



Universidad de Buenos Aires
Facultad de Ingenieria
Taller de Programacion 1

Trabajo Practico

Proyecto: Agentes Autónomos de Prevención

Grupo RustX

Profesores

Agustin Firmapaz

Martin Miletta

Camila Ayala

Integrantes

Agama Avila, Arely

Lescano Maier, Aldana

Morilla, Martin

Smith, Claudio Anibal

Indice

1. Explicación general de la investigación realizada	1
2. Reglas de negocio de la solución	2
3. Decisiones tomadas durante el desarrollo del proyecto	3
4. Funcionamiento del Cliente y Servidor	5
5. Diagramas de Secuencia	7
6. Diagrama de Componentes	11
7. Interfaz Grafica	12

1. Explicación general de la investigación realizada

Decidimos usar el protocolo de mensajería MQTT, dado que es un protocolo ligero y eficiente para dispositivos con recursos limitados. También es ideal para IoT (internet de las cosas). Maneja 3 niveles de calidad de servicio: QoS 0, 1 y 2 y autenticación. Metodología Publish/Subscriber. Además MQTT se encarga de minimizar la latencia para una comunicación rápida y eficiente en tiempo real. AMQP presentaba otras características que no nos eran útiles para la solución. Como transacciones, filtros, etc. Ademas de esto, tuvimos que leer la documentacion sobre la cantidad de bytes a asignar para cada mensaje, para el fixed header, variable header y payload. Tambien tuvimos que investigar sobre la UI, investigamos sobre la librería gráfica GTK, la cual comenzamos con GTK 3 con un mapa png estático y finalmente decidimos cambiar por EGUI, ya que encontramos una librería en la cual podíamos utilizar Open Street Map , posicionar las cámaras, drones y desplazarlos por todo el mapa. También se investigó el uso de librerías de log, formularios y conexiones TCP. Además también se investigó el uso de channels tanto para la comunicación con el cliente del broker como para la interfaz gráfica y el logger.

2. Reglas de negocio de la solución

Se decidió levantar los drones en forma de grilla, en este caso con un maximo de 3 drones por fila.

La encriptación solamente está a nivel de payload de los mensajes.

Los clientes del broker deben autenticarse con el archivo de credenciales.

Los incidentes se crean ingresando su posicion geográfica a traves de la interfaz.

El sistema de cámaras provee un archivo de properties donde también los valores de latitud y longitud deben respetar los valores estándar de mapas

Un incidente se resuelve cuando al menos dos agentes (drones) llegan al lugar del incidente y transcurre un tiempo definido por configuración en esa posición.

Los agentes que reciben la notificación de un incidente deben movilizarse hacia ese incidente solo si: a. El incidente se produjo dentro del area máxima de alcance del agente (dron) b. No se reciben notificaciones de dos agentes mas cercanos que ya esten atendiendo el incidente. c. El nivel de bateria del dron se encuentra sobre los valores mínimos de operacion (definido en el archivo de configuración).

Luego de atender un incidente el agente debe volver a su area de operación asignada.

El nivel de bateria de cada agente se deberá ir descargando con el paso del tiempo (se simulará su descarga).

Las cámaras inician su funcionamiento en modo ahorro de energia y deben pasar a estado de activo cuando se recibe una notificación de un incidente en su area de alcance

Encriptacion

Para la encriptación utilizamos el algoritmo 3DES encriptando la información del payload de los mensajes.

3. Decisiones tomadas durante el desarrollo del proyecto

Una de las decisiones mas importantes para este proyecto, fue decidir que nivel de servicio implementar, optamos por el QoS 1 dado lo siguiente.

Nivel de Servicio

Usamos el nivel de calidad de QoS 1 el cual garantiza que los datos deben ser recibidos al menos una vez, lo cual indica que puede haber una duplicación aceptable. El proceso sería el siguiente:

1. El cliente envía un Publish al Broker
2. El Broker recibe el mensaje Publish y envía el PubAck al cliente
3. El Broker envía el mensaje a los suscriptores interesados.

También la elección de este nivel de servicio es dado que si optamos por QoS 2, esta es menos eficiente en latencia y consumo de recursos, dado que tiene mayor complejidad respecto a manejo de mensajes:

Grafico QoS 1



Figure 1: QosS1

Grafico QoS 2

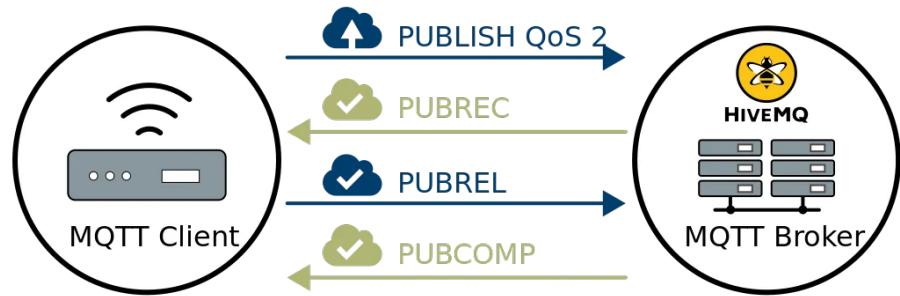


Figure 2: QosS2

4. Funcionamiento del Cliente y Servidor

Server

El server recibe por parámetro el puerto en el que iniciarse. Por cada cliente que inicia una conexión TCP con él, lanza un hilo para manejar la comunicación con dicho cliente. Al recibir un mensaje MQTT de tipo connect, autentica al cliente, validando que el mensaje contenga en sus campos (conocidos, según la especificación del protocolo) un user id y contraseña esperados por el server. En caso de que sean correctos, se envía un ack confirmando el éxito de la conexión y se procede a continuar con la conexión. Luego existe un match en el que se determina el tipo del mensaje, y se actúa en forma acorde, según el comportamiento esperado por mqtt, en cada caso. Además, ese primer mensaje connect enviado por la aplicación del cliente que utiliza el MQTTClient, contiene un client id único para cada cliente, que el server utilizará para identificarlo durante la sesión. En caso de que los valores provistos no sean los correctos, el server envía un ack informando del error, y no se continúa con la conexión.

Cliente

Para el cliente, el procesamiento es análogo en cuando al match por tipo de mensaje, con la salvedad de que no van a llegarle mensajes de tipo connect ya que es él quien inicia la conexión, y recibirá en cambio ack. Tampoco recibirá mensajes de tipo subscribe. En este caso, maneja un único stream, ya que posee una única conexión establecida que es con el server. La dirección ip y el puerto son valores conocidos, se recibe una SocketAddr en la función de la librería que efectúa la conexión.

Dron

Esta aplicación se suscribe al topic para recibir Incidentes. Por cada uno, debe

determinar si debe o no desplazarse en base a una serie de condiciones. Algunas de ellas se tratan de verificar valores que tiene como propiedades y dependen exclusivamente de él. Otra involucra comunicación con otros drones. Por este motivo, el dron también se ha suscripto también al topic para recibir información de drones. Todos los mensajes se reciben a través del cliente MQTT, se extrae el topic del mensaje tipo publish recibido, y se pasa a procesarlo de forma acorde según el mismo.

Los incidentes recibidos fueron publicados por el sistema de monitoreo. Centrámonos en la condición que involucra comunicación con otros drones. El dron tiene un hashmap, privado, que utiliza para evaluar esta condición. Al recibir un incidente, un hilo se encarga de manejarlo de la siguiente manera. Luego de evaluar las condiciones que dependen de él mismo, se agrega el id del incidente recibido a un atributo que representa el id del incidente al que se está evaluando desplazarse; además, se agrega dicho id al hashmap, con la posición de dicho incidente y un vector vacío como valor. Por otra parte, cuando se recibe un dron current info, se verifica que sea de otro dron (y no de él mismo) y en ese caso se procede a manejarlo de la siguiente forma. Si lo informado por otro dron incluye que su id de incidente al que considera desplazarse coincide con el atributo propio, entonces se trata del mismo incidente en cuestión, y cada dron deberá decidir si se desplaza o si por el contrario no se desplaza y deja que lo haga el otro dron. Para esto, teniendo el id: busca la posición del incidente en el hashmap; calcula la distancia de cada dron involucrado (él mismo, y el que le envió la notificación) hasta el incidente, y guarda como valor en el hashmap el id del dron que resultó con menor distancia.

Esto funciona debido a que el hashmap en cuestión está envuelta en un `arc_mutex`, por lo que puede accederse por diferentes hilos. De esta forma, entonces, continuando la explicación de lcaso en el que se recibió incidente, como último paso se consulta el hashmap. Siempre para el valor del hashmap con el id del incidente en cuestión, se ordena los valores del vector de "(dron id, distancia al inc)" para obtener los dos dron id de menor distancia. Se consulta si el id propio del dron se encuentra contenido en esa lista obtenida, y en caso afirmativo el dron concluye que debe desplazarse.

De esta forma se maneja esa condición, antes de iniciar a desplazarse.

5. Diagramas de Secuencia

Sistema Camaras

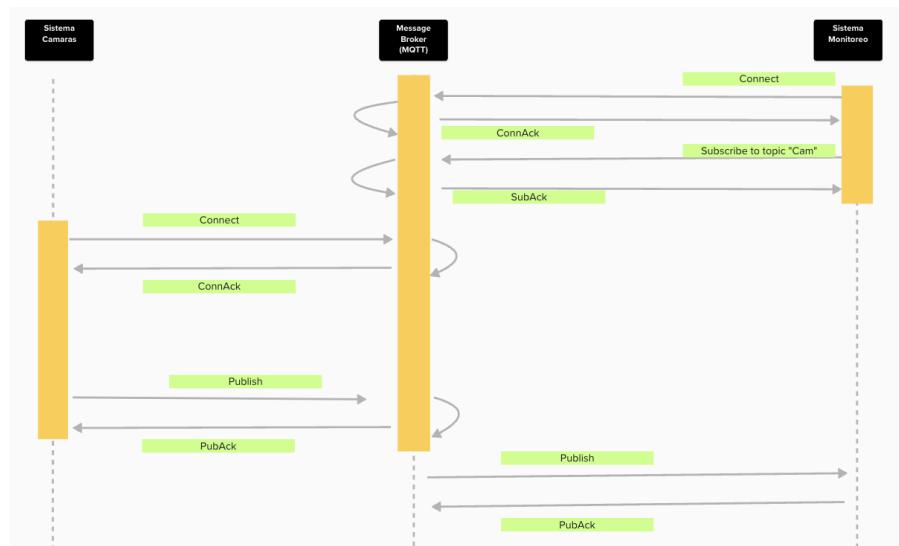


Figure 3: Grafico ABM Camaras

Incidentes

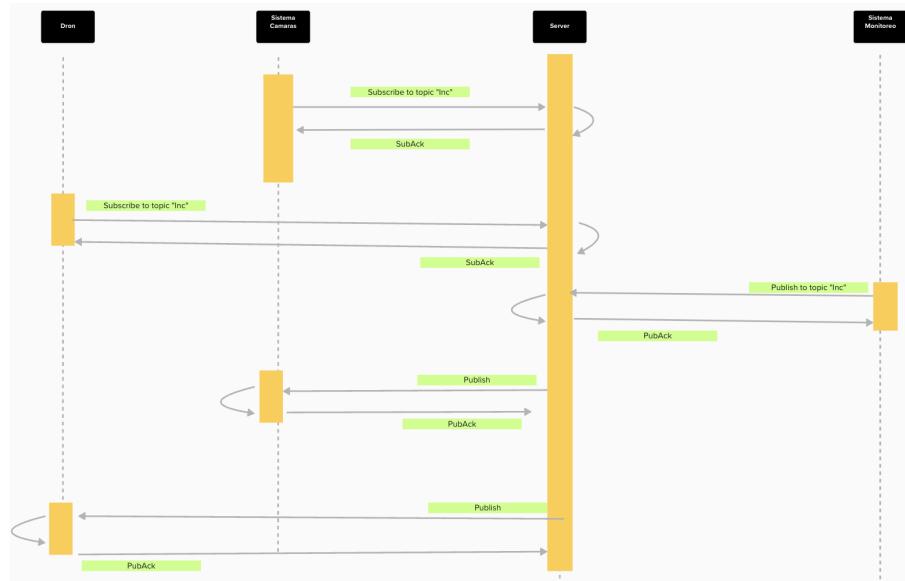


Figure 4: Grafico

Dron

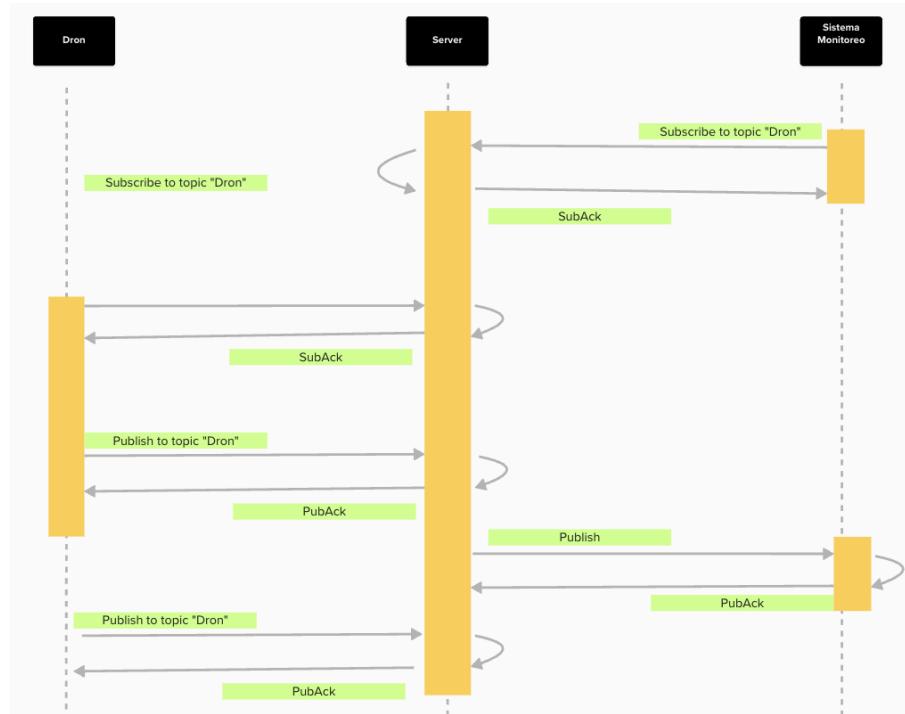


Figure 5: Grafico

Camara - Cambios de Estado

Lo que nos parecio relevante fue el cambio de estados de camara, en este caso Sistema de monitoreo publica un incidente el cual se da de alta por la interfaz grafica, este mismo es enviado a los que se subscribieron a ese topic, en este caso Sistema de Camaras. Al recibir un incidente por primera vez, este mismo esta identificado con un id. Este id se va a guardar en un hashmap tal que tengamos al id del incidente como clave y como valor las camaras que lo tienen en su rango pasando estas ultimas a estado: "ActiveMode". Luego si recibimos un incidente con el mismo id, el procedimiento sera distinto. Tenemos una estructura de camara tal que uno de esos campos es un vector de incidentes, lo que haremos sera eliminar ese incidente de su vector de incidentes. Esto se realiza dado que una Camara podria seguir en modo activo, dado que puede haber otro incidente que tambien este en su area, por lo que dicha camara no deberia cambiar a estado: "SavingMode". Entonces, la cámara internamente luego de eliminar el incidente de su lista, si la misma està vacía, vuelve su estado a SavingMode. Y actualizamos tambièn nuestro hashmap de manera acorde.

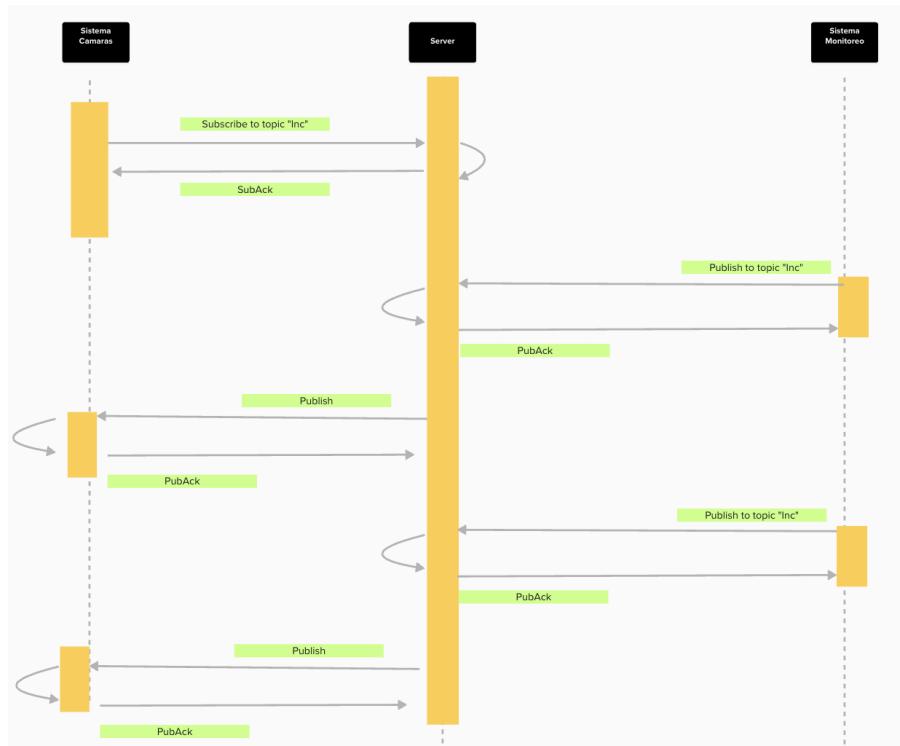


Figure 6: Cambios de Estado

6. Diagrama de Componentes

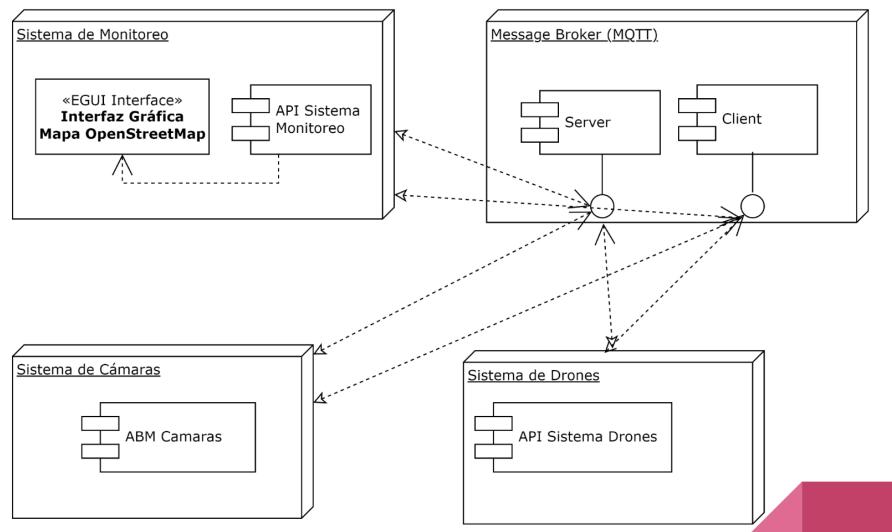


Figure 7: Diagrama de Componentes

7. Interfaz Gráfica

Vemos la UI inicial



Figure 8: UI antes de dar de alta incidente

Al dar de alta un incidente, observamos que algunas cámaras se ponen de color verde, indicando que están dentro del área del incidente. Tambien las lindantes a esas camaras se muestran de ese color.

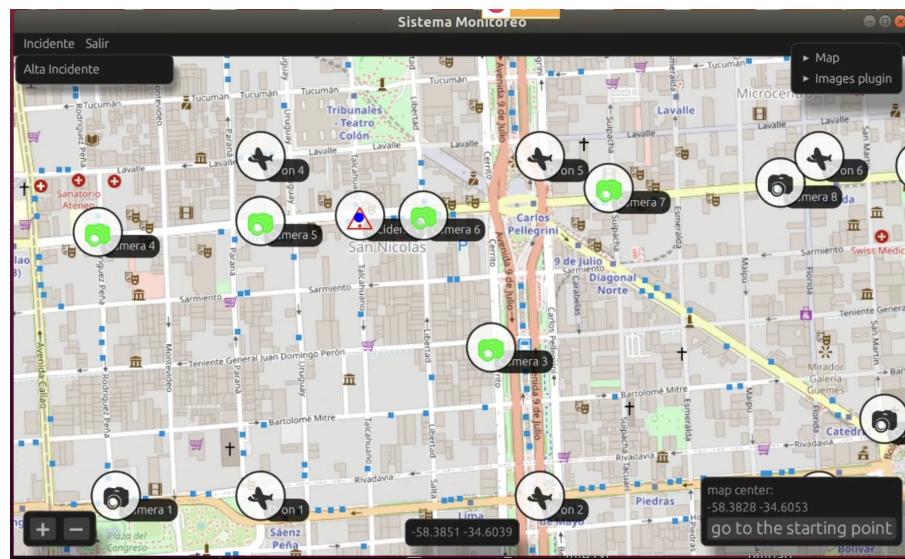


Figure 9: Camaras activas

Vemos a los 2 drones mas cercanos volando hacia el incidente

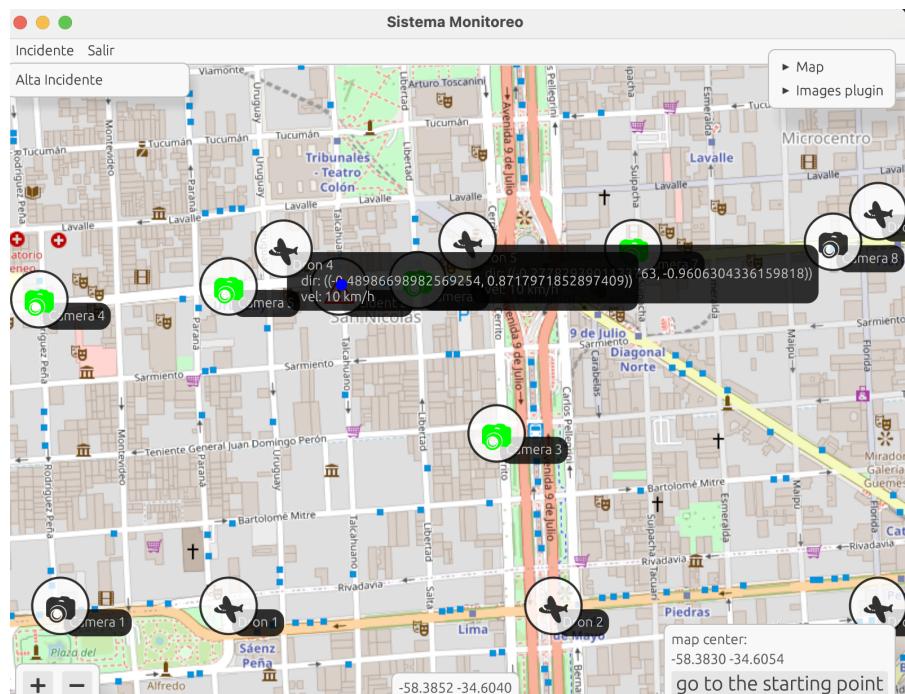


Figure 10: Drones: 4 y 5 volando hacia el incidente

Vemos que los 2 drones en el incidente

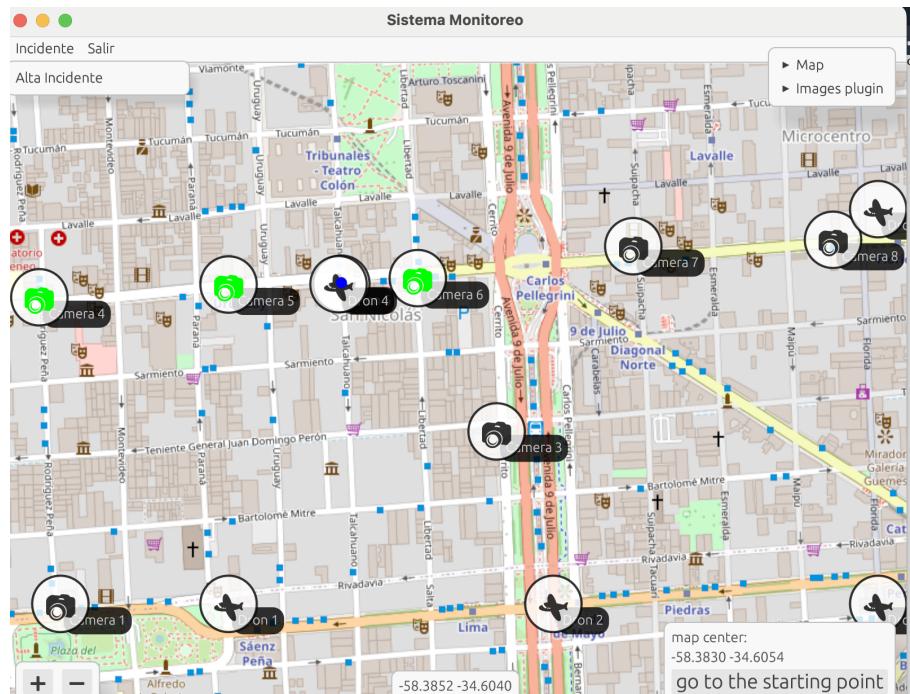


Figure 11: Drones: 4 y 5 en el incidente

Despues vuelven a sus posiciones iniciales

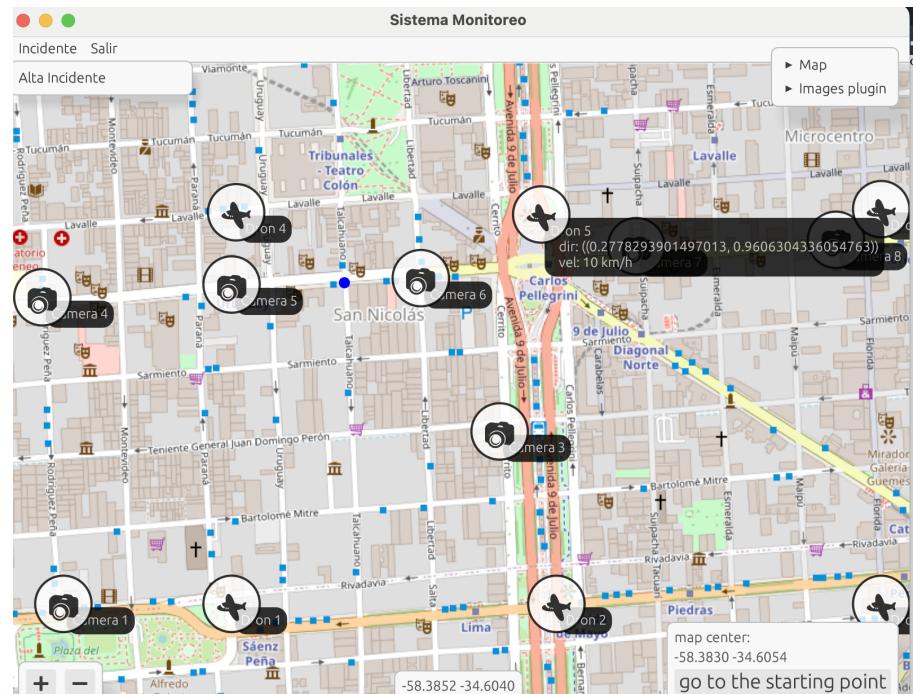


Figure 12: Drones: 4 y 5 vuelven a su posicion