4.2 Aplicación móvil “Geovallado - Búsqueda de WiFi”

La aplicación se enfoca en dispositivos celulares que tengan el sistema operativo Android, para la programación se usó el lenguaje Kotlin [9] junto al entorno de desarrollo oficial de Android, Android Studio[10]. La amplia documentación que ofrece Android para los desarrolladores de aplicaciones y los foros en los que se plantean las dudas fueron de gran ayuda en su desarrollo.

La aplicación cuenta con dos funcionalidades principales, la primera es el geovallado y la segunda es la búsqueda activa de WiFi de forma repetitiva. Estas funcionalidades se vinculan de tal forma que la búsqueda de WiFi se supedita a las activaciones por el geovallado. Además, se garantiza que la aplicación permanezca funcionando sin caer en restricciones impuestas al entrar a segundo plano y cuando el dispositivo entra en modo de descanso. Código de la aplicación en anexo 1.

4.2.1 Geovallado

Como ya se mencionó, el geovallado permite delimitar un área de interés mediante la latitud, la longitud y un radio. Para esto se usó la API de geovallado que ofrece Android [11], se agregaron los geovallados (estaciones de TM), se definió el receptor de activación de geovallados que recibe y actúa cuando se acciona alguno de estos, siguiendo el documentación presentada en [12].

Existen tres activadores que se pueden usar, como se ve en la imagen 9, en esta aplicación se usaron los tres de la siguiente forma.

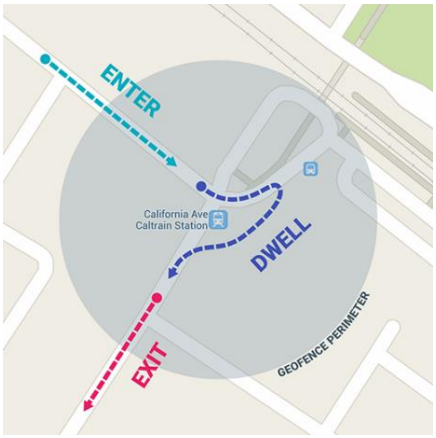


Imagen 9. Activadores de geovallado Android. [12]

1. Entrada: envía notificación y activa la búsqueda repetitiva de WiFi con un periodo inicial.

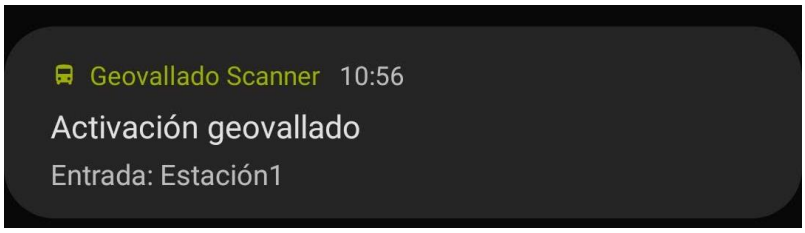


Imagen 10. Notificación de entrada a geovallado

2. Permanencia: cuando se cumple el tiempo definido de permanencia en el mismo sitio se activa, se cambia el periodo de repetición a uno mayor. Este tiempo fue definido de cinco minutos, con el objetivo principal de preservar la batería en el celular.

3. Salida: apaga bandera del geovallado y posterior decide si apagar o no la búsqueda de WiFi.

- Recomendaciones y limitación del geovallado

La documentación presenta las recomendaciones para que el geovallado actúe de una forma óptima, la primera es el valor definido para el radio, como se describe en la imagen 11. Entre mayor el radio habrá una mejor respuesta del geovallado, según la recomendación subrayada y teniendo en cuenta el uso dado (tamaño estaciones de TM), se definió un radio de 60 metros, un poco más amplio para garantizar una activación oportuna. Teniendo en cuenta que esta precisión depende de que exista una red WiFi disponible en el geovallado.

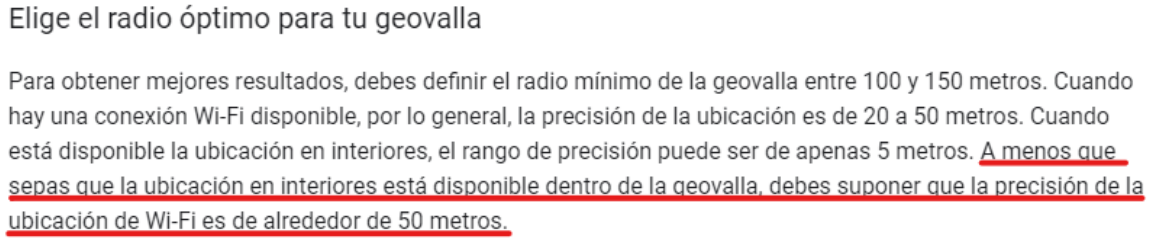


Imagen 11. Radio optimo geovallado. [12]

Además, existen limitaciones que afectan el rendimiento de las activaciones del geovallado, estas dependen de la ubicación en segundo plano [13] que es cuando la aplicación no está visible por el usuario, con el fin de preservar la batería del dispositivo se limita la frecuencia en el que las Apps pueden recibir actualizaciones de ubicación desde Android 8, esto también afecta la API de geovallado, de la siguiente forma.

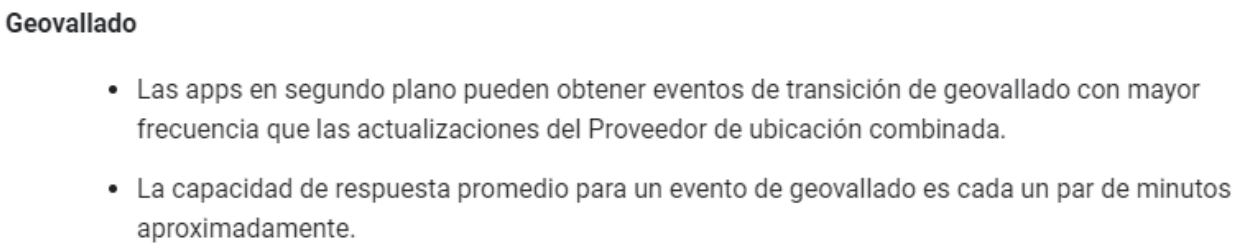


Imagen 12. Limitaciones geovallado en segundo plano. [13]

Esta respuesta tardía a las activaciones del geovallado afecta considerablemente la intensión de precisar el tiempo de duración del usuario en la estación, dadas estas condiciones se depende de factores del dispositivo y del usuario para una activación oportuna.

4.2.2 Búsqueda repetitiva de WiFi activa

Android permite realizar la búsqueda de redes WiFi con la API de WiFiManager [14], con esta se puede obtener la lista de puntos de acceso cercanos, junto a información de estas redes. Esta al ser solicitada es una búsqueda de WiFi de tipo activa, con el envío de mensajes tipo "*Probe Request*" a lo que espera las respuestas con mensajes tipo "*Probe response*" con la información.

Para esto se siguió la documentación presentada en [15], en el que se definen tres pasos fundamentales al realizar la búsqueda y obtener la información de las redes.

1. Registrar un receptor de transmisiones de búsquedas de WiFi: es llamado cuando se completa la solicitud de búsqueda, indica si fue una búsqueda exitosa o no. Para dispositivos con Android 10 o posterior también obtiene los resultados de las búsquedas pasivas.
2. Solicitar búsqueda: para lanzar la búsqueda de WiFi activa se requiere solicitarla mediante la instrucción `WiFiManager.startScan()`, para hacer una búsqueda repetitiva se ejecuta esta instrucción reiteradamente, definida a un tiempo de periodo que incluye restricciones.
3. Obtener resultados búsqueda: presenta los resultados obtenidos de la búsqueda hecha recientemente, recibe información de los puntos de acceso como [16]; SSID, RSSI, BSSID, el tipo de autenticación y otras más.

Con los pasos anteriores la búsqueda es realizada correctamente. Sin embargo, se debe tener en cuenta que Android impone restricciones y limitaciones que restringen el uso de esta funcionalidad, esto para preservar la batería del dispositivo y evitar el mal uso que se le puede dar a esta.

- Limitaciones búsqueda de WiFi [15]

La principal limitación es la frecuencia en la que se hacen las búsquedas, tanto para primer plano, que es cuando el usuario tiene la aplicación visible, como cuando la inició, pero ya no la tiene abierta que es en segundo plano.

La documentación presenta las siguientes frecuencias permitidas, empezando desde Android 8. Si no se cumplen estas frecuencias, al realizar la búsqueda esta falla.

Versión Android	Primer Plano	Segundo plano
8	No menciona	Cada APP 1 cada 30 minutos
9	Cada APP 4 búsquedas cada 2 minutos.	Todas las APP 1 cada 30 minutos
10		

Tabla 2. Frecuencia permitida de solicitud búsqueda WiFi en Android.

Para garantizar que las solicitudes de búsqueda no fallen se definió una repetición con un periodo de 30 segundos, que es lo máximo permitido y que garantiza que exista un seguimiento del usuario de forma continua y precisa. Tener en cuenta que esto es para primer plano.

Para dispositivos con Android 9 y posteriores existe otra limitación, la búsqueda requiere que se tenga activa la ubicación en el dispositivo, sino esta falla.

4.2.3 Vinculación geovallado - Búsqueda de WiFi activa

Dadas las restricciones que se imponen al realizar la búsqueda de WiFi y el propósito de la aplicación que es permitir hacer un seguimiento detallado del usuario en el sistema, existe un conflicto para garantizar ambas. Cuando la aplicación deja de estar visible entra en segundo plano que restringe notablemente la frecuencia de las búsquedas, además, existe otra restricción que es cuando el dispositivo entra en modo descanso.

El modo descanso [17] es cuando el dispositivo entra en modo inactivo, es decir, cuando el usuario no ha hecho uso de este por un tiempo, está con la pantalla apagada y no está conectado al cargador. En este modo se restringen casi la totalidad de las funcionalidades de las aplicaciones y del equipo, los procesos pendientes de las aplicaciones se aplazan para después en una ventana de mantenimiento, como se muestra en la imagen 13. Esta funcionalidad fue agregada desde la versión de Android Marshmallow o 6.

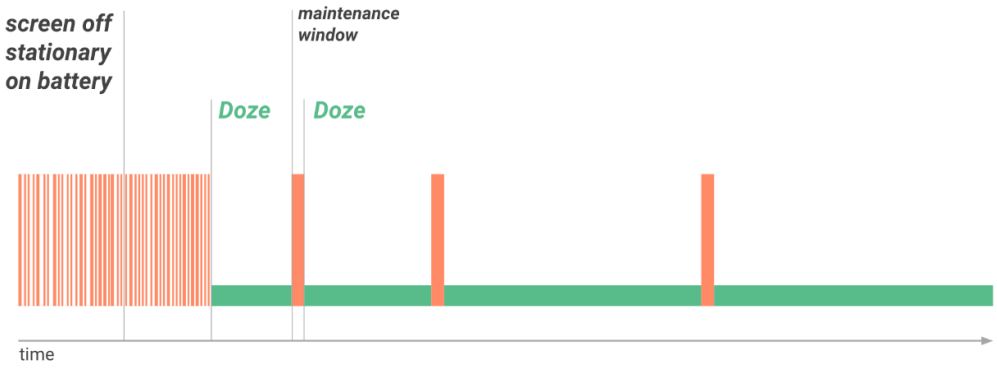


Imagen 13. Representación modo descanso dispositivos móviles. [17]

Las restricciones impuestas al estar en modo descanso se muestran en la imagen 14, donde la más negativa para este trabajo es la imposibilidad de realizar búsquedas de WiFi.

Restricciones del modo Descanso

Durante el modo Descanso, se aplican las siguientes restricciones a tus apps:

- Se suspende el acceso a la red.
- El sistema ignora los bloqueos de activación.
- Las alarmas `AlarmManager` estándar (como `setExact()` y `setWindow()`) se difieren al siguiente período de mantenimiento.
  - Si necesitas configurar alarmas que se activen en Descanso, usa `setAndAllowWhileIdle()` o `setExactAndAllowWhileIdle()`.
  - Las alarmas configuradas con `setAlarmClock()` se activan de forma normal (el sistema desactiva el modo Descanso momentos antes de que se activen las alarmas).
- El sistema no hace búsquedas de Wi-Fi.
- El sistema no permite que se ejecuten los adaptadores de sincronización.
- El sistema no permite que se ejecute `JobScheduler`.

Imagen 14. Restricciones modo descanso. [17]

Para concretar, una búsqueda repetitiva de WiFi se ve completamente afectada cuando la aplicación entra en segundo plano (limita la frecuencia) y cuando el dispositivo entra en modo descanso (imposibilita realizar búsquedas).

- Solución para Android 8 y posteriores.

Para evitar todas estas restricciones se desarrolló un servicio un primer plano [18] en el que la aplicación hace uso como si estuviera visible para el usuario sin estarlo, este evita que el dispositivo entre a segundo plano y al modo descanso, de este servicio el usuario está enterado mediante una notificación inamovible hasta que se detenga el servicio. Esta funcionalidad fue agregada desde Android 8.

Un ejemplo de este servicio en primer plano es una aplicación de reproducción de música, del que el usuario tiene conocimiento por la notificación y que permanece haciendo uso del equipo, aunque el dispositivo esté inactivo por un largo tiempo.

- Solución Android con versión menor a 8

En versiones anteriores a Android 8 no se encontró una solución que evite que la aplicación se vea afectada por el modo de descanso. La única manera fue prevenir que el dispositivo entre en ese estado, para esto se mantuvo al dispositivo con la pantalla encendida mediante la aplicación, de esta forma, la aplicación no ve restricciones al hacer las búsquedas de WiFi, pero si hace uso excesivo del recurso de la batería.

- Permisos de la aplicación [19]

Un tema importante son los permisos que usa la aplicación, estos se clasifican en normales y riesgosos, para hacer uso de estos últimos se debe solicitar al usuario que los acepte. En la siguiente tabla se muestran los permisos solicitados, el tipo y su funcionalidad en la aplicación [20].

Permiso	Tipo	Funcionalidad	Funcionalidad en la App
ACCESS_FINE_LOCATION	Riesgoso	Geovallado – Búsqueda de WiFi	Permite conocer la ubicación de forma precisa.
ACCESS_BACKGROUND_LOCATION	Riesgoso	Geovallado – Búsqueda de WiFi	Permite acceder a la ubicación en segundo plano, únicamente para Android 10 y posteriores.
ACCESS_WIFI_STATE	Normal	Búsqueda de WiFi	Permite acceder a la información sobre las redes WiFi.
CHANGE_WIFI_STATE	Normal	Búsqueda de WiFi	Permite cambiar el estado de la conectividad WiFi.
WAKE_LOCK	Normal	Vinculación G-B	Evita que el procesador entre en suspensión o que la pantalla se atenué.
REQUEST_IGNORE_BATTERY_ OPTIMIZATIONS	Normal	Vinculación G-B	Permiso para olvidar la optimización de la batería a la aplicación.
FOREGROUND_SERVICE	Normal	Vinculación G-B	Permite mantener un servicio en primer plano, para Android 8 y posteriores.

Tabla 3. Permisos de la aplicación

La ubicación es el único permiso riesgoso, la aplicación debe solicitar autorización al usuario de su uso mediante un dialogo [21], como se muestra en la imagen 15. A la izquierda es Android 6 y a la derecha Android 10, que incluye el permiso de acceso a la ubicación en segundo plano cuando pide permitir todo el tiempo.

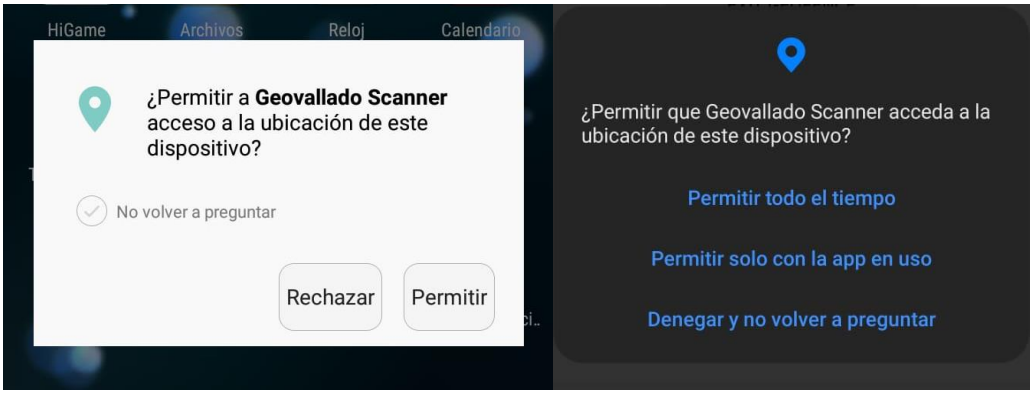


Imagen 15. Dialogo permiso de ubicación, Android 6 y 10.

- Solicitud de activación de la ubicación.

Tanto para el geovallado como para la búsqueda de WiFi en dispositivos Android 9 y superiores, se requiere que el dispositivo tenga la opción de localización activa, por lo que se debe corroborar en la aplicación esta opción, sino está activa la aplicación lanza el siguiente dialogo como se ve en la imagen 16, solicitando la activación del sensor, al dar aceptar esta opción se activa. Como se ve, un dispositivo Android 10 (derecha) hace uso de más sensores y alternativas como la ubicación por redes WiFi para obtener la ubicación, por lo que su precisión mejora.

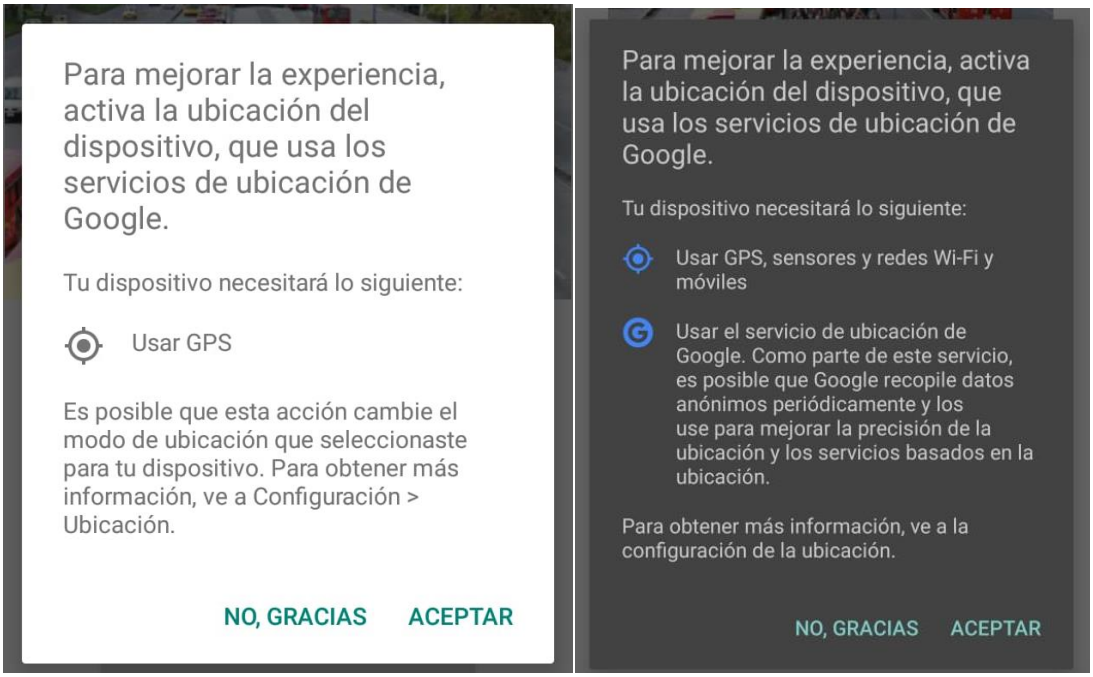


Imagen 16. Solicitud activación de la ubicación

- Diagrama de flujo aplicación.

A continuación, se muestra el diagrama de flujo del procedimiento en general que realiza la aplicación cuando el usuario la abre en la imagen 17, acá ya se contempla la aceptación del permiso de la ubicación, mostrado en la imagen 15. En la parte final del diagrama, al recibir la activación de salida del geovallado, se corrobora los resultados de las búsquedas de WiFi, y al no recibir un nombre o SSID de una red conocida en tres búsquedas se para la búsqueda repetitiva, esto para garantizar que siga funcionando si ha salido del geovallado, pero sigue recibiendo una red que podría ser en un bus del sistema.

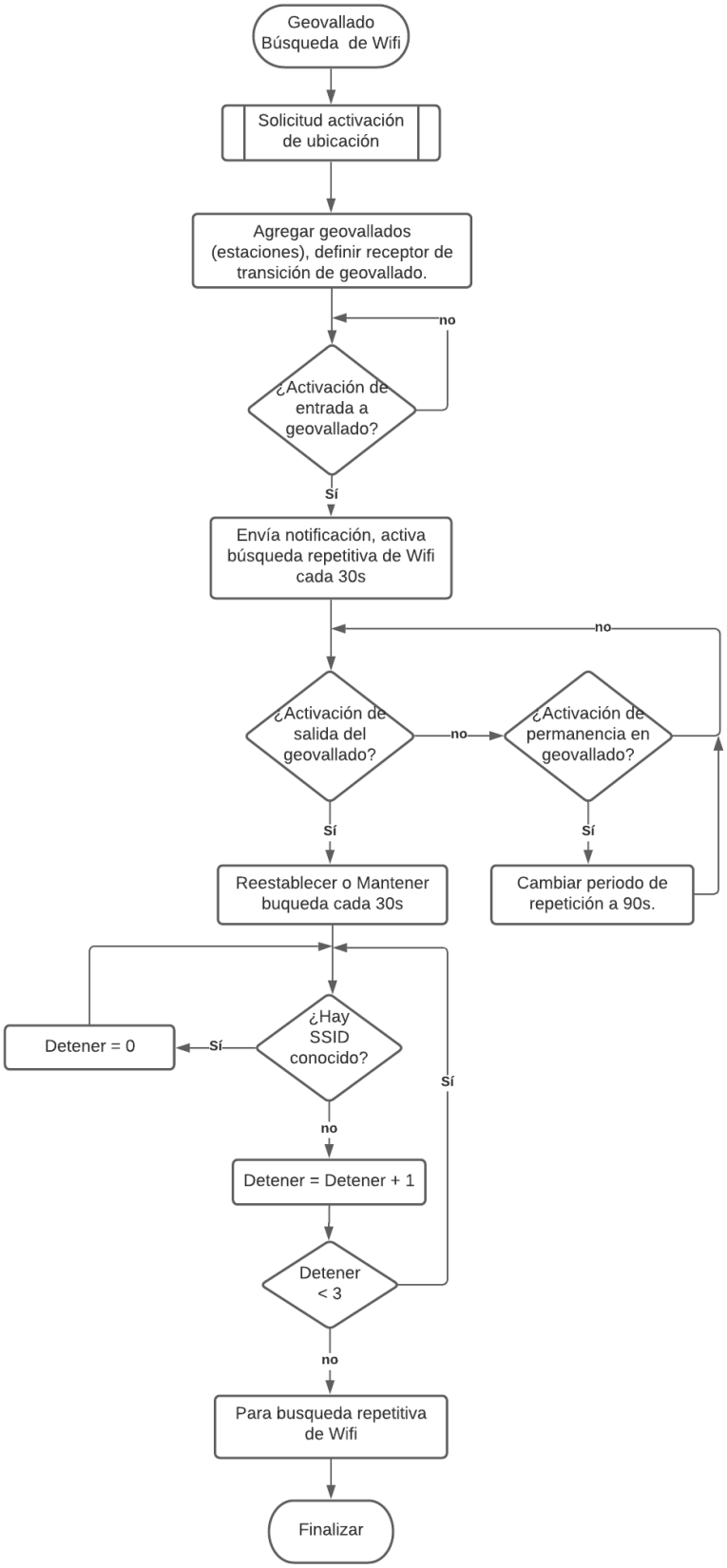
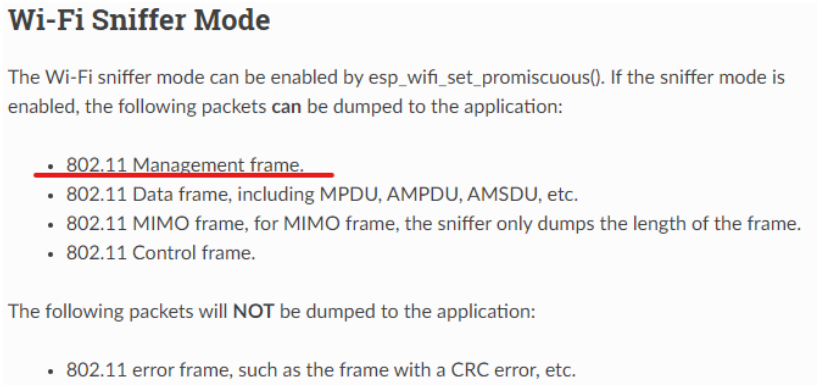


Imagen 17. Diagrama de fuljo aplicación.

4.3 Captador de paquetes *Probe Request*

Para esta funcionalidad fue utilizada la tarjeta de desarrollo ESP32 [22] del fabricante Espressif, conocida por ser ampliamente utilizada en aplicaciones IoT gracias a sus módulos de WiFi y Bluetooth. Para su programación se usó el software oficial de Espressif llamado ESP-IDF [23] y el entorno de desarrollo Eclipse [24], el código se compiló a una tasa de baudios 115200. Ver código de la ESP32 en anexo 2.

La funcionalidad principal es la configuración como modo WiFi sniffer [25] o también llamado modo promiscuo, que permite captar los paquetes que se mueven inalámbricamente con el protocolo IEEE802.11 b/g/n, que son los soportados por la ESP32 [26]. Los tipos de paquetes que pueden ser captados son los mostrados en la imagen 18.



Wi-Fi Sniffer Mode

The Wi-Fi sniffer mode can be enabled by esp_wifi_set_promiscuous(). If the sniffer mode is enabled, the following packets **can** be dumped to the application:

- 802.11 Management frame.
- 802.11 Data frame, including MPDU, AMPDU, AMSDU, etc.
- 802.11 MIMO frame, for MIMO frame, the sniffer only dumps the length of the frame.
- 802.11 Control frame.

The following packets will **NOT** be dumped to the application:

- 802.11 error frame, such as the frame with a CRC error, etc.

Imagen 18. Tipos de paquete WiFi Sniffer

El modo promiscuo realiza un barrido por cada canal inalámbrico (1-13) cada 240 ms, demorándose un total de 3.12 segundos en todos los canales, esto según la programación. Se filtraron los paquetes para captar únicamente los de tipo MGMT y de subtipo *Probe Request* enviados al hacer una búsqueda de WiFi activa, y que hará la aplicación móvil de forma repetitiva. A los paquetes resultantes se le imprime la dirección MAC de origen y el valor de RSSI captado, esto será recibido y tratado en la siguiente sección de tratamiento. Este procedimiento se puede ver en el diagrama de la imagen 19.

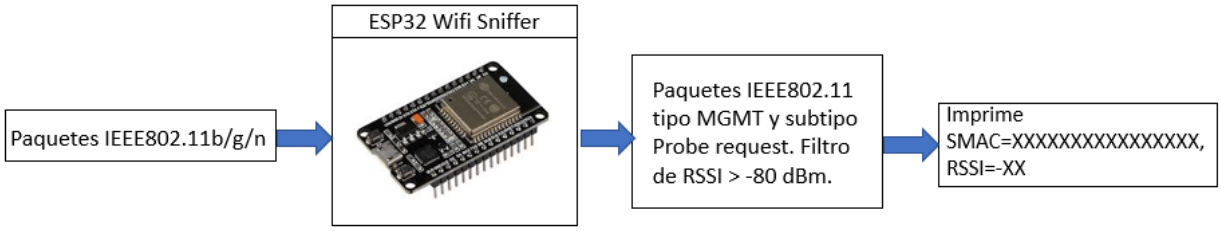


Imagen 19. Diagrama de bloques captador de paquetes Probe Request

Con algunas pruebas hechas se observó que el WiFi sniffer alcanza a captar paquetes con valores de RSSI menores a -90 dBm, valores demasiado bajos que cubren una distancia amplia que superan los 20 metros. Con el fin de obtener solo los paquetes con una cobertura limitada aproximada de 10 metros, se filtró el valor de potencia de RSSI con un máximo de -80 dBm, como se ve en el gráfico de la imagen 20 que presenta la medición de la potencia registrada a partir de la distancia en un lugar cerrado, tomada del trabajo de grado “Dispositivo anti pérdida de objetos personales”[27]. Esta limitación de cobertura también se hace en el caso de que se usen varias ESP32 para cubrir un solo sitio con varias zonas como la separación en una estación de Transmilenio.

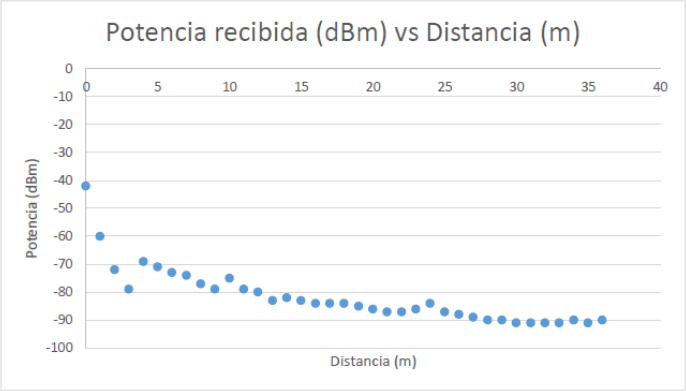


Imagen 20. Gráfico Potencia Vs Distancia, para filtrar cobertura.

A continuación, se muestra la información de interés que imprime la ESP32 para cada paquete filtrado, estas capturas son mostradas con mensajes seguidos o ráfagas para una misma dirección MAC, esto porque son enviados varios mensajes *Probe Request* al hacer la búsqueda de WiFi activa, en las pruebas hechas se pudo identificar la captación promedio de entre 4 a 8 paquetes para cada búsqueda. Finalmente, cuando ya se ha cargado el código en la tarjeta, únicamente necesita alimentación y lo hará indefinidamente.

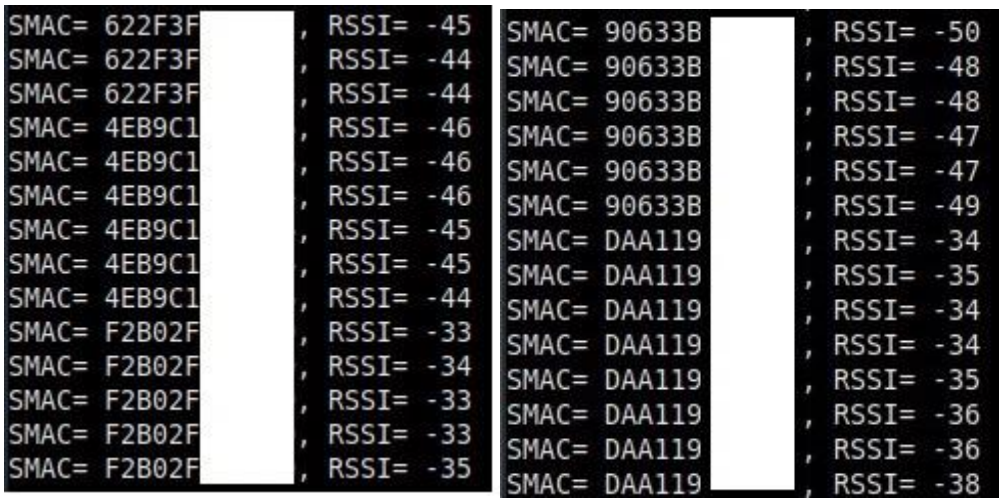


Imagen 21. Capturas resultantes captador de paquetes Probe Request

Por otra parte, la ESP32 también fue configurada como punto de acceso, aunque sea una red que no cuenta con internet, los dispositivos móviles si reciben su información al hacer las búsquedas de WiFi. Con esto, la aplicación móvil en el resultado de las búsquedas puede comprobar si aparece algún SSID de estas redes.

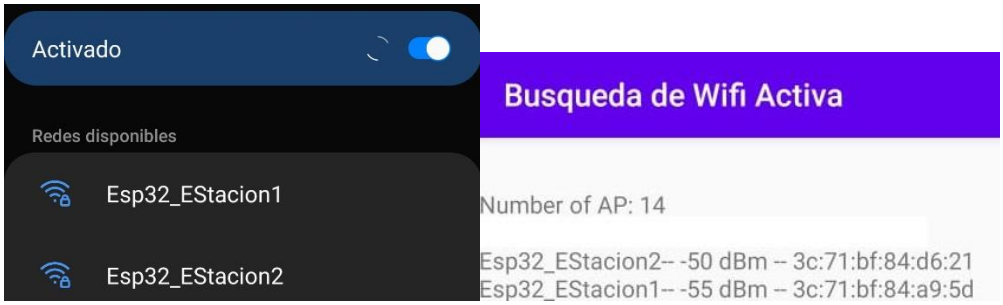


Imagen 22. Red ESP32 como punto de acceso vista en celular y aplicación.

4.4 Tratamiento y comunicación

4.4.1 Raspberry Pi 3 - Python

Esta sección de tratamiento se enfoca en la recepción de las capturas hechas por la ESP32 y con estas realizar la asignación de tiempos y duraciones, por cada dirección MAC. Además, la información se guarda en reportes locales y se envía en tiempo real a la base de datos del servidor de Firebase. Todo lo mencionado es realizado con un script de Python que se ejecuta cuando es encendida la Raspberry Pi. Ver código hecho en Python en anexo 3.

La primera parte es conectar la ESP32 que se hace directamente a la Raspberry por medio de un cable USB, para recibir cada captura impresa por la ESP32 se utilizó el módulo Pyserial [28] en Python, definiendo el puerto y la tasa de baudios a la que imprime la ESP32 (115200). Con esto se leyó cada línea impresa, de esta se obtiene la dirección MAC, se asigna el tiempo de captura, valores que se almacenan en listas. Cuando se recibe una dirección MAC ya guardada calcula la duración con el mensaje anterior y la duración total desde el mensaje inicial. También, para garantizar que los mensajes de las ráfagas (imagen 21) hechos por una misma búsqueda de WiFi no se repitan, se evita obtener una misma dirección MAC de forma seguida.

Para el envío a Firebase se usó el módulo Pyrebase [29] en Python, que mediante una configuración se vincula a la base de datos en tiempo real, y permite enviar y actualizar valores en esta. En esta base de datos solo se almacenan los tiempos inicial y final de captura para cada dirección MAC, el tiempo final es actualizado cuando llega otra captura de esta misma dirección MAC. Otro aspecto es asegurar que únicamente se presenten las direcciones MAC no aleatorias, para esto se empieza a enviar la información de los tiempos a Firebase cuando se han recibido dos capturas de una misma dirección MAC.

A continuación, se muestra el diagrama de máquinas de estado que representa el código hecho en Python para esta sección de tratamiento.

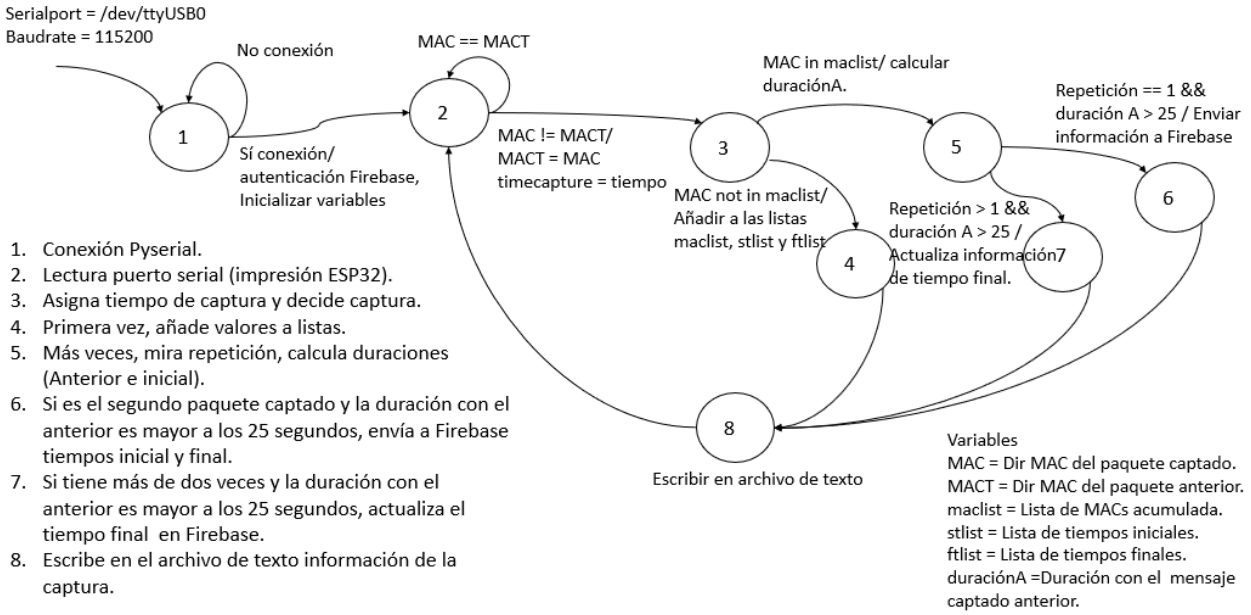


Imagen 23. Diagrama de máquinas de estado sección tratamiento

El script de Python se ejecuta al iniciar la Raspberry Pi, esto se realiza gracias a una instrucción definida, configurándola por medio del comando Crontab en la línea de comandos de Linux de la siguiente manera [30].

1. Abrir editor de Crontab.

```
pi@raspberrypi:~ $ crontab -e
```

2. Escribir instrucción, para este caso se configuró que el script iniciara 20 segundos después del reinicio, esto para garantizar que la Raspberry ya haya hecho las configuraciones como el acceso a internet. Se puede configurar la ejecución de varios scripts.

```
@reboot sleep 20; python3 /home/pi/Desktop/stuTM/ESP32_Serial0.py
#@reboot sleep 20; python3 /home/pi/Desktop/stuTM/ESP32_Serial1.py
```

De esta forma solo es necesario alimentar la Raspberry, encenderla y garantizar que tenga acceso a internet, automáticamente se ejecuta el script y realiza el proceso definido.

Una sola Raspberry puede estar conectada a varias tarjetas ESP32, corriendo un script para cada puerto de conexión a Pyserial.

4.4.2 TAR SIM800 RPI

Transmilenio cuenta con la totalidad de las 147 estaciones con acceso a internet [31]. También, en una noticia reciente, se sabe de la inclusión de red WiFi en los nuevos buses adquiridos para el servicio zonal del sistema SITP [32]. Gracias a esto, personal del sistema y los usuarios tienen acceso gratis a internet en la infraestructura, también, este sistema piloto diseñado se podría conectar a la red de cada estación o bus mediante cable físico o de forma inalámbrica y utilizarla para el envío de datos al servidor de Firebase.

Sin embargo, este sistema piloto también busca garantizar el acceso a internet sin la dependencia de una red externa, por tal motivo se optó por el uso de la tarjeta TAR SIM800 RPI vista en la imagen 24, que es una expansión diseñada para la Raspberry Pi, esta permite la conexión mediante el uso de datos con una tarjeta SIM. Las especificaciones de la tarjeta también se muestran a continuación [33].

Imagen 24. Tarjeta TAR SIM800 RPI y especificaciones

Para realizar la conexión con el módulo GSM/GPRS se realiza una configuración tipo ppp (Point to Point Protocol) que la Raspberry define como la predeterminada, se realizó siguiendo este tutorial [34]. Importante digitar el APN (*Acces Point Name*) correspondiente a la tarjeta SIM.

4.5 Almacenamiento

Como se mencionó, desde la Raspberry Pi se trata la información de las capturas y se almacena de dos formas distintas, la primera es un almacenamiento local en archivos de texto y la segunda es en la base de datos en tiempo real del servidor de Firebase, ambos almacenamientos ofrecen información de utilidad, el archivo de texto más completo con todas las capturas tomadas y en la base de datos en Firebase con únicamente la información de los tiempos, pero con la ventaja de actualizarse en tiempo real y poder acceder a la información de forma remota.

Un archivo de texto y una rama en Firebase es creado cada 15 minutos nombrados con su tiempo de creación, esto para tener la información capturada seccionada en por el tiempo y evitar la congestión de las variables en el script de Python que se mantiene corriendo.

4.5.1 Archivo de texto

De forma local en la Raspberry Pi se guardan las capturas registradas después de pasar el tratamiento hecho, para cada una de estas resultantes se escribe en el archivo de texto; el tiempo de captura, la dirección MAC, el valor de RSSI, el número de repetición o aparición y las duraciones con el mensaje anterior e inicial en segundos (para la segunda captura en adelante), todos estos separados por coma para su posterior análisis en la herramienta de Excel. La forma de guardado en el archivo de texto se ve en la imagen 25.

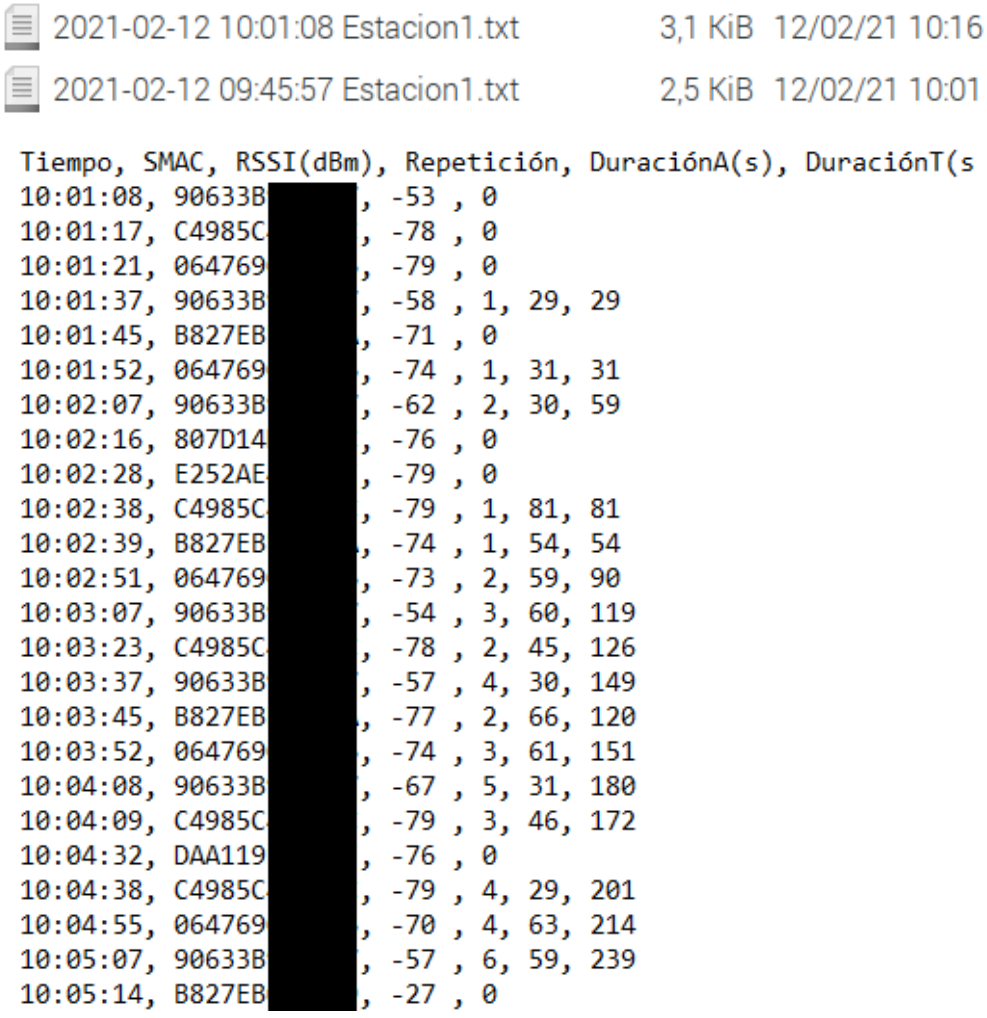


Imagen 25. Almacenamiento en archivo de texto.

Como estos archivos se almacenan de forma local, para acceder a estos es necesario ingresar a la Raspberry Pi y pasarlos al ordenador para su posterior análisis.

4.5.2 Firebase

Este almacenamiento se realizó en la base de datos en tiempo real del proyecto en Firebase [35] a la que se puede acceder a su información de forma remota, esta base maneja el formato JSON caracterizada por ser fácil de leer y de manipular, su forma se asemeja a la de árbol (raíces y ramas). Los únicos valores almacenados de las capturas en la base son los tiempos de captura inicial y final para cada dirección MAC que se haya recibido mínimo dos veces en la sección de tratamiento, junto a la duración registrada para dispositivos que duren 3 minutos (180s) en adelante, pensado como una alerta por demora.

Los métodos usados para interactuar con la base de datos usados en este trabajo fueron los siguientes, mencionados en la documentación de Pyrebase [29].

- Child: crear una ruta o rama en el árbol. Las ramas principales son las de fecha, nombre de estación y tiempo (cada 15 minutos), estas almacenan las ramas secundarias que son las direcciones MAC.
- Set: añade los valores de tiempos inicial y final de captura a las ramas secundarias de direcciones MAC.
- Update: actualiza campo existente en la base de datos, usado para actualizar el tiempo final de captura en las ramas de dirección MAC.

En la imagen 26 se muestra la forma de almacenamiento en Firebase. Para su posterior análisis se colocaron prefijos como ‘D’ para la fecha y ‘H’ para la hora y no tener inconvenientes al ser importados en la herramienta de Matlab.

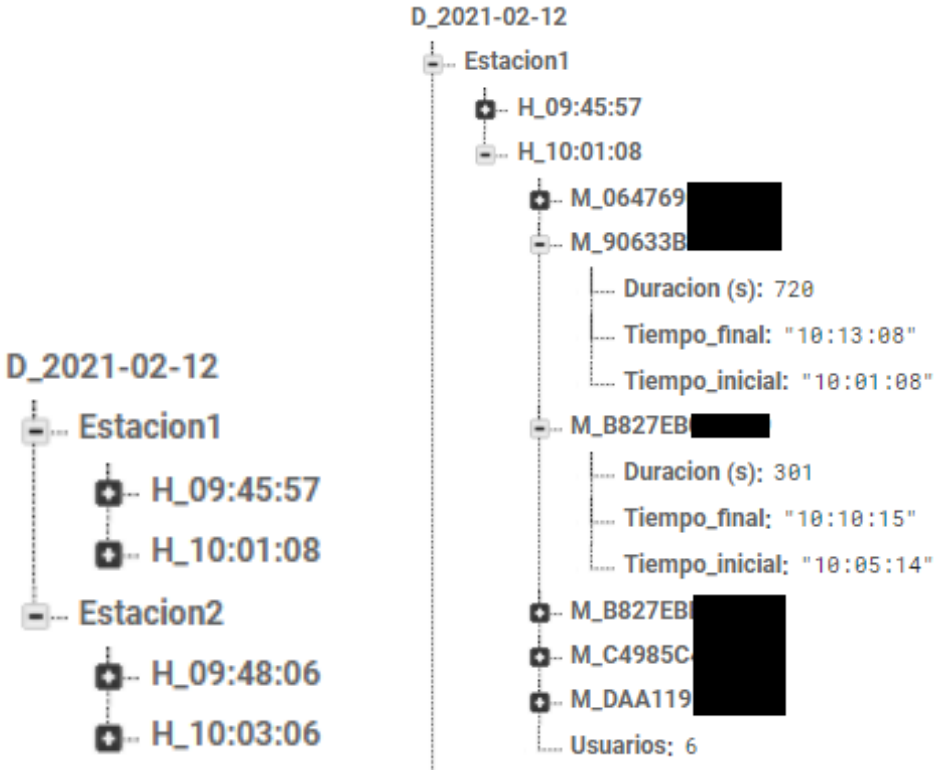


Imagen 26. Almacenamiento en Firebase.

4.6 Análisis

Mediante las formas de almacenamiento ya explicadas se puede realizar esta sección de análisis, ambas formas de almacenamiento permiten obtener información útil para observar y detallar el comportamiento del sistema piloto, y presentar la información objetivo de este trabajo de grado que ayudará en la planificación de un sistema de transporte masivo como Transmilenio. Este análisis fue realizado con las herramientas de Excel y Matlab. Las imágenes mostradas en todo este apartado presentan un ejemplo de este proceso de análisis, mas no resultados de pruebas hechas. Esto se muestra en la sección de protocolo de pruebas.

4.6.1 Análisis en Excel

La funcionalidad principal es examinar los archivos de texto guardados de forma local en la Raspberry Pi y con estas determinar el comportamiento del sistema, lo que tiene que ver con las repeticiones y las duraciones registradas entre las capturas para cada dispositivo probado. También, obtener información de la cantidad de direcciones MAC que fueron captadas e identificar las que posiblemente implementen aleatorización de esta. A continuación, se muestra el resultado del archivo filtrado para un solo dispositivo.

Tiempo	SMAC	RSSI(dBm)	Repetición	DuraciónA(s)	DuraciónT(s)
10:01:08	90633B[redacted]	-53	0		
10:01:37	90633B[redacted]	-58	1	29	29
10:02:07	90633B[redacted]	-62	2	30	59
10:03:07	90633B[redacted]	-54	3	60	119
10:03:37	90633B[redacted]	-57	4	30	149
10:04:08	90633B[redacted]	-67	5	31	180
10:05:07	90633B[redacted]	-57	6	59	239
10:05:37	90633B[redacted]	-57	7	30	269
10:07:37	90633B[redacted]	-56	8	120	389
10:08:07	90633B[redacted]	-60	9	30	419
10:08:37	90633B[redacted]	-57	10	30	449
10:09:37	90633B[redacted]	-55	11	60	509
10:10:07	90633B[redacted]	-67	12	30	539
10:10:37	90633B[redacted]	-58	13	30	569
10:11:07	90633B[redacted]	-56	14	30	599
10:11:37	90633B[redacted]	-54	15	30	629
10:12:07	90633B[redacted]	-66	16	30	659
10:12:37	90633B[redacted]	-59	17	30	689
10:13:08	90633B[redacted]	-57	18	31	720
10:14:07	90633B[redacted]	-58	19	59	779
10:14:37	90633B[redacted]	-66	20	30	809
10:15:08	90633B[redacted]	-55	21	31	840
10:16:07	90633B[redacted]	-58	22	59	899

Imagen 27. Análisis en Excel

Con la información de cada dispositivo probado se observa la diferencia entre las duraciones registradas para las capturas, que se espera sea de 30 segundos, si logra captar la totalidad de paquetes enviados por la aplicación móvil. Esto permite saber el comportamiento del sistema completo, pero principalmente de la ESP32 como captador de paquetes, factores como la cantidad de dispositivos en el área influyen en esta medición. A continuación, se muestra el grafico correspondiente al archivo filtrado de la imagen 27.

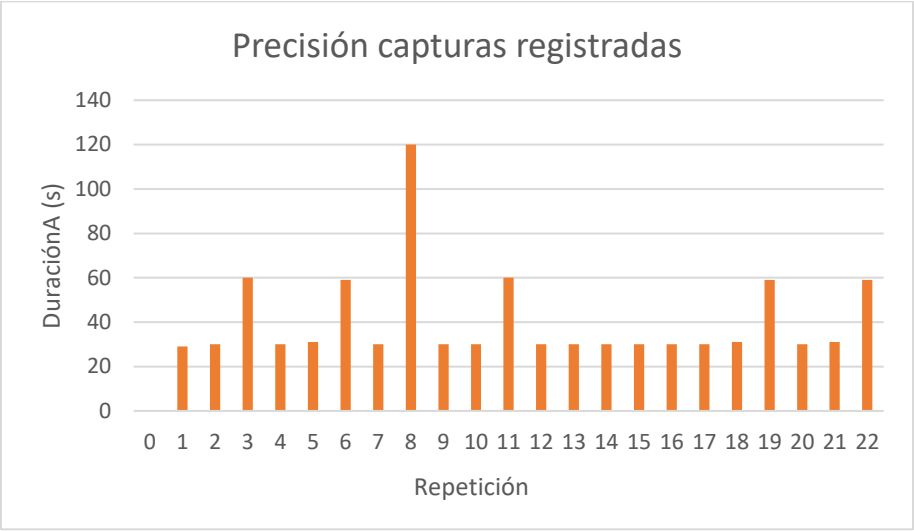


Imagen 28. Precisión capturas registradas captador de paquetes Probe Request.

Otro aspecto de estos archivos de texto es que para cada captura almacena el valor de RSSI obtenido, esto permite hacer el estimado de la distancia a la que cada dispositivo móvil estuvo del sistema piloto en el tiempo que permaneció en la estación y en mi opinión para un posterior análisis serviría para determinar el número de personas por metro cuadrado ya sea en una estación o bus. Se muestra esta información adicional en forma gráfica en la imagen 29, se aclara que en este trabajo de grado no hubo enfoque en esta medida, pero si se destaca su posible uso como información adicional y útil para analizar del sistema piloto.

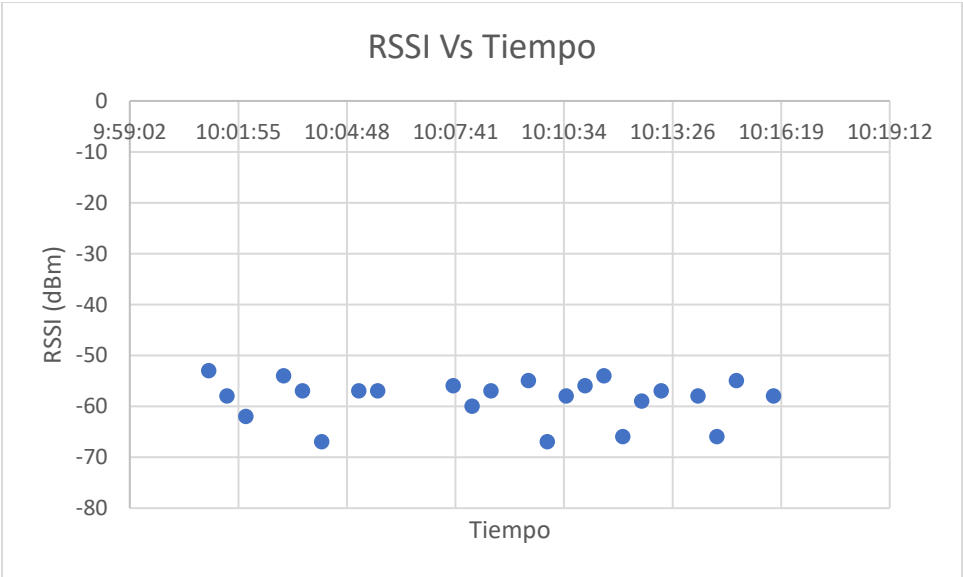


Imagen 29. Medida valores RSSI captados en el tiempo.

4.6.2 Análisis en Matlab

Este programa se utilizó para examinar el almacenamiento en la base de datos en Firebase, descargando y usando el archivo con formato JSON se realizó el código que obtiene esta información y la presenta de forma organizada mediante tablas gráficas. Este análisis en Matlab solo considera reportes tomados diarios, no para varios días. Ver código en anexo 4.

La información útil que se presenta es la que beneficiará a un sistema de transporte masivo de pasajeros, tal como; comportamiento de usuarios en el tiempo (entradas y salidas), duración promedio de los usuarios y el objetivo principal que es la trazabilidad de los usuarios al paso de una estación a otra.

Para cada rama de Firebase (hecha cada 15 minutos) brinda la siguiente información organizada de los tiempos por dirección MAC junto a la duración en la imagen 30. Con esto se puede obtener la información del comportamiento de entradas y salidas en este intervalo imagen 31.

	1 Dir_MAC	2 Tiempos		3 Duracion
1	"90633B[redacted]	10:01:08	10:16:07	0h 14m 59s
2	"C4985C[redacted]"	10:01:17	10:15:51	0h 14m 34s
3	"064769[redacted]	10:01:21	10:16:06	0h 14m 45s
4	"B827EB[redacted]"	10:01:45	10:05:44	0h 3m 59s
5	"DAA11[redacted]"	10:04:32	10:15:26	0h 10m 54s
6	"B827EB[redacted]	10:05:14	10:15:15	0h 10m 1s

Imagen 30. Organización en tabla de reporte de Firebase (15 minutos).

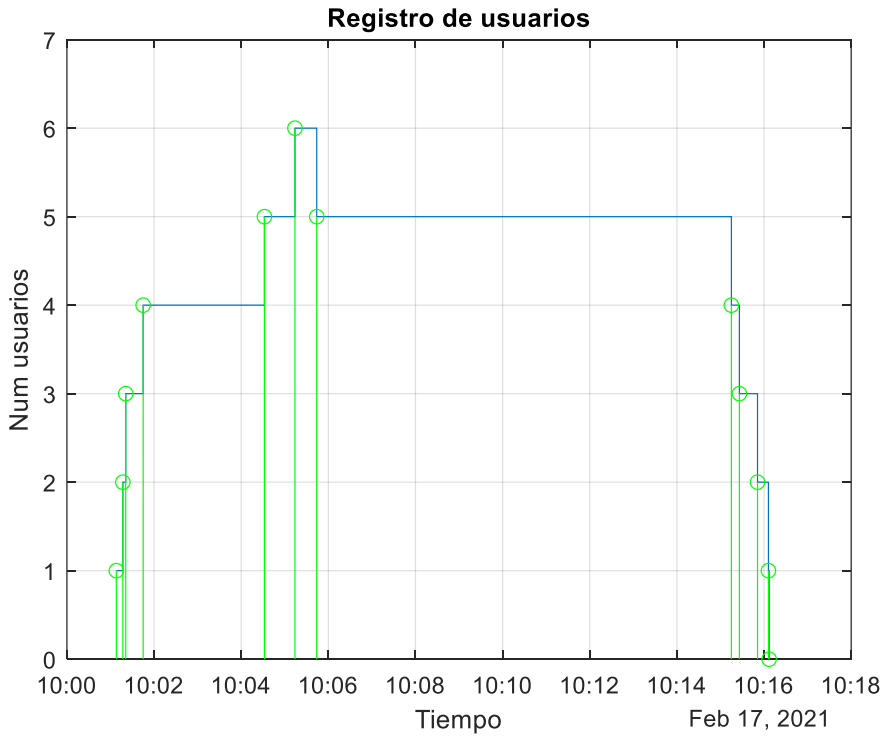


Imagen 31. Comportamiento de entradas y salidas reporte de Firebase (15 minutos).

Para todos los registros de las ramas brinda la información de la cantidad de usuarios registrados (mínimo dos paquetes captados) y la duración promedio de permanencia en minutos de los usuarios en ese intervalo de tiempo.

	1 Hora_captura	2 Usuarios	3 Duracion_promedio
1	09:45:57	6	10.6667
2	10:01:08	6	11.5333
3	10:16:13	6	10.3111
4	10:31:15	5	6.1167

Imagen 32. Tabla cantidad de usuarios y su duración promedio para cada reporte de Firebase (15 minutos).

En la imagen 33, se puede ver gráficamente la información de la tabla de la imagen 32, que muestra el comportamiento del número de usuarios en la estación y su duración en cada intervalo de tiempo. Esto se interpreta para conocer las condiciones en cada momento en la estación, saber la cantidad de usuarios que están esperando un servicio o ruta y el tiempo de espera gastado para abordarlo, pensado para Transmilenio.

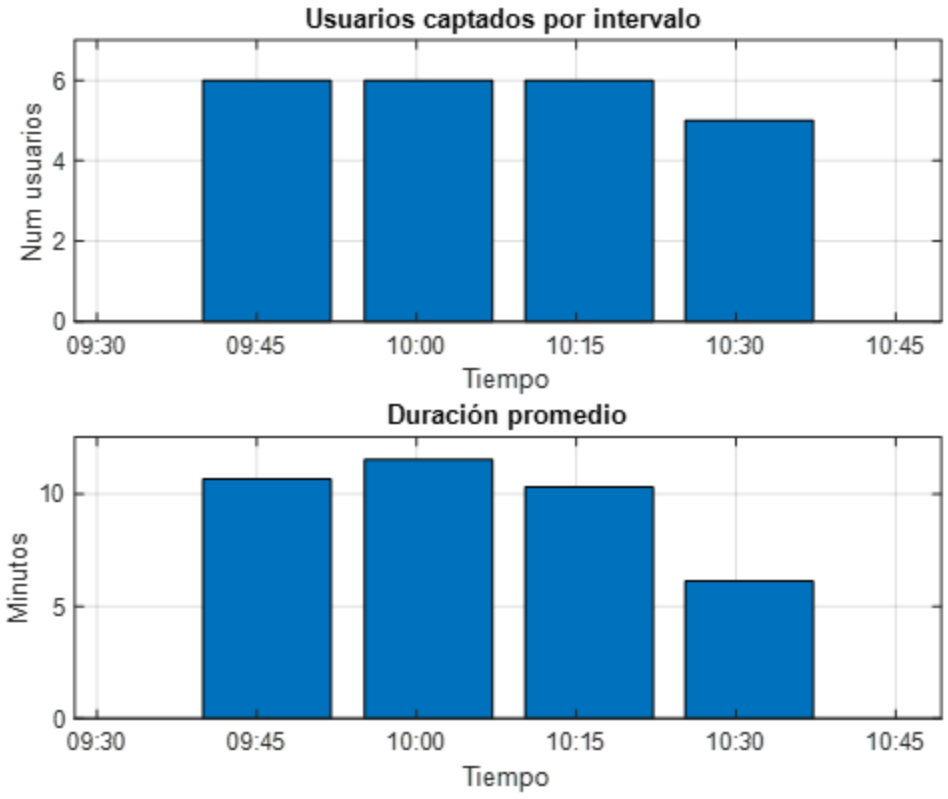


Imagen 33. Gráfico usuarios por intervalo y duración promedio para cada reporte de Firebase (15 minutos).

También, para todos los registros de las ramas se realiza un registro completo y en este identifica si algún usuario (dirección MAC) aparece más de una vez en todos los registros, y organiza si hace parte de una aparición anterior o de otra distinta, revisando si la diferencia entre los registros es mayor a cinco minutos, valor escogido para efectos de mostrar en este apartado. Esto permite saber si un usuario ingresa en distintos

horarios a una misma estación, como se muestra en la imagen 34 en las filas 4 hasta la 8, en las que dos dispositivos estuvieron en distintos tiempos y en la columna 5 en la sección de aparición se muestra el número de repetición.

	1 Dir_MAC	2 Tiempos		3 Duracion	4 Duracion_minutos	5 Aparicion
1	"C4985C4[redacted]	09:46:07	10:29:50	0h 43m 43s	43.7167	1
2	"B827EBB[redacted]	09:47:38	10:05:44	0h 18m 6s	18.1000	1
3	"064769C[redacted]	09:47:44	10:40:51	0h 53m 7s	53.1167	1
4	"DAA119[redacted]	09:48:28	10:15:26	0h 26m 58s	26.9667	1
5	"DAA119[redacted]	10:22:06	10:37:43	0h 15m 37s	15.6167	2
6	"B827EB0[redacted]	09:50:14	10:00:14	0h 10m 0s	10	1
7	"B827EB0[redacted]	10:05:14	10:15:15	0h 10m 1s	10.0167	2
8	"B827EB0[redacted]	10:20:15	10:40:52	0h 20m 37s	20.6167	3
9	"90633B9[redacted]	09:58:38	10:37:41	0h 39m 3s	39.0500	1
10	"1CBFC0[redacted]	10:25:38	10:30:27	0h 4m 49s	4.8167	1
11	"B4BFF6F[redacted]	10:34:47	10:37:22	0h 2m 35s	2.5833	1

Imagen 34. Tabla organizada de todos los reportes de Firebase (15 minutos) y apariciones.

Al analizar para dos estaciones, se comprueba la aparición de un mismo usuario (dirección MAC) en los reportes de la imagen 34 de cada estación. Organizando una tabla de la siguiente forma que muestra los tiempos de entrada y salida en cada estación junto a sus duraciones en estas. Con esto se determina que usuarios realizaron un trayecto entre las dos estaciones.

	1 Dir_MAC	2 Tiempos_Estacion1		3 Tiempos_Estacion2		4 Duracion_Estacion1	5 Duracion_Estacion2
1	"C4985[redacted]	09:46:07	10:29:50	09:48:20	10:05:53	0h 43m 43s	0h 17m 33s
2	"B827EB[redacted]	09:47:38	10:05:44	09:48:44	10:05:44	0h 18m 6s	0h 17m 0s
3	"064769[redacted]	09:47:44	10:40:51	09:48:14	10:04:55	0h 53m 7s	0h 16m 41s
4	"DAA11[redacted]	09:48:28	10:15:26	09:49:36	10:05:59	0h 26m 58s	0h 16m 23s
5	"B827EB[redacted]	09:50:14	10:00:14	09:50:14	10:00:14	0h 10m 0s	0h 10m 0s
6	"90633B[redacted]	09:58:38	10:37:41	09:49:34	09:58:38	0h 39m 3s	0h 9m 4s
7	"1CBFC[redacted]	10:25:38	10:30:27	09:57:09	09:59:35	0h 4m 49s	0h 2m 26s
8	"B4BFF6[redacted]	10:34:47	10:37:22	10:34:47	10:37:22	0h 2m 35s	0h 2m 35s

Imagen 35. Tabla usuarios con aparición en las dos estaciones (trazabilidad).

Para cada una de las direcciones MAC que se encuentre en el reporte de la imagen 35, se grafica su trayecto o trazabilidad por las estaciones de la siguiente forma en la imagen 36.

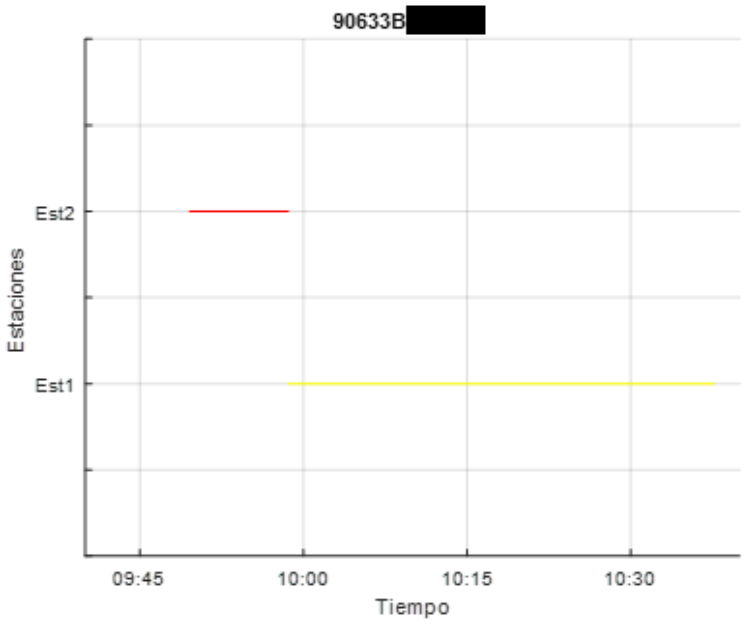


Imagen 36. Gráfico trazabilidad de usuario en su paso de una estación a otra.

5. Protocolo de pruebas

Se prueba el sistema piloto elaborado en las condiciones actuales de la pandemia del Covid-19 y las restricciones impuestas, se emuló una estación de Transmilenio en un salón del edificio de laboratorios de ingeniería, un espacio cerrado sin la presencia de otros equipos y sin el flujo de personas con sus dispositivos celulares común en una estación de Transmilenio real, por esto el sistema piloto fue probado únicamente con una sola ESP32 captando los mensajes y conectada en la Raspberry Pi. En la imagen 37 se muestra el sistema instalado, de igual forma que en el diagrama de bloques mostrado en la imagen 8 cuenta con los siguientes componentes:

- Tarjeta Esp32
- Raspberry Pi3
- Módulo GSM800 RP con tarjeta SIM con datos.
- Cable de alimentación y cable USB.

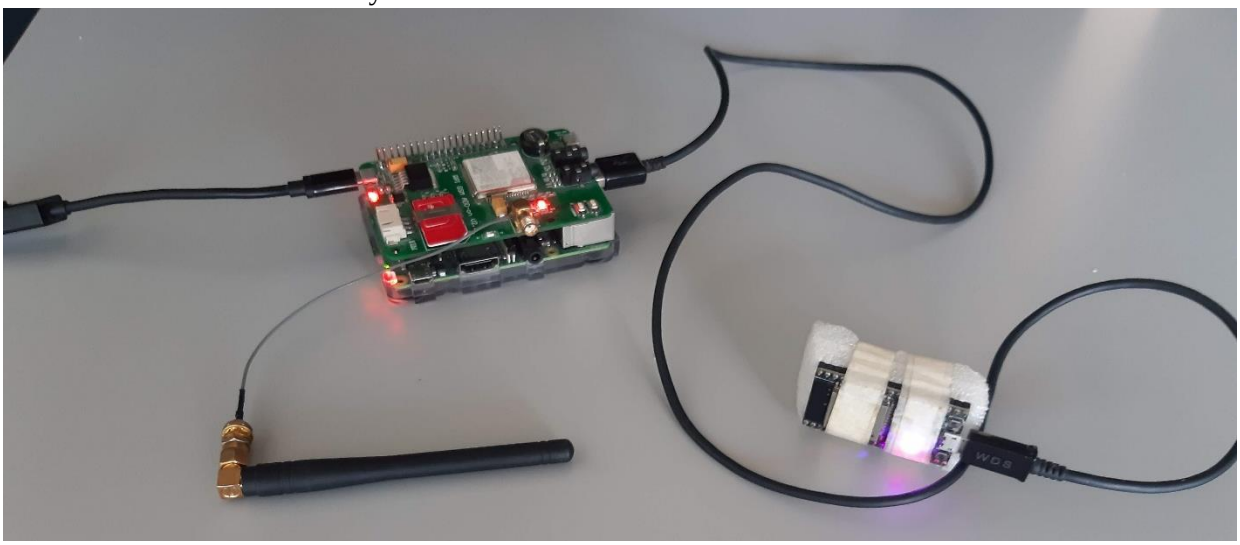


Imagen 37. Instalación sistema piloto

Los dispositivos celulares en los que se instaló la aplicación móvil "Geovallado – Búsqueda de WiFi" y que participaron en las pruebas varían en las siguientes versiones de sistema operativo de Android, con las restricciones mencionadas en el apartado de aleatorización MAC del marco teórico y en la sección de vinculación de geovallado - búsqueda de WiFi de la aplicación.

Dispositivo	Referencia	Versión de Android	Condiciones de Prueba
1	Huawei MYA-L03	6	Sin conexión a WiFi, dispositivo permanece con pantalla encendida pero atenuada. Ambos dispositivos no realizan aleatorización MAC. Aunque el Samsung J2 tiene Android mayor a 8, se identificó que no implementa la aleatorización.
2	Samsung J2	8.1.0	
3	Samsung A30	10	Únicamente con conexión WiFi, fijando dirección MAC del equipo a la red. Dispositivo permanece con pantalla apagada.

Tabla 4. Celulares con Android probados.

5.1 Pruebas comportamiento del sistema piloto

Se busca saber el rendimiento del sistema piloto en tres pruebas específicas; la primera se basa en la precisión de las activaciones de geovallado, la segunda es determinar la duración mínima en la que una estancia ha sido detectada por el sistema y por último conocer el periodo de captación de paquetes.

1. **Precisión activaciones del geovallado;** esta prueba permite saber el retardo presentado cuando el usuario (dispositivo celular con la aplicación) entra, permanece y sale del radio del geovallado, este retardo es importante porque con este se determina la precisión al iniciar, variar el periodo y parar la búsqueda repetitiva de WiFi activa.

2. **Duración mínima de estancia;** para conocer esta estancia el sistema debe captar mínimo dos paquetes de una misma dirección MAC que son enviados por la aplicación, registrarlos y subirlos a la base de datos de Firebase.

3. **Periodo de captación;** con el archivo de texto se observa el número de paquetes captados y la diferencia de tiempos entre las capturas, también se observa la actualización en tiempo real de los tiempos finales en la base de datos de Firebase. Esta prueba da la estimación de la diferencia promedio entre paquetes captados, que indicará si existe la continuidad esperada para trazar detalladamente a un usuario.

5.2 Pruebas para determinar la trazabilidad a varios usuarios

Asimismo, para cumplir con el objetivo principal planteado en este trabajo de grado, que busca conocer el trayecto realizado por varios usuarios de forma precisa, se colocaron dos sistemas pilotos cada uno representando una estación diferente, y haciendo que los dispositivos con la aplicación móvil pasen de una estación a otra. Contemplando dos casos que se pueden presentar respecto a la distancia entre las estaciones.

1. **Estaciones cercanas;** ambas estaciones en un mismo sitio, separadas por una distancia aproximada de 5 metros. Esta prueba se asemeja cuando los usuarios pasan de la estación a un bus, sino que este no se encuentra en movimiento. El procedimiento realizado es el siguiente, con un tiempo estipulado de 3 minutos considerando un menor tiempo gastado en el paso de la estación al bus.

a. El usuario entra al geovallado acercándose a la ubicación del primer sistema (estación), se activa la búsqueda repetitiva de WiFi permanece un tiempo de 3 minutos.

b. El usuario se dirige al punto donde se encuentra el otro sistema (bus) y también permanece un tiempo de 3 minutos.

c. El usuario sale del geovallado y se apaga la búsqueda repetitiva de WiFi.

2. **Estaciones con distancia media**; con una distancia de separación aproximada de 30 metros. Esta prueba pretende mostrar el comportamiento de paso de una estación a otra en el tiempo. El procedimiento realizado es el siguiente, con tiempos mayores considerando la distancia y la demora del paso de una estación a otra.

a. El usuario llega a la primera estación, hay activación por entrada a geovallado y empieza la búsqueda repetitiva de WiFi, permanece suficiente tiempo y sale de la estación.

b. El usuario hace el recorrido caminando para llegar a la segunda estación.

c. El usuario llega a la segunda estación, garantiza que este activa la búsqueda repetitiva de WiFi, permanece suficiente tiempo, sale de la estación y se apaga la búsqueda.

5.3 Resultado pruebas comportamiento del sistema piloto

1. **Precisión de activación del geovallado:** Como ya se mencionó, se busca conocer los retardos de estas activaciones de entrada, permanencia y salida de los geovallados, para que las acciones desarrolladas en cada activación se realicen de forma oportuna, de estos tres activadores el retardo a calcular con mayor importancia es el de entrada que inicia la búsqueda de WiFi con un periodo de 30 segundos.

    a. **Entrada;** cuando la aplicación se abre ya estando dentro de algún geovallado esta activación de entrada es precisa, casi al mismo tiempo de haber agregado los geovallados. Por otra parte, cuando la prueba es dirigiéndose desde afuera hacia adentro del geovallado, esta medida varía según el estado en el que se encuentra el dispositivo que puede estar normal o en modo descanso. Además, como el geovallado fue definido con un radio de 60 metros, habría que medir a que distancia se efectúa esta activación. Esta medida fue tomada con la apertura de la aplicación a una distancia aproximada de 100 metros, dejado el celular apagado y comparando el tiempo de apertura y de la notificación enviada por la aplicación.

Dispositivo	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Promedio
1 Android 6	13 s	24 s	90 s	18 s	36,25 s
2 Android 8.1.0	32 s	74 s	46 s	61 s	53,25 s
3 Android 10	26 s	135 s	81 s	57 s	74,75 s

Tabla 5. Retardo activación de entrada

    b. **Permanencia;** cuando se cumple el tiempo de permanencia definido en la programación para esta activación del geovallado (5 minutos), con esta activación se cambia el periodo de repetición de la búsqueda de WiFi de 30 a 90 segundos, fundamentado para el ahorro de batería en el dispositivo. Para saber este retardo se hizo una variación para que se informara de este cambio a permanencia por medio de otra notificación, esto permitió saber el retardo desde la activación de entrada.

Dispositivo	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Promedio
1 Android 6	5 min	6 min	6 min	6 min	5,75 min
2 Android 8.1.0	5 min	5 min	6 min	5 min	5,25 min
3 Android 10	5 min	7 min	8 min	9 min	7,25 min

Tabla 6. Retardo activación de permanencia

    c. **Salida;** de igual forma que para la entrada esta medición se complicó al no saber con exactitud la salida del radio del geovallado. El otro aspecto verificado es que se cumpla la terminación de la búsqueda repetitiva de WiFi cuando además de la salida no hay una red inalámbrica conocida en tres búsquedas (60 segundos). Por medio de un botón en la aplicación representando la salida y se midió el tiempo en el que se apaga el servicio, este tiempo varía porque no se precisa la activación por salida a la misma vez que hay una repetición de búsqueda.

Dispositivo	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Promedio
1 Android 6	53 s	57 s	62 s	50 s	55,5 s
2 Android 8.1.0	67 s	55 s	45 s	78 s	61,25 s
3 Android 10	75 s	73 s	63 s	74 s	71,25 s

Tabla 7. Retardo terminación búsqueda repetitiva de WiFi

Estos resultados de las activaciones del geovallado demuestran que se presentan retardos inciertos debidos a las restricciones para obtener la ubicación del dispositivo de la API de geovallado en la aplicación y su precisión depende de los sensores usados (GPS, WiFi). Estos retardos para este sistema piloto son relevantes, pero no decisivos, con el de entrada se puede estimar una permanencia mínima al activar la búsqueda repetitiva de WiFi, algo tenido en cuenta en las pruebas de trazabilidad.

2. **Duración mínima de estancia y periodo de captación;** para las pruebas 2 y 3 del comportamiento del sistema se realizaron cuatro pruebas en las que se entró al geovallado, se activó la búsqueda de WiFi y se dejaron los celulares quietos, se tomaron las primeras nueve capturas del reporte (que suman 240 segundos si hay captación precisa cada 30 segundos) esto de cada una de las cuatro pruebas y se graficó de la siguiente forma que muestran los resultados obtenidos que para obtener la duración mínima y el periodo de captación para los tres dispositivos probados, pruebas realizadas para antes de la activación de la permanencia del geovallado.

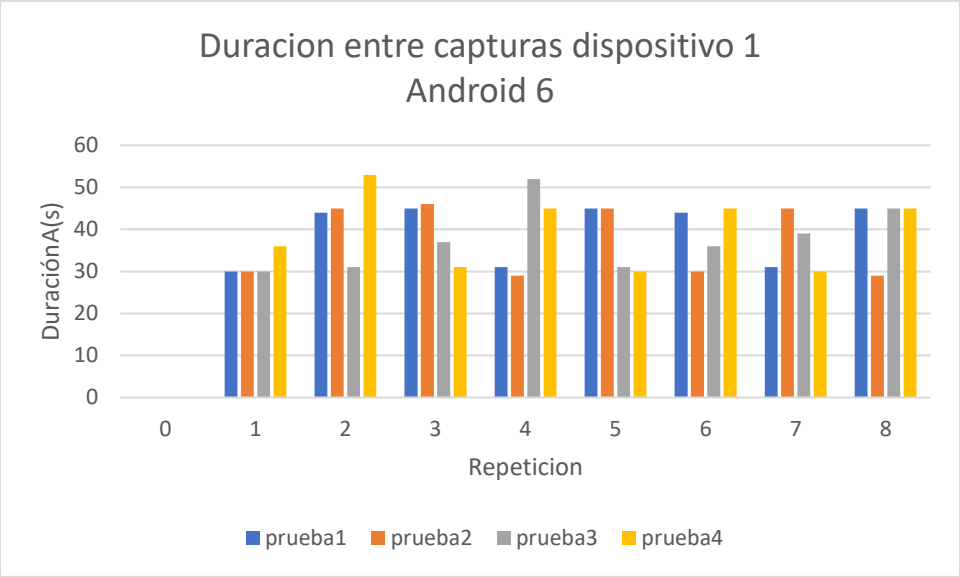


Imagen 38. Resultados pruebas Huawei MYA-L03

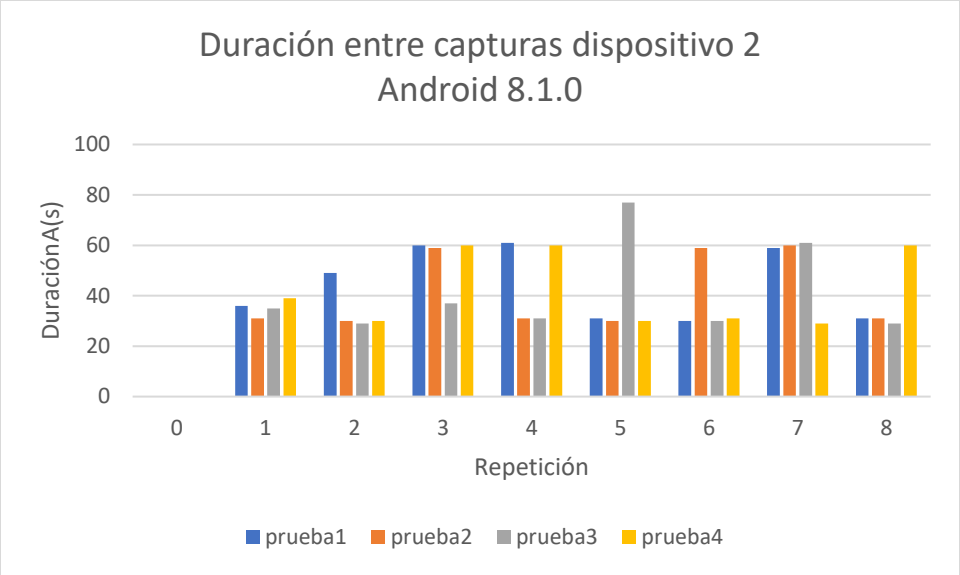


Imagen 39. Resultados pruebas Samsung J2

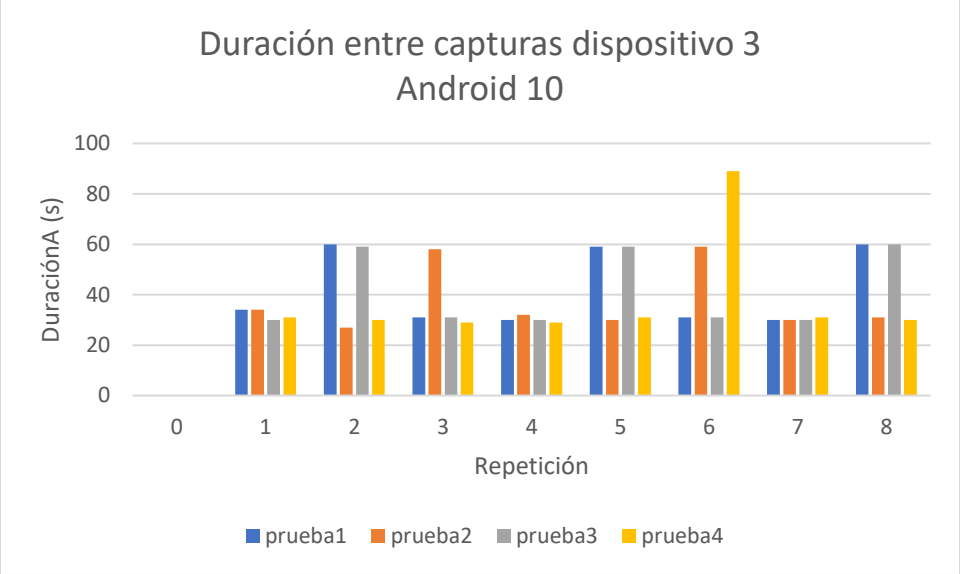


Imagen 40. Resultados pruebas Samsung A30

Los resultados de las capturas registradas en las imágenes 38, 39 y 40 se presentan son a continuación, con los resultados de la estancia mínima en cada prueba en la tabla 8, el promedio de las duraciones entre capturas de cada prueba en la tabla 9 y la suma de las primeras nueve capturas registradas en la tabla 10. En cada una de estas el promedio de las mediciones.

Dispositivo	prueba1	prueba2	prueba3	prueba4	Promedio
1 Android 6	30 s	30 s	30 s	36 s	31,5 s
2 Android 8.1.0	34 s	34 s	30 s	31 s	32,25 s
3 Android 10	36 s	31 s	35 s	39 s	35,25 s

Tabla 8. Tiempo mínimo de estancia.

Dispositivo	prueba1	prueba2	prueba3	prueba4	Promedio
1 Android 6	39,38 s	37,38 s	37,63 s	39,38 s	38,44 s
2 Android 8.1.0	39,67 s	36,78 s	36,56 s	33,33 s	36,58 s
3 Android 10	41,88 s	37,63 s	41,25 s	37,50 s	39,56 s

Tabla 9. Tiempo promedio de captación en cada prueba.

Dispositivo	prueba1	prueba2	prueba3	prueba4	Promedio
1 Android 6	315 s	299 s	301 s	315 s	307,5 s
2 Android 8.1.0	357 s	331 s	329 s	300 s	329,25 s
3 Android 10	335 s	301 s	330 s	300 s	316,5 s

Tabla 10. Tiempo de duración primeros 9 paquetes captados.

A continuación, se muestran los valores obtenidos del promedio junto al valor de desviación resultante con un tiempo esperado para una estancia mínima y periodo de captación de 30 segundos y para un tiempo de estancia de 240 segundos.

$$Desviación = |\frac{Tiempo\ real - Tiempo\ esperado}{Tiempo\ real}| * 100\%$$

Dispositivo	Estancia mínima		Periodo de captación		Tiempo de estancia	
1 Android 6	31,50 s	4,76%	38,44 s	21,96%	307,50 s	21,95%
2 Android 8.1.0	35,25 s	14,89%	37,67 s	20,36%	339,00 s	29,20%
3 Android 10	32,25 s	6,98%	39,56 s	24,17%	316,50 s	24,17%

Tabla 11. Resultados comportamiento del sistema piloto.

5.4 Resultado pruebas para determinar la trazabilidad a varios usuarios

1. **Estaciones cercanas;** los sistemas se colocaron a una distancia aproximada de 5 metros, se redujo la cobertura de la ESP32 de la estación 1. Se permaneció en la primera estación (primer sistema piloto) un tiempo de 5 minutos, hubo una estancia de 3 minutos en un punto intermedio entre las estaciones y en la segunda estación también un tiempo de 5 minutos. El trayecto realizado fue en línea recta de una estación a otra.

   Las primeras imágenes de cada dispositivo muestran la trazabilidad hecha con el reporte de Firebase y las segundas son resultado de los archivos de texto de cada estación que incluyen el valor del RSSI para obtener el transcurso de un lugar a otro de manera más detallada. Estas últimas el eje del tiempo está en segundos.

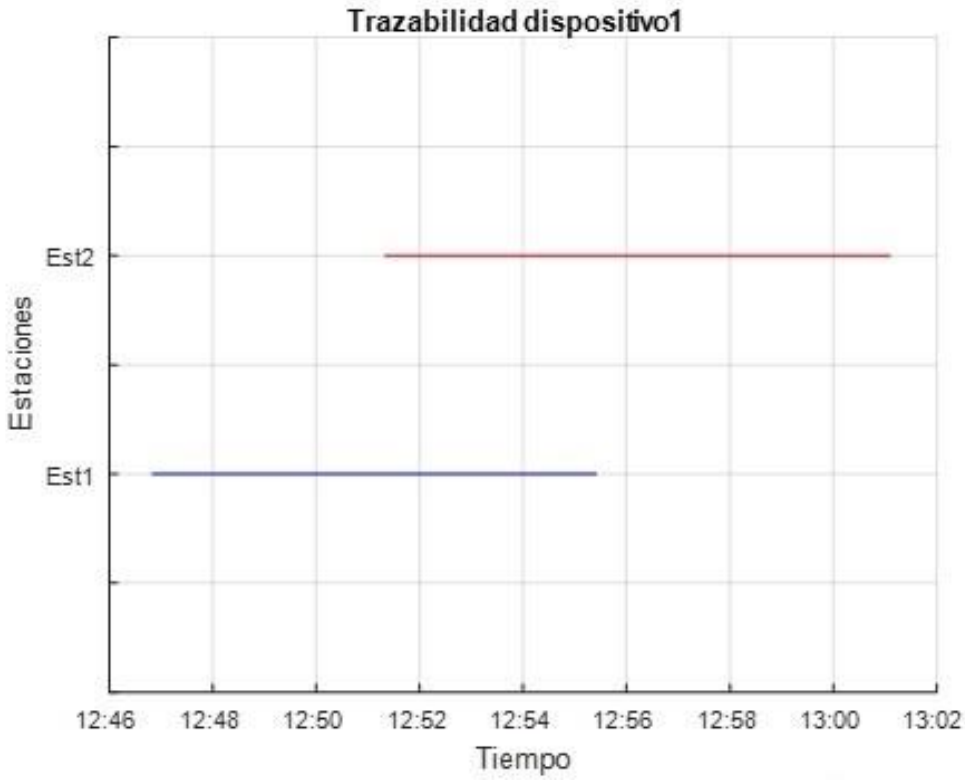


Imagen 41. Trazabilidad del dispositivo 1 con reporte de Firebase estaciones cercanas.

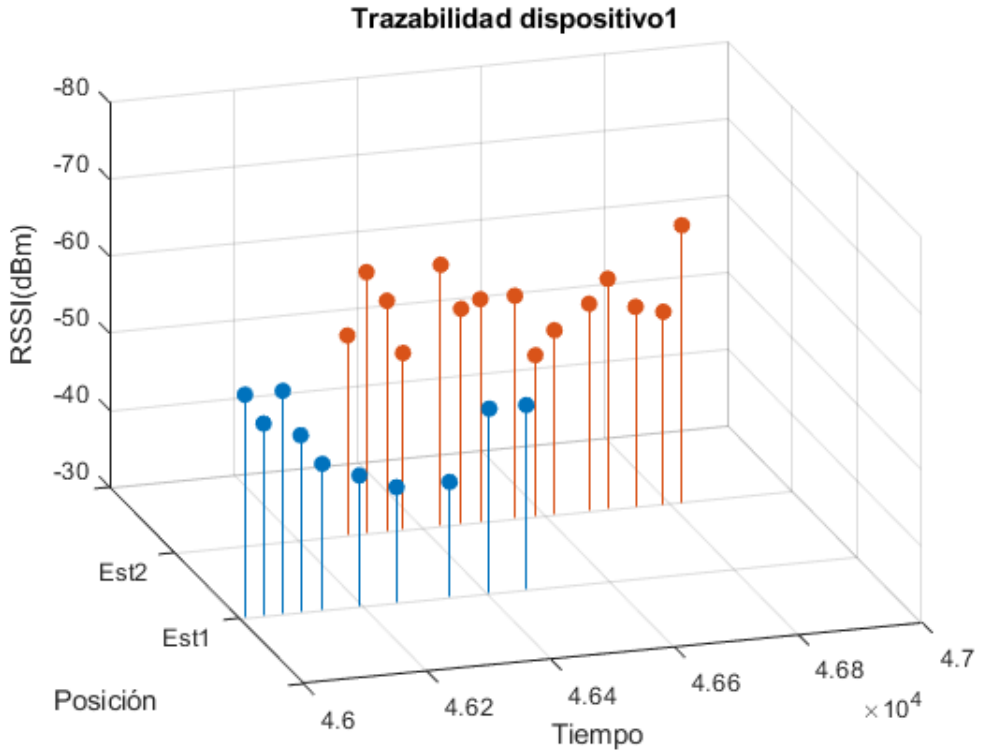


Imagen 42. Trazabilidad del dispositivo 1 con archivos de texto de cada estación estaciones cercanas, eje de tiempo en segundos.

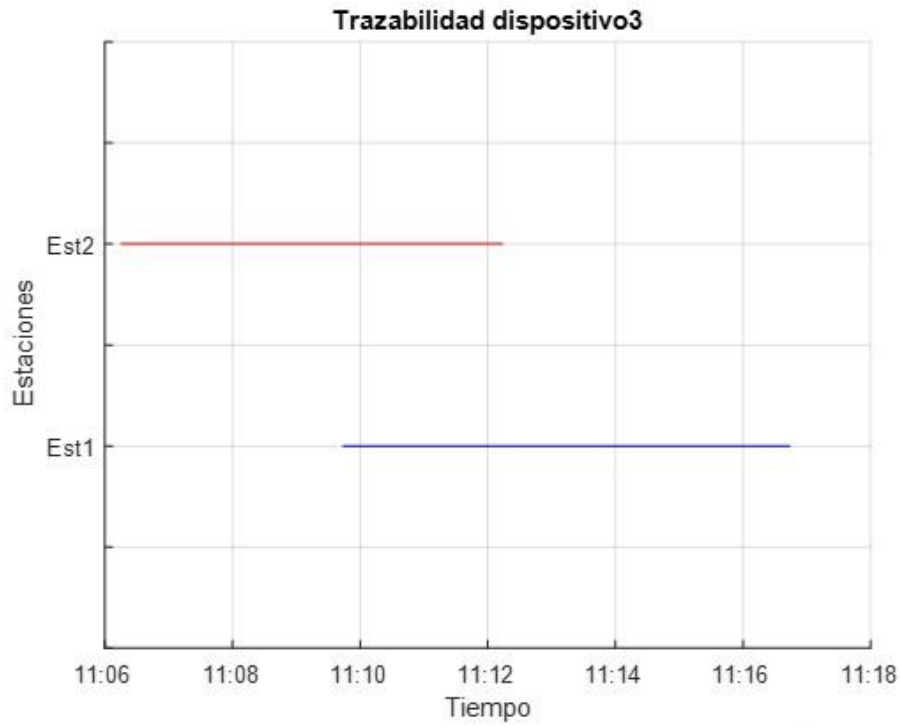


Imagen 43. Trazabilidad del dispositivo 3 con reporte de Firebase estaciones cercanas.

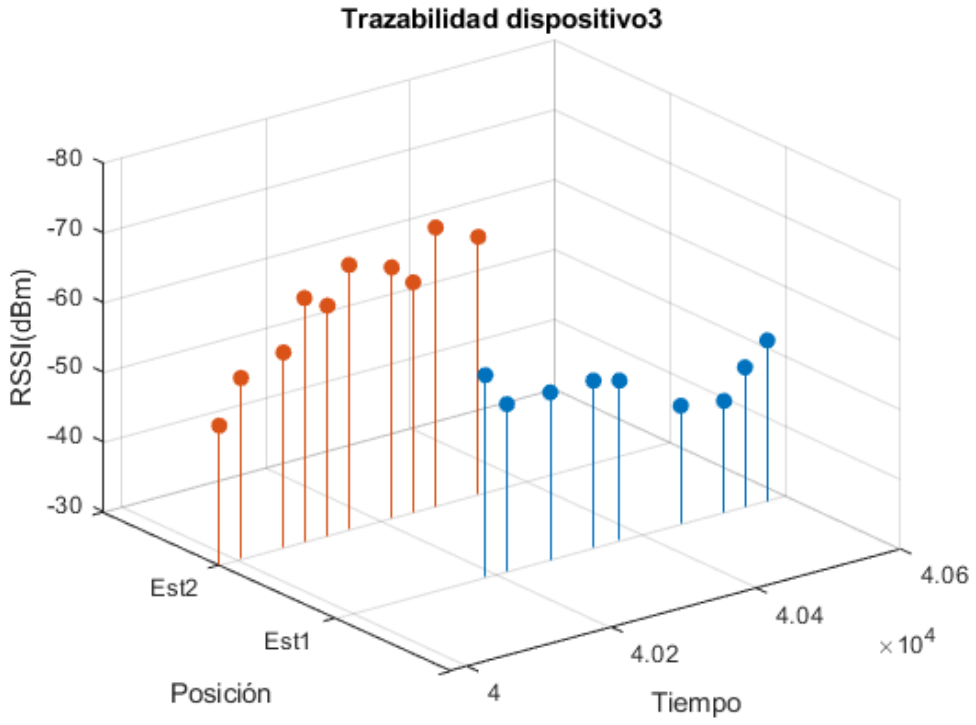


Imagen 44. Trazabilidad del dispositivo 3 con archivos de texto de cada estación estaciones cercanas, eje de tiempo en segundos.

2. **Estaciones con distancia media;** los prototipos se colocaron a una distancia aproximada de 30 metros entre estos, ambos con las ESP32 del valor de RSSI original, se permaneció un tiempo de cinco minutos en la primera, hubo un tiempo de trayecto y de permanencia en la segunda también de cinco minutos, esta prueba mostrará el contraste entre la trazabilidad real y la obtenida por el sistema. En la imagen 45 se muestran los tiempos captados en las estaciones, junto a las duraciones en cada una y del recorrido. (Reporte Firebase)

	1 Dir_MAC	2 Tiempos_Estacion1		3 Tiempos_Estacion2		4 Duracion_Estacion1	5 Duracion_Estacion2	6 Duracion_Recorrido
1	[redacted]	14:59:34	15:05:08	15:11:49	15:14:23	0h 5m 34s	0h 2m 34s	00:06:41
2	D3	14:59:43	15:05:22	15:11:06	15:15:36	0h 5m 39s	0h 4m 30s	00:05:44
3		14:59:50	15:02:18	15:10:56	15:23:24	0h 2m 28s	0h 12m 28s	00:08:38
4		15:00:16	15:03:41	15:00:16	15:03:41	0h 3m 25s	0h 3m 25s	00:03:25
5	D1	15:29:23	15:32:14	15:18:11	15:23:02	0h 2m 51s	0h 4m 51s	00:14:03

Imagen 45. Tabla del trayecto realizado por los dispositivos en prueba

La tabla 12 muestra los tiempos tomados de forma manual (con reloj) los tiempos de ingreso a salida del sistema, estimando 5 metros de distancia del sistema para tomar estos tiempos.

Dispositivo	Tiempo de entrada Estacion1	Tiempo de salida Estacion1	Tiempo de entrada Estacion2	Tiempo de salida Estacion2
1 Android 6	15:29:20	15:32:50	15:18:00	15:25:10
3 Android 10	15:00:00	15:05:40	15:10:20	15:15:45

Tabla 9. Tiempos reales de ingreso y salida de las estaciones pruebas.

Las siguientes imágenes desde la 46 hasta la 49 son las resultantes de los análisis hechos con ambas formas de almacenamiento para los dispositivos de la tabla 4, dispositivo 1 (imágenes 46 y 47) y dispositivo 3 (imágenes 48 y 49), las imágenes 46 y 48 corresponden al análisis de trazabilidad hecho con el almacenamiento de Firebase (tiempos iniciales y finales de cada dispositivo) que incluyen también el recorrido con los tiempos tomados de forma manual (trayectoria real). Las imágenes 47 y 49 muestran el análisis de trazabilidad hecho tomando los archivos de texto almacenados en cada una de las estaciones (sistemas pilotos), y juntándolo para obtener el trayecto en cada una de las estaciones, este incluye todos los mensajes captados, con diferencia de tiempo junto al valor de RSSI de cada uno, el eje tiempo de estos está en minutos.

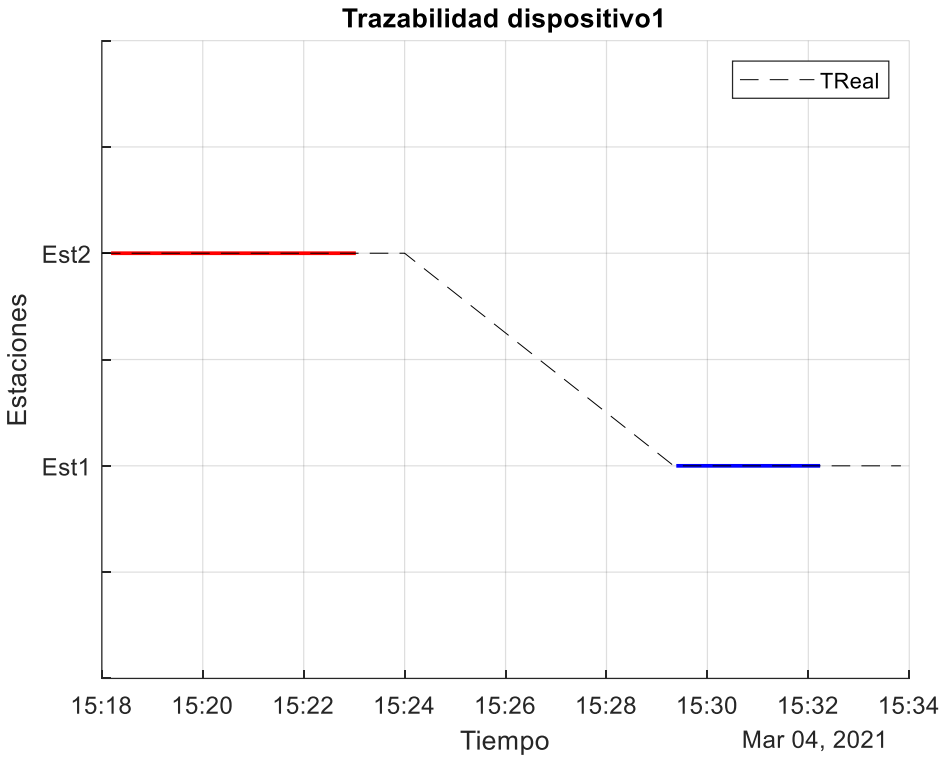


Imagen 46. Trazabilidad del dispositivo 1 con reporte de Firebase junto a trayectoria real estaciones distancia media.

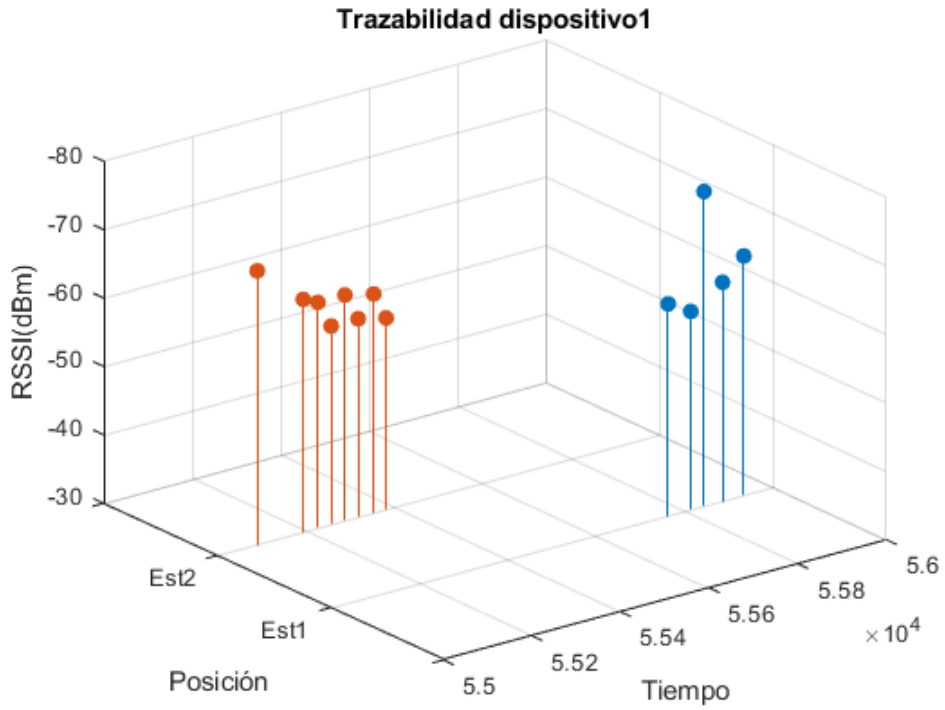


Imagen 47. Trazabilidad del dispositivo 1 con archivos de texto de cada estación, estaciones distancia media, eje de tiempo en segundos.

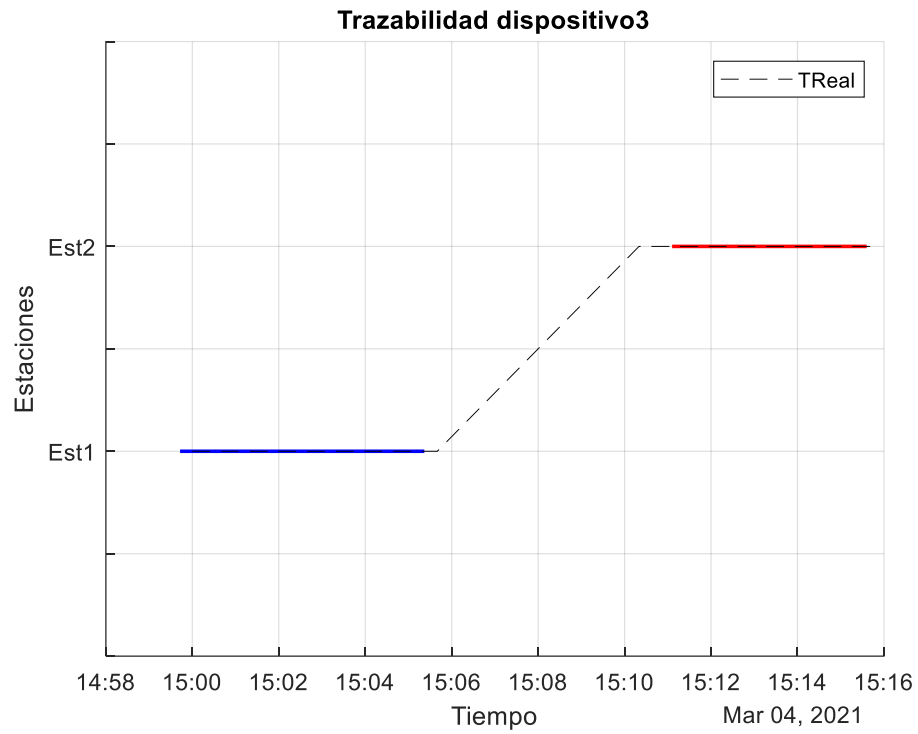


Imagen 48. Trazabilidad del dispositivo 3 con reporte de Firebase junto a trayectoria real.

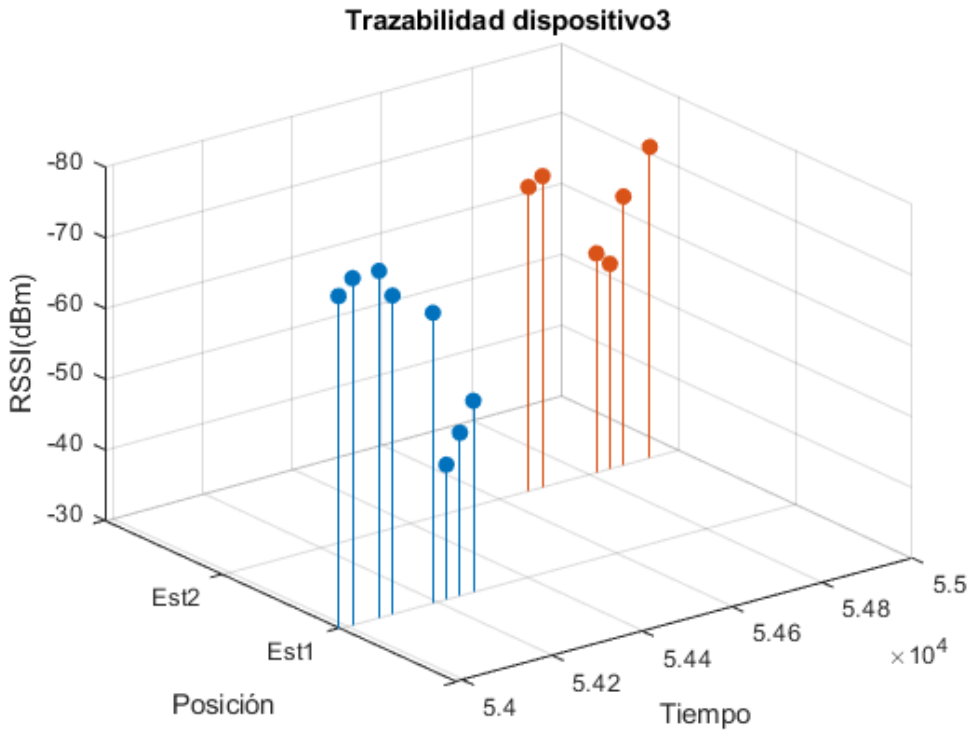


Imagen 49. Trazabilidad del dispositivo 3 con archivos de texto de cada estación, estaciones distancia media, eje de tiempo en segundos.

6. Análisis de resultados

Los resultados obtenidos en este trabajo de grado, mostrados en el capítulo anterior se pueden interpretar de la siguiente forma.

- ✓ Las activaciones del geovallado (entrada, permanencia y salida) tienen un retardo irregular, estas varían dependiendo del equipo probado, identificando un mayor retardo para los dispositivos con sistema operativo superior, debido a mayores restricciones impuestas a la localización en las versiones más actuales de Android. La activación de entrada es la de mayor importancia, esta es inmediata cuando el dispositivo ya está dentro del geovallado, al ir de afuera hacia adentro hubo un retardo promedio de un minuto, este valor aumenta si el equipo se encuentra en modo descanso.

- ✓ La aplicación móvil realiza las búsquedas de WiFi de forma continua cada 30 segundos según su programación, con posibles solicitudes fallidas menores sin mayor importancia. Debido a esta continuidad, la aplicación presenta un alto consumo de batería, su uso prolongado genera alertas expuestas por los mismos dispositivos.

- ✓ Las pruebas de estancia mínima de permanencia y de periodo de repetición presentan resultados favorables que concluyen que se determina una estancia mínima con un máximo de 36 segundos, que se realiza un seguimiento constante y prolongado de los dispositivos, presentando un sistema robusto que capta la mayor cantidad de los mensajes con un periodo de captación con una desviación menor a 25% y una permanencia de estancia completa con un valor menor de 30% de desviación, valores superiores presentados en cada uno para el dispositivo del peor caso.

- ✓ Las pruebas de trazabilidad muestran que el sistema está capacitado para captar a los usuarios en cada estación, y mediante el análisis realizado es posible conocer el trayecto detallado de los dispositivos, con duraciones en cada estación y del trayecto. Comparando con la trayectoria real, si existen imprecisiones que se deben al periodo de repetición y a que no se captan la totalidad de los mensajes. Algo que afecta la exactitud del sistema pero que no representa una dificultad seria respecto a identificar una trazabilidad, contar personas y hacer un estimado de la congestión.

- ✓ Las medidas del valor de RSSI permite hacer una estimación de la distancia a la que estuvo el usuario en cada mensaje captado, con un resultado esperado y acorde a que a una mayor distancia más baja es esta medida, aunque si se presentan variaciones aun cuando el dispositivo esta estático. Este valor de RSSI que es asignado por la ESP32 como modo promiscuo y en la programación de la Raspberry Pi se obtiene este valor únicamente del primer mensaje de cada ráfaga, algo que podría ser cambiado para obtener un valor más preciso, que podría ser el promedio de los valores de RSSI de los mensajes de la ráfaga.

- ✓ El funcionamiento del sistema piloto en conjunto cumple con el objetivo propuesto de brindar un apoyo para la recolección de la información útil requerida por un sistema de transporte masivo de pasajeros para ayudar en su planeación, información además de la trazabilidad de los usuarios, como el conteo de personas y el tiempo promedio de estancia por intervalo para estimar congestión. Adicionalmente, comparando con los métodos actuales que usa Transmilenio, este sistema piloto ofrece una forma de recolección de información más frecuente, ágil y automática, debido a esto se adapta con mayor rapidez y se puede analizar en distintos periodos ya sea corto o a largo plazo.

- ✓ La participación que tiene el usuario es importante ya que es el encargado de usar la aplicación (Geovallado-Búsqueda de WiFi) en su dispositivo móvil, sin la activación de la búsqueda repetitiva de WiFi, la cantidad de mensajes que son captados es casi nula.

7. Conclusiones y recomendaciones

El sistema piloto presentado, producto de la integración de todas las partes documentadas en este documento permite obtener información útil que beneficiaría a un sistema de transporte masivo en las etapas de la planificación (a largo plazo) y una operación (a corto plazo), información adicional y enfocada en el comportamiento de los usuarios, que no tiene en este momento Transmilenio como la trazabilidad detallada de los usuarios en el sistema, el conteo de personas y el tiempo de permanencia para determinar una posible congestión en el sistema. Contrario a los métodos actuales de recolección de la encuesta de movilidad y el reporte bimestral de Transmilenio la información entregada por este sistema piloto se obtiene de una manera frecuente, ágil y automática, en la que es esencial la participación del usuario con la aplicación móvil.

Este sistema piloto fue desarrollado y documentado de tal forma que la replicación por otras personas que estén interesadas sea clara y sencilla de hacer, está abierto a modificaciones que logren realizar mejoras y adaptaciones para su uso en otros ambientes. Todas las partes del sistema pueden ser mejoradas, como recomendación propongo enfocarse en el desarrollo de la aplicación móvil en la que se puede variar el periodo de solicitudes de búsqueda de WiFi activas, logrando optimizar el consumo de batería que se mantiene alto. También, en la programación de la ESP32 como captador de paquetes *Probe Request*, se puede considerar el tiempo de permanencia en cada canal y la cantidad de estos cuando esta hace el barrido en el modo promiscuo, que podría lograr una mejor captación de los paquetes en el ambiente, también, considerar que la ESP32 capta otros dispositivos como computadores, tabletas y dispositivos que puedan hacer conexión inalámbrica. Un mayor desarrollo de la sección de análisis podría presentar más información de utilidad para el sistema, como la recopilación y análisis de varios días para muchas más estaciones podrían ser aplicado y sería más completo.

En cuanto al uso del módulo TAR SIM800 RPI es de gran ayuda por su cobertura cuando el sistema se encuentra en lugares remotos, su uso depende de la red móvil del operador de la tarjeta SIM que no presentó fallos en las pruebas. Sin embargo, si hay posibilidad de conectar la Raspberry a internet por medio de WiFi o mejor por cable ethernet, beneficiaria al sistema en términos de consumo de energético, costos del plan de datos y mayor estabilidad en el envío de datos a internet.

Las pruebas realizadas permitieron conocer el desempeño del sistema en un salón del edificio de laboratorios de ingeniería, en condiciones atípicas a las de una estación real del Transmilenio, para siguientes pruebas el sistema podría ser ubicado en lugares con mayor flujo de personas, que permitiría obtener resultados más acordes al lugar que se piensa instalar.

Algunos hallazgos encontrados fueron, adicionales a los expuestos en análisis de resultados;

- La precisión del geovallado en la aplicación móvil mejora cuando el dispositivo tiene una conexión a WiFi, al tener esta conexión se identificó que los mensajes de la búsqueda de WiFi se realizan únicamente en el canal del punto de acceso y adyacentes. También, para mejorar la precisión del geovallado se puede hacer un servicio en primer plano como el hecho para la búsqueda repetitiva de WiFi. Sin embargo, esto incrementa el uso de batería.

- Filtrar la cobertura de la ESP32 mejora el comportamiento y la captación de los mensajes. La recolección de la medida de RSSI podría ser más útil si se considera obtener un promedio de los mensajes captados para la ráfaga u otro factor, ya que es tomado únicamente el del primer mensaje de la ráfaga.

8. Bibliografía

[1] Alcaldía de Bogotá, “Encuesta de Movilidad 2019 | Bogota.gov.co,” 2019, [Online]. Available: http://www.bogota.gov.co/temas-de-ciudad/movilidad/encuesta-de-movilidad-2019.

[2] Transmilenio, “Estadísticas de oferta y demanda del Sistema Integrado de Transporte Público.” https://www.transmilenio.gov.co/publicaciones/149180/estadisticas-de-oferta-y-demanda-del-sistema-integrado-de-transporte-publico-sitp/.

[3] Portafolio, “Así se ha afectado Transmilenio por la pandemia del Covid-19.” https://www.portafolio.co/economia/transmilenio-perdidas-millonarias-por-covid-19-544002.

[4] B. como Vamos, “Cómo vamos en Movilidad,” [Online]. Available: https://assets.documentcloud.org/documents/6344845/Informe-de-Calidad-De-Vida-Movilidad.pdf.

[5] Wikipedia, “Geovallado.” [Online]. Available: https://es.wikipedia.org/wiki/Geovalla#:~:text=Una%2520geovalla%2520o%2520geocerca%2520es,la%2520situaci%25 C3%25B3n%2520de%2520un%2520punto.

[6] M. Gast, *802.11® Wireless Networks: The Definitive Guide, Second Edition*, no. April. 2005.

[7] Android, “MAC randomization.” https://source.android.com/devices/tech/connect/WiFi-mac-randomization.

[8] Rincónperdición, “MAC aleatoria en Android 10.” https://www.rinconperdicion.com/mac-aleatoria-en-android-10/.

[9] Kotlin, “Kotlin.” https://developer.android.com/kotlin/first.

[10] Android, “Android Studio.” https://developer.android.com/studio?hl=es-419&gclid=CjwKCAiA9bmABhBbEiwASb35V2Oq4nn3xFFfy4HHACxdlVDWtzWL7m675hy1kG1bwYEzqIDnJ7P0rRoCsXIQAvD_BwE&gclsrc=aw.ds.

[11] Android, “API geovallado.” https://developers.google.cn/location-context/geofencing/?hl=es-419.

[12] Android, “Geovallado.” https://developer.android.com/training/location/geofencing?hl=es.

[13] Android, “Limitaciones ubicación segundo plano.” https://developer.android.com/about/versions/oreo/background-location-limits?hl=es.

[14] Android, “WiFi Manager.” https://developer.android.com/reference/android/net/WiFi/WiFiManager.

[15] Android, “WiFi Scan.” https://developer.android.com/guide/topics/connectivity/WiFi-scan?hl=es-419.

[16] Android, “Resultados WiFi Scan.” https://developer.android.com/reference/android/net/WiFi/ScanResult?hl=es-419.

[17] Android, “Doze mode.” https://developer.android.com/training/monitoring-device-state/doze-standby?hl=es-419.

[18] Android, “Foreground service.” https://developer.android.com/guide/components/services?hl=es-419#Foreground.

[19] Android, “Permissions.” https://developer.android.com/guide/topics/permissions/overview?hl=es#normal-dangerous.

[20] Android, “Permisos manifest.pdf.” https://developer.android.com/reference/android/Manifest.permission?hl=es.

[21] Android, “Solicita permisos de ubicación.” https://developer.android.com/training/location/permissions?hl=es-419.

[22] Espressif, “ESP32.” .

[23] Espressif, “ESPIDF.” https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/index.html.

[24] Eclipse, “https://www.eclipse.org/.” .

[25] Espressif Systems, “WiFi sniffer mode ESP32.” https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-guides/WiFi.html#wi-fi-sniffer-mode.

[26] Espressif Systems, “Datasheet ESP32 Series,” *Espr. Syst.*, pp. 1–61, 2019, [Online]. Available: www.espressif.com.

[27] M. M. Barón Juan, “Dispositivo anti-perdidas de objetos personales.,” *Am. J. Orthod. Dentofac. Orthop.*, vol. 20, no. 1, pp. 1–8, 2016.

[28] Github, “pyserial.” [Online]. Available: https://github.com/pyserial/pyserial.

[29] Github, “pyrebase.” [Online]. Available: https://github.com/thisbejim/Pyrebase.

[30] Kolwidi, “COMO EJECUTAR UN SCRIPT PYTHON AL ARRANCAR TU RASPBERRY PI.” [Online]. Available: https://kolwidi.com/blogs/blog-kolwidi/como-ejecutar-un-script-python-al-arrancar-tu-raspberry-pi.

[31] Dinero, “Internet en estaciones de Transmilenio.” https://www.semana.com/tecnologia/articulo/internet-en-transmilenio-como-sera-la-conexion-de-la-etb/280210/.

[32] Conexión Capital, “Nuevos buse con WiFi del SITP.” https://conexioncapital.co/asi-lucen-los-nuevos-buses-del-sitp-que-tienen-WiFi/.

[33] Sigma Electrónica, “TAR SIM800 RPI Sigma.pdf.” https://www.sigmaelectronica.net/producto/tar-sim800-rpi/.

[34] rhydolabz, “Raspberry Pi : How to access the Internet using GSM / GPRS Modem (SIM900/SIM800).” https://www.rhydolabz.com/wiki/?p=16325.

[35] Firebase, “Estructura base de datos.” https://firebase.google.com/docs/database/web/structure-data?hl=es.

9. Anexos

Anexo 1. Geovallado-Búsqueda de WiFi, https://github.com/JuanDSanchez18/Geovallado-BusquedaWiFi

Anexo 1.1 Ubicación, https://github.com/JuanDSanchez18/Location-Android-Studio

Anexo 1.2 Geovallado, https://github.com/JuanDSanchez18/STUTM-Geofence

Anexo 1.3 Búsqueda de WiFi Activa, https://github.com/JuanDSanchez18/STUTM-ScannerWiFi

Anexo 2. Captador de paquetes *Probe Request*, https://github.com/JuanDSanchez18/STUTM-ESP32

Ver ramas, main sin filtros, ESP32 filtrado

Anexo 3. Tratamiento, https://github.com/JuanDSanchez18/STUTM-Tratamiento

Anexo 4. Análisis Matlab-Firebase, https://github.com/JuanDSanchez18/STUTM_analisis

Trabajos de ayuda

Aplicación móvil

- Como agregar geovallados tutorial

  https://developer.android.com/codelabs/advanced-android-kotlin-training-geofencing#0

- Servicio en Android que nunca para

  https://robertohuertas.com/2019/06/29/android_foreground_services/

Esp32 modo promiscuo

- Programa hecho en Arduino como modo promiscuo, junto archivo Python para realizar archivos .pcap desde las tarjetas ESP para el uso del programa Wireshark para analizar. Hecho por Spacehuhn.

  https://github.com/spacehuhn/ArduinoPcap